8. Security

8.1 Overview

The MAC sublayer is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. This standard supports the following security services:

— Data confidentiality
— Data authenticity
— Replay protection

8.2 Functional description

A device may optionally implement security. A device that does not implement security shall not provide a mechanism for the MAC sublayer to perform any cryptographic transformation on incoming and outgoing frames nor require any PIB attributes associated with security. A device that implements security shall provide a mechanism for the MAC sublayer to provide cryptographic transformations on incoming and outgoing frames using information in the PIB attributes associated with security only if the *macSecurityEnabled* attribute is set to TRUE.

If the MAC sublayer is required to transmit a frame or receives an incoming frame, the MAC sublayer shall process the frame as specified in 8.2.1 and 8.2.3, respectively.

8.2.1 Outgoing frame security procedure

The inputs to this procedure are the frame to be secured and the SecurityLevel, KeyIdMode, KeySource, and KeyIndex parameters from the originating primitive or automatic request PIB attributes. The outputs from this procedure are the status of the procedure and, if this status is SUCCESS, the secured frame.

The outgoing frame security procedure involves the following steps:

j) If the *macSecurityEnabled* attribute is set to FALSE and the SecurityLevel parameter is not equal to zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.
k) The procedure shall determine whether the frame to be secured satisfies the constraint on the maximum length of MAC frames, as follows:
    1) The procedure shall determine the length AuthLen, in octets, of the Authentication field, AuthLen, from the SecurityLevel parameter and Table 145.
    2) The procedure shall determine the length AuxLen, in octets, of the auxiliary security header, as described in 8.4, using KeyIdMode and the SecurityLevel parameter.
    3) The procedure shall determine the data expansion as AuxLen + AuthLen.
    4) The procedure shall check whether the length of the frame to be secured, including data expansion and FCS, is less than or equal to *aMaxPHYPacketSize*. If this check fails, the procedure shall return with a status of FRAME_TOO_LONG.
l) If the SecurityLevel parameter is zero, the procedure shall set the secured frame to be the frame to be secured and return with a status of SUCCESS.
m) The procedure shall set the frame counter to the *macFrameCounter* attribute. If the frame counter has the value 0xffffffff, the procedure shall return with a status of COUNTER_ERROR.
n) The procedure shall obtain the KeyDescriptor using the KeyDescriptor lookup procedure as described in 8.2.2 with the device addressing mode set to DstAddrMode, the device PAN ID set to

DstPANId, and the device address set to DstAddr. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.

o) The procedure shall insert the auxiliary security header into the frame, with fields set as follows:
   1) The Security Level field of the Security Control field shall be set to the SecurityLevel parameter.
   2) The Key Identifier Mode field of the Security Control field shall be set to the KeyIdMode parameter.
   3) The Frame Counter field shall be set to the frame counter.
   4) If the KeyIdMode parameter is set to a value not equal to zero, the Key Source and Key Index fields of the Key Identifier field shall be set to the KeySource and KeyIndex parameters, respectively.

p) The Private Payload field and Open Payload field shall be set as indicated in Table 140. The procedure shall then use the Private Payload field, the Open Payload field, the *macExtendedAddress*, the frame counter, the SecurityLevel parameter, and the Key element of the KeyDescriptor to produce the secured frame according to the CCM* transformation process defined in 8.3.4.

Table 140—Private Payload field and Open Payload field definitions

Frame type	Private Payload field	Open Payload field
Beacon	Beacon Payload field	All other fields in the MAC Payload field
Data	Data Payload field	None
MAC command	Command Payload	All other fields in the MAC Payload field

q) The procedure shall increment the frame counter by one and set the *macFrameCounter* attribute to the resulting value.

r) The procedure shall return with a status of SUCCESS.

8.2.2 KeyDescriptor lookup procedure

The inputs to this procedure are the KeyIdMode, KeySource, KeyIndex, device addressing mode, device PAN ID, and device address. The outputs from this procedure are a status and, if passed, a KeyDescriptor.

