An Introduction to Authenticated Encryption

Shawn Fitzgerald — shawn@isecpartners.com

iSEC Partners, Inc
123 Mission Street, Suite 1020
San Francisco, CA 94105
https://www.isecpartners.com

March 7, 2013

Abstract

Over the last decade, authenticated encryption has become popularized and a number of modes have been proposed. This paper presents a technical introduction and analysis of the most well-known and standardized modes.

1 Introduction

Requirements for both confidentiality and authenticity of data have long been treated as synonymous by developers and systems designers. Yet for many years, primitives for establishing each of these have been designed independently. This has led to confusion in the user community and has resulted in a number of high profile system breaks. One need not look further than the break of WEP in 802.11 which used a Cyclic Redundancy Check (CRC) as a hash function along with a stream cipher for encryption [1]. There is a need for both a cryptographically strong authentication mechanism such as a Message Authentication Code (MAC) as well as strong encryption algorithms, but what is not clear or intuitive is how to combine these to achieve security. In fact, a number of competing systems have been defined in widely used security protocols. These include:

EtM: Encrypt-then-MAC. A message M is encrypted under key $K1$. Then the tag T is calculated as $MAC(K_2, C)$. The pair (C, T) becomes the output of the function. This scheme is used in the IPSec protocol.

MtE: MAC-then-Encrypt. The tag is calculated as $MAC(K_2, M) \rightarrow T$. The pair (M, T) is then encrypted under $K1$. This scheme is used in SSL/TLS.

E&M: MAC-and-Encrypt. The message M is first encrypted under K_1. The tag is also calculated on the original message M as $MAC(K_2, M)$. The pair (C, T) becomes the output of the function.[1] The SSH protocol utilizes this method.

These notions were formally analyzed by Bellare and Namprempre [2] who showed that only EtM provided the strongest notions of security in all cases. Briefly these notions of security are as follows.

- Indistinguishability under chosen plaintext attacks (IND-CPA) is the requirement that the probability that an attacker can distinguish between two previously known messages that have been encrypted under an unknown key is negligible.
- Indistinguishability under chosen ciphertext attacks (IND-CCA) is similar, however before making their guess the attacker is also allowed to choose arbitrary ciphertexts and have them decrypted.

[1] Or in other words, the output is the encryption of the message concatenated with the MAC of the message.

- Integrity of plaintext (INT-PTXT) is the requirement that it is computationally infeasible for an attacker to create a valid ciphertext for a corresponding message.
- Integrity of ciphertext (INT-CTXT) is the requirement that it is computationally infeasible for an attacker to create a valid ciphertext, regardless of whether this ciphertext corresponds to a message.

Any system that is INT-CTXT is also INT-PTXT; Additionally if a system has IND-CPA and INT-CTXT it is also IND-CCA secure.

This analysis by Bellare and Namprempre prompted a number of studies of existing security protocols such as [3, 4] where it was demonstrated that even though SSL uses MtE, it is IND-CCA. This security however is tenuous and seemingly innocuous changes such as encrypting the message and MAC separately using different initialisation vectors or using variable length padding and truncated MACs (as specified in TLS 1.2) can break the security of the system.[2] Schemes can also be constructed using E&M that are not IND-CPA secure, a simple example being to use a MAC that provides message integrity but that leaks part of the underlying message; such a MAC would be perfectly acceptable on its own, but results in a trivial break when combined with encryption. It was clear from this that a more formal approach to the construction of encryption and authentication schemes needed to be devised and standardized.

Authenticated Encryption combines message authentication and message integrity into one mode such that the end result is IND-CPA + INT-CTXT and therefore IND-CCA and INT-PTXT as well. It removes the need for developers and system designers to create their own, potentially insecure constructs by providing them one operation to meet these requirements. Additionally, by essentially providing one more layer of abstraction, it promotes cryptographic agility,[3] which is the design of a system such that the underlying algorithms can be changed without large re-designs of the existing system. Traditional schemes that used separate MAC and encryption algorithms required independent keys to be used and securely managed, something that does not always happen. All the authenticated encryption modes discussed here use one key in a manner that does not compromise security.

2 Authenticated Encryption Modes

The following is a brief overview of the most popular authenticated encryption (AE) modes that have been developed as a result of the research discussed in the previous section. The aim is to provide enough detail to allow the reader to understand the potential pros and cons of each mode.

