Network Working Group	J. Schaad
Internet-Draft	Soaring Hawk Consulting
Intended status: Informational	April 06, 2011
Expires: October 08, 2011	

Commentary on the Design of the Authenticated-Enveloped-Data Content Type
draft-schaad-smime-aed-rant-02

Abstract

The Authenticated-Enveloped-Data Content Type allows for the use of Authenticated-Enveloped modes with block cipher algorithms. At the time of the original design there was discussion about the relative location of the authenticated attributes and the encrypted content in the ASN.1 structure. With the benefits of implementation experience I revisit the discussion made at the time and re-evaluate the decision made.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

When the Cryptographic Message Syntax (CMS) [CMS] Authenticated-Enveloped-Data content type (defined in RFC 5083 [CMS-AED]) was being discussed, the S/MIME working group had no actual implementation experience to guide it in some of the decisions that were being made at the time. In this document I am revisiting one of these decisions based on the implementation experience that I have since garnered.
Issues that were discussed at the time included:

- What should the order be for the authenticated attributes, the encrypted data and the authentication code be in the ASN.1 structure. There was uniform agreement that the authentication code should be last, however the placement of the other two fields was hotly disputed. This is the issue that we further address below.

- Should we change from using a SET to a SEQUENCE for the attribute list. Doing so would have simplified the encoding processing for hashing. There was no support for doing this as a common routine exists that already worked for the signed and authenticated data structures.

- What are the security issues that deal with the timing of release of the encrypted content vs. the validation step. This issue was addressed in section 2 with the statement "The recipient MUST verify the integrity of the received content before releasing any information, especially the plaintext of the content."

- Step 5 in section 2 says that padding needed to be done to the block length, however there was some concern that the issue of how padding should be done is better left to the algorithm description rather than being specified here. No changes were made to address the issue.

The major focus of the discussions centered on the relative placement of the encrypted data blob (contained in the authEncryptedContentInfo field) and the authenticated attributes (contained in the authAttrs field). There were three different camps that emerged. These where: 1) The attributes should be before the encrypted data, 2) The attributes should be after the encrypted data, and 3) There should be the ability to place the attributes both before and after the encrypted data and the encoder would choice which to use. As can be seen from the ASN.1 in Figure 1 the final decision was to place the authenticated attributes after the encrypted content. This was counter to the arguments that I made at the time to place the authenticated attributes before the encrypted content.

```
 AuthEnvelopedData ::= SEQUENCE {
   version CMSVersion,
   originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
   recipientInfos RecipientInfos,
   authEncryptedContentInfo EncryptedContentInfo,
   authAttrs [1] IMPLICIT AuthAttributes OPTIONAL,
   mac MessageAuthenticationCode,
   unauthAttrs [2] IMPLICIT UnauthAttributes OPTIONAL }
```

This document is organized as follows:

- Section 2 contains a review of the arguments presented at the time.
- Section 3 has a taxonomy of a number of authenticated encryption algorithms.
- Section 4 presents a set of criteria to be used.
- Section 5 contains my personal conclusions on the issue.
- Section 6 contains rebuttals (or maybe not).

The major part of my discussion focuses on the desirability to use a streaming model for processing the ASN.1 structure and the data contained within it. If one does not want to use streaming in doing the processing, then much of the discussion here is moot. If one is willing to buffer up all of the input to the encryption algorithm before applying it, the order that the inputs are presented are immaterial. This will be further detailed in Section 4.

1.1. Terminology

The following is a list of standardized terms used in the document:

AE is an abbreviation for Authenticated Encryption. This is block cipher mode of operation which simultaneously provides confidentiality and integrity assurances on the data.

AEAD is an abbreviation for Authenticated Encryption with Auxiliary Data. This is a block cipher mode of operation which simultaneously provides confidentiality and integrity assurances on the message data as well as integrity assurances on an additional set of data.

Message Data is the section of the input data that is to be authenticated and encrypted by the AE or AEAD algorithm mode. For

CMS, the encrypted message data is placed in the encryptedContent field of the authEncryptedContentInfo sequence.

