The EAX Mode of Operation

Mihir Bellare[1], Phillip Rogaway[2], and David Wagner[3]

[1] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-mail: mihir@cs.ucsd.edu. URL: www-cse.ucsd.edu/users/mihir

[2] 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. URL: www.cs.ucdavis.edu/~rogaway/

[3] Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, California 94720, USA. E-mail: daw@cs.berkeley.edu. URL: www.cs.berkeley.edu/~daw/

Abstract. We propose a block-cipher mode of operation, EAX, for solving the problem of authenticated-encryption with associated-data (AEAD). Given a nonce N, a message M, and a header H, our mode protects the privacy of M and the authenticity of both M and H. Strings N, M, and H are *arbitrary* bit strings, and the mode uses $2\lceil |M|/n \rceil + \lceil |H|/n \rceil + \lceil |N|/n \rceil$ block-cipher calls when these strings are nonempty and n is the block length of the underlying block cipher. Among EAX's characteristics are that it is on-line (the length of a message isn't needed to begin processing it) and a fixed header can be pre-processed, effectively removing the per-message cost of binding it to the ciphertext.

1 Introduction

An authenticated encryption (AE) scheme is a symmetric-key mechanism by which a message M is a transformed into a ciphertext CT with the goal that CT protect both the privacy *and* the authenticity of M. The last few years has seen the emergence of AE as a recognized cryptographic goal. With this has come the development of new authenticated-encryption schemes and the analysis of old ones. This paper offers up a new authenticated-encryption scheme, EAX, and provides a thorough analysis of it. To understand why we are defining a new AE scheme, we need to give some background.

FLAVORS OF AUTHENTICATED ENCRYPTION. It useful to distinguish two kinds of AE schemes. In a *two-pass* scheme we make two passes through the data, one aimed at providing privacy and the other, authenticity. One way of making a two-pass AE scheme is by *generic composition*, wherein one pass constitutes a (privacy-only) symmetric-encryption scheme, while the other pass is a message authentication code (MAC). The encryption scheme and the MAC each use their own key. Analyses of some generic composition methods can be found in [5, 6, 20].

In a *one-pass* AE scheme we make a single pass through the data, simultaneously doing what is needed to engender both privacy and authenticity. Typically, the computational cost is about half that of a two-pass scheme. Such schemes emerged only recently. They include IAPM, OCB, and XCBC [12, 17, 25].

Soon after the emergence of one-pass AE schemes it was realized that often not all the data should be privacy-protected. Changes were needed to the basic definitions and mechanisms in order to support the possibility that some information, like a packet header, must *not* be encrypted. Thus was born the notion of *authenticated-encryption with associated-data* (AEAD), first formally defined in [24]. The non-secret data is called the *associated data* or the *header*. Like an AE schemes, an AEAD scheme might make one pass or two.

STANDARDIZING A TWO-PASS AEAD SCHEME. Traditionally, it has been the designers of applications and network protocols who were responsible for combining privacy and authenticity mechanisms in order to make a two-pass AEAD scheme. This has not worked well. It turns out that there are numerous ways to go wrong in trying to make a secure AEAD scheme, and many protocols, products, and standards have done just that. (For example, see [11] for a wrong one-pass scheme, see [5] for weaknesses in the AEAD mechanism of SSH, and [6, 20] for attacks on some methods of popular use.)

Nowadays, some standards bodies (including NIST, IETF, and IEEE 802.11) would like to standardize on an AEAD scheme. Indeed IEEE 802.11 has already done so. This is a good direction. Standardized AEAD might help minimize errors in mis-combining cryptographic mechanisms.

So far, standards bodies have been unwilling to standardize on any of the one-pass schemes due to pending patents covering them. There is, accordingly, an established desire for standardizing on a two-pass AEAD scheme. The two-pass scheme should be *as good as possible* subject to the limitation of falling within the two-pass framework.

Generic-composition would seem to be the obvious answer. But defining a generic-composition AEAD scheme is not an approach that has moved forward within any of the standards bodies. There would seem to be a number of reasons. One reason is a relatively minor inefficiency—the fact that generic composition methods must use two keys. Probably a bigger issue is that the architectural advantage of generic composition brings with it an “excessive” degree of choice—after deciding on a generic composition method, one still needs two lower-level specifications, namely a symmetric encryption scheme and a MAC, for each of which numerous block-cipher based choices exist. Standards bodies want something self-contained, as well as being a patent-avoiding, block-cipher based, single-key mechanism.

So far, there has been exactly one proposal for such a method (though see the “contemporaneous work” section below). It is called CCM [26], and is due to Whiting, Housley, and Ferguson [26]. CCM has enjoyed rapid success, and is now the required mechanism for IEEE 802.11 wireless LANs as well as 802.15.4 wireless personal area networks. NIST has indicated that it plans to put out a “Recommendation” based on CCM.

OUR CONTRIBUTIONS. It is our view that CCM has a good deal of pointless complexity and inefficiency. It is the first contribution of this paper to explain these limitations. It is the second and main contribution of this paper to provide a new AEAD scheme, EAX, that avoids these limitations.

CCM LIMITATIONS. A description of CCM, together with a detailed description of its shortcomings, can be found in the full version of this paper [8]. Some of the points we make and elaborate on there are the following. CCM is not on-line, meaning one needs to know the lengths of both the plaintext and the associated data before one can proceed with encryption. This may be inconvenient or inefficient. CCM does not allow pre-processing of static associated data. (If, for example, we have an unchanging header attached to every packet being authenticated, we would like that the cost of authenticating this header be paid only once, meaning header authentication should have no significant cost after a single pre-computation. CCM fails to have this property.) CCM's parameterization is more complex than necessary, including, in addition to the block cipher and tag length, a message-length parameter. CCM's nonce length is restricted in such a way that it may not provide adequate security when nonces are chosen randomly. Finally, CCM implementations could suffer performance hits because the algorithm can disrupt word alignment in the associated data.

EAX AND ITS ATTRIBUTES. EAX is a nonce-using AEAD scheme employing no tool beyond the block cipher $E\colon \mathsf{Key} \times \{0,1\}^n \to \{0,1\}^n$ on which it is based. We expect that E will often be instantiated by AES, but we make no restrictions in this direction. (In particular we do not require that $n = 128$.) Nothing is assumed about the nonces except that they are non-repeating. EAX provides both privacy, in the sense of indistinguishability from random bits, and authenticity, in the sense of an adversary's inability to produce a new but valid $\langle$nonce, header, ciphertext$\rangle$ triple. EAX is simple, avoiding complicated length-annotation. It is a conventional two-pass AEAD scheme, making a separate privacy pass and authenticity pass, using no known intellectual property.

EAX is flexible in the functionality it provides. It supports arbitrary-length messages: the message space is $\{0,1\}^*$. The key space for EAX is the key space Key of the underlying block cipher. EAX supports arbitrary nonces, meaning the nonce space is $\{0,1\}^*$. Any tag length $\tau \in [0 \mathinner{..} n]$ is possible, to allow each user to select how much security she wants from the authenticity guarantees. The only user-selectable parameters are the block cipher E and that tag length τ.

