IEEE P802.15 Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)		
Title	Removing public key suites		
Date Submitted	[January, 2003]		
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:	802.15.3 TG3 Draft D15 for sponsor ballot		
Abstract	[This document is offered as a set of comments and recommended changes to 802.15.3 D15.]		
Purpose	[This document is intended as support for sponsor ballot comments on 802.15.3 D15.]		
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. Scope

This document outlines the changes necessary to remove the public-key suites from 802.15.3.

1.1 Purpose

This document is intended as supporting documentation for sponsor ballot comments on 802.15.3 draft D15. If the working group accepts these ballot comments, text from this document may be incorporated directly into the draft.

1.2 Notes to the Reader

Throughout the document, the author has included notes to the reader that may clarify items, but are not part of the proposed text for amending the draft.

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

Author's note: Many of the diagrams weren't available for the preparation of this document. Since no changes are proposed to the diagrams, those diagrams that should remain in the document appear as grey boxes.

2. Document changes

2.1 Clause 2

Remove the following references:

1) ANSI X9.62-1998
2) ANSI X9.63-2001
3) Efficient Embedded Security Standard (EESS) #1
4) IEEE Std 1363-2000
5) IETF RFC 2459
6) IETF RFC 3279
7) IETF RFC 3280
8) ITU-T Recommendation X.509
9) NIST FIPS 180-2
10) FIPS Pub 186-2
11) FIPS Pub 198
12) PKCS#1 v2.1
13) SEC 1

2.2 Clause 3

Remove the following definitions:

1) certificate authority
2) certificate revocation
3) digital signature

4) private key
5) public key
6) public-key certificate
7) public-key pair
8) signature verification
9) signed data

2.3 Clause 4

Remove the following acronyms and abbreviations:

1) CA
2) EC
3) ECC
4) ECMQV
5) ECQV

2.4 Clause 5

Replace 5.3.4 with:

Some 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 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 if security is to be used at the MAC sublayer. The security mechanisms in this standard provide data encryption, data integrity and symmetric-key update services using keys provided by higher layer processes. The authentication and authorization of devices as well as initial establishment of keying material is the responsibility of the implementer. The security provided by these mechanisms assume the keys are generated, transmitted, and stored in a secure manner. Recommended practices established by recognized industry organizations (e.g., WiMedia Alliance) should be used.

2.5 Clause 6

Update 6.3.7.1 to replace the MLME-AUTHENTICATE.request primitive with:

```
MLME-AUTHENTICATE.request (
                        TrgtID,
                        OIDLength,
                        OID,
                        AuthenticationDataLength,
                        AuthenticationData,
                        AuthTimeout
                        )
```

Replace 6.3.7.1.1 with:

This primitive is generated by the DME for a DEV to perform authentication with another DEV or the PNC.

Update Table 11 to remove the following items:

1) PublicKeyObjectType
2) PublicKeyObjectLength
3) PublicKeyObject
4) AuthResponse
5) Challenge
6) ChallengeResponse
7) SecurityRequirements

Update Table 11 to add the following items:

Table 1—MLME-AUTHENTICATION and MLME-CHALLENGE primitive parameters

AuthenticationDataLength	Integer	Defined by the security suite, Clause 10 {xref}	Specifies the length in octets of the AuthenticationData. The length shall be such that neither the association request nor challenge request commands exceed pMax-FrameSize
AuthenticationData	Octet String	Defined by the security suite, Clause 10 {xref}	Specifies the authentication data in the format as specified by the security suite.
ChallengeDataLength	Integer	Defined by the security suite, Clause 10 {xref}	Specifies the length in octets of the ChallengeData. The length shall be such that neither the association request nor challenge request commands exceed pMaxFrameSize
ChallengeData	Octet String	Defined by the security suite, Clause 10 {xref}	Specifies the challenge data in the format as specified by the security suite.

Update 6.3.7.1 to replace the MLME-AUTHENTICATE.indication primitive with:

```
MLME-AUTHENTICATE.request (
                        OrigID,
                        OIDLength,
                        OID,
                        AuthenticationDataLength,
                        AuthenticationData,
                        )
```

Replace 6.3.7.2.2 with:

The DME is informed of the request for authentication by an associated DEV.

Update 6.3.7.1 to replace the MLME-AUTHENTICATE.response primitive with:

```
MLME-AUTHENTICATE.response (
                    OrigID,
                    AuthenticationDataLength,
                    AuthenticationData,
                    ReasonCode,
                    )
```

Replace 6.3.7.3.2 with:

The MLME sends an authentication response command, 7.5.2.2 {xref}, to the indicated DEV.

Update 6.3.7.4 to replace the MLME-AUTHENTICATE.confirm primitive with:

```
MLME-AUTHENTICATE.confirm (
                    AuthenticationDataLength,
                    AuthenticationData,
                    ReasonCode,
                    ResultCode
                    )
```

Replace 6.3.7.4.1 with:

This primitive is generated as a result of the MLME receiving an authentication response command, {xref}, or due to a timeout. If there is no response from the security manager within AuthTimeout, the ResultCode shall be set to TIMEOUT. Otherwise, the ResultCode shall be set to SUCCESS and the ReasonCode is the value that was returned in the authentication response command as defined in 7.5.2.2 {xref}.

Replace 6.3.7.5 with:

This primitive is used to initiate an authentication challenge to another DEV during an authentication procedure. The semantics of this primitive are:

```
MLME-CHALLENGE.request    (
                    TrgtID,
                    AuthenticationDataLength,
                    AuthenticationData,
                    AuthTimeout
                    )
```

The primitive parameters are defined in Table 11.

Replace 6.3.7.5.1 with:

This primitive is generated by the DME for a DEV to either perform an authentication challenge with another DEV or perform an authentication challenge with the PNC.

Update 6.3.7.6 to replace the MLME-CHALLENGE.indication primitive with:

```
MLME-CHALLENGE.indication (
                        OrigID,
                        ReasonCode,
                        AuthenticationDataLength,
                        AuthenticationData
                        )
```

Replace 6.3.7.6.2 with:

The DME is notified of the authentication challenge from an associated DEV.

Update 6.3.7.7 to replace the MLME-CHALLENGE.response primitive with:

```
MLME-CHALLENGE.response  (
                        OrigID,
                        ReasonCode,
                        AuthenticationDataLength,
                        AuthenticationData
                        )
```

Update 6.3.7.8 to replace the MLME-CHALLENGE.confirm primitive with:

```
MLME-CHALLENGE.confirm   (
                        TrgtID,
                        AuthenticationDataLength,
                        AuthenticationData,
                        ReasonCode,
                        ResultCode
                        )
```

Replace 6.3.7.8.2 with:

The DME is notified of the received challenge response command.

Update Table 35 to remove the MACPIB_AuthenticateResultCode item.

2.6 Clause 7

Replace 7.2.7.1 with:

The SECID field shall be included in the frame body of all secure frames. The SECID field contains a 2-octet identifier for the key that is being used to protect the frame. The SECID for a given key may be com-

municated to the MAC in the MLME-SECID-UPDATE.request primitive, 6.3.11.1 {xref}, in a distribute key request command, 7.5.2.7 {xref}, or a request key response command, 7.5.2.6 {xref}.

Update Table 51 to remove the Public-key object entry.

Remove 7.4.16.

