IEEE P802.15 Wireless Personal Area Networks

Project	IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs)
Title	**IEEE P802-15_TG3 NTRU Full Security Text Proposal**
Date Submitted	[April 22, 2002]
Source	[Daniel V. Bailey, Ari Singer, William Whyte] [NTRU] [5 Burlington Woods Burlington, MA 01803 USA] Voice: [+1 781 418-2522] Fax: [+1 781 418-2532] E-mail: [dbailey@ntru.com]
Re:	802.15.3 TG3 Letter Ballot Draft D09, 02074r1P802.15_TG3-Security-CFP.doc, 02130r1P802-15_TG3-NTRU-Security-Architecture-Proposal.doc
Abstract	[This document is offered as NTRU's submission for the letter ballot vote on the mandatory and optional ECC security suites. The full baseline security text that both the Certicom/Motorola and NTRU submissions are based on is included.]
Purpose	[This document is intended as the NTRU proposal for ECC security suites for the ECC security suite letter ballot. Background security text for both submissions is included to assist the balloters.]
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.

[[The following submission specifies the baseline security text for the 802.15.3 draft standard, incorporating the NTRU submission text for the security suite ballot vote. The clauses specified are as follows: Security (which includes overview, protocols and state machines), Security Suites (which defines the security suites to be used in this standard) and Informative Text (which gives some security considerations for the general protocols and architectural framework). This submission specifies the security clauses voted on for inclusion into the 802.15.3 draft standard at the March, 2002 802.15 TG3 meeting in St. Louis, MO. Specifically, this includes the security text approved by the vote to include 02/130 as the baseline architecture, text supporting the vote to mandate AES, SHA-256 and HMAC, and text supporting the vote to include an ECC security suite in mode 2, an ECC suite in mode 3 using implicit certificates, an ECC suite in mode 3 using a patent unencumbered certificate format such as X.509 and an optional NTRUEncrypt suite in mode 2.]]

[[The NTRU submission text for the ECC security suite letter ballot vote is found in clause 2.3 and clause 4. The submission specific text is pointed out explicitly with notes in this font inside square brackets.]]

1. Security

Wireless networks face unique security challenges and WPANs are no exception. Recognizing the diversity of WPAN applications and devices, this standard supports four different modes of security, ranging from no security to the use of strong cryptography. The instantiation of a cryptographic security mode is a security suite.

1.1 Background Assumptions

All security solutions rely on assumptions about devices and the capabilities of potential attackers to thwart possible threats.

1.1.1 Physical Assumptions

The assumptions below are made about the physical environment for the WPAN. The physical constraints help to determine the security architecture.

— **Open communications medium** – Since the data being transmitted may be received by any device that is sufficiently close and has a sufficiently good receiver, it is assumed that transmissions may be received by devices outside of the WPAN.
— **Low cost** – Like all other components of a WPAN device, security must be provided with careful attention to cost.
— **Dynamic group membership** – Devices are expected to be roaming and it is therefore assumed that the devices may enter or exit the network at any time.
— **No access to external networks** – Security solutions must be effective without access to external networks.
— **Bandwidth** – Since 802.15.3 WPANs provide high data rates, reasonable amounts of bandwidth overhead due to security are acceptable.
— **Computational power** – The devices that will be used are assumed to have very little computational power with only a small portion of that available for cryptographic computations.
— **Memory** – It is assumed that the low end devices implementing 802.15.3 will have little memory available for security.

1.1.2 Network Assumptions

The assumptions below are made about the network structure of the WPAN. The network constraints help to determine the security architecture.

— **Network size** – There is a fixed upper bound of 255 devices in a WPAN and it is assumed that the security solution will scale up to that size if necessary.
— **Controller** – One device, the PNC, has the role of managing time slots and entry into the piconet. It is assumed that the PNC has more resources and functionality than an ordinary device.
— **Dynamic controller** – The PNC is assumed to have the ability to leave the network or hand over the PNC role to other devices.
— **Device relationships** – The wide array of use cases describes several models for the pre-existing relationship of devices in the piconet. It is assumed that devices may have pre-existing security relationships or they may have never met and that both types of relationship may exist within a single piconet.

1.1.3 Attack Model Assumptions

In order to make statements about the effectiveness of security measures, it is necessary to describe the capabilities of the attackers and the nature of the attackers.

— **Computational capabilities** – It is assumed that the attacker has state of the art technologies to perform rapid computations.
— **Listening capabilities** – It is assumed that the attacker is within listening range of the devices in the WPAN and understands the communication mechanism.
— **Broadcast capabilities** – It is assumed that the attacker has sophisticated broadcasting equipment that is able to synchronize with the piconet and transmit data for the devices in the piconet at the appropriate time.
— **Security setup** – The security setup for the devices may occur before entry into the piconet or after the piconet has been established. No assumptions are made about the presence of attackers during security setup.

1.2 Security Services

Security services are protections offered on communications between the PNC and a DEV or two ordinary DEVs.

1.2.1 Access Control List

The Access Control List (ACL) indicates from which devices the MLME will accept an authentication request command. The ACL may consist of a list of authorized Device IDs as well as other device-specific information depending on the mode and security suite in use. Alternatively, the ACL may simply indicate that authentication request commands are to be accepted from any peer DEV. The DME is responsible for maintaining the ACL.

The device stores ACL entries in the MAC PIB as described in table {xref-clause 6.2.2} - MAC PIB ACL Entry. In an ACL entry for a secure relationship, the ACL contains an indication of the security suite to be used and verification information for the public key of the device that depends on the security suite.

1.2.2 Mutual Authentication

The mutual authentication protocol, defined in 9.8.1 {xref}, 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. 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.

When a piconet is operating in Mode 2 or 3, defined in 9.4 {xref}, mutual authentication shall be achieved by means of public-key cryptography.

1.2.3 Verifying Authenticity of Public Keys

To use a public key to achieve mutual authentication, a DEV must establish that the received public key belongs to the intended device. This trust shall be indicated by the key’s representation in the ACL in Mode 2 or by the DEV verifying a digital certificate at the time of authentication in Mode 3.

1.2.4 Key Establishment

The mutual authentication protocol, defined in 9.8.1 {xref} shall result in the establishment of a shared key or keys that may be used for future communications between the devices.

The key(s) that are established shall be used to protect commands between the two devices.

1.2.5 Key Transport

All keys that are transmitted from one DEV to another shall be encrypted as specified in the key request and distribute key protocols 7.4.1{xref}. For example, key transport is used to provide a copy of the piconet-wide key to a DEV.

1.2.6 Data encryption

Encryption 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 all piconet devices or using a key shared between two peers.

1.2.7 Data integrity

Integrity uses a message authentication code 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. Integrity may be provided using a key shared by all piconet devices or using a key shared between two peers.

1.2.8 Beacon integrity protection

The beacon may be integrity-protected. This integrity protection provides evidence to all the DEVs in the piconet that a member of the secure piconet transmitted the beacon.

1.2.9 Freshness protection

To prevent replay of old messages, a strictly-increasing time token is included in the beacon. A DEV may reject as invalid a received time token less than or equal to the current time token. A DEV may reject as invalid a frame received with a time token less than the current time token.

All protected beacons, ACKs, commands and data shall include the current time token and integrity protection.

1.2.10 Command integrity protection

The integrity of commands may be protected just like any other data. PNC-DEV commands shall be protected using the integrity key agreed on during mutual authentication.

1.2.11 ACK integrity protection

The integrity of ACK frames may be protected just like any other data. The key used to protect the integrity of the ACK shall be the key used for the command.

1.3 Security Policies

Security policies determine the actions taken to preserve the security of the piconet.

1.3.1 Group membership change rekey

Sound security practice indicates that only devices that are currently members of the piconet should be allowed to generate, read or modify piconet data. This implies that when a device joins or leaves the piconet, the currently active group keys need to be changed.

Since group membership is controlled by the PNC, the PNC is responsible for determining when a device has joined or left the piconet and for changing and distributing the new group keys.

Before the PNC distributes the group piconet keys to a newly authenticated DEV, the PNC shall change the group piconet keys and issue a distribute key command to each of the authenticated DEVs to distribute the new key.

When the PNC disassociates a DEV from the piconet, the PNC shall change the group piconet keys and issue a distribute key command to each of the authenticated DEVs to distribute the new key.

1.3.2 PNC handover

When a PNC chooses to handover the PNC role to another device in the piconet, the authentication relationships with the old PNC do not apply to the new PNC. When the old PNC hands over the piconet information (using a device information response command), the list of authenticated DEVs is passed to the new PNC.

PNC handover does not affect the group membership, so it does not require a rekey of the group keys. However, in a piconet with payload protection, the command functions of the PNC that relate to specific devices cannot be implemented until the new PNC has performed the authentication protocol with each device in the piconet. When the PNC role has been handed over, the new PNC should set up time slots for each of the authenticated DEVs to perform the authentication protocol with the new PNC.

The old PNC may send public-key verification information about the new PNC to the other devices in the piconet and send public-key verification information about all of the authenticated devices in the piconet to the new PNC when it hands over the role of the PNC using a probe command. If the DME of each devices chooses to accept this public-key verification information, the authentication process with each DEV can proceed without any interruption of service.

1.4 Security Modes

1.4.1 Security Mode 0

A piconet with no security enabled is operating in Security Mode 0.

1.4.2 Security Mode 1

A piconet that restricts association based on Device ID alone is operating in Security Mode 1. No cryptography is used: the PNC only allows a device to associate if its Device ID is in the PNC’s ACL.

1.4.2.1 Goal of Mode 1

Only authorized devices may join a secure piconet.

1.4.2.2 Security Services Provided in Mode 1

— **Access control list**, defined in 1.2.1 {xref}.

1.4.3 Security Mode 2

A piconet in Security Mode 2 uses public-key cryptography to verify the authenticity of a device found in its ACL and to protect piconet data using encryption and integrity.

1.4.3.1 Goals of Mode 2

Only authorized devices may join a secure piconet.

Communication is restricted to authorized devices.

1.4.3.2 Security Services Provided in Mode 2

— **Mutual authentication**, defined in 1.2.2 {xref}
— **Verifying authenticity of public keys**, defined in 1.2.3
— **Access control list**, defined in 1.2.1
— **Key establishment**, defined in 1.2.4
— **Key transport**, defined in 1.2.5
— **Beacon integrity protection**, defined in 1.2.8
— **Freshness protection**, defined in 1.2.9
— **Command integrity protection**, defined in 1.2.10
— **ACK integrity protection**, defined in 1.2.11
— **Data integrity protection**, defined in 1.2.7
— **Data encryption**, defined in 1.2.6

1.4.4 Security Mode 3

A piconet in Security Mode 3 uses public-key cryptography to verify the authenticity of a device using a public-key certificate and to protect piconet data using encryption and integrity.

1.4.4.1 Goals of Mode 3

Only authorized devices may join a secure piconet.

Communication is restricted to authorized devices.

1.4.4.2 Security Services Provided in Mode 3

— **Mutual authentication**, defined in 1.2.2 {xref}
— **Verifying authenticity of public keys**, defined in 1.2.3
— **Access control list**, defined in 1.2.1
— **Key establishment**, defined in 1.2.4
— **Key transport**, defined in 1.2.5
— **Beacon integrity protection**, defined in 1.2.8
— **Freshness protection**, defined in 1.2.9
— **Command integrity protection**, defined in 1.2.10

- **ACK integrity protection**, defined in 1.2.11
- **Data integrity protection**, defined in 1.2.7
- **Data encryption**, defined in 1.2.6

1.5 Security Suites

A security suite defines mechanisms to implement a particular set of security operations. A security sub-suite, identified by a unique OID, specifies the exact selections of operations within a security suite. A security suite in a particular sub-suite shall provide all security services listed above for that particular mode. The mandatory security suite sub-suite and a list of the accepted optional security suites and sub-suites are specified in clause 11{xref}.

