IEEE P802.15
Wireless Personal
Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)		
Title	**IEEE P802-15_TG4 NTRU Security Architecture Proposal**		
Date Submitted	[June 12, 2002]		
Source	[Daniel V. Bailey, Ari Singer] [NTRU] [5 Burlington Woods Burlington, MA 01803 USA]	Voice: Fax: E-mail:	[+1 781 418-2522] [+1 781 418-2532] [dbailey@ntru.com]
Re:	Draft P802.15.4/D14, April-2002		
Abstract	[This document provides proposed security text to complete the security section and related text in other sections of the 802.15.4 draft standard. This text provides a complete specification of the proposed security architecture and a clear framework within which security suite specific requirements may be specified. This proposal focuses on limiting the requirements placed on the MAC while providing the flexibility to accommodate strong security implementations.]		
Purpose	[This document is intended as a security text submission to the 802.15 TG4 for inclusion in the 802.15.4 draft standard. It is intended that this submission will enable TG4 to define a simple security specification to be implemented by the MAC, while providing an interface to allow additional security services to be provided at higher layers. The text from this submission may be incorporated directly into the draft standard.]		
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.		

1. Introduction

The 802.15.4 draft D14 does not specify a complete or cohesive security architecture for providing security at the MAC layer. In an e-mail request from Jose Gutierrez, a number of requirements were put forth for any security solution for 802.15.4. The primary areas of interest are the following.

1) The security architecture must be algorithm agile to allow different implementations to utilize different algorithms with different security and efficiency properties.
2) There must be at least a baseline for interoperability for all devices.
3) The design must lend itself to reduced computational effort, and limited software and hardware complexity.
4) The required bandwidth overhead must be kept to a minimum
5) The cost of adding security must be kept to a minimum
6) The requirements must fit both kinds of 802.15.4 topology and be scalable and easy to maintain

In this proposal, we present a security architecture that focuses on the implementation of symmetric cryptography at the MAC layer, while pushing other security operations such as key management and providing freshness to the higher layers.

1.1 Scope

This document covers all text related to the security architecture for the 802.15.4 draft standard. In particular, this includes descriptions of the security model, security architecture and security services to be provided by 802.15.4 devices and it includes formats for security related messages. Additional informative text and security considerations supporting the security architecture are included as well.

1.2 Purpose

This document is intended to provide a complete security architecture for the 802.15.4 draft standard that more fully satisfies the requirements of the highly constrained and highly cost sensitive 802.15.4 devices. It is intended that this document be used to replace currently existing text in the 802.15.4 draft standard.

1.3 Notes to the Reader

Throughout the document, the author has included notes to the reader that are not part of the proposed text or submission. These notes are supplied to aid in the review process of this document and to indicate directions to the editor for portions of the standard to add or remove. These notes are not intended to be a part of the standard itself.

Author's note: Notes are prefixed by the words "Author's note:" and written in this font to indicate that they are not part of the intended draft text.

2. References

Author's note: The references to be listed in this section are normative references for the security implementations defined in this document.

NIST FIPS Pub 197: Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T., November 2001[1].

[1]NIST FIPS publications are available from the National Institute for Standards and Technology, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900 (http://www.nist.gov).

3. Clause 3 Text (Definitions)

Author's note: The following security related definitions are included to aid in the reading of this document as well as to provide additional definition text for the draft standard.

Access control list: A table used by a device to determine which devices are authorized to perform what functions.

Authentic data: Data that has its integrity cryptographically protected.

Confidentiality: Assurance that communicated data remains private to the parties for whom the data is intended.

Data integrity: Assurance that the data has not been modified from its original form.

Integrity code: A data string generated using a symmetric key that is typically appended to data in order to provide data integrity and source authentication. (also called a message integrity code)

Author's note: This is usually called a message authentication code (MAC) but that term will not be used here because of the confusion it may cause with the medium access control (MAC).

Key establishment: A public-key process by which two entities securely establish a symmetric key that is known only by the participating entities.

Key management: Methods for controlling keying material throughout its life cycle including creation, distribution and destruction

Key transport: A process by which an entity sends a key to another entity.

Message integrity code: See "integrity code."

Payload data: The contents of a data message that is being transmitted.

Payload protection: The generic term for providing security services on payload data, including confidentiality, integrity and authentication.

Pseudo-random number generation: The process of generating a deterministic sequence of bits from a given seed that has the statistical properties of a random sequence of bits when the seed is not known.

Random number generator: A device that provides a sequence of bits that is unpredictable.

Security suite: A group of security operations designed to provide security services on MAC frames.

Symmetric key: A secret key that is shared between two or more parties that may be used for encryption/decryption or integrity protection/integrity verification depending on its intended use.

4. Clause 4 Text (Acronyms)

Author's note: The following security related acronyms are included to aid in the reading of this document as well as to provide modified text for the draft standard.

ACL	access control list
AES	advanced encryption standard
CBC-MAC	cipher block chaining message authentication code
CCM	counter mode + CBC-MAC
CTR	counter mode
MIC	message integrity code
SEC	security

5. Clause 5 Text (General Description)

5.1 Functional overview

Author's note: Replace clause 5.4.2 with the following text.

5.1.1 Security Considerations

Although the diverse range of applications to which this standard is targeted imposes significant constraints on requiring a baseline security implementation in the MAC, some required security functionality is needed in order to provide basic security services and interoperability among all devices implementing this standard. This baseline includes the ability to maintain an access control list and use symmetric cryptography to protect transmitted frames. The ability to perform this security functionality does not imply, however, that security must be used at any given time by any given device. The higher layers determine when security is to be used at the MAC layer and provide all keying material necessary to provide the security services. Key management, device authentication and freshness protection may be provided by the higher layers, but are out of scope for this document.

Author's note: The beacon payload, data payload and command payload will be modified in clause 7 when security is in use. In order to minimize overhead and allow simple implementation of the acknowledgement frame, security will always be turned off in acknowledgement frames. This implies that there is no mechanism provided for a cryptographically assured acknowledgement of receipt. If this is desired, it must be implemented at a higher layer using data frames.

6. Clause 6 Text (PHY layer specification)

Author's note: The security mechanisms in this document deal exclusively with the MAC. No changes are required to the PHY specification.

7. Clause 7 Text (MAC sublayer specification)

7.1 MAC sublayer service specification

Author's note: Due to the increased complexity and cost of specifying key management operations at the MAC layer, we elected to recommend removal of all key management operation from the specification, including authentication. The protocols implemented by the commands that relate to key management and authentication may be implemented at a higher layer. The related commands and references to these commands should be removed from the draft.

Author's note: From the MAC's perspective, key management consists solely of the higher layers writing information into the MAC PIB. The symmetric keys and security relationship information will be written into the MAC PIB using the MLME-SET.request.

7.1.1 MAC data service

7.1.1.1 MCPS-DATA.request

7.1.1.1.1 Semantics of the service primitive

Author's note: Replace the table entry for TxOptions with the following entry:

Table 1—MCPS-DATA.request parameters

Name	Type	Valid Range	Description
TxOptions	Bitmap	0000-xxxx (Where x can be 0 or 1)	The transmission options for this MSDU. These shall be a bitwise OR of one or more of the following: 0x01 = immediate acknowledgment required. 0x02 = transmit in the current GTS. 0x04 = frame to follow 0x08 = security enabled

Author's note: Add the following text at the end of clause 7.1.1.1.3

If the TxOptions parameter specified that the frame does not have security enabled, the MAC sublayer shall set the SEC bit to zero and perform no security operations on the frame. If the TxOptions parameter specified that the frame has security enabled, the MAC sublayer shall set the SEC bit in the frame control field to 1, and obtain from the MAC PIB the appropriate key(s) and security suite (giving precedence to the entry in the ACL over the default) corresponding to the device address in the DstAddr field and perform the operations on the frame as indicated by the security suite using the selected key(s). If there is an error in the secure processing of the frame, the MLME shall return an MLME-SECURITY-ERROR.indication to the SSCS with the appropriate ReasonCode to indicate that the frame could not be protected.

7.1.1.2 MCPS-DATA.indication

7.1.1.2.1 Semantics of the service primitive

Author's note: Replace the MLME primitive in clause 7.1.1.3.1 with the following:

```
MCPS-DATA.indication    (
                        SrcPANId,
                        SrcAddr,
                        DstPANId,
                        DstAddr,
                        msduLength,
                        msdu,
                        mpduLinkQuality,
                        SecurityUse,
                        ACLEntry
                        )
```

Author's note: Add the following entry to Table 30 on page 53.

Table 2—MCPS-DATA.indication parameters

Name	Type	Valid Range	Description
SecurityUse	Boolean	TRUE or FALSE	This indicates to the higher layer if the received data frame had the security suite applied to it.
ACLEntry	Boolean	TRUE or FALSE	This indicates to the higher layer if the sender was found in the ACL.

7.1.2 MAC management service

Author's note: Remove the entries for MLME-AUTHENTICATE, MLME-CHALLENGE, MLME-DE-AUTHENTICATE, MLME-DISTRIBUTE-KEY and MLME-REQUEST-KEY from Table 31 on page 55.

Author's note: Add the following entry to table 31 on page 55.

Table 3—Summary of the primitives accessed through the MLME-SAP

Name	Request	Indication	Response	Confirm
MLME-SECURITY-ERROR		**7.x.x.x**		

7.1.3 Association primitives

7.1.3.1 MLME-ASSOCIATE.request

7.1.3.1.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-ASSOCIATE.request with the following interface:

```
MLME-ASSOCIATE.request      (
                            PID,
                            AssociationAddress,
                            CapabilityInformation,
                            AssocTimeoutPeriod,
                            TxOptions
                            )
```

Author's note: Insert the following entry into Table 32 on page 56:

Table 4—MLME-ASSOCIATE.request parameters

Name	Type	Valid Range	Description
TxOptions	Bitmap	0000-x000 (Where x can be 0 or 1)	The transmission options for this command. These shall be a bitwise OR of one or more of the following: 0x08 = security enabled

7.1.3.1.2 Effect on receipt

Author's note: Add the following text after the first paragraph in clause 7.1.3.1.3:

If the TxOptions parameter specified that the frame does not have security enabled, the MAC sublayer shall set the SEC bit to zero and perform no security operations on the frame. If the TxOptions parameter specified that the frame has security enabled, the MAC sublayer shall set the SEC bit in the frame control field to 1, and obtain from the MAC PIB the appropriate key(s) and security suite (giving precedence to the entry in the ACL over the default) corresponding to the device address in the AssociationAddress field and perform the operations on the frame as indicated by the security suite using the selected key(s). If there is an error in the secure processing of the frame, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the appropriate ReasonCode to indicate that the frame could not be protected.

7.1.3.2 MLME-ASSOCIATE.indication

7.1.3.2.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-ASSOCIATE.indication with the following interface:

```
MLME-ASSOCIATE.indication (
                          DeviceAddress,
                          CapabilityInformation,
                          AssocTimeoutPeriod,
                          SecurityUse,
                          ACLEntry
                          )
```

Author's note: Add the following entry to table 33 on page 57.

Table 5—MLME-ASSOCIATE.indication parameters

Name	Type	Valid Range	Description
SecurityUse	Boolean	TRUE or FALSE	This indicates to the higher layer if the received data frame had the security suite applied to it.
ACLEntry	Boolean	TRUE or FALSE	This indicates to the higher layer if the sender was found in the ACL.

