

---

NIST Special Publication 800-38C

**NIST**

**National Institute of  
Standards and Technology**

Technology Administration  
U.S. Department of Commerce

**Recommendation for Block  
Cipher Modes of Operation:  
The CCM Mode for  
Authentication and  
Confidentiality**

Morris Dworkin

---

C O M P U T E R   S E C U R I T Y

---



NIST Special Publication 800-38C

# **Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality**

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 2004



**U.S. Department of Commerce**

*Donald L. Evans, Secretary*

**Technology Administration**

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

**National Institute of Standards and Technology**

*Arden L. Bement, Jr., 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 800-38C**  
**Natl. Inst. Stand. Technol. Spec. Publ. 800-38C 25 pages (May 2004)**  
**CODEN: NSPUE2**

## Abstract

This Recommendation defines a mode of operation, called CCM, for a symmetric key block cipher algorithm. CCM may be used to provide assurance of the confidentiality and the authenticity of computer data by combining the techniques of the Counter (CTR) mode and the Cipher Block Chaining-Message Authentication Code (CBC-MAC) algorithm.

**KEY WORDS:** authenticated encryption; authentication; block cipher; confidentiality; cryptography; encryption; information security; message authentication code; mode of operation.

## Table of Contents

<b>1</b>	<b>PURPOSE</b> .....	<b>1</b>
<b>2</b>	<b>AUTHORITY</b> .....	<b>1</b>
<b>3</b>	<b>INTRODUCTION</b> .....	<b>1</b>
<b>4</b>	<b>DEFINITIONS, ABBREVIATIONS, AND SYMBOLS</b> .....	<b>2</b>
4.1	DEFINITIONS AND ABBREVIATIONS .....	2
4.2	SYMBOLS .....	4
4.2.1	<i>Variables</i> .....	4
4.2.2	<i>Operations and Functions</i> .....	6
<b>5</b>	<b>PRELIMINARIES</b> .....	<b>6</b>
5.1	UNDERLYING BLOCK CIPHER ALGORITHM .....	6
5.2	CRYPTOGRAPHIC PRIMITIVES .....	7
5.3	DATA ELEMENTS.....	7
5.4	INPUT FORMATTING .....	8
5.5	EXAMPLES OF OPERATIONS AND FUNCTIONS .....	8
<b>6</b>	<b>CCM SPECIFICATION</b> .....	<b>9</b>
6.1	GENERATION-ENCRYPTION PROCESS .....	9
6.2	DECRYPTION-VERIFICATION PROCESS .....	10
<b>APPENDIX A: EXAMPLE OF A FORMATTING AND COUNTER GENERATION FUNCTION</b> .....		<b>12</b>
A.1	LENGTH REQUIREMENTS .....	12
A.2	FORMATTING OF THE INPUT DATA .....	12
A.2.1	<i>Formatting of the Control Information and the Nonce</i> .....	13
A.2.2	<i>Formatting of the Associated Data</i> .....	13
A.2.3	<i>Formatting of the Payload</i> .....	14
A.3	FORMATTING OF THE COUNTER BLOCKS.....	14
<b>APPENDIX B: LENGTH OF THE MAC</b> .....		<b>15</b>
B.1	AUTHENTICATION ASSURANCE .....	15
B.2	SELECTION OF THE MAC LENGTH.....	16
<b>APPENDIX C: EXAMPLE VECTORS</b> .....		<b>17</b>
C.1	EXAMPLE 1.....	17
C.2	EXAMPLE 2.....	17
C.3	EXAMPLE 3.....	18
C.4	EXAMPLE 4.....	19
<b>APPENDIX D: REFERENCES</b> .....		<b>21</b>

## Table of Figures

Table 1:	Formatting of the Flags Octet in $B_0$ .....	13
Table 2:	Formatting of $B_0$ .....	13
Table 3:	Formatting of $Ctr_i$ .....	14
Table 4:	Formatting of the Flags Field in $Ctr_i$ .....	14

## **1 Purpose**

This publication is the third Part in a series of Recommendations regarding modes of operation of symmetric key block cipher algorithms.

## **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 an algorithm, Counter with Cipher Block Chaining-Message Authentication Code [1], abbreviated CCM, that can provide assurance of the confidentiality and authenticity of data. CCM is based on an approved symmetric key block cipher algorithm whose block size is 128 bits, such as the Advanced Encryption Standard (AES) algorithm currently specified in Federal Information Processing Standard (FIPS) Pub. 197 [2]; thus, CCM cannot be used with the Triple Data Encryption Algorithm [3], whose block size is 64 bits. CCM can be considered a mode of operation of the block cipher algorithm. As with other modes of operation,

a single key to the block cipher must be established beforehand among the parties to the data; thus, CCM should be implemented within a well-designed key management structure. The security properties of CCM depend, at a minimum, on the secrecy of the key.

CCM is intended for use in a packet environment, i.e., when all of the data is available in storage before CCM is applied; CCM is not designed to support partial processing or stream processing. The input to CCM includes three elements: 1) data that will be both authenticated and encrypted, called the payload; 2) associated data, e.g., a header, that will be authenticated but not encrypted; and 3) a unique value, called a nonce, that is assigned to the payload and the associated data.

CCM consists of two related processes: generation-encryption and decryption-verification, which combine two cryptographic primitives: counter mode encryption and cipher block chaining-based authentication. Only the forward cipher function of the block cipher algorithm is used within these primitives.