A security suite shall also specify each of the following:

1.5.1 OID

Each security suite shall have a globally unique OID associated with it that will never change. The OID shall refer explicitly to a security suite defined for this standard. OIDs shall be built from the 802.15.3 arc, specified in sub-clause 11.2.1{xref}.

1.5.2 Data Formats

Each security suite shall specify the data elements that are not specified in the frame formats in clause 7. For each data element, the number of bytes shall be specified that is used in the command, the expected value (when appropriate) and a description of the meaning and nature of the data element.

1.5.3 Protocol Operations

For each protocol used by the security suite, the security suite shall provide a list of operations that are performed by each participating DEV. These protocol operations will typically include cryptographic operations, data manipulation and determination of the appropriate action based on the result of the operations.

1.5.3.1 Cryptographic Implementations

For all cryptographic implementations in the security suite, the security suite specification shall provide a complete and unambiguous description of the cryptographic operations performed. These descriptions may be provided by reference to external documents.

1.5.4 Security Considerations

Since different security suites employ different security methods, the security suite specification should include a security considerations section that justifies the security services claimed. This section should also describe any deficiencies in the security of the security suite.

1.5.5 Additional Information

It may be desirable for each security suite to provide additional information to the implementers and users of the security suite. This information may include efficiency statistics, methods for efficiently implementing the protocols or data elements that should be stored in the PAN information base (PIB). For instance, if public keys or symmetric keys need to be stored by the PNC or by individual DEVs, the security suite should mention these requirements.

1.6 Protocols and State Machines

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:

— Mutual authentication of DEV and security manager (SM)
— Key request
— Key distribution
— Beacon protection
— Data protection

Each of the protocols described in this section may be performed between the PNC and a normal DEV or between two DEVs in a peer-to-peer relationship. If they are performed in the peer-to-peer setting, the initiating DEV shall act in the role of DEV and the receiving 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 that implements both authentication and payload protection.

1.7 Protocol Selection Criteria

The protocols in this document have been selected based on:

— **Time to Market:** The protocols make use of currently available technology.
— **Selectable Components:** The protocol framework must provide flexibility to allow different algorithms to be selected for use in the standard.
— **Flexibility:** The protocols described in this document are designed to meet a large range of security requirements. They should support the various security scenarios identified for 802.15.3 piconets.
— **Market Suitability:** The protocols in this document will be reviewed by 802.15.3 vendors to ensure that they satisfy their requirements.

1.8 High-level Protocol Descriptions

1.8.1 Authentication and Key Establishment Protocol

At the initial stages of the piconet setup, a controller is selected from among the local devices to perform the PNC and security manager roles. The security manager acts as the central security point for all devices to obtain keying material for the piconet. The following diagram shows the authentication protocol between the PNC and another device. In the peer-to-peer scenario, the diagram would be the same except that both devices would receive the beacon transmission from the PNC. One device would act as the security manager while the other would act as normal device, depending on which device initiates the protocol.

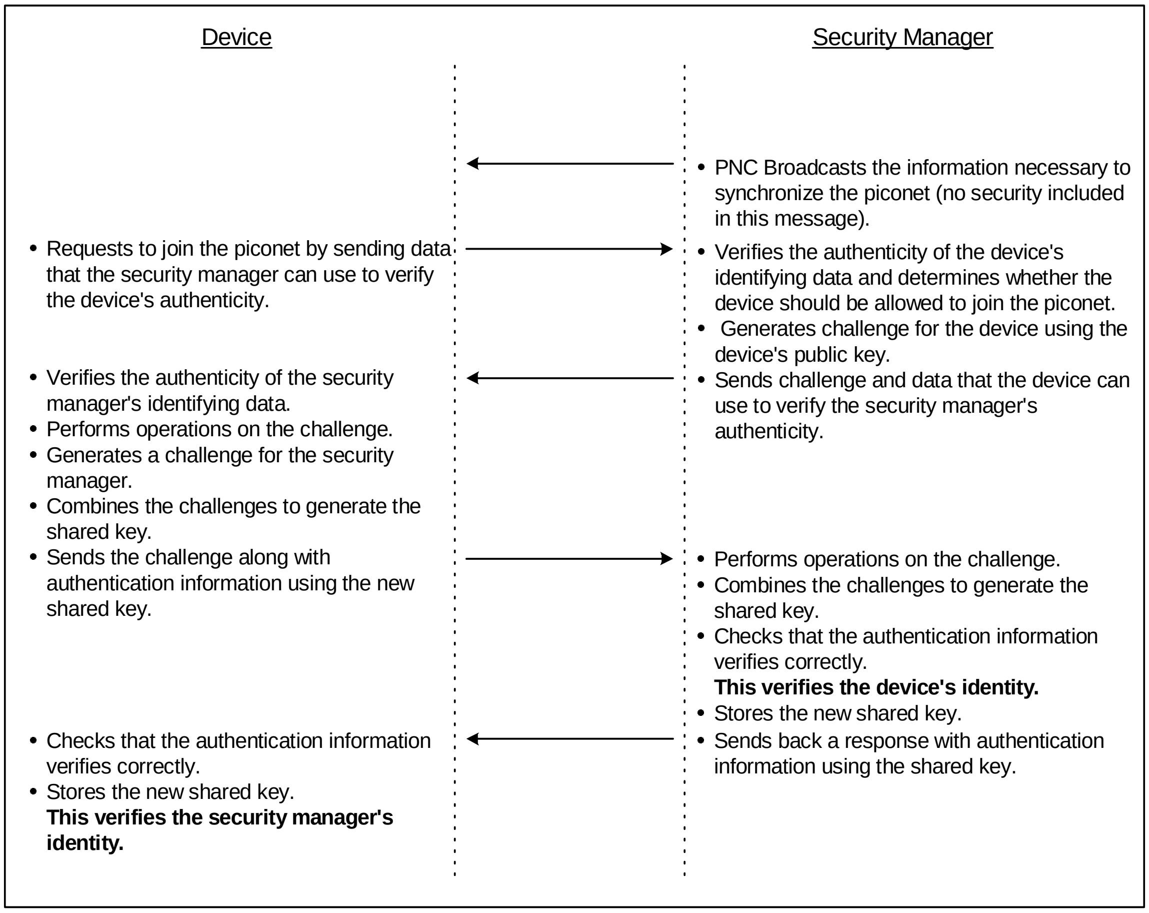
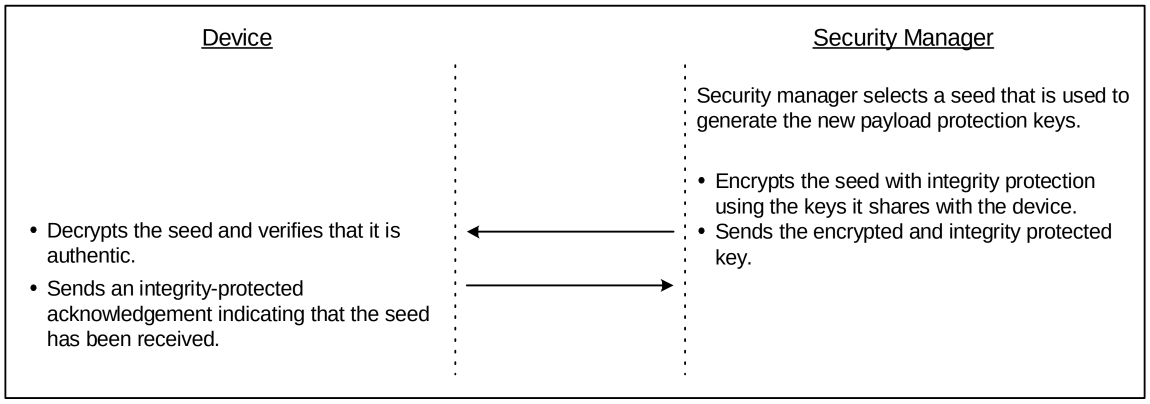

Device
Security Manager
• PNC Broadcasts the information necessary to synchronize the piconet (no security included in this message).
• Requests to join the piconet by sending data that the security manager can use to verify the device's authenticity.
• Verifies the authenticity of the device's identifying data and determines whether the device should be allowed to join the piconet.
• Generates challenge for the device using the device's public key.
• Verifies the authenticity of the security manager's identifying data.
• Performs operations on the challenge.
• Generates a challenge for the security manager.
• Combines the challenges to generate the shared key.
• Sends the challenge along with authentication information using the new shared key.
• Sends challenge and data that the device can use to verify the security manager's authenticity.
• Performs operations on the challenge.
• Combines the challenges to generate the shared key.
• Checks that the authentication information verifies correctly.
This verifies the device's identity.
• Stores the new shared key.
• Checks that the authentication information verifies correctly.
• Stores the new shared key.
This verifies the security manager's identity.
• Sends back a response with authentication information using the shared key.

Figure 1—Authentication protocol overview

After the completion of mutual authentication, the security manager may use the shared keys to transmit the payload protection keys for that security relationship. If the security manager is acting as the PNC, these keys are used to provide confidentiality and integrity protection on all piconet data. If the security manager is acting as a DEV, these keys are used to provide confidentiality and integrity for peer-to-peer communications.

1.8.2 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 may send the new key to each authorized device using the shared secret key agreed upon in the authentication protocol. The PNC needs to store the symmetric keys shared with each device in the piconet in order to distribute a new key, but it does not need to store the public keys of these devices. Each device (other than the PNC) need only store the symmetric keys it shares with the PNC and with any device with which it has a secure peer-to-peer relationship.



Security manager selects a seed that is used to generate the new payload protection keys.

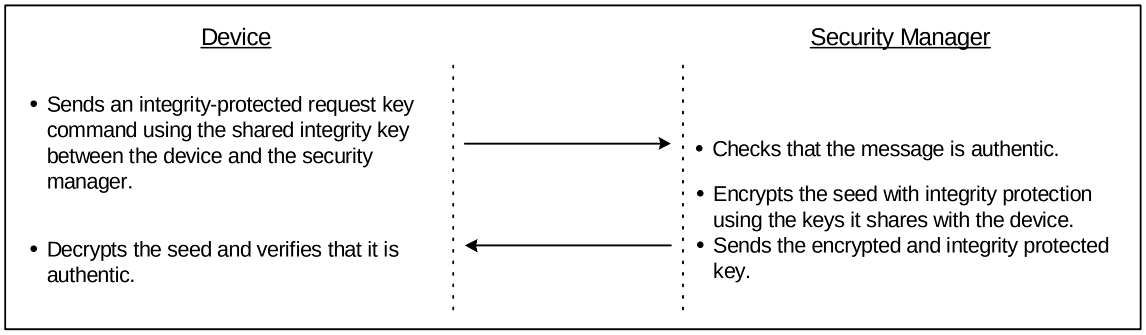
- Encrypts the seed with integrity protection using the keys it shares with the device.
- Sends the encrypted and integrity protected key.

- Decrypts the seed and verifies that it is authentic.
- Sends an integrity-protected acknowledgement indicating that the seed has been received.

Figure 2—Key distribution protocol overview

1.8.3 Key Request Protocol

When the security manager updates the piconet protection key, the device may not have received the new key properly when it wants to start sending or receiving data. When this occurs, the device may request the current key from the security manager.



- Sends an integrity-protected request key command using the shared integrity key between the device and the security manager.

- Checks that the message is authentic.
- Encrypts the seed with integrity protection using the keys it shares with the device.
- Sends the encrypted and integrity protected key.

- Decrypts the seed and verifies that it is authentic.