7.1.3.3 MLME-ASSOCIATE.response

7.1.3.3.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-ASSOCIATE.response with the following interface:

```
MLME-ASSOCIATE.response (
                        DeviceAddress,
                        AssocDeviceAddress,
                        status,
                        TxOptions
                        )
```

Author's note: Insert the following entry into Table 34 on page 57:

Table 6—MLME-ASSOCIATE.reponse parameters

Name	Type	Valid Range	Description
TxOptions	Bitmap	0000-x000 (Where x can be 0 or 1)	The transmission options for this command. These shall be a bitwise OR of one or more of the following: 0x08 = security enabled

7.1.3.3.2 Effect on receipt

Author's note: Add the following text after the first paragraph in clause 7.1.3.3.3:

If the TxOptions parameter specified that the frame does not have security enabled, the MAC sublayer shall set the SEC bit to zero and perform no security operations on the frame. If the TxOptions parameter specified that the frame has security enabled, the MAC sublayer shall set the SEC bit in the frame control field to 1, and obtain from the MAC PIB the appropriate key(s) and security suite (giving precedence to the entry in the ACL over the default) corresponding to the device address in the AssocDeviceAddress field and perform the operations on the frame as indicated by the security suite using the selected key(s). If there is an error in the secure processing of the frame, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the appropriate ReasonCode to indicate that the frame could not be protected.

7.1.4 Disassociation primitives

7.1.4.1 MLME-DISASSOCIATE.request

7.1.4.1.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-DISASSOCIATE.request with the following interface:

```
MLME-DISASSOCIATE.request       (
                                DeviceAddress,
                                DisassociationReason,
                                TxOptions
                                )
```

Author's note: Insert the following entry into Table 36 on page 60:

Table 7—MLME-DISASSOCIATE.request parameters

Name	Type	Valid Range	Description
TxOptions	Bitmap	0000-x000 (Where x can be 0 or 1)	The transmission options for this command. These shall be a bitwise OR of one or more of the following: 0x08 = security enabled

7.1.4.1.2 Effect on receipt

Author's note: Add the following text after the first paragraph in clause 7.1.4.1.3:

If the TxOptions parameter specified that the frame does not have security enabled, the MAC sublayer shall set the SEC bit to zero and perform no security operations on the frame. If the TxOptions parameter specified that the frame has security enabled, the MAC sublayer shall set the SEC bit in the frame control field to 1, and obtain from the MAC PIB the appropriate key(s) and security suite (giving precedence to the entry in the ACL over the default) corresponding to the device address in the DeviceAddress field and perform the operations on the frame as indicated by the security suite using the selected key(s). If there is an error in the secure processing of the frame, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the appropriate ReasonCode to indicate that the frame could not be protected.

7.1.4.2 MLME-DISASSOCIATE.indication

7.1.4.2.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-DISASSOCIATE.indication with the following interface:

```
MLME-DISASSOCIATE.indication    (
                                DeviceAddress,
                                DisassociationReason,
                                SecurityUse,
                                ACLEntry
                                )
```

Author's note: Add the following entry to table 37 on page 60.

Table 8—MLME-DISASSOCIATE.indication parameters

Name	Type	Valid Range	Description
Security-Use	Boolean	TRUE or FALSE	This indicates to the higher layer if the received data frame had the security suite applied to it.
ACLEntry	Boolean	TRUE or FALSE	This indicates to the higher layer if the sender was found in the ACL.

7.1.5 Authentication and challenge

Author's note: Remove sub-clause 7.1.5 in its entirety.

7.1.6 Request key

Author's note: Remove sub-clause 7.1.6 in its entirety.

7.1.7 Distribute key

Author's note: Remove sub-clause 7.1.7 in its entirety.

7.1.8 De-authentication

Author's note: Remove sub-clause 7.1.8 in its entirety.

7.1.9 Beacon notification primitive

7.1.9.1 MLME-BEACON-NOTIFY.indication

7.1.9.1.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-BEACON-NOTIFY.indication with the following interface:

```
MLME-BEACON-NOTIFY.indication (
                              PANid,
                              SrcAddr,
                              SuperframeSpec,
                              PendAddrSpec,
                              AddrList,
                              sduLength,
                              sdu,
                              LinkQuality,
                              TimeStamp,
                              SecurityUse,
                              ACLEntry
                              )
```

Author's note: Add the following entry to table 57 on page 82.

Table 9—MLME-BEACON-NOTIFY.indication parameters

Name	Type	Valid Range	Description
Security-Use	Boolean	TRUE or FALSE	This indicates to the higher layer if the received data frame had the security suite applied to it.
ACLEntry	Boolean	TRUE or FALSE	This indicates to the higher layer if the sender was found in the ACL.

7.1.10 Primitives for reading MAC PIB attributes

7.1.10.1 MLME-GET.request

7.1.10.1.1 Semantics of the service primitive

Author's note: Since a new table was added for the MAC PIB ACL, this should be referenced in this sub-clause. Add a reference to Table 18 from this document under the Valid Range for PIBAttribute in Table 59 on page 84.

7.1.10.2 MLME-GET.confirm

7.1.10.2.1 Semantics of the service primitive

Author's note: Since a new table was added for the MAC PIB ACL, this should be referenced in this sub-clause. Add a reference to Table 18 from this document under the Valid Range for PIBAttribute and PIBAttributeValue in Table 60 on page 85.

7.1.11 GTS management primitives

7.1.11.1 MLME-GTS.request

7.1.11.1.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-GTS.request with the following interface:

```
MLME-GTS.request        (
                        GTSId,
                        GTSCharacteristics,
                        TxOptions
                        )
```

Author's note: Insert the following entry into Table 61 on page 86:

Table 10—MLME-GTS.request parameters

Name	Type	Valid Range	Description
TxOptions	Bitmap	0000-x000 (Where x can be 0 or 1)	The transmission options for this command. These shall be a bitwise OR of one or more of the following: 0x08 = security enabled

7.1.11.1.2 Effect on receipt

Author's note: Add the following text after the first paragraph in clause 7.1.4.1.3:

If the TxOptions parameter specified that the frame does not have security enabled, the MAC sublayer shall set the SEC bit to zero and perform no security operations on the frame. If the TxOptions parameter specified that the frame has security enabled, the MAC sublayer shall set the SEC bit in the frame control field to 1, and obtain from the MAC PIB the appropriate key(s) and security suite (giving precedence to the entry in the ACL over the default) corresponding to the device address of the PAN coordinator and perform the operations on the frame as indicated by the security suite using the selected key(s). If there is an error in the secure processing of the frame, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the appropriate ReasonCode to indicate that the frame could not be protected.

7.1.11.2 MLME-GTS.indication

7.1.11.2.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-GTS.indication with the following interface:

```
MLME-GTS.indication     (
                        GTSId,
                        SecurityUse,
                        ACLEntry
                        )
```

Author's note: Add the following entry to table 63 on page 88.

Table 11—MLME-GTS.indication parameters

Name	Type	Valid Range	Description
Security-Use	Boolean	TRUE or FALSE	This indicates to the higher layer if the received data frame had the security suite applied to it.
ACLEntry	Boolean	TRUE or FALSE	This indicates to the higher layer if the sender was found in the ACL.

7.1.12 Primitives for orphan notification

7.1.12.1 MLME-ORPHAN.indication

7.1.12.1.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-ORPHAN.indication with the following interface:

```
MLME-ORPHAN.indication  (
                        OrphanAddress,
                        SecurityUse,
                        ACLEntry
                        )
```

Author's note: Add the following entry to table 64 on page 90.

Table 12—MLME-ORPHAN.indication parameters

Name	Type	Valid Range	Description
Security-Use	Boolean	TRUE or FALSE	This indicates to the higher layer if the received data frame had the security suite applied to it.
ACLEntry	Boolean	TRUE or FALSE	This indicates to the higher layer if the sender was found in the ACL.

7.1.12.2 MLME-ORPHAN.response

7.1.12.2.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-ORPHAN.response with the following interface:

```
MLME-ORPHAN.response        (
                            ExtendedAddress,
                            AllocatedAddress,
                            PANMember,
                            TxOptions
                            )
```

Author's note: Insert the following entry into Table 65 on page 91:

Table 13—MLME-ORPHAN.reponse parameters

Name	Type	Valid Range	Description
TxOptions	Bitmap	0000-x000 (Where x can be 0 or 1)	The transmission options for this command. These shall be a bitwise OR of one or more of the following: 0x08 = security enabled

7.1.12.2.2 Effect on receipt

Author's note: Add the following text after the first paragraph in clause 7.1.13.2.3:

If the TxOptions parameter specified that the frame does not have security enabled, the MAC sublayer shall set the SEC bit to zero and perform no security operations on the frame. If the TxOptions parameter specified that the frame has security enabled, the MAC sublayer shall set the SEC bit in the frame control field to 1, and obtain from the MAC PIB the appropriate key(s) and security suite (giving precedence to the entry in the ACL over the default) corresponding to the device address of the orphaned device and perform the operations on the frame as indicated by the security suite using the selected key(s). If there is an error in the secure processing of the frame, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the appropriate ReasonCode to indicate that the frame could not be protected.

7.1.13 Primitives for writing MAC PIB attributes

7.1.13.1 MLME-SET.request

7.1.13.1.1 Semantics of the service primitive

Author's note: Since a new table was added for the MAC PIB ACL, this should be referenced in this sub-clause. Add a reference to Table 18 from this document under the Valid Range for PIBAttribute and PIBAttributeValue in Table 70 on page 96.

7.1.13.2 MLME-SET.confirm

7.1.13.2.1 Semantics of the service primitive

Author's note: Since a new table was added for the MAC PIB ACL, this should be referenced in this sub-clause. Add a reference to Table 18 from this document under the Valid Range for PIBAttribute in Table 71 on page 97.

7.1.14 Primitives for starting beacon transmissions

7.1.14.1 MLME-START.request

7.1.14.1.1 Semantics of the service primitive

Author's note: Replace the interface for MLME-START.request with the following interface:

```
MLME-START.request      (
                        BPID,
                        LogicalChannel,
                        SuperframeOrder,
                        PANCoordinator,
                        GTSCapability,
                        TxOptions
                        )
```

Author's note: Insert the following entry into Table 72 on page 99:

Table 14—MLME-START.request parameters

Name	Type	Valid Range	Description
TxOptions	Bitmap	0000-x000 (Where x can be 0 or 1)	The transmission options for this command. These shall be a bitwise OR of one or more of the following: 0x08 = security enabled

7.1.14.1.2 Effect on receipt

Author's note: Add the following text after the first paragraph in clause 7.1.17.1.3:

If the TxOptions parameter specified that the beacon frame does not have security enabled, the MAC sublayer shall set the SEC bit to zero and perform no security operations on the beacon frame. If the TxOptions parameter specified that the frame has security enabled, the MAC sublayer shall set the SEC bit in the frame control field to 1, and obtain from the MAC PIB the appropriate key(s) and security suite (giving precedence to the entry in the ACL over the default) corresponding to the broadcast address and perform the operations on the frame as indicated by the security suite using the selected key(s). If there is an error in the secure processing of the frame, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the appropriate ReasonCode to indicate that the frame could not be protected.

Author's note: Add the following sub-clause to clause 7.1.

7.1.15 Security management primitives

These primitives define how the MLME communicates security related events to the SSCS.