In generation-encryption, cipher block chaining is applied to the payload, the associated data, and the nonce to generate a message authentication code (MAC); then, counter mode encryption is applied to the MAC and the payload to transform them into an unreadable form, called the ciphertext. Thus, CCM generation-encryption expands the size of the payload by the size of the MAC. In decryption-verification, counter mode decryption is applied to the purported ciphertext to recover the MAC and the corresponding payload; then, cipher block chaining is applied to the payload, the received associated data, and the received nonce to verify the correctness of the MAC. Successful verification provides assurance that the payload and the associated data originated from a source with access to the key.

A MAC provides stronger assurance of authenticity than a checksum or an error detecting code. The verification of a (non-cryptographic) checksum or an error detecting code is designed to detect only accidental modifications of the data, while the verification of a MAC, as occurs in CCM, is designed to detect intentional, unauthorized modifications of the data, as well as accidental modifications.

This specification of CCM is intended to be compatible with the use of CCM within the draft amendment [4] to the IEEE Standard 802.11 for wireless local area networks [5].

## 4 Definitions, Abbreviations, and Symbols

### 4.1 Definitions and Abbreviations

Approved	FIPS approved or NIST recommended: an algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation.
Associated Data	Input data to the CCM generation-encryption process that is authenticated but not encrypted.



Authenticity	The property that data originated from its purported source.
Bit	A binary digit: 0 or 1.
Bit Length	The number of bits in a bit string.
Bit String	An ordered sequence of bits.
Block	A bit string whose length is the block size of the block cipher algorithm.
Block Cipher Algorithm	A family of functions and their inverses that is parameterized by cryptographic keys; the functions map bit strings of a fixed length to bit strings of the same length.
Block Size	The bit length of an input (or output) block of the block cipher.
CBC-MAC	Cipher Block Chaining-Message Authentication Code
CCM	Counter with Cipher Block Chaining-Message Authentication Code.
Ciphertext	The output of the CCM encryption-generation process.
Cryptographic Key	A parameter used in the block cipher algorithm that determines the forward cipher function.
CTR	Counter.
Decryption-Verification	The process of CCM in which a purported ciphertext is decrypted and the authenticity of the resulting payload and the associated data is verified.
Exclusive-OR	The bitwise addition, modulo 2, of two bit strings of equal length.
FIPS	Federal Information Processing Standard.
Formatting Function	The function that transforms the payload, associated data, and nonce into a sequence of complete blocks.
Forward Cipher Function	One of the two functions of the block cipher algorithm that is determined by the choice of a cryptographic key.
Generation-Encryption	The process of CCM in which a MAC is generated on the payload and the associated data, and encryption is applied to the payload and the MAC.

IEEE	Institute of Electrical and Electronics Engineers
Inverse Cipher Function	The inverse function of the forward cipher function for a given cryptographic key.
Least Significant Bit(s)	The right-most bit(s) of a bit string.
Message Authentication Code (MAC)	A cryptographic checksum on data that is designed to reveal both accidental errors and intentional modifications of the data.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that features a symmetric key block cipher algorithm.
Most Significant Bit(s)	The left-most bit(s) of a bit string.
Nonce	A value that is used only once within a specified context.
Octet	A string of eight bits.
Octet Length	The number of octets in an octet string.
Octet String	An ordered sequence of octets.
Payload	The input data to the CCM generation-encryption process that is both authenticated and encrypted.
Valid Data Element	A payload, an associated data string, or a nonce that satisfies the restrictions of the formatting function.

## 4.2 Symbols

### 4.2.1 Variables

$a$  The octet length of the associated data.

$A$  The associated data string.

$A_{len}$  The bit length of the associated data.

$B_i$  The  $i$ th block of the formatted input.

$C$  The ciphertext.

$Clen$	The bit length of the ciphertext.
$Ctr_i$	The $i$ th counter block.
$K$	The block cipher key.
$Klen$	The bit length of the block cipher key.
$m$	The number of blocks in the formatted payload.
$MaxErrs$	The maximum number of times that the output of any implementation of the decryption-verification process can be INVALID before the key is retired.
$n$	The octet length of the nonce.
$N$	The nonce.
$Nlen$	The bit length of the nonce.
$P$	The payload.
$Plen$	The bit length of the payload.
$q$	The octet length of the binary representation of the octet length of the payload.
$Q$	A bit string representation of the octet length of $P$ .
$r$	The number of blocks in the formatted input data $(N, A, P)$ .
$Risk$	The highest acceptable probability for an inauthentic message to pass the decryption-verification process.
$t$	The octet length of the MAC.
$T$	The MAC that is generated as an internal variable in the CCM processes.
$Tlen$	The bit length of the MAC.

#### 4.2.2 Operations and Functions

$0x$	The prefix to a bit string that is represented in hexadecimal characters.
$[x]_s$	The binary representation of the non-negative integer $x$ , in $s$ bits, where $x < 2^s$ .
$\lceil x \rceil$	The least integer that is not less than the real number $x$ .
$X \parallel Y$	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 algorithm under the key $K$ applied to the data block $X$ .
$\text{LSB}_s(X)$	The bit string consisting of the $s$ right-most bits of the bit string $X$ .
$\text{MSB}_s(X)$	The bit string consisting of the $s$ left-most bits of the bit string $X$ .
$\lg(x)$	The base 2 logarithm of the positive real number $x$ .