Figure 3—Key request protocol overview

1.8.4 Beacon Protection Protocol

The beacon protection protocol provides integrity protection (and source authentication) on the beacon from the PNC. This protocol is also used on all commands and acknowledgements that are protected. Each beacon, command and acknowledgement that is protected has a header that includes an identifier of the key that is being used, the source and destination addresses, the frame data and a message authentication code using the integrity key.

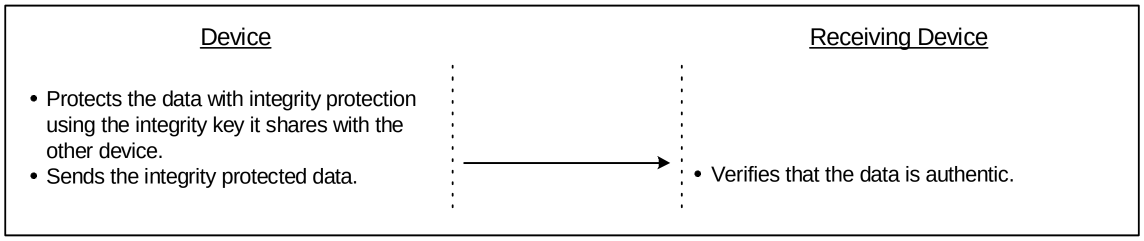

Device
Receiving Device
• Protects the data with integrity protection using the integrity key it shares with the other device.
• Sends the integrity protected data.
• Verifies that the data is authentic.

Figure 4—Beacon protection protocol overview

1.8.5 Data Protection Protocol

Data in the piconet is protected by payload protection keys that provide both privacy and integrity (and source authentication). Each data message that is protected has a header that includes an identifier of the key that is being used, the source and destination addresses, the data encrypted with the encryption key and a message authentication code using the integrity key. These keys are derived from the payload protection seed that is generated by the security manager.

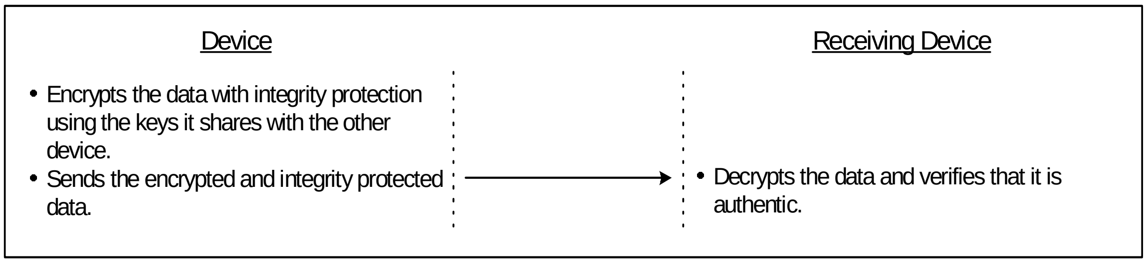

Device
Receiving Device
• Encrypts the data with integrity protection using the keys it shares with the other device.
• Sends the encrypted and integrity protected data.
• Decrypts the data and verifies that it is authentic.

Figure 5—Data protection protocol overview

1.9 Notation

The security protocols defined in this clause were tailored for use in this standard. Each of the objects defined in the protocols are either specified in clause 7 or by the security suite. The mapping of the objects in this standard with the protocol notation is shown by the following table.

Table 1: Notation mapping

Protocol Notation	Clause 7 Transmission Name	Meaning
ID_D	Device Address	The 48-bit IEEE MAC address uniquely identifying the device.
ID_SM	Device Address	The 48-bit IEEE MAC address uniquely identifying the security manager.

Table 1: Notation mapping

PKObj_D[a]	Public-key object type, Public-key object length and Public-key object	The public key object belonging to the device that is transmitted in the authentication request command specified in clause 7.4.1.1{xref}. This contains information to obtain the public key of the device and may contain additional information such as the issuer, issue data, expiry data, the identity of the key owner and key usage.
Pub_D[a]	Not used	The device's public key that is used in the challenge process.
Pr_D[a]	Not used	The device's private key that is used in the challenge process.
PKObj_SM[a]	Public-key object type, Public-key object length and Public-key object	The public key object belonging to the security manager that is transmitted in the challenge request command specified in clause 7.4.1.3{xref}. This contains information to obtain the public key of the device and may contain additional information such as the issuer, issue data, expiry data, the identity of the key owner and key usage.
Pub_SM[a]	Not used	The security manager's public key that is used in the authentication process specified in clause 1.10.1.
Pr_SM[a]	Not used	The security manager's private key that is used in the authentication process specified in clause 1.10.1.
CHAL1[a]	Challenge type, Challenge length and Challenge	The challenge generated by the security manager that is transmitted in the challenge request command specified in clause 7.4.1.3{xref}.
CHAL2[a]	Challenge type, Challenge length and Challenge	The challenge generated by the device that is transmitted in the challenge response command specified in clause 7.4.1.4{xref}.
OID[a]	OID length and OID	The globally unique object identifier that is specified for the security suite and sub-suite.
SSID_D[a]	SSID	The 8-octet random value chosen by the security manager to uniquely identify the management keys used to communicate securely with the device. This SSID is used in the security header for all frames protected with the management keys and is explicitly sent in the challenge request command specified in clause 7.4.1.3{xref}.
SSID_G[a]	SSID	The 8-octet random value chosen by the security manager to uniquely identify the keys used to protect data payloads between the members of the group. This SSID is used in the security header for all frames protected with the payload protection keys and is explicitly sent in the request key response command 7.4.1.6{xref}, distribute key request command 7.4.1.7{xref} and the distribute key response command 7.4.1.8{xref}.
Enc_D[a]	Not used	Symmetric management encryption key associated with a particular SSID_D, to be used for key encryption.
Int_D[a]	Not used	Symmetric management integrity key associated with a particular SSID_D, to be used for management integrity protection.

Table 1: Notation mapping

Seed_D[a]	Not used	The shared secret seed value agreed on during the authentication protocol associated with a particular SSID_D used to generate the encryption key Enc_D and integrity key Int_D.
Seed_G[a] (SymE(Seed_G, Enc_D))[a]	Not used (Encrypted key type, Encrypted key length and Encrypted key)	The random value associated with a particular SSID_G used to generate the encryption key Enc_G and integrity key Int_G. This seed is transmitted in encrypted form in the request key response command 7.4.1.6{xref} and the distribute key request command 7.4.1.7{xref}.
Enc_G[a]	Not used	Symmetric encryption key associated with a particular SSID_G, to be used for payload encryption.
Int_G[a]	Not used	Symmetric management integrity key associated with a particular SSID_G, to be used for payload integrity protection and group message integrity protection.
seq_num_SM	Sequence Counter	4-octet integer in network byte order associated with a particular SSID_D, used to count commands sent by the security manager using that key. The sequence number shall begin counting with 0. This sequence number is used in all commands protected by the management keys sent form the security manager as specified in clause 7.2.3 {xref}.
seq_sum_D	Sequence Counter	4-octet integer in network byte order associated with a particular SSID_D, used to count commands sent by the device using that key. The sequence number shall begin counting with 0. This sequence number is used in all commands protected by the management keys sent form the device as specified in clause 7.2.3 {xref}.
SymE(m, Enc)[a] (SymE(data, Enc_G))	Not used (IV and Encrypted Data)	The result of symmetric encryption of the message *m* with the key *Enc*. This operation is used for payload encryption in data frames as defined in clause 7.2.4 {xref} and in the request key response command 7.4.1.6{xref} and the distribute key request command 7.4.1.7{xref}.
SymI(m, Int)[a]	Not used	The result of calculating a message authentication code on the message *m* with the symmetric key *Int*. If *m* is "…", the message authentication code is computed over all preceding fields in the frame.
H(m)[a]	Not used	The result of a cryptographic hash on the message *m*.
m||n	Not used	The concatenation of two messages *m* and *n*.
Key(m)[a]	Not used	The transformation of a message *m* into the form of a symmetric key.
KeyPurpose[a]	Key purpose	The numerical representation of the use for which the key will be used. This is explicitly sent in the request key command 7.4.1.5{xref}, request key response command 7.4.1.6{xref}, distribute key request command 7.4.1.7{xref} and the distribute key response command 7.4.1.8{xref}.
TimeToken	TimeToken	A strictly increasing fresh value transmitted in each beacon 7.2.1{xref}. This is included in the security header of all secure frames.

Table 1: Notation mapping

AReq	MAC frame header and Command type and Length	The frame header of an authentication request command.
CReq	MAC frame header and Command type and Length	The frame header of a challenge request command.
CRes	MAC frame header and Command type and Length	The frame header of a challenge response command.
ARes	MAC frame header and Command type and Length	The frame header of an authentication response command.
DKReq	MAC frame header	The frame header of a distribute key request command.
DKRes	MAC frame header	The frame header of a distribute key response command.
KRReq	MAC frame header	The frame header of a request key request command.
KRRes	MAC frame header	The frame header of a request key response command.
SDH	MAC frame header	The frame header of a secure data frame.
BH	MAC frame header	The frame header of a secure beacon (or secure ACK or secure command)
BD	Information Elements or Command Payload	The contents of a secure beacon as specified in clause 7.2.1{xref}or a secure command as specified in clause 7.2.3{xref}.
finished1[a]	Challenge response type, Challenge response length and Challenge response	SymI(m, Int) where *m* is the entire set of data in order in the preceding protocol up to the point of the message authentication code and *Int* is Int_D. This data is transmitted in the challenge response command specified in clause 7.4.1.4{xref}
finished2[a]	Auth response type, Auth response length and Auth response	SymI(m, Int) where *m* is the entire set of data in order in the preceding protocol up to the point of the message authentication code and *Int* is Int_D. This data is transmitted in the authentication response command specified in clause 7.4.1.2{xref}

[a]This object is specified by the security suite and sub-suite.

1.10 Protocol Details

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 as specified in clause 7{xref}. The algorithm choices for each operation in the protocols are determined by the selected security suites, which are specified in clause 11{xref}.

The states described refer to a single relationship, rather than an overall state for the device. Therefore it is assumed that if both peer-to-peer relationships and piconet-wide relationships are used, devices may either behave in a multi-threaded fashion or maintain the different states through information stored in the MAC PIB.

1.10.1 Authentication and Key Establishment

This sub-clause specifies the authentication and key establishment protocol that is used for mutual device authentication. In this protocol, the device and the security manager each authenticate each other, agree on a secret (symmetric) key between the two of them and the device obtains the group payload protection keys.