Replace 7.5.2.1 with:

This command is used to request authentication with a security manager, which may be the PNC or another DEV. The SEC field in the frame control field shall be set to 0. The authentication request command shall be formatted as illustrated in Figure 1 {xref}.

octets: L_n	**2**	**L_m**	**1**	**2**	**2**
Authentication data	Authentication data length	OID	OID Length	Length (=7+L_m 1+ L_n)	Command type

Figure 1—Authentication request command format

The authentication data field is specified by the security suite, Clause 10 {xref}.

Replace 7.5.2.2 with:

The authentication response command is used by the security manager to respond to an authentication request command from the DEV. The SEC field in the frame control field shall be set to 0. The authentication response command shall be formatted as illustrated in Figure 2 {xref}.

octets: L_n	**2**	**1**	**2**	**2**
Authentication data	Authentication data length	Reason code	Length (=2+L_n)	Command type

Figure 2—Authentication response command format

The reason code is used to indicate the result of authentication process. The valid reason code values are:

0 -> Success
1 -> Failure
2-> OID not accepted
3->Timed out
4-> DEV not a security manager
5-255 -> Reserved

Replace 7.5.2.3 with:

The challenge request command is used by an associated DEV to send a cryptographic challenge to another DEV or the PNC. The SEC field in the frame control field shall be set to 0. The challenge request command shall be formatted as illustrated in Figure 3 {xref}.

octets: L_n	**2**	**2**	**2**
Challenge data	Challenge data length	Length (=2+L_n)	Command type

Figure 3—Challenge request command format

The authentication data field is specified by the security suite, Clause 10 {xref}.

Replace 7.5.2.4 with:

The challenge response command is used by an associated DEV to respond to a challenge sent by another DEV. The SEC field in the frame control field shall be set to 0. The challenge response command shall be formatted as illustrated in Figure 4 {xref}.

.

octets: L_n	**2**	**1**	**2**	**2**
Challenge data	Challenge data length	Reason code	Length (=2+L_n)	Command type

Figure 4—Challenge response command format

The reason code is used to indicate the result of the preceding challenge request command. The valid reason code values are:

0 -> Success
1 -> Challenge failure
2-> Timed out
3-255 -> Reserved

The challenge data field is specified by the security suite, Clause 10.

Replace 7.5.4.3 with:

The ACL information request command enables a DEV to request authentication information regarding a single DEV or all DEVs. The ACL information request command shall be formatted as illustrated in Figure 5 {xref}.

octets: 1	**2**	**2**
Queried DEVID	Length (=1)	Command type

Figure 5—ACL information request command format

The queried DEVID indicates the DEV whose ACL information is being requested. If the value of this field is the BcstID, then the DEV is requesting all of the ACL information maintained by the target DEV.

Replace 7.5.4.4 with:

The ACL information command shall be formatted as illustrated in Figure 6 {xref}.

octets: L_m	**---**	**L_2**	**L_1**	**1**	**1**	**2**	**2**
DEV-m ACL record	…	DEV-2 ACL record	DEV-1 ACL record	Sequence number	Total number of frames	Length (=2+L_1 +L_2+…+L_m)	Command type

Figure 6—ACL information command

The total number of frames field indicates the number of frames that will be sent to complete this request.

The sequence number field indicates which frame in the sequence is in this command.

The ACL record field shall be formatted as illustrated in Figure 7 {xref}.

octets: L_n	**2**	**L_m**	**1**	**1**	**6**	**2**
Verification info	Verification info length (=L_n)	OID	OID Length (=L_m)	DEVID	DEV address	Length (=10+L_n)

Figure 7—Format of an ACL record in an ACL information command

The DEV address is the address of the DEV, 7.1, corresponding to the DEVID.

The DEVID is the ID assigned to the DEV by the PNC. If the DEV is not currently associated in this piconet, the field shall be set to the UnassocID. This field shall not contain the broadcast or multicast DEVIDs.

The verification info length indicates the length of the verification information that is included in the ACL entry. If this length is zero, no verification information field shall be included.

The verification info type indicates the type of verification information that is included in the transmitted ACL entry. The verification info types are security suite specific. The definition and format of the valid verification information elements are defined by the security suites, Clause 10.

Update Table 57 to remove Public-key object.

Update Table 58 to remove Public-key object.

Replace 9.1.2 with:

An authentication protocol is the initial source of all cryptographic protection within a piconet. This protocol may be used for either DEV-PNC mutual authentication (for joining the piconet) or for peer-to-peer mutual authentication for peer-to-peer communications.

Authentication between the DEV and PNC is used to provide evidence to the PNC that the DEV is authorized to join the secure piconet and that the DEV accepts the PNC. Authentication between two DEVs is used to provide evidence to each DEV that the other is authorized to establish a secure peer-to-peer relationship with that DEV.

This standard provides MAC frames with fields that other standards may specify for use in an authentication protocol. Authentication methods are not specified in this standard.

Remove 9.1.3.

Replace 9.2.6 with:

If a DEV wishes to join a secure piconet, it should associate with the PNC in order to be assigned a local DEVID and CTAs to perform the authentication process. Since the device shall be associated before the authentication process has taken place, the association command and response have the SEC field in the frame control field set to 0 and use the non-secure command frame format, 7.3.3.1. Once the DEV is associated, the PNC shall allocate an MCTA if commands are not allowed in the CAP, to allow the DEV to proceed with the authentication protocol. Before the authentication process is initiated, the DEV or PNC may choose to send probe commands to each other to request or transmit preferred OIDs. The DEV and PNC may also exchange additional information before authentication if desired. After the DEV has associated and exchanged the desired information with the PNC, the DEV should initiate the authentication protocol. The authentication and challenge commands are designed to be used with security turned off. In the authentication request command, the DEV should select either the security suite OID received in the association response or an OID received in a probe command after associating. While in the authentication process, the authentication commands shall have the SEC field in the frame control field set to 0. If during the authentication process there is a security check failure of any kind, the DEV or PNC shall return the appropriate error in the challenge response command or authentication response command respectively and exit from the authentication protocol.

Update Table 66 to add an entry for ACL information with the “piconet group data key” and “peer-to-peer data key” boxes checked and comment: “ACL information commands between devices that share a peer-to-peer key shall use the peer-to-peer data key, otherwise they shall use the piconet group data key.”

Replace 9.3 with:

The security mode indicates whether a DEV is currently implementing frame protection in the piconet. The security mode in use is determined by the MACPIB_SecurityOptionImplemented entry in the MAC PIB.

Replace 9.3.2 with:

Security mode 1 provides a mechanism for a device to perform cryptographic security on frames transmitted in the piconet. DEVs operating in security mode 1 use an authentication protocol verify the authenticity of other DEVs in the piconet and symmetric-key cryptography to protect frames using encryption and integrity.

In order to properly operate in mode 1, DEVs should maintain an ACL that contains information about which devices are authorized to authenticate. The manner in which the ACL is used depends on the application and the security suite in use. For example, some security suites require the use of digital certificates to authorize a device to authenticate. Other security suites may use different methods to authorize a device to authenticate.

While in mode 1, the cryptographic operations used for secure frames exchanged with the PNC and with other members of the piconet security group shall be performed as specified by the security suite in use for this relationship. The cryptographic operations performed for secure frames exchanged with a peer DEV shall be performed as specified by the security suite associated with that peer security relationship. While in this mode, if the MAC receives a frame with the SEC field in the frame control field set to a value different than expected as defined in Table 53, the MLME shall generate an MLME-SECURITY-ERROR.indication with the ReasonCode set to INVALID-SEC-VALUE

Update 9.3.2.2 to remove mutual authentication, verifying authenticity of public keys, and key establishment.