The procedure involves the following steps:

a) If the KeyIdMode parameter is set to 0x00, then for each KeyDescriptor in the *macKeyTable* attribute and for each KeyIdLookupDescriptor in the KeyIdLookupList of the KeyDescriptor:
   1) If the device addressing mode is set to NO_ADDRESS, then the device PAN ID shall be set to *macPANId*. Otherwise, the device PAN ID shall be the value passed to the procedure.
   2) If the device addressing mode is set to NO_ADDRESS and the frame type is beacon, then the device address shall be *macCoordExtendedAddress*.
   3) If the device addressing mode is set to NO_ADDRESS and the frame type is not beacon, then:
      i) If the *macCoordShortAddress* attribute is set to 0xfffe, then the device address shall be set to the *macCoordExtendedAddress*.
      ii) If the *macCoordShortAddress* attribute is set to a value of 0x0000–0xfffd, then the device address shall be set to the *macCoordShortAddress*.

iii) If the *macCoordShortAddress* attribute is set to 0xffff, the procedure shall return with a failed status.

4) If the device addressing mode is set to SHORT_ADDRESS or EXTENDED_ADDRESS, then the device address shall be the value passed to the procedure.

5) If the device addressing mode, device PAN ID, and device address match the DeviceAddrMode, DevicePANId, and DeviceAddress of a KeyIdLookupDescriptor, then the procedure returns with the corresponding KeyDescriptor and passed status.

b) If the KeyID mode parameter is set to 0x01, then for each KeyDescriptor in the *macKeyTable* attribute and for each KeyIdLookupDescriptor in the KeyIdLookupList of the KeyDescriptor, if the KeyIndex matches the KeyIndex of a KeyIdLookupDescriptor and the KeySource matches *macDefaultKeySource*, then the procedure returns with the corresponding KeyDescriptor and passed status.

c) If the KeyID mode parameter is set to 0x02 or 0x03, then for each KeyDescriptor in the *macKeyTable* attribute and for each KeyIdLookupDescriptor in the KeyIdLookupList of the KeyDescriptor, if the KeySource and KeyIndex match the KeySource and KeyIndex of a KeyIdLookupDescriptor, then the procedure returns with the KeyDescriptor and passed status.

d) The procedure shall return with a failed status.

NOTE—For broadcast frames, the KeyDescriptor lookup procedure will result in a failed status if implicit key identification is used. Hence, explicit key identification should be used for broadcast frames.[9]

8.2.3 Incoming frame security procedure

The input to this procedure is the frame to be unsecured. The outputs from this procedure are the status of the procedure and, if this status is SUCCESS, the unsecured frame, the security level, the key identifier mode, the key source, and the key index.

All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure.

The incoming frame security procedure involves the following steps:

a) If the Security Enabled field of the frame to be unsecured is set to zero, the procedure shall set the security level to zero.

b) If the Security Enabled field of the frame to be unsecured is set to one and the Frame Version field of the frame to be unsecured is set to zero, the procedure shall return with a status of UNSUPPORTED_LEGACY.

c) If the Security Enabled field of the frame to be unsecured is set to one, the procedure shall set the security level and the key identifier mode to the corresponding fields of the Security Control field of the auxiliary security header of the frame to be unsecured, and the key source and key index to the corresponding fields of the Key Identifier field of the auxiliary security header of the frame to be unsecured, if present. If the resulting security level is zero, the procedure shall return with a status of UNSUPPORTED_SECURITY.

d) If the *macSecurityEnabled* attribute is set to FALSE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with a status of SUCCESS if the security level is equal to zero and with a status of UNSUPPORTED_SECURITY otherwise.

e) The device PAN ID shall be set to the Source PAN Identifier field, if it is present. If the PAN ID compression field is set to one, then the device PAN ID shall be set to the Destination PAN Identifier field. The device addressing mode shall be set according to the Source Addressing Mode field, as defined in Table 141. The device address shall be set to the Source Address, if present. The KeyIdMode shall be set to the Key Identifier Mode field, the KeyIndex shall be set to the Key Index field, if present, and the KeySource shall be set to the Key Source field, if present.

