Authenticated Encryption

J. Black *

November 12, 2003

1 Introduction

Often when two parties communicate over a network, they have two main security goals: privacy and authentication. In fact, there is compelling evidence that one should never use encryption without also providing authentication [8, 14]. Many solutions for the privacy and authentication problems have existed for decades, and the traditional approach to solving both simultaneously has been to combine them in a straightforward manner using so-called "generic composition." However, recently there have been a number of new constructions which achieve both privacy and authenticity simultaneously, often much faster than any solution which uses generic composition. In this article we will explore the various approaches to achieving both privacy and authenticity, the so-called "Authenticated Encryption" problem. We will often abbreviate this as simply "AE." We will start with generic composition methods and then explore the newer combined methods.

Background. Throughout this article we will consider the AE problem in the "symmetric-key model." This means that we assume our two communicating parties, traditionally called "Alice" and "Bob," share a copy of some bit-string K, called the "key." This key is typically chosen at random and then distributed to Alice and Bob via one of various methods. This is the starting point for our work. We now wish to provide Alice and Bob with an AE algorithm such that Alice can select a message M from a pre-defined message-space, process it with the AE algorithm along with the key (and possibly a "nonce" N–a counter or random value), and then send the resulting output to Bob. The output will be the ciphertext C, the nonce N, and a short message authentication tag, σ. Bob should be able to recover M just given C, N, and his copy of the key K. He should also be able to certify that Alice was the originator by computing a verification algorithm using the above values along with the tag σ.

But what makes an AE algorithm "good?" We may have many requirements, and the relative importance of these requirements may vary according to the problem domain. Certainly one requirement is that the AE algorithm be "secure." We will speak more about what this means in a moment. But many other attributes of the algorithm may be important for us as well: performance, portability, simplicity/elegance, parallelizability, availability of reference implementations, or freedom from patents; we will pay attention to each of these concerns to varying levels as well.

Security. Certainly an AE scheme is not going to serve our needs unless it is secure. An AE scheme has two goals: privacy and authenticity. And each of these goals has a precise mathematical meaning [2, 3, 19]. In addition there is a precise definition for "authenticated encryption," the combination of both goals [5, 6, 26]. It would take us too far afield to carefully define each notion, but we will give a brief intuitive idea of what is meant. In our discussion we will use the term "adversary" to mean someone who is trying to subvert the security of the AE scheme, who knows the definition of the AE scheme, but who does not possess the key K.

Privacy means, intuitively, that a passive adversary who views the ciphertext C and the nonce N, cannot "understand" the content of the message M. One way to achieve this is to make C indistinguishable from random bits, and indeed this is one definition of security for an encryption scheme that is sometimes used, although it is quite a strong one.

* Department of Computer Science, 430 UCB, Boulder, Colorado 80309 USA. E-mail: jrblack@cs.colorado.edu WWW: www.cs.colorado.edu/∼jrblack/

Scheme	#Passes	Provably Secure	Assoc Data	Parallelizable	On-line	Patent-Free
IAPM	1	✓		✓	✓	
XECB	1	✓		✓	✓	
OCB	1	✓		✓	✓	
CCM	2	✓	✓			✓
EAX	2	✓	✓		✓	✓
CWC	2	✓	✓	✓	✓	✓
Helix	1		✓		✓	✓
SOBER-128	1		✓		✓	✓

Figure 1: A comparison of the various AE schemes. Generic composition is omitted since answers would depend on the particular instantiation. For the schemes which do not support associated data, subsequent methods have been suggested to remedy this; for example, see [32].

Authenticity means, intuitively, that an *active* adversary cannot successfully fabricate a ciphertext C, a nonce N, and a tag σ in such a way that Bob will believe that Alice was the originator. In the formal security model we allow the adversary to generate tags for messages of his choice as if he were Alice for some period of time, and then he must attempt a forgery. We do not give him credit for simply "replaying" a previously-generated message and tag, of course: he must construct a new value. If he does so with any significant probability of success, the authentication scheme is considered insecure.

Associated Data. In many application settings we wish not only to encrypt and authenticate message M, but we wish also to include auxiliary data H which should be authenticated, but left unencrypted. An example might be a network packet where the payload should be encrypted (and authenticated) but the header should be unencrypted (and authenticated). The reason being that routers must be able to read the headers of packets in order to know how to properly route them.

This need spurred some designers of AE schemes to allow "associated data" to be included as input to their schemes. Such schemes have been termed AEAD schemes (Authenticated Encryption with Associated Data), a notion which was first formalized by Rogaway [32]. As we will see, the AEAD problem is easily solved in the generic composition setting, but can become challenging when designing the more complex schemes. In his paper, Rogaway describes a few simple, but limited, ways to include associated data in any AE scheme, and then presents a specific method to efficiently add associated data to the OCB scheme, which we discuss below.

Provable Security. One unfortunate aspect of most cryptographic schemes is that we cannot prove that any scheme meets the formal goals required of it. However, we can prove *some* things related to security, but it depends on the *type* of cryptographic object we are analyzing. If the object is a "primitive," such as a block cipher, no proof of security is possible so instead we hope for security once we have shown that no known attacks (eg, differential cryptanalysis) seem to work. However, for algorithms which are built on top of these primitives, called "modes," we *can* prove some things about their security; namely that they are as secure as the primitives which underlie them. Almost all of the AE schemes we will describe here are modes; only two of them are primitives.

