Table of contents

1 Comments on NIST Draft Pub 800-38C : CCM mode of operation

NIST Draft Pub 800-38C specifies the so-called CCM mode, a mode of operation that operates on block-ciphers with a 128-bit block size and involves a particular combination of the so-called Counter (CTR) mode of operation and the Cipher-Block Chaining (CBC) mode of operation [10], using a single key. Appendix A of this draft specification gives a particular invocation of the generic CCM mode, which coincides with the CCM specification, as contained in the Draft Amendment (as of July 2003) to the IEEE 802.11 WLAN standard [5].

1.1 General Comments

It is unclear what criteria NIST applied to decide on standardization of the CCM mode of operation proposed in [4], rather than one of the alternative proposed combined encryption and authentication modes. In particular, the CCM mode has some disadvantages not shared by some of these alternatives, such as it being only defined with block ciphers with 128-bit block size and it requiring the length of the input data to be known beforehand. This being said, it is laudable that NIST recognizes the relevance the CCM mode of operation has acquired, due to the incorporation hereof in quite a few wireless standards that recently emerged, including the IEEE 802.11 WLAN standard [5], and the IEEE 802.15 High-Rate and Low-Rate WPAN standards [6], [7].

Below, I will give some more detailed comments on the draft and some suggestions for its improvement. Most comments are inspired by the particular invocation of the CCM mode in Appendix A of the draft specification, which corresponds to the original CCM specification proposed in [4].

1.2 Specific comments

1.2.1 Generality and usefulness of CCM mode

The draft CCM specification is at times very general and at times too restrictive.

1. NIST elected to specify the CCM mode in such a general way that implementers of the draft standard have ample room to select their own formatting and counter generating functions. Although the conditions under which the proof of the CCM mode [8], [9] applies might justify some generality, it is hard to see how this generality would promote interoperability, a major objective of standardization.
2. Despite the general approach followed, the draft CCM specification is unnecessarily restricted elsewhere: it is very well possible to define the CCM mode without completely fixing the bit representation of integers (e.g., by representing integers as octet strings and leaving the representation of octet strings as bit strings up to the implementation environment). For an example of how this may be realized, see Section 2 of this review note.
3. The original CCM mode[4] provides for data authentication and, possibly, confidentiality, but does not provide for confidentiality only. This is unfortunate, since not all implementation environments call for data authenticity (e.g., since data authenticity is provided by an external mechanism). It is a pity that NIST has not seized the opportunity to extend the definition of the original CCM mode, such as to provide for confidentiality-only services, in addition to the other security service options already offered. For an example of how to extend the original CCM mode to provide for any combination of data authenticity and confidentiality, see Section 2 of this review note.

4. The original CCM mode [4] is known to be vulnerable to specific attacks, if used with variable-length authentication tags rather than with fixed-length authentication tags only (see, e.g., Section 3.4 of [12]). Thus, the original CCM mode can only be used in settings with fixed-length authentication tags. It is a pity that NIST does not seem to have incorporated the results of that paper, such as to avoid these attacks altogether. For an example of how to adapt the original CCM mode such that the resulting mode can be used securely with variable-length authentication tags, rather than fixed-length authentication tags only, see, again, Section 2 of this review note. (Variable-length authentication tags are useful in, e.g., secured wireless sensor networks [7], where applications on a device might have different protection requirements, but would have to share the same key, due to resource constraints.)

1.2.2 Other specific comments

1.2.2.1 Section 3

1. Page 4, line 2: replace 'that can provide … data' by 'that may provide assurances of the authenticity and, possibly, confidentiality of data'.
2. Page 4, line –3: replace 'CCM shall not be used with 3DES' by 'CCM cannot be used with 3DES' (after all, the block size of 3DES does not fit with the use of CCM).
3. Page 5, line 8: it is strange that the number of block cipher invocations with the same key is expressed in terms of number of octets. Replace by 'The generation-encryption process of the CCM mode shall invoke at most 2^{60} block cipher calls with the same key'. The rationale for this constraint is unclear to me: why not having as upper bound of 2^{64} block cipher invocations?
4. Page 5, line –1: the reference to the IEEE 802.11 WLAN standard is missing. See [5] in Section 2.6 of this review note.

1.2.2.2 Section 4

Page 5, Item 2 ('Authenticity'): replace 'The property that data originated from its purported source' by 'Evidence that data originated from its purported source'.

Page 6, Item 8 ('Data inegrity'): the term 'unauthorized entity' is not defined.