[9]Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

Table 141—Mapping of Source Addressing Mode field to device addressing mode

Source Addressing Mode field	Device addressing mode
0x00	NO_ADDRESS
0x02	SHORT_ADDRESS
0x03	EXTENDED_ADDRESS

f) The procedure shall obtain the KeyDescriptor using the KeyDescriptor lookup procedure as described in 8.2.2 with using the KeyIdMode, KeyIndex, KeySource, device addressing mode, device PANID, and device address. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.

g) The procedure shall obtain the DeviceDescriptor using the DeviceDescriptor lookup procedure described in 8.2.4. If that procedure fails, then the procedure shall return with a status of UNAVAILABLE_DEVICE.

h) The procedure shall obtain the SecurityLevelDescriptor by passing the frame type and, if the frame is a MAC command, the Command Identifier, to the SecurityLevelDescriptor lookup procedure described in 8.2.5. If that procedure fails, the procedure shall return with a status of UNAVAILABLE_SECURITY_LEVEL.

i) The procedure shall determine whether the frame to be unsecured conforms to the security level policy by passing the SecurityLevelDescriptor and the security level to the incoming security level checking procedure, as described in 8.2.6. If that procedure returns with a failed status, the procedure shall return with a status of IMPROPER_SECURITY_LEVEL; otherwise, if that procedure returns with a passed status and the security level is equal to zero, the procedure shall set the unsecured frame to be the frame to be unsecured and return with a status of SUCCESS.

j) If the incoming security level checking procedure of step i) had as output the ‘conditionally passed’ status and the Exempt element of the DeviceDescriptor is set to TRUE, the procedure shall set the unsecured frame to be the frame to be unsecured and return with a status of SUCCESS.

k) If the incoming security level checking procedure of step i) had as output the ‘conditionally passed’ status and the Exempt element of the DeviceDescriptor is set to FALSE, the procedure shall return with a status of IMPROPER_SECURITY_LEVEL.

l) The procedure shall set the frame counter to the Frame Counter field of the frame to be unsecured. If the frame counter has the value 0xffffffff, the procedure shall return with a status of COUNTER_ERROR.

m) If the frame counter is less than the FrameCounter element of the DeviceDescriptor, the procedure shall return with a status of COUNTER_ERROR.

n) The procedure shall determine whether the frame to be unsecured conforms to the key usage policy by passing the KeyDescriptor, the frame type, and, if the frame is a MAC command, the Command Identifier field, to the incoming key usage policy checking procedure, as described in 8.2.7. If that procedure fails, the procedure shall return with a status of IMPROPER_KEY_TYPE.

o) The Private Payload field and Open Payload field shall be set as indicated in Table 140. The procedure shall then use the Private Payload field, the Open Payload field, the ExtAddress element of the DeviceDescriptor, the frame counter, the security level, and the Key element of the KeyDescriptor to produce the unsecured frame, according to the CCM* inverse transformation process described in the security operations, as described in 8.3.5.

p) If the CCM* inverse transformation process fails, the procedure shall return with a status of SECURITY_ERROR.

q) The procedure shall increment the frame counter by one and set the FrameCounter element of the DeviceDescriptor to the resulting value.

r) The procedure shall return with a status of SUCCESS.

8.2.4 DeviceDescriptor lookup procedure

The inputs to this procedure are the device addressing mode, the device PAN ID, and the device address. The output from this procedure is a passed or failed status and, if passed, a DeviceDescriptor.

The DeviceDescriptor lookup procedure involves the following steps:

a) If the device addressing mode is set to NO_ADDRESS, then the device PAN ID shall be set to *macPANId*. Otherwise, the device PAN ID shall be the value passed to the procedure.
b) If the device addressing mode is set to NO_ADDRESS, then:
    1) If the *macCoordShortAddress* attribute is set to 0xfffe, then the device address shall be set to the *macCoordExtendedAddress*.
    2) If the *macCoordShortAddress* attribute is set to a value of 0x0000–0xfffd, then the device address shall be set to the *macCoordShortAddress*.
    3) If the *macCoordShortAddress* attribute is set to 0xffff, the procedure shall return with a failed status.
c) If the device addressing mode is set to SHORT_ADDRESS or EXTENDED_ADDRESS, then the device address shall be the value passed to the procedure.
d) For each DeviceDescriptor in DeviceDescriptorHandleList, if the device PAN ID matches the PANId and the and device address matches the ShortAddress, if the device addressing mode is set to SHORT_ADDRESS, or the ExtAddress, if the device addressing mode is set to EXTENDED_ADDRESS, then the procedure shall return with the corresponding DevicesDescriptor and a passed status.
e) The procedure shall return with a failed status.

