A Critique of CCM

P. Rogaway * D. Wagner †

2 February 2003

Abstract

CCM is a conventional authenticated-encryption scheme obtained from a 128-bit block cipher. The mechanism has been adopted as the mandatory encryption algorithm in an IEEE 802.11 draft standard [15], and its use has been proposed more broadly [16, 17]. In this note we point out a number of limitations of CCM. A related note provides an alternative to CCM [5].

1 Introduction

History. Authenticated encryption (AE) schemes are symmetric-key mechanisms by which a message M is a transformed into a ciphertext $\mathcal{C}$ in such a way that $\mathcal{C}$ protects both privacy *and* authenticity. Though AE schemes go back more than 20 years, only recently did AE get recognized as a distinct and significant cryptographic goal [3, 4, 10, 13]. Two factors seem to have triggered this. First was the realization that people had been doing rather poorly when they tried to glue together a traditional (privacy-only) encryption scheme and a message authentication code (MAC) [2, 3, 11]; second was the emergence of a class of "melded" AE schemes, beginning with [9], that did *not* work by gluing together an encryption scheme and a MAC.

One of these new AE schemes, an algorithm called OCB [14], was selected for a draft IEEE 802.11 standard for Wireless LANs. But there emerged opposition to OCB among some individuals who were active in the standards body. The opposition centered around the fact that OCB (and, more broadly, the new-generation of AE techniques) would have to be licensed. To avoid the authenticated-encryption IP, three of the 802.11 participants—Neils Ferguson, Russ Housley, and Doug Whiting—decided to invent their own algorithm for AE [15]. The mode they developed is called CCM. Though CCM makes about twice the number of block-cipher calls as OCB, there are many environments where this does not matter, and CCM is believed to be patent-free. Thus CCM came to replace OCB as the mandatory mechanism in the emerging IEEE 802.11 standard. CCM's inventors went on to propose their mode for applications beyond IEEE 802.11, providing it to the IETF [17] and NIST [16]. NIST has already signaled their inclination to move forward with a CCM-based recommendation [7].

This document. It is the authors' suspicion that acceptance of CCM is, in large part, a consequence of the fact that it has received little criticism and no counter-proposals. There is a need for a fully-specified AE scheme and a desire for cryptographic technology to be patent-unencumbered. CCM is the only mechanism having a writeup and satisfying those two constraints. The algorithm thus seems poised to emerge as a victor by default. We would find that unfortunate, because there are problems with CCM that ought to be addressed. The purpose of this document, then, is to explain what are the problems with CCM.

2 Definition of CCM Mode

Preliminaries. All strings in this note are over the binary alphabet, $\{0, 1\}$. For $\mathcal{L}$ a set of strings and $n \geq 0$ a number, we let $\mathcal{L}^n$ and $\mathcal{L}^*$ have their usual meanings. The concatenation of strings X and Y is denoted

* Department of Computer Science, University of California at Davis, Davis, California 95616, USA; and Department of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand. E-mail: rogaway@cs.ucdavis.edu WWW: www.cs.ucdavis.edu/~rogaway/

† Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, California 94720, USA. E-mail: daw@cs.berkeley.edu WWW: http://www.cs.berkeley.edu/~daw/

Algorithm $\mathrm{CBC}_K\ (M)$	**Algorithm** $\mathrm{CTR}_K^{\mathbf{N}}\ (M)$
Parse M into $M_1 \cdots M_m$ where $\lvert M_i\rvert = n$	$m \leftarrow \lceil \lvert M\rvert / n \rceil$
$C_0 \leftarrow 0^n$	$S \leftarrow E_K(\mathbf{N}) \parallel E_K(\mathbf{N}{+}1) \parallel \cdots \parallel E_K(\mathbf{N}{+}m{-}1)$
for $i \leftarrow 1$ **to** m **do**	$C \leftarrow M \oplus S$ [first $\lvert M\rvert$ bits]
$\quad C_i \leftarrow E_K(M_i \oplus C_{i-1})$	**return** C
return C_m	

Figure 1: Algorithms CBC and CTR, building blocks for this writeup. In both cases $E\colon \mathsf{Key} \times \{0,1\}^n \to \{0,1\}^n$ is fixed and $K \in \mathsf{Key}$. For CBC we have $M \in (\{0,1\}^n)^+$ and for CTR we have $M \in \{0,1\}^*$ and $S \in \{0,1\}^n$.