Replace 9.4 with:

A security suite, identified by a unique OID, defines mechanisms that may be used to perform the authentication process. After two DEVs perform the authentication process using any security suite, the two DEVs share keying material for use in the symmetric operations defined in 10.2.4. The PNC may perform the authentication process using different security suites with different devices in the same piconet, since the resulting keying material will be of the same form in all cases.

A security suite shall also specify each of the following:

Replace 9.4.1 with:

Each security suite shall have a globally unique OID associated with it that will never change.

Remove 9.4.4 and **9.4.5.**

Replace 9.5 with:

The protocols in this clause are defined for implementing a secure piconet. The device states described in this clause specify the behavior for each DEV and for the PNC for performing the following security protocols:

— Key distribution
— Key request
— Beacon protection
— Data protection

Each of the protocols described in this section may be performed between the PNC and a DEV or between two DEVs in a peer-to-peer relationship. If they are performed in the peer-to-peer setting, the originating DEV shall act in the role of DEV and the target DEV shall act in the role of security manager for the duration of the security relationship.

The protocols in this clause are algorithm independent and may be used for any security suite.

Replace 9.6 with:

9.6.1 Key distribution protocol

The security manager may need to update the payload protection keys periodically for security reasons. This may be due to a change in the group membership or some other reason that is implementation specific. When this occurs, the security manager shall send the new key to each authorized DEV using the shared

secret key agreed upon in the authentication protocol. The PNC needs to store the symmetric keys shared with each DEV in the piconet in order to distribute a new key, but it does not need to store the public keys of these DEVs. Each DEV (other than the PNC) only needs to store the symmetric keys it shares with the PNC and with any DEV with which it has a secure peer-to-peer relationship. An overview of the key distribution protocol is illustrated in Figure 8.

Figure 8—Key distribution protocol overview

9.6.2 Key request protocol

When the security manager updates the piconet protection key, the DEV may not have received the new key properly when it wants to start sending or receiving data. When this occurs, the DEV may request the current key from the security manager. An overview of the key request protocol is illustrated in Figure 9.

Figure 9—Key request protocol overview

9.6.3 Beacon and command protection protocol

The beacon and command protection protocol provides integrity protection and source authentication on the beacon from the PNC and on all commands that are protected. Each beacon and command that is protected has a header that includes an identifier of the key that is being used (the SECID), the source and destination

addresses, the frame data and an integrity code on the data using the shared key(s). An overview of the beacon and command protection protocol is illustrated in Figure 10

Figure 10—Beacon and command protection protocol overview

9.6.4 Data protection protocol

Data in the piconet is protected by payload protection keys that provide both privacy, integrity, and source authentication. Each data message that is protected has a header that includes an identifier of the key(s) being used, the source and destination addresses, the data encrypted using the shared key(s) and an integrity code using the shared key(s).

Figure 11—Data protection protocol overview

Update 9.7 to remove the following items:

1) AD_D
2) AD_SM
3) PKObj_D
4) Pub_key_D
5) Pr_key_D
6) PKObj_SM
7) Pub_key_SM
8) Pr_key_SM
9) CHAL1
10) CHAL2
11) AReq
12) CReq
13) CRes
14) ARes
15) SRF
16) data1
17) data2

18) ChalRes
19) AuthRes

Replace 9.8 with:

The following protocol details include all cryptographic components and headers for the frames. The headers should be interpreted as being headers in the MAC frames. In addition, each element should be interpreted as including type and length fields when appropriate as specified in Clause 7. The algorithm choices for each operation in the protocols are determined by the selected security suite. Note that all frames transmitted in this sub-clause are sent with immediate ACK unless specified otherwise. The acknowledgement frames do not affect the security of the protocols and are omitted from all diagrams except message sequence charts.

For each protocol described in this sub-clause, tables are included to specify the requirements for the DEV and security manager to successfully implement the protocol. The setup table specifies the required data that shall be stored by each device, denoted the initial owner, before the protocol is initiated. The capabilities table specifies the required functionality for each device to perform its respective role in the protocol.

9.8.1 State machines

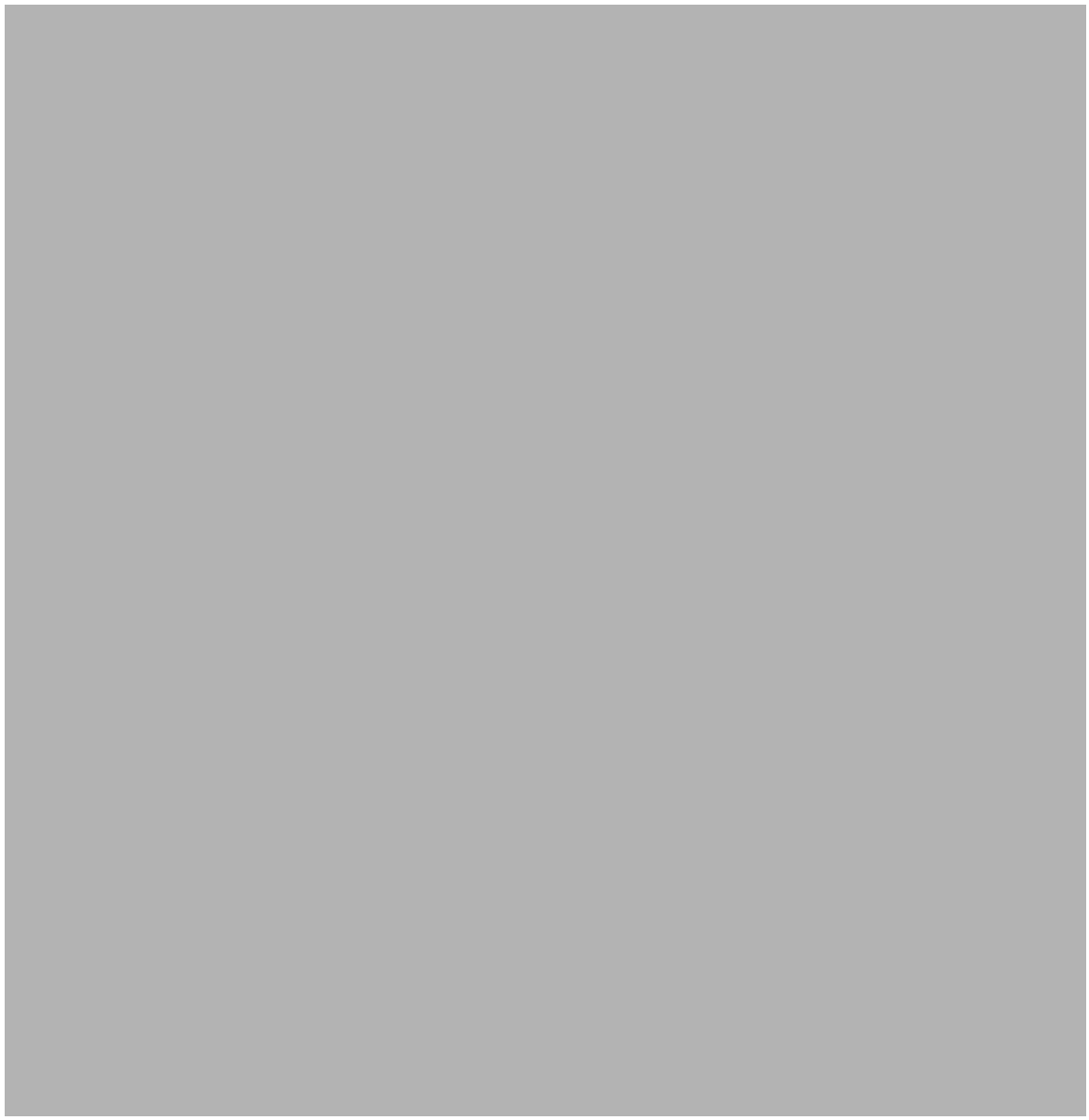
Devices shall maintain security states for each secure relationship they are willing to participate in (where the PNC is considered to be separate from the DEV that is performing the PNC function). The security states are conceptually maintained in security state machines that process incoming frames and generate outgoing frames based on their current state. The states described refer to a single relationship, rather than an overall state for the DEV. Therefore it is assumed that if both peer-to-peer relationships and piconet-wide relationships are used, DEVs may either behave in a multi-threaded fashion or maintain the different states through information stored in the MAC PIB. For each relationship, there is a set of three state machines; one state machine that controls the authentication operations, another state machine that controls the key management operations and a third that controls the processing of other secure frames.

