Submission to NIST. This is a shorted version of [7].

The CWC Authenticated Encryption (Associated Data) Mode

Tadayoshi Kohno
UC San Diego
9500 Gilman Drive, MC 0114
La Jolla, CA 92093
tkohno@cs.ucsd.edu

John Viega
Virginia Tech
6066 Leesburg Pike, Suite 500
Falls Church, VA 22041
viega@securesoftware.com

Doug Whiting
Hifn, Inc.
5973 Avenida Encinas, Suite 110
Carlsbad, CA 92009
dwhiting@hifn.com

May 27, 2003

Abstract

We introduce CWC, a new block cipher mode of operation designed to protect both the privacy and the authenticity of encapsulated data. Important properties of CWC include:

1. **Performance.** CWC is **parallelizable** and is **efficient** in both hardware and software.
2. **Security.** CWC is **provably secure** and its provable security depends only on the pseudorandomness of the underlying block cipher. No other cryptographic primitives are used and no other assumptions are made.
3. **Patent-free.** To the best of our knowledge CWC is **not covered by any patents**.

CWC is currently the only dedicated authenticated encryption with associated data (AEAD) scheme that simultaneously has these three properties (e.g., CCM and EAX are not parallelizable and OCB is not patent-free). Having all three of these properties makes CWC a strong candidate for use with future high-performance systems.

1 Introduction

There has recently been significant interest in developing block cipher modes of operation capable of simultaneously protecting both the *privacy* and the *authenticity/integrity* of encapsulated data. Such modes of operation are often called *authenticated encryption (AE) schemes* or, if the schemes are capable of authenticating more data than they encrypt, *authenticated encryption with associated data (AEAD) schemes*.

In this work we propose a dedicated AEAD scheme that is **patent-free**, **provably-secure**, **parallelizable** and **efficient** in both hardware and software. To our knowledge, ours is the only dedicated AEAD scheme that simultaneously has all of these properties. (Our construction also has other desirable properties. For example, it is clean and simple (in our opinion), on-line, uses a single block cipher key, and allows for pre-processing of associated data or other header fields.)

Our general construction, **CWC**, is based on what is called "the Encrypt-then-Authenticate generic composition paradigm." In particular, CWC essentially combines a Carter-Wegman message authentication scheme [14] with CTR mode encryption in an Encrypt-then-Authenticate manner. The general idea is as follows: given a pair of strings (A, M) and a nonce N as input, the CWC encapsulation algorithm encrypts M with CTR mode to get some intermediate ciphertext σ. It then uses a Carter-Wegman MAC and the nonce N to MAC the pair (A, σ). If we let τ denote the resulting MAC tag, then the output of the CWC encapsulation algorithm is the concatenation of σ and τ. CWC is designed to protect the privacy of M and the integrity of both A and M. Although based on the Encrypt-then-Authenticate generic composition paradigm, CWC is not a generic composition construction; for example, the CWC encryption and MAC components share the same block cipher key. This means, among other things, that we had to prove the security of CWC directly, rather than invoke previous results about the generic composition paradigm.

Let us begin by discussing some of the motivations for CWC.

Why do we want dedicated authenticated encryption schemes? The traditional approach to achieving authenticated encryption is to combine some standard encryption scheme (e.g., CBC mode) with some standard message authentication scheme (e.g., HMAC). This is known as the *generic-composition* approach and was first explored in [1] and [8]. Unfortunately, such generic-composition constructions are often *ad hoc* and, as illustrated in [1] and [8], it is very easy to accidentally combine secure encryption schemes with secure MACs and still get insecure authenticated encryption schemes.

One of the biggest advantages of dedicated AEAD schemes over generic-composition AEAD schemes is that dedicated AEAD schemes are not prone to such accidental errors. That is, since dedicated AEAD schemes clearly specify how to achieve both privacy and authenticity, there is no longer the risk of someone accidentally combing a privacy/encryption component with an authenticity/MAC component in an insecure fashion. Furthermore, since most applications that require privacy also require integrity, it is logical to focus on tools capable of providing both services simultaneously. There is thus great value in developing and standardizing dedicated AEAD schemes, as evidenced by a wealth of papers in this area [2, 4, 5, 6, 11, 12, 15].

Patents. Pragmatically, patents are a major impediment to the standardization and wide-spread deployment of some of the modes presented in the above-mentioned papers. In particular, three independent parties have applied for patents on single-pass authenticated encryption schemes. It is not our purpose to describe the specifics of these patent applications (and, indeed, the specifics are not completely known to the public). Rather, we point out that the existence of these patent applications makes many existing authenticated encryption modes less attractive, and therefore less amenable to standardization and deployment. To exemplify this point, we note that although

Rogaway, Bellare, Black, and Krovetz's OCB mode [12] is very efficient and elegant, it was apparently rejected from the IEEE 802.11 wireless working group largely because of the fact that it was covered by patent applications from multiple parties.

WHAT IS NEEDED? Noting the need for patent-free dedicated AEAD schemes, Whiting, Ferguson, and Housley proposed a patent-free AEAD scheme called CCM [15] which, apparently because of its patent-free nature, has been adopted by the IEEE 802.11 working group. CCM was recently followed by another construction, called EAX, by Bellare, Rogaway, and Wagner [2]. Since CCM and EAX are based on the generic-composition approach (they both essentially combine standard CTR mode encryption with variants of CBC-MAC message authentication), CCM and EAX do not fall under the aforementioned patent applications.

There is, however, one significant disadvantage with both CCM and EAX: the CCM and EAX encryption and decryption operations are not parallelizable. That is, although the CTR mode portions of CCM and EAX are clearly parallelizable, their CBC-MAC portions are not. Parallelizability is, however, very important. For example, without the ability to parallelize the encryption process, using current technology it does not seem possible to build a single hardware engine for CCM or EAX capable of encrypting beyond approximately 2 Gbps.[1] Although 2 Gbps might be adequate for today's applications, such speeds will not be adequate for the coming 10 Gbps network devices.

Therefore, there is a need for a patent-free dedicated mode of operation capable of encrypting and authenticating data at 10 Gbps.

THE CWC SOLUTION. We propose a general paradigm, called CWC, that addresses all the aforementioned issues. In particular, our preferred instantiation of CWC for 128-bit block ciphers is un-patented, provably-secure, parallelizable, and efficient in both hardware and software. The parallelizability enables high-speed CWC hardware implementations to encrypt at 10 Gbps when using AES.

Throughout the body of this paper we will focus on our instantiation of the CWC paradigm for 128-bit block ciphers.[2] In particular, we focus on CWC-AES-kl, a CWC instantiation with AES-kl as the underlying block cipher (here AES-kl denotes kl-bit AES, where the key length $\mathsf{kl} \in \{128, 192, 256\}$). When our results apply to AES with all key lengths, we shall simply refer to CWC-AES. Instead of writing CWC-AES-kl for some appropriate kl, we shall write CWC-BC or simply CWC when we mean the general CWC paradigm instantiated like CWC-AES-kl but with any 128-bit block cipher BC in place of AES-kl.