EAX has desirable performance attributes. Message expansion is minimal: the length of the ciphertext (which, following the conventions of [25], excludes the nonce) is only τ bits more than the length of the plaintext. Implementations can profitably pre-process static associated data. (If an unchanging header is attached to every packet, authenticating this header has no significant cost after a single pre-computation.) Key-setup is efficient: all block-cipher calls use the same underlying key, so that we do not incur the cost of key scheduling more than once. For both encryption and decryption, EAX uses only the forward direction of the block cipher, so that hardware implementations do not need to implement the decryption functionality of the block cipher. The scheme is on-line for both the plaintext M and the associated data H, which means that one can process streaming data on-the-fly, using constant memory, not knowing when the stream will stop.

PROVABLE SECURITY. We prove that EAX is secure assuming that the block cipher that it uses is a secure pseudorandom permutation (PRP). Security for EAX means indistinguishability from random bits *and* authenticity of ciphertexts. The combination implies other desirable goals, like nonmalleability and indistinguishability under a chosen-ciphertext attack.

The proof of security for EAX is surprisingly complex. The key-collapse of EAX2 destroys a fundamental abstraction boundary. Our security proof relies on a result about the security of a tweakable extension of OMAC (Lemma 3) in which an adversary can obtain not only a tag for a message of its choice, but also an associated key-stream.

PRAGMATICS. The main reason there is any interest in two-pass schemes, as we have already discussed, is that one-pass schemes would seem to be subject to patents. Motivated by this, standardization bodies have expressed the intent of standardizing on a conventional, two-pass scheme, even understanding the factor-of-two performance hit. The merit of this judgment is debatable, but the pragmatic reality is that there has emerged a desire for a conventional scheme, like EAX, that is *as good as possible* subject to the two-pass constraint. Lack of a scheme like EAX will simply lead to an inferior scheme being standardized, which is to the disadvantage of the user community. Accordingly, EAX addresses a real and practical design problem. We took up work on this design problem at the suggestion of the co-Chair of the IRTF (Internet Research Task Force), which supports the standardization efforts of the IETF. We believe that EAX has the potential for widespread adoption and use.

AFTERWARDS. One non-goal of EAX was to be parallelizable. Another recent two-pass design, CWC [19], is parallelizable. It pays for this advantage with a somewhat complex algorithm, based on Carter-Wegman hashing using polynomial evaluation over a prime field. More recent still is GCM [22], a parallelizable, two-pass design based on multiplication in the finite field with 2^{128} elements.

Other recent AEAD mechanisms include Helix [10] and SOBER-128 [13]. These are stream ciphers that aim to provide authenticity. The provable-security methodology does not apply to these objects since they are built directly rather than from lower level primitives.

2 Preliminaries

All strings in this paper 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 $X \parallel Y$ or simply $X\,Y$. The string of length 0, called the *empty string*, is denoted ε. If $X \in \{0,1\}^*$ we let $|X|$ denote its length, in bits. If $X \in \{0,1\}^*$ and $\ell \leq |X|$ then the first ℓ bits of X are denoted $X\ [\text{first } \ell \text{ bits}]$. 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*. If $X \in \textsc{Byte}^*$ we let $\|X\|_8 = |X|/8$ denote its length in bytes. For $\ell \geq 1$ a number, we write $\textsc{Byte}^{<\ell}$ for all byte strings having fewer than ℓ bytes. If $X \in \textsc{Byte}^*$ and $\ell \leq \|X\|_n$ then the first ℓ bytes of X are denoted $X\ [\text{first } \ell \text{ bytes}]$. When $X \in \{0,1\}^n$ is a nonempty string and $t \in \mathbb{N}$ is a number we let $X + t$ be the n-bit string that results from regarding X as a nonnegative number x (binary notation, most-significant-bit first), adding x to t, taking the result modulo 2^n, and converting this number back into an n-bit string. If $t \in [0..2^n - 1]$ we let $[t]_n$ denote the encoding of t into an n-bit binary string (msb first, lsb last). If X and P are strings then we let $X \oplus\!\!\!\rightarrow P$ (the *xor-at-the-end* operator) denote the string of length $\ell = \max\{|X|, |P|\}$ bits that is obtained by prepending $\big||X| - |P|\big|$

```
**Algorithm** $\mathrm{CBC}_K\ (M)$
10  Let $M_1 \cdots M_m \leftarrow M$ where $|M_i| = n$
11  $C_0 \leftarrow 0^n$
12  **for** $i \leftarrow 1$ **to** $m$ **do**
13      $C_i \leftarrow E_K(M_i \oplus C_{i-1})$
14  **return** $C_m$
```

```
**Algorithm** $\mathrm{CTR}_K^{\mathcal{N}}\ (M)$
20  $m \leftarrow \lceil |M|/n \rceil$
21  $S \leftarrow E_K(\mathcal{N}) \,\|\, \cdots \,\|\, E_K(\mathcal{N}+m-1)$
22  $C \leftarrow M \oplus S$ [first $|M|$ bits]
23  **return** $C$
```

```
**Algorithm** pad $(M;\ B, P)$
30  **if** $|M| \in \{n, 2n, 3n, \ldots\}$
31  **then return** $M \oplus\!\!\rightarrow B$,
32  **else return** $(M \,\|\, 10^{n-1-(|M| \bmod n)}) \oplus\!\!\rightarrow P$
```

```
**Algorithm** $\mathrm{OMAC}_K\ (M)$
40  $L \leftarrow E_K(0^n);\ \ B \leftarrow 2L;\ \ P \leftarrow 4L$
41  **return** $\mathrm{CBC}_K(\mathsf{pad}\,(M;\ B, P))$
```

```
**Algorithm** $\mathrm{OMAC}_K^t\ (M)$
50  **return** $\mathrm{OMAC}_K([t]_n \,\|\, M)$
```

Fig. 1. Basic building blocks. The block cipher $E\colon \mathsf{Key} \times \{0,1\}^n \to \{0,1\}^n$ is fixed and $K \in \mathsf{Key}$. For CBC, $M \in (\{0,1\}^n)^+$. For CTR, $M \in \{0,1\}^*$ and $\mathcal{N} \in \{0,1\}^n$. For pad, $M \in \{0,1\}^*$ and $B, P \in \{0,1\}^n$ and the operation $\oplus\!\!\rightarrow$ xors the shorter string into the end of longer one. For OMAC, $M \in \{0,1\}^*$ and $t \in [0..2^n - 1]$ and the multiplication of a number by a string L is done in $\mathrm{GF}(2^n)$.

zero-bits to the shorter string and then xoring this with the other string. (In other words, xor the shorter string into the *end* of the longer string.) 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.* Throughout this note we fix such a block cipher E.

In Figure 1 we define the algorithms CBC, CTR, pad, OMAC (no superscript), and OMAC$^{\bullet}$ (with superscript). The algorithms CBC (the CBC MAC) and CTR (counter-mode encryption) are standard. Algorithm pad is used only to define OMAC. Algorithm OMAC [14] is a pseudorandom function (PRF) that is a one-key variant of the algorithm XCBC [9]. Algorithm OMAC$^{\bullet}$ is like OMAC but takes an extra argument, the integer t. This algorithm is a "tweakable" PRF [21], tweaked in the most simple way possible.