## 5 Preliminaries

The selection of a block cipher algorithm and secret key are discussed in Section 5.1. The two cryptographic primitives that CCM requires for its operation are discussed in Section 5.2. The data elements of CCM are discussed in Section 5.3. The formatting of valid data elements is discussed in Section 5.4. Examples of operations and functions are given in Section 5.5.

### 5.1 Underlying Block Cipher Algorithm

The CCM algorithm depends on the choice of an underlying symmetric key block cipher algorithm. The CCM algorithm is thus a mode of operation (mode, for short) of the symmetric key block cipher. The underlying block cipher algorithm shall be approved, and a secret key for the block cipher algorithm 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. Moreover, the key should be established for the parties to the information by an approved key establishment method. The key shall be kept secret and shall only be used for the CCM mode. The total number of invocations of the block cipher algorithm during the lifetime of the key shall be limited to  $2^{61}$ . Key establishment and key management are outside the scope of this Recommendation.

For any given key, the underlying block cipher algorithm of the mode consists of two functions that are inverses of each other. As part of the choice of the block cipher algorithm, one of the two

functions of the block cipher algorithm is designated as the forward cipher function. \* The inverse of this process is called the inverse cipher function; however, the CCM mode does not require the inverse cipher function.

The forward cipher function is a function on bit strings of a fixed bit length; the strings are called blocks, and their length is called the block size. For CCM, the block size of the block cipher algorithm shall be 128 bits; currently, the AES algorithm is the only approved block cipher algorithm with this block size.

The CCM key, denoted  $K$ , is the block cipher key; the forward cipher function of the block cipher with this key is denoted  $CIPH_K$ . The bit length of  $K$  is denoted  $Klen$ .

## 5.2 *Cryptographic Primitives*

The CCM specification essentially combines two cryptographic mechanisms that are based on the forward cipher function. One mechanism is the Counter (CTR) mode for confidentiality, which is specified for general use in the first Part of this Recommendation [6]. The CTR mode requires the generation of a sufficiently long sequence of blocks called the counter blocks. The counter blocks must be distinct within a single invocation and across all other invocations of the CTR mode under any given key, but they need not be secret. This requirement on the counter blocks extends to the CCM mode. See [6] for further discussion of the generation of counter blocks; see Section 5.4 below for an additional requirement on the counter blocks in CCM.

The other cryptographic mechanism within CCM is an adaptation of the cipher block chaining (CBC) technique from [6] to provide assurance of authenticity. Specifically, the CBC technique with an initialization vector of zero is applied to the data to be authenticated; the final block of the resulting CBC output, possibly truncated, serves as a message authentication code (MAC) of the data. The algorithm for generating a MAC in this fashion is commonly called CBC-MAC. This Recommendation does not approve CBC-MAC as an authentication mode outside of the context of the CCM specification; however, a variation of CBC-MAC is proposed for general use in the second Part of this Recommendation [7].

The same key,  $K$ , is used for both the CTR and CBC-MAC mechanisms within CCM.

## 5.3 *Data Elements*

The data that CCM protects consists of a message, i.e., a bit string, called the payload, denoted  $P$ , of bit length denoted  $Plen$ , and a bit string, called the associated data, denoted  $A$ . The associated data is optional, i.e.,  $A$  may be the empty string.\*\* CCM provides assurance of the confidentiality of  $P$  and assurance of the authenticity of the origin of both  $A$  and  $P$ ; confidentiality is not provided for  $A$ .

A bit string called the nonce, denoted  $N$ , is assigned to the data pair to be protected, i.e., the

---

\* For the AES algorithm, the forward cipher is explicitly identified in [2].

\*\* The payload may also be empty, in which case the specification degenerates to an authentication mode on the associated data.

payload and its associated data. The nonce shall be non-repeating in the sense that any two distinct data pairs to be protected by CCM during the lifetime of the key shall be assigned distinct nonces. In effect, the nonce determines an invocation of CCM. The nonce is not required to be random.

The MAC that is generated within CCM is an internal variable, denoted  $T$ . The bit length of  $T$ , denoted  $Tlen$ , is a parameter of the mode that shall be fixed for all invocations of CCM with the given key. The requirements for the selection of  $Tlen$  are discussed in Appendix B.

#### 5.4 Input Formatting

The CCM data elements  $N$ ,  $P$ , and  $A$  shall be formatted into a non-empty sequence of complete data blocks, denoted  $B_0, B_1, \dots, B_r$  for some non-negative integer  $r$ , in accordance with a function called the formatting function (in [8], the formatting function is called the encoding function,  $\beta$ ). The value of  $r$  depends on the formatting function and the input elements. For any given key, the following three properties shall hold for the formatting function:

1. The first block,  $B_0$ , uniquely determines the nonce  $N$ .
2. The formatted data uniquely determines  $P$  and  $A$ ; moreover, if  $(N, P, A)$  and  $(N, P', A')$  are distinct input triples whose formatting is  $B_0, B_1, \dots, B_r$  and  $B_0', B_1', \dots, B_r'$ , then  $B_i$  is distinct from  $B_i'$  for some index  $i$  such that  $i \leq r$  and  $i \leq r'$ . \*
3. The first block,  $B_0$ , is distinct from any counter blocks that are used across all invocations of CCM under the key.

The third property suggests that the formatting function and the counter generation function should not be selected or constructed independently of each other.