$X \parallel Y$ or simply $X\,Y$. The set $\textsc{Byte} = \{0,1\}^8$ contains all the strings of length 8, and a string $X \in \textsc{Byte}^*$ is called a *byte string* or an *octet string*. The string of length 0, called the *empty string*, is denoted ε. If $X \in \{0,1\}^*$ we let $\lvert X\rvert$ denote its length in bits, and if $X \in \textsc{Byte}^*$ we let $\lvert X\rvert_8 = \lvert X\rvert/8$ denote its length in bytes. For $\ell \geq 1$ a number, we write $\textsc{Byte}^{<\ell}$ for all byte string having fewer than ℓ bytes. If $X \in \{0,1\}^*$ and $\ell \leq \lvert X\rvert$ then the first ℓ bits of X are denoted X [first ℓ bits], while if $X \in \textsc{Byte}^*$ and $\ell \leq \lvert X\rvert_8$ then the first ℓ bytes of X are denoted X [first ℓ bytes]. When ℓ is a positive integer and $i \in [0..2^\ell - 1]$ we let $[\,i\,]_\ell$ be the ℓ-bit string that represents i in binary (most significant bit first). When $X \in \{0,1\}^\ell$ is a nonempty string and $i \in \mathsf{N}$ is a number we let $X + i$ be the ℓ-bit string which results from regarding X as a nonnegative number x (binary notation, most-significant-bit first), adding x to i, taking the result modulo 2^n, and converting this number back into an ℓ-bit string. String constants can be written in hexadecimal as in `0x00` for 0^8 and `0xFFFE` for $1^{15}0$. A *block cipher* is a function $E\colon \mathsf{Key} \times \{0,1\}^n \to \{0,1\}^n$ where Key is a finite, nonempty set and $n \geq 1$ is a number and $E_K(\cdot) = E(K,\cdot)$ is a permutation on $\{0,1\}^n$. The number n is called the *block length.* With E a block cipher as above, we describe two algorithms: the CBC MAC of message $M \in (\{0,1\}^n)^+$ under key $K \in \mathsf{Key}$; and CTR encryption of message $M \in \{0,1\}^*$ under key $K \in \mathsf{Key}$ and with IV $\mathbf{N} \in \{0,1\}^n$. These algorithms are defined in Figure 1.

We emphasize that CBC is insecure across messages of varying lengths; unless the message length is fixed, CBC should be viewed as a building-block for making a MAC and not by itself a MAC. We have also specified a raw form of CTR mode for which we are not asserting any particular security property.

CCM PARAMETERS. We are now ready to define the CCM mode of operation. The mode depends on three parameters—values that must be specified before the modes is well defined:

- E — the *block cipher* — where $E\colon \mathsf{Key} \times \{0,1\}^{128} \to \{0,1\}^{128}$
- $\boldsymbol{\tau}$ — the *tag length*[1]— where $\boldsymbol{\tau} \in \{4,6,8,10,12,14,16\}$
- λ — the *length-of-the-message-length-field* — where $\lambda \in \{2,3,4,5,6,7,8\}$

Once parameters $(E, \boldsymbol{\tau}, \lambda)$ have been fixed, where $E\colon \mathsf{Key} \times \{0,1\}^{128} \to \{0,1\}^{128}$ is a block cipher, CCM can be regarded as a pair (CCM.Encrypt, CCM.Decrypt) which is an authenticated-encryption with associated-data (AEAD) scheme, as defined in [13].[2] Encryption and decryption have the following signatures:

$$\begin{aligned} \text{CCM.Encrypt}\colon &\quad \mathsf{Key} \times \mathsf{Nonce} \times \mathsf{Header} \times \mathsf{Plaintext} \to \mathsf{Ciphertext} \\ \text{CCM.Decrypt}\colon &\quad \mathsf{Key} \times \mathsf{Nonce} \times \mathsf{Header} \times \mathsf{Ciphertext} \to \mathsf{Plaintext} \cup \{\textsc{Invalid}\} \end{aligned}$$

where

$$\mathsf{Nonce} = \textsc{Byte}^{15-\lambda} \qquad \mathsf{Header} = \textsc{Byte}^{<2^{64}} \qquad \mathsf{Plaintext} = \textsc{Byte}^{<2^{8\lambda}} \qquad \mathsf{Ciphertext} = \textsc{Byte}^*$$

Thus there is a parameterized tradeoff between the length of nonces, $\eta = \lvert N\rvert = 15 - \lambda$ bytes, and the length of the longest permitted message, $256^\lambda - 1$ bytes: a larger nonce space means a smaller message space.