9.8.2 Security state controller

There is conceptually a central entity that is part of the MLME called the security state controller that controls the processing of incoming frames. The security state controller directs incoming frames to the appropriate state machine, receives indications from the state machines about changes in authentication status and communicates the authentication status information between the state machines when necessary.

A device may act as the security manager for some relationships and as the DEV for others. If the DEV is the PNC, the DEV shall act as the security manager for all of the piconet wide security relationships. If the DEV is not the PNC, the DEV shall have only one piconet wide security relationship in which the PNC serves the role of security manager and the DEV serves the role of device in the relationship. Figure 12 shows the interfaces between the security state controller and the state machines.

Figure 12—Security state controller



9.8.2.1 Managing states for incoming frames

The security state controller shall select the state machine to pass a frame to based on the SECID, the SrcID and the frame type indicated in the frame. The security state controller shall check the first octet of the SECID in the frame, if present, to determine whether the role of the DEV is the security manager role or device role. The security state controller shall then check the frame type to determine which type of state machine to direct the frame to and check the SrcID to determine which relationship the frame corresponds to. Table 2 summarizes this process and specifies the frame types, the state machine that shall process those

frame types and the action that shall be taken by the security state controller upon receipt of a non-ACK frame. ACK frames are not considered in the security states.

Table 2—Security state controller frame processing

Incoming frame type	State machine type	Action
Association response command, authentication response command or challenge response command	Authentication state machine - device role	The security state controller shall pass these commands to the authentication state machine with the SM role corresponding to the SrcID in the frame.
Association request command, authentication request command or challenge request command	Authentication state machine - SM role	The security state controller shall pass these commands to the authentication state machine with the device role corresponding to the SrcID in the frame.
Secure disassociate command or deauthenticate command	Key management state machine - either role	The security state controller shall pass these commands to the key management state machine with the SM role corresponding to the first byte of the SECID in the frame. If the first byte of the SECID is the SrcID or the broadcast address, the device role shall correspond to the DEV itself. If the SM is the DEV, the device role shall correspond to the SrcID in the frame. If the SECID
Request key response command and Distribute key command	Key management state machine - device role	The security state controller shall pass these commands to the key management state machine with the SM role corresponding to the SrcID in the frame.
Request key command and distribute key response command	Key management state machine - SM role	The security state controller shall pass these commands to the key management state machine with the device role corresponding to the SrcID in the frame.
Secure beacon frames	Secure frame processing state machine- piconet-wide device role	The security state controller shall pass beacons to the secure frame processing state machine that corresponds to the relationship with the PNC. The device shall process beacons as specified in 9.2.8
All other secure commands and secure data frames	Secure frame processing state machine- either role	The security state controller shall pass all other secure commands and all secure data frames to the secure frame processing state machine with the SM role corresponding to the first byte of the SECID in the frame. If the first byte of the SECID is the SrcID or the broadcast address, the device role shall correspond to the DEV itself. If the SM is the DEV, the device role shall correspond to the SrcID in the frame.

Table 2—Security state controller frame processing

Incoming frame type	State machine type	Action
All other non-secure commands	Secure frame processing state machine- either role	Upon receipt of a non-secure command not specified elsewhere in this table, the security state controller shall determine if it is authenticated as a device or security manager with the DEV indicated by the SrcID in the frame. If it is authenticated, it shall pass the frame to a secure frame processing state machine that is authenticated with the DEV. If it is not-authenticated, it may pass the frame to the secure frame processing state machine for either role.
All non-secure data frames	Secure frame processing state machine- either role	Upon receipt of a non-secure data frame, the security state controller shall determine if the DEV is authenticated as a device or security manager with the DEV indicated by the SrcID in the frame. If it is authenticated, it shall pass the frame to a secure frame processing state machine that is authenticated with the DEV. If it is not-authenticated, it may pass the frame to the secure frame processing state machine for either role.
Non-secure beacon frames	Secure frame processing state machine- device role	Upon receipt of a non-secure beacon frame, the security state controller shall pass the beacon frame to the piconet-wide secure frame processing state machine in the device role.

9.8.2.2 Managing security related events

Certain events such as a change in authentication state and problems with beacon reception affect the way in which a DEV processes incoming frames. When a DEV transitions from being unauthenticated to authenticated or vice-versa, the DEV shall ensure that the keying material is updated accordingly and that all of the related state machines reflect the change in authentication status. When a DEV receives a beacon with a different SECID than expected, the DEV shall ensure that the correct keys are used to process incoming frames and request the new key.

There are four different types of events that cause a change in authentication status: successful authentication, successful secure disassociation, succesful de-authentication and PNC handover. Table 3 indicates the cause for the changes in authentication status or receipt of a beacon with an unexpected SECID or a PNC handover IE and the response of the security state controller upon receiving an indication of the change. The processes within the state machines requiring the change in status and the processes performed by the state

machines upon receipt of an indication of a change in authentication status is specified in the sub-clauses describing the state machines.

Table 3—Authentication status changes

Event	Cause	Action
Authentication	Successful completion of an authentication protocol.	The authentication state machine for a particular security relationship indicates to the security state controller that an authentication protocol has been completed. The security state controller shall indicate to the corresponding key management state machine and secure frame processing state machine that the authentication has occurred.
Secure disassociation	Successful receipt or transmission of a valid secure disassociation command.	The key management state machine for a piconet-wide security relationship indicates to the security state controller that the device has been disassociated. The security state controller shall indicate to the corresponding secure frame processing state machine that the disassociation has occurred.
Deauthentication	Successful receipt or transmission of a valid secure disassociation command	The key management state machine for a peer-to-peer security relationship indicates to the security state controller that the device has been de-authenticated. The security state controller shall indicate to the corresponding secure frame processing state machine that the de-authentication has occurred.
PNC Handover	Successful reception of a beacon with the PNC handover IE and timeout	The key management state machine for a piconet-wide security relationship indicates to the security state controller that the PNC has changed. The security state controller shall indicate to all of the piconet-wide key management state machines and secure frame processing state machines that the PNC has changed.
Unknown SECID in the beacon	Successful reception of a beacon with an unknown SECID	The secure frame processing state machine for the piconet-wide security relationship for the device role indicates to the security state controller that the received beacon had an unknown SECID. The security state controller shall indicate to the piconet-wide key management state machine that the SECID in the beacon was unknown.