Before discussing individual modes, it is useful to understand a few key properties of authenticated encryption modes (although not all modes discussed here exhibit these properties). A useful property of an AE mode is to support both data that is to be encrypted and authenticated and data that is not encrypted but that needs to be authenticated; this concept is referred to as authenticated encryption with associated data (AEAD).[4] A conceptual example of this is a network packet that contains a payload and routing information; it would be useful to encrypt and authenticate the payload while allowing the routing information to be in plaintext, while still being authenticated and cryptographically bound to the payload. Another concept of AE is that of being on-line. On-line means that the length of the message and any associated data are not required to be known before encryption is performed; this property can be beneficial in streaming and other environments where the size of the encrypted message is transparent to the algorithm. There are other important concepts such as parallelization and whether the mode is one or two pass, but these will be discussed in the relevant sections.

All AE modes discussed in this paper share the same underlying encryption mode called counter mode (CTR). Counter mode takes as input a message M, a key K and a nonce N. The message is broken into m blocks where m is the block length of the underlying algorithm. A counter value is created from the nonce such that there are

[2] It should be noted that these analysis generally do not model attacks that exploit the ability to differentiate the cause of decryption failures such as padding failures or authentication failures or attacks involving chained IVs in CBC.

[3] To be clear, only some authenticated encryption modes allow any secure underlying algorithm to be used

[4] For simplicity's sake, none of the diagrams presented in this document reference associated data.

m unique counter values. This can be achieved, for example, by combining the nonce and a counter. Each counter value is encrypted with K and the result is XORed with the corresponding message block. Although the nonce need not be secret, it must be unique such that no messages encrypted under the same key shall have the same counter value. If this does occur, portions of the message will leak.

2.1 CCM — Counter with Cipher Block Chain

Counter with Cipher Block Chain is a NIST approved [5] mode of operation that combines a 128-bit block cipher in counter mode for encryption and a CBC-MAC for integrity. It supports AEAD and is accompanied by a security proof [6]. Tag lengths are variable, although they must be byte aligned at 4, 6, 10, 12, 14 or 16 bytes. A single key is used for both primitives, which can be used for at most 2^{61} invocations of the algorithm.