8.2.5 SecurityLevelDescriptor lookup procedure

The inputs to this procedure are the frame type and, if the frame is a MAC command, the Command Identifier field. The output from this procedure are a passed or failed status and, if passed, a SecurityLevelDescriptor.

The SecurityLevelDescriptor lookup procedure involves the following steps:

a) For each SecurityLevelDescriptor in the *macSecurityLevelTable* attribute:
    1) If the frame type indicates that the frame is not a MAC command and the frame type is equal to the FrameType element of the SecurityLevelDescriptor (i.e., there is a match), the procedure shall return with the SecurityLevelDescriptor and a passed status.
    2) If the frame type indicates that the frame is a MAC command and the frame type is equal to the FrameType element of the SecurityLevelDescriptor and the Command Identifier field is equal to the CommandIdentifier element of the SecurityLevelDescriptor, the procedure shall return with the SecurityLevelDescriptor and a passed status.
b) The procedure shall return with a failed status.

8.2.6 Incoming security level checking procedure

The inputs to this procedure are the SecurityLevelDescriptor and the incoming security level. The output from this procedure is a passed, failed, or ‘conditionally passed’ status.

The incoming security level checking procedure involves the following steps:

a) If the AllowedSecurityLevels is empty, then the procedure shall compare the incoming security level (as SEC1) with the SecurityMinimum element of the SecurityLevelDescriptor (as SEC2) according to the algorithm described in 8.4.1.1. If this comparison evaluates to TRUE, the procedure shall return with a passed status.

b) If the AllowedSecurityLevels is not empty, the procedure shall check whether the incoming security level is equal to any of the elements of the AllowedSecurityLevels. If this check is successful (i.e., there is a match), the procedure shall return with a passed status.

c) If the incoming security level is equal to 0x00 and the DeviceOverrideSecurityMinimum element of the SecurityLevelDescriptor is set to TRUE, the procedure shall return with a 'conditionally passed' status.

d) The procedure shall return with a failed status.

8.2.7 Incoming key usage policy checking procedure

The inputs to this procedure are the KeyDescriptor, the Frame Type field, and the Command Identifier field. The output from this procedure is a passed or failed status.

The incoming key usage policy checking procedure involves the following steps:

a) For each KeyUsageDescriptor in the KeyUsageList of the KeyDescriptor:
   1) If the frame type indicates that the frame is not a MAC command and the frame type is equal to the FrameType element of the KeyUsageDescriptor, the procedure shall return with a passed status.
   2) If the frame type indicates that the frame is a MAC command, the frame type is equal to the FrameType element of the KeyUsageDescriptor, and the Command Identifier field is equal to the CommandIdentifier element of the KeyUsageDescriptor, the procedure shall return with a passed status.

b) The procedure shall return with a failed status.

8.3 Security operations

This subclause describes the parameters for the CCM* security operations, as specified in B.3.2.

8.3.1 Integer and octet representation

The integer and octet representation conventions specified in B.2 are used throughout this subclause.

8.3.2 CCM* Nonce

The CCM* nonce is a 13-octet string and is used for the advanced encryption standard (AES)-CCM* mode of operation, as described in B.3.2. The nonce shall be formatted as shown in Figure 216, with the leftmost field in the figure defining the first (and leftmost) octets and the rightmost field defining the last (and rightmost) octet of the nonce.

Figure 216—CCM* nonce

Octets: 8	**4**	**1**
Source address	Frame counter	Security level

The source address shall be set to the extended address *macExtendedAddress* of the device originating the frame, the frame counter to the value of the respective field in the auxiliary security header, as defined in 8.4, and the security level to the value corresponding to the Security Level field, as defined in Table 145.

The source address, frame counter, and security level shall be represented as specified in 8.3.1.

When the *macFrameCounterMode* = 0x05, or the incoming frame counter size is 5 octets as described in 8.4.1.4, (such as when operating in TSCH mode, including when in active or passive scan to join a TSCH PAN), the nonce shall be formatted as shown in Figure 217.

Figure 217—CCM* nonce in TCSH mode

Octets: 8	5
Source address	Frame counter

The Source Address shall be set to match the source address of the device originating the frame, i.e., the extended address *aExtendedAddress* if an 8-octet source is used, or the *macShortAddress* if a 2-octet source address is used. If no source address is used in the frame, then the *macShortAddress* of the device originating the frame is used.