AE Schemes. The remainder of this article is devoted to the description and discussion of various AE algorithms. For convenience we list them in Figure 1. Note that we omit generic composition from the table since this approach comprises a class of schemes rather than a particular scheme.

Conventions. Let ϵ denote the empty string. Let Σ^n denote the set of all n-bit strings. In general, if S is a set we write S^+ to mean 1 or more repetitions of elements from S; that is, the set $\{s_1 s_2 \cdots s_m \mid m > 0,\ s_i \in S,\ 1 \leq i \leq m\}$. Thus $(\Sigma^n)^+$ is the set of all binary strings whose lengths are a positive multiple of n. If we write S^* we mean zero or more repetitions of elements from S. In other words, $S^* = S^+ \cup \{\epsilon\}$. We write $A \oplus B$ to mean the exclusive-or of strings A and B.

Many of our schemes use a block cipher. Throughout, n will be understood to be the block-size of the

underlying block cipher and k will be the size of its key. For block cipher E, we will write $E_K(P)$ to indicate invocation of block cipher E using the k-bit key K on the n-bit plaintext block P.

In order to process a message $M \in (\Sigma^n)^+$ we will often wish to break M into m strings, $M_1, \cdots, M_m$, each having n-bits such that $M = M_1 M_2 \cdots M_m$. For brevity, we will say "write $M = M_1 \cdots M_m$" and understand it to mean the above.

2 Generic Composition

Although AE did not get a formal definition until recently, the goal has certainly been implicit for decades. The traditional way of achieving both authenticity and privacy was to simply find an algorithm which yields each one and then use the combination of these two algorithms on our message. Intuitively it seems that this approach is obvious, straightforward, and completely safe. Unfortunately, there are many pitfalls accidentally "discovered" by well-meaning protocol designers.

One commonly-made mistake is the assumption that AE can be achieved by using a non-cryptographic non-keyed hash function h and a good encryption scheme like CBC mode (Cipher Block Chaining mode; see CBC-MAC and variants) with key K and initialization vector N. One produces $\text{CBC}_{K,N}(M, h(M))$ and hopes this yields a secure AE scheme. However, these schemes are virtually always broken. Perhaps the best-known example is the Wired Equivalent Privacy protocol (WEP) used with 802.11 wireless networks. This protocol instantiates h as a Cyclic Redundancy Code (CRC) and then uses a stream cipher to encrypt. Borisov, Goldberg, and Wagner showed, among other things, that it was easy to circumvent the authentication mechanism [15].

Another common pitfall is "key reuse." In other words, using some key K both for the encryption scheme and the MAC scheme. This approach appliedly blindly almost always fails. We will later see that all of our "combined modes," listed after this section, do in fact use a single key, but they are carefully designed to retain security in spite of this.

It is now clear to researchers that one needs to use a *keyed* hash (ie, a MAC) with some appropriate key $K1$ along with a secure encryption scheme with an independent key $K2$. However, it is unclear in what order these modes should be applied to a message M in order to achieve authenticated encryption. There are three obvious choices:

- **MtE:** MAC-then-Encrypt. We first MAC M under key $K1$ to yield tag σ and then encrypt the resulting pair (M, σ) under key $K2$.
- **EtM:** Encrypt-then-MAC. We first encrypt M under key $K2$ to yield ciphertext C and then compute $\sigma \leftarrow \text{MAC}_{K1}(C)$ to yield the pair (C, σ).
- **E&M:** Encrypt-and-MAC. We first encrypt M under key $K2$ to yield ciphertext C and then compute $\sigma \leftarrow \text{MAC}_{K1}(M)$ to yield the pair (C, σ).

Also note that decryption and verification are straightforward for each approach above: for MtE decrypt first, then verify. For EtM and E&M verify first, then decrypt.

Security. In 2000, Bellare and Namprempre gave formal definitions for AE [5], and then systematically examined each of the three approaches described above in this formal setting. Their results show that if the MAC has a property called "strongly unforgeable," then it possible to achieve the strongest definition of security for AE only via the EtM approach. They further show that some known-good encryption schemes fail to provide privacy in the AE setting when using the E&M approach, and fail to provide a slightly stronger notion of privacy with the MtE approach.

These theoretical results generated a great deal of interest since three major pre-existing protocols, SSL/TLS, IPSec, and SSH, each used a different one of these three approaches: the SSL/TLS protocol uses MtE, IPSec uses EtM, and SSH uses E&M. One might think that perhaps security flaws exist in SSL/TLS and SSH because of the results of Bellare and Namprempre, however concurrent with their work, Krawczyk showed that SSL/TLS was in fact secure because of the encoding used alongside the MtE mechanism [29]. And later Bellare, Kohno, and Namprempre showed that despite some identified security-flaws in SSH, it could be made provably-secure via a number of simply modifications despite its E&M approach.