Page 6: Add definition of 'Inverse cipher function'.

Page 7, Item 2 ('m'): According to Section 6.1, Step 5, the integer *m* refers to the length of the payload, in number of blocks (rounded upwards), rather than to the number of blocks in the formatted input (note that formatted input also contains, e.g., associated data).

Page 7, Item –6: add a 'full stop' at the end.

1.2.2.3 Section 5

1. Section 5.1, Page 9, line –1, -2: the bit length of the AES key is irrelevant in this specification. Replace by 'The CCM key is the block cipher key *K* of length *Klen* bits. With this key, the forward cipher function of the block cipher is denoted $CIPH_K$.
2. Section 5.2, Page 9, line 10: the term 'data origin authentication' is not defined in Section 4.1.
3. Section 5.3, Page 9, lines 1-5: what about encryption-only? See also Comment 3 of Section 1.2.1 of this review note.
4. Section 5.3, Page 9, lines 6-9: to allow the secure use of the CCM mode with variable-length authentication tags, one has to impose the following additional constraint: "one shall be able to uniquely determine the length of the applicable authentication tag from the counters blocks". See also Comment 4 of Section 1.2.1 of this review note and Section 2.3 of this review note.
5. Section 5.3, Page 9, footnote: the term 'mode' seems to have a specific meaning. Replace by 'authentication mode' by 'authentication provision'.

6. Section 5.4, Page 10, lines 7-8 (Property 2): this property contains some typos, me thinks. Why not replace the whole property by the following description?: 'The authentication transformation operates on input strings $B_0 \| B_1 \| B_2 \| ... \| B_t$ from which one can uniquely determine the input strings *a* and *m* (as well as the nonce *N*). In fact, for any two input strings corresponding to distinct triples *(N, m, a)*, neither one is a prefix string of the other.
7. Section 5.4, Page 10, lines 9-10 (Property 3): I suggest replacing this property by the following: "Over the lifetime of the key, all the counter blocks are distinct from the B_0 fields *that are actually used*". This slightly relaxes the conditions under which the CCM security proof applies. For a further security discussion on this, see also Section 2.3 of these review notes.
8. Section 5.4, Page 10, line 11: replace 'implies' by 'suggests'. After all, a logical implication cannot yield a non-mandatory statement ('should not').
9. Section 5.5, Page 10, line 1: there does not seem to be a need to completely specify the representation of integers within the CCM specification. See also Comment 2 of Section 1.2.1 of this review note.

1.2.2.4 Section 6

1. Section 6.1, Page 11, Step 3: replace 'Y_j' by 'Y_i', i.e., replace the subscript '*j*' by the subscript '*i*'.
2. Section 6.1, Page 11, line –2, -1: the valid nonce is not formatted according to the formatting function, as suggested. I suggest replacement of this sentence by the following: "The input data to the generation-encryption function are a valid nonce, a valid payload string, and a valid associated data string. The input data is transformed using the formatting function."
3. Section 6.1, Page 11-12: the upper bound on the number of block cipher invocations, as hinted at in Section 3, Page 5, line 8 (see also Comment 3 of Section 1.2.2.1 of this review note) is not included as a mandatory requirement in the CCM mode specification. Either the requirement is real, in which case it needs to be included, or it is unreal, in which case it needs to be dropped from Section 3 of the draft specification document.

1.2.2.5 Appendix A

1. Appendix A2.1, Page 15, line 6: Replace "The *Reserved* bit … is set to '0'" by "The *Reserved* bit … shall be set to '0'".
2. Appendix A, Pages 14-16: One can generalize the original CCM mode, to allow a wider application appeal. See Section 2 of this review note. This extension is compatible with the CCM mode as used in, e.g., the 802.11 WLAN standard (See Section 2.4 of this review note).

1.2.2.6 Appendix B

1. Appendix B, Page 17, 3rd paragraph: the rationale for the so-called 'default recommendation' for *Tlen* is unclear to me. Why not use a formulation similar to Appendix B in FIPS Pub 198 [3]? From a practical perspective, the 'proper' value of the length of the authentication tag is always the result of a risk-benefit analysis. Any absolute statement on 'proper' authentication lengths is, therefore, to be disadvised.

1.2.2.7 Appendix C

1. Appendix C, Page 18, line 3: replace 'the the' by 'the'.

1.2.2.8 Appendix D

Appendix D, Page 22: add the following references:

[1] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.11-1999, IEEE Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements – Part

11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, New York: IEEE Press, 1999.