Authenticated Data is the section of input data that is to be authenticated but not encrypted. For CMS, the authenticated data is the sequence in the authAttrs field.

Authentication Tag is a value that is generated by the mode which is used to validate the integrity of the data. The Authentication Tag is sometimes implicit and does not exist as an independent value. For CMS, it is assumed that the use of the algorithm will define an explicit tag and the tag will be placed in the mac field.

Streaming Model is a method of doing the processing such that the ASN.1 processing and the cryptographic processing can be interleaved with each other.

2. Historic Arguments

I have gone through the archived mailing list from the time to find the arguments that were being advanced. The arguments are laid out with the pro side being for attribute being placed after the data except for the last item in the list.

1. Consistency with the existing CMS data types:

   *PRO: We have working implementations of both AuthenticatedData and SignedData which work. In both of these cases the data structures are ordered such that the message data precedes the authenticated data. Keeping the order consistent makes coding easier and leads to fewer mistakes.

   *CON: Being constant is nice, however if it does not work correctly that does not matter.

2. Authenticated attributes may be derived from the message content:

   *PRO: It should be possible to create authenticated attributes based on the content of the encrypted data and have these attributes authenticated. Placing the attribute before the message content means that one must buffer the message content to do this. The example of this presented on the mailing list was the ability for a sender to process the body of the message on fly by a virus checker and publish the result of the virus checking as an authenticated attribute. This is the same thing that currently happens today for both SignedData and AuthenticatedData where the hash of the message data is computed on the fly and then

placed in the signed/authenticated attributes which are then processed to compute the signature or mac values.

*CON: Placing this information after the message data means that the recipient cannot know to perform matching processing, if necessary, in order to check the value presented by the sender. The analogous step for the SignedData structure is the need for the recipient to hash the message data during processing in order to correctly validate the signed attribute fields.

3. The decision should be dictated by Algorithm Characteristics:

*PRO: The order of placing the attributes before the message data was dictated by a specific choice of algorithms (CCM and GCM) and that other authenticated encryption algorithms (specifically CWC) would naturally place the attributes second.

*CON: No detailed analysis of algorithms was done. However, the attribute data should be expected to be much smaller than the message data and thus it makes more sense to cache the attributes for later processing than to cache the message data for later processing.

4. Resource requirements for the sender and recipient:

*What happens with resource constrained devices that are acting as senders or recipients? The initial argument dealt with the question of resource limited senders that would not be able to store intermediate data, but the same question applies to resource limited recipients. We know that this was intended to be used with firmware upgrades as one option, but it could equally be used by a device sending out reports to a central server. This is a case where a close analysis would need to be done on the algorithm being used and how it will affect the resources needed.

5. Relative frequency of processing:

*There was a certain amount of discussion of the question of the relative frequency of processing between the sender and the recipient of a message. This would have bearing on the question of which entity the decisions should be optimized for. One set of people argued that recipients process messages more frequently than senders. Another set of people argued that there exist applications where the sender may create messages that are never verified.

6. Attributes should be placed in both locations.

    *There were a couple of people who attempted to argue that the discussions should be made by the sender of the message rather than by the object designers. In this case we should have two different locations where the authenticated attributes could be place, either before or after the data, but only one of the two could be used. The message creator would then select one or the other based on characteristics of their choosing. Recipients would then be required to deal with the attributes occurring in either location. It was generally felt that the additional complexity on the recipient side was not worth the added flexibility.

3. Algorithm Taxonomy

In item 3 in the previous section, one of the issues was what would a rigorous analysis of the AEAD algorithms lead us to believe about how the choice should be laid out. At the time we were using only hearsay facts about what would make for a good choice. In this section, I define a set of criteria that I will use to analysis the set of algorithms and then describe how each algorithm fits the criteria.
NIST has been gathering information on Authenticated Encryption Modes over the last decade. Information on these modes can be found at http://crc.nist.gov/groups/ST/toolkit/BCM/modes_development.html. For simplicity I used this as the set of algorithms to look at in order to characterize the requirements for the purposes of comparison with the characteristics required by the Authenticated Encryption data structure.
In this section we will look at 11 AE algorithms from the NIST submissions along with an algorithm being developed by Peter Gutmann. Since we are interested in how to setup a streaming model, the criteria we are looking at are chosen with that in mode. The major characteristics we are going to be looking at are:

1. What are the parameters used for the algorithm? This contains a list of the elements that are needed for processing exclusive of the key value. These are the items that would need to be encoded in the ASN.1 parameters field of an AlgorithmInformation structure.