In TSCH mode, the Frame Counter shall be set to the global frame counter value, i.e., the Absolute Slot Number as described in 5.2.7.2, regardless of whether the frame counter is carried in the auxiliary security header.

8.3.3 CCM* prerequisites

Securing a frame involves the use of the CCM* mode encryption and authentication transformation, as described in B.4.1. Unsecuring a frame involves the use of the CCM* decryption and authentication checking transformation, as described in B.4.2.

The length *M* of the Authentication field for the CCM* forward transformation and the CCM* inverse transformation is determined from Table 145, using the Security Level field of the Security Control field of the auxiliary security header of the frame.

8.3.4 CCM* transformation data representation

This subclause describes how the inputs and outputs of the CCM* forward transformation, as described in B.4.1, are formed.

The inputs are:

— Key

— Nonce

— *a* data

— *m* data

The output is *c* data.

8.3.4.1 Key and nonce data inputs

The Key data for the CCM* forward transformation is passed by the outgoing frame security procedure described in 8.2.1. The Nonce data for the CCM* transformation is constructed as described in 8.3.2.

8.3.4.2 *a* data and *m* data

In the CCM* transformation process, the data fields shall be applied as in Table 142.

Table 142—*a* data and *m* data for all security levels

Security level	*a* data	*m* data
0	None	None
1	MHR || Open Payload field || Unsecured Private Payload field	None
2	MHR || Open Payload field || Unsecured Private Payload field	None
3	MHR || Open Payload field || Unsecured Private Payload field	None
4	None	Unsecured Private Payload field
5	MHR || Open Payload field	Unsecured Private Payload field
6	MHR || Open Payload field	Unsecured Private Payload field
7	MHR || Open Payload field	Unsecured Private Payload field

NOTE—The MHR contains the Auxiliary Security Header field, as defined in 6.2.

8.3.4.3 *c* data output

In the CCM* transformation process, the data fields that are applied, or right-concatenated and applied, represent octet strings.

The Private Payload field of the original unsecured frame shall be replaced by the right-concatenation of that field and the *c* field if data confidentiality is not provided and shall be replaced by the *c* field otherwise. The contents of the *c* data for each of the security levels is defined in Table 143.

Table 143—*c* data for all security levels

Security level	*c* data
0	None
1	MIC-32
2	MIC-64
3	MIC-128

Table 143—*c* data for all security levels *(continued)*

Security level	*c* data
4	Encrypted Private Payload field
5	Encrypted Private Payload field || MIC-32
6	Encrypted Private Payload field || MIC-64
7	Encrypted Private Payload field || MIC-128

8.3.5 CCM* inverse transformation data representation

This subclause describes how the inputs and outputs of the CCM* inverse transformation, as described in B.4.2, are formed.

The inputs are:

— Key
— Nonce
— *c* data
— *a* data

The output is *m* data.

8.3.5.1 Key and nonce data inputs

The Key data for the CCM* inverse transformation is passed by the incoming frame security procedure described in 8.2.3. The Nonce data for the CCM* transformation is constructed as described in 8.3.2.

8.3.5.2 *c* data and *a* data

In the CCM* inverse transformation process, the data fields shall be applied as in Table 144.

Table 144—*c* data and *a* data for all security levels

Security level	*c* data	*a* data
0	None	None
1	MIC-32	MHR || Open Payload field || Private Payload field
2	MIC-64	MHR || Open Payload field || Private Payload field
3	MIC-128	MHR || Open Payload field || Private Payload field
4	Encrypted Private Payload field	MHR || Open Payload field
5	Encrypted Private Payload field || MIC-32	MHR || Open Payload field
6	Encrypted Private Payload field || MIC-64	MHR || Open Payload field
7	Encrypted Private Payload field || MIC-128	MHR || Open Payload field

NOTE—The MHR contains the Auxiliary Security Header field, as defined in 6.2.

8.3.5.3 *m* data output

The Private Payload field of the MAC Payload shall be set to the *m* data if frame security includes providing confidentiality and shall be set to the Private Payload field of the MAC Payload, with the rightmost substring, *c*, deleted, otherwise.

8.4 Auxiliary security header

The Auxiliary Security Header field has a variable length and contains information required for security processing, including a Security Control field, a Frame Counter field, and a Key Identifier field. The Auxiliary Security Header field shall be present only if the Security Enabled field is set to one. The Auxiliary Security Header field shall be formatted as illustrated in Figure 218.