[2] J. Jonsson, On the Security of CTR + CBC-MAC, in Proceedings of Selected Areas in Cryptography – SAC 2002, K. Nyberg, H. Heys, Eds., Lecture Notes in Computer Science, Vol. 2595, pp. 76-93, Berlin: Springer, 2002.

[3] P. Rogaway, D. Wagner, A Critique of CCM, IACR ePrint Archive 2003-070, April 13, 2003.

2 Formal specification of the generic CCM* mode of operation

CCM* is a generic combined encryption and authentication block cipher mode. CCM* is only defined for use with block ciphers with a 128-bit block size, such as AES-128 [2]. The CCM* ideas can easily be extended to other block sizes, but this will require further definitions.

The CCM* mode coincides with the original CCM mode specification ([4], Appendix A of [10]) for messages that require authentication and, possibly, encryption, but does also offer support for messages that require only encryption. As with the CCM mode, the CCM* mode requires only one key. The security proof for the CCM mode [8], [9] carries over to the CCM* mode described here. The design of the CCM* mode takes into account the results of [12], thus allowing it to be securely used in implementation environments for which the use of variable-length authentication tags, rather than fixed-length authentication tags only, is beneficial.

2.1 Notation and representation

2.1.1 Strings and string operations

A string is a sequence of symbols over a specific set (e.g., the binary alphabet {0,1} or the set of all octets). The length of a string is the number of symbols it contains (over the same alphabet). The right-concatenation of two strings x and y (over the same alphabet) of length m and n respectively (notation: $x \,||\, y$), is the string z of length $m+n$ that coincides with x on its leftmost m symbols and with y on its rightmost n symbols. An octet is a symbol string of length 8. In our context, all octets are strings over the binary alphabet.

2.1.2 Integers and their representation

Throughout this specification, the representation of integers as octet strings and of octets as binary strings shall be fixed. All integers shall be represented as octet strings in most-significant-octet first order. This representation conforms to the conventions in Section 4.3 of ANSI X9.63-2001 [1].

2.2 Specification of CCM* mode of operation (in 'ANSI style')

Prerequisites: The following are the prerequisites for the operation of the generic CCM* mode:

1. A block-cipher encryption function E shall have been chosen, with a 128-bit block size. The length in bits of the keys used by the chosen encryption function is denoted by *keylen*.
2. A fixed representation of octets as binary strings shall have been chosen (e.g., most-significant-bit first order or least-significant-bit-first order).
3. The length L of the message length field, in octets, shall have been chosen. Valid values for L are the integers 2, 3, ..., 8 (the value L=1 is reserved).
4. The length M of the authentication field, in octets, shall have been chosen. Valid values for M are the integers 0, 4, 6, 8, 10, 12, 14, and 16. (The value M=0 corresponds to disabling authenticity, since then the authentication field is the empty string.)

2.2.1 CCM* mode encryption and authentication transformation

Input: The CCM* mode forward transformation takes as inputs:

1. A bit string *Key* of length *keylen* bits to be used as the key. Each entity shall have evidence that access to this key is restricted to the entity itself and its intended key sharing group member(s).
2. A nonce N of 15-L octets. Within the scope of any encryption key *Key*, the nonce value shall be unique.

3. An octet string *m* of length *l(m)* octets, where $0 \le l(m) < 2^{8L}$.
4. An octet string *a* of length *l(a)* octets, where $0 \le l(a) < 2^{64}$.

The nonce *N* shall encode the potential values for *M* such that one can uniquely determine from *N* the actually used value of *M*. The exact format of the nonce *N* is outside the scope of this specification and shall be determined and fixed by the actual implementation environment of the CCM* mode.

Note: The exact format of the nonce *N* is left to the application, to allow simplified hardware and software implementations in particular settings. Actual implementations of the CCM* mode may restrict the values of *M* that are allowed throughout the life-cycle of the encryption key *Key* to a strict subset of those allowed in the generic CCM* mode. If so, the format of the nonce *N* shall be such that one can uniquely determine from *N* the actually used value of *M* in that particular subset. In particular, if *M* is fixed and the value *M*=0 is not allowed, then there are no restrictions on *N*, in which case the CCM* mode reduces to the CCM mode.

2.2.1.1 Input transformation

This step involves the transformation of the input strings *a* and *m* to the strings *AuthData* and *PlainTextData*, to be used by the authentication transformation and the encryption transformation, respectively.