2. What information is directly authenticated? This is a list of the data which is directly authenticated in the order of authentication. (It is possible that this list may change depending on the parameters. Thus if HMAC-SHA1 is used, the length of the data is directly authenticated but it would not be if MAC-AES-128-CCBC was used.)

3. What information is required before the first byte of message data can be processed? Assuming that the first byte of message data is to be processed upon it being decoded from the ASN.1 (or encoded to ASN.1), what items of information are needed by the encryption/decryption algorithm prior to it being processed.

4. What information is required before the first byte authenticated data can be processed? Assuming that the first byte of authenticated data is to be processed upon it being decoded from the ASN.1 (or encoded to ASN.1), what items of information are needed by the encryption/decryption algorithm prior to it being processed.

NIST is currently in the middle of doing a review and selection process for new modes to adopt as US security standards. For simplicity the set of algorithms that I will be looking at come from the current set of candidate algorithms that are being reviewed for this purpose. One additional algorithm added to this is a simple hash and encrypt algorithm that has been proposed by Peter Gutmann.

3.1. CCM: Counter with CBC-MAC

The Counter with CBC-MAC (CCM) mode was designed and documented by Doug Whiting, Russ Housley and Niels Ferguson. A full description of the mode can be found in RFC 3610 [RFC3610] and on the NIST website. CCM is one of the standardized NIST modes (see [NIST-800-38C]) and is one of the two modes that are currently documented for use with the CMS Authenticated-Enveloped structures.
The characteristics of the algorithm are:

1. The parameters of the algorithm are the nonce (IV) and the length of the tag to be generated.

2. The data authenticated is:

    a. The nonce value,

    b. The length of authentication tag,

    c. The length of message data,

    d. The length of authenticated data,

    e. The authenticated data,

    f. The message data

3. Before the first byte of message data can be processed, you must know:

    a. The nonce value

    b. The length of the authentication tag

    c. The length of the message

    d. The length of authenticated data,

    e. The authenticated data

4. Before the first byte of the authenticated data can be processed, you must know:

    a. The nonce value,

    b. The length of the authentication tag

    c. The length of the message

    d. The length of authenticated data,

This algorithm mode provides major problems for a sender to process in a streaming model. The lengths of the message data and the authenticated data are both required to be known before any bytes of the message data or authenticated data can be processed. Except in cases where fixed length messages will be generated, it is required that the message data be cached prior to encrypting.
This algorithm provides some problems for recipients in processing, but under the correct circumstances can be processed under a streaming model. The length of the message data must be presented to the recipient before the message data is given. The authenticated data must be presented before the message data is presented. Optimal use of this algorithm would require that 1) the authenticated data be moved before the message data bytes and 2) a requirement be established that either the message data be DER encoded or the message data length be published as part of the authenticated data. Given that this algorithm uses counter mode for encryption, the length of the message is already known so publishing it as part of the authenticated data would not leak any additional information.

3.2. CS: Cipher-State

Cipher-State is an algorithm that supports an AE mode of operation, but not an AEAD mode of operation. As such it does not matter where the authenticated parameters would be placed as they are not supported by the mode. This mode is therefore not of interest to this discussion.