The formatting function may impose restrictions on the contents and the bit lengths of the input data  $N$ ,  $P$ , and  $A$ , as well as the parameter  $Tlen$ . For example, the bit lengths may be restricted to multiples of eight and to a given range of values. Values that satisfy the restrictions of the formatting function are called valid. No value of  $Tlen$  smaller than 32 shall be valid.

An example of a formatting function and counter generation function that together satisfy these requirements is given in Appendix A. Alternative formatting functions may be developed in the future.

#### 5.5 Examples of Operations and Functions

Given a positive integer  $s$  and a non-negative integer  $x$  that is less than  $2^s$ , the binary representation of  $x$  in  $s$  bits is denoted  $[x]_s$ . For example, for the (base 10) integer 45, the binary representation (base 2) is 101101, so  $[45]_8 = 00101101$ .

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

---

\* This property deals with the case that a nonce is repeated within two distinct input triples. Although the legitimate source/sender of the data is already required to assign distinct nonces to different input triples, the security model in [8] takes into account that the recipient of purported ciphertxts might not monitor whether a nonce is repeated.

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,  $10011 \oplus 10101 = 00110$ .

The functions  $LSB_s$  and  $MSB_s$  return the  $s$  least significant (i.e., right-most) bits and the  $s$  most significant (i.e., left-most) bits of their arguments, respectively. For example,  $LSB_3(111011010) = 010$ , and  $MSB_4(111011010) = 1110$ .

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

## 6 CCM Specification

The two CCM processes are called generation-encryption and decryption-verification. The prerequisites, inputs, outputs, steps, and summaries for the execution of the two CCM processes are specified in Sections 6.1 and 6.2 below. The prerequisites are inputs that are typically fixed across many invocations of CCM. The prerequisites and inputs shall meet the requirements in Section 5.

There is some flexibility in the order of the steps of the two processes. For example, the generation of the counter blocks can occur at any time before they are used; in fact, the counter blocks may be generated in advance and be considered as inputs to the processes.

### 6.1 Generation-Encryption Process

The following is a specification of the generation-encryption process of CCM:

*Prerequisites:*

block cipher algorithm;  
key  $K$ ;  
counter generation function;  
formatting function;  
MAC length  $Tlen$ .

*Input:*

valid nonce  $N$ ;  
valid payload  $P$  of length  $Plen$  bits;  
valid associated data  $A$ ;

*Output:*

ciphertext  $C$ .

*Steps:*

1. Apply the formatting function to  $(N, A, P)$  to produce the blocks  $B_0, B_1, \dots, B_r$ .
2. Set  $Y_0 = \text{CIPH}_K(B_0)$ .
3. For  $i = 1$  to  $r$ , do  $Y_i = \text{CIPH}_K(B_i \oplus Y_{i-1})$ .

4. Set  $T = \text{MSB}_{Tlen}(Y_r)$ .
5. Apply the counter generation function to generate the counter blocks  $Ctr_0, Ctr_1, \dots, Ctr_m$ , where  $m = \lceil Plen/128 \rceil$ .
6. For  $j=0$  to  $m$ , do  $S_j = \text{CIPH}_K(Ctr_j)$ .
7. Set  $S = S_1 \parallel S_2 \parallel \dots \parallel S_m$ .
8. Return  $C = (P \oplus \text{MSB}_{Plen}(S)) \parallel (T \oplus \text{MSB}_{Tlen}(S_0))$ .

The input data to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, which are formatted according to the formatting function. The CBC-MAC mechanism is applied to the formatted data to generate a MAC, whose length is a prerequisite. Counter mode encryption, which requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting data, called the ciphertext, denoted  $C$ , is the output of the generation-encryption process.

## 6.2 Decryption-Verification Process

*Prerequisites:*

block cipher algorithm;  
 key  $K$ ;  
 counter generation function;  
 formatting function;  
 valid MAC length  $Tlen$ .

*Input:*

nonce  $N$ ;  
 associated data  $A$ ;  
 purported ciphertext  $C$  of length  $Clen$  bits;

*Output:*

either the payload  $P$  or INVALID.

*Steps:*

1. If  $Clen \leq Tlen$ , then return INVALID.
2. Apply the counter generation function to generate the counter blocks  $Ctr_0, Ctr_1, \dots, Ctr_m$ , where  $m = \lceil (Clen - Tlen)/128 \rceil$ .
3. For  $j=0$  to  $m$ , do  $S_j = \text{CIPH}_K(Ctr_j)$ .
4. Set  $S = S_1 \parallel S_2 \parallel \dots \parallel S_m$ .
5. Set  $P = \text{MSB}_{Clen-Tlen}(C) \oplus \text{MSB}_{Clen-Tlen}(S)$ .
6. Set  $T = \text{LSB}_{Tlen}(C) \oplus \text{MSB}_{Tlen}(S_0)$ .
7. If  $N, A$ , or  $P$  is not valid, as discussed in Section 5.4, then return INVALID, else apply the formatting function to  $(N, A, P)$  to produce the blocks  $B_0, B_1, \dots, B_r$ .
8. Set  $Y_0 = \text{CIPH}_K(B_0)$ .
9. For  $i = 1$  to  $r$ , do  $Y_j = \text{CIPH}_K(B_i \oplus Y_{i-1})$ .
10. If  $T \neq \text{MSB}_{Tlen}(Y_r)$ , then return INVALID, else return  $P$ .



