IEEE P802.15
Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)
Title	**Draft D17 Annex B Security Recommendation for Low-Rate IEEE 802.15.4 WPAN**
Date Submitted	[14 November, 2002]
Source	[Rene Struik] [Certicom Corporation] [5520 Explorer Drive, 4th floor, Mississauga, ON Canada L4W 5L1] Voice: [905-501-6083] Fax: [905-507-4230] E-mail: [rstruik@certicom.com]
Re:	[]
Abstract	[This document gives some security and security architectural recommendations to assist in improving the security and flexibility of the Draft D17 for the IEEE 802.15.4 Low-Rate WPAN]
Purpose	[Assist sponsor ballot comment resolution for the Draft D17 for the IEEE 802.15.4 WPAN.]
Notice	This document has been prepared to assist the IEEE P802.15. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15.

Editorial note: Annex B provisional replacement text, to take away the current security vulnerabilities of Draft D17 (for details, see also 02/474r1).

Annex A Security implementation

A.1 Generic CCM* mode

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

Editorial note: The CCM* mode coincides with the original CCM mode specification {xref CCM specification} for messages that require both encryption and authentication, but also offers support for messages that require only encryption. As with the CCM mode, the CCM* mode requires only one key. The security proof for the original CCM mode carries over to the CCM* mode described here.

For the generic CCM* mode there are two parameter choices to be made. The first choice is M, the size of the authentication field, in octets. The choice of the value for M involves a trade-off between message expansion and the probability that an attacker can undetectably modify a message. Valid values for M are 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.) The second choice is L, the size of the length field, in octets. This value requires a trade-off between the maximum message size and the size of the Nonce. Different applications require different trade-offs, so L is a parameter. Valid values for L are 2 to 8 (the value L=1 is reserved). The parameters L and M are encoded as shown in Table {xref} B.1.

Table B.1—Parameters of CCM* mode

Name	Description	Field Size	Encoding of field
M	Number of octets in authentication field	3 bits	0 if M=0; (M-2)/2 otherwise
L	Number of octets in length field	3 bits	L-1

Throughout this specification, the representation of integers as bit strings or octet strings shall be fixed. All integers shall be represented as octet strings in most-significant-octet first order.

A.1.1 Inputs

To encrypt a message using the CCM* mode, the sender must provide the following information:

— An encryption key K suitable for the block cipher.

— A nonce N of 15-L octets. Within the scope of any encryption key K, the nonce value shall be unique. That is, the set of nonce values used with any given key shall not contain any duplicate values. Using the same nonce for two different messages encrypted with the same key destroys the security properties of this mode.

— The message m, consisting of a string of $l(m)$ octets where $0 \le l(m) < 2^{8L}$. The length restriction ensures that $l(m)$ can be encoded in a field of L octets.

— Additional authenticated data a, consisting of a string of $l(a)$ octets where $0 \le l(a) < 2^{64}$. This additional data is authenticated but not encrypted, and is not included in the output of this mode. It can be used to authenticate plaintext headers, or contextual information that affects the interpretation of the message. Users who do not wish to authenticate additional data can provide a string of length zero.

Table B.2—Inputs for CCM*

Name	Description	Field Size	Encoding of field
K	Block cipher key	Depends on block cipher	String of octets.
N	Nonce	14-*L* octets	Not specified
m	Message to be encrypted and sent	*l(m)* octets	String of octets.
a	Additional authenticated data	*l(a)* octets	String of octets.

A.1.2 Authentication

The first step is to compute the authentication field *T*. This is done using the CBC-MAC mechanism, truncating the output to the appropriate size. We first define a sequence of blocks B_0, B_1, ... B_n and then apply CBC-MAC to these blocks.

The first block B_0 is formatted as follows.

Table B.3—First authentication block B_0

Octet no:	0	1 ... 14-*L*	15-*L*	16-*L* ... 15
Contents:	Flags	Nonce *N*	SecField	*l(m)*

(DELETE: The value *l(m)* is encoded in most-significant-octet first order).