We explain the notation used in the definition of OMAC. The value of iL (line 40: i an integer in $\{2,4\}$ and $L \in \{0,1\}^n$) is the n-bit string that is obtained by multiplying L by the n-bit string that represents the number i. The multiplication is done in the finite field $\mathrm{GF}(2^n)$ using a canonical polynomial to represent field points. The canonical polynomial we select is the lexicographically first polynomial among the irreducible polynomials of degree n that have a minimum number of nonzero coefficients. For $n = 128$ the indicated polynomial is $\mathsf{x}^{128} + \mathsf{x}^7 + \mathsf{x}^2 + \mathsf{x} + 1$. In that case, $2L = L{\ll}1$ if the first bit of L is 0 and $2L = (L{\ll}1) \oplus 0^{120}10000111$ otherwise, where $L{\ll}1$ means the left shift of L by one position (the first bit vanishing and a zero entering into the last bit). The value of $4L$ is simply $2(2L)$. We warn that to avoid side-channel attacks one must implement the doubling operation in a constant-time manner.

```
**Algorithm** $\text{EAX.Encrypt}_K^{N\,H}(M)$
10 $\mathcal{N} \leftarrow \text{OMAC}_K^0(N)$
11 $\mathcal{H} \leftarrow \text{OMAC}_K^1(H)$
12 $C \leftarrow \text{CTR}_K^{\mathcal{N}}(M)$
13 $\mathcal{C} \leftarrow \text{OMAC}_K^2(C)$
14 $\mathit{Tag} \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
15 $T \leftarrow \mathit{Tag}$ [first $\tau$ bits]
16 **return** $\mathit{CT} \leftarrow C \parallel T$

**Algorithm** $\text{EAX.Decrypt}_K^{N\,H}(\mathit{CT})$
20 **if** $|\mathit{CT}| < \tau$ **then return** INVALID
21 Let $C \parallel T \leftarrow \mathit{CT}$ where $|T| = \tau$
22 $\mathcal{N} \leftarrow \text{OMAC}_K^0(N)$
23 $\mathcal{H} \leftarrow \text{OMAC}_K^1(H)$
24 $\mathcal{C} \leftarrow \text{OMAC}_K^2(C)$
25 $\mathit{Tag}' \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
26 $T' \leftarrow \mathit{Tag}'$ [first $\tau$ bits]
27 **if** $T \neq T'$ **then return** INVALID
28 $M \leftarrow \text{CTR}_K^{\mathcal{N}}(C)$
29 **return** $M$
```

Fig. 2. Encryption and decryption under EAX mode. The plaintext is M, the ciphertext is CT, the key is K, the nonce is N, and the header is H. The mode depends on a block cipher E (that CTR and OMAC implicitly use) and a tag length τ.

We have made a small modification to the OMAC algorithm as it was originally presented, changing one of its two constants. Specifically, the constant 4 at line 40 was the constant $1/2$ (the multiplicative inverse of 2) in the original definition of OMAC [14]. The OMAC authors indicate that they will promulgate this modification [15], which slightly simplifies implementations.

3 The EAX Algorithm

ALGORITHM. Fix a block cipher $E\colon \mathsf{Key} \times \{0,1\}^n \to \{0,1\}^n$ and a tag length $\tau \in [0..n]$. These parameters should be fixed at the beginning of a particular session that will use EAX mode. Typically, the parameters would be agreed to in an authenticated manner between the sender and the receiver, or they would be fixed for all time for some particular application. Given these parameters, EAX provides a nonce-based AEAD scheme EAX$[E,\tau]$ whose encryption algorithm has signature $\mathsf{Key} \times \mathsf{Nonce} \times \mathsf{Header} \times \mathsf{Plaintext} \to \mathsf{Ciphertext}$ and whose decryption algorithm has signature $\mathsf{Key} \times \mathsf{Nonce} \times \mathsf{Header} \times \mathsf{Ciphertext} \to \mathsf{Plaintext} \cup \{\text{INVALID}\}$ where Nonce, Header, Plaintext, and Ciphertext are all $\{0,1\}^*$. The EAX algorithm is specified in Figure 2 and a picture illustrating EAX encryption is given in Figure 3. We now discuss various features of our algorithm and choices underlying the design.

NO ENCODINGS. We have avoided any nontrivial encoding of multiple strings into a single one.[1] Some other approaches that we considered required a PRF to be applied to what was logically a tuple, like (N,H,C). Doing this raises encoding issues we did not want to deal with because, ultimately, there would seem to be no simple, efficient, compelling, on-line way to encode multiple strings into a single one. Alternatively, one

[1] One could view the prefixing of $[t]_n$ to M in the definition of $\text{OMAC}_K^t(M)$ as an encoding, but $[t]_n$ is a constant, fixed-length string, and the aim here is just to “tweak” the PRF. This is very different from needing to encode arbitrary-length strings into a single string.

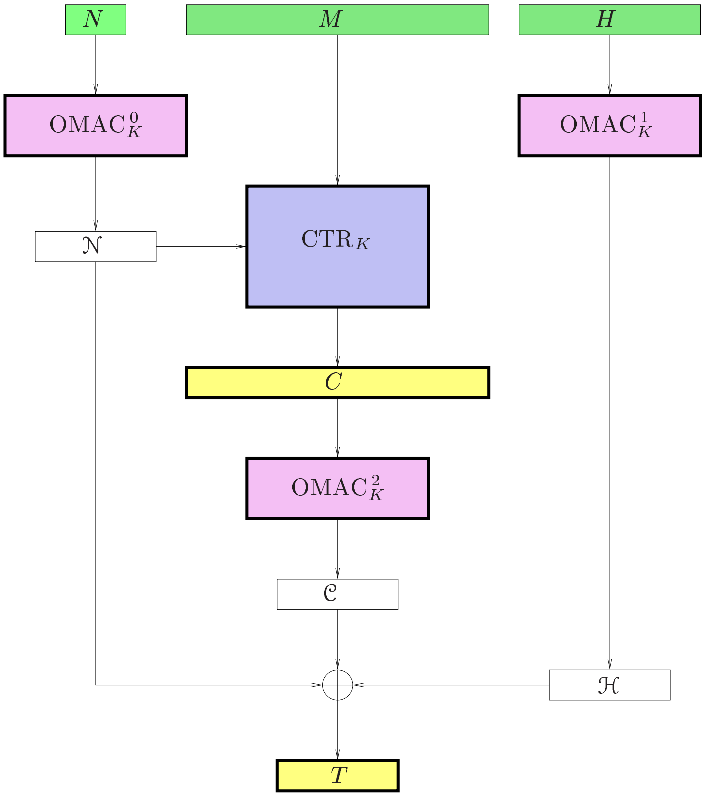

Fig. 3. Encryption under EAX. The message is M, the key is K, and the header is H. The ciphertext is $CT = C \parallel T$.

could avoid encodings and consider a new kind of primitive, a multi-argument PRF. But this would be a non-standard tool and we didn't want to use any non-standard tools. All in all, it seemed best to find a way to sidestep the need to do encodings.