Table 3—Authentication status changes

Event	Cause	Action
Unexpected valid beacon	Successful reception of a beacon with a valid SECID and integrity code after having received a beacon with an unknown SECID.	The secure frame processing state machine for the piconet-wide security relationship for the device role indicates to the security state controller that the received beacon had a known SECID. The security state controller shall indicate to the piconet-wide key management state machine that the SECID in the beacon was the known SECID.
PNC handover IE received in beacon	Successful reception of a beacon with a valid PNC handover IE and integrity code .	The secure frame processing state machine for the piconet-wide security relationship for the device role indicates to the security state controller that the received beacon had a PNC handover IE. The security state controller shall indicate to the piconet-wide key management state machine that the device has received a PNC handover IE in the beacon.

9.8.3 ACL information request and distribution

In order to facilitate the authentication process, a DEV may request or send ACL information to another DEV. This most often is done directly before or during the PNC handover process, but may be done at any time.

Figure 13 illustrates the message flows for ACL information request between two peer DEVs.

Figure 13—Message sequence chart for DEV-DEV ACL information request

Figure 14 illustrates the message flows for an ACL information request from the new PNC to the old PNC.

Figure 14—Message sequence chart for New PNC-Old PNC ACL information request

9.8.4 Beacon protection

In a secure piconet, the security manager shall use the current group key(s) to provide integrity protection on the beacon. In addition, each beacon includes a strictly increasing time token that is used by the DEVs to guarantee freshness, 7.3.1.2. The beacon protection protocol described here is also used for command protection. When generating secure beacons, the PNC shall follow the procedures specified in 9.2.7. When processing beacons in a secure piconet, the DEV shall follow the procedures specified in 9.2.8. The initial setup required for this protocol is listed in Table 4.

Table 4—Setup for beacon protection

Symbol	Initial owner	
	DEV	Security manager
Sym_keys_G	✓	✓
SECID_G	✓	✓
Time token	_	✓
SFC_SM	_	✓

The cryptographic functionality required to implement this protocol is described in Table 5.

Table 5—Capabilities for beacon protection

Functionality	Required	
	DEV	Security manager
Symmetric integrity code	✓	✓

The security manager should initiate this protocol each time it transmits a beacon. The beacon protection protocol is illustrated in Figure 15.

Figure 15—Beacon protection protocol

9.8.5 Command protection

In a secure piconet, DEVs use shared keys to provide integrity protection on commands to each other. The command protection protocol applies to all secure commands that do not include an encrypted key. The key used to protect the command depends on the source, destination and command type, see Table 66. The time token from that superframe is used to provide freshness protection on the commands. The command protection protocol described here is the same protocol used for beacon protection. When generating secure commands, DEVs shall follow the procedures specified in 9.2.7. When processing secure commands, DEVs shall follow the procedures specified in 9.2.8. The initial setup required for this protocol is listed in Table 6.

Table 6—Setup for command protection

Symbol	Initial owner	
	DEV	Security manager
Sym_keys_S	✓	✓
SECID_S	✓	✓
Time token	✓	✓
SFC_S	_	✓

The cryptographic functionality required to implement this protocol is described in Table 7.

Table 7—Capabilities for command protection

Functionality	Required	
	DEV	Security manager
Symmetric integrity code	✓	✓

The source DEV should initiate this protocol each time it transmits a secure command. The command protection protocol is illustrated in Figure 16.

Figure 16—Command protection protocol

9.8.6 Distribute key protocol

In a secure piconet or in a secure peer-to-peer relationship, the security manager may wish to update the current data protection key by initiating the distribute key protocol described here. For a change in the piconet group data key, the PNC sends the new piconet group data key to each authenticated DEV before changing

the key using the distribute key protocol. For a change in a peer data key, the security manager in the relationship initiates the distribute key protocol. The initial setup required for this protocol is listed in Table 8.

Table 8—Setup for key distribution

Symbol	Initial owner	
	DEV	Security manager
Sym_keys_D	✓	✓
SECID_D	✓	✓
Time token	✓	✓
Sym_keys_G (current key)	_	✓
SFC_D	✓	_
SFC_SM	_	✓

The cryptographic functionality required to implement this protocol is described in Table 9.

Table 9—Capabilities for key distribution

Functionality	Required	
	DEV	Security manager
Symmetric decryption	✓	_
Symmetric encryption	_	✓
Symmetric integrity code	✓	✓

The security manager should initiate this protocol with each DEV with their respective shared keys whenever the key is updated. The distribute key protocol is illustrated in Figure 17.

Figure 18 illustrates the message flows for the key distribution protocol between the PNC and a DEV. Figure 19 illustrates the message flows for the key distribution protocol between a DEV acting as the security manager of the relationship and a DEV. Note that for the PNC-DEV key distribution, the DEV does not send a distribute key response to the PNC.

Figure 18—Message sequence chart for PNC-DEV key distribution

Figure 19—Message sequence chart for peer-to-peer key distribution

9.8.7 Key request protocol

In a secure piconet, if a DEV receives a frame or beacon with an unknown SECID, it may initiate the request key protocol described here in order to obtain the unknown key from the security manager of the relationship. The initial setup required for this protocol is listed in Table 10.

Table 10—Setup for key request

Symbol	Initial owner	
	DEV	Security manager
Sym_keys_D	✓	✓
SECID_D	✓	✓
Time token	✓	✓
Sym_keys_G (current key)	_	✓
SFC_D	✓	_
SFC_SM	_	✓

The cryptographic functionality required to implement this protocol is described in Table 11.

Table 11—Capabilities for key request

Functionality	Required	
	DEV	Security manager
Symmetric decryption	✓	–
Symmetric encryption	–	✓
Symmetric integrity code	✓	✓

The DEV should initiate this protocol with the security manager if it is already authenticated, but does not have the current payload protection key. The key request protocol is illustrated in Figure 20.

Figure 21 illustrates the message flows for the key request protocol between a DEV and the PNC.

Figure 21—Message sequence chart for DEV-PNC key request

9.8.8 Key management states

This subclause specifies the device states that relate to key management, but that are not part of the authentication protocol.

9.8.8.1 Device states

Figure 22 shows the states and state transitions that apply to the DEV during PNC handover and the distribute key and key request protocols. The MSCs that correspond with this state machine are shown in Figure 155, Figure 18, Figure 19, Figure 21, and Figure 24.

Figure 22—Key management state machine – DEV role

Table 12 describes the key management device states.

Table 12—Device key management states

State	Name	Description	Action
D7.0	Unavailable key	Default state for a DEV. State in which the DEV is not currently authenticated and does not possess a key to perform secure commands.	DEV ignores all frames directed to this state machine from the SM. If the security state controller sends an indication of an authentication with the SM, the DEV shall transition to the secure group membership state.
D8.0	Secure group membership	State in which the DEV is authenticated and may actively participate in the secure relationship.	DEV ignores all frames directed to this state machine from the SM except: • disassociate command • de-authenticate command (if a peer-to-peer relationship) • distribute key request command If the security state controller sends an indication of a disassociation, de-authentication or PNC handover, the DEV shall transition to the unavailable key state. If the security state controller sends an indication that a PNC handover response was sent, the DEV shall transition to the waiting for PNC handover state. If the security state controllersecurity state controller sends an indication of an unknown SECID in the beacon, the DEV shall send a key request command and transition to the waiting for key response state.
D9.0	Waiting for key response	DEV is waiting to receive a key and is unable to verify the validity of the beacon.	DEV ignores all frames directed to this state machine from the SM except: • distribute key command • request key response command • disassociate command • de-authenticate command (if a peer-to-peer relationship) If the security state controller sends an indication of a disassociation, de-authentication or PNC handover, the DEV shall transition to the unavailable key state. If the security state controller sends an indication that a beacon with a known key was received, the DEV shall transition to the secure group membership state. If the security state controller sends an indication that a PNC handover response was sent, the DEV shall transition to the waiting for PNC handover state.
D10.0	Waiting for PNC handover	DEV has been selected as the next PNC and is waiting for the appropriate time to assume the role of PNC.	DEV ignores all frames directed to this state machine from the SM. If the security state controller sends an indication of a PNC handover, the DEV shall transition to the security manager waiting state and assume the role of the PNC.