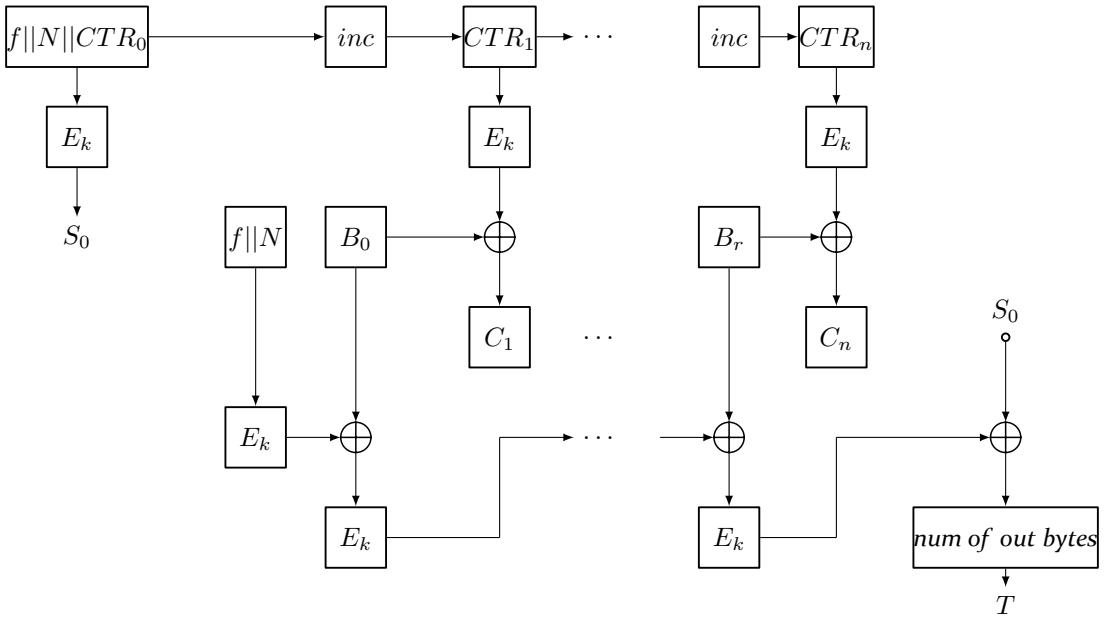

Figure 1: CCM Mode

Both CTR and CBC-MAC are constructed in a standard manner; however, the formatting function (i.e. the padding mechanism) and the generation for counter blocks are complex. The algorithm takes as input a nonce N, message M and data to be authenticated (but not encrypted) A. The formatting function is applied to the message M creating $(B_0, B_1, \ldots\ B_r)$ where r is the number of blocks in the message. The formatted message is CBC-MACed and the desired tag length in bytes is created from the last block of the encryption. It should be noted that the IV is created by encrypting the formatted nonce. The counter blocks are created from zero to m, where m is the message length divided by 128 (i.e., the number of blocks in the message) by concatenating formatting data, the nonce and an incrementing counter value. Each of these counter values are then concatenated. This value is XORed with the message and concatenated with the result of the tag value XORed with S_0 to create the ciphertext. Decryption is the reverse of this, that is counter values are created and XORed with the ciphertext to produce the message, and then the CBC-MAC is created from the message and checked.

CCM was created as a non-patented alternative to OCB, and due to this has had widespread adoption in standards and cryptographic libraries. As an example, both OpenSSL[5] and Bouncy Castle[6] implement CCM. It is relatively

[5] http://www.openssl.org
[6] http://www.bouncycastle.org

simple to implement, with small memory requirements, although the padding mechanisms are complex. It is, however, not fully parallelizable — only CTR encryption can be performed in a parallel manner. Additionally, it is not on-line. This makes it unsuitable for streaming environments. Tag lengths can be between 32 and 128-bits in multiples of 16; however, 64 bits should be considered the minimum for any real security, since the likelihood of an attacker's guess succeeding is $\frac{1}{2^{Tlen}}$. Additionally, any implementation should not accept shorter tag lengths than what has been generated as this could open the possibility of brute force attacks. For example, for a 4-byte tag an attacker will successfully guess a tag on an arbitrary ciphertext in 2^{32} tries; that is if they want to modify a given plaintext M by some difference X they can calculate $C \oplus X||T^{'}$ and cycle through all possible tag values. The majority will be rejected but one will be accepted and decrypt as $M \oplus X$. [7, 8]

2.2 EAX Mode

EAX was designed as an active two pass scheme aimed at addressing perceived shortcomings of CCM while not being encumbered by patents. [8] It supports AEAD and is supported by a security proof. EAX allows any suitable block cipher to be used; however AES-128 or AES-256 are the most common specifications. Additionally, EAX allows any size authentication tag to be specified, depending on the security requirements of the system. It should be noted that a variant called EAX′ (EAX prime) exists — this is a performance optimized variant of EAX that has been broken and should not be used. [9]

EAX is essentially composed of a block cipher in CTR mode for encryption and an OMAC (one-key CBC MAC) for authentication. CTR is of the standard construct; however, since OMAC is less widely understood, a brief overview is provided here.

OMAC takes as input a key K, a binary string t [7] and a message and performs a CBC encryption on a specific transform of t concatenated with the message. This transform, referred to as a pad, is essentially an XOR of the message and one of two constants created from the block of all zeros encrypted with the key. Critically, these steps must be performed in a time constant manner, as the algorithm contains a conditional branch depending on whether the first bit of the encrypted block of zeroes is set or not. Failure to do this may result in vulnerability to timing attacks. Once the pad is applied to the message, CBC encryption is performed and the last block is output.

EAX takes as input a key K, nonce N, header to be authenticated H and Message M. An OMAC is taken of the nonce, which is provided as a counter input into the CTR operation. The output of the encryption operation is also authenticated by taking an OMAC of the ciphertext. Additionally, if a static header is provided, it is also authenticated with OMAC. These are then XORed to produce the tag, with the final output being the ciphertext concatenated with the tag. Decryption takes place by authenticating the three intermediate OMAC values and then decrypting the message.