This step involves the following steps, in order:

1. Form the octet string representation *L(a)* of the length *l(a)* of the octet string *a*, as follows:
   a. If *l(a)*=0, then *L(a)* is the empty string.
   b. If $0 < l(a) < 2^{16}-2^8$, then *L(a)* is the 2-octets encoding of *l(a)*.
   c. If $2^{16}-2^8 \le l(a) < 2^{32}$, then *L(a)* is the right-concatenation of the octet 0xff, the octet 0xfe, and the 4-octets encoding of *l(a)*.
   d. If $2^{32} \le l(a) < 2^{64}$, then *L(a)* is the right-concatenation of the octet 0xff, the octet 0xff, and the 8-octets encoding of *l(a)*.

2. Right-concatenate the octet string *L(a)* with the octet string *a* itself. Note that the resulting string contains *l(a)* and *a* encoded in a reversible manner.

3. Form the padded message *AddAuthData* by right-concatenating the resulting string with the smallest non-negative number of all-zero octets such that the octet string *AddAuthData* has length divisible by 16.

4. Form the padded message *PlaintextData* by right-concatenating the octet string *m* with the smallest non-negative number of all-zero octets such that the octet string *PlaintextData* has length divisible by 16.

5. Form the message *AuthData* consisting of the octet strings *AddAuthData* and *PlaintextData*:

   $$AuthData = AddAuthData \,||\, PlaintextData.$$

2.2.1.2 Authentication transformation

The data *AuthData* that was established above shall be tagged using the tagging transformation as follows:

1. Form the 1-octet *Flags* field consisting of the 1-bit *Reserved* field, the 1-bit *Adata* field, and the 3-bit representations of the integers *M* and *L*, as follows:

   $$Flags = Reserved \,||\, Adata \,||\, M \,||\, L.$$

   Here, the 1-bit *Reserved* field is reserved for future expansions and shall be set to '0'. The 1-bit *Adata* field is set to '0' if *l(a)*=0, and set to '1' if *l(a)*>0. The *M* field is the 3-bit representation of the integer (*M*-2)/2 if *M>0* and of the integer 0 if *M*=0, in most-significant-bit-first order. The *L* field is the 3-bit representation of the integer *L*-1, in most-significant-bit-first order.

2. Form the 16-octet B_0 field consisting of the 1-octet *Flags* field defined above, the 15-*L* octet nonce field *N*, and the *L*-octet representation of the length field *l(m)*, as follows:

$$B_0 = Flags \,||\, Nonce\ N \,||\, l(m).$$

3. Parse the message *AuthData* as $B_1 \,\|\, B_2 \,||\, \ldots \,||B_t$, where each message block B_i is a 16-octet string.

4. The CBC-MAC value X_{t+1} is defined by

$$X_0 := 0^{128};\ X_{i+1} := E(Key,\ X_i \oplus B_i) \quad \text{for } i=0, \ldots, t.$$

Here, $E(K, x)$ is the cipher-text that results from encryption of the plaintext *x*, using the established block-cipher encryption function *E* with key *Key*; the string 0^{128} is the 16-octet all-zero bit string.

5. The authentication tag *T* is the result of omitting all but the leftmost *M* octets of the CBC-MAC value X_{t+1} thus computed.

2.2.1.3 Encryption transformation

The data *PlaintextData* that was established in clause 2.2.1.1 (step 4) and the authentication tag *T* that was established in clause 2.2.1.2 (step 5) shall be encrypted using the encryption transformation as follows:

1. Form the 1-octet *Flags* field consisting of two 1-bit *Reserved* fields, and the 3-bit representations of the integers *0* and *L*, as follows:

$$Flags = Reserved \,||\, Reserved \,||\, 0 \,||\, L.$$

Here, the two 1-bit *Reserved* fields are reserved for future expansions and shall be set to '0'. The *'0'* field is the 3-bit representation of the integer 0, in most-significant-bit-first order. The *L* field is the 3-bit representation of the integer *L*-1, in most-significant-bit-first order.

2. Define the 16-octet A_i field consisting of the 1-octet *Flags* field defined above, the 15-*L* octet nonce field *N*, and the *L*-octet representation of the integer *i*, as follows:

$$A_i = Flags \,||\, Nonce\ N \,||\, Counter\ i, \text{ for } i=0, 1, 2, \ldots$$

Note that this definition ensures that all the A_i fields are distinct from the B_0 fields that are actually used, as those have a *Flags* field with a non-zero encoding of *M* in the positions where all A_i fields have an all-zero encoding of the integer 0 (see clause 2.2.1.2, step 2).