9.8.8.2 Device state transitions

Table 13 describes the processes and causes of the state transitions.

Table 13—Device key management state transitions

State	Transition	Description
D7.8	Unavailable key to secure group membership.	When a DEV in the unavailable key state receives an indication that an authentication has been completed, it performs this transition to the secure group membership state.
D8.7	Secure group membership to unavailable key.	When a DEV is in the secure group membership state and receives a disassociate command or de-authenticate command from the security manager or the DME or receives an indication from the security state controller that a new PNC has assumed control of the piconet, the DEV sends the disassociate or de-authenticate command (if applicable), securely deletes the shared keys with the security manager and performs this transition to the unavailable key state.
D8.9	Secure group membership to waiting for key response.	When a DEV is in the secure group membership state and receives an indication from the security state controller that a beacon was received with a SECID that is unfamiliar to it, it sends a request key command to the security manager, starts a counter to determine the length of time it will wait for the key response command and performs this transition to the waiting for key response state.
D9.7	Waiting for key response to unavailable key.	When a DEV is in the waiting for key response state and receives a disassociate or de-authenticate command from the security manager or the DME or receives an indication from the security state controller that a new PNC has assumed control of the piconet, the DEV sends the disassociate or de-authenticate command (if applicable), securely deletes its shared keys with the security manager and performs this transition to the unavailable key state.
D9.8(1)	Waiting for key response to secure group membership (1).	When a DEV is in the waiting for key response state and times out, receives a valid key response command or receives an indication from the security state controller that a beacon with a recognizable SECID was received, the DEV updates its current key (if applicable) and performs this transition to the secure group membership state.
D9.8(2)	Waiting for key response to secure group membership (2).	When a DEV is in the waiting for key response state and receives a valid distribute key command from the security manager, the DEV updates its current key and performs this transition to the secure group membership state.
D8.8	Secure group membership to secure group membership.	When a DEV is in the secure group membership state and receives a valid distribute key command, the DEV shall update the key, send a distribute key response command and remain in the secure group membership state.
D8.10	Secure group membership to waiting for PNC info.	When a DEV that is an alternate PNC is in secure group membership state and receives an indication from the security state controller that a PNC handover response command has been sent, it performs this transition to the waiting for PNC info state.

Table 13—Device key management state transitions

State	Transition	Description
D9.10	Waiting for key response to waiting for PNC info.	When a DEV that is an alternate PNC is in waiting for key response state and receives an indication from the security state controller that a PNC handover response command has been sent, it performs this transition to the waiting for PNC info state.
D10.SM1	Waiting for PNC info to (security manager) waiting state.	When a DEV is in the waiting for PNC info state and receives the final PNC handover information command or times out, the DEV shall update its DEV information table (if applicable) and perform this transition to the (security manager) waiting state. At this point, the DEV takes on the role of security manager and PNC.

9.8.8.3 Security manager states

Figure 23 shows the states and state transitions that apply to the security manager during PNC handover and the distribute key and key request protocols. The MSCs that correspond with this state machine are shown in Figure 155, Figure 18, Figure 19, Figure 21, and Figure 24.

Figure 23—Key management state machine – security manager role

Table 14 describes the key management security manager states.

Table 14—Security manager key management state diagrams

State	Name	Description	Action
SM5.0	Unavailable key	Default state for a security manager. Security manager generates new group keys and sends distribute key commands to each of the authenticated DEVs in the piconet. When all of the distribute key commands have been sent (which should occur very rapidly), the security manager will transition to the secure state.	• Security manager ignores all frames directed to this state machine. If the security state controller sends an indication of a successful authentication, the SM shall transition to the pending key state.
SM6.0	Pending key	Security manager generates new group keys and sends distribute key commands to the authenticated DEV in the relationship when it is ready. For the piconet-wide security relationship, the PNC may wait until it has completed all pending authentication protocols before sending the distribute key commands.When all of the distribute key commands have been sent (which should occur very rapidly), the security manager will transition to the secure state.	Security manager ignores all frames directed to this state machine from the device role except: • De-authenticate command • Disassociate command If the security state controller sends an indication of a de-authentication or disassociation, the SM shall transition to the unavailable key state.
SM7.0	Secure state	Normal operating state when the security manager shares a key with a DEV.	Security manager ignores all frames directed to this state machine from the device role except: • De-authenticate command • Disassociate command • request key command • distribute key response command If the security state controller sends an indication of a de-authentication or disassociation, the SM shall transition to the unavailable key state. If the security state controller sends an indication that a valid PNC handover response command has been received, the SM shall transition to the PNC handover pending state.
SM8.0	PNC handover pending	PNC has already recieved a PNC handover response command and is in the process of PNC handover. After the final beacon is sent the security manager shall transition to the unavailable key state, which is a DEV state.	Security manager ignores all frames directed to this state machine while in this state.

9.8.8.4 Security manager state transitions

Table 15 describes the processes and causes of the state transitions.

Table 15—Security manager key management state transitions

State	Transition	Description
SM5.6	Unavailable key to pending key (1)	When a security manager is in unavailable key state and completes all of the authentication protocols for the startup process, the security manager performs this transition to the pending key state.
SM6.5	Pending key to unavailable key	If the security manager receives a valid secure de-authentication or disassociation command or receives an indication from the security state controller of a disassociation or de-authentication, the security manager shall securely delete the relationship key(s) and perform this transition to the unavailable key state.
SM6.7	Pending key to secure state	When a security manager is in pending key state, has generated a new group key for the security relationship and has sent the distribute key command to the authenticated DEV, the security manager shall perform this transition to the secure state. If the PNC has transitioned out of pending key state for all piconet-wide security relationships, the PNC shall change the beacon to include the security session key of the new key and protect future beacons with that key.
SM7.6	Secure state to pending key	When a security manager is in secure state and a DEV is disassociated (if the device is the PNC), a new DEV is authenticated (if the device is the PNC), upon instruction by the DME or if the security state controller sends an indication of a new authentication with that DEV, the security manager shall prepare to update the key and perform this transition to the pending key state.
SM7.7	Secure state to secure state	When the security manager is in secure state and receives a valid key request command from an authenticated DEV, the security manager shall send a key response command to the DEV and remain in the secure state.
SM7.5	Secure state to unavailable key	When the security manager is in secure state and receives a valid seucre de-authenticate command or disassociation command from an authenticated DEV or if the security state controller sends an indication that a de-authentication or disassociation has occurred, the security manager shall securely delete the keys shared with that DEV and perform this transition to the unavailable key state.
SM7.8	Secure state to PNC handover pending	When the PNC is in secure state and receives an indication from the security state controller that a PNC handover response has been received, the PNC shall performs this transition to the PNC handover pending state.
SM8.D7	PNC handover pending to (device) unavailable key state	When the PNC is in PNC handover pending state and the security state controller sends an indication that the PNC handover has occurred or, the PNC ceases sending beacons and performs this transition to the (device) unavailable key state.

9.8.8.5 Combined De-authentication states

Figure 24 illustrates the message flows for DEV-3 ending its peer-to-peer authentication relationship with DEV-2.