Note the difference in font between CWC, the general paradigm, and CWC, our specific proposal.

ACHIEVING PARALLELISM. Clearly the CTR mode portion of CWC is parallelizable. Furthermore, the core of the Carter-Wegman MAC portion of CWC (a.k.a. the universal hashing portion of CWC) can be made parallelizable. In the case of CWC, the universal hashing step works by computing

$$Y_1 x^n + Y_2 x^{n-1} + Y_3 x^{n-2} + Y_4 x^{n-3} + \cdots + Y_n x + Y_{n+1} \bmod 2^{127} - 1 \; .$$

where $Y_1, \ldots, Y_{n+1}$ are 96-bit integers[3] corresponding to the pair (A, σ) and x is an integer modulo the prime $2^{127} - 1$. It is well-known that the computation of this polynomial is parallelizable. For

[1] It is always possible to build two totally independent units and process two packets at a time, but this is dramatically more complex, requiring twice the area, plus a load balancer.

[2] If desired, it is possible to instantiate the general CWC paradigm with 64-bit block ciphers, although certain limitations (e.g., nonce size) apply to such variants. We do not present a 64-bit CWC variant here since we are primarily concerned with new, high-speed systems using AES, not legacy applications.

[3] Actually, Y_{n+1} may be more than 96-bits long, but we ignore that detail here.

example, if we have two processors available, we can rewrite the above polynomial as

$$\Big(Y_1 y^m + Y_3 y^{m-1} + \cdots + Y_n\Big)x + \Big(Y_2 y^m + Y_4 y^{m-1} + \cdots + Y_{n+1}\Big) \bmod 2^{127} - 1 \ ,$$

where $y = x^2 \bmod 2^{127} - 1$, $m = (n-1)/2$, and we assume for illustrative purposes that n is odd. We can then compute both the left and the right portions of the above in parallel. Additional parallelism can be achieved by further splitting the original polynomial into j polynomials in $y' = x^j \bmod 2^{127} - 1$.

SINGLE KEY. The CWC paradigm uses a single block cipher key K. The key K is used in all applications of the underlying block cipher and is used to derive a subkey K_h for use in CWC's universal hashing step. In the case of our CWC instantiation, deriving K_h requires one block cipher invocation. The main advantage with deriving subkey K_h from K is that it simplifies key management and reduces the costs associated with fetching key material in hardware, which can be a bottleneck.

PERFORMANCE. Let (A, M) be some input to the CWC encapsulation algorithm (recall that A is the associated data and M is the message to encrypt). Assuming that the universal hashing subkey is maintained across invocations, encapsulating (A, M) takes $\lceil |M|/128 \rceil + 2$ block cipher invocations. The polynomial used in CWC's universal hashing step will have degree $d = \lceil |A|/96 \rceil + \lceil |M|/96 \rceil$. There are several ways to evaluate this polynomial (details in [7]). For example, assuming no precomputation, we could evaluate this polynomial using d 128-by-128-bit multiplies. As another example, assuming n precomputed powers of the hash subkey, which are cheap to maintain in software for reasonable n, we could evaluate the polynomial using $d - m$ 96-by-128-bit multiplies and m 128-by-128-bit multiplies, where $m = \lceil (d+1)/n \rceil - 1$.

As noted before, it is possible to implement CWC-AES in hardware at 10 Gbps using conventional ASIC technology. Specifically, at 0.13 micron, it should take about 300 Kgates to reach 10 Gbps throughput. In software, experimental results show that the current best implementation of CWC-AES-128 on a Pentium III (due to Brian Gladman) runs significantly faster than CCM and EAX with 128-bit AES and implemented with popular crypto libraries. Using the precomputation approach from Bernstein [3], we anticipate reducing the cost of CWC's universal hashing step to around 8 cpb, thereby significantly improving the performance of CWC-AES in software.

PROVABLE SECURITY. CWC is a provably-secure AEAD scheme assuming that the underlying block cipher is a secure pseudorandom function or pseudorandom permutation. Consequently, if we believe AES to be a secure pseudorandom permutation (which is a widely-held belief), then CWC-AES is secure. For our proofs of security, we use Rogaway's AEAD notions from [11]. In our provable security results we clearly show that the same block cipher key can be used in CWC's CTR mode portion, in the generation of the hash subkey K_h, and in the block cipher applications used within CWC's message authentication portion.

1.1 Background and related work

The notion of an *authenticated encryption (AE) scheme* was formalized in Katz–Yung [6] and Bellare–Namprempre [1] and the notion of an *authenticated encryption with associated data (AEAD) scheme* was formalized in Rogaway [11]. In [1, 8], Bellare–Namprempre and Krawczyk explored ways to combine standard encryption schemes with MACs to achieve authenticated encryption. A number of dedicated AE and AEAD schemes also exist, including RPC [6], XCBC [4], IACBC [5], OCB [12], CCM [15], and EAX [2]. Within the scope of dedicated block cipher-based AEAD schemes, CWC's closest relatives are CCM and EAX, which also use two passes and are unpatented. From a broader perspective, CWC is similar to the combination of McGrew's UST [10]

and TMMH [9], where one of the main advantages of CWC over UST+TMMH is CWC's small key size, which can be a bottleneck for UST+TMMH in hardware at high speeds.

Rogaway and Wagner recently released a critique of CCM [13]. For each issue raised in [13], we find that we have already addressed the issue (e.g., we designed CWC to be on-line) or we disagree with the issue (e.g., we feel that it is sufficient for new modes of operation to handle arbitrary octet-length, as opposed to arbitrary bit-length, messages[4]).

The integrity portion of CWC builds on top of the Carter-Wegman universal hashing approach to message authentication [14]. Like Bernstein's hash127 [3], CWC's universal hash function evaluates a polynomial over the integers modulo the prime $2^{127} - 1$. The main difference between hash127 and the CWC universal hash function is that hash127 uses signed 32-bit coefficients and CWC uses unsigned 96-bit coefficients.

In April 2003 we introduced an Internet-Draft, within the IRTF Crypto Forum Research Group, specifying the CWC-AES mode of operation. The latest version of the Internet-Draft can be found at `http://www.zork.org/cwc` or on the IETF website `http://www.ietf.org`.

1.2 Outline

We begin in Section 2 with some preliminaries and then describe the CWC mode of operation in Section 3. In Section 4 we present our formal statements of security for CWC. Appendix A presents a summary of CWC's properties. Appendix B contains our intellectual property statement. Appendix C contains test vectors.

VERSIONS. In this "submission to NIST" we specify the CWC mode of operation. This is an abbreviated version of [7], which contains security proofs for CWC, further notes on some of the design decisions we made, and more detailed implementation and performance discussions.