WHY NOT GENERIC COMPOSITION? Why have we specified a block-cipher based (BC-based) AEAD scheme instead of following the generic-composition approach of combining a (privacy-only) encryption method and a message authentication code? In fact, there are reasonable arguments in favor of generic composition, based on aesthetic or architectural sensibilities. One can argue that generic composition better separates conceptually independent elements (privacy and authenticity) and, correspondingly, allows greater implementation flexibility [6, 20]. Correctness becomes much simpler and clearer as well. All the same, BC-based AEAD modes have some important advantages

	CCM	**EAX**
Functionality	AE with AD	AE with AD
Built from	Block cipher E with 128-bit blocksize	Block cipher E with n-bit blocksize
Parameters	Block cipher E Tag length $\boldsymbol{\tau} \in \{4, 6, 8, 10, 12, 14, 16\}$ Length of message length field $\boldsymbol{\lambda} \in [2..8]$	Block cipher E Tag length $\tau \in [0..n]$
Message space	Parameterized: 7 choices: $\boldsymbol{\lambda} \in [2..8]$. Each possible message space a subset of $\textsc{Byte}^*$, from $\textsc{Byte}^{2^{16}-1}$ to $\textsc{Byte}^{<2^{64}-1}$	$\{0,1\}^*$
Nonce space	Parameterized, with a value of $15 - \boldsymbol{\lambda}$ bytes. From 56 bits to 104 bits	$\{0,1\}^*$
Key space	One block-cipher key	One block-cipher key
Ciphertext expansion	$\boldsymbol{\tau}$ bytes	τ bits
Block-cipher calls	$2\left\lceil\frac{\lvert M\rvert}{128}\right\rceil + \left\lceil\frac{\lvert H\rvert}{128}\right\rceil + 2 + \delta$, for $\delta \in \{0,1\}$	$2\left\lceil\frac{\lvert M\rvert}{n}\right\rceil + \left\lceil\frac{\lvert H\rvert}{n}\right\rceil + \left\lceil\frac{\lvert N\rvert}{n}\right\rceil$
Block-cipher calls with static header	$2\left\lceil\frac{\lvert M\rvert}{128}\right\rceil + \left\lceil\frac{\lvert H\rvert}{128}\right\rceil + 2 + \delta$, for $\delta \in \{0,1\}$	$2\left\lceil\frac{\lvert M\rvert}{n}\right\rceil + \left\lceil\frac{\lvert N\rvert}{n}\right\rceil$
Key setup	Block cipher subkeys	Block cipher subkeys 3 block-cipher calls
IV requirements	Non-repeating nonce	Non-repeating nonce
Parallelizable?	No	No
On-line?	No	Yes
Preprocessing (/msg)	Limited (key stream)	Limited (key stream, header)
Memory rqmts	Small constant	Small constant
Provable security?	Yes (if E is a good PRP) Bound of $\Theta(\sigma^2/2^{128})$	Yes (if E is a good PRP) Bound of $\Theta(\sigma^2/2^n)$
Patent-encumbered?	No	No

Fig. 4. A comparison of basic characteristics of CCM and EAX. The count on block-cipher calls for EAX ignores key-setup costs. We denote by τ the length of the EAX tag in bits, and by $\boldsymbol{\tau}$ (boldface) the length of the CCM tag in bytes.

of their own. They make it easier for implementors to use a scheme without knowing a lot of cryptography, presenting a simpler abstraction boundary. They make it easier to obtain interoperably. They reduce the risk that implementors will choose insecure parameters. They can save on key bits and key-setup time, as generic-composition methods invariably require a pair of separate keys.

EAX can be viewed as having been derived from a generic-composition scheme we call EAX2, described in Section 4. Specifically, one instantiates EAX2 using CTR mode (counter mode) and OMAC, and then collapses the two keys into one. If one favors generic composition, EAX2 is a nice algorithm for it.

ON-LINE. We say that an algorithm is *on-line* if it is 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. A failure to be on-line has been regarded as a significant defect for an encryption scheme or a MAC. EAX is on-line.

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.

ABILITY TO 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 include 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 $\text{Encrypt}_K^{N\,H}(M)$ and $\text{Decrypt}_K^{N\,H}(C)$ should be independent of $|H|$, disregarding the work done in a preprocessing step. The significance of this goal was already explained in [24]. EAX achieves this goal.

ADDITIONAL FEATURES. Invalid messages can be rejected at half the cost of decryption. This is one of the benefits of following what is basically an encrypt-then-authenticate approach as opposed to an authenticate-then-encrypt approach.

To obtain a MAC as efficient as the PRF underlying EAX define $\mathrm{MAC}_K(H) = \text{Encrypt}_K^{0^n\,H}(\varepsilon)$.

COMPARISON WITH CCM. Figure 4 compares CCM and EAX along a few relevant dimensions. A description of CCM and an extended comparison can be found in the full version of this paper [8].

4 EAX2 Algorithm

To understand the the proof of security of EAX and the approach taken for its design, we introduce EAX2, a generic composition method. EAX is EAX2 for the particular case of CTR encryption and OMAC authentication, but then collapsed to a single key.

EAX2 COMPOSITION. Let $F\colon \mathsf{Key1} \times \{0,1\}^* \to \{0,1\}^n$ be a PRF, where $n \geq 2$. Let $\Pi = (\mathcal{E}, \mathcal{D})$ be an IV-based encryption scheme having key space $\mathsf{Key2}$ and IV space $\{0,1\}^n$. This means that $\mathcal{E}\colon\ \mathsf{Key2} \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^*$ and $\mathcal{D}\colon\ \mathsf{Key2} \times$

```
Algorithm EAX2.Encrypt$^{N\,H}_{K1,K2}\,(M)$
$\mathcal{N} \leftarrow F^{0}_{K1}(N)$
$\mathcal{H} \leftarrow F^{1}_{K1}(H)$
$C \leftarrow \mathcal{E}^{\mathcal{N}}_{K2}(M)$
$\mathcal{C} \leftarrow F^{2}_{K1}(C)$
$Tag \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
$T \leftarrow Tag$ [first $\tau$ bits]
return $CT \leftarrow C \parallel T$
```

```
Algorithm EAX2.Decrypt$^{N\,H}_{K1,K2}\,(CT)$
if $|CT| < \tau$ then return INVALID
Let $C \parallel T \leftarrow CT$ where $|T| = \tau$
$\mathcal{N} \leftarrow F^{0}_{K1}(N)$
$\mathcal{H} \leftarrow F^{1}_{K1}(H)$
$\mathcal{C} \leftarrow F^{2}_{K1}(C)$
$Tag' \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
$T' \leftarrow Tag'$ [first $\tau$ bits]
if $T \neq T'$ then return INVALID
$M \leftarrow \mathcal{D}^{\mathcal{N}}_{K2}(C)$
return $M$
```

Fig. 5. Encryption and decryption under EAX2. The mode is built from a PRF $F\colon$ Key1 $\times$ $\{0,1\}^* \to \{0,1\}^n$ and an IV-based encryption scheme $\Pi = (\mathcal{E}, \mathcal{D})$ having key space Key2 and message space $\{0,1\}^*$. The plaintext is M and the key is $(K1, K2)$ and the header is H. By F^i_K we mean the function where $F^i_K(M) = F_K([i]_n \parallel M)$.