[1] We have changed the name of this and several other variables compared to that which is in the WHF writeup [16]; our intent was to choose more conventional and sensible names. For those who have studied the WHF writeup [16], the following may be useful: their M (the tag length) is our $\boldsymbol{\tau}$; their L (the length-of-the-message-length-field) is our λ; their a (the associated data, or header) is our H; their m (the message being encrypted) is our M; their U (the authentication tag) is our T.

[2] We have made a slight change in syntax to that of [13], absorbing the key generation into the finite set Key which is part of the signature of the encryption and decryption routines.

Algorithm $\mathrm{CCM.Encrypt}_K^{N\,H}(M)$ // $N \in \mathrm{Byte}^{15-\lambda}$ and $H \in \mathrm{Byte}^{<2^{64}}$ and $M \in \mathrm{Byte}^{<2^{\lambda}}$

$B \leftarrow 0 \;\|\; \textbf{if } H = \varepsilon \textbf{ then } 0 \textbf{ else } 1 \textbf{ endif} \;\|\; [\,\boldsymbol{\tau}/2 - 1\,]_3 \;\|\; [\,\lambda - 1\,]_3 \;\|$
$\quad N \;\|\; [\,|M|_8\,]_{8\lambda} \;\|$
$\quad \textbf{if } H = \varepsilon \textbf{ then } \varepsilon \textbf{ elseif } |H|_8 < 62580 \textbf{ then } [\,|H|_8\,]_{16}$
$\quad \textbf{elseif } |H|_8 < 2^{32} \textbf{ then } \texttt{0xFFFE} \;\|\; [\,|H|_8\,]_{32} \textbf{ else } \texttt{0xFFFF} \;\|\; [\,|H|_8\,]_{64} \textbf{ endif} \;\|$
$\quad H \;\|$
$\quad \textbf{if } H = \varepsilon \textbf{ then } \varepsilon \textbf{ elseif } |H|_8 < 62580 \textbf{ then } \texttt{(0x00)}^{(14-|H|_8) \bmod 16}$
$\quad \textbf{elseif } |H|_8 < 2^{32} \textbf{ then } \texttt{(0x00)}^{(10-|H|_8) \bmod 16} \textbf{ else } \texttt{(0x00)}^{(6-|H|_8) \bmod 16} \textbf{ endif} \;\|$
$\quad M \;\|$
$\quad \texttt{(0x00)}^{(-|M|_8) \bmod 16}$
$U \leftarrow \mathrm{CBC}_K(B)$
$A_0 \leftarrow [\,\lambda - 1\,]_8 \;\|\; N \;\|\; \texttt{(0x00)}^{15-\lambda}$
$V \,\|\, C \leftarrow \mathrm{CTR}_K^{A_0}(U \,\|\, M)$ where $|V| = 128$
$T \leftarrow V$ [first $\boldsymbol{\tau}$ bytes]
return $\mathcal{C} \leftarrow C \,\|\, T$

Algorithm $\mathrm{CCM.Decrypt}_K^{N\,H}(\mathcal{C})$ // $N \in \mathrm{Byte}^{15-\lambda}$ and $H \in \mathrm{Byte}^{<2^{64}}$ and $\mathcal{C} \in \mathrm{Byte}^{*}$

if $|\mathcal{C}|_8 < \boldsymbol{\tau}$ **then return** Invalid
Partition $\mathcal{C}$ into $C \,\|\, T$ where $|T|_8 = \boldsymbol{\tau}$
if $|C|_8 > 2^{\lambda} - 1$ **then return** Invalid

$A_0 \leftarrow [\,\lambda - 1\,]_8 \;\|\; N \;\|\; \texttt{(0x00)}^{15-\lambda}$
$M \leftarrow \mathrm{CTR}_K^{A_0+1}(C)$