2 Preliminaries

NOTATION. If x is a string then $|x|$ denotes its length in *bits* (not octets). Let ε denote the empty string. If x and y are two equal-length strings, then $x \oplus y$ denotes the XOR of x and y. If X and Y are sets, then $\mathsf{Func}(X, Y)$ denotes the set of all functions from X to Y and $\mathsf{Perm}(X)$ denotes the set of all permutations on X. If l and L are positive integers, then $\mathsf{Func}(l, L)$ denotes the set of all functions from $\{0,1\}^l$ to $\{0,1\}^L$ and $\mathsf{Perm}(L)$ denotes the set of all permutations on $\{0,1\}^L$.

If N is a non-negative integer and l is an integer such that $0 \leq N < 2^l$, then $\mathsf{tostr}(N, l)$ denotes the encoding of N as an l-bit string in big-endian format. If x is a string, then $\mathsf{toint}(x)$ denotes the integer corresponding to string x in big-endian format (the most significant bit is *not* interpreted as a sign bit). For example, $\mathsf{toint}(10000010) = 2^7 + 2 = 130$.

Let $x \leftarrow y$ denote the assignment of y to x. If X is a set, let $x \xleftarrow{\$} X$ denote the process of uniformly selecting at random an element from X and assigning it to x. If f is a randomized algorithm, let $x \xleftarrow{\$} f(y)$ denote the process of running f with input y and a uniformly selected random tape.

When we refer to the time of an algorithm or experiment, we include the size of the code (in some fixed encoding). There is also an implicit big-$\mathcal{O}$ surrounding all time references.

AUTHENTICATED ENCRYPTION SCHEMES WITH ASSOCIATED DATA. We use Rogaway's notion of an *authenticated encryption with associated data (AEAD) scheme* [11].

[4] Although we stress that, if desired, it is easy to modify CWC to handle arbitrary bit-length messages. See [7].

An AEAD scheme $\mathcal{SE} = (\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ consists of three algorithms and is defined over some key space $\mathsf{KeySp}_{\mathcal{SE}}$, some nonce space $\mathsf{NonceSp}_{\mathcal{SE}} = \{0,1\}^n$, n a positive integer, some associated data (header) space $\mathsf{AdSp}_{\mathcal{SE}} \subseteq \{0,1\}^*$, and some payload message space $\mathsf{MsgSp}_{\mathcal{SE}} \subseteq \{0,1\}^*$. We require that membership in $\mathsf{MsgSp}_{\mathcal{SE}}$ and $\mathsf{AdSp}_{\mathcal{SE}}$ can be efficiently tested and that if M, M' are two strings such that $M \in \mathsf{MsgSp}_{\mathcal{SE}}$ and $|M'| = |M|$, then $M' \in \mathsf{MsgSp}_{\mathcal{SE}}$.

The randomized key generation algorithm $\mathcal{K}_e$ returns a key $K \in \mathsf{KeySp}_{\mathcal{SE}}$; we denote this process as $K \xleftarrow{\$} \mathcal{K}_e$. The deterministic encryption algorithm $\mathcal{E}$ takes as input a key $K \in \mathsf{KeySp}_{\mathcal{SE}}$, a nonce $N \in \mathsf{NonceSp}_{\mathcal{SE}}$, a header (or associated data) $A \in \mathsf{AdSp}_{\mathcal{SE}}$, and a payload message $M \in \mathsf{MsgSp}_{\mathcal{SE}}$, and returns a ciphertext $C \in \{0,1\}^*$; we denote this process as $C \leftarrow \mathcal{E}_K^{N,A}(M)$ or $C \leftarrow \mathcal{E}_K(N, A, M)$. The deterministic decryption algorithm $\mathcal{D}$ takes as input a key $K \in \mathsf{KeySp}_{\mathcal{SE}}$, a nonce $N \in \mathsf{NonceSp}_{\mathcal{SE}}$, a header $A \in \mathsf{AdSp}_{\mathcal{SE}}$, and a string $C \in \{0,1\}^*$ and outputs a message $M \in \mathsf{MsgSp}_{\mathcal{SE}}$ or the special symbol $\mathsf{INVALID}$ on error; we denote this process as $M \leftarrow \mathcal{D}_K^{N,A}(C)$. We require that $\mathcal{D}_K^{N,A}(\mathcal{E}_K^{N,A}(M)) = M$ for all $K \in \mathsf{KeySp}_{\mathcal{SE}}$, $N \in \mathsf{NonceSp}_{\mathcal{SE}}$, $A \in \mathsf{AdSp}_{\mathcal{SE}}$, and $M \in \mathsf{MsgSp}_{\mathcal{SE}}$. Let $l(\cdot)$ denote the *length function* of $\mathcal{SE}$; i.e., for all keys K, nonces N, headers A, and messages M, $|\mathcal{E}_K^{N,A}(M)| = l(|M|)$.

PRIVACY. Let $\$(\cdot,\cdot,\cdot)$ be an oracle that, on input $(N, A, M) \in \mathsf{NonceSp}_{\mathcal{SE}} \times \mathsf{AdSp}_{\mathcal{SE}} \times \mathsf{MsgSp}_{\mathcal{SE}}$, returns a random string of length $l(|M|)$. Let B be an adversary with access to an oracle and that returns a bit. Then

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathcal{SE}}(B) = \Pr\left[K \xleftarrow{\$} \mathcal{K}_e \ : \ B^{\mathcal{E}_K(\cdot,\cdot,\cdot)} = 1 \right] - \Pr\left[B^{\$(\cdot,\cdot,\cdot)} = 1 \right]$$

is the IND$-CPA-*advantage* of B in breaking the privacy of $\mathcal{SE}$ under chosen-plaintext attacks; i.e., $\mathbf{Adv}^{\mathrm{priv}}_{\mathcal{SE}}(B)$ is the advantage of B in distinguishing between ciphertexts from $\mathcal{E}_K(\cdot,\cdot,\cdot)$ and random strings. An adversary B is *nonce-respecting* if it never queries its oracle with the same nonce twice. Intuitively, a scheme $\mathcal{SE}$ preserves privacy under chosen plaintext attacks if the IND$-CPA-advantage of all nonce-respecting adversaries using reasonable resources is small.

INTEGRITY/AUTHENTICITY. Let F be a forging adversary and consider an experiment in which we first pick a random key $K \xleftarrow{\$} \mathcal{K}_e$ and then run F with oracle access to $\mathcal{E}_K(\cdot,\cdot,\cdot)$. We say that F *forges* if F returns a pair (N, A, C) such that $\mathcal{D}_K^{N,A}(C) \neq \mathsf{INVALID}$ but F did not make a query (N, A, M) to $\mathcal{E}_K(\cdot,\cdot,\cdot)$ that resulted in a response C. Then

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathcal{SE}}(F) = \Pr\left[K \xleftarrow{\$} \mathcal{K}_e \ : \ F^{\mathcal{E}_K(\cdot,\cdot,\cdot)} \text{ forges } \right]$$

is the AUTH-*advantage* of F in breaking the integrity/authenticity of $\mathcal{SE}$. Intuitively, $\mathcal{SE}$ preserves integrity if the AUTH-advantage of all nonce-respecting adversaries using reasonable resources is small.