EAX has a number of benefits over other non-patented two pass schemes. EAX is on-line, meaning the size of the message is not required to be known in advance. Header values can be pre-computed requiring only one XOR operation; this may be important for static values, such as packet information that does not often change. The algorithm is relatively simple to implement and only uses encryption operations in a forward manner. Finally, message expansion is only by the number of bits in the tag.

Although EAX has existed for a number of years, it has not seen widespread adoption; however, it has been implemented in the Bouncy Castle cryptographic provider, among others. Although it does have a number of advantages over CCM, it has not been NIST approved, and unlike GCM and OCB, it is not parallelizable.

[7] Represented by the superscript values in the diagram

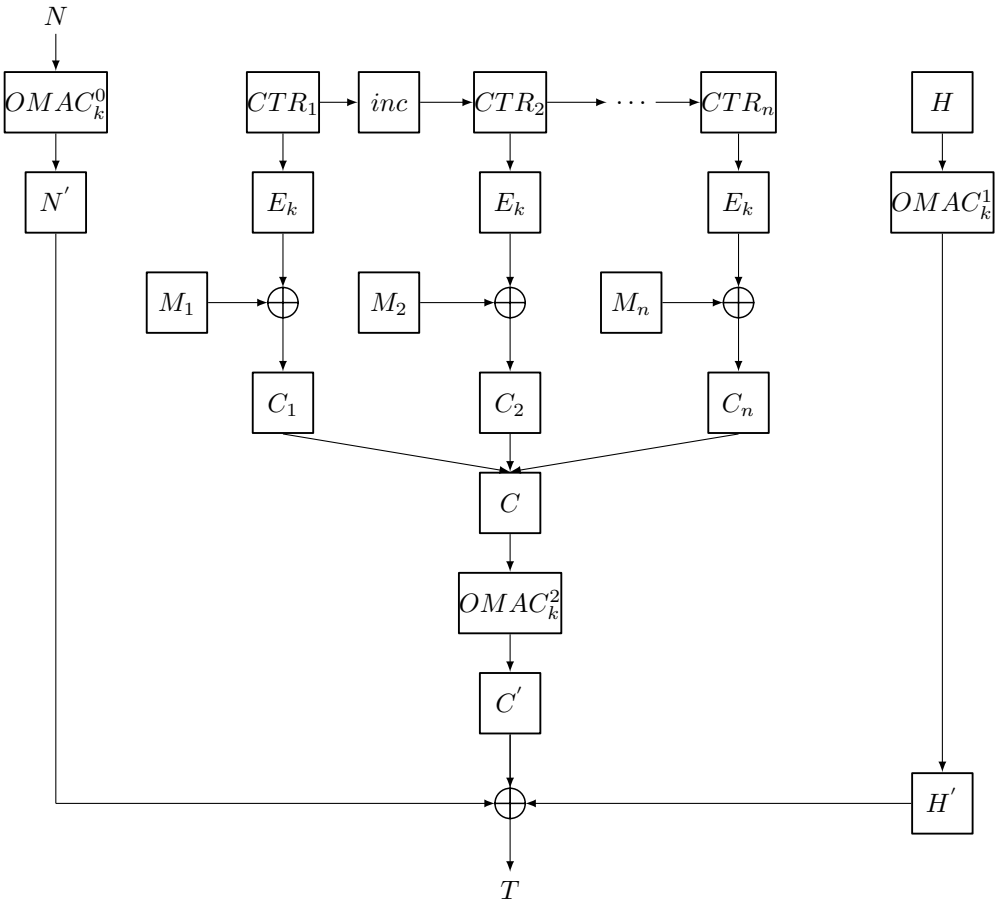

Figure 2: EAX Mode

2.3 GCM — Galois Counter Mode

Galois Counter Mode (GCM) is a NIST approved mode of operation that combines a 128-bit block cipher in counter mode for encryption with a special MAC for integrity.[10, 11] This MAC, referred to as a GHASH (or $\bullet H$ in the specification), is a Wegman-Carter MAC that is built from multiplication over the finite field[8] $GF(2^{128})$. As with many other AE modes, it supports AEAD and is supported by a security proof. Additionally, it can also be used as a MAC only. It also has no patents associated with it. It is arguably the most complex of the modes presented in this paper, and consequently an abstract of its design is discussed in the following section.