7.1.15.1 MLME-SECURITY-ERROR.indication

This primitive allows the MLME to indicate a failed security processing operation.

7.1.15.1.1 Semantics of the service primitive

This primitive shall provide the following interface:

```
MLME-SECURITY-ERROR.indication (
                                PANId
                                SrcAddress
                                DestAddress
                                ReasonCode
                                )
```

Table 15 specified the parameters for the MLME-SECURITY-ERROR.indication primitive.

Table 15—MLME-SECURITY-ERROR.indication parameters

Name	Type	Valid Range	Description
PANId	Integer	0x0000 - 0xffff	The 16-bit PAN identifier of the device from which the frame was received or to which the frame was being sent.
SrcAddress	Device address	A short 8-bit, allocated address or an extended 64-bit, IEEE address.	The individual device address of the entity from which the frame causing the error originated.
DestAddress	Device address	A short 8-bit, allocated address or an extended 64-bit, IEEE address.	The individual device address of the device that the frame was intended for.
ReasonCode	Enumeration	UNAVAILABLE-KEY, FAILED-SECURITY-CHECK	The reason for the security error.

7.1.15.1.2 When generated

This primitive is issued by the MLME when it receives an MLME message from a higher layer that requires security, but it is unable to find an appropriate key in the ACL or when it receives a validly formatted frame from another device that induces a failed security check according to the security suite.

7.1.15.1.3 Effect on receipt

On receipt of this primitive, the next higher layer of the initiating device is notified of a security error and the reason for the security error.

Author's note: The four errors that are being caught here are the case when the higher layer tells the MAC to protect a frame that it is unable to protect (e.g. the ACL is missing a key to be used for that device), the case when the MAC is unable to apply the security to the outgoing frame, the case when it receives a frame that has the SEC bit set to 1, but it is unable to process the security (e.g. the ACL is missing a key to be used for that device), and the case when it receives a frame that has the SEC bit set to 1 and the security fails a check (e.g. a faulty integrity code).

7.2 MAC frame formats

7.2.1 General MAC frame format

7.2.1.1 Frame control field

Author's note: Replace table 78 with the following table:

Table 16—Format of the frame control field

Bits: 0-2	3	4	5	6	7-8	9	10-11	12	13	14-15
Frame type	Dest. fields present	Dest. addressing mode	Source fields present	Source addressing mode	Reserved	SEC	Frame fragment specifier	Frame sequence bit	Frame following	Ack. policy

Author's note: Add the following sub-clause to clause 7.2.1.1:

7.2.1.1.1 SEC field

The SEC field is one bit in length. When the SEC bit is set to 0, the frame is not cryptographically protected by the MAC. When the SEC bit is set to 1, the frame is protected using the keys stored in the MAC PIB for that security relationship. The cryptographic operations used to protect the frame are defined by the security suite selected for that security relationship. If no security suite is defined for that relationship, the SEC bit shall be set to 0.

7.2.1.2 Payload field

Author's note: Replace clause 7.2.1.7 with the following text:

The payload field has a variable length and contains information specific to individual frame types. If the SEC bit is set to 1 in the frame control field, the payload is protected as defined by the security suite selected for that relationship.

7.2.2 Format of individual frame types

7.2.2.1 Beacon frame format

7.2.2.1.1 Beacon frame MAC header field

Author's note: Replace the second paragraph in clause 7.2.2.1.1 with the following text:

In the frame control field, the frame type field shall contain the value which indicates a beacon frame, as shown in Table 79, the source fields present field shall be set to one and the source addressing mode field shall be set as appropriate for the address of the beacon. If security is used for the beacon, the SEC bit shall be set to one. All other fields shall be set to zero and ignored on reception.

7.2.2.1.2 Beacon frame payload field

Author's note: Append the following sentence to the end of the paragraph in clause 7.2.2.1.5

If the SEC field is set to 1 in the frame control field, the device shall process the frame according to the selected security suite.

7.2.2.2 Data frame format

7.2.2.2.1 Data payload field

Author's note: Replace the text in clause 7.2.2.2.2 with the following text:

If the SEC bit in the frame control field is set to 0, the payload of a data frame shall contain the sequence of bytes which the next higher layer has requested the MAC sublayer to transmit. If the SEC bit is set to 1, the device shall process the frame according to the selected security suite before passing the data to the higher layer.

7.2.2.3 MAC command frame format

7.2.2.3.1 Command payload field

Author's note: Replace the text in clause 7.2.2.4.3 with the following text:

If the SEC bit in the frame control field is set to 0, the command payload field contains the MAC command itself. If the SEC bit in the frame control field is set to 1, the device shall process the frame according to the selected security suite before processing the command. The formats of the individual commands are described in clause 7.3.

7.3 MAC command frames

7.3.1 Association and disassociation

Author's note: Remove the entries for Authentication request, Authentication response, Challenge request, Challenge response, Request key request, Request key response, Distribute key request, Distribute key response and De-Authenticate request.

7.3.1.1 Association request command

Author's note: Add the following text to clause 7.3.1.1 after the third paragraph:

If security is used for the association request command, the SEC bit shall be set to 1 and the frame shall be protected by the method defined by the selected security suite. Otherwise, the SEC bit shall be set to 0.

7.3.1.1.1 Capability information field

Author's note: Replace table 90 with the following table.

Table 17—Capability information field format

bits: b0	b1	b2	b3-b4	b5	b6	b7
Alternative coordinator	GTS Capability	Power Source	Reserved	Security capability	Allocate address	Withhold address

Author's note: Add the following text after the third paragraph after table 90 in clause 7.3.1.1.1:

The security capability subfield is one bit in length and shall be set to 1 if the device is capable of sending and receiving MAC frames secured by the mandatory security suite. Otherwise the security capability subfield shall be set to 0.

7.3.1.2 Association response command

Author's note: Add the following text after the third paragraph in clause 7.3.1.2

If security is used for the association response command, the SEC bit shall be set to 1 and the frame shall be protected by the method defined by the selected security suite. Otherwise, the SEC bit shall be set to 0.

7.3.1.3 Disassociation notification command

Author's note: Add the following text after the second paragraph in clause 7.3.1.2

If security is used for the disassociation notification response command, the SEC bit shall be set to 1 and the frame shall be protected by the method defined by the selected security suite. Otherwise, the SEC bit shall be set to 0.

7.3.2 Authentication and challenge

Author's note: Remove sub-clause 7.3.2 in its entirety.

7.3.3 Request key

Author's note: Remove sub-clause 7.3.3 in its entirety

7.3.4 Distribute key

Author's note: Remove sub-clause 7.3.4 in its entirety

7.3.5 De-authenticate

Author's note: Remove sub-clause 7.3.5 in its entirety

7.3.6 GTS allocation and deallocation

7.3.6.1 GTS request command

Author's note: Add the following text after the second paragraph in clause 7.3.6.1

If security is used for the GTS request command, the SEC bit shall be set to 1 and the frame shall be protected by the method defined by the selected security suite. Otherwise, the SEC bit shall be set to 0.

7.3.6.2 GTS allocation command

Author's note: Add the following text after the second paragraph in clause 7.3.6.2

If security is used for the GTS allocation command, the SEC bit shall be set to 1 and the frame shall be protected by the method defined by the selected security suite. Otherwise, the SEC bit shall be set to 0.

7.3.7 Coordinator interaction

7.3.7.1 Coordinator interaction command

Author's note: Add the following text after the first paragraph after table 108 in clause 7.3.7.1

If security is used for the coordinator interaction command, the SEC bit shall be set to 1 and the frame shall be protected by the method defined by the selected security suite. Otherwise, the SEC bit shall be set to 0.

7.3.7.2 Coordinator realignment command

Author's note: Add the following text after the second paragraph in clause 7.3.7.2

If security is used for the coordinator realignment command, the SEC bit shall be set to 1 and the frame shall be protected by the method defined by the selected security suite. Otherwise, the SEC bit shall be set to 0.

7.4 MAC functional description

Author's note: The following table should be included along with Table 114 to indicate the MAC PIB Security Attributes.

Author's note:

Table 18—MAC PIB ACL Entries

Attribute	Identifier	Type	Range	Description	Default
macSecurity-Mode	mSM	Integer	0-2	The identifier of the security mode in use as specified in 7.4.1. 0 = Unsecured mode 1 = ACL mode 2 = Secured mode	0
macDefaultSecu-rity	mDS	Boolean	TRUE or FALSE	This indicates if the device is able to accept or transmit secure frames to devices that are not in the ACL.	FALSE
macDefaultSecu-ritySuite	mDSS	Integer	0x00 - 0xff	The unique identifier of the security suite to be used to protect communica-tions between the MAC and devices not in the ACL as specified in Table 19.	0
macDefaultSecu-rityMaterial	mDSM	Byte string	Variable	The specific security material to be used to protect frames between the MAC and devices not in the ACL.	Empty string
macACLPANId	mACLPI	Integer	0x0000 - 0xffff	The 16-bit PAN identifier of the device in this ACL entry.	Device specific
macACLAddress	mACLA	DevAddress	Any valid 64-bit device address	The 64-bit extended IEEE address of the device in this ACL entry.	Device specific
macACLPANA-dress	mACLPA	Integer	0x00 - 0xfe	The allocated device address of the device in this ACL entry.	Device specific

macACLSecuri-tySuite	mACLSS	Integer	0x00 - 0xff	The unique identifier of the security suite to be used to protect communications between the MAC and the device indicated by the associated macACLAddress as specified in Table 19.	0
macACLSecuri-tyMaterial	mACLSM	Byte string	Variable	The specific keying material to be used to protect frames between the MAC and the device indicated by the associated macACLAddress.	Empty string

Author's note: Add the following text to the end of the text in clause 7.5:

When the next higher layer requests that security be implemented on a particular frame or upon receipt of a frame, the MAC uses the MAC PIB ACL mechanisms defined in 7.4.1.

Author's note: Add the following sub-clauses to the end of clause 7.5 (after clause 7.5.9):

7.4.1 Frame Security

The MAC is responsible for providing security services on specified incoming and outgoing frames when requested to do so by the higher layers. The information for when and how to provide the security is found in the MAC PIB ACL.

The MAC PIB ACL contains an indication of the security mode in which the device is operating. There are three different modes of security that may be implemented at the MAC layer: unsecured mode, ACL mode and secured mode. All devices shall support unsecured mode and devices that support secured mode shall also support ACL mode. The current mode is specified by the macSecurityMode field in the MAC PIB ACL.

The MAC PIB ACL also contains a single default ACL entry and a number of additional ACL entries. Each ACL entry contains an indication of a device or devices, an indication of the cryptographic operations it is to perform on sent and received frames from the device(s), and security material (including a cryptographic key or keys) that shall be used to perform the cryptographic operations. The cryptographic operations performed on the frames are defined by the security suites and utilize the associated security material. Each device may have a number of security suites that it is able to implement. The security suites are specified in 7.6.

The default ACL entry consists of the macDefaultSecurity field, which indicates if security is in use for devices not in the ACL, the macDefaultSecuritySuite field, which indicates the default security suite to use for frames to and from devices not in the ACL and the macDefaultSecurityMaterial field, which indicates the keying material to use in secure communications to and from devices not in the ACL. If the macDefaultSecurity field is set to FALSE, the macDefaultSecuritySuite field and the macDefaultSecurityMaterial fields shall not be used.