Table 2—Setup for authentication and key establishment

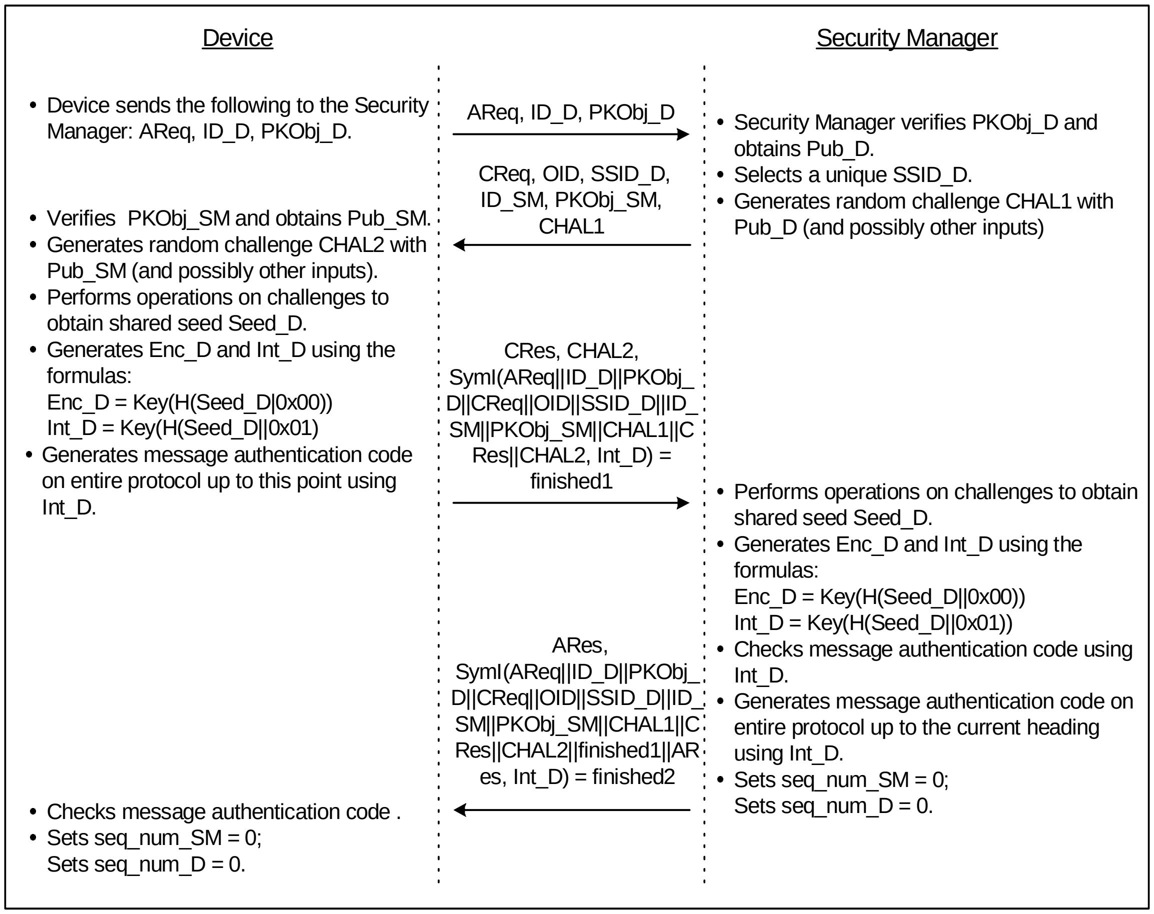
Symbol	Initial Owner	
	Device	Security Manager
PKObj_D	✓	–
Pr_D	✓	–
ID_D	✓	–
Verification information for Pub_SM and ID_SM	✓	–
PKObj_SM	–	✓
Pr_SM	–	✓
ID_SM	–	✓
Verification information for Pub_D and ID_D	–	✓
OID	–	✓

The cryptographic functionality required to implement this protocol is:

Table 3—Capabilities for authentication protocol

Functionality	Required	
	Device	Security Manager
Public-key verification	✓	✓
Public-key challenge generation	✓	✓
Public-key challenge response	✓	✓
Symmetric message authentication code	✓	✓

The following figure shows the authentication protocol.



Device

Security Manager

- Device sends the following to the Security Manager: AReq, ID_D, PKObj_D.

AReq, ID_D, PKObj_D

- Security Manager verifies PKObj_D and obtains Pub_D.
- Selects a unique SSID_D.
- Generates random challenge CHAL1 with Pub_D (and possibly other inputs)

CReq, OID, SSID_D, ID_SM, PKObj_SM, CHAL1

- Verifies PKObj_SM and obtains Pub_SM.
- Generates random challenge CHAL2 with Pub_SM (and possibly other inputs).
- Performs operations on challenges to obtain shared seed Seed_D.
- Generates Enc_D and Int_D using the formulas:
  Enc_D = Key(H(Seed_D|0x00))
  Int_D = Key(H(Seed_D||0x01)
- Generates message authentication code on entire protocol up to this point using Int_D.

CRes, CHAL2, SymI(AReq||ID_D||PKObj_D||CReq||OID||SSID_D||ID_SM||PKObj_SM||CHAL1||CRes||CHAL2, Int_D) = finished1

- Performs operations on challenges to obtain shared seed Seed_D.
- Generates Enc_D and Int_D using the formulas:
  Enc_D = Key(H(Seed_D||0x00))
  Int_D = Key(H(Seed_D||0x01))
- Checks message authentication code using Int_D.
- Generates message authentication code on entire protocol up to the current heading using Int_D.
- Sets seq_num_SM = 0;
  Sets seq_num_D = 0.

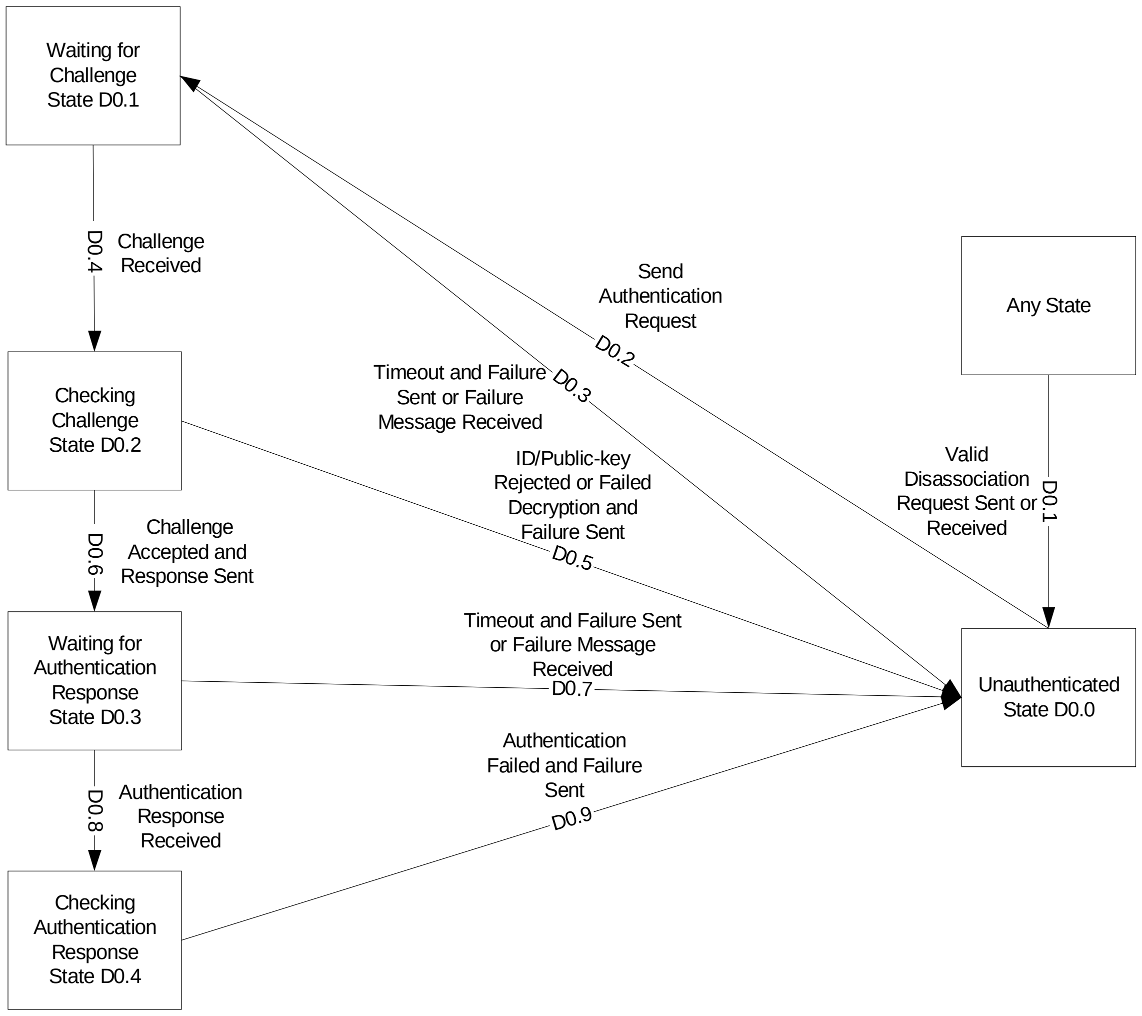
ARes, SymI(AReq||ID_D||PKObj_D||CReq||OID||SSID_D||ID_SM||PKObj_SM||CHAL1||CRes||CHAL2||finished1||ARes, Int_D) = finished2

- Checks message authentication code .
- Sets seq_num_SM = 0;
  Sets seq_num_D = 0.

Figure 6—Authentication Protocol

1.10.1.1 Device States

This figure shows the states and state transitions that apply to the device during the authentication and key exchange protocol.



Waiting for Challenge State D0.1
D0.4 Challenge Received
Checking Challenge State D0.2
D0.6 Challenge Accepted and Response Sent
Waiting for Authentication Response State D0.3
D0.8 Authentication Response Received
Checking Authentication Response State D0.4
Send Authentication Request D0.2
Timeout and Failure Sent or Failure Message Received D0.3
ID/Public-key Rejected or Failed Decryption and Failure Sent D0.5
Timeout and Failure Sent or Failure Message Received D0.7
Authentication Failed and Failure Sent D0.9
Any State
Valid Disassociation Request Sent or Received D0.1
Unauthenticated State D0.0

Figure 7—Authentication state diagram - device perspective

Figure 7 shows the states and the state transitions for the device role in the authentication protocol. The following table describes the device states during the authentication protocol.

Table 4—Device authentication states

State	Name	Description	Action
D0.0	Unauthenticated	Default state for the device; this is also the state returned to if a failure occurs during the authentication protocol or upon disassociation.	Device ignores all security-related commands except the following: • disassociate • PNC handover • distribute info The only state a device can transition to is the "Waiting for Challenge" state.
D0.1	Waiting for Challenge	Device waits for challenge from security manager.	Device ignores all security-related commands except the following: • authentication error • challenge request • disassociate • PNC handover • distribute info
D0.2	Checking Challenge	Device processes the challenge sent by security manager.	Device verifies that the public key and ID hash to the stored value and decrypts the challenge. If both succeed, the device generates own challenge and proof and sends a challenge response command to security manager. If either, or both, do not succeed, the device aborts the authentication process and returns the appropriate error message. This is a transient and processing-only state wherein the device ignores all commands.
D0.3	Waiting for Authentication Response	Device waits for an authentication response from security manager.	Device ignores all security-related commands except the following: • authentication error • authentication response • disassociate • PNC handover • distribute info
D0.4	Checking Authentication Response	Device processes authentication response from security manager.	Device verifies message authentication code calculated on the authentication protocol. If this passes, the device enters secure mode. If this does not pass, the device aborts the authentication process and returns the appropriate error message.

1.10.1.2 Device State Transitions

Figure 7 shows the states and the state transitions for the device role in the authentication protocol. This section describes the processes and causes of the state transitions.

Table 5—Device authentication state transitions

State	Transition	Description
D0.1	Any State to Unauthenticated	At any time during the relationship, the device may choose to disassociate or the security manager may send a disassociate message. If the device has the current key for the relationship, the disassociate message shall be protected. When the disassociation message is sent or received, the relationship key is securely deleted and the device returns to the unauthenticated state.
D0.2	Unauthenticated to Waiting for Challenge	When a device is in the unauthenticated state, it may decide to attempt to authenticate in a piconet or with a peer. For instance, this can occur after the device has associated or directly after a PNC handover. The device generates and sends an authentication request to the current security manager and starts a counter to determine how long it will wait for the challenge command. When this message has been sent and the counter started, the device performs this transition to the waiting for challenge state.
D0.3	Waiting for Challenge to Unauthenticated	When a device is in the waiting for challenge state and receives an authentication error message or has not received a challenge before the timeout, it sends an appropriate authentication error message (if applicable) and performs this transition to the unauthenticated state. Note that the authentication error is not protected by any key.
D0.4	Waiting for Challenge to Checking Challenge	When a device is in the waiting for challenge state and receives a challenge from the security manager, it determines if the message is formatted correctly and comes from the correct device. If the challenge is formatted correctly, it performs this transition to the checking challenge state. Note that the challenge request is not protected by any key.
D0.5	Checking Challenge to Unauthenticated	When a device is in the checking challenge state and determines that the received challenge should be rejected, it sends an appropriate authentication error message and performs this transition to the unauthenticated state.
D0.6	Checking Challenge to Waiting for Authentication	When a device is in the checking challenge state and determines that the challenge passes the checks, it generates and sends a challenge response for the security manager. In addition, it starts a counter to determine how long it will wait for the authentication response command and performs this transition to the waiting for authentication state.
D0.7	Waiting for Authentication to Unauthenticated	When a device is in the waiting for authentication state and receives an authentication error or has not received an authentication response before the timeout, it sends an appropriate authentication error message (if applicable) and performs this transition to the unauthenticated state. Note that the authentication error is not protected by any key.