The GHASH function computes an authentication tag using multiplication in $GF(2^{128})$. It should be noted that this function as described in the specification should not be used on its own as a MAC. The message is broken into a bit string $M = M_1||M_2||\ldots||M_n$ (i.e., into 128-bit blocks). The hash subkey is calculated by encrypting a 128-bit block of zeros with the key. Each 128-bit block of the bit string is then multiplied by the subkey, where multiplication is defined over $GF(2^{128})$.[9] The last block Y_n is then the output of the GHASH function.

The GCTR takes the initial block counter and increments it up to the number of blocks in the message. These values

[8] Throughout this paper we use the term finite field and Galois field (GF) synonymously

[9] That is, multiplication is not defined in the traditional sense but as an operation that satisfies specific properties in a field.

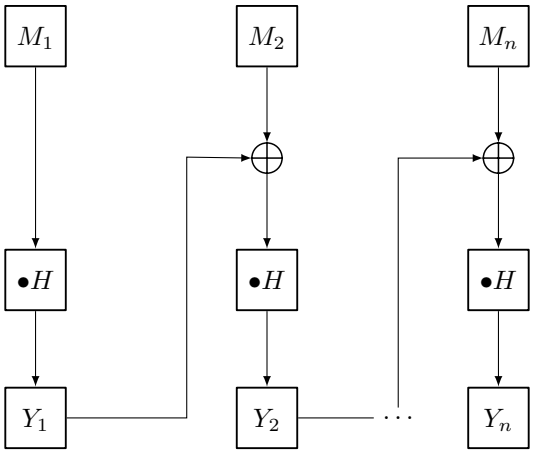

Figure 3: GHASH Function

are then encrypted and XORed with each block of the message. The final result of these operations is returned as the encrypted string C.

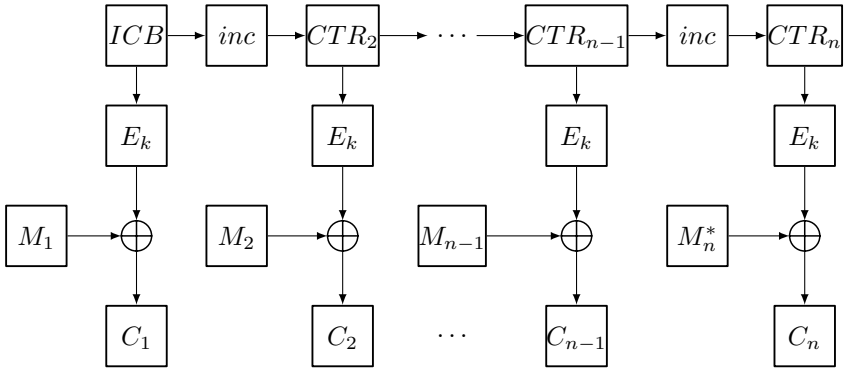

Figure 4: GCTR Function

The full GCM operation can be viewed as the superimposing of GCTR and GHASH as described above. The inputs are the initialization vector IV, the key K, the message M (where the last block is represented by M_n^* which may not be a full block length) and any authentication data A. The subkey H is calculated and the initial counter block (ICB) is either the IV if the IV length is 96 bits, or the GHASH of the IV if the IV length is any value other than 96 bits. The counter is incremented and each plaintext block is encrypted with the counter value. Each ciphertext block is then multiplied by H and then XORed with the next block. The exception to this is that the encryption of the first counter which is XORed with the last GHASH operation to create the authentication tag. The authentication only data A is mixed in a similar manner. Decryption follows in the usual manner, where the encrypted counter values are created and XORed with the ciphertext to produce the original message and then the GHASH in created.

GCM has the same requirements for uniqueness of the IV (i.e., the counter value) as traditional stream ciphers and counter modes. Depending on the length of the IV either a counter or pseudo random number generator based construction can be used for generation, but must never repeat. As is the case with other counter modes, repeating the counter value for two messages breaks the confidentiality of those messages (i.e., an attacker can learn $M1 \oplus M2$); however, Ferguson and Joux have shown GCM suffers a more critical break in that a repeated IV can lead to leakage of the authentication key.[12, 13] In addition, the use of a short authentication tag can increase