Octets: 1	**0/4/5**	**0/1/5/9**
Security Control	Frame Counter	Key Identifier

Figure 218—Format of the auxiliary security header

The auxiliary security header uses the representation conventions specified in 6.1.

8.4.1 Security Control field

The Security Control field is used to provide information about what protection is applied to the frame. The Security Control field shall be formatted as shown in Figure 219.

Bit: 0–2	**3–4**	**5**	**6**	**7**
Security Level	Key Identifier Mode	Frame Counter Suppression	Frame Counter Size	Reserved

Figure 219—Security Control field format

8.4.1.1 Security Level field

The Security Level field indicates the actual frame protection that is provided. This value can be adapted on a frame-by-frame basis and allows for varying levels of data authenticity (to allow minimization of security overhead in transmitted frames where required) and for optional data confidentiality. The cryptographic protection offered by the various security levels is shown in Table 145. When nontrivial protection is required, replay protection is always provided.

Security levels can be ordered according to the corresponding cryptographic protection offered. Here, a first security level SEC1 is greater than or equal to a second security level SEC2 if and only if SEC1 offers at least the protection offered by SEC2, both with respect to data confidentiality and with respect to data authenticity. The statement "SEC1 is greater than or equal to SEC2" shall be evaluated as TRUE if both of the following conditions apply:

Table 145—Security levels available to the MAC sublayer

Security level	Security level field $b_2\ b_1\ b_0$	Security attributes	Data confidentiality	Data authenticity	Encrypted authentication tag length, *M*, (octets)
0	000	None	OFF	NO	0
1	001	MIC-32	OFF	YES	4
2	010	MIC-64	OFF	YES	8
3	011	MIC-128	OFF	YES	16
4	100	ENC	ON	NO	0
5	101	ENC-MIC-32	ON	YES	4
6	110	ENC-MIC-64	ON	YES	8
7	111	ENC-MIC-128	ON	YES	16

a) Bit position b_2 in SEC1 is greater than or equal to bit position b_2 in SEC2 (where Encryption OFF < Encryption ON).

b) The integer value of bit positions $b_1\ b_0$ in SEC1 is greater than or equal to the integer value of bit positions $b_1\ b_0$ in SEC2 (where increasing integer values indicate increasing levels of data authenticity provided, i.e., message integrity code MIC-0 < MIC-32 < MIC-64 < MIC-128).

Otherwise, the statement shall be evaluated as FALSE.

For example, ENC-MIC-64 ≥ MIC-64 is TRUE because ENC-MIC-64 offers the same data authenticity protection as MIC-64, plus confidentiality. On the other hand, MIC-128 ≥ ENC-MIC-64 is FALSE because even though MIC-128 offers stronger data authenticity than ENC-MIC-64, it offers no confidentiality.

8.4.1.2 Key Identifier Mode field

The Key Identifier Mode field indicates whether the key that is used to protect the frame can be derived implicitly or explicitly; furthermore, it is used to indicate the particular representations of the Key Identifier field, as defined in 8.4.3, if derived explicitly. The Key Identifier Mode field shall be set to one of the values listed in Table 146. The Key Identifier field of the auxiliary security header, as defined in 8.4.3, shall be present only if this field has a value that is not equal to 0x00.

8.4.1.3 Frame Counter Suppression field

The Frame Counter Suppression field is set to zero when the frame counter is carried in the frame. When set to one, the frame counter is not carried in the frame, and the frame counter used to construct the nonce defined in 8.3.2 is either an incrementing shared global frame counter such as ASN, or in the case of an enhanced acknowledgment, the frame counter of the frame being acknowledged.

8.4.1.4 Frame Counter Size field

The Frame Counter Size field is set to zero when the frame counter is 4 octets, and the nonce is constructed as shown in Figure 216. When set to one, the frame counter is 5 octets, and the nonce is constructed as shown in Figure 217.