PSEUDORANDOM FUNCTIONS AND PERMUTATIONS. Let F be a family of functions from D to R. Let A be an adversary with access to an oracle and that returns a bit. Then

$$\mathbf{Adv}^{\mathrm{prf}}_{F}(A) = \Pr\left[f \xleftarrow{\$} F \ : \ A^{f(\cdot)} = 1 \right] - \Pr\left[g \xleftarrow{\$} \mathsf{Func}(D, R) \ : \ A^{g(\cdot)} = 1 \right]$$

denotes the PRF-advantage of A in distinguishing a random instance of F from a random function. Intuitively, we say that F is a secure PRF if the PRF-advantages of all adversaries using reasonable resources is small.

Let F be a a family of functions from D to D. Let A be an adversary with access to an oracle and that returns a bit. Then

$$\mathbf{Adv}^{\mathrm{prp}}_{F}(A) = \Pr\left[f \xleftarrow{\$} F \ : \ A^{f(\cdot)} = 1 \right] - \Pr\left[g \xleftarrow{\$} \mathsf{Perm}(D) \ : \ A^{g(\cdot)} = 1 \right]$$

denotes the PRP-advantage of A in distinguishing a random instance of F from a random permutation. Intuitively, we say that F is a secure PRP if the PRP-advantages of all adversaries using reasonable resources is small.

We often model block ciphers as pseudorandom functions or permutations. In this case, given a block cipher $E : \{0,1\}^k \times \{0,1\}^L \to \{0,1\}^L$, we use $E_K(\cdot)$, $K \in \{0,1\}^k$, to denote the function $E(K,\cdot)$ and we use $f \xleftarrow{\$} E$ as short hand for $K \xleftarrow{\$} \{0,1\}^k$; $f \leftarrow E_K$. We call k the key length of E and we call L the block length.

3 The CWC mode of operation

We now describe the CWC mode of operation for 128-bit block ciphers. (See [7] for a description of the general CWC construction.)

If BC denotes a block cipher with 128-bit blocks and kl-bit keys, and if tl is the desired tag length for CWC in bits, then let CWC-BC-tl denote the CWC mode of operation instantiated with BC using tag length tl. Throughout the remainder of this section, fix BC and tl and let CWC-BC-tl $= (\mathcal{K}, \mathsf{CWC\text{-}ENC}, \mathsf{CWC\text{-}DEC})$.

We associate to CWC-BC-tl the following sets:

$$\begin{aligned}
\mathsf{MsgSp}_{\mathsf{CWC\text{-}BC\text{-}tl}} &= \{\, x \in (\{0,1\}^8)^* \;:\; |x| \le \mathsf{MaxMsgLen} \,\} \\
\mathsf{AdSp}_{\mathsf{CWC\text{-}BC\text{-}tl}} &= \{\, x \in (\{0,1\}^8)^* \;:\; |x| \le \mathsf{MaxAdLen} \,\} \\
\mathsf{KeySp}_{\mathsf{CWC\text{-}BC\text{-}tl}} &= \{0,1\}^{\mathsf{kl}} \\
\mathsf{NonceSp}_{\mathsf{CWC\text{-}BC\text{-}tl}} &= \{0,1\}^{88}
\end{aligned}$$

where MaxMsgLen and MaxAdLen are both $128 \cdot (2^{32}-1)$. That is, the payload and associated data spaces for CWC-BC-tl consist of all strings of octets that are at most $2^{32}-1$ blocks long.

3.1 The CWC core

The key generation algorithm $\mathcal{K}$ returns a randomly selected key from $\mathsf{KeySp}_{\mathsf{CWC\text{-}BC\text{-}tl}}$ (i.e., the key generation returns a random kl-bit string).

The encryption algorithm CWC-ENC works as follows:

Algorithm $\mathsf{CWC\text{-}ENC}_K(N, A, M)$ // CWC encryption
  $\sigma \leftarrow \mathsf{CWC\text{-}CTR}_K(N, M)$
  $\tau \leftarrow \mathsf{CWC\text{-}MAC}_K(N, A, \sigma)$
  Return $\sigma \| \tau$

where CWC-CTR and CWC-MAC are described in Section 3.2. The decryption algorithm CWC-DEC works as follows:

Algorithm $\mathsf{CWC\text{-}DEC}_K(N, A, C)$ // CWC decryption
  If $|C| < \mathsf{tl}$ then return INVALID
  Parse C as $\sigma \| \tau$ where $|\tau| = \mathsf{tl}$
  If $A \notin \mathsf{AdSp}_{\mathsf{CWC\text{-}BC\text{-}tl}}$ or $\sigma \notin \mathsf{MsgSp}_{\mathsf{CWC\text{-}BC\text{-}tl}}$ then return INVALID
  If $\tau = \mathsf{CWC\text{-}MAC}_K(N, A, \sigma)$ then return INVALID
  Return $\mathsf{CWC\text{-}CTR}_K(N, \sigma)$

3.2 The CWC subroutines

The remaining CWC algorithms are defined as follows:

Algorithm $\mathsf{CWC\text{-}CTR}_K(N, M)$ // CWC counter mode module
  $\alpha \leftarrow \lceil |M|/128 \rceil$
  For $i = 1$ to α do
    $\mathsf{ks}_i \leftarrow \mathsf{BC}_K(10^7 \| N \| \mathsf{tostr}(i, 32))$ // Note that 10^7 means a one bit followed by 7 zeros
  $\sigma \leftarrow$ (first $|M|$ bits of $\mathsf{ks}_1 \| \mathsf{ks}_2 \| \cdots \| \mathsf{ks}_\alpha) \oplus M$
  Return σ

Algorithm $\mathsf{CWC\text{-}MAC}_K(N, A, \sigma)$ // CWC authentication module
  $R \leftarrow \mathsf{BC}_K(\mathsf{CWC\text{-}HASH}_K(A, \sigma))$
  $\tau \leftarrow \mathsf{BC}_K(10^7 \| N \| 0^{32}) \oplus R$
  Return first tl bits of τ

Algorithm $\mathsf{CWC\text{-}HASH}_K(A, \sigma)$ // CWC universal hashing module
  $Z \leftarrow$ last 127 bits of $\mathsf{BC}_K(110^{126})$
  $K_h \leftarrow \mathsf{toint}(Z)$ // The same K_h value is used in every invocation of $\mathsf{CWC\text{-}HASH}_K$.
  $l \leftarrow$ minimum integer such that 96 divides $A \| 0^l$
  $l' \leftarrow$ minimum integer such that 96 divides $\sigma \| 0^{l'}$
  $X \leftarrow A \| 0^l \| \sigma \| 0^{l'}$; $\beta \leftarrow |X|/96$; $l_\sigma \leftarrow |\sigma|/8$; $l_A \leftarrow |A|/8$
  Break X into chunks $X_1, X_2, \ldots, X_\beta$ // $|X_1| = |X_2| = \cdots = |X_\beta| = 96$
  For $i = 1$ to β do
    $Y_i \leftarrow \mathsf{toint}(X_i)$
  $Y_{\beta+1} \leftarrow 2^{64} \cdot l_A + l_\sigma$ // Include the lengths of A and σ in the polynomial.
  $R \leftarrow Y_1 K_h^\beta + \cdots + Y_\beta K_h + Y_{\beta+1} \bmod 2^{127} - 1$
  Return $\mathsf{tostr}(R, 128)$ // Note: first bit of result will always be 0