Table 5—Device authentication state transitions

D0.8	Waiting for Authentication to Checking Authentication Response	When a device is in the waiting for authentication state and receives an authentication response from the security manager, it determines if the message is formatted correctly and comes from the correct device. If the challenge is formatted correctly, it performs this transition to the checking authentication response state. Note that the authentication response is not protected by any key.
D0.9	Checking Authentication Response to Unauthenticated	When a device is in the checking authentication response state and determines that the authentication response should be rejected, it sends an appropriate authentication error message and performs this transition to the unauthenticated state.

1.10.1.3 Security Manager States

The security manager maintains a separate authentication state for each of the devices it is willing to authenticate with. While the security manager is authenticating a specific device, it may perform other operations unrelated to the authentication protocol. In particular, the PNC may authenticate multiple devices in separate threads at the same time. Figure 8 shows the states and the state transitions for the security manager role in the authentication protocol. The following table describes the security manager states during the authentication protocol. The intermediate steps are marked as a critical section to indicate that the PNC may need to perform other functions while waiting for the completion of an authentication protocol.

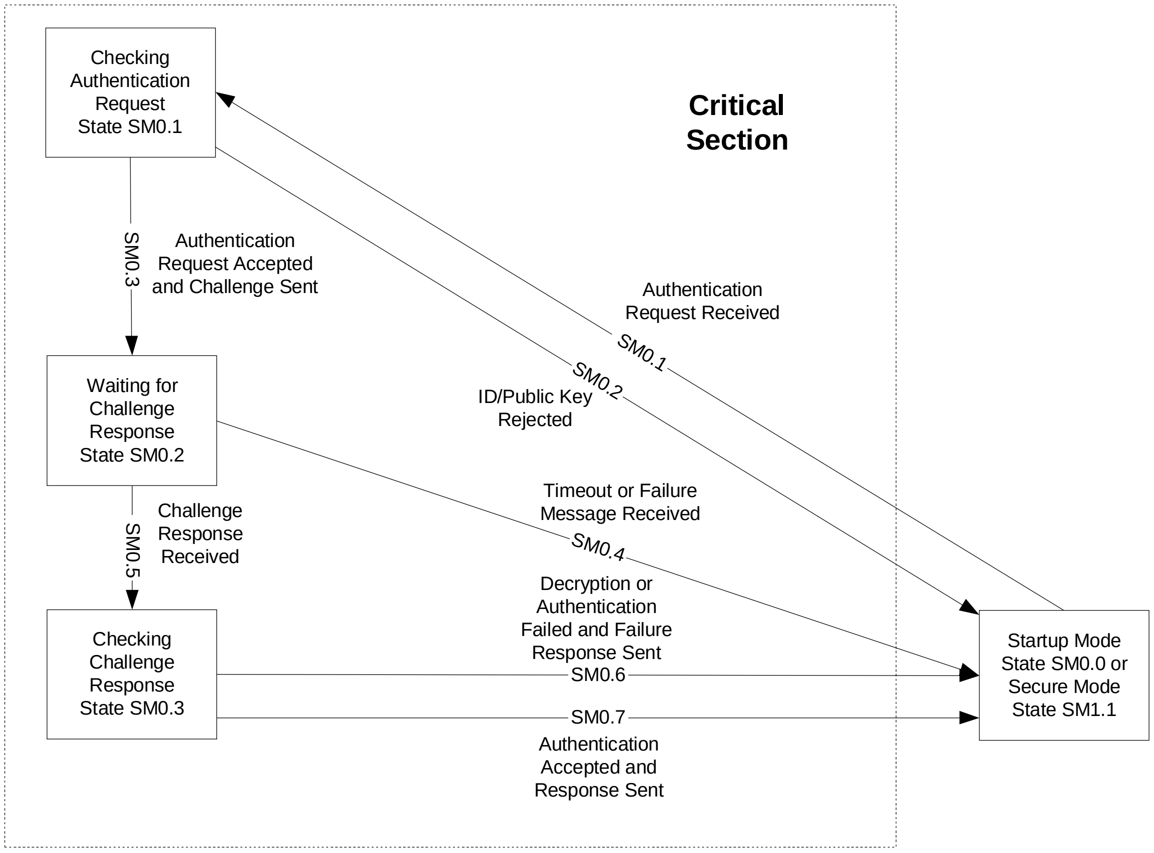

Checking Authentication Request State SM0.1
Critical Section
SM0.3
Authentication Request Accepted and Challenge Sent
Authentication Request Received
SM0.1
SM0.2
ID/Public Key Rejected
Waiting for Challenge Response State SM0.2
Timeout or Failure Message Received
SM0.4
Challenge Response Received
SM0.5
Decryption or Authentication Failed and Failure Response Sent
SM0.6
Checking Challenge Response State SM0.3
SM0.7
Authentication Accepted and Response Sent
Startup Mode State SM0.0 or Secure Mode State SM1.1

Figure 8—Authentication state diagram – security manager perspective

Figure 8 shows the states and the state transitions for the security manager role in the authentication protocol. The following table describes the security manager states during the authentication protocol.

Table 6—Security manager authentication states

State	Name	Description	Action
SM0.0	Startup Mode	Initial state for the security manager when it starts any group. The security manager has not yet sent keys to any other device in the group. • If the relationship is peer-to-peer, this state shall be transitioned out of immediately following successful authentication. • If the security manager is acting as the first PNC of the secure piconet, the security manager can choose to authenticate several devices before distributing the group keys or it may choose to transition out of this state after the first successful authentication. • If the security manager has just completed PNC handover, the device should attempt to authenticate each device on the list of (previously) authenticated devices before transitioning out of this state in order to prevent interruption of service.	Security manager ignores all security-related commands from the device except the following: • disassociate • authentication request • distribute info commands The only state, not in a critical section, the security manager can transition to is the "Pending Key" state.
SM0.1	Checking Authentication Request	A processing-only state in which the device is processing an authentication request. This state is in the critical section and should be considered independent of any non-critical states.	Security manager ignores all commands from the device. Security manager checks that the public key and ID hash to the correct value. • If so, the security manager shall send a challenge request command to the device. • If not, the security manager shall return the appropriate authentication error message.

Table 6—Security manager authentication states

SM0.2	Waiting for Challenge Response	Security manager has sent a challenge to the device and is waiting for a challenge response command. This state is in the critical section and should be considered independent of any non-critical states.	Security manager ignores all security-related commands from the device except the following: • challenge response • authentication error • authentication response • disassociate • distribute info
SM0.3	Checking Challenge Response	A processing-only state in which the security manager is processing a challenge response command.	Security manager decrypts the challenge and checks the device authentication. • If these checks succeed, the security manager sends an authentication response to the device. • If not, the security manager returns the appropriate error message and aborts the authentication protocol. Security manager ignores all commands from the device.

1.10.1.4 Security Manager State Transitions

Figure 8 shows the states and the state transitions for the security manager role in the authentication protocol. The following table describes the security manager states during the authentication protocol. Table 6 shows the security manager states during the authentication and key exchange protocol. This section describes the processes and causes of the state transitions.

Table 7—Security manager authentication state transitions

State	Transition	Description
SM0.1	Startup Mode or Secure Mode to Checking Authentication Request	When a security manager is in the startup mode or secure mode state, it may receive an authentication request from a device. If this request is properly formatted and the security manager is willing to authenticate devices, it accepts the command and performs this transition to the checking authentication request state.
SM0.2	Checking Authentication Request to Startup Mode or Secure Mode	When a security manager is in the checking authentication request state and determines that the authentication request should be rejected, it sends an authentication failure message to the device and performs this transition to return the device to the state it entered the protocol from. If the security manager is performing multi-threading, then this transition may be thought of simply as aborting the authentication thread with this device.
SM0.3	Checking Authentication Request to Waiting for Challenge Response	When a security manager is in the checking authentication request state and determines that the device is authorized to attempt to authenticate to the piconet, it sends a challenge request to the device and starts a counter to determine how long it will wait for the challenge response command. When this message has been sent and the counter started, the device performs this transition to the waiting for challenge response state.

Table 7—Security manager authentication state transitions

SM0.4	Waiting for Challenge Response to Startup Mode or Secure Mode	When a security manager is in the waiting for challenge response state and receives an authentication error message or has not received a challenge response before the timeout, it sends an authentication error message (if applicable) and performs this transition to the state it entered the protocol from. Note that the authentication error message is not protected by any key.
SM0.5	Waiting for Challenge Response to Checking Challenge Response	When a security manager is in the waiting for challenge response state and receives a properly formatted challenge response command, the security manager accepts the command for processing and performs this transition to the checking challenge response state.
SM0.6	Checking Challenge Response to Startup Mode or Secure Mode (1)	When a security manager is in the checking challenge response state and determines that the challenge response should be rejected, it sends an authentication failure message to the device and performs this transition to return the device to the state it entered the protocol from. If the security manager is performing multi-threading, then this transition may be thought of simply as aborting the authentication thread with this device.
SM0.7	Checking Challenge Response to Startup Mode or Secure Mode (2)	When a security manager is in the checking challenge response state and determines that the challenge response is valid, it sends an authentication response command, stores the agreed on key in its table and performs this transition to the state it entered the protocol from.

1.10.1.5 Combined Authentication States

The following figure shows the states of both entities during the authentication protocol and the transitions between states.