$B \leftarrow 0 \;\|\; \textbf{if } H = \varepsilon \textbf{ then } 0 \textbf{ else } 1 \textbf{ endif} \;\|\; [\,\boldsymbol{\tau}/2 - 1\,]_3 \;\|\; [\,\lambda - 1\,]_3 \;\|$
$\quad N \;\|\; [\,|M|_8\,]_{8\lambda} \;\|$
$\quad \textbf{if } H = \varepsilon \textbf{ then } \varepsilon \textbf{ elseif } |H|_8 < 62580 \textbf{ then } [\,|H|_8\,]_{16}$
$\quad \textbf{elseif } |H|_8 < 2^{32} \textbf{ then } \texttt{0xFFFE} \;\|\; [\,|H|_8\,]_{32} \textbf{ else } \texttt{0xFFFF} \;\|\; [\,|H|_8\,]_{64} \textbf{ endif} \;\|$
$\quad H \;\|$
$\quad \textbf{if } H = \varepsilon \textbf{ then } \varepsilon \textbf{ elseif } |H|_8 < 62580 \textbf{ then } \texttt{(0x00)}^{(14-|H|_8) \bmod 16}$
$\quad \textbf{elseif } |H|_8 < 2^{32} \textbf{ then } \texttt{(0x00)}^{(10-|H|_8) \bmod 16} \textbf{ else } \texttt{(0x00)}^{(6-|H|_8) \bmod 16} \textbf{ endif} \;\|$
$\quad M \;\|$
$\quad \texttt{(0x00)}^{(-|M|_8) \bmod 16}$
$U \leftarrow \mathrm{CBC}_K(B)$
$V \leftarrow E_K(A_0) \oplus U$
$T' \leftarrow V$ [first $\boldsymbol{\tau}$ bytes]
if $T \neq T'$ **then return** Invalid
return M

Figure 2: Encryption and decryption under $\mathrm{CCM}[E, \boldsymbol{\tau}, \lambda]$. The block cipher is $E\colon \mathrm{Key} \times \{0,1\}^{128} \to \{0,1\}^{128}$ and the tag length (in bytes) is $\boldsymbol{\tau} \in \{4, 6, 8, 10, 12, 14, 16\}$ and the length-of-the-message-length-field (in bytes) is $\lambda \in \{2, 3, 4, 5, 6, 7, 8\}$ and the nonce length (in bytes) is $\eta = 15 - \lambda$.

CCM ALGORITHM. Fix parameters $\boldsymbol{\tau}$, λ, and $E\colon \mathsf{Key} \times \{0,1\}^{128} \to \{0,1\}^{128}$ as indicated above. We write $\mathrm{CCM.Encrypt}_K^{N\,H}(M)$ instead of $\mathrm{CCM.Encrypt}(K, N, H, M)$. Similarly, we write $\mathrm{CCM.Decrypt}_K^{N\,H}(C)$ instead of $\mathrm{CCM.Decrypt}(K, N, H, C)$. Encryption and decryption under CCM works as specified in Figure 2. Note that the value returned by $\mathrm{CCM}_K^{N\,H}(M)$ is a byte string $\mathcal{C}$ having $|M|_{\mathrm{s}} + \boldsymbol{\tau}$ bytes.

3 Criticism of CCM

We partition our criticism into five categories: efficiency, parameterization, complexity, variable-tag-length subtleties, and some wrong security claims.

3.1 Efficiency Issues

We discuss three efficiency problems with CCM: (a) CCM is not on-line, (b) CCM disrupts word-alignment, and (c) CCM can't pre-process static associated data.

NOT ON-LINE. Here, an algorithm being *on-line* refers to its being able to process a stream of data as it arrives, with constant memory, not knowing in advance when the stream will end. Observe then that on-line methods should not require knowledge of the length of a message until the message is finished.

CCM fails to be on-line in both the plaintext and the associated data: one needs to know the length of both of these before one can proceed with encryption.

For message authentication codes, the significance of being on-line was brought out by the work of Petrank and Rackoff [12], whose work was motivated by the observation that the length-prepend CBC MAC (and other suggestions appearing in [1]) were not on-line. Since their paper, a failure to be on-line has been regarded as a significant defect for an encryption scheme or a MAC.

Now it is true that in many contexts where one would be encrypting a string one *does* know the length of the string in advance. For example, many protocols will already have "packaged up" the string length at a lower level. In effect, such strings have been represented in the computing system as sequence of bytes and a count of those bytes. But there are also contexts where one does *not* know the length of a message in advance of getting an indication that it is over. For examples, a printable string is often represented in computer systems as a sequence of non-zero bytes followed by a terminal zero-byte. Certainly one should be able to efficiently encrypt a string which has been represented in this way.

DISRUPTS WORD-ALIGNMENT. Length-prepend annotation causes an additional problem for the associated data (also called the header)—namely, CCM disrupts its word-alignment. This is a problem when the associated data, H, is long. To understand this issue, remember that most modern machines perform operations much more efficiently when pointers into memory fall along word-boundaries (which typically occur every 4 or 8 bytes). A typical software implementation of a CBC MAC, for example, will exhibit much worse performance if it is called on an argument which is not word-aligned. By prepending length-annotation to the associated data H, this length-annotation *not* being a multiple of 16 or even 4 bytes, one can expect that typical implementations will suffer a big performance hit.