Figure 24—MLME-DE-AUTHENTICATE message sequence chart

9.8.9 Data protection protocol

When a DEV transmits (or recieve) a secure data frame, the DEV shall protect (or verify) the frame using the data protection protocol described here. The initial setup required for this protocol is listed in Table 16.

Table 16—Setup for data protection protocol

Symbol	Initial owner	
	Sending DEV	Receiving DEV
data	✓	_
Sym_keys_S	✓	✓
SECID_S	✓	✓
Time token	✓	✓
SFC_S	✓	_

The cryptographic functionality required to implement this protocol is described in Table 17.

Table 17—Capabilities for data protection protocol

Functionality	Required	
	Sending DEV	Receiving DEV
Symmetric decryption	–	✓
Symmetric encryption	✓	–
Symmetric integrity code	✓	✓

The sending DEV may be acting as a normal DEV or a security manager for the particular key. In either case, the key is mutually shared between all members of the group. The data protection protocol is illustrated in Figure 25.

Figure 25—Data protection protocol

9.8.9.1 Device states

Figure 26 shows the states and state transitions that apply to the DEV when it receives secure data frames.

Figure 26—Secure data reception state machine – DEV role

Table 18 describes the device states when receiving secure data frames.

Table 18—Device secure data reception states

State	Name	Description	Action
D11.0	Unavailable key	Default state for a DEV before authenticating. The DEV does not share a key with the SM.	DEV accepts all frames directed to this state machine while in this state.
D12.0	Insecure checking frame	A processing-only state in which the DEV processes the frame to determine what to do with it. If the DEV receives a secure non-beacon frame, the DEV shall reject the frame. If the DEV receives a secure beacon frame, it shall process the frame as specified in 9.2.8. If the frame is an insecure data or beacon frame, the frame shall be accepted. If the frame is an insecure command frame, the frame shall be processed as specified in 7.5.	DEV shall queue all frames all frames directed to this state machine while in this state.
D13.0	Secure data reception	The normal operating state for an authenticated DEV. The DEV shares a key with the SM.	DEV accepts all frames directed to this state machine while in this state.
D14.0	Secure checking frame	A processing-only state in which the DEV processes the frame to determine what to do with it. If the DEV receives an insecure data or beacon frame, the DEV shall reject the frame. If the DEV receives an insecure command frame, the frame shall be processed as specified in 7.5. If the DEV receives a secure frame, the frame shall be processed as specified in 9.2.8.	DEV shall queue all frames directed to this state machine while in this state.

9.8.9.2 Device state transitions

Table 19 describes the device state transitions while receiving secure data frames.

Table 19—Device secure data reception state transitions

State	Transition	Description
D11.12	Unavailable key to insecure checking frame	When a DEV is in unavailable key state and receives a frame from the security manager, the DEV performs this transition to the insecure checking frame state.
D12.11(1)	Insecure checking frame to unavailable key (1)	When a DEV in the insecure checking frame state determines that the frame should be rejected the frame is rejected and the DEV transitions back to the unavailable key state.
D12.11(2)	Insecure checking frame to unavailable key (2)	When a DEV in the insecure checking frame state determines that the frame should be accepted, the frame is accepted and the DEV transitions back to the unavailable key state.
D11.13	Unavailable key to secure data reception	When a DEV is in unavailable key state and receives an indication from the security state controller that the device is authenticated, it performs this transition to the secure data reception state.
D13.14	Secure data reception to secure checking frame	When a DEV is in the secure data reception state and receives a frame from the security manager, the DEV performs this transition to the secure checking frame state.
D14.13(1)	Secure checking frame to secure data reception (1)	When a DEV in the secure checking frame state determines that the frame should be rejected the frame is rejected and the DEV transitions back to the secure data reception state.
D14.13(2)	Secure checking frame to secure data reception (2)	When a DEV in the secure checking frame state determines that the frame should be accepted, the frame is accepted and the DEV transitions back to the secure data reception state. If the frame is a secure beacon frame with an unexpected SECID or with a PNC handover IE, the secure frame processing state machine shall send an indication to the security state controller indicating the result of the beacon processing.
D13.11	Secure data reception to unavailable key	When a DEV is in the secure data reception state and recieves an indication from the security state controller that the device is de-authenticated or disassociated or that a PNC handover has occurred, it performs this transition to the unavailable key state.

9.8.9.3 Security manager states

Figure 27 shows the states and state transitions that apply to the security manager when it receives secure data frames.

Figure 27—Secure data reception state machine – security manager role

Table 20 describes the security manager states while receiving secure data frames.

Table 20—Security manager secure data reception states

State	Name	Description	Action
SM9.0	Unavailable key	Default state for a security manager before authenticating. The SM does not share a key with the DEV.	SM accepts all frames directed to this state machine while in this state.
SM10.0	Insecure checking frame	A processing-only state in which the SM processes the frame to determine what to do with it. If the DEV receives a secure frame, the DEV shall reject the frame. If the frame is an insecure data frame, the frame shall be accepted. If the frame is an insecure command frame, the frame shall be processed as specified in 7.5.	SM shall queue all frames all frames directed to this state machine while in this state.
SM11.0	Secure data reception	The normal operating state for an authenticated SM. The SM shares a key with the DEV.	SM accepts all frames directed to this state machine while in this state.
SM12.0	Secure checking frame	A processing-only state in which the SM processes the frame to determine what to do with it. If the DEV receives an insecure data frame, the DEV shall reject the frame. If the DEV receives an insecure command frame, the frame shall be processed as specified in 7.5. If the DEV receives a secure frame, the frame shall be processed as specified in 9.2.8.	SM shall queue all frames directed to this state machine while in this state.

9.8.9.4 Security manager state transitions

Table 21 describes the processes and causes of the state transitions for secure data reception.

Table 21—Security manager secure data reception state transitions

State	Transition	Description
SM9.10	Unavailable key to insecure checking frame	When a security manager is in unavailable key state and receives a frame from the DEV, the SM performs this transition to the insecure checking frame state.
SM10.9(1)	Insecure checking frame to unavailable key (1)	When a security manager in the insecure checking frame state determines that the frame should be rejected the frame is rejected and the SM transitions back to the unavailable key state.
SM10.9(2)	Insecure checking frame to unavailable key (2)	When a security manager in the insecure checking frame state determines that the frame should be accepted, the frame is accepted and the SM transitions back to the unavailable key state.
SM9.11	Unavailable key to secure data reception	When a security manager is in unavailable key state and receives an indication from the security state controller that the device is authenticated, it performs this transition to the secure data reception state.
SM11.12	Secure data reception to secure checking frame	When a security manager is in the secure data reception state and receives a frame from the DEV, the SM performs this transition to the secure checking frame state.
SM12.11(1)	Secure checking frame to secure data reception (1)	When a security manager in the secure checking frame state determines that the frame should be rejected, the frame is rejected and the SM transitions back to the secure data reception state.
SM12.11(2)	Secure checking frame to secure data reception (2)	When a security manager in the secure checking frame state determines that the frame should be accepted, the frame is accepted and the SM transitions back to the secure data reception state.
SM11.9	Secure data reception to unavailable key	When a security manager is in the secure data reception state and recieves an indication from the security state controller that the device is de-authenticated or disassociated or that a PNC handover has occurred, it performs this transition to the unavailable key state.

Replace Clause 10.1 with:

When a security suite is selected, DEVs perform secure operations in mode 1. This mode is defined in 9.3. Security suites are not defined for mode 0.