3. Parse the message *PlaintextData* as $M_1 \,\|\, \ldots \,||M_t$, where each message block M_i is a 16-octet string.

4. The ciphertext blocks $C_1, \ldots, C_t$ are defined by

$$C_i := E(\ Key,\ A_i\ ) \oplus M_i \text{ for } i=1, 2, \ldots, t.$$

5. The string *Ciphertext* is the result of omitting all but the leftmost *l(m)* octets of the string $C_1 \,||\, \ldots \,||\, C_t$.

6. Define the 16-octet encryption block S_0 by

$$S_0 := E(\ Key,\ A_0\ ).$$

7. The encrypted authentication tag *U* is the result of XOR-ing the string consisting of the leftmost *M* octets of S_0 and the authentication tag *T*.

Output: If any of the above operations has failed, then output 'invalid'. Otherwise, output the right-concatenation of the encrypted message *Ciphertext* and the encrypted authentication tag *U.*

2.2.2 CCM* mode decryption and authentication checking transformation

Input: The CCM* inverse transformation takes as inputs:

1. A bit string *Key* of length *keylen* bits to be used as the key. Each entity shall have evidence that access to this key is restricted to the entity itself and its intended key-sharing group member(s).
2. A nonce *N* of 15-*L* octets. Within the scope of any encryption key *Key*, the nonce value shall be unique.
3. An octet string *c* of length *l(c)* octets, where $0 \le l(c)\text{-}M < 2^{8L}$.
4. An octet string *a* of length *l(a)* octets, where $0 \le l(a) < 2^{64}$.

2.2.2.1 Decryption transformation

The decryption transformation involves the following steps, in order:

1. Parse the message *c* as $C \| U$, where the right-most string *U* is an *M*-octet string. If this operation fails, output 'invalid' and stop. *U* is the purported encrypted authentication tag. Note that the leftmost string *C* has length *l(c)-M* octets.
2. Form the padded message *CiphertextData* by right-concatenating the string *C* with the smallest non-negative number of all-zero octets such that the octet string *CiphertextData* has length divisible by 16.
3. Use the encryption transformation in clause 2.2.1.3, with as inputs the data *CipherTextData* and the tag *U.*
4. Parse the output string resulting from applying this transformation as $m \| T$, where the right-most string *T* is an *M*-octet string. *T* is the purported authentication tag. Note that the leftmost string *m* has length *l(c)-M* octets.

2.2.2.2 Authentication checking transformation

The authentication checking transformation involves the following steps, in order:

1. Form the message *AuthData* using the input transformation in Clause 2.2.1.1, with as inputs the string *a* and the octet string *m* that was established in clause 2.2.2.1 (step 4).
2. Use the authentication transformation in Clause 2.2.1.2, with as input the message *AuthData*.
3. Compare the output tag *MACTag* resulting from this transformation with the tag *T* that was established in clause 2.2.2.1 (step 4). If *MACTag=T*, output 'valid'; otherwise, output 'invalid' and stop.

Output: If any of the above verifications has failed, then output 'invalid' and reject the octet string *m.* Otherwise, accept the octet string *m* and accept one of the key sharing group member(s) as the source of *m.*

2.2.3 Restrictions

All implementations shall limit the total amount of data that is encrypted with a single key. The CCM* encryption transformation shall invoke not more than 2^{61} block-cipher encryption function operations in total, both for the CBC-MAC and for the CTR encryption operations.

At CCM* decryption, one shall verify the (truncated) CBC-MAC before releasing any information, such as, e.g., plaintext. If the CBC-MAC verification fails, only the fact that the CBC-MAC verification failed shall be exposed; all other information shall be destroyed.

2.3 Security of CCM* mode of operation

The CCM* mode coincides with the original CCM mode specification [4] for messages that require authentication and, possibly, encryption, but also offers support for messages that require only encryption. As with the CCM mode, the CCM* mode requires only one key. The CCM* specification differs from the CCM specification, as follows:

- The CCM* mode allows the length of the authentication field M to be zero as well (the value M=0 corresponding to disabling authenticity, since then the authentication field is the empty string).
- The CCM* mode imposes a further restriction on the nonce N: it shall encode the potential values for M such that one can uniquely determine from N the actually used value of M.

As a result, if M is fixed and the value M=0 is not allowed, then there are no additional restrictions on N, in which case the CCM* mode reduces to the CCM mode. In particular, the proof of the CCM mode (see [8], [9]) applies.