The Flags field is formatted as

Table B.4—Authentication flags octet

Bit no:	0	1	2	3	4	5	6	7
Contents:		*L*			*M*		Adata	Reserved

The Reserved bit is reserved for future expansions and shall be set to zero. The Adata bit is set to zero if *l(a)*=0, and set to one if *l(a)*>0. The *L* field and the *M* field are the 3-bit representations of the integer encodings of the CCM mode parameters *L* and *M*, as specified in Table {xref} B.1.

The octet SecField in Table {xref} B.3 shall encode the potential values of *M* such, that this parameter can be uniquely recovered. The exact format of the octet SecField is outside the scope of this specification and shall be determined and fixed by the actual implementation environment of the CCM* mode.

Note 1: The exact format of the field SecField 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 *K* to a strict subset of those allowed in the generic CCM* mode. If so, the format of the SecField shall be such that each of the potential values of *M* in that particular subset can be uniquely recovered

Note 2: The 1-octet field SecField takes one octet away from the nonce, which might be undesirable. An alternative would be to incorporate the SecField in the *L*-octet wide length field, which effectively reduces this to an *L*-1 octet field for length encoding purposes. The essential cryptographic requirement is as follows: The first authentication block (Table B.3) and the encryption blocks (Table B.6) shall encode the potential values of *M* such, that this parameter can be uniquely recovered. This allows other options for encoding than the one presented in this specification (e.g., in the current 802.15.4 specification, the length of a message to be encrypted is at most 127 bytes, so *l(m)* could be represented using only 7 bits rather than the 2 octets currently reserved for this. Also, the number of application of a 128-bit block cipher would be at most 8, so the number of different counter values per frame (see Table B.6 below) can be represented using only 3 bits, rather than the 2 octets currently reserved for this).

If $l(a)>0$ (as indicated by the Adata field), then one or more blocks of authentication data are added. These blocks contain *l(a)* and *a* encoded in a reversible manner. We first construct a string that encodes *l(a)*.

If $0 < l(a) < 2^{16}-2^8$ then the length field is encoded as two octets which contain the value *l(a)* (DELETE: in most-significant-octet first order).

If $2^{16}-2^8 \leq l(a) < 2^{32}$ then the length field is encoded as six octets consisting of the octets 0xff, 0xfe, and four octets encoding *l(a)* (DELETE: in most-significant-octet-first order).

If $2^{32} \leq l(a) < 2^{64}$ then the length field is encoded as ten octets consisting of the octets 0xff, 0xff, and eight octets encoding *l(a)* (DELETE: in most-significant-octet-first order).

This is summarized in the following table. (DELETE: Note that all fields are interpreted in most-significant-octet first order).

Table B.5—Length encoding for additional authentication data

First two octets	Followed by	Comment
0x0000		Reserved
0x0001 ... 0xFEFF		For $0 < l(a) < 2^{16} - 2^8$
0xFF00 ... 0xFFFD		Reserved
0xFFFE	four octets *l(a)*	For $2^{16} - 2^8 \leq l(a) < 2^{32}$
0xFFFF	eight octets *l(a)*	For $2^{32} \leq l(a) < 2^{64}$

The blocks encoding *a* are formed by right-concatenating the octet string that encodes *l(a)* with *a* itself, and splitting the result into 16-octet blocks, right-padding the last block with zeroes if necessary. These blocks are right-appended to the first block B_0.

After the (optional) additional authentication blocks have been added to the first block *B*, we right-concatenate the message blocks hereto. The message blocks are formed by splitting the message *m* into 16-octet blocks, right-padding the last block with zeroes if necessary. Note that if the message *m* is the empty string, no blocks are added in this step.

The result of these two steps is the sequence of 16-octet blocks $B_0, B_1, ..., B_n$.

The CBC-MAC is now computed as follows:

$$X_1 := E(K, B_0);$$

$$X_{i+1} := E(K, X_i \oplus B_i) \quad \text{for } i=1, ... , n,$$

where E() is the block cipher encryption function and. The 16-octet string X_{n+1} is the resulting CBC-MAC value.