The additional ACL entries consist of a PANId, extended IEEE address and allocated PAN address of a peer device stored in macACLPANId, macACLAddress and macACLPANAdress fields respectively, an associated security suite, stored in a macACLSecuritySuite field, and associated keying material that is stored in a macACLSecurityMaterial field.

7.4.1.1 Unsecured mode

Unsecured mode is the default security mode for the MAC and provides no MAC layer security. A device operating in unsecured mode shall not utilize the ACL entries and shall not perform any security related

operations on MAC frames. While in this mode, if the MAC receives a frame with the SEC field set to 1, the MAC shall discard the frame and the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layer with the ReasonCode set to UNAVAILABLE-KEY. If the MAC receives a frame with the SEC field set to 0, the MLME shall set the SecurityUse and ACLEntry fields to FALSE in the indication to the higher layer.

7.4.1.2 ACL mode

Author's note: The decision was made for the MAC to accept frames from devices regardless of whether they are in the ACL and to simply indicate to the higher layers (using the ACLEntry) field if it found the device address in the ACL.

ACL mode provides a mechanism for the MAC to indicate to the higher layer if a received frame purportedly originated from a device in the ACL. A device operating in ACL mode shall not perform any security related operations on MAC frames. While in this mode, if the MAC receives a frame with the SEC field set to 1, the MAC shall discard the frame and the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layer with the ReasonCode set to UNAVAILABLE-KEY. If the MAC receives a frame with the SEC field set to 0, the MLME shall set the SecurityUse field to FALSE and the ACLEntry field to TRUE or FALSE depending on if the sender is in the ACL in the indication to the higher layer. The device shall determine if a device is in the ACL by searching for an individual ACL entry with a macACLPANId that matches the received PANId and with a macACLAddress or macACLPANAddress that matches the received source address. If no source address is present in the frame, the ACLEntry field shall be set to FALSE.

7.4.1.3 Secured mode

Secured mode provides a mechanism for the MAC to both use the ACL functionality and provide cryptographic protection on incoming and outgoing frames. While in this mode, if the MAC receives a frame or a request from the higher layer to transmit a frame, the MAC shall process the frame as specified in the following sub-clauses.

7.4.1.3.1 Using the ACL for outgoing frames

If the MLME receives a message from a higher layer to prepare a secure frame for transmission, the MLME shall scan the entries in the MAC PIB ACL for the correct entry to use. The MLME shall first search through the list of macACLPANId and associated macACLAddress and macACLPANAddress entries to find an entry that matches the destination address of the frame to be created. If a match is found, the MLME shall perform operations on the frame according to the security suite from the associated macACLSecuritySuite field and utilize the keying material from the associated macACLSecurityMaterial field.

If the MLME is unable to locate a macACLPANId and macACLAddress or macACLPANAddress that matches the destination address of the frame to be created, the MLME shall examine the macDefaultSecurity field. If the macDefaultSecurity field is set to TRUE, the MLME shall perform operations on the frame according to the security suite from the macDefaultSecuritySuite field and utilize the keying material from the macDefaultSecurityMaterial field.

If the MLME is unable to locate a macACLPANId and macACLAddress or macACLPANAddress that matches the destination address of the frame to be created and the macDefaultSecurity field is set to FALSE, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layer with the ReasonCode set to UNAVAILABLE-KEY.

7.4.1.3.2 Using the ACL for incoming frames

Author's note: The decision was made for the MAC to accept both secured (SEC = 1) and unsecured (SEC = 0) frames whenever it receives them and to simply indicate to the higher layers (using the SecurityUse field) if it was secured or

unsecured. The decision to accept or reject the frame based on security considerations is left to the higher layers. This implies that the MAC should not perform any action based on a received frame aside from processing it and sending the appropriate indication to the higher layers. It might be possible instead to add information to the ACL indicating when the MAC should accept or reject unsecured commands if the group desires that feature.

Any incoming frame may be protected by security. If the MLME receives a frame from another device, the MLME shall check the SEC bit in the frame control field to determine if security is used for that frame. If the MLME is able to process the frame properly, the MLME shall set the SecurityUse field to TRUE or FALSE depending on if the SEC bit was set to 1 or 0 and set the ACLEntry field to TRUE or FALSE depending on if the sender is in the ACL in the indication to the higher layer.

If the SEC bit in the frame control field is set to 0, the device shall not perform security operations on the frame and shall process the frame as if it were in ACL mode.

If the SEC bit in the frame control field is set to 1 the MLME shall scan the entries in the MAC PIB ACL for the correct entry to use. The MLME shall first search through the list of macACLPANId and associated macACLAddress and macACLPANAddress entries to find an entry that matches the source address of the frame received. If a match is found, the MLME shall perform operations on the frame according to the security suite from the associated macACLSecuritySuite field and utilize the keying material from the associated macACLSecurityMaterial field.

If the MLME is unable to locate a macACLPANId and macACLAddress or macACLPANAddress that matches the source address of the received frame, the MLME shall examine the macDefaultSecurity field. If the macDefaultSecurity field is set to TRUE, the MLME shall perform operations on the frame according to the security suite from the macDefaultSecuritySuite field and utilize the keying material from the macDefaultSecurityMaterial field.

If the MLME is unable to locate a macACLPANId and macACLAddress or macACLPANAddress that matches the source address of the received frame and the macDefaultSecurity field is set to FALSE, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layer with the ReasonCode set to UNAVAILABLE-KEY.

7.4.1.3.3 Applying security to frames

After the MLME receives a request to transmit a secure frame and obtains the appropriate security suite and key(s) from the ACL, the MAC shall apply the operations defined by the security suite using the key(s) to the frame.

Author's note: Since we define encryption and integrity fields to only be included in payload fields of MAC frames, ACK frames are not affected by the security suite and the SEC bit in the frame control field should be set to 0 for all ACK frames.

If the security suite defines encryption, the encryption operation shall be applied to data in the MAC payload field of the frame only. The remaining fields shall be left unencrypted. The result of the encryption operation shall be inserted into the MAC payload field of the frame in the place of the data that was encrypted.

If the security suite defines an integrity code, the integrity code shall be applied to the MAC header concatenated with the MAC payload. The result of the integrity code computation shall be placed in the MAC payload field of the frame in addition to any other data in the payload field.

The ordering of the encryption and integrity operations and the placement within the MAC payload field of the outputs of those operations is defined by the security suite.

If any of the security operations fail, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the ReasonCode set to FAILED-SECURITY-CHECK and shall not transmit the requested frame.

If the security operations have been successfully performed and the payload field has been modified appropriately, the device shall then compute the FCS over the modified frame.

Author's note: In the case that the security suite implements an integrity code, it may be possible to remove the FCS, since the integrity code passing implies that the frame was received in proper order. If bandwidth requirements demand this, the specification can be modified appropriately to include the FCS as a security suite specific object on secure frames.

7.4.1.3.4 Removing security from frames

After the MLME receives a frame that has the SEC bit set to 1 and obtains the appropriate security suite and key(s) from the ACL, the MAC shall apply the operations defined by the security suite using the key(s) to the frame.

Before the security operations have been performed and the payload field has been modified, the MLME shall check the FCS.

If the security suite defines encryption, the decryption operation shall be applied to data in the MAC payload field of the frame only. The result of the decryption operation shall be inserted into the MAC payload field of the frame in the place of the encrypted data.

If the security suite defines an integrity code, the integrity code shall be checked by first removing the integrity code from the MAC payload and then verifying the integrity code on the MAC header concatenated with the MAC payload.

The ordering of the encryption and integrity operations and the location of the security data within the payload field is defined by the security suite.

If any of the security operations fail, the MLME shall return an MLME-SECURITY-ERROR.indication to the higher layers with the ReasonCode set to FAILED-SECURITY-CHECK and shall not perform any additional operations on the received frame.

If the security operations have been successfully performed and the payload field has been modified appropriately, the device may then continue to process the frame.

Author's note: Add the following sub-clause on security services to clause 7.

7.5 Security services

Security services are protections offered on communication between devices. Security services such as authentication and key establishment are outside of scope for this standard and are left to the higher layers.

7.5.1 Access control

Access control is a security service that provides the ability for a device to select which devices it is willing to communicate with. In this standard, if the access control service is provided, the device shall maintain a list of devices in its ACL from which it expects to receive frames.

7.5.2 Data encryption

Data encryption is a security service that uses a symmetric cipher to protect data from being read by parties without the cryptographic key. Data may be encrypted using a key shared by a group of devices (typically stored as the default key) or using a key shared between two peers (typically stored in an individual ACL entry). In this standard, data encryption may be provided on beacon payloads, command payloads and data payloads.

7.5.3 Frame integrity

Frame integrity is a security service that uses a message integrity code (MIC) to protect data from being modified by parties without the cryptographic key. It further provides assurance that data came from a party with the cryptographic key. In this standard, integrity may be provided on data frames, beacon frames and command frames. The key used to provide frame integrity may be shared by a group of devices (typically stored as the default key) or by two peers (typically stored in an individual ACL entry).

7.5.4 Sequential freshness

Sequential freshness is a security service that uses an ordered sequence of inputs to protect data with an old sequence value from being replayed. This service provides evidence that the received data is newer than the last data received by that device, but it does not provide a strict sense of time.

7.5.5 Security services for the modes

Depending on the mode that the device is operating in, the MAC may be able to provide different security services.

7.5.5.1 Security services for unsecured mode

Since security is not used for unsecured mode, no security services are provided by devices operating in unsecured mode.

7.5.5.2 Security services for ACL mode

Devices operating in ACL mode may be able to provide limited security services for communications with other devices. While in ACL mode, the device may provide the following security service:

— Access control, specified in 7.5.1

Author's note: Providing the access control functionality without integrity protection on the frames may allow a device not in the ACL to spoof frames from devices that are in the ACL. If ACL mode is used, the sender and receiver should have other means to verify the source and authenticity of frames. However, since this is outside the scope of the MAC/PHY functionality, this observation need not be included as part of the standards text.

7.5.5.3 Security services for secured mode

Devices operating in secured mode may be able to provide any of the security services defined in 7.5. The specific security services are dependent on the security suite in use and these services are specified by the security suite itself. Services that may be provided while in secured mode include:

— Access control, specified in 7.5.1
— Data encryption, specified in 7.5.2
— Frame integrity, specified in 7.5.3
— Sequential freshness, specified in 7.5.4

Author's note: Providing a security suite that offers the data encryption functionality without integrity protection on the frames may allow a device not in the ACL to spoof frames from devices that are in the ACL and modify the encrypted data. If data encryption is provided, the receiver should have other means to verify the source and authenticity of frames. Again, this is outside the scope of the MAC/PHY functionality, so this need not be included in the standard.

7.6 Security Suite Specifications

7.6.1 Overview

Each device that implements security shall support one or more security suites. Security suites may be used when a device is operating in secured mode. A security suite consists of a set of operations to perform on MAC frames that provide security services on the MAC frames. Each security suite is specified by a one-byte value as shown in Table 19.

Table 19—Security suite list

Security Suite Name	Identifier
AES-CTR	0x01
AES-CCM-128	0x02
AES-CCM-64	0x03
AES-CCM-32	0x04
AES-CBC-MAC-128	0x05
AES-CBC-MAC-64	0x06
AES-CBC-MAC-32	0x07

Author's note: The group requested AES-CCM security suites with message integrity code lengths of 40-bits and 128-bits, but the CCM definition does not provide for 40-bit message integrity codes. Therefore, we decided to define the 64-bit message integrity code, which has a smaller size than 128 bits and should provide sufficient security even for reasonably high security applications and we also defined the 32-bit message integrity code for the highly space constrained devices that don't need the longer MIC.