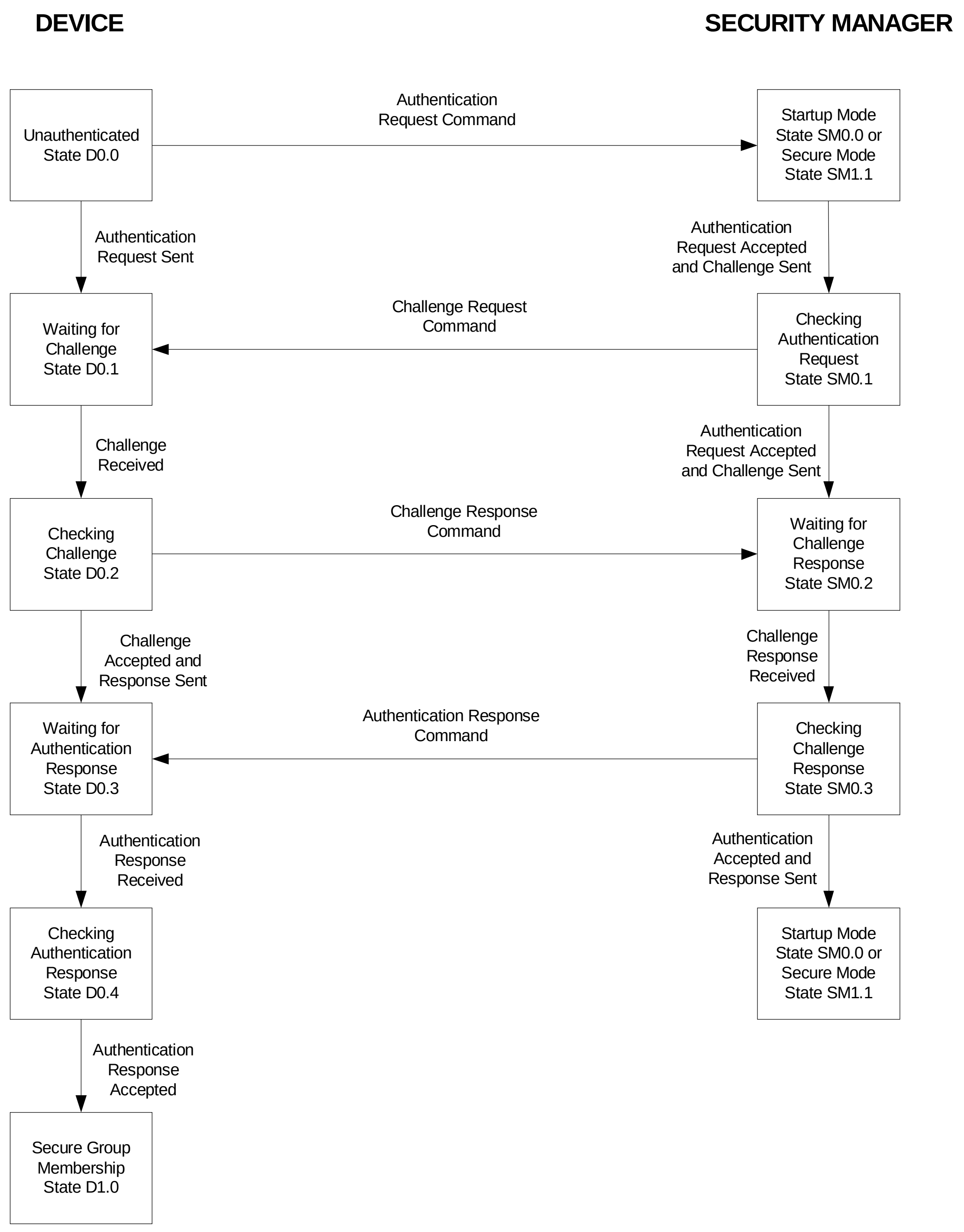

DEVICE
SECURITY MANAGER
Unauthenticated State D0.0
Authentication Request Command
Startup Mode State SM0.0 or Secure Mode State SM1.1
Authentication Request Sent
Authentication Request Accepted and Challenge Sent
Waiting for Challenge State D0.1
Challenge Request Command
Checking Authentication Request State SM0.1
Challenge Received
Authentication Request Accepted and Challenge Sent
Checking Challenge State D0.2
Challenge Response Command
Waiting for Challenge Response State SM0.2
Challenge Accepted and Response Sent
Challenge Response Received
Waiting for Authentication Response State D0.3
Authentication Response Command
Checking Challenge Response State SM0.3
Authentication Response Received
Authentication Accepted and Response Sent
Checking Authentication Response State D0.4
Startup Mode State SM0.0 or Secure Mode State SM1.1
Authentication Response Accepted
Secure Group Membership State D1.0

Figure 9—Successful authentication protocol run

1.10.2 Beacon Protection

In a secure WPAN, the security manager shall use the current group data integrity key to provide integrity protection on the beacon. In addition, each beacon shall include a strictly increasing time token counter that is used by the devices to guarantee freshness. This time token is used in all of the following protocols except for the authentication and key establishment protocol. The beacon protection protocol described here is also used for command protection and ACK protection.

Table 8—Setup for beacon protection

Symbol	Initial Owner	
	Device	Security Manager
Int_G	✓	✓
SSID_G	✓	✓
TimeToken	–	✓

The cryptographic functionality required to implement this protocol is:

Table 9—Capabilities for beacon protection

Functionality	Required	
	Device	Security Manager
Public-key verification	–	–
Public-key encryption	–	–
Public-key decryption	–	–
Symmetric message authentication code	✓	✓

The security manager should initiate this protocol each time it transmits a beacon.

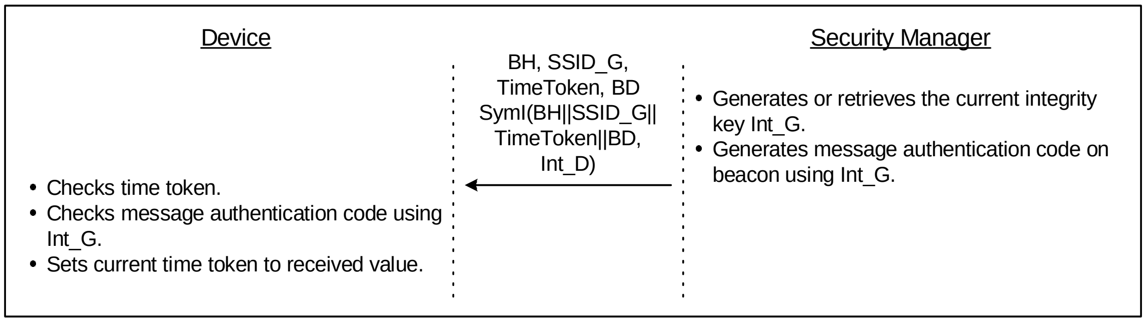

Device
Security Manager
BH, SSID_G, TimeToken, BD SymI(BH||SSID_G|| TimeToken||BD, Int_D)
• Generates or retrieves the current integrity key Int_G.
• Generates message authentication code on beacon using Int_G.
• Checks time token.
• Checks message authentication code using Int_G.
• Sets current time token to received value.

Figure 10—Beacon Protection Protocol

1.10.3 Distribute Key Protocol

Table 10—Setup for key distribution

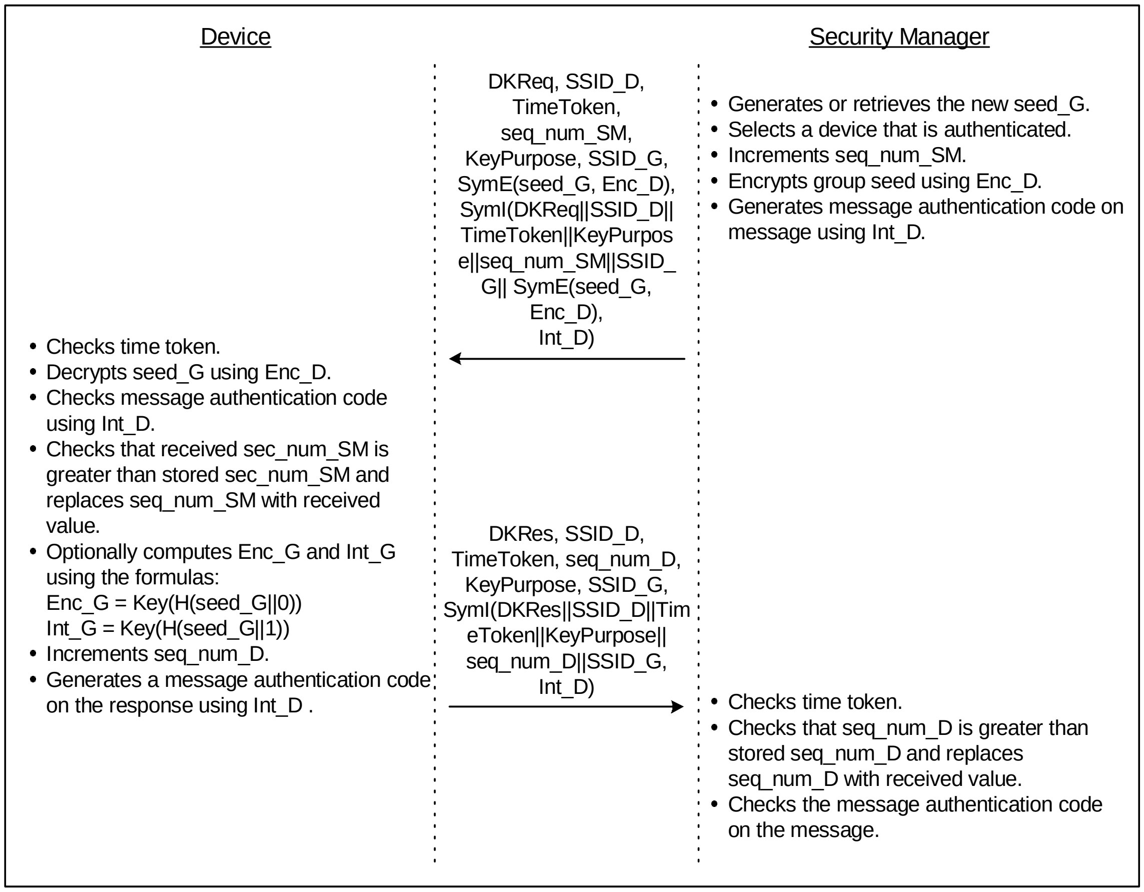
Symbol	Initial Owner	
	Device	Security Manager
Enc_D	✓	✓
Int_D	✓	✓
SSID_D	✓	✓
TimeToken	✓	✓
seq_num_D	✓	✓
seq_num_SM	✓	✓
seed_G (for current key)	–	✓

The cryptographic functionality required to implement this protocol is:

Table 11—Capabilities for key distribution

Functionality	Required	
	Device	Security Manager
Symmetric decryption	✓	–
Symmetric encryption	–	✓
Symmetric message authentication code	✓	✓

The security manager should initiate this protocol with each device with their respective shared keys whenever the key is updated.


Device

Security Manager

- Generates or retrieves the new seed_G.
- Selects a device that is authenticated.
- Increments seq_num_SM.
- Encrypts group seed using Enc_D.
- Generates message authentication code on message using Int_D.

DKReq, SSID_D, TimeToken, seq_num_SM, KeyPurpose, SSID_G, SymE(seed_G, Enc_D), SymI(DKReq||SSID_D||TimeToken||KeyPurpose||seq_num_SM||SSID_G|| SymE(seed_G, Enc_D), Int_D)

- Checks time token.
- Decrypts seed_G using Enc_D.
- Checks message authentication code using Int_D.
- Checks that received sec_num_SM is greater than stored sec_num_SM and replaces seq_num_SM with received value.
- Optionally computes Enc_G and Int_G using the formulas:
  Enc_G = Key(H(seed_G||0))
  Int_G = Key(H(seed_G||1))
- Increments seq_num_D.
- Generates a message authentication code on the response using Int_D .

DKRes, SSID_D, TimeToken, seq_num_D, KeyPurpose, SSID_G, SymI(DKRes||SSID_D||TimeToken||KeyPurpose|| seq_num_D||SSID_G, Int_D)

- Checks time token.
- Checks that seq_num_D is greater than stored seq_num_D and replaces seq_num_D with received value.
- Checks the message authentication code on the message.

Figure 11—Distribute Key Protocol

1.10.4 Key Request Protocol

Table 12—Setup for Key Request

Symbol	Initial Owner	
	Device	Security Manager
Enc_D	✓	✓
Int_D	✓	✓
SSID_D	✓	✓
TimeToken	✓	✓
seq_num_D	✓	✓
seq_num_SM	✓	✓
seed_G (for current key)	–	✓

The cryptographic functionality required to implement this protocol is:

Table 13—Capabilities for Key Request

Functionality	Required	
	Device	Security Manager
Symmetric decryption	✓	–
Symmetric encryption	–	✓
Symmetric message authentication code	✓	✓

The device should initiate this protocol with the security manager if it is already authenticated, but does not have the current payload protection key.