The authentication tag *T* is the result of truncating X_{n+1} to the leftmost *M* octets hereof.

A.1.3 Encryption

To encrypt the message data we use the CTR mode. We first define the key stream blocks by

$$S_i := E(K, A_i) \text{ for } i=0, 1, 2, ... , \text{ where}$$

the encryption blocks A_i are formatted as shown in Table {xref} B.6.

Table B.6—Encryption blocks A_i

Octet no:	0	1 ... 14-*L*	15-*L*	16-*L* ... 15
Contents:	Flags	Nonce *N*	SecField	Counter *i*

(DELETE: where *i* is encoded in most-significant-octet first order).

The octet SecField in Table {xref} B.6 shall be formatted in the same way as the corresponding field in Table {xref} B.3.

Table B.7—Encryption flags octet

Bit no:	0	1	2	3	4	5	6	7
Contents:		*L*			0		Reserved	Reserved

The Reserved bits are reserved for future expansions and shall be set to zero. Bit 6 corresponds to the Adata bit in the B_0 block, but as this bit is not used here, it is reserved. Bits 3, 4, and 5 are set to 0. This ensures that all the *A* blocks are distinct from B_0, which has the non-zero encoding of *M* in this position. Bits 0, 1, and 2 contain *L*, using the same encoding as in B_0.

The encrypted message is the result of XORing the octets of the message *m*, in order, with the leftmost *l(m)* octets of the right-concatenation of S_1, S_2, S_3, ... Note that S_0 is not used to encrypt the message; it is used for encrypting the authentication field *T* instead.

The encrypted authentication value *U* is the result of XORing the octets of the authentication field *T* determined in Subclause {xref} B.1.2 with the leftmost *M* octets of the key stream block S_0, i.e.,

$$U := T \oplus \text{leftmost-}M\text{-octets}(S_0)$$

A.1.4 Output

The encrypted and authenticated message *c* that is the output of the generic CCM* mode computation is the right-concatenation of the encrypted message and the encrypted authentication value *U*.

A.1.5 Decryption

To decrypt a message the following information is required:

- The encryption key *K*.
- The nonce *N*.
- The additional authenticated data *a*.
- The encrypted and authenticated message *c*.

Decryption starts by recomputing the key stream to recover the message *m* and the authentication field *T*. The message and additional authentication data is then used to recompute the CBC-MAC value and check *T*.

If the *T* value is not correct, the receiver shall not reveal any information except for the fact that *T* is incorrect. In particular, the receiver shall not reveal the decrypted message, the value *T*, or any other information.

A.1.6 Restrictions

All implementations shall limit the total amount of data that is encrypted with a single key. The sender shall ensure that the total number of block cipher encryption operations in the CBC-MAC and encryption together shall not exceed 2^{61}. (This allows close to 2^{64} octets to be encrypted and authenticated using CCM*, which should be more than enough for most applications.) Receivers that do not expect to decrypt the same message twice may also implement this limit.

The recipient shall verify the (truncated) CBC-MAC before releasing any information such as the plaintext. If the CBC-MAC verification fails, the receiver shall destroy all information, except for the fact that the CBC-MAC verification failed

A.1.7 List of symbols

Table 136 provides a list of the symbols used for the above specification of CCM*.

Table B.8—List of symbols

Name	Description	Size	Comment
a	Additional authenticated data	*l(a)* octets	Use empty string if not desired.
A_i	Counter block to generate key stream	16 octets	Contains block counter, nonce, and flags.
B_i	Input block for CBC-MAC	16 octets	Together encode *N*, *L*, *M*, *m*, and *a* uniquely.
c	Ciphertext	*l(m)* + *M* octets	Includes the encrypted MAC
K	Block cipher key	N/A.	At least 128 bits, preferably 256 bits.
L	Number of octets in length field	3 bits	Values 1 ... 8, encoded in 3 bits as *L*-1.
m	Message to be encrypted and sent	*l(m)* octets	Subject to $0 \le l(m) < 2^{8L}$
M	Number of octets in authentication field	3 bits	Values 4, 6, 8, ..., 16. Encoded value is (*M*-2)/2
N	Nonce	14-*L* octets	Nonce shall never be repeated for same key.
S_i	Block of the encryption key stream	16 octets	Use S_0, S_1, S_2, ... to encrypt *m* and *T*.
T	Unencrypted authentication tag	*M* octets	
U	Encrypted authentication tag	*M* octets	Appended to the message after encryption
X_i	Intermediate value of CBC-MAC	16 octets	