$\{0,1\}^n \times \{0,1\}^* \to \{0,1\}^*$ and Key2 is a set of keys and for every $K \in$ Key2 and $\mathcal{N} \in \{0,1\}^n$ and $M \in \{0,1\}^*$, if $C = \mathcal{E}^{\mathcal{N}}_K(M)$ then $\mathcal{D}^{\mathcal{N}}_K(C) = M$. Let $\tau \leq n$ be a number. Now given F and Π and τ we define an AEAD scheme EAX2$[\Pi, F, \tau]$ = (EAX2.Encrypt, EAX2.Decrypt) as follows. Set $F^t_K(M) = F_K([t]_n \parallel M)$. Set Key = Key1$\times$Key2. Then the encryption algorithm EAX2.Encrypt: Key$\times\{0,1\}^*\times\{0,1\}^* \to \{0,1\}^*$ and the decryption algorithm EAX2.Decrypt: Key $\times$ $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{$INVALID$\}$ are defined in Figure 5. Scheme EAX2$[\Pi, F, \tau]$ is provably secure under natural assumptions about Π and F. See Section 6.

EAX1 COMPOSITION. Let EAX1 be the single-key variant of EAX2 where one insists that Key = Key1 = Key2 and where one keys F, $\mathcal{E}$, and $\mathcal{D}$ with a single key $K \in$ Key. One associates to F and Π the scheme EAX1$[\Pi, F, \tau]$ that is defined as with EAX2 but where the one key K keys everything. Notice that EAX$[E, \tau]$ = EAX1$[\mathrm{CTR}[E], \mathrm{OMAC}[E], \tau]$. This is a useful way to look at EAX.

5 Definitions

AEAD SCHEMES. A *set of keys* is a nonempty set having a distribution (the uniform distribution when the set is finite). A (nonce-based) *authenticated-encryption with associated-data* (AEAD) scheme is a pair of algorithms $\mathbf{\Pi} = (\mathbf{E}, \mathbf{D})$ where $\mathbf{E}$ is a deterministic *encryption* algorithm $\mathbf{E}\colon$ Key $\times$ Nonce $\times$ Header $\times$ Plaintext $\to$ Ciphertext and a $\mathbf{D}$ is a deterministic *decryption* algorithm $\mathbf{D}\colon$ Key$\times$Nonce$\times$Header$\times$ Ciphertext $\to$ Plaintext $\cup$ $\{$INVALID$\}$. The *key space* Key is a set of keys while the *nonce space* Nonce and the *header space* Header (also called the space of *associated data*) are nonempty sets of strings. We write $\mathbf{E}^{N\,H}_K(M)$ for $\mathbf{E}(K, N, H, M)$ and $\mathbf{D}^{N\,H}_K(CT)$ for $\mathbf{D}(K, N, H, CT)$. We require that $\mathbf{D}^{N\,H}_K(\mathbf{E}^{N\,H}_K(M)) = M$ for all $K \in$ Key and $N \in$ Nonce and $H \in$ Header and $M \in$ Plaintext. In this note we assume, for notational simplicity, that Nonce, Header, Plaintext, and Ciphertext are all

$\{0,1\}^*$ and that $|\mathbf{E}_K^{N\,H}(M)| = |M|$. An adversary is a program with access to one or more oracles.

NONCE-RESPECTING. Suppose A is an adversary with access to an *encryption oracle* $\mathbf{E}_K^{\cdot\,\cdot}(\cdot)$. This oracle, on input (N, H, M), returns $\mathbf{E}_K^{N\,H}(M)$. Let $(N_1, H_1, M_1), \ldots,$ (N_q, H_q, M_q) denote its oracle queries. The adversary is said to be *nonce-respecting* if $N_1, \ldots, N_q$ are always distinct, regardless of oracle responses and regardless of A's internal coins.

PRIVACY OF AEAD SCHEMES. We consider adversaries with access to an encryption oracle $\mathbf{E}_K^{\cdot\,\cdot}(\cdot)$. We assume that any privacy-attacking adversary is nonce-respecting. The advantage of such an adversary A in violating the privacy of AEAD scheme $\mathbf{\Pi} = (\mathbf{E}, \mathbf{D})$ having key space Key is

$$\mathbf{Adv}_{\mathbf{\Pi}}^{\mathbf{priv}}(A) \;=\; \Pr\left[K \xleftarrow{\$} \mathsf{Key} :\; A^{\mathbf{E}_K^{\cdot\,\cdot}(\cdot)} = 1\right] - \Pr\left[K \xleftarrow{\$} \mathsf{Key} :\; A^{\$^{\cdot\,\cdot}(\cdot)} = 1\right]$$

where $\$^{\cdot\,\cdot}(\cdot)$ denotes the oracle that on input (N, H, M) returns a random string of length $|M|$.

AUTHENTICITY OF AEAD SCHEMES. This time we provide the adversary with two oracles, an encryption oracle $\mathbf{E}_K^{\cdot\,\cdot}(\cdot)$ as above and also a *verification oracle* $\widehat{\mathbf{D}}_K^{\cdot\,\cdot}(\cdot)$. The latter oracle takes input (N, H, CT) and returns 1 if $\mathbf{D}_K^{N\,H}(CT) \in \mathsf{Plaintext}$ and returns 0 if $\mathbf{D}_K^{N\,H}(CT) = \textsc{Invalid}$. The adversary is assumed to satisfy three conditions, and these must hold regardless of the responses to its oracle queries and regardless of A's internal coins:

- Adversary A must be nonce-respecting. (The condition is understood to apply only to the adversary's encryption oracle. Thus a nonce used in an encryption-oracle query may be used in a verification-oracle query.)
- Adversary A may never make a verification-oracle query (N, H, CT) such that the encryption oracle previously returned CT in response to a query (N, H, M).
- Adversary A must call its verification-oracle exactly once, and may not subsequently call its encryption oracle. (That is, it makes a sequence of encryption-oracle queries, then a verification-oracle query, and then halts.)

We say that such an adversary *forges* if its verification oracle returns 1 in response to the single query made to it. The advantage of such an adversary A in violating the authenticity of AEAD scheme $\mathbf{\Pi} = (\mathbf{E}, \mathbf{D})$ having key space Key is

$$\mathbf{Adv}_{\mathbf{\Pi}}^{\mathbf{auth}}(A) \;=\; \Pr\left[K \xleftarrow{\$} \mathsf{Key} :\; A^{\mathbf{E}_K^{\cdot\,\cdot}(\cdot),\, \widehat{\mathbf{D}}_K^{\cdot\,\cdot}(\cdot)} \text{ forges}\right] \;.$$

IV-BASED ENCRYPTION. An *IV-based encryption scheme* (an IVE scheme) is a pair of algorithms $\mathit{\Pi} = (\mathcal{E}, \mathcal{D})$ where $\mathcal{E} :\; \mathsf{Key} \times \mathsf{IV} \times \mathsf{Plaintext} \to \mathsf{Ciphertext}$ is a deterministic *encryption* algorithm and $\mathcal{D} :\; \mathsf{Key} \times \mathsf{IV} \times \mathsf{Ciphertext} \to \mathsf{Plaintext} \cup \{\textsc{Invalid}\}$ is a deterministic *decryption* algorithm. The *key space* Key is a set of keys and the *plaintext space* $\mathsf{Plaintext}$ and *ciphertext space* $\mathsf{Ciphertext}$ and *IV space* IV are all nonempty sets of strings. We write $\mathcal{E}_K^R(M)$ for $\mathcal{E}(K, R, M)$ and $\mathcal{D}_K^R(C)$ for $\mathcal{D}(K, R, C)$. We require that $\mathcal{D}_K^R(\mathcal{E}_K^R(M)) = M$ for all $K \in \mathsf{Key}$ and $R \in \mathsf{IV}$ and $M \in \mathsf{Plaintext}$. We assume, as before, that $\mathsf{Plaintext} = \mathsf{Ciphertext} = \{0,1\}^*$ and that $|\mathcal{E}_K^R(M)| = |M|$. We also assume that $\mathsf{IV} = \{0,1\}^n$ for some $n \geq 1$ called the *IV length*.