4 Theorem statements

The CWC mode is a provably secure AEAD scheme assuming that the underlying block cipher (e.g., AES) is a secure pseudorandom function or pseudorandom permutation. This is a quite reasonable assumption since most modern block ciphers (including AES) are believed to be pseudorandom. Furthermore, all provably-secure block cipher modes of operation that we are aware of make the same assumptions we make (and some modes, e.g. OCB [12], make even stronger, albeit still reasonable, assumptions).

The specific results for CWC appear as Theorem 4.1 and Theorem 4.2 below. The proofs of these theorems appear in [7].

4.1 Integrity/authenticity

Theorem 4.1 [Integrity/authenticity of CWC.] Let CWC-BC-tl be as in Section 3. (Recall that the block cipher is BC and tag length is tl.) Consider a nonce-respecting AUTH adversary A against CWC-BC-tl. Assume the execution environment allows A to query its oracle with associated data that are at most $n \le$ MaxAdLen bits long and with messages that are at most $m \le$ MaxMsgLen bits long. Assume A makes at most $q - 1$ oracle queries and the total length of all the payload data

(both in these $q-1$ oracle queries and the forgery attempt) is at most μ. Then given A we can construct a PRF adversary B_A and a PRP adversary C_A against BC such that

$$\mathbf{Adv}^{\text{auth}}_{\mathsf{CWC\text{-}BC\text{-}tl}}(A) \leq \mathbf{Adv}^{\text{prf}}_{\mathsf{BC}}(B_A) + \frac{n+m}{2^{133}} + \frac{1}{2^{125}} + \frac{1}{2^{\mathsf{tl}}}$$

and

$$\mathbf{Adv}^{\text{auth}}_{\mathsf{CWC\text{-}BC\text{-}tl}}(A) \leq \mathbf{Adv}^{\text{prp}}_{\mathsf{BC}}(C_A) + \frac{(\mu/128 + 3q + 1)^2}{2^{129}} + \frac{n+m}{2^{133}} + \frac{1}{2^{125}} + \frac{1}{2^{\mathsf{tl}}} \ . \tag{1}$$

Furthermore, the experiments for B_A and C_A take the same time as the experiment for A and B_A and C_A make at most $\mu/128 + 3q + 1$ oracle queries. ▌

The above theorem means that if the underlying block cipher is a secure pseudorandom function or a secure pseudorandom permutation, then CWC-BC will preserve authenticity. If the underlying block cipher is something like AES, then this initial assumption seems quite reasonable and, therefore, CWC-AES will preserve authenticity.

Let us elaborate on this reasoning. Assume BC is a secure block cipher. This means that $\mathbf{Adv}^{\text{prp}}_{\mathsf{BC}}(C)$ must be small for all adversaries C using reasonable reasonable resources and, in particular, this means that, for C_A as described in the above theorem statement, $\mathbf{Adv}^{\text{prp}}_{\mathsf{BC}}(C_A)$ must be small assuming that A uses reasonable resources. And if $\mathbf{Adv}^{\text{prp}}_{\mathsf{BC}}(C_A)$ is small and μ, q, m and n are small, then, because of the above equations, $\mathbf{Adv}^{\text{auth}}_{\mathsf{CWC\text{-}BC\text{-}tl}}(A)$ must also be small as well. I.e., any adversary A using reasonable resources will only be able to break the authenticity of CWC-BC-tl with some small probability.

4.2 Privacy

Theorem 4.2 [Privacy of CWC.] Let CWC-BC-tl be as in Section 3. Then given a nonce-respecting IND$-CPA adversary A against CWC-BC-tl one can construct a PRF adversary B_A and a PRP adversary C_A against BC such that

$$\mathbf{Adv}^{\text{priv}}_{\mathsf{CWC\text{-}BC\text{-}tl}}(A) \leq \mathbf{Adv}^{\text{prf}}_{\mathsf{BC}}(B_A)$$

and, if A makes at most q oracle queries totaling at most μ bits of payload message data,

$$\mathbf{Adv}^{\text{priv}}_{\mathsf{CWC\text{-}BC\text{-}tl}}(A) \leq \mathbf{Adv}^{\text{prp}}_{\mathsf{BC}}(C_A) + \frac{(\mu/128 + 3q + 1)^2}{2^{129}} \ . \tag{2}$$

Furthermore, the experiments for B_A and C_A take the same time as the experiment for A and B_A and C_A make at most $\mu/128 + 3q + 1$ oracle queries. ▌

We interpret Theorem 4.2 in the same way we interpreted Theorem 4.1. In particular, this theorem shows that if BC is a secure pseudorandom function or pseudorandom permutation, then CWC-BC-tl preserves privacy under chosen-plaintext attacks.

CHOSEN-CIPHERTEXT PRIVACY. Since CWC-BC-tl preserves privacy under chosen-plaintext attacks (Theorem 4.2) *and* provides integrity (Theorem 4.1) assuming that BC is pseudorandom, it also provides privacy under chosen-ciphertext attacks. (See [1, 11] for a discussion of the relationship between chosen-plaintext privacy, integrity, and chosen-ciphertext privacy; this relationship was also used, for example, by [12].)

Acknowledgments

We would like to thank Peter Gutmann, David McGrew, and David Wagner for their comments. Additionally, we would like to thank Brian Gladman for helping to validate our test vectors and for providing us timing information for an optimized CWC implementation. T. Kohno was supported by a National Defense Science and Engineering Fellowship.

References

[1] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, Berlin Germany, Dec. 2000.

[2] M. Bellare, P. Rogaway, and D. Wagner. A conventional authenticated-encryption mode, Apr. 2003. Available at `http://eprint.iacr.org/2003/069/`.

[3] D. Bernstein. Floating-point arithmetic and message authentication, 2000. Available at `http://cr.yp.to/papers.html#hash127`.

[4] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In *Fast Software Encryption 2001*, Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, 2001.

[5] C. Jutla. Encryption modes with almost free message integrity. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer-Verlag, Berlin Germany, May 2001.

[6] J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In B. Schneier, editor, *Fast Software Encryption 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer-Verlag, Berlin Germany, Apr. 2000.

[7] T. Kohno, J. Viega, and D. Whiting. The CWC authenticated encryption (associated data) mode, May 2003. Available at `http://eprint.iacr.org/2003/106/`.

[8] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer-Verlag, Berlin Germany, Aug. 2001.