~~A.2 CTR Encryption (REMOVE COMPLETELY)~~

~~The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message: across all of the messages that are encrypted under the given key, all of the counters must be distinct. In this standard, the counters for a given message are denoted $T_1, T_2, \ldots, T_n$. Given a sequence of counters, $T_1, T_2, \ldots, T_n$, the CTR mode is defined as follows:~~

~~CTR Encryption:~~

~~$O_j = CIPH_K(T_j)$ for $j = 1, 2 \ldots n$;~~
~~$C_j = P_j \oplus O_j$ for $j = 1, 2 \ldots n-1$;~~
~~$C^*_n = P^*_n \oplus MSB_u(O_n)$.~~

CTR Decryption: $O_j = CIPH_K(T_j)$ for $j = 1, 2 \ldots n$;
$P_j = C_j \oplus O_j$ for $j = 1, 2 \ldots n\text{-}1$;
$P^*_n = C^*_n \oplus MSB_u(O_n)$.

In CTR encryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-ORed with the corresponding plaintext blocks to produce the ciphertext blocks. For the last block, which may be a partial block of *u* bits, the most significant *u* bits of the last output block are used for the exclusive-OR operation; the remaining b-*u* bits of the last output block are discarded, where b is the length in bits of the block cipher.

In CTR decryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-ORed with the corresponding ciphertext blocks to recover the plaintext blocks. For the last block, which may be a partial block of *u* bits, the most significant *u* bits of the last output block are used for the exclusive-OR operation; the remaining b-*u* bits of the last output block are discarded.

In both CTR encryption and CTR decryption, the forward cipher functions can be performed in parallel; similarly, the plaintext block that corresponds to any particular ciphertext block can be recovered independently from the other plaintext blocks if the corresponding counter block can be determined. Moreover, the forward cipher functions can be applied to the counters prior to the availability of the plaintext or ciphertext data.

Figure B.1—The CTR Mode

The CTR mode is illustrated in Figure 1.

~~A.3 CBC-MAC (REMOVE COMPLETELY)~~

~~The CBC-MAC algorithm makes use of an underlying block cipher to provide data integrity on input data. The block cipher transforms (or encrypts) input vectors of the block size to output vectors of the block size using a cryptographic key. Let D be any input vector and assume a key has been selected. The vector of length equal to the block size, O, which is the output of the block cipher when applied to D, using the enciphering operation, is represented as follows.~~

$$O = e(D)$$

~~The data (e.g., record, file, message, or program) to be authenticated is grouped into contiguous blocks: D_1, D_2,.... D_n each with length equal to the block size. If the number of data bits is not a multiple of the block size, then the final input block will be a partial block of data, left justified. with zeroes appended to form a full block. The calculation of the MIC (message integrity code) is given by the following equations where ⊕ represents the Exclusive-OR of two vectors.~~

$$O_1 = e(D_1)$$

$$O_2 = e(D_2 \oplus O_1)$$

$$O_3 = e(D_3 \oplus O_2)$$

$$\cdots$$

$$O_n = e(D_n \oplus O_{n-1})$$

~~The MIC is selected from O_n. Devices that implement CBC-MAC shall be capable of selecting the leftmost M bits of O_n as the MIC, where $32 < M < 128$ and M is a multiple of 8. A block diagram of the MIC gener-~~

~~ation is given in Figure 2.~~

~~Figure B.2 – CBC-MAC~~