The input to the decryption-verification process is a purported ciphertext, an associated data string, and the nonce that was purportedly used in the generation of the purported ciphertext. Counter mode decryption is applied to the purported ciphertext to produce the corresponding MAC and payload. If the nonce, the associated data string, and the payload are valid, as discussed in Section 5.4, then these strings are formatted into blocks according to the formatting function, and the CBC-MAC mechanism is applied to verify the MAC. If verification succeeds, the decryption-verification process returns the payload as output; otherwise, only the error message INVALID is returned.

When the error message INVALID is returned, the payload  $P$  and the MAC  $T$  shall not be revealed. Moreover, the implementation shall ensure that an unauthorized party cannot distinguish whether the error message results from Step 7 or from Step 10, for example, from the timing of the error message.

## Appendix A: Example of a Formatting and Counter Generation Function

In this appendix, a formatting function for the input data and a counter generation function are specified that together satisfy the requirements of Section 5.4. With these functions, this specification of CCM is essentially equivalent to the specification of CCM in the draft amendment [4] to the IEEE Standard 802.11 for wireless local area networks [5].

The requirements that this particular formatting function imposes on the lengths of the variables in CCM are given in Section A.1; the formatting of the input data is specified in Section A.2; the counter generation function is specified in Section A.3.

### A.1 Length Requirements

The bit length of each input string, i.e.,  $N$ ,  $A$ , and  $P$ , shall be a multiple of 8 bits, i.e., each input string shall be an octet string. The octet lengths of these strings are denoted  $n$ ,  $a$ , and  $p$ ; thus,  $n$ ,  $a$  and  $p$  are integers. Similarly, the parameter  $t$  denotes the octet length of  $T$ . The octet length of  $P$  (i.e., the integer  $p$ ) is represented within the first block of the formatted data as an octet string denoted  $Q$ . The octet length of  $Q$ , denoted  $q$ , is a parameter of the formatting function. Thus,  $Q$  is equivalent to  $[p]_{8q}$ , the binary representation of  $p$  in  $q$  octets. For example, if  $q = 3$  and  $P$  is a string that consists of 4096 bits, i.e.,  $p = 512$ , then  $Q$  is the string 00000000 00000010 00000000.

The formatting in this appendix imposes the following length conditions:

- $t$  is an element of  $\{4, 6, 8, 10, 12, 14, 16\}$ ;
- $q$  is an element of  $\{2, 3, 4, 5, 6, 7, 8\}$ ;
- $n$  is an element of  $\{7, 8, 9, 10, 11, 12, 13\}$
- $n+q=15$ ;
- $a < 2^{64}$ .

The parameter  $q$  determines the maximum length of the payload: by definition,  $p < 2^{8q}$ , so  $P$  consists of fewer than  $2^{8q}$  octets, i.e., fewer than  $2^{8q-4}$  128-bit blocks. The fourth condition implies that a choice for  $q$  determines the value of  $n$ , namely,  $n=15-q$ . (Equivalently, the nonce length,  $n$ , may be considered as the parameter of the formatting function that determines the value of  $q$ .) The value of  $n$ , in turn, determines the maximum number of distinct nonces, namely,  $2^{8n}$ . Thus, the fourth condition amounts to a tradeoff between the maximum number of invocations of CCM under a given key and the maximum payload length for those invocations.

### A.2 Formatting of the Input Data

The formatting of the input data  $(N, A, P)$  into a sequence of blocks  $B_0, B_1, \dots, B_r$  is presented in the following three sections: in Section A.2.1, the formatting of the nonce and control information such as length indicators is specified; in Section A.2.2, the formatting of  $A$  is specified; in Section A.2.3, the formatting of  $P$  is specified.

### A.2.1 Formatting of the Control Information and the Nonce

The leading octet of the first block of the formatting,  $B_0$ , contains four flags for control information: two single bits, called *Reserved* and *Adata*, and two strings of three bits, to encode the values  $t$  and  $q$ . The encoding of  $t$  is  $[(t-2)/2]_3$ , and the encoding of  $q$  is  $[q-1]_3$ . Thus, for example, if the MAC length is 8 octets, then  $t$  is encoded as 011. Note that the encoding 000 in both cases does not correspond to a permitted value of  $t$  or  $q$ . The *Reserved* bit is reserved to enable future extensions of the formatting; it shall be set to '0'. The *Adata* bit is '0' if  $a=0$  and '1' if  $a>0$ . The ordering of the flags within the octet is given in Table 1.

Table 1: Formatting of the Flags Octet in  $B_0$

Bit number	7	6	5	4	3	2	1	0
Contents	<i>Reserved</i>	<i>Adata</i>	$[(t-2)/2]_3$			$[q-1]_3$		

The remaining 15 octets of the first block of the formatting are devoted to the nonce and the binary representation of the message length in  $q$  octets, as given in Table 2.

Table 2: Formatting of  $B_0$

Octet number	0	1 ... 15- $q$	16- $q$ ... 15
Contents	Flags	$N$	$Q$

For example, if  $B_0$  is

```
01101110 00010011 11010100 10100011 01011101 01110001 10100101 00000000
00000000 00000000 00000000 00000000 00000000 00000000 01000100 00000001:
```

- The associated data will not be empty (because  $Adata=1$ ).
- The MAC will consist of 12 octets (because  $[(t-2)/2]_3=101$ ).
- The octet length of  $Q$  is 7 (because  $[q-1]_3=110$ ), so  $Q$  is 00000000 00000000 00000000 00000000 01000100 00000001.
- The payload will consist of 17,409 octets (because  $Q=[17409]_{56}$ ).
- The octet length of  $N$  is 8 (because  $n=15-q$  and  $q=7$ ), so  $N=00010011 11010100 10100011 01011101 01110001 10100101 00000000 00000000$ .