The security manager in any security relationship selects an OID that corresponds to the security suite and sub-suite that it will use. In the association response command , the security manager shall send the desired OID for the security suite to assist that DEV in starting the authentication process. The DEV or PNC may also send probe commands to each other before the authentication process to determine which OIDs are mutually supported. If in the authentication request command, {xref}, the DEV sends an OID that is not accepted by the security manager, the security manager shall reject the authentication request and return a failed authentication response command indicating that the OID was not accepted. Otherwise, the security manager should continue with the authentication protocol.

Replace 10.2 with:

10.2 Symmetric cryptography building blocks

The following cryptographic primitives and data elements are defined for use in all security suites specified in this standard.

10.2.1 Security interfaces

When transmitting and interpreting security material in this standard, the first byte transmitted shall be the first byte of the security material and represented on the left of the other bytes. The bit ordering within the byte for security operations shall be most significant bit first and least significant bit last. This ordering shall be irrespective of the transmission order of the bits. See Figure 4 for the mapping of bit transmission order to most significant or least significant.

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

The combined symmetric encryption and authentication mechanisms used in this security suite 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 B.1. The parameters for these operations shall be as specified in 10.2.3.

10.2.3 CCM parameters

The CCM operations shall be parameterized by the following selections: the AES encryption algorithm as specified in 10.2.5, the length in octets of the length field L shall be 2 octets, the length of the authentication field M shall be 8 octets, the nonce shall be formatted as specified in 10.2.4.

10.2.4 Nonce value

The nonce used for CCM encryption and authentication shall be a 13-octet field, dependent on the frame in which it is used, consisting of the 8-bit SrcID followed by the 8-bit DestID followed by the current 6-octet time token followed by the 2-octet secure frame counter followed by the 3-octet fragmentation control field from the MAC header. In order to preserve the security of the symmetric algorithms, this nonce shall be unique. As a result, the DEV shall not reuse any 2-octet sequence number within a single superframe that is intented for a particular DEVID (as this would cause a repeated nonce). This uniqueness is guaranteed by the use of the SrcID, which guarantees that different DEVs sharing the same key will use a different nonce, by the time token, which is different for every superframe with a given key and by the DestID and secure frame counter, which guarantee uniqueness within a superframe as long as a DEV does not send more than 65536 frames to a particular DestID within that superframe. If a frame is retransmitted and a single bit in the header or frame body has changed, a new nonce shall be used. To ensure this, each time a frame is retransmitted the secure frame counter shall be incremented.

Figure 28 specifies the format of the nonce that is input to the CCM algorithm. The SrcID, DestID, secure frame counter and fragmentation control field shall be included in the frame that is being protected. The time token shall be the time token from the beacon for this superframe.

3	2	6	1	octets: 1
Fragmentation control field	Secure frame counter	Time token	DestID	SrcID

Figure 28—CCM nonce format

10.2.5 AES encryption

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

10.3 Symmetric cryptography implementation

All of the sub-suites defined in this standard perform symmetric operations in the same manner as specified in this subclause.

10.3.1 Symmetric cryptography data formats

Table 22 specifies the length and meaning of the symmetric cryptography related security suite specific data elements from Clause 7. The operations performed to obtain the variable data values are specified in 10.3.2.

Table 22—Symmetric cryptography frame object formats

Notation	Length	Value	Description
Encrypted key	16	Variable	The encrypted key consists of the result of the encryption of a 16-octet key (not including the integrity code) using CCM encryption as specified in 10.2.2.
Integrity code	8	Variable	The integrity code consists of the encrypted integrity code that is the result of a CCM computation as specified in 10.2.2 that is computed along with the encrypted seed.
Encrypted data	Variable	Variable	The encrypted data consists of the result of the encryption of the specified data (not including the integrity code) using CCM encryption as specified in 10.2.2.

10.3.2 Symmetric cryptographic operations

Table 23 specifies the symmetric cryptography related operations on all secure frames defined in Clause 7.

Table 23—Symmetric cryptographic operations

Operation	Specification
Secure beacon integrity code generation	The integrity code included in the beacon is generated by computing the encrypted integrity code with the piconet-wide payload protection key using CCM authentication and encryption as specified in 10.2.2 with the entire beacon up to the integrity code information element as the authentication data input *a* and the empty string as the plaintext input *m* for encryption.
Secure command integrity code generation	The integrity code included in command frames is generated by computing the encrypted integrity code with the payload protection key using CCM authentication and encryption as specified in 10.2.2 with the entire command up to the integrity code as the authentication data input *a* and the empty string as the plaintext input *m* for encryption.
Data integrity code generation	The integrity code included in data frames is generated by computing the encrypted integrity code with the payload protection key using CCM authentication and encryption as specified in 10.2.2 on the entire data frame up to the encrypted data field as the authentication data input *a* and the data to be encrypted as the plaintext input *m* for encryption (and authentication).
Seed encryption operation	The seed for key transport is encrypted using CCM authentication and encryption on the seed as specified in 10.2.2 using the management payload protection key with the entire command frame up to the encrypted seed field as the authentication data input *a* and the 16-octet pre-encrypted seed as the plaintext input *m* for encryption.
Data encryption generation	Data in a data frame is encrypted using CCM authenticated encryption as specified in 10.2.2 using the data payload protection key with the entire data frame up to the encrypted data field as the authentication data input *a* and the data to be encrypted as the plaintext input *m* for encryption (and authentication).

Figure 29 specifies the length information and data input to the CCM operation for secure beacons. The auth data length *l(a)* shall be set to the length of all of the protected data and the enc data length *l(m)* shall be set to 0. The data input to CCM shall be taken in the order it is received in the frame, omitting the HCS, FCS and integrity code.

Octets: 2	**2**
Enc Data Length *l(m)* = 0	Auth Data Length *l(a)* = $27+L_1+...+L_{n-1}$

L_{n-1}	...	L_1	13	2	2	10
Information element-(n-1)	...	Information element-1	Piconet synch. parameters	Secure frame counter	SECID	Frame header

Figure 29—CCM input for secure beacons

Figure 30 specifies the length information and data input to the CCM operation for secure commands. For all commands except for the request key response command and distribute key request command, the auth data length *l(a)* shall be set to the length of all of the protected data and the length of encrypted data *l(m)* shall be set to 0. For the request key response command and distribute key request command, the auth data length *l(a)* shall be set to the length of all of the protected data minus 16 (the length of the key) and the enc data length shall be set to 16 (the length of the key). The data input to CCM shall be taken in the order it is received in the frame, omitting the HCS, FCS and integrity code.

Octets: 2	**2**
Enc Data Length *l(m)* = L_2	Auth Data Length *l(a)* = 18+L_1

L_2	**L_1**	**2**	**2**	**2**	**2**	**Octets: 10**
Enc data	Auth data	Length (=4+L_1+L_2)	Command type	Secure frame counter	SECID	Frame header

Figure 30—CCM input for secure commands

Figure 31 specifies the length information and data input to the CCM operation for secure data frames. The auth data length *l(a)* shall be set to 14 and the length of encrypted data *l(m)* shall be set to the length of the data payload. The data input to CCM shall be taken in the order it is received in the frame, omitting the HCS, FCS and integrity code

Octets: 2	**2**
Enc Data Length *l(m)* = L_1	Auth Data Length *l(a)* = 14

L_1	**2**	**2**	**Octets: 10**
Pre-encrypted data	Secure frame counter	SECID	Frame header

Figure 31—CCM input for secure data frames

Remove 10.3-10.5

Remove Annex B.2-B.3

Remove C.2.1 and Table C.1

Remove C.3-C.5

Update Table E.5 to remove the following items:

1) S2.1-S2.6
2) S4.1-S4.13