For fixed-length authentication tags, the CCM* mode is equally secure as the original CCM mode. For variable-length authentication tags, the CCM* mode completely avoids – by design – the vulnerabilities that do apply to the original CCM mode.

For fixed-length authentication tags, the security proof of the original CCM mode carries over to that of the CCM* mode (also for $M=0$), by observing that the proof of the original CCM mode relies on the following properties, which slightly relax those stated in [8], [9] (relaxed property indicated in italics):

- The B_0 field uniquely determines the value of the nonce N.
- The authentication transformation operates on input strings $B_0 \| B_1 \| B_2 \| \ldots \| B_t$ from which one can uniquely determine the input strings a and m (as well as the nonce N). In fact, for any two input strings corresponding to distinct triples (N, m, a), neither one is a prefix string of the other.
- All the A_i fields are distinct from the B_0 fields *that are actually used* (over the lifetime of the key), as those have a *Flags* field with a non-zero encoding of M in the positions where all A_i fields have an all-zero encoding of the integer 0.

Hence, if M is fixed, then the CCM* mode offers the same security properties as the original CCM mode: confidentiality over the input string m and data authenticity over the input strings a and m, relative to the length of the authentication tag. Obviously, if M=0, then no data authenticity is provided by the CCM* mode itself (but may be provided by an external mechanism).

For variable-length authentication tags, the original CCM mode is known to be vulnerable to specific attacks (see, e.g., Section 3.4 of [12]). These attacks may arise with the original CCM mode, since the decryption transformation does not depend on the length of the authentication tag itself. The CCM* mode avoids these attacks altogether, by requiring that one shall be able to uniquely determine the length of the applicable authentication tag from the A_i fields (i.e., from the counters blocks).

2.4 Interoperability between CCM mode and CCM* mode of operation

The CCM* mode reduces to the CCM mode in all implementation environments where the length of the authentication tag is fixed and where the value M=0 (encryption-only) is not allowed. In particular, the CCM* mode is compatible with the CCM mode, as specified in the Draft Amendment (as of July 2003) to the IEEE 802.11 WLAN standard [5] and as specified in the IEEE 802.15.3 WPAN standard [6]. The IEEE 802.15.4 WPAN standard [7] currently incorporates the CCM mode with variable-length authentication tags; the upcoming security amendment is anticipated to involve replacement of the CCM mode by the CCM* mode of operation, to securely support variable-length authentication tags in its target application area – low-cost sensor networks.

2.5 Test vectors for CCM* mode of operation (in 'ANSI style')

Prerequisites: The following prerequisites are established for the operation of the mode of operation:

The block-cipher mode of operation used in this specification shall be the CCM* mode of operation, as specified in clause 2.2, with the following instantiations:

1. Each entity shall use the block-cipher AES-128 as specified in [2];
2. All octets shall be represented in most-significant-bit-first order;
3. The parameter *L* shall have the integer value 2;
4. The parameter *M* shall have the integer value 8.

2.5.1 CCM* mode encryption and authentication transformation

Input: The inputs to the mode of operation are:

1. The key *Key* of size *keylen*=128 bits to be used:

   Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.

2. The nonce *N* of 15-*L*=13 octets to be used:

   Nonce = A0 A1 A2 A3 A4 A5 A6 A7 || 03 02 01 00 || 06.

3. The octet string *m* of length *l(m)*=23 octets to be used:

   m = 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E.

4. The octet string *a* of length *l(a)*=8 octets to be used:

   a = 00 01 02 03 04 05 06 07.

2.5.1.1 Input transformation

This step involves the transformation of the input strings *a* and *m* to the strings *AuthData* and *PlainTextData*, to be used by the authentication transformation and the encryption transformation, respectively.

1. Form the octet string representation *L(a)* of the length *l(a)* of the octet string *a*:

   L(a) = 00 08.

2. Right-concatenate the octet string *L(a)* and the octet string *a* itself:

   L(a) || *a* =00 08 || 00 01 02 03 04 05 06 07.

3. Form the padded message *AddAuthData* by right-concatenating the resulting string with the smallest non-negative number of all-zero octets such that the octet string *AddAuthData* has length divisible by 16.

   AddAuthData = 00 08 || 00 01 02 03 04 05 06 07 || 00 00 00 00 00 00.

4. Form the padded message *PlaintextData* by right-concatenating the octet string *m* with the smallest non-negative number of all-zero octets such that the octet string *PlaintextData* has length divisible by 16:

   PlaintextData = 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 ||
   18 19 1A 1B 1C 1D 1E || 00 00 00 00 00 00 00 00 00.