7.6.2 Mandatory security suite

Devices may support no security suites. However, in order to ensure that all security enabled devices are able to interoperate, all devices that implement any security suite shall implement the AES-CCM-64 security suite. This security suite is specified in clause 7.6.6. All other security suites may be implemented by a compliant device.

7.6.3 Security suite building blocks

The following definitions and primitives are defined for use in the security suites specified in this standard.

7.6.3.1 Bit ordering

For security operations in this standard, a bit is defined to be an element of the set {0, 1}. A bit string is defined to be an ordered array of bits. A byte (also called an octet) is defined to be a bit string of length 8. A

byte string (also called an octet string) is an ordered array of bytes. The terms first and last, leftmost and rightmost, and most significant and least significant are used to distinguish the ends of these sequences (first, leftmost and most significant are equivalent; last, rightmost and least significant are equivalent). Within a byte, we additionally refer to the high-order and low-order bits, where high-order is equivalent to first and low-order is equivalent to last.

Note that when a string is represented as a sequence, it may be indexed from left to right or from right to left, starting with any index. For example, consider the octet string of two octets: 2a 1b. This corresponds to the bit string 0010 1010 0001 1011. No matter what indexing system is used, the first octet is still 2a, the first bit is still 0, the last octet is still 1b, and the last bit is still 1. The high-order bit of the second octet is 0; the low-order bit of the second octet is 1.

7.6.3.2 Concatenation

In this standard, concatenation of two byte strings *a* and *b* of length *m* and *n* respectively, denoted *a*||*b*, consists of the byte string of length *m*+*n* with the leftmost *m* bytes equal to *a* and the rightmost *n* bytes equal to *b*.

7.6.3.3 Integer encoding and counter incrementing

The counter incrementing operation in this standard takes as input an integer that is encoded as a byte string of length *n* in which the bits from right to left give the binary representation of the integer in increasing significance. When the counter incrementing operation is invoked, the integer shall be incremented (increased by 1). If the incremented integer is less than 2^{8n}, the operation shall return the byte string encoding of the new integer, otherwise the operation shall return an error.

7.6.3.4 CTR encryption

The CTR symmetric encryption algorithm used in this standard consists of the generation of a key stream using a block cipher in counter mode with a given key and nonce and performing an XOR of the key stream with the plaintext and integrity code. The decryption operation consists of the generation of the key stream and the XOR of the key stream with the ciphertext to obtain the plaintext.

All of the above operations shall be performed as specified in Annex A.2. Security suites implementing CTR encryption shall specify the parameters for these operations.

7.6.3.5 CBC-MAC authentication

The CBC-MAC symmetric authentication algorithm used in this standard consists of the generation of an integrity code using a block cipher in CBC mode computed on a message that includes the length of the authenticated data at the beginning of the data itself. The verification operation consists of the computation of this integrity code and comparison to the received integrity code.

All of the above operations shall be performed as specified in Annex A.3. Security suites implementing CBC-MAC authentication shall specify the parameters for these operations.

7.6.3.6 CTR + CBC-MAC (CCM) combined encryption and authentication

The CCM combined symmetric encryption and authentication mechanisms used in this standard consists of the generation of an integrity code followed by the encryption of plaintext data and the integrity code. The output consists of the encrypted data and the encrypted integrity code.

The symmetric authentication operation used in this security suite consists of the generation of an integrity code using a block cipher in CBC mode computed on a nonce followed by (optional) padded authentication

data followed by (optional) padded plaintext data. The verification operation consists of the computation of this integrity code and comparison to the received integrity code.

The symmetric encryption operation used in this security suite consists of the generation of a key stream using a block cipher in counter mode with a given key and nonce and performing an XOR of the key stream with the plaintext and integrity code. The decryption operation consists of the generation of the key stream and the XOR of the key stream with the ciphertext to obtain the plaintext and integrity code.

All of the above operations shall be performed as specified in Annex A.1. Security suites implementing CCM shall specify the parameters for these operations.

7.6.3.7 AES encryption

The advanced encryption standard (AES) encryption algorithm used in this security suite shall be performed as specified in the FIPS 197 standard [{xref}FIP197]. This encryption algorithm is parameterized by the use of 128-bit keys and 128-bit block size.

7.6.4 AES-CTR security suite

The AES-CTR security suite consists of performing AES-CTR encryption (or decryption) on the MAC payload using shared data, the frame counter, the key sequence counter and the (encrypted) MAC payload.

The AES-CTR security suite provides the following security services:

— Access control, specified in 7.5.1
— Data encryption, specified in 7.5.2
— Sequential freshness, specified in 7.5.4 (optional)

7.6.4.1 Data formats

The following sub-clauses define the data formats used in this security suite.

7.6.4.1.1 MAC PIB formats

Author's note: For all of the CTR and CCM modes in this document, we have included optional fields and operations that may be performed on the recipient side to provide sequential freshness. Since the MAC increments its frame counter each time it uses it, performing the freshness check will should not usually cause any errors except that it will catch duplicates and reject replayed messages. Since the key sequence counter is controlled only by the higher layers, it is up to the higher layers to ensure that a smaller key sequence counter is not inserted into the ACL if sequential freshness is being used.

For the AES-CTR security suite, the security material stored in the macDefaultSecurityMaterial or macACLSecurityMaterial fields in the MAC PIB ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are used, this security suite provides sequential freshness on received frames. Table 20 specifies the order and length of the AES-CTR security material components.

Table 20—AES-CTR security material

Octets: 16	**4**	**1**	**(4)**	**(1)**
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

7.6.4.1.2 Protected MAC payload formats

In the AES-CTR security suite, the MAC payload field of a protected frame consists of the frame counter, the key sequence counter and the encrypted MAC payload. Table 21 specifies the order and length of the subfields of the MAC payload of an AES-CTR secured frame. The length of the encrypted MAC payload is equal to the length of the MAC payload before it was encrypted.

Table 21—AES-CTR MAC payload

Octets: 4	**1**	**variable**
Frame counter	Key sequence counter	Encrypted MAC payload

7.6.4.1.3 CTR input blocks

In the AES-CTR security suite, the input blocks to the CTR encryption function for generating the key stream consist of data that is known by each party and data that is transmitted. Table 22 specified the order and length of the subfields of the CTR input blocks.

Table 22—AES-CTR input block

Octets: 1	**8**	**4**	**1**	**2**
Flags	Source address	Frame counter	Key sequence counter	Block counter

The flags octet used in AES-CTR mode is formatted as specified in Table 23.

Table 23—AES-CTR flags field

Bits: 0	**1**	**2-6**	**7**
0	1	0	1

Author's note: The above flags field was defined to be distinct from the AES-CCM flags bytes.

7.6.4.2 Security parameters

The CTR encryption operation as defined in 7.6.3.4 in this security suite shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 7.6.3.7, the counter input blocks shall be formatted as specified in 7.6.4.1.3 where each input block is the same except that the block counter is set to 0 for the first block and is incremented as specified in 7.6.3.3 for each successive input block. In other words, the i^{th} input block T_i shall have the block counter value set to i-1.

7.6.4.3 Security operations

The following sub-clauses specify the operations performed on outgoing and incoming frames.

7.6.4.3.1 Outgoing frame operations

When the AES-CTR security suite is invoked to protect an outgoing frame, the MAC shall perform the following operations.

1) Obtain its own 8-byte IEEE MAC address, the frame counter and the sequence counter from the MAC PIB and construct the counter input blocks as specified in 7.6.4.2.
2) Encrypt the MAC payload in the frame using CTR encryption as specified in 7.6.3.4 with the parameters specified in 7.6.4.2 using the counter input blocks from step 1.
3) Combine the frame counter, sequence counter and output from step 2 as specified in 7.6.4.1.2 to obtain the new MAC payload.
4) Increment the frame counter as specified in 7.6.3.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails, the device shall abort the operation and return an error to the higher layer.

7.6.4.3.2 Incoming frame operations

When the AES-CTR security suite is invoked to protect an incoming frame, the MAC shall perform the following operations.

1) (optional) Verify that the received key sequence counter is greater than or equal to the last known key sequence counter from that device. If the key sequence counter is equal to the last known key sequence counter, verify that the received frame counter is greater than or equal to the last known frame counter from that device. If either of these checks fail, the device shall reject the frame and return an error to the higher layer. If the checks succeed, the last known sequence counter and last known frame counter shall be set to the received values.
2) Obtain the 8-byte IEEE MAC address of the source either from the frame or from the ACL, extract the frame counter and sequence counter from the MAC payload and construct the counter input blocks as specified in 7.6.4.2. If the nonce cannot be constructed due to the data being unavailable, the device shall return an error to the higher layers.
3) Decrypt the encrypted payload in the frame using CTR decryption as specified in 7.6.3.4 with the parameters specified in 7.6.4.2 using the counter input blocks from step 2.
4) Replace the existing MAC payload with the decrypted data from step 3.

7.6.5 AES-CCM-128 security suite

Author's note: For all of the CCM and CBC-MAC security suites, the difference between the different numbers (128, 64 and 32) is simply a change in the length of the integrity code.

The AES-CCM-128 security suite consists of performing AES-CCM authentication (or verification) on the MAC header concatenated with the MAC payload and encryption (or decryption) on the MAC payload using shared data, the frame counter, the key sequence counter, the MAC header and the (encrypted) MAC payload.

The AES-CCM-128 security suite provides the following security services:

— Access control, specified in 7.5.1
— Data encryption, specified in 7.5.2
— Frame integrity, specified in 7.5.3
— Sequential freshness, specified in 7.5.4 (optional)

7.6.5.1 Data formats

The following sub-clauses define the data formats used in this security suite.

7.6.5.1.1 MAC PIB formats

For the AES-CCM-128 security suite, the security material stored in the macDefaultSecurityMaterial or macACLSecurityMaterial fields in the MAC PIB ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are used, this security suite provides sequential freshness on received frames. Table 24 specifies the order and length of the AES-CCM security material components.

Table 24—AES-CCM-128 security material

Octets: 16	**4**	**1**	**(4)**	**(1)**
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

7.6.5.1.2 Protected MAC payload formats

In the AES-CCM-128 security suite, the MAC payload field of a protected frame consists of the frame counter, the key sequence counter, the encrypted MAC payload and the encrypted integrity code. Table 25 specifies the order and length of the subfields of the MAC payload of an AES-CCM-128 secured frame. The length of the encrypted MAC payload is equal to the length of the MAC payload before it was encrypted.

Table 25—AES-CCM-128 MAC payload

Octets: 4	**1**	**variable**	**16**
Frame counter	Key sequence counter	Encrypted MAC payload	Encrypted integrity code

7.6.5.1.3 CCM nonce

In the AES-CCM-128 security suite, the nonce input used for the CCM authentication and encryption function consists of data that is known by each party and data that is transmitted. Table 26 specified the order and length of the subfields of the CCM nonce.

Table 26—AES-CCM-128 nonce

Octets: 8	**4**	**1**
Source address	Frame counter	Key sequence counter

7.6.5.2 Security parameters

In this security suite, the CCM operations as defined in 7.6.3.6 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 7.6.3.7, the length in octets of the length field L shall be 2 bytes, the length of the authentication field M shall be 16 bytes, the nonce shall be formatted as specified in 7.6.5.1.3.