The disruption of word-alignment is not a big concern if the associated data is just a few bytes, as we expect that it often will be. But, again, NIST and others are considering CCM for use as a general-purpose AEAD algorithm. We have no idea how long will be the associated data. For all we know, the user is primarily interested in authenticating traffic, is doing this to a large volume of traffic, and is encrypting nothing or almost nothing. We don't want the authentication side of an AEAD scheme to be significantly more costly than using a dedicated CBC MAC.

CAN'T PRE-PROCESS STATIC AD. In many scenarios the associated data H will be static over the course of a communications session. For example, the associated data may including information such as the IP address of the sender, the receiver, and fixed cryptographic parameters associated to this session. In such a case one would like that the amount of time to compute $\mathrm{Encrypt}_K^{N\,H}(M)$ and $\mathrm{Decrypt}_K^{N\,H}(C)$ should be independent of $|H|$ (disregarding the work done in a preprocessing step). The significance of this goal was already explained in [13], and a simple approach for achieving this goal was given there. Basically, the reason

λ	2	3	4	5	6	7	8
msgs shorter than	64 KBytes	16 MBytes	8 GBytes	2^{40} bytes	2^{48} bytes	2^{56} bytes	2^{64} bytes
$\eta = 15 - \lambda$	13	12	11	10	9	8	7
8η (nonce length)	104 bits	96bits	88 bits	80 bits	72 bits	64 bits	56 bits

Figure 3: CCM-allowed values of the length-of-the-message-length-field, λ, the corresponding bound on message lengths, and the resulting length of nonces $\eta = |N|_8$ that one must use, measured in bytes and then in bits.

that CCM fails to allow pre-processing of associated data H is that the algorithm encodes the nonce N and the message length $|M|_8$ before H rather than after it.

3.2 Parameterization

We criticize a few aspects of CCM related to its parameterization: (a) the requirement for the user to specify a length-of-the-message-length-field parameter; (b) the fact that this choice involves a trade-off against a conceptually unrelated quantity, the nonce length; (c) the marginal utility of the mode for random nonces; (d) the mode's strict byte orientation.

An inappropriate parameter. The user of CCM is presented with a parameter, the-length-of-the-message-length-field, that she really has no business seeing. This parameter can be regarded as a surrogate for the maximum message length. While it is reasonable to fix a suitably large maximum message length, such as $2^{64} - 1$ bytes, it seems undesirable to force the user to think about choosing smaller message spaces. The very name of this parameter makes clear that this was always conceived of as an implementation-oriented parameter and not a fundamental characteristic of an AEAD scheme.

Tradeoff between nonce length and maximum message length. Worse than the fact that the user must choose a parameter value that she shouldn't have to think about is that the definition of CCM involves a tradeoff between two things that are conceptually unrelated: the maximal message length and the length of the scheme's nonce. The tradeoff is summarized in Figure 3, which shows the seven allowed values of the-length-of-the-message-length-field parameter, the message space one gets as a result (restricted to octet string), and the corresponding value that is mandated for the the nonce length.

There seem to us several reasonable choices for what should be the nonce length of an AEAD scheme that is based on a 128-bit block cipher. (i) One reasonable choice is 64 bits. This value is large enough to handle a counter, as no real application will encrypt as many as 2^{64} messages. (ii) Another reasonable choice is 128 bits. This choice is natural for matching the block length and being the right length for handling a random value for the nonce. (iii) A third natural choice is any string (or any byte string). This is conceptually clean, and it might be convenient, for example, to allow nonces that are initially 1 byte, and then grow to 2 bytes, and so forth.

One thing that does not make sense to us is to say that the nonce length is a number of bytes that is 15 minus the log base two of one more than the size of longest permissible message length. The message space and the nonce space have nothing to do with each other.

Marginal utility with random nonces. The fact that CCM nonce lengths are allowed to exceed 64 bits suggests that its inventors are thinking of random values as possible choices for the nonce. Though the maximal nonce length (104 bits) may be acceptable for a random nonce, the minimal nonce length (56 bits) probably is not. In general, the suitability of CCM for random nonces is linked to the length of the longest message one wishes to be able to handle, an unnecessary and undesirable connection.