The message here is that EtM with a provably-secure encryption scheme and a provably-secure MAC each with independent keys is the best approach for achieving AE. Although MtE and E&M can be secure, security will often depend on subtle details of how the data are encoded and on the particular MAC and encryption schemes used.

PERFORMANCE. Simple methods for doing very fast encryption have been known for quite some time. For example, CBC mode encryption has very little overhead beyond the calls to the block cipher. Even more attractive is CTR mode (CounTeR mode; see counter mode), which similarly has little overhead and in addition is parallelizable. However, MACing quickly is not so simple. The CBC MAC (Cipher Block Chaining Message Authentication Code; see CBC-MAC and variants) is quite simple and just as fast as CBC mode encryption, but there are well-known ways to go faster. The fastest software MAC in common use today is HMAC [1, 20]. HMAC uses a cryptographic hash function to process the message M and this is faster than processing M block-by-block with a block cipher. However even faster approaches have been invented using the Wegman-Carter construction [34]. This approach involves using a non-cryptographic hash function to process M, and then uses a cryptographic function to process the hash output. The non-cryptographic hash is randomly selected from a carefully-designed family of hash functions, all with a common domain and range. The goal is to produce a family such that distinct messages are unlikely to hash to the same value when the hash function is randomly chosen from that family. This is the so-called "universal hash family" [16]. The fastest known MACs are based on the Wegman-Carter approach. The speed champions are UMAC [11] and hash127 [10], though neither of these are in common use yet.

ASSOCIATED DATA. As we mentioned in the introduction, it is a common requirement in cryptographic protocols that we allow authenticated but non-encrypted data to be included in our message. Although the single-pass modes we describe next do not naturally allow for associated data, due to the fact that their encryption and authentication methods are intricately interwoven, we do not have this problem with generically-composed schemes. Since the encryption and MAC schemes are entirely independent, we simply run the MAC on all the data and run the encryption scheme only on the data to be kept private.

CAN WE DO BETTER? One obvious question when considering generically-composed AE schemes is "can we do better?" In other words, might there be a way of achieving AE without using two different algorithms, with two different keys, and making two separate passes over the message. The answer is "yes," and a discussion of these results comprise the remainder of this article.

3 Single-Pass Combined Modes

It had long been a goal of cryptographers to find a mode of operation which achieved AE using only a single pass over the message M. Many attempts were made at such schemes, but all were broken. Therefore until the year 2000, people still used generic composition to achieve AE, which as we have seen requires two passes over M.

3.1 IAPM

In 2000, Jutla of IBM invented two schemes which were the first correct single-pass AE modes [25]. He called these modes IACBC (Integrity-Aware Cipher Block Chaining) and IAPM (Integrity-Aware Parallelizable Mode). The first mode somewhat resembles CBC-mode encryption, however offsets were added in before and after each block-cipher invocation, a technique known as "whitening." However, as we know, CBC-mode encryption is inherently serial: we cannot begin computation for the $k+1$-st block-cipher invocation until we have the result of the k-th invocation. Therefore, more interest has been generated around the second mode, IAPM, which does not have this disadvantage. Let's look at how IAPM works.

IAPM accepts a message $M \in (\Sigma^n)^+$, a nonce $N \in \Sigma^n$, and a key pair $K1, K2$ each selected from Σ^k for use with the underlying block cipher E. The key pair is set up and distributed in advance between the communicating parties; the keys are reused for a large number of messages. However, N and (usually) M vary with each transmission. First we break M into $M_1 \cdots M_{m-1}$ and proceed as follows.

There are two main steps: (1) offset-generation and (2) encryption/tag-generation. For offset generation we encipher N to get a seed value, and then encipher sequential seed values to get the remaining seed values. In other words, set $W_1 \leftarrow E_{K2}(N)$ and then set $W_i \leftarrow E_{K2}(W_1 + i - 2)$ for $2 \le i \le t$ where $t = \lceil \lg(m+2) \rceil$. Here lg means $\log_2$, so if we had a message M with 256 n-bit blocks, we would require $\lceil \lg(259) \rceil = 9$ block-cipher invocations to generate the W_i values. Finally, to derive our $m+1$ offsets from the seed values, for i from 1 to $m+1$, we compute $S_{i-1} \leftarrow \bigoplus_{j=1}^{t} (i[j] \cdot W_j)$ where $i[j]$ is the j-th bit of i.

Armed with S_0 through S_m we are now ready to process M. First we encrypt each block of M by computing $C_i \leftarrow E_{K1}(M_i \oplus S_i) \oplus S_i$ for $1 \le i \le m-1$. This xoring of S_i before and after the block-cipher invocation is the whitening we spoke of previously, and is the main idea in all schemes discussed in this section. Next we compute the authentication tag σ: set $\sigma \leftarrow E_{K1}(S_m \oplus \bigoplus_{i=1}^{m-1} M_i) \oplus S_0$. Notice that we are whitening the simple sum of the plaintext blocks with two *different* offset values, S_0 and S_m. Finally, output $(N, C_1, \cdots, C_{m-1}, \sigma)$ as the authenticated ciphertext. Note that the output length is two n-bit blocks longer than M. This "ciphertext expansion," comparable to what we saw with generic composition, is quite minimal.