Privacy of IVE schemes with random IVs. Let $\Pi = (\mathcal{E}, \mathcal{D})$ be an IVE scheme with key space Key and IV space $\mathsf{IV} = \{0,1\}^n$. Let $\mathcal{E}^\$$ be the probabilistic algorithm defined from $\mathcal{E}$ that, on input K and M, chooses an IV R at random from $\{0,1\}^n$, computes $C \leftarrow \mathcal{E}_K^R(M)$, and then returns C along with the chosen IV:

```
Algorithm $\mathcal{E}_K^\$(M)$        // The probabilistic encryption scheme built from IVE scheme $\mathcal{E}$
$R \stackrel{\$}{\leftarrow} \{0,1\}^n$ ; $C \leftarrow \mathcal{E}_K^R(M)$ ; return $R \,\|\, C$
```

Then we define the advantage of an adversary A in violating the privacy of Π (as an encryption scheme using random IV) by

$$\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A) \;=\; \Pr\left[K \stackrel{\$}{\leftarrow} \mathsf{Key} :\; A^{\mathcal{E}_K^\$(\cdot)} = 1\right] - \Pr\left[K \stackrel{\$}{\leftarrow} \mathsf{Key} :\; A^{\$(\cdot)} = 1\right]$$

where $\$(\cdot)$ denotes the oracle that on input M returns a random string of length $n+|M|$. This is just the ind\$-privacy of the randomized symmetric encryption scheme associated to Π. We comment that we have used a superscript of "priv" for an IVE scheme and "$\mathbf{priv}$" (bold font) for an AEAD scheme.

Pseudorandom functions. A *family of functions*, or a *pseudorandom function* (PRF), is a map $F\colon \mathsf{Key} \times D \to \{0,1\}^n$ where Key is a set of keys and D is a nonempty set of strings. We call n the *output length* of F. We write F_K for the function $F(K, \cdot)$ and we write $f \stackrel{\$}{\leftarrow} F$ to mean $K \stackrel{\$}{\leftarrow} \mathsf{Key}\,;\; f \leftarrow F_K$. We denote by $\mathcal{R}_n^*$ the set of all functions with domain $\{0,1\}^*$ and range $\{0,1\}^n$; by $\mathcal{R}_n^n$ the set of all functions with domain $\{0,1\}^n$ and range $\{0,1\}^n$; and by $\mathcal{R}_n^I$ the set of all functions with domain I and range $\{0,1\}^n$. We identify a function with its key, making $\mathcal{R}_n^n$, $\mathcal{R}_n^*$ and $\mathcal{R}_n^I$ pseudorandom functions. The advantage of adversary A in violating the pseudorandomness of the family of functions $F\colon \mathsf{Key} \times \{0,1\}^* \to \{0,1\}^n$ is

$$\mathbf{Adv}_{F}^{\mathrm{prf}}(A) \;=\; \Pr\left[K \stackrel{\$}{\leftarrow} \mathsf{Key} :\; A^{F_K(\cdot)} = 1\right] - \Pr\left[\rho \stackrel{\$}{\leftarrow} \mathcal{R}_n^* :\; A^{\rho(\cdot)} = 1\right]$$

A family of functions $E\colon \mathsf{Key} \times D \to \{0,1\}^n$ is a *block cipher* if $D = \{0,1\}^n$ and each E_K is a permutation. We let $\mathcal{P}_n$ denote all the permutations on $\{0,1\}^n$ and define

$$\mathbf{Adv}_{E}^{\mathrm{prp}}(A) \;=\; \Pr\left[K \stackrel{\$}{\leftarrow} \mathsf{Key} :\; A^{E_K(\cdot)} = 1\right] - \Pr\left[\pi \stackrel{\$}{\leftarrow} \mathcal{P}_n :\; A^{\pi(\cdot)} = 1\right]$$

Resources. If xxx is an advantage notion for which $\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}(A)$ has been defined we write $\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}(R)$ for the maximal value of $\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}(A)$ over all adversaries A that use resources at most R. When counting the resource usage of an adversary, one maximizes over all possible oracle responses, including those that could not be returned by any experiment we have specified for adversarial advantage. Resources of interest are: t—the running time; q—the total number of oracle queries; q_{e}—the number of oracle queries to the adversary's first oracle; q_{v}—the number of oracle queries to the adversary's second oracle; and σ—the data complexity. The running time t of an algorithm is its actual running time (relative to some fixed RAM model of computation) plus its description size (relative to some standard encoding of algorithms). The data complexity σ is defined as the sum of the lengths of all strings encoded in the adversary's oracle queries, plus the total number of all of these strings.[2] In this paper the length of strings is measured in n-bit blocks, for some understood value n. The number of blocks in a string M is defined as $\|M\|_n = \max\{1, \lceil |M|/n \rceil\}$, so that the empty string counts as one block. As an example, an adversary that asks queries $(N_1, H_1, M_1), (N_2, H_2, M_2)$ to its first oracle and query (N, H, M) to its second oracle has data complexity $\|N_1\|_n + \|H_1\|_n + \|M_1\|_n + \|N_2\|_n + \|H_2\|_n + \|M_2\|_n + \|N\|_n + \|H\|_n + \|M\|_n + 9$. The name of a resource measure (t, t', q, etc.) will be enough to make clear what resource it refers to.

When we use big-O notation it is understood that the constant hidden inside the notation may depend on n. We write $\widetilde{O}(f(x))$ for $O(f(x)\lg(f(x)))$. When F is a function we write $\mathrm{Time}_F(\sigma))$ for the maximal amount of time to compute the function F over inputs of total length σ. When $\mathit{\Pi} = (\mathcal{E}, \mathcal{D})$ is an AEAD scheme or an IVE scheme with key space Key we write $\mathrm{Time}_{\mathcal{E}}(\sigma)$ for the time to compute a random element $K \xleftarrow{\$} \mathsf{Key}$ plus the maximal amount of time to compute the function $\mathcal{E}_K$ on arguments of total length σ.

6 Security Results

We first obtain results about the security of EAX2 and then prove a result about the security of a tweakable-OMAC extension. These results are applied to derive results about the security of EAX. The notation and security measures referred to below are defined in Section 5.

SECURITY OF EAX2. We begin by considering the EAX2$[\mathit{\Pi}, F, \tau]$ scheme with F being equal to $\mathcal{R}_n^n$, the set of all functions with domain $\{0,1\}^n$ and range $\{0,1\}^n$. In other words, we are considering the case where F_{K1} is a random function with domain $\{0,1\}^n$ and range $\{0,1\}^n$. First we show that EAX2$[\mathit{\Pi}, \mathcal{R}_n^n, \tau]$ inherits the privacy of the underlying IVE scheme $\mathit{\Pi}$. The proof of the following is in the full version of this paper [8].