7.6.5.3 Security operations

The following sub-clauses specify the operations performed on outgoing and incoming frames.

7.6.5.3.1 Outgoing frame operations

When the AES-CCM-128 security suite is invoked to protect an outgoing frame, the MAC shall perform the following operations.

1) Obtain its own 8-byte IEEE MAC address, the frame counter and the sequence counter from the MAC PIB and construct the nonce as specified in 7.6.5.1.3.
2) Encrypt and authenticate the MAC header and MAC payload in the frame using CCM authentication and encryption as specified in 7.6.3.6 with the parameters specified in 7.6.5.2 using the MAC header as the authentication data *a*, the MAC payload as the message *m* and the nonce computed in step 1.
3) Combine the frame counter, sequence counter and output from step 2 (including the encrypted MAC payload and encrypted integrity code) as specified in 7.6.5.1.2 to obtain the new MAC payload.
4) Increment the frame counter as specified in 7.6.3.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails, the device shall abort the operation and return an error to the higher layers.

7.6.5.3.2 Incoming frame operations

When the AES-CCM-128 security suite is invoked to protect an incoming frame, the MAC shall perform the following operations.

1) (optional) Verify that the received key sequence counter is greater than or equal to the last known key sequence counter from that device. If the key sequence counter is equal to the last known key sequence counter, verify that the received frame counter is greater than or equal to the last known frame counter from that device. If either of these checks fail, the device shall reject the frame and return an error to the higher layer. If the checks succeed, the last known sequence counter and last known frame counter shall be set to the received values.
2) Obtain the 8-byte IEEE MAC address of the source either from the frame or from the ACL, extract the frame counter and sequence counter from the MAC payload and construct the nonce as specified in 7.6.5.1.3. If the nonce cannot be constructed due to the data being unavailable, the device shall reject the frame and return an error to the higher layers.
3) Decrypt the encrypted payload and verify the integrity code using CCM decryption and verification as specified in 7.6.3.6 with the parameters specified in 7.6.5.2 using the MAC header as the authentication data *a*, the encrypted payload as the message *m* and the nonce computed in step 2. If the integrity code fails, the device shall discard the frame and return an error to the higher layers.
4) Replace the existing MAC payload with the decrypted data from step 3.

7.6.6 AES-CCM-64 security suite

The AES-CCM-64 security suite consists of performing AES-CCM authentication (or verification) on the MAC header concatenated with the MAC payload and encryption (or decryption) on the MAC payload using shared data, the frame counter, the key sequence counter, the MAC header and the (encrypted) MAC payload.

The AES-CCM-64 security suite provides the following security services:

— Access control, specified in 7.5.1

— Data encryption, specified in 7.5.2
— Frame integrity, specified in 7.5.3
— Sequential freshness, specified in 7.5.4 (optional)

7.6.6.1 Data formats

The following sub-clauses define the data formats used in this security suite.

7.6.6.1.1 MAC PIB formats

For the AES-CCM-64 security suite, the security material stored in the macDefaultSecurityMaterial or macACLSecurityMaterial fields in the MAC PIB ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are used, this security suite provides sequential freshness on received frames. Table 27 specifies the order and length of the AES-CCM security material components.

Table 27—AES-CCM-64 security material

Octets: 16	4	1	(4)	(1)
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

7.6.6.1.2 Protected MAC payload formats

In the AES-CCM-64 security suite, the MAC payload field of a protected frame consists of the frame counter, the key sequence counter, the encrypted MAC payload and the encrypted integrity code. Table 28 specifies the order and length of the subfields of the MAC payload of an AES-CCM-128 secured frame. The length of the encrypted MAC payload is equal to the length of the MAC payload before it was encrypted.

Table 28—AES-CCM-64 MAC payload

Octets: 4	1	variable	8
Frame counter	Key sequence counter	Encrypted MAC payload	Encrypted integrity code

7.6.6.1.3 CCM nonce

In the AES-CCM-64 security suite, the nonce input used for the CCM authentication and encryption function consists of data that is known by each party and data that is transmitted. Table 29 specified the order and length of the subfields of the CCM nonce.

Table 29—AES-CCM-64 nonce

Octets: 8	4	1
Source address	Frame counter	Key sequence counter

7.6.6.2 Security parameters

In this security suite, the CCM operations as defined in 7.6.3.6 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 7.6.3.7, the length in octets of the length field L shall be 2 bytes, the length of the authentication field M shall be 8 bytes, the nonce shall be formatted as specified in 7.6.5.1.3.

7.6.6.3 Security operations

The following sub-clauses specify the operations performed on outgoing and incoming frames.

7.6.6.3.1 Outgoing frame operations

When the AES-CCM-64 security suite is invoked to protect an outgoing frame, the MAC shall perform the following operations.

1) Obtain its own 8-byte IEEE MAC address, the frame counter and the sequence counter from the MAC PIB and construct the nonce as specified in 7.6.6.1.3.
2) Encrypt and authenticate the MAC header and MAC payload in the frame using CCM authentication and encryption as specified in 7.6.3.6 with the parameters specified in 7.6.6.2 using the MAC header as the authentication data *a*, the MAC payload as the message *m* and the nonce computed in step 1.
3) Combine the frame counter, sequence counter and output from step 2 (including the encrypted MAC payload and encrypted integrity code) as specified in 7.6.6.1.2 to obtain the new MAC payload.
4) Increment the frame counter as specified in 7.6.3.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails, the device shall abort the operation and return an error to the higher layers.

7.6.6.3.2 Incoming frame operations

When the AES-CCM-64 security suite is invoked to protect an incoming frame, the MAC shall perform the following operations.

1) (optional) Verify that the received key sequence counter is greater than or equal to the last known key sequence counter from that device. If the key sequence counter is equal to the last known key sequence counter, verify that the received frame counter is greater than or equal to the last known frame counter from that device. If either of these checks fail, the device shall reject the frame and return an error to the higher layer. If the checks succeed, the last known sequence counter and last known frame counter shall be set to the received values.
2) Obtain the 8-byte IEEE MAC address of the source either from the frame or from the ACL, extract the frame counter and sequence counter from the MAC payload and construct the nonce as specified in 7.6.6.1.3. If the nonce cannot be constructed due to the data being unavailable, the device shall reject the frame and return an error to the higher layers.
3) Decrypt the encrypted payload and verify the integrity code using CCM decryption and verification as specified in 7.6.3.6 with the parameters specified in 7.6.6.2 using the MAC header as the authentication data *a*, the encrypted payload as the message *m* and the nonce computed in step 1. If the integrity code fails, the device shall discard the frame and return an error to the higher layers.
4) Replace the existing MAC payload with the decrypted data from step 2.

7.6.7 AES-CCM-32 security suite

The AES-CCM-32 security suite consists of performing AES-CCM authentication (or verification) on the MAC header concatenated with the MAC payload and encryption (or decryption) on the MAC payload using shared data, the frame counter, the key sequence counter, the MAC header and the (encrypted) MAC payload.

The AES-CCM-32 security suite provides the following security services:

— Access control, specified in 7.5.1
— Data encryption, specified in 7.5.2
— Frame integrity, specified in 7.5.3
— Sequential freshness, specified in 7.5.4 (optional)

7.6.7.1 Data formats

The following sub-clauses define the data formats used in this security suite.

7.6.7.1.1 MAC PIB formats

For the AES-CCM-32 security suite, the security material stored in the macDefaultSecurityMaterial or macACLSecurityMaterial fields in the MAC PIB ACL consists of a symmetric key, a frame counter and a key sequence counter as well as an optional frame counter and sequence counter that may be used for incoming frames. If these optional fields are used, this security suite provides sequential freshness on received frames. Table 30 specifies the order and length of the AES-CCM security material components.

Table 30—AES-CCM-32 security material

Octets: 16	**4**	**1**	**(4)**	**(1)**
Symmetric key	Frame counter	Key sequence counter	Optional external frame counter	Optional external key sequence counter

7.6.7.1.2 Protected MAC payload formats

In the AES-CCM-32 security suite, the MAC payload field of a protected frame consists of the frame counter, the key sequence counter, the encrypted MAC payload and the encrypted integrity code. Table 31 specifies the order and length of the subfields of the MAC payload of an AES-CCM-32 secured frame. The length of the encrypted MAC payload is equal to the length of the MAC payload before it was encrypted.

Table 31—AES-CCM-32 MAC payload

Octets: 4	**1**	**variable**	**4**
Frame counter	Key sequence counter	Encrypted MAC payload	Encrypted integrity code

7.6.7.1.3 CCM nonce

In the AES-CCM-32 security suite, the nonce input used for the CCM authentication and encryption function consists of data that is known by each party and data that is transmitted. Table 32 specified the order and length of the subfields of the CCM nonce.

Table 32—AES-CCM-32 nonce

Octets: 8	4	1
Source address	Frame counter	Key sequence counter

7.6.7.2 Security parameters

In this security suite, the CCM operations as defined in 7.6.3.6 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 7.6.3.7, the length in octets of the length field L shall be 2 bytes, the length of the authentication field M shall be 4 bytes, the nonce shall be formatted as specified in 7.6.7.1.3.

7.6.7.3 Security operations

The following sub-clauses specify the operations performed on outgoing and incoming frames.

7.6.7.3.1 Outgoing frame operations

When the AES-CCM-32 security suite is invoked to protect an outgoing frame, the MAC shall perform the following operations.

1) Obtain its own 8-byte IEEE MAC address, the frame counter and the sequence counter from the MAC PIB and construct the nonce as specified in 7.6.7.1.3.
2) Encrypt and authenticate the MAC header and MAC payload in the frame using CCM authentication and encryption as specified in 7.6.3.6 with the parameters specified in 7.6.7.2 using the MAC header as the authentication data *a*, the MAC payload as the message *m* and the nonce computed in step 1.
3) Combine the frame counter, sequence counter and output from step 2 (including the encrypted MAC payload and encrypted integrity code) as specified in 7.6.7.1.2 to obtain the new MAC payload.
4) Increment the frame counter as specified in 7.6.3.3 and, if the incrementing succeeds, insert the new counter value into the MAC PIB. If the incrementing operation fails, the device shall abort the operation and return an error to the higher layers.

7.6.7.3.2 Incoming frame operations

When the AES-CCM-32 security suite is invoked to protect an incoming frame, the MAC shall perform the following operations.

1) (optional) Verify that the received key sequence counter is greater than or equal to the last known key sequence counter from that device. If the key sequence counter is equal to the last known key sequence counter, verify that the received frame counter is greater than or equal to the last known frame counter from that device. If either of these checks fail, the device shall

reject the frame and return an error to the higher layer. If the checks succeed, the last known sequence counter and last known frame counter shall be set to the received values.

2) Obtain the 8-byte IEEE MAC address of the source either from the frame or from the ACL, extract the frame counter and sequence counter from the MAC payload and construct the nonce as specified in 7.6.7.1.3. If the nonce cannot be constructed due to the data being unavailable, the device shall reject the frame and return an error to the higher layers.
3) Decrypt the encrypted payload and verify the integrity code using CCM decryption and verification as specified in 7.6.3.6 with the parameters specified in 7.6.7.2 using the MAC header as the authentication data *a*, the encrypted payload as the message *m* and the nonce computed in step 1. If the integrity code fails, the device shall discard the frame and return an error to the higher layers.
4) Replace the existing MAC payload with the decrypted data from step 2.