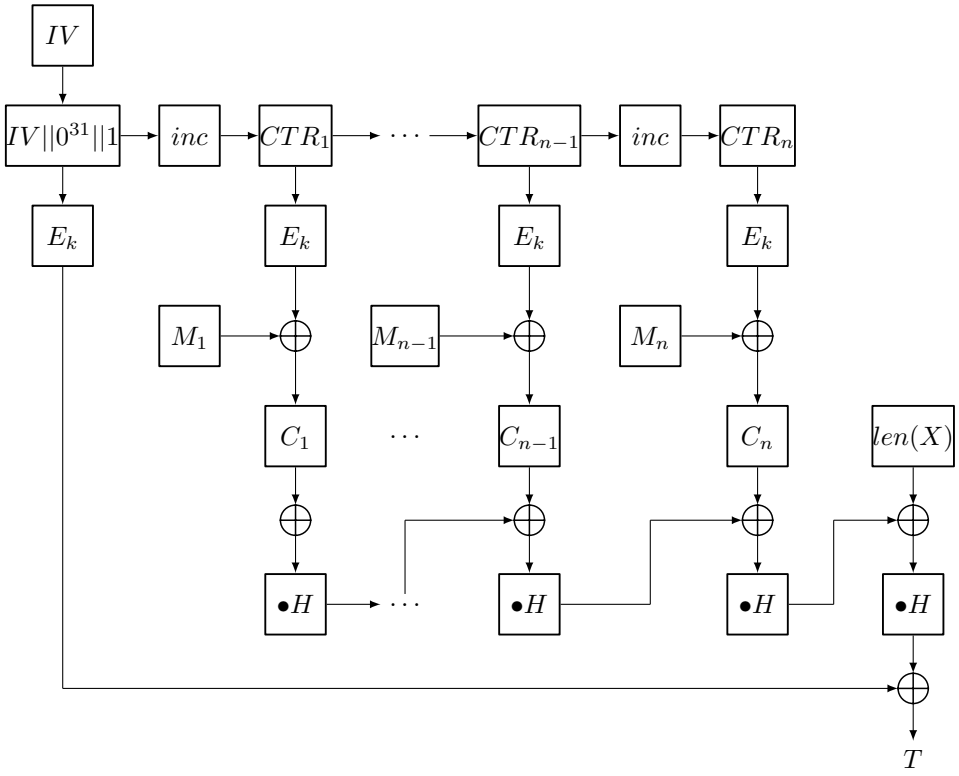

Figure 5: GCM Mode

the chance of collisions, which can allow for a total break of the authentication scheme. Thus it is critical that IVs never repeat and that any potential attacker is not able to coerce repetition of IVs. Even though SP 800-38D allows the usage of shorter tags, a safe recommendation is to always use 128-bit tags.

Although GCM is a two pass mode, it relies on an extremely fast Wegman-Carter MAC. Furthermore, Intel has added the PCLMULQDQ instruction allowing fast multiplication in $GF(2^n)$, and recent patches to OpenSSL and NSS also helped further optimize GCM.

2.4 OCB — Offset Code Book Mode

OCB is a patented mode of operation that allows for various block ciphers and key lengths.[10] Like all other AE modes discussed here, it is based on counter mode along with an underlying function for integrity.[14, 15] Although the first version of OCB does not support associated data (i.e., AEAD), both later versions do. It is also supported by a security proof and tag lengths are variable up to 128-bits. The previous versions of OCB are not recommended — unless stated explicitly, OCB refers to OCB3.

As with any CTR-based scheme, counter values must be calculated in order to perform encryption. Here they are established in the following manner: The first 90 bits of the 96 bit nonce are padded and encrypted with the key K. This value is referred to as *Ktop*. *Ktop* is then XORed with itself left bit shifted 8 and the result is appended to *Ktop*. The result of this value is then left bit shifted by the last 6 bits of our original nonce.

[10] See http://www.cs.ucdavis.edu/~rogaway/ocb/ for the most up to date information including reference implementations.