Byte orientation. Though some would view this criticism as strictly a matter of taste, we ourselves do not like that CCM is only defined on octet strings. It is not that one is all that likely to need to use an AEAD scheme on strings that are not octet strings. It is more that cryptographic algorithms reach beyond technological conventions like the prevalence byte-orientation in computing systems. To put things in perspective: most cryptographers would have viewed it as a poor choice if MD4, MD5, and SHA1 had

only been defined on octet strings. It is no less a defect if a general-purpose AEAD scheme is only defined on octet strings.[3]

3.3 Complexity

While human-perceived complexity is inherently subjective, we ourselves find CCM to be complex. The authors of the current document can not even remember the definition of CCM without consulting its defining document. To us, a mode of operation with so many details that one cannot easily remember it (even after working with it for a few days) is off to a bad start.

Bit manipulations. We see two underlying causes for this complexity. The first is all the bit twiddling that CCM does. We believe that it is preferable for a mode of operation to avoid bit manipulations beyond standard padding or length annotation. The central concern for a mode of operation is the correct and efficient use of the block cipher. That purpose has never been shown to need tricky ways to package up arguments or encode string lengths.

Missing abstraction boundary. More fundamentally, we see the complexity of CCM as stemming from the fact that it is designed directly on top of a block cipher—in particular, it was not designed to use any particular message authentication code. In our own exposition of CCM we have done our best to "push upwards" the abstraction boundary to which CCM writes, so that we could "call out" to the raw CBC MAC and CTR encryption. Doing this is at odds with the defining document [16], but CCM's authors have expressed the viewpoint that they are combining such primitives, and so one rather expects a description of CCM in those terms. But the raw CBC MAC is not a secure MAC and the authentication tag that CCM computes can only be seen as something computed by a process integral to the entire mode. There's no sense in which one can say "do CCM using this other message authentication code." CCM employs no autonomous MAC.

It is our view that an AEAD scheme should be designed on top of higher-level primitives than a block cipher. Even if the higher-level primitives are to be implemented using a block cipher, the abstraction boundary helps the scheme to be conceptually clean and support a convincing security analysis. The complexity that one is concerned with isn't the number of lines to implement the mode (which is certainly small) or write it down in pseudocode. It is the conceptual complexity of an algorithm as induced by the distance between it and what it sits on top of.

Illustration 1. A glimpse of CCM's complexity can be seen from the fact that correctness crucially depends on the encoding convention: namely, the authors have excluded the possibility of $\boldsymbol{\tau} = 2$ (i.e., two-byte tags), which means that at least one of bits 3, 4, or 5 of the first block of B, which holds $[\,\boldsymbol{\tau}/2 - 1\,]_3$ is non-zero, while these bits are always zero in the initial counter value A_0. If one allowed a tag of $\boldsymbol{\tau} = 2$ bytes[4] or if one had encoded $\boldsymbol{\tau}$ as $[\,\boldsymbol{\tau}/2 - 2\,]_3$ instead of $[\,\boldsymbol{\tau}/2 - 1\,]_3$ the CCM method would be wrong.[5]

Illustration 2. A final way to get at the complexity of CCM is to precisely answer the question, *how many block cipher calls does CCM use?* The answer is given by the following expression:

$$\mathit{NumCallsCCM}(M, H) \quad = \quad 2\left\lceil \frac{|M|}{128} \right\rceil + \left\lceil \frac{|H|}{128} \right\rceil + 2 + \delta(|H|)$$

[3] Nothing in this paragraph should be understood to suggest that an implementation is under any compulsion to implement an AEAD scheme that operates on arbitrary bit strings (no more than people implement SHA1 to operate on arbitrary bit strings). We are simply saying that it should be well-defined.

[4] We comment that allowing tags of one byte, or even one bit, is a reasonable thing to do, as there are contexts, like authenticating a video frame, where one has to forge many messages to have a detrimental effect.

[5] Criticism that an algorithm "would be wrong if the following change was made" is never-ending and inherently unconvincing—"but the algorithm *isn't* that way" is a quite sufficient response. Here the criticism is simply being used to emphasize that CCM's correctness is integrally wrapped up in its encoding-scheme details.

where $\delta(i) \in \{0,1\}$ is defined as follows: letting

$$\lambda(i) \quad = \quad \begin{cases} 0 & \text{if } i = 0 \\ 16 & \text{if } i \in [8, 8 \cdot 62572] \\ 48 & \text{if } i \in [8 \cdot 62580, 2^{35} - 8] \\ 80 & \text{if } i \in [2^{35}, 2^{64} - 8] \end{cases}$$

we set $\delta(i) = 0$ if $(i \bmod 128) + \ell(i) \leq 128$, and $\delta(i) = 1$ otherwise. It appears that much of this complexity is so that $\delta(|H|)$ will be more often 0 than 1.