Given the $K1$, $K2$, and some output $(N, C_1, \cdots, C_{m-1}, \sigma)$, it is fairly straightforward to recover M and check the authenticity of the transmission. Notice that N is sent in the clear and so using $K2$ we can compute the W_i values and therefore the S_i values. We compute $M_i \leftarrow E_{K1}^{-1}(C_i \oplus S_i) \oplus S_i$ for $1 \le i \le m-1$ to recover M. Then we check $E_{K1}(S_m \oplus \bigoplus_{i=1}^{m-1} M_i) \oplus S_0$ to ensure it matches σ. If we get a match, we accept the transmission as authentic, and if not we reject the transmission as an attempted forgery.

Comments on IAPM. Compared to generic composition, where we needed about $2m$ block-cipher invocations per message (assuming our encryption and authentication modes were block-cipher-based), we are now using only around $m \lg(m)$ invocations. Further refinements to IAPM reduce this even more, so the number of block-cipher invocations is nearly m in these optimized versions meaning that one can achieve AE at nearly the same cost of encryption alone.

Proving a scheme like IAPM secure is not a simple task, and indeed we cannot present such a proof here. The interested reader is encouraged to read Halevi's article which contains a rigorous proof that if the underlying block cipher is secure, then so are IACBC and IAPM [21].

3.2 XCBC and OCB

Quickly after announcement of IACBC and IAPM other researchers went to work on finding similar single-pass AE schemes. Soon two other parties announced similar schemes: Gligor and Donescu produced a host of schemes, each with various advantages and disadvantages [18], and Rogaway, Bellare, Black, and Krovetz announced their OCB scheme [33], which is similar to IAPM but with a long list of added optimizations.

Gligor and Donescu presented two classes of schemes: XCBC and XECB. XCBC is similar to CBC mode encryption just as IACBC was above, and XECB is similar to ECB mode encryption which allows parallelism to be exploited, much like the IAPM method presented above. Since many practitioners desire parallelizable modes, the largest share of attention has been paid to XECB. Similar to IAPM, XECB uses an offset to each message block, applied before and after a block cipher invocation. However, XECB generates these offsets in a very efficient manner, using arithmetic mod 2^n, which is very fast on most commodity processors. Once again, both schemes are highly-optimized and provide AE at a cost very close to that of encryption alone. Proofs of security are included in the paper, using the reductionist approach we described above.

Rogaway, Bellare, Black, and Krovetz produced a single scheme called OCB (Offset CodeBook). This work was a follow-on to Jutla's IAPM scheme, designed to be fully-parallelizable, along with a long list of other improvements. In comparison to IAPM, OCB uses a single block-cipher key, provides a message space of Σ^* so we never have to pad, and is nearly endian-neutral. Once again, a full detailed proof of security is included in the paper, demonstrating that the security of OCB is directly related to the security of the underlying block cipher.

OCB is no-doubt the most aggressively-optimized scheme of those discussed in this section. Performance tests indicate that OCB is about 6.4% slower than CBC mode encryption, and this is without exploiting the parallelism that OCB offers up. For more information, one can find an in-depth FAQ, all relevant publications, reference code, test vectors, and performance figures on the OCB web page at `http://www.cs.ucdavis.edu/~rogaway/ocb/`.

Associated Data. In many settings, the ability to handle associated data is crucial. Rogaway [32] suggests methods to handle associated data in all three of the single-pass schemes mentioned above, and for OCB gives an extension which uses PMAC [13] to give a particularly efficient variant of OCB which handles associated data.

Intellectual Property. Given the importance of these new highly-efficient AE algorithms, all of the authors decided to file for patents. Therefore, IBM, Gligor, and Rogaway all have intellectual property claims for their algorithms and perhaps on some of the overriding ideas involved. To date, none of these patents has been tested in court, so the extent to which they are conflicting or interrelated is unclear. One effect, however, is that many would-be users of this new technology are worried that the possible legal entanglements are not worth the benefits offered by this technology. Despite this, OCB has appeared in the 802.11 draft standard as an alternate mode, and has been licensed several times. However, without IP claims it is possible all of these algorithms would be in common use today.

It was the complications engendered by the IP claims which spurred new teams of researchers to find further efficient AE algorithms which would not be covered by patents. Although not as fast as the single-pass modes described here, they still offer significant performance improvements over generic composition schemes. These schemes include CCM, CWC, and EAX, the latter invented in part by two researchers from the OCB team. We discuss these schemes next.

4 Two-Pass Combined Modes

If we have highly-efficient single-pass AE modes, why would researchers subsequently work to develop less efficient multi-pass AE schemes? Well, as we just discussed, this work was entirely motivated by the desire to provide patent-free AE schemes. The first such scheme proposed was CCM (CBC MAC with Counter Mode) by Ferguson, Housley and Whiting. Citing several drawbacks to CCM, Bellare, Rogaway, and Wagner proposed EAX, another patent-free mode which addresses these drawbacks. And independently, Kohno, Viega, and Whiting proposed the CWC mode (Carter-Wegman with Counter mode encryption). CWC is also patent-free and, unlike the previous two modes, is fully parallelizable. We now discuss each of these modes in turn.