Lemma 1. [Privacy of EAX2 with a random PRF] *Let $\mathit{\Pi}$ be an IVE scheme with IV space $\{0,1\}^n$ and let $\tau \in [0..n]$. Then*

$$\mathbf{Adv}^{\mathbf{priv}}_{\mathrm{EAX2}[\mathit{\Pi}, \mathcal{R}_n^n, \tau]}(t, q, \sigma) \leq \mathbf{Adv}^{\mathrm{priv}}_{\mathit{\Pi}}(t', q, \sigma)$$

where $t' = t + \widetilde{O}(\sigma)$. □

We now turn to authenticity. The following shows that EAX2$[\mathit{\Pi}, \mathcal{R}_n^n, \tau]$ provides authenticity under the assumption that the underlying IVE scheme $\mathit{\Pi}$ provides privacy. The proof is in the full version of this paper [8].

[2] There is a certain amount of arbitrariness in this convention, but it is reasonable and simplifies subsequent accounting.

Lemma 2. [Authenticity of EAX2 with a random PRF] *Let $\mathit{\Pi}$ be an IVE scheme with IV space $\{0,1\}^n$ and let $\tau \in [0..n]$. Then*

$$\mathbf{Adv}^{\mathbf{auth}}_{\mathrm{EAX2}[\mathit{\Pi},\mathcal{R}^n_n,\tau]}(t,q,\sigma) \le \mathbf{Adv}^{\mathrm{priv}}_{\mathit{\Pi}}(t',q,\sigma) + 2^{-\tau}$$

where $t' = t + \widetilde{O}(\sigma)$. □

Our definition of authenticity allows the adversary only one query to its verification oracle, meaning only one forgery attempt. A standard argument says that the advantage of an adversary making q_{v} verification queries can grow by a factor of at most q_{v}. As per the above this means it is at most $q_{\mathrm{v}} \cdot [2^{-\tau} + \mathbf{Adv}^{\mathrm{priv}}_{\mathit{\Pi}}(t',q,\sigma)]$. We believe that in fact the bound is better than this, namely that it is $q_{\mathrm{v}} 2^{-\tau} + \mathbf{Adv}^{\mathrm{priv}}_{\mathit{\Pi}}(t',q,\sigma)$. However, we do not have a proof of this stronger bound.

The above allows us to obtain results about the security of the general $\mathrm{EAX2}[\mathit{\Pi},F,\tau]$ scheme based on assumptions about the security of the component schemes. The proof of the following is in the full version of this paper [8].

Theorem 1. [Security of EAX2] *Let $F\colon \mathsf{Key1} \times \{0,1\}^* \to \{0,1\}^n$ be a family of functions, let $\mathit{\Pi} = (\mathcal{E},\mathcal{D})$ be an IVE scheme with IV space $\{0,1\}^n$ and let $\tau \in [0..n]$. Then*

$$\mathbf{Adv}^{\mathbf{auth}}_{\mathrm{EAX2}[\mathit{\Pi},F,\tau]}(t,q,\sigma) \;\le\; \mathbf{Adv}^{\mathrm{priv}}_{\mathit{\Pi}}(t_2,q,\sigma) + \mathbf{Adv}^{\mathrm{prf}}_{F}(t_1,3q+3,\sigma) + 2^{-\tau} \quad (1)$$

$$\mathbf{Adv}^{\mathbf{priv}}_{\mathrm{EAX2}[\mathit{\Pi},F,\tau]}(t,q,\sigma) \;\le\; \mathbf{Adv}^{\mathrm{priv}}_{\mathit{\Pi}}(t_2,q,\sigma) + \mathbf{Adv}^{\mathrm{prf}}_{F}(t_3,3q,\sigma) \quad (2)$$

where $t_1 = t + \mathrm{Time}_{\mathcal{E}}(\sigma) + \widetilde{O}(\sigma)$ and $t_2 = t + \widetilde{O}(\sigma + nq)$ and $t_3 = t + \mathrm{Time}_{\mathcal{E}}(\sigma) + \widetilde{O}(\sigma)$. □

We remark that although "birthday" terms of the form $\sigma^2/2^n$ or $q^2/2^n$ do not appear explicitly in the bounds above, they may appear when we bound the $\mathbf{Adv}^{\mathrm{priv}}_{\mathit{\Pi}}(\cdot,\cdot,\cdot)$ and $\mathbf{Adv}^{\mathrm{prf}}_{F}(\cdot,\cdot,\cdot)$ in terms of their arguments.

SECURITY OF A TWEAKABLE-OMAC EXTENSION. This section develops the core result underlying why key-reuse "works" across OMAC and CTR modes. To do this, we consider the following extension of the tweakable-OMAC construction. Fix $n \ge 1$ and let $t \in \{0,1,2\}$ and $\rho \in \mathcal{R}^n_n$ and $M \in \{0,1\}^*$ and $s \in \mathbb{N}$. Then define

```
Algorithm $\mathbb{OMAC}_\rho(t,M,s)$
$R \leftarrow \mathrm{OMAC}^t_\rho(M)$
for $j \leftarrow 0$ to $s-1$ do $S_j \leftarrow \rho(R+j)$
return $R\,S_0S_1\cdots S_{s-1}$
```

Thus an $\mathbb{OMAC}_\rho$ oracle, when asked (t,M,s), returns not only $R = \mathrm{OMAC}^t_\rho(M)$ but also a key stream $S_0S_1\ldots S_s$ formed using CTR-mode and start-index R. We emphasize that the key stream is formed using the *same* function ρ (that is, the same key) that underlies the OMAC computation. Note too that we have limited the tweak t to a small set, $\{0, 1, 2\}$.

We imagine providing an adversary A with one of two kinds of oracles. The first is an oracle $\mathbb{OMAC}_\rho(\cdot,\cdot,\cdot)$ for a randomly chosen $\rho \in \mathcal{R}_n^n$. The second is an oracle $\$_n(\cdot,\cdot,\cdot)$ that, on input (t, M, s), returns $n(s + 1)$ random bits. Either way, we assume that the adversary is *length-committing*: if the adversary asks a query (t, M, s) it does not ask any subsequent query (t, M, s'). As the adversary runs, it asks some sequence of queries $(t_1, M_1, s_1), \ldots, (t_q, M_q, s_q)$. The resources of interest to us are the sum of the block lengths of the messages being MACed, $\sigma_1 = \sum \|M_i\|_n$, and the total number $\sigma_2 = \sum s_i$ of key-stream blocks that the adversary requests. We claim that a reasonable adversary will have little advantage in telling apart the two oracles, and we bound its distinguishing probability in terms of the resources σ_1 and σ_2 that it expends. Recall that for oracles X and Y and an adversary A we measure A's ability to distinguish between oracles X and Y by the number $\mathbf{Adv}^{\mathrm{dist}}_{X,Y}(A) = \Pr[A^X = 1] - \Pr[A^Y = 1]$. The proof of the following is in the full version of this paper [8].