3.3. CWC: Carter Wegman with Counter

The Carter Wegman with Counter Authenticated Encryption mode was designed by Tadayoshi Kohno, John Viega and Doug Whiting. A full description of the mode can be found in [CWC] and on the NIST website. The characteristics of the algorithm are:

1. The only parameter of the algorithm is a nonce.
2. The data actually authenticated is:
    a. The nonce,
    b. The authenticated data,
    c. The encrypted message data
3. Before the first byte of data can be processed, you must know:
    a. The nonce value,
    b. The authenticated data
4. Before the first byte of authenticated data can be processed, you must know:
    a. The nonce value

It should be noted that the analysis above is for a simplistic implementation of the algorithm such as would normally be done in software. The algorithm is designed so that it can be performed in parallel, it would be possible for message data bytes to be fully processed before the authenticated data bytes are processed. The full details of this approach are not spelled out in the referenced documents.
This algorithm can be easily streamed for the sender provided that the authenticated data are generated prior to the message data being generated.
This algorithm can be easily streamed for the recipient provided that the authenticated data is presented prior to the message data being presented.

3.4. EAX: A Conventional Authenticated-Encryption Mode

A Conventional Authenticated-Encryption Mode was designed and documented by M. Bellare, P. Rogaway and D. Wagner. A full description of the algorithm can be found at [EAX] and on the NIST website.
The characteristics of the algorithm are:

1. The only parameter of the algorithm is a nonce.

2. The data actually authenticated is:

    a. The nonce,

    b. The authenticated attributes,

    c. The encrypted message.

3. Before the first byte of data can be processed, you must know:

    a. The nonce value.

4. Before the first byte of the data can be processed, you must know:

    a. The nonce value.

5. Before the first byte of authenticated data can be processed, you must know: nothing.

This mode computes the authentication value on the authenticated data and on the encrypted message separately - so they can be computed in any order - and combines the results together after the entire message has been processed.
This algorithm can easily be streamed for the sender. The order of generating the authenticated data and message data is immaterial.
This algorithm can easily be streamed for the recipient. The order of presenting the authenticated data and the message data is immaterial.

3.5. GCM: Galois/Counter Mode

The Galois/Counter Mode of Operation (GCM) was designed and documented by David McGrew and John Viega. A full description of the algorithm can be found on the NIST website. GCM is one of the standardized NIST modes (see [NIST-800-38D]) and is one of the two modes that are currently documented for use with the CMS Authenticated-Enveloped structures.
The characteristics of the algorithm are:

1. The parameters of the algorithm are a nonce and the length of the tag to be generated.

2. The data actually authenticated is:

    a. The authenticated data,

    b. The encrypted message data,

    c. The length of the authenticated data,

    d. The length of the message data.

3. Before the first byte of message data can be processed, you must know:

    a. The nonce value.

    b. The authenticated data.

4. Before the first byte of authenticated data can be processed you must know: nothing.

This mode can easily be used in a stream model for senders provided the authenticated data is generated prior to the message data.
This mode can easily be used in a stream model for recipients provided that the authenticated data is presented prior to the message data.

3.6. IACBC: Integrity Aware Cipher Block Chaining

Integrity Aware Cipher Block Chaining is an algorithm that supports an AE mode of operation, but not an AEAD mode of operation. As such it does not matter where the authenticated parameters would be placed as they are not supported by the mode. This mode is therefore not of interest to this discussion.

3.7. IAPM: Integrity Aware Parallelizable Mode

Integrity Aware Parallelizable Mode is an algorithm that supports an AE mode of operation, but not an AEAD mode of operation. As such it does not matter where the authenticated parameters would be placed as they are not supported by the mode. This mode is therefore not of interest to this discussion.

3.8. OCB: Offset Codebook

Offset Codebook mode is an algorithm that supports an AE mode of operation, but not an AEAD mode of operation. As such it does not matter where the authenticated parameters would be placed as they are not supported by the mode. This mode is therefore not of interest to this discussion.
However, an addendum to the original mode submission described a method of adding the AEAD capability to any AE algorithm. This was described by Phillip Rogaway in [OCB-AD1] as section 5 and designated as Ciphertext Translation.
The characteristics of this algorithm are:

1. This mode adds no additional parameters to the underlying AE algorithm parameters.

2. The data actually authenticated is:

    a. The message data

b. The authenticated data

3. Before the first byte of data can be processed, you must know: the same information as for the AE mode by itself.

4. Before the first byte of authenticated data can be processed you must know: nothing.

It needs to be noted that before one can process the last t bytes of the message (for either encryption or decryption) the authenticated data must be known. The value t is equal to the length of the output function for the authenticated data processor. This does mean that an indication that one is in the last t bytes of processing the data is needed for both encryption and decryption modes.
The sender can operate using a streaming model as long as it buffers the last t bytes of message data so that it can be correctly tagged and sent to the cryptographic code as needing special processing. The authenticated data must be computed prior to the last t bytes of the encryption stream being produced. One possible way of dealing with this is to make the last t bytes the authentication tag as there is no explicit authentication tag created.
The recipient can operate using a streaming model as long as it buffers the last t bytes of encrypted data so that it can be correctly tagged when sent to the cryptographic code. As no separate authentication tag is created by the algorithm, the authenticated attributes must be presented prior to the last bytes of the encrypted data stream being decrypted.

3.9. PCFB: Propagating Cipher Feedback

Propagating Cipher Feedback is an algorithm that supports an AE mode of operation, but not an AEAD mode of operation. As such it does not matter where the authenticated parameters would be placed as they are not supported by the mode. This mode is therefore not of interest to this discussion.

3.10. SIV: Synthetic IV

The Synthetic IV (SIV) mode was designed and documented by Phillip Rogaway and Thomas Shrimpton. A full description of the algorithm can be found on the NIST website at [SIV].
The characteristics of the algorithm are:

1. The parameters of the algorithm are:

   a. None for the sender of the message

   b. An IV value for the recipient of the message. (The IV value acts as the authentication tag.)

2. The data actually authenticated is:

    a. The authenticated data

    b. The message data

3. Before the first byte of data can be processed, you must know:

    a. The authenticated attributes.

4. Before the first byte of authenticated data can be processed, you must know: nothing.

The algorithm does not use a nonce value, instead the IV used for the counter mode is computed from the authenticated data and message data. The IV is then emitted as the authentication tag. Note that this also means that the message data must processed twice by the cryptographic code. Once to do the authentication computation and produce the IV and one to do the counter mode encryption.
This algorithm cannot be streamed by the sender. Since the IV used for the counter mode encryption of the message data depends on all of the message data, the message data must actually be processed twice by the encryption algorithm.
The algorithm can easily be streamed by the recipient. The requirement is that the authenticated attributes and the IV be presented to the recipient before the message data is presented. The authentication check is then done by comparing the IV passed in with the IV computed.

3.11. XCBC: eXtended Cipher Block Chaining Encryption

eXtended Cipher Block Chaining Encryption is an algorithm that supports an AE mode of operation, but not an AEAD mode of operation. As such it does not matter where the authenticated parameters would be placed as they are not supported by the mode. This mode is therefore not of interest to this discussion.

3.12. MAC-Authenticated Encryption

The MAC-Authenticated Encryption mode has been documented by Peter Gutmann. This mode is documented in [GUTMANN].
The characteristics of the algorithm are:

1. The parameters of the algorithm are:

    a. A key derivation algorithm,

    b. A keyed MAC algorithm,

    c. An encryption algorithm

2. The data actually authenticated is:

    a. The encrypted message,

    b. The authenticated attributes.

3. Before the first byte of the message data can be processed, you must know: nothing.

4. Before the first byte of the authenticated data can be processed, you must know:

    a. The encrypted message data.

This algorithm can easily be used in a streaming model by the sender.
This algorithm can easily be used in a streaming model by the recipient.
Note: In the series of messages that I exchanged with Peter during the design of this algorithm, one of the things he noted was that to make streaming easier he should put the authenticated attributes after the message data. Thus the algorithm was designed to make sure that streaming worked well with the current encoding.

4. My Assumptions

This section will list the set of criteria that I am using in making my conclusions. Again, the most important thing in my mind is the ability to implement a streaming model for encode and decode operations.