[9] D. McGrew. The truncated multi-modular hash function (TMMH), version two, Oct. 2002. Available at `http://www.ietf.org/internet-drafts/draft-irtf-cfrg-tmmh-00.txt`.

[10] D. McGrew. The universal security transform, Oct. 2002. Available at `http://www.ietf.org/internet-drafts/draft-irtf-cfrg-ust-00.txt`.

[11] P. Rogaway. Authenticated encryption with associated data. In *Proceedings of the 9th Conference on Computer and Communications Security*, Nov. 2002.

[12] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *Proceedings of the 8th Conference on Computer and Communications Security*, pages 196–205. ACM Press, 2001.

[13] P. Rogaway and D. Wagner. A critique of CCM, Apr. 2003. Available at `http://eprint.iacr.org/2003/070/`.

[14] M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[15] D. Whiting, N. Ferguson, and R. Housley. Counter with CBC-MAC (CCM). Submission to NIST. Available at `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/`, 2002.

A Summary of properties

In this appendix we summarize some of the properties of CWC. We include all of the properties listed in the submission guidelines on the NIST Modes of Operation website. We also discuss some additional properties that we feel are important.

Security function. CWC is a provably secure *authenticated encryption* with associated data (AEAD) scheme. Informally, this means that the encapsulation algorithm, on input a pair of messages (A, M) and some nonce N, encapsulates (A, M) in a way that protects the privacy of M and the integrity of both A and M. Our formal security statements appear in Section 4 and the proofs appear in [7].

Error propagation. Assuming that the underlying block cipher is a secure pseudorandom function or permutation, any attempt, by an adversary using reasonable resources, to forge a new ciphertext will, with very high probably, be detected. This follows from the fact that CWC is a provably-secure AEAD scheme.

Synchronization. Synchronization is based on the nonce. As with other nonce-based AEAD schemes, the nonce must either be sent with the ciphertext or the receiver must know how to derive the nonce on its own.

Parallelizability. CWC is *parallelizable*. The amount of parallelism for the hashing portion can be determined by the implementor without affecting interoperability.

Keying material required. CWC is defined to be a *single-key* AEAD scheme. However, CWC does internally use two keys (the main block cipher key and a hash key which is derived using the block cipher key). Implementors can decide whether to store the derived hash key in memory or whether to re-derive it as needed.

Counter/IV/nonce requirements. CWC uses a 11-octet *nonce*. CWC is provably secure as long as one does not query the encryption algorithm twice with the same nonce. Although it is possible to instantiate the generic CWC paradigm with other nonce lengths, for CWC the nonce size is fixed at 11-octets in order to minimize interoperability issues.

Memory requirements. The software memory requirements are basically those of the underlying block cipher. For example, fast AES in software requires 4K bytes of table, and about 200 bytes of expanded key material. In some situations, software implementations may precompute powers of the hash subkey.

Pre-processing capability. The underlying CTR mode keystream can be *precomputed*. The only block cipher input that cannot be precomputed is the output of CWC-HASH.

CWC can preprocess its associated data, thereby reducing computation time if the associated data remains static or changes only infrequently.

Message length requirements. The associated data and message can both be any string of octets with length at most $128 \cdot (2^{32} - 1)$ bits. Because there does not appear to be a need to handle

strings of arbitrary bit-length, CWC as currently specified cannot encapsulate arbitrary bit-length messages. (As discussed in [7], it is easy to modify CWC to handle arbitrary bit-length messages, if desired.)

Ciphertext expansion. The ciphertext expansion is the minimum possible while still providing a tl-bit tag. That is, on input a pair (A, M), a nonce N, and a key K, CWC-ENC$_K(N, A, M)$ outputs a ciphertext C with length $|C| = |M| + \mathsf{tl}$.

Block cipher invocations. If the hash subkey K_h is computed as part of the key generation process and not during each invocation of the CWC encapsulation routine, then CWC makes one block cipher invocation during key setup and $\lceil |M|/128 \rceil + 2$ block cipher invocations during encapsulation and decapsulation. If the hash subkey K_h is not computed as part of the key generation process, then CWC makes no block cipher invocations during key setup and $\lceil |M|/128 \rceil + 3$ block cipher invocations during encapsulation and decapsulation.

Provable security. CWC is a *provably-secure* AEAD scheme assuming that the underlying block cipher (e.g., AES) is a secure pseudorandom function or permutation. The proofs of security do not require the block cipher to satisfy the strong notion of super-pseudorandomness required by some other block cipher modes of operation.

Number of options and interoperability. CWC uses a minimal number of options. The only options are the choice of the underlying block cipher and the tag length. Having fewer options makes interoperability easier.

On-line. The CWC encryption algorithm is on-line. This means that CWC can process data as it arrives, rather than waiting for the entire message to be buffered before beginning the encryption processes. This may be advantageous when encrypting streaming data sources. (Note, however, that, like any other AEAD scheme, the decryptor should still buffer the entire message and check the tag τ before revealing the plaintext and associated data.)

Patent status. To the best of our knowledge CWC is *not* covered by any patents.

Performance. CWC is efficient in both hardware and software. In hardware, CWC can process data at 10 Gbps.

Simplicity. Although simplicity is a matter of perspective, we believe that CWC is a very simple construction. It combines standard CTR mode encryption with the evaluation of a polynomial modulo $2^{127}-1$. Because of its simplicity, we believe that CWC is easy to implement and understand.

B Intellectual property statement

The authors hereby explicitly release any intellectual property rights to CWC mode into the public domain. Further, the authors are not aware of any patent or patent application anywhere in the world that cover this mode.