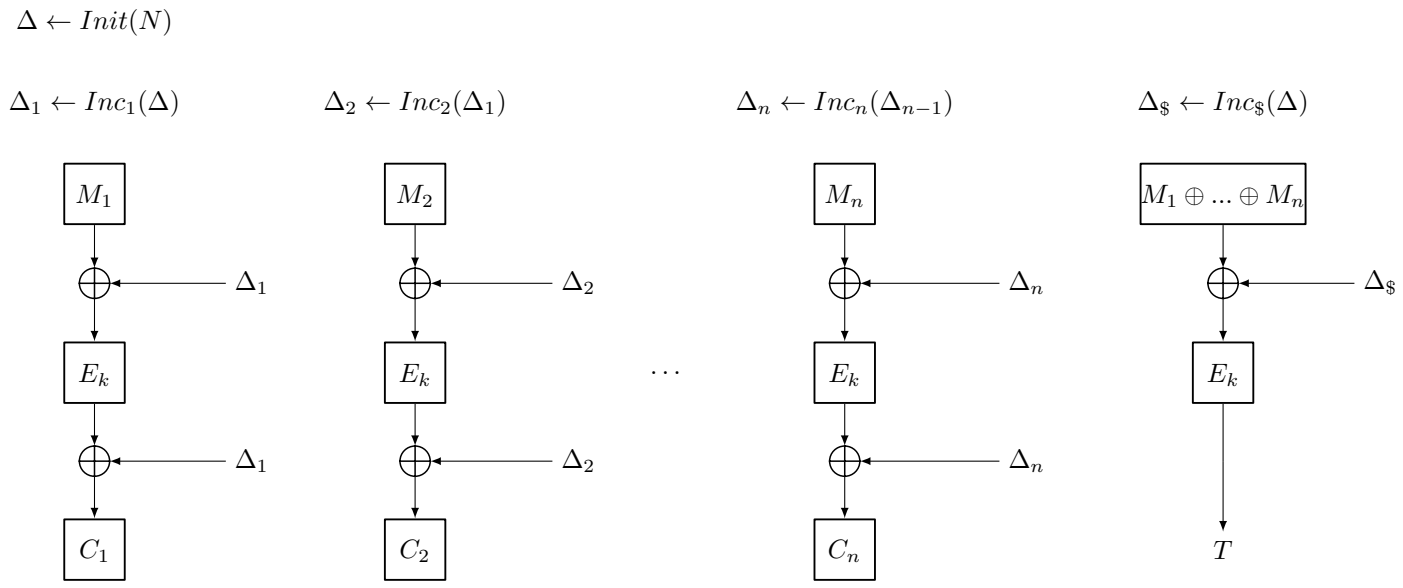

Figure 6: OCB Mode

The output of this function becomes $Init(N)$, and is the initial counter value. The incrementing function encrypts a block of zeros with K and, for each block to be encrypted, left bit shifts this value by one and XORs it with $Init(N)$ (although the first increment is saved for the checksum). It should be noted that these can be pre-computed for a given key. For each block to be encrypted, the message is XORed with the counter, encrypted, and then XORed with the counter again to produce the ciphertext. A checksum is also created by breaking the message M into n blocks and then XORing each of these blocks together. The result is then XORed with the counter value, and the result is encrypted with key K. The result of this operation is then output as the tag.

Mainly due to the fact that OCB is a one-pass scheme, it is on the order of two times faster than the fastest two-pass scheme GCM. In addition to this, it is on-line and parallelizable. It has, however, not generally been adopted by standards bodies and is not included in any of the popular cryptographic libraries (although optimized reference code does exist). The primary reason for this has been OCB is patented. As of January 2013, the patent restrictions have been loosened to allow inclusion in open source software and non-military usage.[11]

3 Comparisons

The following section compares properties that are relevant to developers and system architects when selecting a mode of operation. Some properties such as complexity and security are subjective, but are still very important to consider.

Support for associated data: All modes discussed here except for the first version of OCB are full AEAD schemes meaning that they support associated data.

Support for on-line processing: Only CCM is not on-line, all others are fully on-line modes.

Parallelization: GCM and OCB and fully parallelizable, where as CCM and EAX only allow parallelization in the encryption operation.

[11] See http://www.cs.ucdavis.edu/~rogaway/ocb/license.htm for more information.

Complexity: GCM and CCM are perhaps the most complex to implement, due to the extensive use of finite field arithmetic in the case of GCM, and complex padding scheme in the case of CCM. These issues can be somewhat mitigated by well-designed cryptographic libraries and mature reference implementations, but should be considered if there is a requirement to implement a mode from scratch.