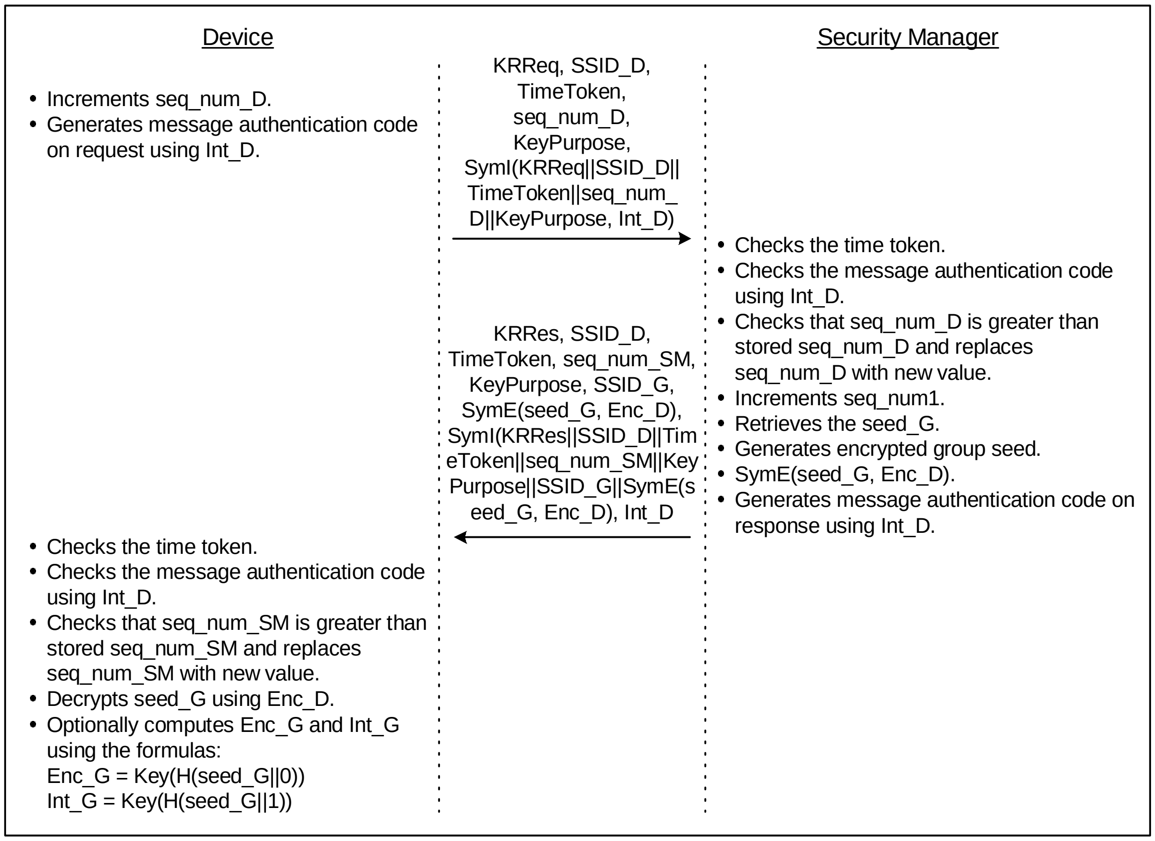

Device
Security Manager
• Increments seq_num_D.
• Generates message authentication code on request using Int_D.
KRReq, SSID_D, TimeToken, seq_num_D, KeyPurpose, SymI(KRReq||SSID_D|| TimeToken||seq_num_ D||KeyPurpose, Int_D)
• Checks the time token.
• Checks the message authentication code using Int_D.
• Checks that seq_num_D is greater than stored seq_num_D and replaces seq_num_D with new value.
• Increments seq_num1.
• Retrieves the seed_G.
• Generates encrypted group seed.
• SymE(seed_G, Enc_D).
• Generates message authentication code on response using Int_D.
KRRes, SSID_D, TimeToken, seq_num_SM, KeyPurpose, SSID_G, SymE(seed_G, Enc_D), SymI(KRRes||SSID_D||TimeToken||seq_num_SM||KeyPurpose||SSID_G||SymE(seed_G, Enc_D), Int_D
• Checks the time token.
• Checks the message authentication code using Int_D.
• Checks that seq_num_SM is greater than stored seq_num_SM and replaces seq_num_SM with new value.
• Decrypts seed_G using Enc_D.
• Optionally computes Enc_G and Int_G using the formulas:
Enc_G = Key(H(seed_G||0))
Int_G = Key(H(seed_G||1))

Figure 12—Key Request Protocol

1.10.4.1 Device States

The following figure shows the states and state transitions that apply to the device during the distribute key and key request protocols.

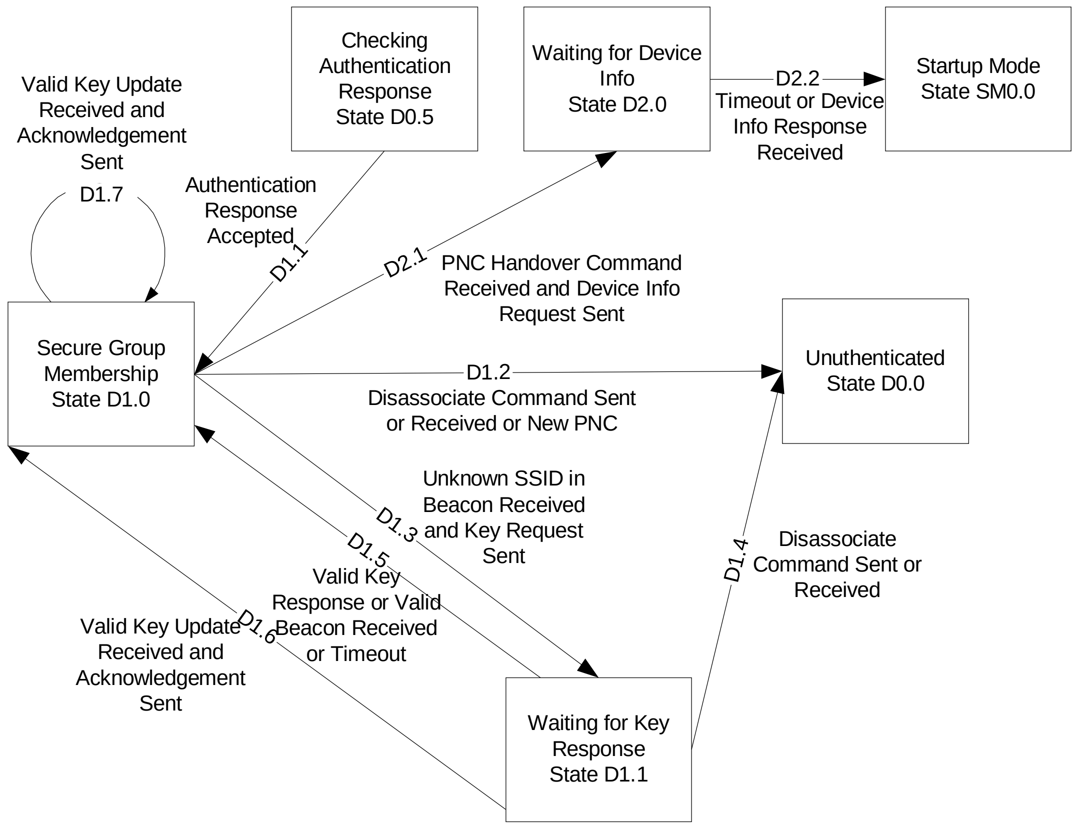

Checking Authentication Response State D0.5
Waiting for Device Info State D2.0
Startup Mode State SM0.0
D2.2 Timeout or Device Info Response Received
Valid Key Update Received and Acknowledgement Sent
D1.7
Authentication Response Accepted
D1.1
D2.1
PNC Handover Command Received and Device Info Request Sent
Secure Group Membership State D1.0
D1.2 Disassociate Command Sent or Received or New PNC
Unauthenticated State D0.0
Unknown SSID in Beacon Received and Key Request Sent
D1.3
D1.5
Valid Key Response or Valid Beacon Received or Timeout
D1.4
Disassociate Command Sent or Received
Valid Key Update Received and Acknowledgement Sent
D1.6
Waiting for Key Response State D1.1

Figure 13—Key management state diagram – device perspective

Figure 13 shows the states and the state transitions for the device role for key management. The following table describes the key management device states.

Table 14—Device key management states

State	Name	Description	Action
D1.0	Secure Group Membership	State in which the device is authenticated and may actively participate in the secure piconet.	Device may accept secure data communications, secure disassociate commands, distribute info commands, PNC handover commands and distribute key requests.
D1.1	Waiting for Key Response	Device is waiting to receive a key and is unable to verify the validity of the beacon.	Device ignores all commands except the following: • distribute key • distribute info commands • key response • disassociate
D2.0	Waiting for Device Info	Device has been selected as the next PNC and is waiting to receive information from the old PNC about the authenticated devices in the piconet before the transition to PNC.	Device ignores all commands except a secure device information response command.

1.10.4.2 Device State Transitions

Figure 13 shows the states and the state transitions for the device role for key management functions. Table 14 shows the key management related device states. This section describes the processes and causes of the state transitions.

Table 15—Device key management state transitions

State	Transition	Description
D1.1	Checking Authentication Response to Secure Group Membership	When a device is in the checking authentication response state and determines that the authentication response should be accepted, it sends an authentication acknowledgement to the security manager and performs this transition to the secure group membership state.
D1.2	Secure Group Membership to Unauthenticated	When a device is in the secure group membership state and receives a disassociate command from the PNC or the DME or receives a new PNC command, the device securely deletes the shared keys with the security manager and performs this transition to the unauthenticated state.
D1.3	Secure Group Membership to Waiting for Key Response	When a device is in the secure group membership state and receives a beacon with a security session ID that is unfamiliar to it, it sends a key request command to the security manager, starts a counter to determine how long it will wait for the key response command and performs this transition to the waiting for key response state.
D1.4	Waiting for Key Response to Unauthenticated	When a device is in the waiting for key response state and receives a disassociate command from the security manager or the DME, the device sends the disassociate command (if applicable), securely deletes its shared keys with the security manager and performs this transition to the unauthenticated state.
D1.5	Waiting for Key Response to Secure Group Membership (1)	When a device is in the waiting for key response state and times out, receives a valid key response command or receives a beacon with a recognizable security session ID, the device updates its current key (if applicable) and performs this transition to the secure group membership state.
D1.6	Waiting for Key Response to Secure Group Membership (2)	When a device is in the waiting for key response state and receives a valid distribute key command from the security manager, the device updates its current key and performs this transition to the secure group membership state.
D1.7	Secure Group Membership to Secure Group Membership	When a device is in the secure group membership state and receives a valid distribute key command, the device shall update the key, send a distribute key response command and remain in the secure group membership state.

Table 15—Device key management state transitions

D2.1	Secure Group Membership to Waiting for Device Info	When a device that is an alternate PNC is in secure group membership mode and receives a secure PNC handover command, it sends a device information request command for the whole piconet to the PNC (security manager), sets the timeout to be the appropriate value and performs this transition to the waiting for device info state.
D2.2	Waiting for Device Info to Startup Mode	When a device is in the waiting for device info state and receives a device information response command or times out, the device shall update its device information table (if applicable) and performs this transition to the startup mode, which is a security manager state. At this point, the device takes on the role of security manager and PNC.

1.10.4.3 Security Manager States

The following figure shows the states and state transitions that apply to the security manager during the distribute key and key request protocols.