### A.2.2 Formatting of the Associated Data

If  $a=0$ , as indicated by the *Adata* field in the first octet of  $B_0$ , then there are no blocks devoted to the associated data in the formatted data. If  $a>0$ , then  $a$  is encoded as described below, and the encoding of  $a$  is concatenated with the associated data  $A$ , followed by the minimum number of '0' bits, possibly none, such that the resulting string can be partitioned into 16-octet blocks. These blocks are denoted in the formatted data as  $B_1, B_2, \dots B_u$  for some positive integer  $u$  that depends on  $a$ .

The value  $a$  is encoded according to the following three cases:

- If  $0 < a < 2^{16} \cdot 2^8$ , then  $a$  is encoded as  $[a]_{16}$ , i.e., two octets.
- If  $2^{16} \cdot 2^8 \leq a < 2^{32}$ , then  $a$  is encoded as  $0\text{xff} \parallel 0\text{xfe} \parallel [a]_{32}$ , i.e., six octets.
- If  $2^{32} \leq a < 2^{64}$ , then  $a$  is encoded as  $0\text{xff} \parallel 0\text{xff} \parallel [a]_{64}$ , i.e., ten octets.

For example, if  $a=2^{16}$ , the encoding of  $a$  is 11111111 11111110 00000000 00000001 00000000 00000000.

The formatting of distinct sets of associated data will not overlap, because for distinct values of  $a$ , the leading bits of the encodings of  $a$  are distinct: in the first case, the first octet will not be  $0\text{xff}$  as it will for the second and third cases; the second and third cases can be distinguished by the second octet. Encodings that are not specified in these three cases are reserved, e.g., when the first two octets are  $0\text{x}0000$ ,  $0\text{xff}00$ ,  $0\text{xff}01$ , etc.

### A.2.3 Formatting of the Payload

The associated data blocks, if any, are followed in the sequence of formatted blocks by the payload blocks. The payload is concatenated with the minimum number of ‘0’ bits, possibly none, such that the result can be partitioned into 16-octet blocks. These blocks are denoted in the formatted data as  $B_{u+1}, B_{u+2} \dots B_r$ , where  $r=u+\lceil p/16 \rceil$ .

### A.3 Formatting of the Counter Blocks

The counter generation function in this section is equivalent to a formatting of the counter index  $i$  into a complete data block. The counter blocks  $Ctr_i$  are formatted as shown in Table 3 below.

Table 3: Formatting of  $Ctr_i$

Octet number:	0	1 ... 15- $q$	16- $q$ ... 15
Contents:	Flags	$N$	$[i]_{8q}$

Within each block  $Ctr_i$ , the Flags field is formatted as shown in Table 4 below.

Table 4: Formatting of the Flags Field in  $Ctr_i$

Bit number	7	6	5	4	3	2	1	0
Contents	Reserved	Reserved	0	0	0	$[q-1]_3$		

The *Reserved* bits are reserved for future expansions and shall be set to 0. Bits 3, 4, and 5 shall also be set to 0, to ensure that all the counter blocks are distinct from  $B_0$  (as specified in A.2.1 above). Bits 0, 1, and 2 contain the same encoding of  $q$  as in  $B_0$ .

## Appendix B: Length of the MAC

The length,  $Tlen$ , of the MAC within CCM is an important security parameter. The role of this parameter in providing authentication assurance is outlined in Section B.1, and guidance in the selection of  $Tlen$  is given in Section B.2.

### B.1 Authentication Assurance

The decryption-verification process determines whether a purported ciphertext input is, in fact, a ciphertext, i.e., the output of the generation-encryption process with the given key, associated data string, and nonce. The basis of the CCM authentication assurance is the scarcity of ciphertexts, which suggests that an attacker, i.e., a party without access to the key or to the generation-encryption process, cannot easily generate a ciphertext. Therefore, any purported ciphertext that passes the decryption-verification process was probably generated legitimately.

For any purported ciphertext that is at least  $Tlen$  bits long, the rightmost  $Tlen$  bits correspond to an encrypted MAC, and the remaining bits correspond to an encrypted payload. The decryption-verification process checks the correctness of the MAC for the payload, the associated data string, and the nonce. Depending on the result, the output is either the error message INVALID or the payload. The nature of the authentication assurance, in turn, depends on the output:

- If the output is INVALID, then the payload and the associated data string cannot both be authentic, i.e., they cannot have originated from a source that executed the generation-encryption process on them with the given nonce to produce the purported ciphertext.
- If the output is the payload, then the design of the mode provides assurance that the payload and the associated data are authentic. This assurance, however, is not absolute: an attacker can generate a ciphertext with a certain probability.

In the second case, the scarcity of ciphertexts and, thus, the expected probability that an attacker can guess a ciphertext, is regulated by the size of  $Tlen$ . In particular, the probability is no greater than  $1$  in  $2^{Tlen}$  that a single inauthentic purported ciphertext will pass the decryption-verification process with a given associated data string and nonce. Of course, an attacker may attempt to present many purported ciphertexts to the decryption-verification process and thereby increase the probability that at least one of them will turn out to be a true ciphertext.