Speed: A number of comparative studies have been performed on AE modes, including recent studies taking advantage of Intel's AES-NI / PCLMULQDQ accelerations for AES-GCM. Using an x86-64 AES-NI architecture and encrypting 4kb messages, OCB3 has the best performance numbers, at 1.48 CPU cycles per byte, followed by AES-128 GCM and CCM with 2.53 and 4.17 respectively. Performance on an x86-32 AES-NI architectures yield relatively similar results.

Security: Although all modes discussed in this paper have been published with an accompanying proof of security, there are other potential issues that are not covered by these proofs. In all modes, care must be taken to ensure the nonce value or counter values are not repeated with the same key. This could have the affect of leaking message contents, or in the case of GCM, leaking the hash key. Truncating tags can lead to a higher probability of collations and extreme care should be taken when using shorter tag lengths. All modes have the potential for timing vulnerabilities and care must be taken to ensure critical operations are implemented in a time constant manner. Even a cursory examination of AE implementations[12] has revealed potential timing vulnerabilities, including a cryptographic API that failed to perform an operation in a constant time manner, despite the specification mandating it.

Intellectual property restrictions: OCB is the only mode discussed here that is covered by patents. This is probably the main reason that it is not widely used, despite its apparent advantages.

Cryptographic library support: GCM is the most widely supported, and is implemented in OpenSSL, Bouncy Castle and NSS. CCM has been implemented in OpenSSL and Bouncy Castle, and EAX has been implemented in Bouncy Castle. OCB does not appear to be implemented in any major cryptographic library, but optimized reference implementations exist.

4 Conclusion

While the current authenticated encryption modes have a number of similarities, there are key differences that emerge which make them more suitable for a generic recommendation — however, any of these modes are preferable to constructions based on MtE, E&M or any custom construction. If patent restrictions are not a concern, and performance is paramount, OCB is the most attractive mode. It is the fastest, especially in environments without Intel's optimizations. The potential downside is that existing cryptographic libraries do not support it.

Among the non-patented modes, GCM is the most attractive — it is the fastest of the two pass schemes, and much work has been performed on optimizations. The majority of cryptographic libraries also support it. It does, however, have a number of potential security issues, and care must be taken to ensure that all critical operations are implemented in a time constant manner and that nonce values are never re-used. As a final note, unless the environment absolutely requires it, it is recommended to use implementations from well-known and well-vetted cryptographic libraries. If custom implementations must be used, a qualified cryptographic expert should review them.

[12] See for example http://www.imperialviolet.org/2013/01/13/rwc03.html

References

[1] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11. *MOBICOM*, 2001.

[2] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology — ASIACRYPT*, 2000.

[3] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). *Advances in Cryptology — CRYPTO*, 2001.

[4] K. G. Paterson, T. Ristenpart, and T. Shrimpton. Tag size does matter: Attacks and proofs for the tls record protocol. *ASIACRYPT*, 2011.

[5] M. Dworkin. Recommendation for block cipher modes of operation: the ccm mode for authentication and confidentiality. *NIST Special Publication 800-38C*, 2004.

[6] J. Jonsson. On the security of ctr + cbc-mac. In *Proceedings of Selected Areas of Cryptography (SAC)*, 2002.

[7] P. Rogaway and D. Wagner. A critique of ccm. *IACR Cryptology ePrint Archive*, 2003/70, 2003.

[8] M. Bellare, P. Rogaway, and D. Wagner. The eax mode of operation: (a two-pass authenticated-encryption scheme optimized for simplicity and efficiency). *Fast Software Encryption*, 2004.

[9] M. Kazuhiko, M. Hiraku, and I. Tetsu. Cryptanalysis of eaxprime. *IACR Cryptology ePrint Archive*, 2012/18, 2012.

[10] M. Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. *NIST Special Publication 800-38D*, 2007.

[11] D. McGrew and J. Viega. The security and performance of the galois/countermode(gcm) of operation. *INDOCRYPT*, 2004.

[12] M. Ferguson. Authentication weaknesses in gcm. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf, 2005.

[13] A. Joux. Authentication failures in nist version of gcm. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/Joux_comments.pdf, 2005.

[14] P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes ocb and pmac. *Advances in Cryptology — ASIACRYPT*, 2004.

[15] T. Krovetz and P. Rogaway. The software performance of authenticated-encryption modes. *Fast Software Encryption*, 2011.