5. Form the message *AuthData* consisting of the octet strings *AddAuthData* and *PlaintextData*:

   AuthData = 00 08 00 01 02 03 04 05 06 07 00 00 00 00 00 00 ||
   08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
   18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00.

2.5.1.2 Authentication transformation

The data *AuthData* that was established above shall be tagged using the tagging transformation as follows:

1. Form the 1-octet *Flags* field as follows:

   Flags = 59.

2. Form the 16-octet B_0 field as follows:

   B_0 = 59 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 17.

3. Parse the message *AuthData* as $B_1 \| B_2 \| B_3$, where each message block B_i is a 16-octet string.

4. The CBC-MAC value X_4 is computed as follows:

i	***B_i***	***X_i***
0	59 A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 00 17	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1	00 08 00 01 02 03 04 05 06 07 00 00 00 00 00 00	F7 74 D1 6E A7 2D C0 B3 E4 5E 36 CA 8F 24 3B 1A
2	08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17	90 2E 72 58 AE 5A 4B 5D 85 7A 25 19 F3 C7 3A B3
3	18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00	5A B2 C8 6E 3E DA 23 D2 7C 49 7D DF 49 BB B4 09
4	—	B9 D7 89 67 04 BC FA 20 B2 10 36 74 45 F9 83 D6

5. The authentication tag *T* is the result of omitting all but the leftmost *M*=8 octets of the CBC-MAC value X_4:

   T = B9 D7 89 67 04 BC FA 20.

2.5.1.3 Encryption transformation

The data *PlaintextData* shall be encrypted using the encryption transformation as follows:

1. Form the 1-octet Flags field as follows:

   Flags = 01.

2. Define the 16-octet A_i field as follows:

i	***A_i***
0	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 00
1	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 01
2	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 02

3. Parse the message *PlaintextData* as $M_1 \| M_2$, where each message block M_i is a 16-octet string.

4. The ciphertext blocks C_1, C_2 are computed as follows:

i	***AES(Key,A_i)***	***C_i = AES(Key,A_i)*** Å ***M_i***
1	12 5C A9 61 B7 61 6F 02 16 7A 21 66 70 89 F9 07	1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10
2	CC 7F 54 D1 C4 49 B6 35 46 21 46 03 AA C6 2A 17	D4 66 4E CA D8 54 A8 35 46 21 46 03 AA C6 2A 17

5. The string *Ciphertext* is the result of omitting all but the leftmost *l(m)*=23 octets of the string $C_1 \| C_2$:

   CipherText = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8.

6. Define the 16-octet encryption block S_0 by

   $S_0 = E(Key, A_0) =$ B3 5E D5 A6 DC 43 6E 49 D6 17 2F 54 77 EB B4 39.

7. The encrypted authentication tag *U* is the result of XOR-ing the string consisting of the leftmost *M=8* octets of S_0 and the authentication tag *T*:

   U=0A 89 5C C1 D8 FF 94 69.

Output: the right-concatenation *c* of the encrypted message *Ciphertext* and the encrypted authentication tag *U*:

c = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8 ||
0A 89 5C C1 D8 FF 94 69.

2.5.2 CCM* mode decryption and authentication checking transformation

Input: The inputs to the inverse mode of operation are:

1. The key *Key* of size *keylen*=128 bits to be used:
   Key = C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF.
2. The nonce *N* of 15-*L*=13 octets to be used:
   Nonce = A0 A1 A2 A3 A4 A5 A6 A7 || 03 02 01 00 || 06.
3. The octet string *c* of length *l(c)*=31 octets to be used:
   c = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8 ||
   0A 89 5C C1 D8 FF 94 69.
4. The octet string *a* of length *l(a)*=8 octets to be used:
   a = 00 01 02 03 04 05 06 07.

2.5.2.1 Decryption transformation

The decryption transformation involves the following steps, in order:

1. Parse the message *c* as $C \| U$, where the right-most string *U* is an *M*-octet string:
   C = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 || D4 66 4E CA D8 54 A8;
   U = 0A 89 5C C1 D8 FF 94 69.
2. Form the padded message *CiphertextData* by right-concatenating the string *C* with the smallest non-negative number of all-zero octets s uch that the octet string *CiphertextData* has length divisible by 16.
   CipherTextData = 1A 55 A3 6A BB 6C 61 0D 06 6B 33 75 64 9C EF 10 ||
   D4 66 4E CA D8 54 A8 || 00 00 00 00 00 00 00 00.
3. Form the 1-octet Flags field as follows:
   Flags = 01.
4. Define the 16-octet A_i field as follows:

i	A_i
0	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 00
1	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 01
2	01 || A0 A1 A2 A3 A4 A5 A6 A7 03 02 01 00 06 || 00 02