3.4 Subtleties of Variable-Length Authentication Tags

In CCM, the authentication tag $\boldsymbol{\tau}$ is of variable length: it is permitted to be 4, 6, 8, 10, 12, 14, or 16 bytes long. Variable-length tags come with some security risks, if the schemes are not implemented carefully or what they achieve is not stated clearly.

A scenario and an attack. In the design of Internet protocols, a common slogan is "be conservative in what you send, and liberal in what you accept." Imagine a CCM implementation takes this literally: the sender always creates messages with a 16-byte tag, but the receiver accepts messages if they have a valid tag of any permitted length.

How secure would such an implementation be? Of course, since the attacker is free to choose a tag of any length, a smart attacker will choose the tag length that is most convenient for him: presumably, 4-byte tags. Clearly such an attacker can generate a valid ciphertext within 2^{32} tries. This vulnerability is unavoidable in any scheme with authentication tags that are only four bytes long. However, this attack might be of limited value to the attacker, because it is a *blind* forgery: the attacker cannot control what message will be accepted by the recipient.

We point out that a worse attack on CCM is possible in the envisaged scenario: a more clever attacker can fully control *what* message the recipient will be tricked into accepting. The reason for this is that, in CCM, the transmitted ciphertext has the form $C \parallel T$ where T is an authentication tag and where the received message M is computed as a function of C but not $\boldsymbol{\tau}$. The *directed* forgery attack on CCM is as follows. Suppose the attacker intercepts a single ciphertext $C \parallel T_{16}$ that is the encryption of some message M formed by the legitimate sender. Imagine that the attacker has a difference Δ that it would like to xor into the message; for instance, the attacker might want to flip certain bit positions in M. Then the attacker can generate the 2^{32} ciphertexts of the form $(C \oplus \Delta) \parallel T_4$ where T_4 varies over all 4-byte values. Most of these will be rejected by the receiver as having invalid authentication tags, but one will be accepted as a valid encryption of the modified message $M \oplus \Delta$. Thus an adversary can forge any message it likes with 2^{32} tries, given a single ciphertext (with known plaintext) that was authenticated with a 128-bit tag. One can reasonably maintain that authenticating a message with a 128-bit tag should not have had this consequence.

Interpretation. Does the attack above mean that the tag length $\boldsymbol{\tau}$ should have been used for computing the ciphertext core C? In our opinion, the answer is: "not necessarily." Rather, the attack highlights that the specification [16] has made it unclear what the goal is with respect handling a multiplicity of tag lengths.

We suggest that the tag length parameter $\boldsymbol{\tau}$ should be fixed at key-negotiation time, bound securely to the key, and negotiated authentically between both parties. Once a session has begun, there should be only a single value $\boldsymbol{\tau}$ that will be accepted by the receiver, and this should remain unchanged throughout the lifetime of the session. The receiver shouldn't accept a new $\boldsymbol{\tau}$ in the middle of a session any more than it would accept a new block cipher E. All parameters should have the same status. Under this interpretation the inclusion of $\boldsymbol{\tau}$ within the string B in CCM was not necessary to achieve the security goals. This is not a flaw in CCM, but it underscores the need to think carefully about the desired security goals.

3.5 Meaningless Security Claims

Jonsson has done an admirable job of finding an abstraction of CCM that permits a security proof, and going ahead and giving such a proof [8]. Though neither of us have studied the proof in detail, it seems credible and well-conceived. This is fortunate, because many of the security comments in the WHF writeup

itself [16] can only be described as uninformed. For example, in Section 1.8 the authors claim that CCM "is secure against attackers limited to 2^{128} steps of operation if the key K is 256 bits or longer." Such a claim is unsupported by any known results and would seem to be wrong under any reasonable interpretation, as privacy itself vanishes by the time that 2^{64} blocks have been enciphered. Later, in Section 1.10 [16], we hear that "[by enciphering the CBC MAC] we avoid CBC-MAC collision attacks. If the block cipher behaves as a pseudo-random permutation, then the key stream is indistinguishable from a random string. Thus the attacker gets no information about the CBC-MAC result. The only avenue of attack that is left is differential-style attack, which has no significant chance of success if the block cipher is a pseudo-random permutation." This paragraph is so far from saying something technically accurate that we wouldn't know where to begin. Of course wrong or unscientific security claims are not an indictment of the method they speak about; our only point is that one needs to ignore the security statements of the WHF writeups [15–17], regard it only as an algorithm specification, and turn to Jonsson [8] for more scientific assertions.