4.1 CCM Mode

CCM was designed with AES specifically in mind. It therefore is hard-coded to assume a 128-bit block size, though it could be recast for other block sizes. Giving all the details of the mode would be cumbersome, so we will just present the overriding ideas. For complete details, see the CCM specification [35].

CCM is parameterized. It requires that you specify a 128-bit block-cipher (eg, AES), a tag length (which must be one of 4, 6, 8, 10, 12, 14, or 16), and the message-length field's size (which induces an upperbound on the message length). Like all other schemes we mention, CCM uses a nonce N each time it is invoked, and the size of N depends on the the parameters chosen above; specifically, if we choose a longer maximum message-length, we must accept a shorter nonce. It is left to the user to decide which parameters to use, but typical values might be to limit the maximum message length to 16 MBytes and then use a 96-bit nonce.

Once the parameters are decided, we invoke CCM by providing four inputs: the key K which will be used with AES, the nonce N of the proper size, associated data H which will be authenticated but not encrypted, and the plaintext M which will be authenticated and encrypted. CCM operates in two passes: first we encode the above parameters into an initial block, prepend this block to H and M, and then run CBC MAC over this entire byte-string using K. This yields the authentication tag σ. (The precise details of how the above concatenation is done *are* important for the security of CCM, but are omitted here.)

Next we form a counter-value using one of the scheme's parameters along with N and any necessary padding to reach 128 bits. This counter is then used with CTR mode encryption on $(\sigma \parallel M)$ under K to produce the ciphertext. The first 128 bits are the authentication tag, and we return the appropriate number of bytes according to the tag-length parameter. The subsequent bytes are the encryption of M and are always included in the output.

Decryption and verification are quite straightforward: N produces the counter-value and allows the recovery of M. Re-running CBC MAC on the same input used above allows verification of the tag.

COMMENTS ON CCM. It would seem that CCM is not much better than simple generic composition; after all, it uses a MAC scheme (the CBC MAC) and an encryption scheme (CTR mode encryption), which are both well-known and provably-secure modes. But CCM *does* offer advantages over the straightforward use of these two primitives generically composed; in particular it uses the same key K for both the MAC and the encryption steps. Normally this practice would be very dangerous and unlikely to work, but the designers were careful to ensure the security of CCM despite this normally-risky practice. The CCM specification does not include performance data or a proof of security. However, a rigorous proof was published by Jonsson [24]. CCM is currently the mandatory mode for the 802.11 wireless standard as well as currently being considered by NIST as a FIPS standard.

4.2 EAX Mode

Subsequent to the publication and subsequent popularity of CCM, three researchers decided to examine the shortcomings of CCM and see if they could be remedied. Their offering is called EAX [7] and addresses several perceived problems with CCM, including the following:

1. If the associated data field is fixed from message to message, CCM does not take advantage of this, but rather re-processes this data anew with each invocation.
2. Message lengths must be known in advance because the length is encoded into the first block before processing begins. This is not a problem in some settings, but in many applications we do not know the message length in advance.
3. The parameterization is awkward and, in particular, the trade-off between maximum message length and the size of the nonce seems unnatural.
4. The definition of CCM (especially the encodings of the parameters and length information in the message before it is processed) is complex and difficult to understand. Moreover, the correctness of CCM strongly depends on the details of this encoding.

Like CCM, EAX is a combination of a type of CBC MAC and CTR mode encryption. However, unlike CCM, the MAC used is not raw CBC MAC, but rather a variant. Two well-known problems exist with CBC MAC: (1) all messages must be of the same fixed length, and (2) that length must be a positive multiple of n. If we violate the first property, security is lost. Several variants to the CBC MAC have been proposed to address these problems: EMAC [9, 31] adds an extra block-cipher call to the end of CBC MAC to solve problem (1). Not to be confused with the AE mode of the same name above, XCBC [12] solves both problems (1) and (2) without any extra block-cipher invocations, but requires $k + 2n$ key bits. Finally, OMAC [23] improves XCBC so that only k bits of key are needed. The EAX designers chose to use OMAC with an extra input called a "tweak" which allows them to essentially get several different MACs by using distinct values for this tweak input. This is closely-related to an idea of Liskov, Rivest, and Wagner who introduced tweakable block ciphers [30].

We now describe EAX at a high level. Unlike CCM, the only EAX parameters are the choice of block cipher, which may have any block size n, and the number of authentication tag bits to be output, τ. To invoke EAX, we pass in a nonce $N \in \Sigma^n$, a header $H \in \Sigma^*$ which will be authenticated but not encrypted, and the message $M \in \Sigma^*$ which will be authenticated and encrypted, and finally the key K, appropriate for the chosen block cipher. We will be using OMAC under key K three times, each time with a different tweak, written OMAC^0_K, OMAC^1_K, and OMAC^2_K; it's conceptually easiest to think of these three OMAC invocations as three separate MACs, although this is not strictly true. First, we compute $\text{ctr} \leftarrow \text{OMAC}^0_K(N)$ to obtain the counter value we will use with CTR mode encryption. Then we compute $\sigma_H \leftarrow \text{OMAC}^1_K(H)$ to get an authentication tag for H. Then we encrypt and authenticate M with $C \leftarrow \text{OMAC}^2_K(\text{CTR}^{\text{ctr}}_K(M))$. And finally we output the first τ bits of $\sigma = (\text{ctr} \oplus C \oplus \sigma_H)$ as the authentication tag. We also output the nonce N, the associated data H, and the ciphertext C. The decryption and verification steps are quite straightforward.