C Test vectors

```
Vector #1:  CWC-AES-128
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
PLAINTEXT:  00 01 02 03  04 05 06 07
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
```

```
HASH KEY:   34 AE 6A 6F  E9 51 78 94  AC CC BB 9E  BA E7 20 8C
HASH VALUE: 2B 9E AE BE  67 3F AE 03  6B 16 EA 31  DC A7 AE 6B
AES(HVAL):  FC DC 06 4C  CD CA FE E3  DE 7A A3 CF  5C 5D B9 7B
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  AB 89 DD E9  C4 55 C1 FE  BE 7E E7 58  82 D4 8A D2
CIPHERTEXT: 88 B8 DF 06  28 FD 51 CC  57 55 DB A5  09 9F 3F 1D
            60 04 44 97  DE 89 33 A9

Vector #2:  CWC-AES-192
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80
PLAINTEXT:  00 01 02 03  04 05 06 07
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   4F A8 88 AF  06 83 60 0C  AB 35 75 EF  0A E6 01 A5
HASH VALUE: 40 E6 24 83  4B 27 9A 7B  15 42 C7 FE  29 EB 29 A3
AES(HVAL):  69 CC 0E 3D  96 98 EB 75  1F 06 A5 90  9B C2 4F 5A
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  C6 B6 F4 33  F9 12 39 4F  6A 8C B9 D3  F2 7B 0C 72
CIPHERTEXT: F0 DB A9 74  12 30 01 B0  AF 7A FA 0E  6F 8A D2 3A
            75 8A 1C 43  69 B9 43 28

Vector #3:  CWC-AES-256
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80  70 60 50 40  30 20 10 00
PLAINTEXT:  00 01 02 03  04 05 06 07
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   35 8F 2B 0C  FF E9 84 BE  F9 EE EE 55  85 36 BC E5
HASH VALUE: 18 99 E1 A6  1E 6E 37 65  C6 3A 41 99  56 8C D1 BF
AES(HVAL):  1C 56 65 0A  22 BC B5 94  AC F3 CA 24  46 03 B8 5E
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  92 0A 3B 46  82 25 16 F1  5A A3 1B AE  8D EB 72 A0
CIPHERTEXT: 7B CF 73 BE  46 9C 46 0B  8E 5C 5E 4C  A0 99 A3 65
            F6 50 D1 8A  CB E8 CA FE

Vector #4:  CWC-AES-128
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
PLAINTEXT:  00 01 02 03  04 05 06 07
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   34 AE 6A 6F  E9 51 78 94  AC CC BB 9E  BA E7 20 8C
HASH VALUE: 2E A9 2A A5  28 B1 1C 08  1C C8 2F 24  9B E4 19 8D
AES(HVAL):  EA 54 F8 3D  56 7F 53 05  88 B1 EA 96  36 79 CD AC
```

```
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  AB 89 DD E9  C4 55 C1 FE  BE 7E E7 58  82 D4 8A D2
CIPHERTEXT: 88 B8 DF 06  28 FD 51 CC  41 DD 25 D4  92 2A 92 FB
            36 CF 0D CE  B4 AD 47 7E

Vector #5:  CWC-AES-192
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80
PLAINTEXT:  00 01 02 03  04 05 06 07
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   4F A8 88 AF  06 83 60 0C  AB 35 75 EF  0A E6 01 A5
HASH VALUE: 60 3F FC 24  71 64 2E D9  57 E1 B1 EA  F2 F8 B0 34
AES(HVAL):  D8 39 86 2A  33 5A 54 68  C8 16 DA 47  69 A2 10 EB
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  C6 B6 F4 33  F9 12 39 4F  6A 8C B9 D3  F2 7B 0C 72
CIPHERTEXT: F0 DB A9 74  12 30 01 B0  1E 8F 72 19  CA 48 6D 27
            A2 9A 63 94  9B D9 1C 99

Vector #6:  CWC-AES-256
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80  70 60 50 40  30 20 10 00
PLAINTEXT:  00 01 02 03  04 05 06 07
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   35 8F 2B 0C  FF E9 84 BE  F9 EE EE 55  85 36 BC E5
HASH VALUE: 0A C6 B1 39  57 7F 26 DA  94 16 42 E1  6D 73 EC B5
AES(HVAL):  4B A5 AD 1E  74 A2 C5 BE  AB D0 DA 4D  F4 29 83 0C
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  92 0A 3B 46  82 25 16 F1  5A A3 1B AE  8D EB 72 A0
CIPHERTEXT: 7B CF 73 BE  46 9C 46 0B  D9 AF 96 58  F6 87 D3 4F
            F1 73 C1 E3  79 C2 F1 AC

Vector #7:  CWC-AES-128
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   34 AE 6A 6F  E9 51 78 94  AC CC BB 9E  BA E7 20 8C
HASH VALUE: 79 00 74 72  E1 C8 36 96  ED 7A B1 F9  03 6E 94 8B
AES(HVAL):  2B 0F 24 69  B1 2B BE 39  C9 40 67 BA  F1 25 E2 5B
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  AB 89 DD E9  C4 55 C1 FE  BE 7E E7 58  82 D4 8A D2
```

```
CIPHERTEXT: 88 B8 DF 06  28 FD 51 CC  31 E6 6E 57  0B 0F 77 80
            86 F9 80 75  7E 7F C7 77  3E 80 E2 73  F1 68 89

Vector #8:  CWC-AES-192
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   4F A8 88 AF  06 83 60 0C  AB 35 75 EF  0A E6 01 A5
HASH VALUE: 2C 5E 3A A4  37 1C 27 D6  E8 6B 76 DC  3D 93 BC 87
AES(HVAL):  48 6E 9C E5  C3 16 3E A6  9C D4 D7 E2  7C 9D 92 D2
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  C6 B6 F4 33  F9 12 39 4F  6A 8C B9 D3  F2 7B 0C 72
CIPHERTEXT: F0 DB A9 74  12 30 01 B0  E1 42 B7 58  87 C9 00 8E
            D8 68 D6 3A  04 07 E9 F6  58 6E 31 8E  E6 9E A0

Vector #9:  CWC-AES-256
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80  70 60 50 40  30 20 10 00
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   35 8F 2B 0C  FF E9 84 BE  F9 EE EE 55  85 36 BC E5
HASH VALUE: 4A 70 29 CC  58 25 52 CB  75 AD C9 60  FF B3 F7 55
AES(HVAL):  2B 64 0E 02  CE 51 DE 22  B2 0F 2A 8D  C4 23 CD C0
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  92 0A 3B 46  82 25 16 F1  5A A3 1B AE  8D EB 72 A0
CIPHERTEXT: 7B CF 73 BE  46 9C 46 0B  9B C6 2D DE  26 DD 47 B9
            6E 35 44 4C  74 C8 D3 E8  AC 31 23 49  C8 BF 60

Vector #10:  CWC-AES-128
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   34 AE 6A 6F  E9 51 78 94  AC CC BB 9E  BA E7 20 8C
HASH VALUE: 51 AE 9D 7E  86 BD E0 B2  AA 18 2C 91  87 0A 9C A5
AES(HVAL):  DF 48 30 BD  1D DC E0 59  B1 C2 0B 29  01 4F 80 10
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  AB 89 DD E9  C4 55 C1 FE  BE 7E E7 58  82 D4 8A D2
CIPHERTEXT: 88 B8 DF 06  28 FD 51 CC  31 E6 6E 57  0B 0F 77 74
            C1 ED 54 D9  89 21 A7 0F  BC EC 71 83  9B 0A C2
```

```
Vector #11:  CWC-AES-192