1. We want to implement using a single pass streaming module to encode and decode the structures. There are many reasons to do so:

    1. The amount of resources used is minimized by not buffering the entirety of the message at each level of wrapping.

    2. The fact that not all messages are DER encode means that there is no single buffer in the original message that can be treated as a single input buffer.

    3. The message may be feed to the encoder/decode in chunks due to the way things are read from files, the fact that nodes in trees are emitted serially or the fact that removal of MIME content transfer encoding is normally done on small buffers.

    *There is one argument that says one should buffer up the entire encrypted buffer, decrypt in one chunk and then pass on the data in one piece. Since the name of the algorithm class is encrypted and authenticated, one should perhaps

actually authenticate that the data is correct prior to releasing the data for additional processing.

*I believe that it is sufficient to check that the encrypted buffer has been authenticated prior to acting on the data contained in the encrypted buffer. Thus I believe it makes sense to continue doing the decode and either fail on the decode operation and propagate a failure up either when the decode itself fails or when the authentication check is actually made. In this way it is no different than the processing of a signed message where the signature may be checked long after the message has been fully decoded. In fact this is the normal case for an S/MIME client where the content is often viewable with some indication that the validation of the signature failed for some reason.

2. The relative lengths of the data to be encrypted and the attributes to be protected are such that the encrypted data is generally much larger than the attributes. Thus if one has to cache one in a streaming mode, it is preferable to cache the attributes.

5. Conclusions

I now look again at the arguments presented in Section 2 and review the arguments presented. All of the opinions in this section are mine and may or may not be represent those of any other people. Section 6 contains the opinions of other people.

A foolish consistency is the hobgoblin of little minds,
adored by little statesmen and philosophers and divines.
(Ralph Waldo Emerson 1841)

1. Consistency with the existing CMS data types:

*This criteria should only be used a tie breaker in the event that all other criteria come out equal. When looking at this argument I am reminded of the following:

2. Authenticated attributes that are derived from the message content:

*This argument is slightly more believable than it was before I began this document as I now have an attribute which is derived from the message content, however this attribute is the length of the message data and in order to be useful it needs to be placed before the message data is consumed. (See Section 3.1.)

* I found this argument to be difficult to believe at the time it was presented, and I have not changed my mind since then. The argument that this means the authenticated attributes comes second would mean that this is an attribute that is attested to by the sender, but is not verified in any way by the recipient. If the recipient needed to do any processing then it would be much more desirable to have the attribute occur before the message data so that the recipient can setup to do the necessary processing prior to processing the message data.

* In the process of writing [XOR-HASH] I have become convinced that there is a fundamental problem which is going to be coming in the future with the signed data structure. Since the recipient does not "know" the correct set of hash algorithms to be used when processing a message the vast majority look at the list presented and then augment it with a number of different algorithms. This often means that one is computing four or five different hash functions on the content just on the off-chance that they may be needed. Many systems will not attempt a recovery if they find a signer info structure which uses a hash algorithm they did not realize that they needed even if it is known to the system because of the work involved in doing a restart after having parsed in all of the data. This means that similar behavior should be expected for any attributes that need to be validated by the recipient after having been generated by the sender. The problem is worse since there is no similar field to the set of digest algorithms that can filled at the beginning of a signed data object.

* I believe that this criteria was mis-applied. The issues of how a recipient was supposed to deal with these types of attributes was completely ignored in the decision process and it should have had paramount importance.

3. The decision should be dictated by Algorithm Characteristics:

* Looking at the taxonomy of algorithms that is presented in Section 3 we come up with the following results:

  - The algorithms which cannot be easily streamed are: CCM, SIV (for sender)

  - The algorithms which need attributes before the message body are: CWC (serialized implementation), GCM, SIV (for recipient), CCM (for recipient in special circumstances)

- The algorithms which need the message body before the attributes are: MAC-Authenticated

- The algorithms which can have either the body or the attributes first are: CWC (parallelized implementation), EAX, OCB

* We can see that CCM and SIV will never be easily streamed for the sender. It is unfortunate for people wanting to stream the CCM is one of the two algorithms that we have standardized on. It should be noted that both of these algorithms can be setup to be streamed for the recipient of the message, but CCM requires an additional restriction to be applied. If either of these algorithms is used then the entire question discussed above about a sender processing the content on sending would be academic as the message data needs to be buffered anyway.