Note that each of the problem areas cited above have been addressed by the EAX mode: no restriction on message length, no inter-dependence between the tag length and maximum message length, a performance savings when there is static header data, and no need for message length to be known up-front. Also, EAX

is arguably simpler to specify and implement. Once again, proving EAX secure is more difficult than just appealing to proofs of security for generically-composed schemes since the key K is re-used in several contexts which is normally not a safe practice.

4.3 CWC Mode

The CWC Mode [28] is also a two-pass mode: it uses a Wegman-Carter MAC along with CTR mode encryption under a common key K. It's main advantage over CCM and EAX is that it is parallelizable whereas the prior two are not (due to their use of the inherently-sequential CBC MAC type algorithms). Also, CWC strives to be very fast in hardware, a consideration which was not given nearly as much attention in the design of the other modes. In fact, the CWC designers claim that CWC should be able to encrypt and authenticate data at 10Gbps in hardware, whereas CCM and EAX will be limited to about 2Gbps because of their serial constraints.

As we discussed above in the section on generic composition, Wegman-Carter MACs require one specify a family of hash functions on a common domain and range. Typically we want these functions to (1) be fast-to-compute and (2) have a low collision probability. The CWC designers also looked for a family with additional properties: (3) parallelizability, and (4) good performance in hardware. The function family they settled on is the well-known polynomial-hash. Here a function from the family is named by choosing a value for x in some specified range, and then the polynomial

$$Y_1 x^{\ell} + Y_2 x^{\ell-1} + \cdots + Y_{\ell} x + Y_{\ell+1}$$

is computed modulo some integer, typically a prime. The specific family chosen by the CWC designers fixes $Y_1, \cdots, Y_{\ell}$ to be 96-bit integers, and $Y_{\ell+1}$ to be a 127-bit integer; their values are determined by the message being hashed. The modulus is set to the prime, $2^{127} - 1$.

Although it is possible to evaluate this polynomial quickly on a serial machine using Horner's Method (and in fact, this may make sense in some cases), it is also possible to exploit parallelism in the computation of this polynomial. Assume n is odd and set $m = (n-1)/2$ and $y = x^2 \bmod 2^{127} - 1$. Then we can rewrite the function above as

$$\left(Y_1 y^m + Y_3 y^{m-1} + \cdots + Y_{\ell}\right) x + \left(Y_2 y^m + Y_4 y^{m-1} + \cdots + Y_{\ell+1}\right) \bmod 2^{127} - 1.$$

This means that we can sub-divide the work for evaluating this polynomial and then recombine the results using addition modulo $2^{127} - 1$. Building a MAC from this hash family is fairly straightforward, and therefore CWC yields a parallelizable scheme since CTR is clearly parallelizable.

The CWC designers go on to provide benchmark data to compare CCM, EAX, and CWC on a Pentium III, showing that the speed differences are not that significant. However, this is without exploiting any parallelism available with CWC. They do not compare the speed of CWC with that of OCB, where we would expect OCB to be faster even in parallel implementations.

CWC comes with a rigorous proof of security via a reduction to the underlying 128-bit block cipher (typically AES), and the paper includes a readable discussion of why the various design choices were made. In particular, it does not suffer from any of the above-mentioned problems with CCM.

5 AE Primitives

Every scheme discussed to this point has been a mode of operation. In fact with the possible exception of some of the MAC schemes, every mode has used a block cipher as its underlying primitive. In this section we consider two recently-developed modes which are stream ciphers which provide authentication in addition to privacy. That is to say, these are *primitives* which provide AE.

This immediately means there is no proof of their security, nor is there likely to ever be one. The security of primitives is usually a matter of opinion: does the object withstand all known attacks? Has it been in use for a long enough time? Have good cryptanalysts examined it?

With new objects, it is often hard to know how much trust to place in their security. Sometimes the schemes break, and sometimes they do not. We will discuss two schemes in this section: Helix and SOBER-128. Both were designed by teams of experienced cryptographers who paid close attention to their security as well as to their efficiency.

5.1 Helix

Helix was designed by Ferguson, Whiting, Schneier, Kelsey, Lucks, and Kohno [17]. Their goal was to produce a fast, simple, patent-free stream cipher which also provided authentication. The team claims speeds of about 7 cycles per byte on a Pentium II, which is quite a bit faster than the fastest-known implementations of AES, which run at about 15 cycles per byte. At first glance this might be quite surprising: after all, AES does about 160 table look-ups and 160 32-bit XORs to encipher 16 bytes. This means AES uses about 10 look-ups and 10 XORs per byte. As we will see in a moment, Helix uses more operations than this per-byte! But a key difference is that AES does memory look-ups from large tables which perhaps are not in cache whereas Helix confines its work to the register file.