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   4F A8 88 AF  06 83 60 0C  AB 35 75 EF  0A E6 01 A5
HASH VALUE: 51 60 E7 81  DC 64 F9 CD  54 BA 02 40  A2 E8 EE 99
AES(HVAL):  A0 30 58 13  22 B6 80 53  64 B0 3E 52  41 D2 2D 0A
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  C6 B6 F4 33  F9 12 39 4F  6A 8C B9 D3  F2 7B 0C 72
CIPHERTEXT: F0 DB A9 74  12 30 01 B0  E1 42 B7 58  87 C9 00 66
            86 AC 20 DB  A4 B9 1C 0E  3C 87 81 B3  A9 21 78

Vector #12:  CWC-AES-256
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80  70 60 50 40  30 20 10 00
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   35 8F 2B 0C  FF E9 84 BE  F9 EE EE 55  85 36 BC E5
HASH VALUE: 3F F5 0C 60  E6 01 7A 3C  A1 BB B3 54  65 02 85 7C
AES(HVAL):  3E EF A2 E4  97 91 82 86  73 0C F6 E9  46 2C CA 15
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  92 0A 3B 46  82 25 16 F1  5A A3 1B AE  8D EB 72 A0
CIPHERTEXT: 7B CF 73 BE  46 9C 46 0B  9B C6 2D DE  26 DD 47 AC
            E5 99 A2 15  B4 94 77 29  AF ED 47 CB  C7 B8 B5

Vector #13:  CWC-AES-128
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            80 81 82 83  84 85 86 87  88 89 8A 8B  8C 8D 8E 8F
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   34 AE 6A 6F  E9 51 78 94  AC CC BB 9E  BA E7 20 8C
HASH VALUE: 58 D5 28 89  4F 1F 6A 52  A6 44 FA 69  65 C0 73 A6
AES(HVAL):  A3 9E F3 6F  67 1F FA F8  71 0C 83 BB  49 A6 6E BC
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  AB 89 DD E9  C4 55 C1 FE  BE 7E E7 58  82 D4 8A D2
CIPHERTEXT: 88 B8 DF 06  28 FD 51 CC  31 E6 6E 57  0B 0F 77 0F
            48 5B 82 64  6E CF B9 F9  A0 B0 75 4F  D5 94 36 5A
            08 17 2E 86  A3 4A 3B 06  CF 72 64 E3  CB 72 E4 6E
```

```
Vector #14:  CWC-AES-192
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            80 81 82 83  84 85 86 87  88 89 8A 8B  8C 8D 8E 8F
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   4F A8 88 AF  06 83 60 0C  AB 35 75 EF  0A E6 01 A5
HASH VALUE: 0D 0A D2 78  1E 8F E8 47  00 85 31 28  B1 E3 49 3A
AES(HVAL):  5A 05 AA 45  88 06 A9 C1  DC 5A F6 AF  6F 8F EC F6
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  C6 B6 F4 33  F9 12 39 4F  6A 8C B9 D3  F2 7B 0C 72
CIPHERTEXT: F0 DB A9 74  12 30 01 B0  E1 42 B7 58  87 C9 00 A3
            A4 C4 70 6D  40 41 F4 F9  58 E1 3F D0  D7 60 4D 1E
            9C B3 5E 76  71 14 90 8E  B6 D6 4F 7C  9D F4 E0 84

Vector #15:  CWC-AES-256
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80  70 60 50 40  30 20 10 00
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            80 81 82 83  84 85 86 87  88 89 8A 8B  8C 8D 8E 8F
ASSOC DATA: <None>
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   35 8F 2B 0C  FF E9 84 BE  F9 EE EE 55  85 36 BC E5
HASH VALUE: 02 F2 DA E9  83 72 0E BC  DC 77 89 3B  67 CB 3D B7
AES(HVAL):  B7 F6 AE DE  A3 95 35 FE  03 93 08 DF  E0 C7 F1 78
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  92 0A 3B 46  82 25 16 F1  5A A3 1B AE  8D EB 72 A0
CIPHERTEXT: 7B CF 73 BE  46 9C 46 0B  9B C6 2D DE  26 DD 47 B5
            D2 41 06 CA  5D EB 80 A7  B5 71 0A 38  A4 39 8D BA
            25 FC 95 98  21 B0 23 0F  59 30 13 71  6D 2C 83 D8

Vector #16:  CWC-AES-128
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            80 81 82 83  84 85 86 87  88 89 8A 8B  8C 8D 8E 8F
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   34 AE 6A 6F  E9 51 78 94  AC CC BB 9E  BA E7 20 8C
HASH VALUE: 05 EE B6 CB  DF A6 E5 B8  4C 65 DD F4  8C C8 25 23
AES(HVAL):  62 E5 23 FE  48 8F BC 14  E3 77 15 6C  4D 0F D0 8B
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  AB 89 DD E9  C4 55 C1 FE  BE 7E E7 58  82 D4 8A D2
CIPHERTEXT: 88 B8 DF 06  28 FD 51 CC  31 E6 6E 57  0B 0F 77 0F
```

```
            48 5B 82 64  6E CF B9 F9  A0 B0 75 4F  D5 94 36 5A
            C9 6C FE 17  8C DA 7D EA  5D 09 F2 34  CF DB 5A 59

Vector #17:  CWC-AES-192
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            80 81 82 83  84 85 86 87  88 89 8A 8B  8C 8D 8E 8F
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   4F A8 88 AF  06 83 60 0C  AB 35 75 EF  0A E6 01 A5
HASH VALUE: 10 E1 48 E2  D0 68 39 EC  C4 0A 6C A3  D6 8B 47 54
AES(HVAL):  23 0A 37 C3  48 7C 9F 76  05 B9 5D 1A  21 D5 D5 FD
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  C6 B6 F4 33  F9 12 39 4F  6A 8C B9 D3  F2 7B 0C 72
CIPHERTEXT: F0 DB A9 74  12 30 01 B0  E1 42 B7 58  87 C9 00 A3
            A4 C4 70 6D  40 41 F4 F9  58 E1 3F D0  D7 60 4D 1E
            E5 BC C3 F0  B1 6E A6 39  6F 35 E4 C9  D3 AE D9 8F

Vector #18:  CWC-AES-256
AES KEY:    00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            F0 E0 D0 C0  B0 A0 90 80  70 60 50 40  30 20 10 00
PLAINTEXT:  00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F
            80 81 82 83  84 85 86 87  88 89 8A 8B  8C 8D 8E 8F
ASSOC DATA: 54 68 69 73  20 69 73 20  61 20 70 6C  61 69 6E 74
            65 78 74 20  68 65 61 64  65 72 2E 00
NONCE:      FF EE DD CC  BB AA 99 88  77 66 55
------------------------------------------------------------
HASH KEY:   35 8F 2B 0C  FF E9 84 BE  F9 EE EE 55  85 36 BC E5
HASH VALUE: 09 4D C5 21  94 79 E0 58  4E E9 C1 2C  29 6A E3 A4
AES(HVAL):  E9 69 49 47  09 07 62 3B  A9 8D AD 51  9F D5 D1 F7
MAC CTR PT: 80 FF EE DD  CC BB AA 99  88 77 66 55  00 00 00 00
AES(MCPT):  92 0A 3B 46  82 25 16 F1  5A A3 1B AE  8D EB 72 A0
CIPHERTEXT: 7B CF 73 BE  46 9C 46 0B  9B C6 2D DE  26 DD 47 B5
            D2 41 06 CA  5D EB 80 A7  B5 71 0A 38  A4 39 8D BA
            7B 63 72 01  8B 22 74 CA  F3 2E B6 FF  12 3E A3 57
```