Lemma 3. [Pseudorandomness of $\mathbb{OMAC}$] *Fix $n \geq 2$. Then, for length-committing adversaries,*

$$\mathbf{Adv}^{\mathrm{dist}}_{\mathbb{OMAC}[\mathcal{R}_n^n],\$_n}(\sigma_1, \sigma_2) \leq \frac{(\sigma_1 + \sigma_2 + 3)^2}{2^n} \qquad \square$$

SECURITY OF EAX. We are now ready to consider the security of EAX. The proof of the following is in the full version of this paper [8].

Theorem 2. [Security of EAX] *Let $n \geq 2$ and $\tau \in [0..n]$. Then*

$$\begin{aligned} \mathbf{Adv}^{\mathbf{priv}}_{\mathrm{EAX}[\mathcal{R}_n^n,\tau]}(\sigma) &\leq \frac{9\,\sigma^2}{2^n} \\ \mathbf{Adv}^{\mathbf{auth}}_{\mathrm{EAX}[\mathcal{R}_n^n,\tau]}(\sigma) &\leq \frac{10.5\,\sigma^2}{2^n} + \frac{1}{2^\tau} \qquad \square \end{aligned}$$

Finally, we may, in the customary way, pass to the corresponding complexity-theoretic result where we start with an arbitrary block cipher E.

Corollary 1. [Security of EAX] *Let $n \geq 2$ and $E\colon \mathsf{Key} \times \{0,1\}^n \times \{0,1\}^n$ be a block cipher and let $\tau \in [0..n]$. Then*

$$\begin{aligned} \mathbf{Adv}^{\mathbf{priv}}_{\mathrm{EAX}[E,\tau]}(t,\sigma) &\leq \frac{9.5\,\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}}_E(t',\sigma) \\ \mathbf{Adv}^{\mathbf{auth}}_{\mathrm{EAX}[E,\tau]}(t,\sigma) &\leq \frac{11\,\sigma^2}{2^n} + \frac{1}{2^\tau} + \mathbf{Adv}^{\mathrm{prp}}_E(t',\sigma) \end{aligned}$$

where $t' = t + O(\sigma)$. □

We omit the proof, which is completely standard.

References

1. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997.
2. M. Bellare, R. Guérin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. *Advances in Cryptology – CRYPTO '95*, Lecture Notes in Computer Science, vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.
3. M. Bellare, O. Goldreich, and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science, vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
4. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences* (JCSS), vol. 61, no. 3, pp. 362–399, Dec 2000.
5. M. Bellare, T. Kohno, and C. Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. *Proceedings of the 9th Annual Conference on Computer and Communications Security*, ACM, 2002.
6. 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.
7. 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.
8. M. Bellare, P. Rogaway and D. Wagner. The EAX mode of operation (A two-Pass authenticated-encryption scheme optimized for simplicity and efficiency). Full version of this paper, available via http://www.cs.ucdavis.edu/∼rogaway
9. J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. *Advances in Cryptology – CRYPTO '00*, Lecture Notes in Computer Science, vol. 1880, M. Bellare ed., Springer-Verlag, 2000.
10. N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, and T. Kohno. Helix: Fast encryption and authentication in a single cryptographic primitive. *Fast Software Encryption* (FSE 2003), Lecture Notes in Computer Science, vol. 2887, Springer-Verlag, pp. 330–346, 2003.
11. V. Gligor and P. Donescu. Integrity-aware PCBC encryption. Security Protocols, 7th International Workshop. Lecture Notes in Computer Science, vol. 1796, Springer-Verlag, pp. 153–171, 1999.
12. V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Presented at the 2nd NIST Workshop on AES Modes of Operation, Santa Barbara, CA, August 24, 2001.
13. P. Hawkes and G. Rose. Primitive specification for SOBER-128. Cryptology ePrint Archive Report 2003/48. April 2003.
14. T. Iwata and K. Kurosawa. OMAC: One-key CBC MAC. *Fast Software Encryption* (FSE 2003), Lecture Notes in Computer Science, vol. 2887, Springer-Verlag, pp. 129–153, 2003.
15. T. Iwata and K. Kurosawa. Personal communications, January 2002.
16. J. Jonsson. On the security of CTR + CBC-MAC. *Proceedings of Selected Areas of Cryptography (SAC)*, 2002.
17. C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology – EUROCRYPT '01*, Lecture Notes in Computer Science, vol. 2045 , B. Pfitzmann ed., Springer-Verlag, 2001.

18. J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption '00*, Lecture Notes in Computer Science, vol. 1978, B. Schneier ed., Springer-Verlag, 2000.
19. T. Kohno, J. Viega, and D. Whiting. A high-performance conventional authenticated encryption mode. These proceedings.
20. 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, J. Kilian ed., Springer-Verlag, 2001.
21. M. Liskov, R. Rivest, and D. Wagner. *Advances in Cryptology – CRYPTO '02*, Lecture Notes in Computer Science, vol. 2442, pp. 31–46, Springer-Verlag, 2002.
22. D. McGrew and J. Viega. Flexible and efficient message authentication in hardware and software. Manuscript, 2003. Available from http://www.zork.org/
23. E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *Journal of Cryptology*, vol. 13, no. 3 pp. 315–338, 2000.
24. P. Rogaway. Authenticated-encryption with associated-data. *Proceedings of the 9th Annual Conference on Computer and Communications Security* (CCS-9), pp. 98–107, ACM, 2002.
25. P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security* (TISSEC), vol. 6, no. 3, pp. 365–403, Aug. 2003.
26. D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). June 2002. Available at http://csrc.nist.gov/encryption/modes/proposedmodes/

A Definition of CCM

Since CCM [26] was a major motivation for our work, we recall its definition, writing it in a new form. First some notation. Write string constants in hexadecimal, as in $\texttt{0xFFFE}$. When $X \in \{0,1\}^\ell$ is a nonempty string and $i \in \mathbb{N}$ is a number we let $X + i$ be the ℓ-bit string that results from regarding X as a nonnegative number x (binary notation, msb first), adding x to i, taking the result modulo 2^n, and converting this number back into an ℓ-bit string. Now CCM depends on three parameters:

- E — the *block cipher* — where $E\colon \mathsf{Key} \times \{0,1\}^{128} \to \{0,1\}^{128}$
- $\boldsymbol{\tau}$ — the *tag length* — 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 is the AE scheme specified in Figure 6. The nonce space is $\mathsf{Nonce} = \textsc{Byte}^{15-\lambda}$ and the header space is $\mathsf{Header} = \textsc{Byte}^{<2^{64}}$ and the message space is $\mathsf{Plaintext} = \textsc{Byte}^{<2^{8\lambda}}$. There is a tradeoff between the length of nonces, $\eta = |N| = 15 - \lambda$ bytes, and the longest permitted message, $256^\lambda - 1$ bytes.

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

Algorithm CCM.Decrypt$_K^{N\ H}(CT)$
if $\|CT\|_n < \boldsymbol{\tau}$ then return INVALID
Partition $CT$ into $C \| T$ where $\|T\|_n = \boldsymbol{\tau}$
if $\|C\|_n > 2^{\lambda} - 1$ then return INVALID

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

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

Fig. 6. Encryption and decryption under $\mathrm{CCM}[E, \boldsymbol{\tau}, \lambda]$.