Helix takes a key K up to 32 bytes in length, and a 16-byte nonce N and a message $M \in (\Sigma^8)^*$. As usual, K will allow the encryption of a large amount of data before it needs to be changed, and N will be issued anew with each message encrypted, never to repeat throughout the life of K. Helix uses only a few simple operations: addition modulo 2^{32}, exclusive-or of 32-bit strings, and bitwise rotations. However, each iteration of Helix, called a "block," uses 11 XORs, 12 modular additions, and 20 bitwise rotations by fixed amounts on 32-bit words. So Helix is not simple to specify; instead we give a high-level description.

Helix keeps its "state" in five 32-bit registers (the designers were thinking of the Intel family of processors). The i-th block of Helix emits one 32-bit word of key-stream S_i, requires two 32-bit words scheduled from K and N, and also requires the i-th plaintext word M_i. It is highly-unusual for a stream cipher to use the plaintext stream as part of its key-stream generation, but this feature is what allows Helix to achieve authentication as well as generating a key-stream.

As usual, the key-stream is used as a one-time pad to encrypt the plaintext. In other words, the i-th ciphertext block C_i is simply $M_i \oplus S_i$. The 5-word state resulting from block i is then fed into block $i+1$ and the process continues until we have a long enough key-stream to encrypt M. At this point, a constant is XORed into one of the words of the resulting state, twelve more blocks are generated using a fixed plaintext word based on the length of M, with the key-stream of the four last blocks yielding the 128-bit authentication tag.

5.2 SOBER-128

A competitor to Helix is an offering from Hawkes and Rose called SOBER-128 [22]. This algorithm evolved from a family of simple stream ciphers (ie, ciphers which did not attempt simultaneous authentication) called the SOBER family, the first of which was introduced in 1998 by Rose. SOBER-128 retains many of the characteristics of its ancestors, but introduces a method for authenticating messages as well. We will not describe the internals of SOBER-128 but rather describe a few of its attributes at a higher level.

SOBER-128 uses a linear-feedback shift register in combination with several non-linear components, in particular a carefully-designed S-box which lies at its heart. To use SOBER-128 for AE one first generates a keystream used to XOR with the message M and then uses a separate API call "maconly" to process the associated data. The method of feeding back plaintext into the keystream generator is modeled after Helix, and the authors are still evaluating whether this change to SOBER-128 might introduce weaknesses.

Tests by Hawkes and Rose indicate that SOBER-128 is comparable in speed to Helix, however both are quite new and are still undergoing cryptanalytic scrutiny—a crucial process when designing primitives. Time will help us determine their security.

6 Beyond AE and AEAD

Real protocols often require more than just an AE scheme or an AEAD scheme: perhaps they require something that more resembles a network transport protocol. Desirable properties might include resistance to replay and prevention against packet loss or packet reordering. In fact, protocols like SSH aim to achieve precisely this.

Work is currently underway to extend AE notions to encompass a broader range of such goals [27]. This is an extension to the SSH analysis referred to above [4], but considers the various EtM, MtE, and E&M approaches rather than focusing on just one. Such research is another step in closing the gap between what cryptographers produce and what consumers of cryptographic protocols require. The hope is that

we will reach the point where methods will be available to practitioners which relieve them from inventing cryptography (which, as we have seen, is a subtle area with many insidious pitfalls) and yet allow them easy access to provably-secure cryptographic protocols. We anticipate further work in this area.

7 Notes on the Bibliography

Note that AE and its extensions continue to be an active area of research. Therefore, many of the bibliographic references are currently to unpublished pre-prints of works in progress. It would be prudent for the reader to look for more mature versions of many of these research reports to obtain the latest revisions.

References

[1] Bellare, M., Canetti, R., and Krawczyk, H. Keying hash functions for message authentication. In *Advances in Cryptology – CRYPTO '96* (1996), vol. 1109 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–15.

[2] Bellare, M., Desai, A., Pointcheval, D., and Rogaway, P. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – CRYPTO '98* (1998), H. Krawczyk, Ed., vol. 1462 of *LNCS*, Springer-Verlag, pp. 232–249.

[3] Bellare, M., Kilian, J., and Rogaway, P. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences (JCSS) 61*, 3 (Dec. 2000), 362–399. Earlier version in CRYPTO '94. See `www.cs.ucdavis.edu/~rogaway`.

[4] Bellare, M., Kohno, T., and Namprempre, C. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In *ACM Conference on Computer and Communications Security (CCS-9)* (2002), ACM Press, pp. 1–11.

[5] Bellare, M., and Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology – ASIACRYPT '00* (2000), vol. 1976 of *Lecture Notes in Computer Science*, Springer-Verlag.

[6] Bellare, M., and Rogaway, P. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient encryption. In *Advances in Cryptology — ASIACRYPT '00* (2000), T. Okamoto, Ed., vol. 1976 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 317–330. See `www.cs.ucdavis.edu/~rogaway`.

[7] Bellare, M., Rogaway, P., and Wagner, D. EAX: A conventional authenticated-encryption mode. Cryptology ePrint archive, reference number 2003/069, submitted Apr 13, 2003, revised Sep 9, 2003, 2003. See `eprint.iacr.org`.

[8] Bellovin, S. Problem areas for the IP security protocols. In *Proceedings of the Sixth USENIX Security Symposium* (July 1996), pp. 1–16.