7.6.8 AES-CBC-MAC-128 security suite

Author's note: There are several possibilities for how to provide integrity only on frames using AES. CBC-MAC with 1-byte prepended length was selected in order to minimize (as well as possible) bandwidth, storage space and computational effort (in that order). There are other variants of CBC-MAC that may be appropriate for use in this context such as XCBC and there are other methods of formatting the input to the CBC-MAC algorithm, such as encoding the length as a 16-byte integer instead of a 1-byte integer. This last possibility might be useful for interoperability with devices that may be authenticating more than 256-bytes of data at a time and it may also be more efficient in software on some processors. On the other hand, the use of a 1-byte length minimizes the extra AES computations (you only have to do an extra AES computation if the length of the data you are protecting is an exact multiple of 16 bytes). Freshness is not provided in this security suite since it would cost extra bandwidth, but it may be provided at the higher layers easily.

The AES-CBC-MAC-128 security suite consists of performing AES-CBC-MAC authentication (or verification) on the encoding of the length of the combined MAC header and MAC payload concatenated with the MAC header concatenated with the MAC payload.

The AES-CBC-MAC-128 security suite provides the following security services:

— Access control, specified in 7.5.1
— Frame integrity, specified in 7.5.3

7.6.8.1 Data formats

The following sub-clauses define the data formats used in this security suite.

7.6.8.1.1 MAC PIB formats

For the AES-CBC-MAC-128 security suite, the security material stored in the macDefaultSecurityMaterial or macACLSecurityMaterial fields in the MAC PIB ACL consists of a symmetric key. No state information is required between frames. Table 33 specifies the order and length of the AES-CBC-MAC security material components.

Table 33—AES-CBC-MAC-128 security material

Octets: 16
Symmetric key

7.6.8.1.2 Protected MAC payload formats

In the AES-CBC-MAC-128 security suite, the MAC payload field of a protected frame consists of the existing MAC payload followed by the integrity code. Table 34 specifies the order and length of the subfields of the MAC payload of an AES-CBC-MAC-128 secured frame.

Table 34—AES-CBC-MAC-128 MAC payload

Octets: variable	16
MAC payload	Integrity code

7.6.8.1.3 CBC-MAC input blocks

In the AES-CBC-MAC-128 security suite, the input to the CBC-MAC authentication function for generating the integrity code consists of the length of the data to authenticate (not including the length field itself) followed by the MAC header followed by the MAC payload. The input is broken up into 16-byte blocks starting from the left and proceeding to the right until the last block, which may be smaller than 16-bytes depending on the length of the total input. Table 35 specified the order and length of the subfields of the CBC-MAC input.

Table 35—AES-CBC-MAC-128 input

Octets: 1	variable = n	variable = m
Length = n + m	MAC header	MAC payload

7.6.8.2 Security parameters

In this security suite, the CBC-MAC operations as defined in 7.6.3.5 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 7.6.3.7, the input to the CBC-MAC function shall be formatted as specified in 7.6.8.1.3, the length of the integrity code M shall be 128 bits.

7.6.8.3 Security operations

The following sub-clauses specify the operations performed on outgoing and incoming frames.

7.6.8.3.1 Outgoing frame operations

Author's note: Since the draft currently specifies that the length of the frame length in the PHY header is only 1 byte, we assume that a 1-byte encoding of the length of the MAC header and payload is sufficient.

When the AES-CBC-MAC-128 security suite is invoked to protect an outgoing frame, the MAC shall perform the following operations.

1) Determine the length in bytes of the MAC header concatenated with the MAC payload (before the security operations are performed) and encode that length as a 1-byte integer as specified in 7.6.3.3.

2) Compute the integrity code on the MAC header and MAC payload in the frame using CBC-MAC authentication as specified in 7.6.3.5 with the parameters specified in 7.6.8.2.
3) Combine the MAC payload and output from step 2 as specified in 7.6.8.1.2 to obtain the new MAC payload.

7.6.8.3.2 Incoming frame operations

Author's note: The length computed in step 1 here should be the same value as the length computed in step 1 of 7.6.8.3.1. In this case, it should be equal to the frame length - 18.

When the AES-CBC-MAC-128 security suite is invoked to protect an incoming frame, the MAC shall perform the following operations.

1) Determine the length in bytes of the MAC header concatenated with the MAC payload before the security operations were applied and encode that length as a 1-byte integer as specified in 7.6.3.3.
2) Parse the MAC payload into the MAC payload and integrity code as specified in 7.6.8.1.2 and verify the integrity code using CBC-MAC verification as specified in 7.6.3.5 with the parameters specified in 7.6.8.2 using the length from step 1, the MAC header from the frame and the MAC payload (without the integrity code) as the input data. If the integrity code fails, the device shall discard the frame and return an error to the higher layers.
3) Remove the integrity code from the MAC payload.

7.6.9 AES-CBC-MAC-64 security suite

The AES-CBC-MAC-64 security suite consists of performing AES-CBC-MAC authentication (or verification) on the encoding of the length of the combined MAC header and MAC payload concatenated with the MAC header concatenated with the MAC payload.

The AES-CBC-MAC-64 security suite provides the following security services:

— Access control, specified in 7.5.1
— Frame integrity, specified in 7.5.3

7.6.9.1 Data formats

The following sub-clauses define the data formats used in this security suite.

7.6.9.1.1 MAC PIB formats

For the AES-CBC-MAC-64 security suite, the security material stored in the macDefaultSecurityMaterial or macACLSecurityMaterial fields in the MAC PIB ACL consists of a symmetric key. No state information is required between frames. Table 36 specifies the order and length of the AES-CBC-MAC security material components.

Table 36—AES-CBC-MAC-64 security material

Octets: 16
Symmetric key

7.6.9.1.2 Protected MAC payload formats

In the AES-CBC-MAC-64 security suite, the MAC payload field of a protected frame consists of the existing MAC payload followed by the integrity code. Table 37 specifies the order and length of the subfields of the MAC payload of an AES-CBC-MAC-64 secured frame.

Table 37—AES-CBC-MAC-64 MAC payload

Octets: variable	8
MAC payload	Integrity code

7.6.9.1.3 CBC-MAC input blocks

In the AES-CBC-MAC-64 security suite, the input to the CBC-MAC authentication function for generating the integrity code consists of the length of the data to authenticate (not including the length field itself) followed by the MAC header followed by the MAC payload. The input is broken up into 16-byte blocks starting from the left and proceeding to the right until the last block, which may be smaller than 16-bytes depending on the length of the total input. Table 38 specified the order and length of the subfields of the CBC-MAC input.

Table 38—AES-CBC-MAC-64 input

Octets: 1	variable = n	variable = m
Length = n + m	MAC header	MAC payload

7.6.9.2 Security parameters

In this security suite, the CBC-MAC operations as defined in 7.6.3.5 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 7.6.3.7, the input to the CBC-MAC function shall be formatted as specified in 7.6.9.1.3, the length of the integrity code M shall be 64 bits.

7.6.9.3 Security operations

The following sub-clauses specify the operations performed on outgoing and incoming frames.

7.6.9.3.1 Outgoing frame operations

Author's note: Since the draft currently specifies that the length of the frame length in the PHY header is only 1 byte, we assume that a 1-byte encoding of the length of the MAC header and payload is sufficient.

When the AES-CBC-MAC-64 security suite is invoked to protect an outgoing frame, the MAC shall perform the following operations.

1) Determine the length in bytes of the MAC header concatenated with the MAC payload (before the security operations are performed) and encode that length as a 1-byte integer as specified in 7.6.3.3.

2) Compute the integrity code on the MAC header and MAC payload in the frame using CBC-MAC authentication as specified in 7.6.3.5 with the parameters specified in 7.6.9.2.
3) Combine the MAC payload and output from step 2 as specified in 7.6.9.1.2 to obtain the new MAC payload.

7.6.9.3.2 Incoming frame operations

Author's note: The length computed in step 1 here should be the same value as the length computed in step 1 of 7.6.9.3.1. In this case, it should be equal to the frame length - 10.

When the AES-CBC-MAC-64 security suite is invoked to protect an incoming frame, the MAC shall perform the following operations.

1) Determine the length in bytes of the MAC header concatenated with the MAC payload before the security operations were applied and encode that length as a 1-byte integer as specified in 7.6.3.3.
2) Parse the MAC payload into the MAC payload and integrity code as specified in 7.6.9.1.2 and verify the integrity code using CBC-MAC verification as specified in 7.6.3.5 with the parameters specified in 7.6.9.2 using the length from step 1, the MAC header from the frame and the MAC payload (without the integrity code) as the input data. If the integrity code fails, the device shall discard the frame and return an error to the higher layers.
3) Remove the integrity code from the MAC payload.

7.6.10 AES-CBC-MAC-32 security suite

The AES-CBC-MAC-32 security suite consists of performing AES-CBC-MAC authentication (or verification) on the encoding of the length of the combined MAC header and MAC payload concatenated with the MAC header concatenated with the MAC payload.

The AES-CBC-MAC-32 security suite provides the following security services:

— Access control, specified in 7.5.1
— Frame integrity, specified in 7.5.3

7.6.10.1 Data formats

The following sub-clauses define the data formats used in this security suite.

7.6.10.1.1 MAC PIB formats

For the AES-CBC-MAC-32 security suite, the security material stored in the macDefaultSecurityMaterial or macACLSecurityMaterial fields in the MAC PIB ACL consists of a symmetric key. No state information is required between frames. Table 39 specifies the order and length of the AES-CBC-MAC security material components.

Table 39—AES-CBC-MAC-32 security material

Octets: 16
Symmetric key

7.6.10.1.2 Protected MAC payload formats

In the AES-CBC-MAC-32 security suite, the MAC payload field of a protected frame consists of the existing MAC payload followed by the integrity code. Table 40 specifies the order and length of the subfields of the MAC payload of an AES-CBC-MAC-32 secured frame.

Table 40—AES-CBC-MAC-32 MAC payload

Octets: variable	4
MAC payload	Integrity code

7.6.10.1.3 CBC-MAC input blocks

In the AES-CBC-MAC-32 security suite, the input to the CBC-MAC authentication function for generating the integrity code consists of the length of the data to authenticate (not including the length field itself) followed by the MAC header followed by the MAC payload. The input is broken up into 16-byte blocks starting from the left and proceeding to the right until the last block, which may be smaller than 16-bytes depending on the length of the total input. Table 41 specified the order and length of the subfields of the CBC-MAC input.

Table 41—AES-CBC-MAC-32 input

Octets: 1	variable = n	variable = m
Length = n + m	MAC header	MAC payload

7.6.10.2 Security parameters

In this security suite, the CBC-MAC operations as defined in 7.6.3.5 shall be parameterized by the following: the underlying block cipher shall be the AES encryption algorithm as specified in 7.6.3.7, the input to the CBC-MAC function shall be formatted as specified in 7.6.10.1.3, the length of the integrity code M shall be 32 bits.

7.6.10.3 Security operations

The following sub-clauses specify the operations performed on outgoing and incoming frames.

7.6.10.3.1 Outgoing frame operations

Author's note: Since the draft currently specifies that the length of the frame length in the PHY header is only 1 byte, we assume that a 1-byte encoding of the length of the MAC header and payload is sufficient.

When the AES-CBC-MAC-32 security suite is invoked to protect an outgoing frame, the MAC shall perform the following operations.