Table 146—Values of the key identifier mode

Key identifier mode	Key Identifier Mode field b_1 b_0	Description	Key Identifier field length (octets)
0x00	00	Key is determined implicitly from the originator and recipient(s) of the frame, as indicated in the frame header.	0
0x01	01	Key is determined from the Key Index field in conjunction with *macDefault-KeySource.*	1
0x02	10	Key is determined explicitly from the 4-octet Key Source field and the Key Index field.	5
0x03	11	Key is determined explicitly from the 8-octet Key Source field and the Key Index field.	9

8.4.2 Frame Counter field

The Frame Counter field represents the *macFrameCounter* attribute of the originator of a protected frame. It is used to provide semantic security of the cryptographic mechanism used to protect a frame and to offer replay protection.

The Frame Counter field may be included in each secured frame and is one of the elements required for the unsecuring operation at the recipient(s). The Frame Counter field is incremented each time an outgoing frame is secured, as described in the outgoing frame security procedure 7.2.1. When the Frame Counter field reaches its maximum value (0xffffffff for a 4 octet Frame Counter field or 0xffffffffff for a 5-octet Frame Counter field), the associated keying material shall no longer be used, thus requiring all keys associated with the device to be updated. This provides a mechanism for ensuring that the keying material for every frame is unique and, thereby, provides for sequential freshness.

8.4.3 Key Identifier field

The Key Identifier field has a variable length and identifies the key that is used for cryptographic protection of outgoing frames, either explicitly or in conjunction with implicitly defined side information. The Key Identifier field shall be present only if the Key Identifier Mode field, as defined in 8.4.1.2, is set to a value different from 0x00. The Key Identifier field shall be formatted as illustrated in Figure 220.

Figure 220—Format for the Key Identifier field, if present

Octets: 0/4/8	1
Key Source	Key Index

8.4.3.1 Key Source field

The KeySource field, when present, indicates the originator of a group key. If the Key Identifier Mode field indicates a 4 octet Key Source field, then the Key Source field shall be the *macPANId* of the originator of the group key right concatenated with the *macShortAddress* of the originator of the group key. If the Key

Identifier Mode field indicates an 8 octet Key Source field, then the Key Source field shall be set to the *macExtendedAddress* of the originator of the group key.

8.4.3.2 Key Index field

The Key Index field allows unique identification of different keys with the same originator.

It is the responsibility of each key originator to make sure that the actively used keys that it issues have distinct key indices and that the key indices are all different from 0x00.

8.5 Security-related MAC PIB attributes

The PIB security-related attributes are defined in Table 147. A MAC implementation may impose additional constraints on read/write operations for the security-related PIB attributes.

Table 147— Security-related MAC PIB attributes

Attribute	Type	Range	Description	Default
macKeyTable	Set of *Key Descriptors*, as defined Table 148	—	KeyDescriptor entries, each containing keys and security related information.	(empty)
macDeviceTable	Set of *Device Descriptors*, as defined in Table 151	—	DeviceDescriptors for each remote device with which this device securely communicates.	(empty)
macSecurity LevelTable	Set of *SecurityLevel-Descriptors*, as defined in Table 150	—	Provides information about the security level required for each MAC frame type and subtype.	(empty)
macFrameCounter	Integer	For *macFrameCounterMode* = 0x04 or *macFrameCounterMode* not implemented, 0x00000000–0xffffffff For *macFrameCounterMode* = 0x05, 0x0000000000–0xffffffffff	The outgoing frame counter for this device.	0x00000000
macAutoRequest SecurityLevel	Integer	As defined in Table 145	The security level used for automatic data requests.	0x06
macAutoRequest KeyIdMode	Integer	0x00–0x03	The key identifier mode used for automatic data requests. This attribute is invalid if the *macAutoRequestSecurityLevel* attribute is set to 0x00.	0x00

Table 147— Security-related MAC PIB attributes *(continued)*

Attribute	Type	Range	Description	Default
macAutoRequest KeySource	As specified by the *macAutoRequesKey-IdMode* parameter	—	The originator of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* element is invalid or set to 0x00.	All octets 0xff
macAutoRequest KeyIndex	Integer	0x01–0xff	The index of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* attribute is invalid or set to 0x00.	0xff
macDefaultKey Source	Set of 8 octets	—	The originator of the default key used for key identifier mode 0x01.	All octets 0xff
macPANCoord ExtendedAddress	IEEE address	An extended IEEE address	The extended address of the PAN coordinator.	—
macPANCoord ShortAddress	Integer	0x0000–0xffff	The short address assigned to the PAN coordinator. A value of 0xfffe indicates that the PAN coordinator is only using its extended address. A value of 0xffff indicates that this value is unknown.	0x0000
macFrameCounter-Mode	Integer	0x04–0x05	Size of the frame counter: 0x04 = 4 octets 0x05 = 5 octets	0x04

Table 148 defines the elements in a KeyDescriptor.