[9] Berendschot, A., den Boer, B., Boly, J., Bosselaers, A., Brandt, J., Chaum, D., Damgård, I., Dichtl, M., Fumy, W., van der Ham, M., Jansen, C., Landrock, P., Preneel, B., Roelofsen, G., de Rooij, P., and Vandewalle, J. *Final Report of Race Integrity Primitives*, vol. 1007 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[10] Bernstein, D. Floating-point arithmetic and message authentication. Available from `http://cr.yp.to/hash127.html`.

[11] Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P. UMAC: Fast and secure message authentication. In *Advances in Cryptology – CRYPTO '99* (1999), Lecture Notes in Computer Science, Springer-Verlag.

[12] Black, J., and Rogaway, P. CBC MACs for arbitrary-length messages: The three-key constructions. In *Advances in Cryptology – CRYPTO '00* (2000), Lecture Notes in Computer Science, Springer-Verlag.

[13] Black, J., and Rogaway, P. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology — EUROCRYPT 2002* (2002), L. Knudsen, Ed., vol. 2332 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 384–397.

[14] Black, J., and Urtubia, H. Side-channel attacks on symmetric encryption schemes: The case for authenticated encryption. In *Proceedings of the Eleventh USENIX Security Symposium* (Aug. 2002), D. Boneh, Ed., pp. 327–338.

[15] Borisov, N., Goldberg, I., and Wagner, D. Intercepting mobile communications: The insecurity of 802.11. In *MOBICOM* (2001), ACM, pp. 180–189.

[16] Carter, L., and Wegman, M. Universal hash functions. *J. of Computer and System Sciences*, 18 (1979), 143–154.

[17] Ferguson, N., Whiting, D., Schneier, B., Kelsey, J., Lucks, S., and Kohno, T. Helix: Fast encryption and authentication in a single cryptographic primitive. In *Fast Software Encryption, 10th International Workshop, FSE 2003* (2003), T. Johansson, Ed., Lecture Notes in Computer Science, Springer-Verlag.

[18] Gligor, V., and Donescu, P. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In *Fast Software Encryption, 8th International Workshop, FSE 2001* (2002), M. Matsui, Ed., vol. 2355 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 92–108. See `www.ece.umd.edu/~gligor/`.

[19] Goldwasser, S., Micali, S., and Rivest, R. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing 17*, 2 (Apr. 1988), 281–308.

[20] H. Krawczyk, M. B., and Canetti, R. HMAC: Keyed hashing for message authentication. IETF RFC-2104, 1997.

[21] Halevi, S. An observation regarding Jutla's modes of operation. Cryptology ePrint archive, reference number 2001/015, submitted Feb 22, 2001, revised Apr 2, 2001, 2001. See `eprint.iacr.org`.

[22] Hawkes, P., and Rose, G. Primitive specification for SOBER-128. Available from `http://www.qualcomm.com.au/Sober128.html`.

[23] Iwata, T., and Kurosawa, K. OMAC: One-key CBC MAC. In *Fast Software Encryption* (2003), T. Johansson, Ed., vol. 2887 of *Lecture Notes in Computer Science*, Springer-Verlag.

[24] Jonsson, J. On the security of CTR + CBC-MAC. In *Selected Areas in Cryptography–SAC 2002* (2002), K. Nyberg and H. M. Heys, Eds., vol. 2595 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 76–93.

[25] Jutla, C. Encryption modes with almost free message integrity. In *Advances in Cryptology — EUROCRYPT 2001* (2001), B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 529–544.

[26] Katz, J., and Yung, M. Complete characterization of security notions for probabilistic private-key encryption. In *Proceedings of the 32nd Annual Symposium on the Theory of Computing (STOC)* (2000), ACM Press.

[27] Kohno, T., Palacio, A., and Black, J. Building secure cryptographic transforms, or how to encrypt and MAC. Cryptology ePrint archive, reference number 2003/177, submitted Aug 28, 2003. See `eprint.iacr.org`.

[28] Kohno, T., Viega, J., and Whiting, D. High-speed encryption and authentication: A patent-free solution for 10 Gbps network devices. Cryptology ePrint archive, reference number 2003/106, submitted May 27, 2003, revised Sep 1, 2003, 2003. See `eprint.iacr.org`.

[29] Krawczyk, H. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Advances in Cryptology – CRYPTO 2001* (2001), vol. 2139 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 310–331.

[30] Liskov, M., Rivest, R., and Wagner, D. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO '02* (2002), M. Yung, Ed., vol. 2442 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 31–46.

[31] Petrank, E., and Rackoff, C. CBC MAC for real-time data sources. *Journal of Cryptology 13*, 3 (2000), 315–338.

[32] Rogaway, P. Authenticated-encryption with associated-data. In *ACM Conference on Computer and Communications Security (CCS-9)* (2002), ACM Press, pp. 196–205.

[33] Rogaway, P., Bellare, M., and Black, J. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC) 6*, 3 (Aug. 2003), 365–403.

[34] Wegman, M., and Carter, L. New hash functions and their use in authentication and set equality. In *J. of Comp. and System Sciences* (1981), vol. 22, pp. 265–279.

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