Moreover, an attacker may be able to choose the purported ciphertext in such a way as to control every bit of the corresponding payload, as described in [9].\*

---

\* In particular, given a single valid ciphertext, and the corresponding payload and nonce, the attacker chooses any set of bits to flip in the payload and flips the corresponding set of bits in the ciphertext. The attacker then tries to induce the decryption-verification process to reuse the associated counter blocks with the altered ciphertext. When the counter blocks are generated as in Appendix A, this step is equivalent to inducing the decryption-verification process to reuse the nonce. In principle, a system could monitor the nonces that are presented to the decryption-verification process to facilitate some defense against this attack, but such monitoring may not be feasible in practice.

decryption-verification process is a payload, the contents of the payload do not constitute evidence for its authenticity beyond the assurance that the mode already provides.

Also, an attacker may be able to intercept a legitimate ciphertext and “replay” it for verification at a later time. To defend against such an event, the controlling protocol or application should typically provide the receiver with a means to detect replayed messages, out-of-sequence messages, and missing messages, for example, by numbering legitimate messages sequentially.

## ***B.2 Selection of the MAC Length***

Larger values of  $Tlen$  provide greater authentication assurance, as described in Section B.1. The performance tradeoff is that larger values of  $Tlen$  require more bandwidth/storage for the ciphertext.

Although the formatting function in Appendix A permits  $Tlen$  to be any integer multiple of 16 between 32 and 128, inclusive, a value of  $Tlen$  that is less than 64 shall not be used without a careful analysis of the risks of accepting inauthentic data as authentic.

In particular, a value of  $Tlen$  smaller than 64 should not be used unless the controlling protocol or application environment sufficiently restricts the number of times that the decryption-verification process can return INVALID, across all implementations under 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.

Similarly, for larger values of  $Tlen$ , the controlling protocol or the application environment should limit the number of inauthentic input sets that may be presented to the decryption-verification process to a number commensurate with the value of the protected data.

This guidance can be quantified in terms of the following two bounds: 1) the highest acceptable probability for an inauthentic message to pass the decryption-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 decryption-verification process under the key. Given estimates of these quantities, denoted  $Risk$  and  $MaxErrs$ , respectively,  $Tlen$  should satisfy the following inequality:

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

For example, suppose that a system will not output INVALID for more than 1024 messages before retiring the key (i.e.,  $MaxErrs=2^{10}$ ), 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,  $Tlen$  may be as low as 32. On the other hand, if  $MaxErrs=2^{32}$  and  $Risk=2^{-32}$ , then  $Tlen$  should be at least 64.

## Appendix C: Example Vectors

In this appendix, four examples are provided for the encryption-generation process of CCM with the formatting and counter generation functions that are specified in Appendix A. The underlying block cipher algorithm is the AES algorithm [2] under a key of 128 bits. The MACs, the nonces, the associated data strings, and the payload strings have different lengths in each example, including a very long, repetitive string of associated data in Example 4. The bit strings are represented in hexadecimal notation.

From each example, a corresponding example of the decryption-verification process of CCM is straightforward to construct.

### C.1 Example 1

In the following example,  $Klen = 128$ ,  $Tlen=32$ ,  $Nlen = 56$ ,  $Alen = 64$ , and  $Plen = 32$ .

```
K:          40414243 44454647 48494a4b 4c4d4e4f
N:          10111213 141516
A:          00010203 04050607
P:          20212223

B:          4f101112 13141516 00000000 00000004
           00080001 02030405 06070000 00000000
           20212223 00000000 00000000 00000000

T:          6084341b

Ctr0:      07101112 13141516 00000000 00000000
S0:       2d281146 10676c26 32bad748 559a679a

Ctr1:      07101112 13141516 00000000 00000001
S1:       51432378 e474b339 71318484 103cddfbb

C:          7162015b 4dac255d
```

### C.2 Example 2

In the following example,  $Klen = 128$ ,  $Tlen=48$ ,  $Nlen = 64$ ,  $Alen = 128$ , and  $Plen = 128$ .

```
K:          40414243 44454647 48494a4b 4c4d4e4f
N:          10111213 14151617
A:          00010203 04050607 08090a0b 0c0d0e0f
P:          20212223 24252627 28292a2b 2c2d2e2f
```

```

B:          56101112 13141516 17000000 00000010
            00100001 02030405 06070809 0a0b0c0d
            0e0f0000 00000000 00000000 00000000
            20212223 24252627 28292a2b 2c2d2e2f

T:          7f479ffc a464

Ctr0:      06101112 13141516 17000000 00000000
S0:       6081d043 08a97dcc 20cdcc60 bf947b78

Ctr1:      06101112 13141516 17000000 00000001
S1:       f280d2c3 75cf7945 20335db9 2b107712

C:          d2a1f0e0 51ea5f62 081a7792 073d593d
            1fc64fbf accd

```

### C.3 Example 3

In the following example,  $Klen = 128$ ,  $Tlen=64$ ,  $Nlen = 96$ ,  $Alen = 160$ , and  $Plen = 192$ .

```

K:          40414243 44454647 48494a4b 4c4d4e4f
N:          10111213 14151617 18191a1b
A:          00010203 04050607 08090a0b 0c0d0e0f
            10111213
P:          20212223 24252627 28292a2b 2c2d2e2f
            30313233 34353637

B:          5a101112 13141516 1718191a 1b000018
            00140001 02030405 06070809 0a0b0c0d
            0e0f1011 12130000 00000000 00000000
            20212223 24252627 28292a2b 2c2d2e2f
            30313233 34353637 00000000 00000000

T:          67c99240 c7d51048

Ctr0:      02101112 13141516 1718191a 1b000000
S0:       2f8a00bb 06658919 c3a040a6 eaed1a7f

Ctr1:      02101112 13141516 1718191a 1b000001
S1:       c393238a d1923c5d b335c0c7 e1bac924

Ctr2:      02101112 13141516 1718191a 1b000002
S2:       514798ea 9077bc92 6c22ebef 2ac732dc

C:          e3b201a9 f5b71a7a 9b1ceaec cd97e70b