Table 148—Elements of KeyDescriptor

Name	Type	Range	Description
KeyIdLookup List	List of KeyIdLookupDescriptor entries, as defined in Table 152	—	A list of KeyIdLookupDescriptor entries used to identify this KeyDescriptor.
DeviceDescriptor HandleList	—	—	A list of implementation specific handles to DeviceDescriptor entries in *macDeviceTable* for each of the devices that are currently using this key.
KeyUsageList	List of KeyUsageDescriptor entries, as defined in Table 149	—	A list of KeyUsageDescriptor entries indicating the frame types with which this key may be used.
Key	Set of 16 octets	—	The value of the key.

Table 149 defines the elements of a KeyUsageDescriptor.

Table 150 defines the elements of a SecurityLevelDescriptor.

Table 149—Elements of KeyUsageDescriptor

Name	Type	Range	Description
FrameType	Integer	As defined in 6.2.1.1	As defined in 6.2.1.1.
CommandIdentifier	Integer	As defined in Table 40	As defined in Table 40.

Table 150—Elements of SecurityLevelDescriptor

Name	Type	Range	Description
FrameType	Integer	As defined in 6.2.1.1	As defined in 6.2.1.1.
CommandIdentifier	Integer	As defined in Table 40	As defined in Table 40.
SecurityMinimum	Integer	As defined in Table 145	The minimal required/expected security level, as defined in Table 145, for incoming MAC frames with the indicated frame type and, if present, Command Identifier.
DeviceOverrideSecurity Minimum	Boolean	TRUE, FALSE	Indication of whether originating devices for which the Exempt flag is set may override the security level indicated by the AllowedSecurityLevels, or if the or SecurityMinimum. If TRUE, this indicates that for originating devices with Exempt status, the incoming security level zero is acceptable, in addition to the incoming security levels meeting the minimum expected security level indicated by the SecurityMinimum element.
AllowedSecurityLevels	Set of integers	—	A set of allowed security levels, as defined in Table 145, for incoming MAC frames with the indicated frame type, and, if present, Command Identifier field. If the set is empty, then the SecurityMinimum parameter applies instead.

Table 151 defines the elements of a DeviceDescriptor.

Table 152 defines the elements of a KeyIdLookupDescriptor.

Table 151—Elements of DeviceDescriptor

Name	Type	Range	Description
PANId	Device PAN ID	0x0000–0xffff	The PAN identifier of the device in this DeviceDescriptor.
ShortAddress	Device short address	0x0000–0xffff	The short address of the device in this DeviceDescriptor. A value of 0xfffe indicates that this device is using only its extended address. A value of 0xffff indicates that this value is unknown.
ExtAddress	IEEE address	Any valid extended IEEE address	The extended IEEE address of the device.
FrameCounter	Integer	0x00000000–0xffffffff	The incoming frame counter of the device.
Exempt	Boolean	TRUE, FALSE	Indication of whether the device may override the minimum security level settings defined in Table 150.

Table 152—Elements of KeyIdLookupDescriptor

Name	Type	Range	Description
KeyIdMode	Integer	As defined in Table 146	The mode used to for this descriptor.
KeySource	Set of octets	As defined in 8.4.3.1	Information to identify the key. Present only if KeyIdMode is equal to 0x02 or 0x03
KeyIndex	Integer	As defined in 8.4.3.1	Information used to identify the key. Present only if KeyIdMode is not equal to 0x00.
DeviceAddrMode	Enumeration	NO_ADDRESS, SHORT_ADDRESS, EXTENDED_ADDRES	The addressing mode for this descriptor. Present only if KeyIdMode is equal to 0x00.
DevicePANId	Integer	0x0000–0xffff	The PAN identifier for this descriptor. Present only if KeyIdMode is equal to 0x00.
DeviceAddress	Device address	As specified by the DeviceAddrMode parameter	The device address for this descriptor. Present only if KeyIdMode is equal to 0x00.