4 Conclusion

We feel that CCM is not the best choice for a general-purpose standard. Although we have identified no grave or urgent problems with the mode, we think that one can do better. In a companion document [5] we present an alternative algorithm, called EAX. That mode retains the major attributes of CCM, but without the disadvantages discussed here.

5 Acknowledgments

Phil Rogaway's work was funded by NSF CCR-0208842 and a gift from CISCO Systems. David Wagner's work was funded by NSF CCR-0113941.

References

[1] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *J. of Computer and System Science*, vol. 61, no. 3, pp. 362–399, December 2000. Earlier version in *Advances in Cryptology—Crypto '94*, Lecture Notes in Computer Science, Vol. 839. Y. Desmedt, ed., Springer-Verlag, 1994. `www.cs.ucdavis.edu/∼rogaway/`

[2] M. Bellare, T. Kohno, and C. Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. *Proceedings of the 9th ACM conference on Computer and Communications Security* (CCS-02), ACM Press, 2002.

[3] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology – ASIACRYPT '00*. Lecture Notes in Computer Science, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. `www-cse.ucsd.edu/users/mihir/`

[4] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient encryption. *Advances in Cryptology – ASIACRYPT '00*. Lecture Notes in Computer Science, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. `www.cs.ucdavis.edu/∼rogaway/`

[5] M. Bellare, P. Rogaway, and D. Wagner. A conventional authenticated-encryption mode. Manuscript, April 2003. `www.cs.ucdavis.edu/∼rogaway`

[6] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. *Advances in Cryptology—Crypto 2000*, Lecture Notes in Computer Science, vol. 1880, Springer-Verlag, Mihir Bellare, editor, pp. 197–215, 2000. `www.cs.ucdavis.edu/~rogaway`

[7] W. Burr (attributed). Oral presentation at an IETF CFRG meeting in Atlanta, Georgia, USA. 19 November 2002. From email with subject heading: "Final minutes for CFRG meeting in Atlanta," scribed by B. Weiss and posted to the cfrg@ietf.org mailing list, 4 December 2002. CFRG mail archive at `www.irtf.org/cfrg`

[8] J. Jonsson. On the security of CTR + CBC-MAC. Contribution to NIST. Available from `csrc.nist.gov/encryption/modes/proposedmodes/`. Proceedings version to appearing in *Proceedings from Selected Areas of Cryptography (SAC) 2002*.

[9] C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology – EUROCRYPT 2001*. Lecture Notes in Computer Science, vol. 2045, Springer-Verlag, 2001. Earlier version in Cryptology ePrint archive, reference number 2000/039, August 1, 2000, `eprint.iacr.org/`

[10] J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption '00*. Lecture Notes in Computer Science, B. Schneier, ed., 2000.

[11] H. Krawczyk. The order of encryption and authentication for protecting communications (or: how Secure is SSL?). Advances in Cryptology – CRYPTO '01, Lecture Notes in Computer Science, vol. 2139, Springer-Verlag, 2001. See Cryptology ePrint Report 2001/045, `eprint.iacr.org`

[12] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *Journal of Cryptology*, Vol. 13, No. 3 pp. 315–338, 2000.

[13] P. Rogaway. Authenticated-encryption with associated-data. *Ninth ACM Conference on Computer and Communications Security* (CCS-9). ACM Press, 2002. `www.cs.ucdavis.edu/~rogaway`

[14] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. *Eighth ACM Conference on Computer and Communications Security* (CCS-8). ACM Press, 2001. `www.cs.ucdavis.edu/~rogaway`

[15] D. Whiting, R. Housley, and N. Ferguson. AES Encryption & Authentication Using CTR Mode & CBC-MAC. IEEE P802.11 doc 02/001r2, May 2002.

[16] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). June 2002. Available from `csrc.nist.gov/encryption/modes/proposedmodes/`

[17] D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). IETF Internet Draft with file name draft-housley-ccm-mode-01.txt, September 2002. Available from `www.ietf.org/internet-drafts/draft-housley-ccm-mode-01.txt`