* We have only one algorithm were the attributes are logically placed after the message data, that being the MAC-Authenticated, which was explicitly designed to be that way so that it could be streamed using the current data layout. Additionally there are two algorithms that are agnostic of the order of attributes and data plus one that can be implemented to be agnostic.

* For recipients, only the MAC-Authenticated algorithm necessitates that the attributes be cached until the message data has been processed. All of the other algorithms can be made work with the attributes preceding the message data without any problems.

* In current practice, and in part because of NIST standardization, the only two modes that have significant use are the CCM and GCM modes. It is possible that the MAC-Authenticated mode will also get traction since it is easy for people to understand and implement. This should also be taken into consideration when looking at the algorithm characteristics.

* If we had done this analysis at the time the decision was made then we should have made the decision to place the attributes first.

4. Resource requirements for the sender and recipient:

* It is no more likely that the sender of a message is resource constrained than it is for the recipient of the message to be resource constrained. This means that it is better for a set of algorithms and layout to be chosen that

will work well in a streaming model under normal circumstances than to optimize for either the sender or the recipient.

5. Relative frequency of processing:

   *In my opinion, most of the time messages that are created using an authenticated encryption algorithm will be decrypted by at least one recipient. Messages which are not decrypted will exist, either from being lost in the ether or from being cached until needed, but these will be the smallest part of the set. Messages which need to be decrypted multiple times by a single recipient will generally be a small number as well, unless it because part of the S/MIME standard. However I believe that a significant number of messages will be created that will have multiple recipients. This may be done by creating multiple lock boxes up front, or by creating the lock boxes on demand in cases where it does not matter than a traffic analysis can be done that multiple recipients have gotten the same message. (An example of this might be sending a firmware upgrade to multiple devices, where the message is transferred on demand and it does not matter that an observer can see that the same set of firmware is being installed on multiple machines. This would be something that could probably be assumed anyway.)

   *I therefore think that overall more messages will be decoded and decrypted than encrypted and encoded. This would mean that a bias should be placed for the recipients of messages not the sender of messages in making decisions.

Based on the above, I would say that we should modify the order of these fields in the event that the document is updated.

6. Responses

An opportunity was provided to the Russ Housley as the author of [CMS-AED] and to others that were involved on the mailing list to provide a formal response. Nobody took advantage of the offer.

7. Security Considerations

This document discusses a security related document, however it makes no changes to the document. As such there are no actual security implications for this document.

8. IANA Considerations

No action by IANA is required for this document.

9. References

9.1. Normative References

[RFC3610]	Whiting, D., Housley, R. and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.
[CMS]	Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.
[CMS-AED]	Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, November 2007.
[GUTMANN]	Gutmann, P., "Using MAC-authenticated Encryption in the Cryptographic Message Syntax (CMS)", .
[NIST-800-38C]	Dworkin, M., "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality", NIST Special Publication 800-38C, May 2004.
[NIST-800-38D]	Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, November 2007.
[CWC]	Kohno, T., Viega, J. and D. Whiting, "The CWC authenticated encryption (associated data) mode", May 2003.
[EAX]	Bellare, M., Rogaway, P. and D. Wagner, "EAX: A Conventional Authenticated-Encryption Mode", 2003.
[OCB-AD1]	Rogaway, P., "The Associated-Data Problem", November 2001.
[SIV]	Rogaway, P. and T. Shrimpton, "The SIV Mode of Operation for Deterministic Authenticated-Encryption (Key Wrap) and Misuse-Resistant Nonce-Based Authenticated-Encryption", August 2007.

9.2. Informative References

[XOR-HASH]	Schaad, J, "Experiment: Hash functions with parameters in CMS and S/MIME", Internet-Draft draft-schaad-smime-hash-experiment-06, January 2011.

Author's Address

Jim Schaad Schaad Soaring Hawk Consulting EMail: jimsch@augustcellars.com