5. Parse the message *CiphertextData* as $C_1 \| C_2$, where each message block C_i is a 16-octet string.
6. The ciphertext blocks P_1, P_2 are computed as follows:

i	***AES(Key,A$_i$)***	***P$_i$= AES(Key,A$_i$)*** Å ***C$_i$***
1	12 5C A9 61 B7 61 6F 02 16 7A 21 66 70 89 F9 07	08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
2	CC 7F 54 D1 C4 49 B6 35 46 21 46 03 AA C6 2A 17	18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00

7. The octet string *m* is the result of omitting all but the leftmost *l(m)*=23 octets of the string $P_1 \| P_2$:
   m = 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 || 18 19 1A 1B 1C 1D 1E.

8. Define the 16-octet encryption block S_0 by

   $S_0 = E(Key, A_0) =$ B3 5E D5 A6 DC 43 6E 49 D6 17 2F 54 77 EB B4 39.

9. The purported authentication tag T is the result of XOR-ing the string consisting of the leftmost $M=8$ octets of S_0 and the octet string U:

   T = B9 D7 89 67 04 BC FA 20.

2.5.2.2 Authentication checking transformation

The authentication checking transformation involves the following steps, in order:

1. Form the message *AuthData* using the input transformation in Clause 2.5.1.1, with as inputs the string a and the octet string m that was established in clause 2.5.2.1(step 7):

   AuthData = 08 00 01 02 03 04 05 06 07 00 00 00 00 00 00 00 ||
   08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
   18 19 1A 1B 1C 1D 1E 00 00 00 00 00 00 00 00 00.

2. Use the authentication transformation in Clause 2.5.1.2, with as input the message *AuthData* to compute the authentication tag *MACTag*:

   MACTag = B9 D7 89 67 04 BC FA 20.

3. Compare the output tag *MACTag* resulting from this transformation with the tag T that was established in clause 2.5.2.1(step 9):

   T = B9 D7 89 67 04 BC FA 20 = *MACTag*.

Output: Since *MACTag*=T, output 'valid' and accept the octet string m and accept one of the key sharing group member(s) as the source of m.

2.6 References

[1] ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography, American Bankers Association, November 20, 2001. Available from http://www.ansi.org.

[2] FIPS Pub 197, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T, Springfield, Virginia, November 26, 2001. Available from http://csrc.nist.gov/.

[3] FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198, US Department of Commerce/N.I.S.T., Springfield, Virginia, March 6, 2002. Available from http://csrc.nist.gov/.

[4] R. Housley, D. Whiting, N. Ferguson, Counter with CBC-MAC (CCM), submitted to N.I.S.T., June 3, 2002. Available from http://csrc.nist.gov/encryption/modes/proposedmodes/.

[5] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.11-1999, IEEE Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, New York: IEEE Press, 1999.

[6] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.3-2003, IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPAN). New York: IEEE Press. 2003.

[7] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4-2003, IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPAN). New York: IEEE Press. 2003.

[8] J. Jonsson, On the Security of CTR + CBC-MAC, in Proceedings of Selected Areas in Cryptography – SAC 2002, K. Nyberg, H. Heys, Eds., Lecture Notes in Computer Science, Vol. 2595, pp. 76-93, Berlin: Springer, 2002.

[9] J. Jonsson, On the Security of CTR + CBC-MAC, NIST Mode of Operation – Additional CCM Documentation. Available from http://csrc.nist.gov/encryption/modes/proposedmodes/.

[10] NIST Pub 800-38A 2001 ED, Recommendation for Block Cipher Modes of Operation – Methods and Techniques, NIST Special Publication 800-38A, 2001 Edition, US Department of Commerce/N.I.S.T., December 2001. Available from http://csrc.nist.gov/.

[11] NIST Pub 800-38C, DRAFT Recommendation for Block Cipher Modes of Operation – The CCM Mode for Authentication and Confidentiality, NIST Special Publication 800-38C, Draft, US Department of Commerce/N.I.S.T., Springfield, Virginia, September 4, 2003. Available from http://csrc.nist.gov/.

[12] P. Rogaway, D. Wagner, A Critique of CCM, IACR ePrint Archive 2003-070, April 13, 2003.