```



6176aad9 a4428aa5 484392fb c1b09951

#### C.4 Example 4

In the following example,  $Klen = 128$ ,  $Tlen=112$ ,  $Nlen = 104$ ,  $Alen = 524288$ , and  $Plen = 256$ . The associated data string is too large to comfortably present in its entirety; therefore, the given string of the first sixteen blocks of the associated data string is concatenated with itself repeatedly to form a string of 524288 bits. Similarly, only the beginning and the end of the resulting formatted data string  $B$  are presented.

```
K:      40414243 44454647 48494a4b 4c4d4e4f
N:      10111213 14151617 18191a1b 1c
A:      00010203 04050607 08090a0b 0c0d0e0f
        10111213 14151617 18191a1b 1c1d1e1f
        20212223 24252627 28292a2b 2c2d2e2f
        30313233 34353637 38393a3b 3c3d3e3f
        40414243 44454647 48494a4b 4c4d4e4f
        50515253 54555657 58595a5b 5c5d5e5f
        60616263 64656667 68696a6b 6c6d6e6f
        70717273 74757677 78797a7b 7c7d7e7f
        80818283 84858687 88898a8b 8c8d8e8f
        90919293 94959697 98999a9b 9c9d9e9f
        a0a1a2a3 a4a5a6a7 a8a9aaab acadaaef
        b0b1b2b3 b4b5b6b7 b8b9babb bcbdbebf
        c0c1c2c3 c4c5c6c7 c8c9cacb cccdcecf
        d0d1d2d3 d4d5d6d7 d8d9dadb dcdddedf
        e0e1e2e3 e4e5e6e7 e8e9eae ecedeeef
        f0f1f2f3 f4f5f6f7 f8f9fafb fcfdfeff
        ...
P:      20212223 24252627 28292a2b 2c2d2e2f
        30313233 34353637 38393a3b 3c3d3e3f
B:      71101112 13141516 1718191a 1b1c0020
        fffe0001 00000001 02030405 06070809
        0a0b0c0d 0e0f1011 12131415 16171819
        1a1b1c1d 1e1f2021 22232425 26272829
        ...
        dadbdcdd dedfe0e1 e2e3e4e5 e6e7e8e9
        eaebebed eeef0f1 f2f3f4f5 f6f7f8f9
        fafbfcfd feff0000 00000000 00000000
        20212223 24252627 28292a2b 2c2d2e2f
        30313233 34353637 38393a3b 3c3d3e3f
```

*T*: f4dd5d0e e4046172 25ffe34f ce91

*Ctr*<sub>0</sub>: 01101112 13141516 1718191a 1b1c0000  
*S*<sub>0</sub>: 407136e2 77ec38fc 5af24ef3 24ca1178

*Ctr*<sub>1</sub>: 01101112 13141516 1718191a 1b1c0001  
*S*<sub>1</sub>: 49b07f8e 3aa1e010 4241e8bd 5260854e

*Ctr*<sub>2</sub>: 01101112 13141516 1718191a 1b1c0002  
*S*<sub>2</sub>: 6ad1cf2c 9af17af3 bcbbbf12 7a01f14d

*C*: 69915dad 1e84c637 6a68c296 7e4dab61  
5ae0fd1f aec44cc4 84828529 463ccf72  
b4ac6bec 93e8598e 7f0dadbc ea5b

## Appendix D: References

- [1] D. Whiting, R. Housley, N. Ferguson, *Counter with CBC-MAC (CCM)*. Available at <http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/>.
- [2] FIPS Publication 197, *Advanced Encryption Standard (AES)*. U.S. DoC/NIST, November 26, 2001. Available at <http://csrc.nist.gov/publications/>.
- [3] FIPS Publication 46-3, *Data Encryption Standard (DES)*. U.S. DoC/NIST, October 25, 1999. Available at <http://csrc.nist.gov/publications/>.
- [4] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.11i/D7.0, *Draft Amendment to Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Security Enhancements*, October 2003.
- [5] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.11-1999, *IEEE Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, New York: IEEE Press, 1999.
- [6] NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation – Methods and Techniques*. U.S. DoC/NIST, December 2001. Available at <http://csrc.nist.gov/publications/>.
- [7] Draft NIST Special Publication 800-38B, *Recommendation for Block Cipher Modes of Operation: the CMAC Authentication Mode*. U.S. DoC/NIST, October 2003. Available at <http://csrc.nist.gov/CryptoToolkit/modes>.
- [8] J. Jonsson, *On the Security of CTR + CBC-MAC*, in *Proceedings of Selected Areas in Cryptography – SAC, 2002*, K. Nyberg, H. Heys, Eds., *Lecture Notes in Computer Science*, Vol. 2595, pp. 76-93, Berlin: Springer, 2002.
- [9] P. Rogaway and D. Wagner, *A Critique of CCM*. *Cryptology ePrint Archive: Report 2003/070*, April, 2003. Available at <http://eprint.iacr.org/2003/070/>.