1) Determine the length in bytes of the MAC header concatenated with the MAC payload (before the security operations are performed) and encode that length as a 1-byte integer as specified in 7.6.3.3.

2) Compute the integrity code on the MAC header and MAC payload in the frame using CBC-MAC authentication as specified in 7.6.3.5 with the parameters specified in 7.6.10.2.
3) Combine the MAC payload and output from step 2 as specified in 7.6.10.1.2 to obtain the new MAC payload.

7.6.10.3.2 Incoming frame operations

Author's note: The length computed in step 1 here should be the same value as the length computed in step 1 of 7.6.10.3.1. In this case, it should be equal to the frame length - 6.

When the AES-CBC-MAC-32 security suite is invoked to protect an incoming frame, the MAC shall perform the following operations.

1) Determine the length in bytes of the MAC header concatenated with the MAC payload before the security operations were applied and encode that length as a 1-byte integer as specified in 7.6.3.3.
2) Parse the MAC payload into the MAC payload and integrity code as specified in 7.6.10.1.2 and verify the integrity code using CBC-MAC verification as specified in 7.6.3.5 with the parameters specified in 7.6.10.2 using the length from step 1, the MAC header from the frame and the MAC payload (without the integrity code) as the input data. If the integrity code fails, the device shall discard the frame and return an error to the higher layers.
3) Remove the integrity code from the MAC payload.

Author's note: The following normative annex should be added after Annex B.

Annex A

(normative)

Security Implementation

A.1 Generic CCM mode

CCM is a generic authenticate-and-encrypt 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.

For the generic CCM mode there are two parameter choices to be made. The first choice is *M*, the size of the authentication field. 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 are 4, 6, 8, 10, 12, 14, and 16 octets. The second choice is *L*, the size of the length field. 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 are 2 to 8 octets (the value *L*=1 is reserved).

Table 42: Parameters of CCM mode

Name	Description	Field Size	Encoding of field
M	Number of octets in authentication field	3 bits	(*M*-2)/2
L	Number of octets in length field	3 bits	*L*-1

A.1.1 Inputs

To send a message 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 43: Inputs for CCM

Name	Description	Field Size	Encoding of field
K	Block cipher key	Depends on block cipher	String of octets.
N	Nonce	15-*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 CBC-MAC. We first define a sequence of blocks $B_0, B_1, \ldots B_n$ and then apply CBC-MAC to these blocks.

The first block B_0 is formatted as follows.

Table 44: First authentication block B_0

Octet no:	0	1 ... 15-*L*	16-*L* ... 15
Con-tents:	Flags	Nonce *N*	*l(m)*

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

The Flags field is formatted as

Table 45: Authentication flags octet

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

The Reserved bit is reserved for future expansions and should always be set to zero. The Adata bit is set to zero if *l(a)*=0, and set to one if *l(a)*>0. The *M* field encodes the value of *M* as (*M*-2)/2. As *M* can take on the even values from 4 to 16, the 3-bit field can take on the values from 1 to 7. The *L* field encodes the size of the length field used to store *l(m)*. The parameter *L* can take on the values from 2 to 8 (the value L=1 is reserved). This value is encoded in the 3-bit field using the values from 1 to 7 by choosing the field value as *L*-1 (the zero value is reserved).

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 $l(a)$ in most-significant-byte first order.

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

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

This is summarized in the following table. Note that all fields are interpreted in most-significant-byte first order.

Table 46: 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 $2^{16} - 2^8 \le l(a) < 2^{32}$
0xFFFF	eight octets $l(a)$	For $2^{32} \le l(a) < 2^{64}$

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

After the (optional) additional authentication blocks have been added, we add the message blocks. The message blocks are formed by splitting the message m into 16-octet blocks, padding the last block with zeroes if necessary. If the message m consists of the empty string, then no blocks are added in this step.

The result is a sequence of blocks $B_0, B_1, ..., B_n$. The CBC-MAC is now computed by:

$$X_1 := \mathrm{E}(K, B_0)$$

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

$$T := \text{first-}M\text{-bytes}(X_{n+1})$$

where E() is the block cipher encryption function and T is the MAC value. Note that the last block B_n is XORed with X_n and encrypted with the block cipher to give T.

A.1.3 Encryption

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

$$S_i := \mathrm{E}(K, A_i)$$

for i=0, 1, 2, ...

The values A_i are formatted as

Table 47: Encryption blocks A_i

Octet no:	0	1 ... 15-*L*	16-*L* ... 15
Con-tents:	Flags	Nonce *N*	Counter *i*

where *i* is encoded in most-significant-byte first order.

The Flags field is formatted as

Table 48: Encryption flags octet

Bit no:	7	6	5	4	3	2	1	0
Con-tents:	Reserved	Reserved	0			*L*		

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 message is encrypted by XORing the octets of message *m* with the first *l(m)* octets of the concatenation of S_1, S_2, S_3, ... Note that S_0 is not used to encrypt the message.

The authentication value *U* is computed by encrypting *T* with the key stream block S_0 and truncating it to the desired length.

$$U := T \oplus \text{first-}M\text{-bytes}(S_0)$$

A.1.4 Output

The final result *c* consists of the encrypted message *m*, followed by 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 MAC value *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 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 49: 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	15-*L* octets	Nonce should never be repeated for same key.

Table 49: List of symbols

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

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, … , T_n. Given a sequence of counters, T_1, T_2, … , 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 $j = 1, 2 \ldots n\text{-}1$;
$C^*_n = P^*_n \oplus MSB_u(O_n)$.

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.

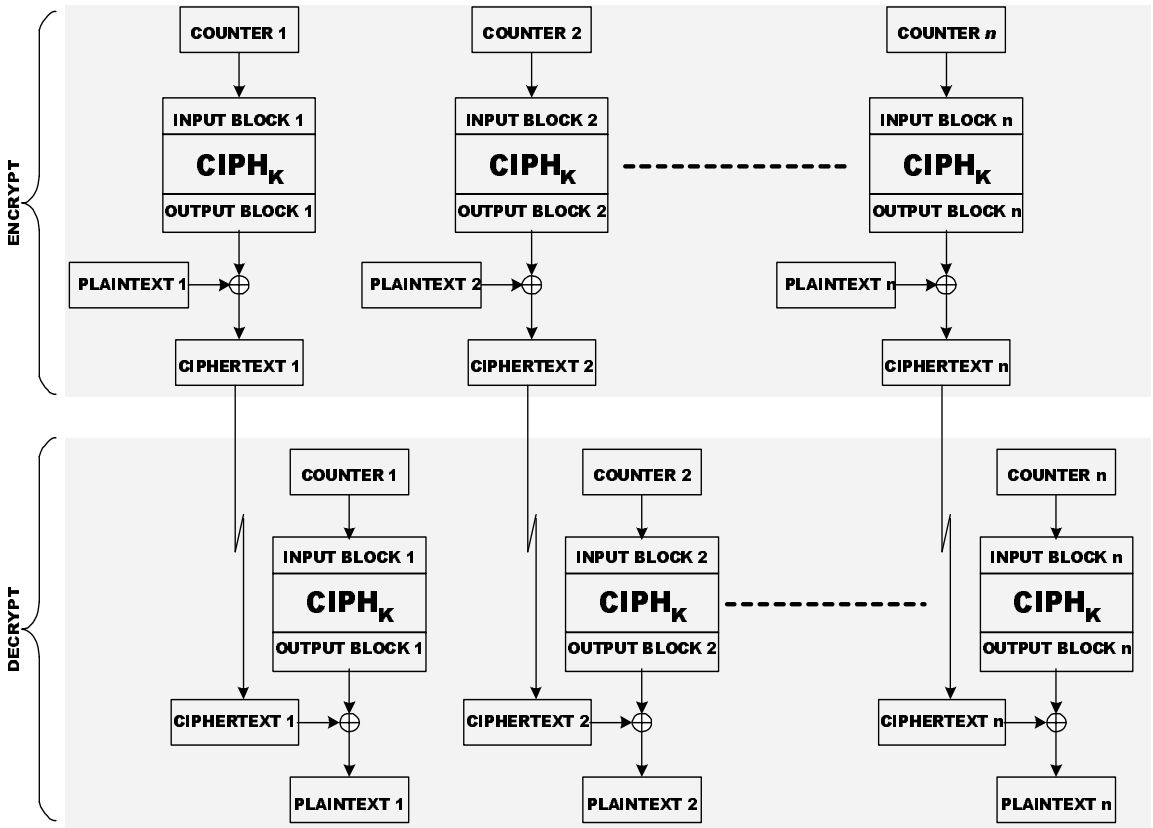

Figure 1—The CTR Mode

The CTR mode is illustrated in Figure 1.

A.3 CBC-MAC

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 $\oplus$ represents the Exclusive-OR of two vectors.

$$O_1 = e(D_1)$$

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

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

. . .

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

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 generation is given in Figure 2.

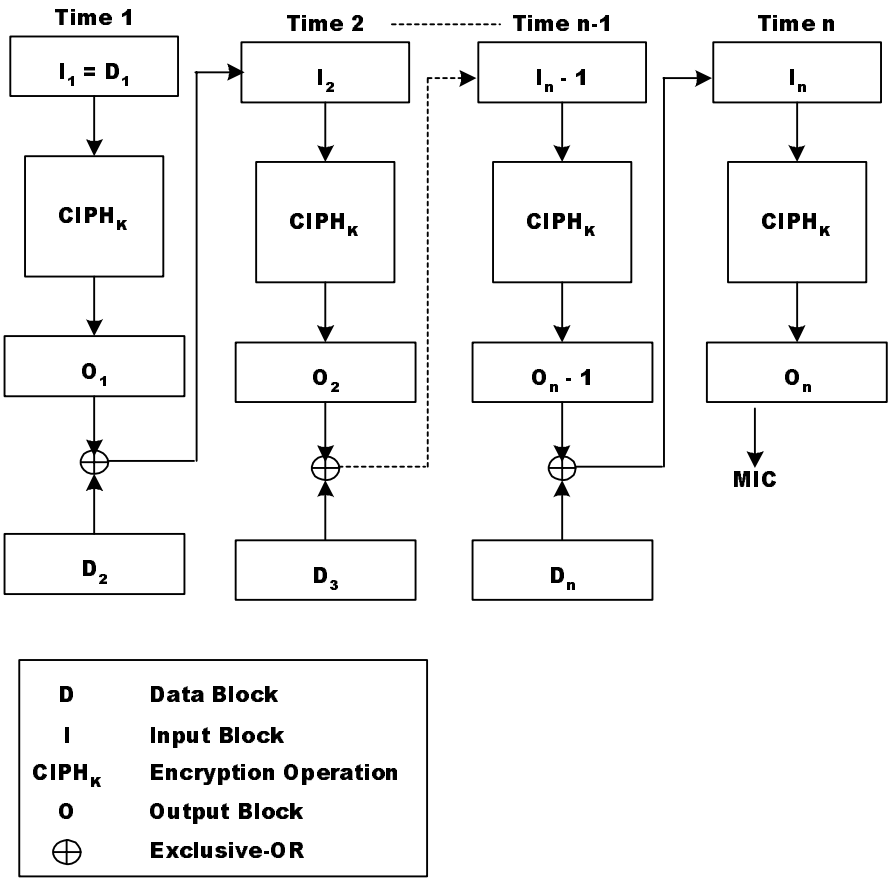

Figure 2—CBC-MAC