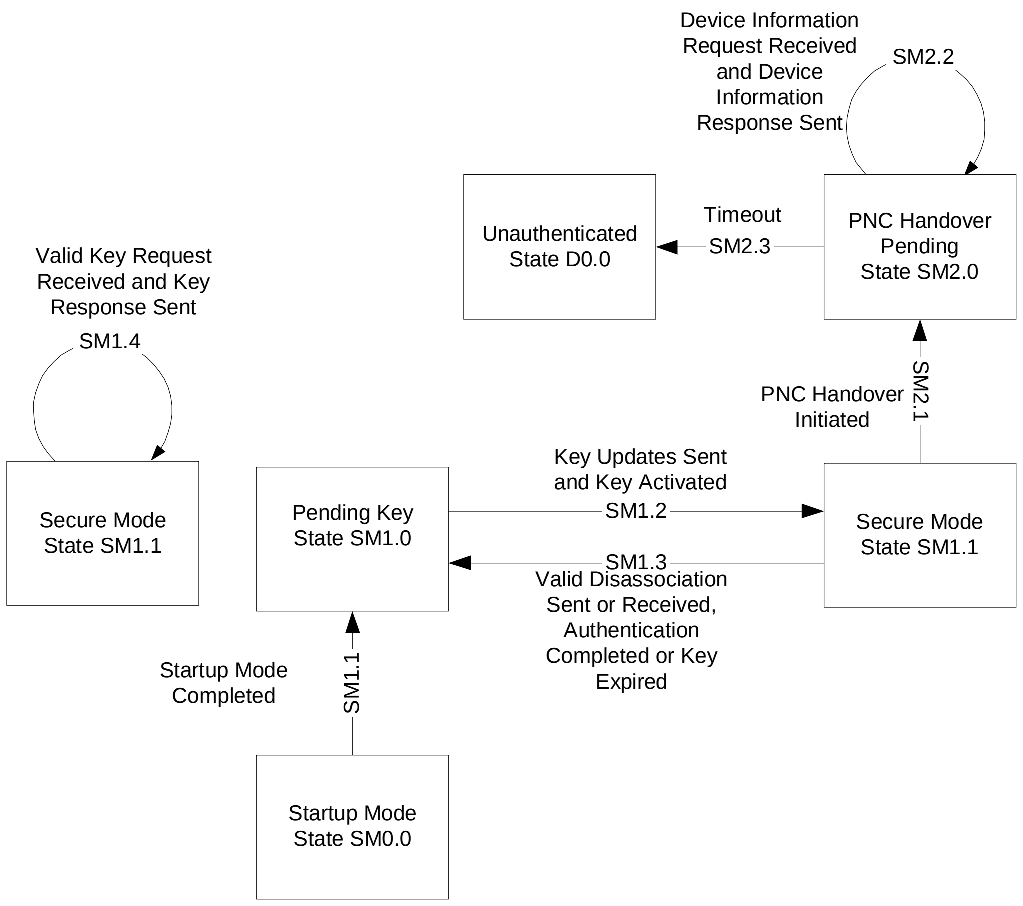

Device Information Request Received and Device Information Response Sent
SM2.2
Unauthenticated State D0.0
Timeout
SM2.3
PNC Handover Pending State SM2.0
Valid Key Request Received and Key Response Sent
SM1.4
PNC Handover Initiated
SM2.1
Key Updates Sent and Key Activated
SM1.2
Secure Mode State SM1.1
Pending Key State SM1.0
Secure Mode State SM1.1
SM1.3
Valid Disassociation Sent or Received, Authentication Completed or Key Expired
Startup Mode Completed
SM1.1
Startup Mode State SM0.0

Figure 14—Key management state diagram – security manager perspective

Figure 14 shows the states and the state transitions for the security manager role for key management. The following table describes the key management security manager states during.

Table 16—Security manager key management state diagrams

State	Name	Description	Action
SM1.0	Pending Key	Security manager generates new group keys and sends distribute key commands to each of the authenticated devices 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 mode state.	Security manager shall accept all valid commands.
SM1.1	Secure Mode	Default state for the security manager in a secure piconet.	Security manager shall accept all valid commands.
SM2.0	PNC Handover Pending	PNC has already sent a PNC handover command and is waiting for the timeout to complete the PNC handover. After the timeout, the security manager will transition to the unauthenticated state, which is a device state.	Security manager shall only accept device information request commands from the next PNC.

1.10.4.4 Security Manager State Transitions

Table 16 shows the security manager states relating to key management and PNC handover. This section describes the processes and causes of the state transitions.

Table 17—Security manager key management state transitions

State	Transition	Description
SM1.1	Startup Mode to Pending Key (1)	When a security manager is in startup mode and completes all of the authentication protocols for the startup process, the security manager performs this transition to the pending key state.
SM1.2	Pending Key to Secure Mode	When a security manager is in pending key state, has generated a new group key for the security relationship and has sent distribute key commands to each of the authenticated devices in the group, the security manager shall change the beacon to include the security session key of the new key (if it is the PNC) and perform this transition to the secure mode state.
SM1.3	Secure Mode to Pending Key	When a security manager is in secure mode and a device is disassociated, a new device is authenticated or upon instruction by the DME, the security manager shall prepare to update the key and perform this transition to the pending key state.
SM1.4	Secure Mode to Secure Mode	When the security manager is in secure mode and receives a valid key request command from an authenticated device, the security manager shall send a key response command to the device and remain in the secure mode state.

Table 17—Security manager key management state transitions

SM2.1	Secure Mode to PNC Handover Pending	When the PNC is in secure mode and receives a command from the DME to perform PNC handover, the PNC sends a PNC handover command to an appropriate device and performs this transition to the PNC handover pending state.
SM2.2	PNC Handover Pending to PNC Handover Pending	When the PNC is in PNC handover pending mode and receives a valid device information request from the next PNC, the security manager shall send a device information response and remain in the PNC handover pending state.
SM2.3	PNC Handover Pending to Unauthenticated	When the PNC is in PNC handover pending mode and the specified time for PNC handover completion has occurred, the PNC ceases sending beacons and performs the transition to the unauthenticated state, which is a device state.

1.10.4.5 Combined Key Request States

The following figure shows the states of both entities during the key request protocol and the transitions between states.

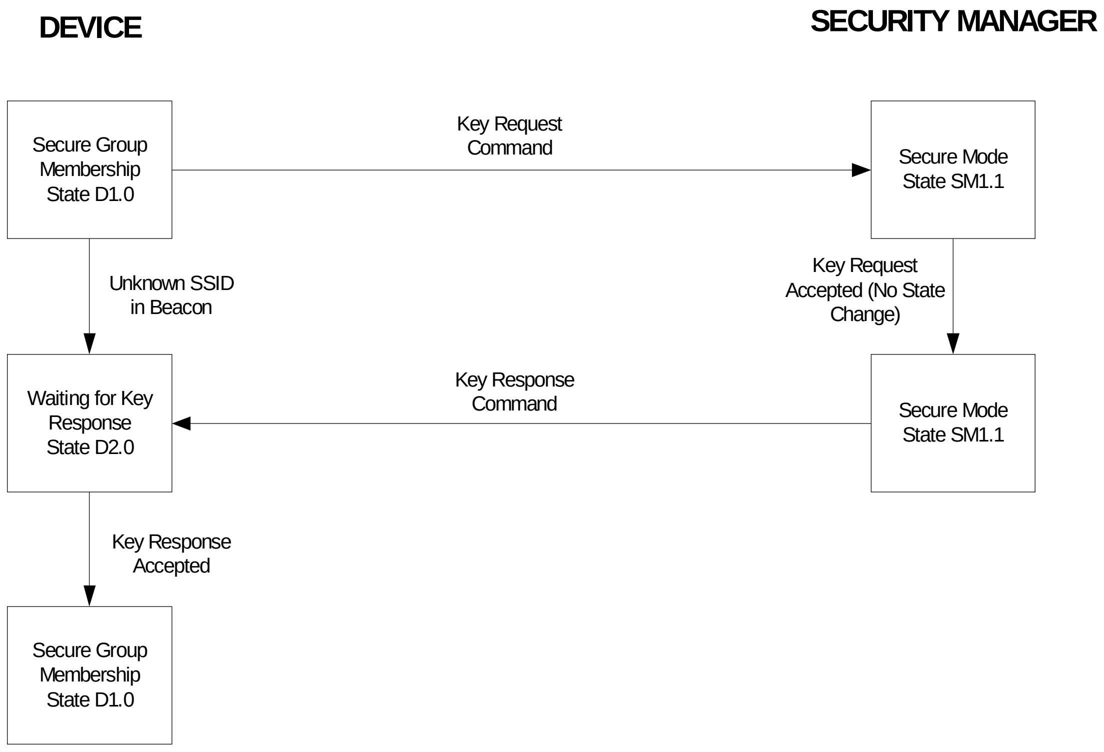


Figure 15—Successful key request protocol run

1.10.4.6 Combined Key Distribution States

The following figure shows the states of both entities during the key distribution protocol and the transitions between states.

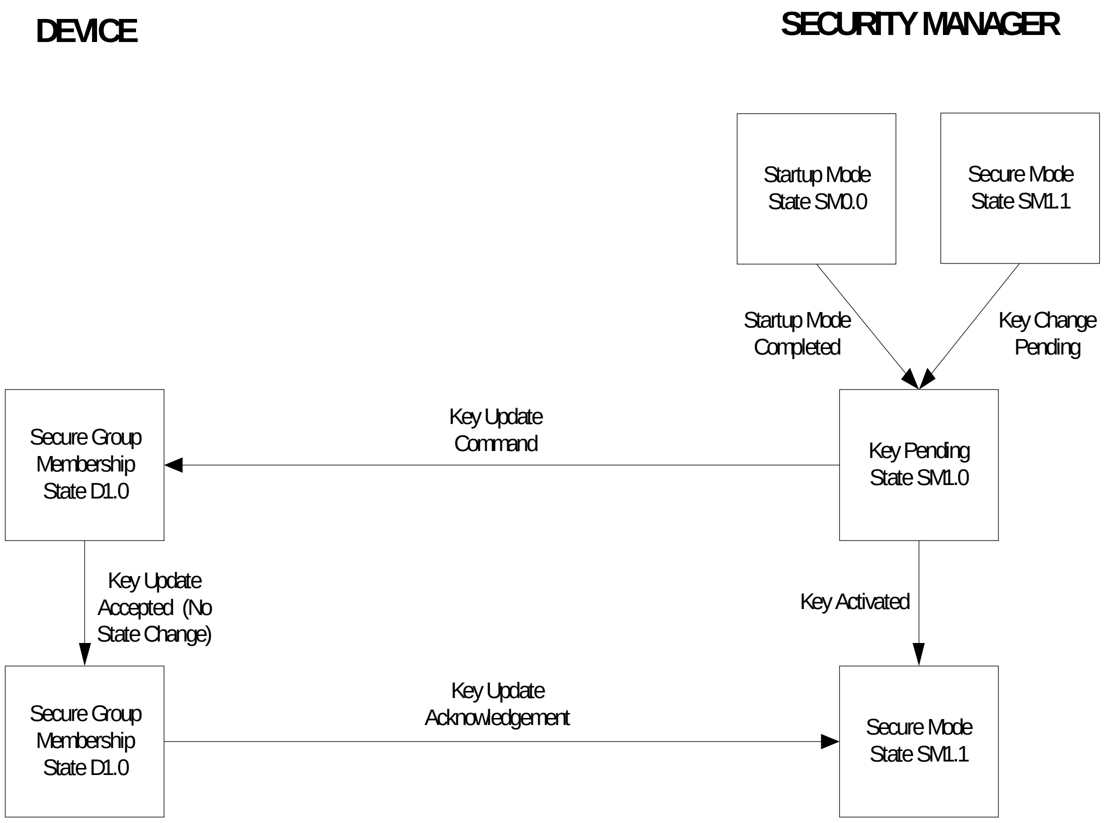


Figure 16—Successful key distribution protocol run

1.10.4.7 Combined PNC Handover States

The following figure shows the states of both entities during the PNC handover protocol and the transitions between states. Other devices transition to the unauthenticated state after the device ID in the beacon is modified to indicate the new PNC.

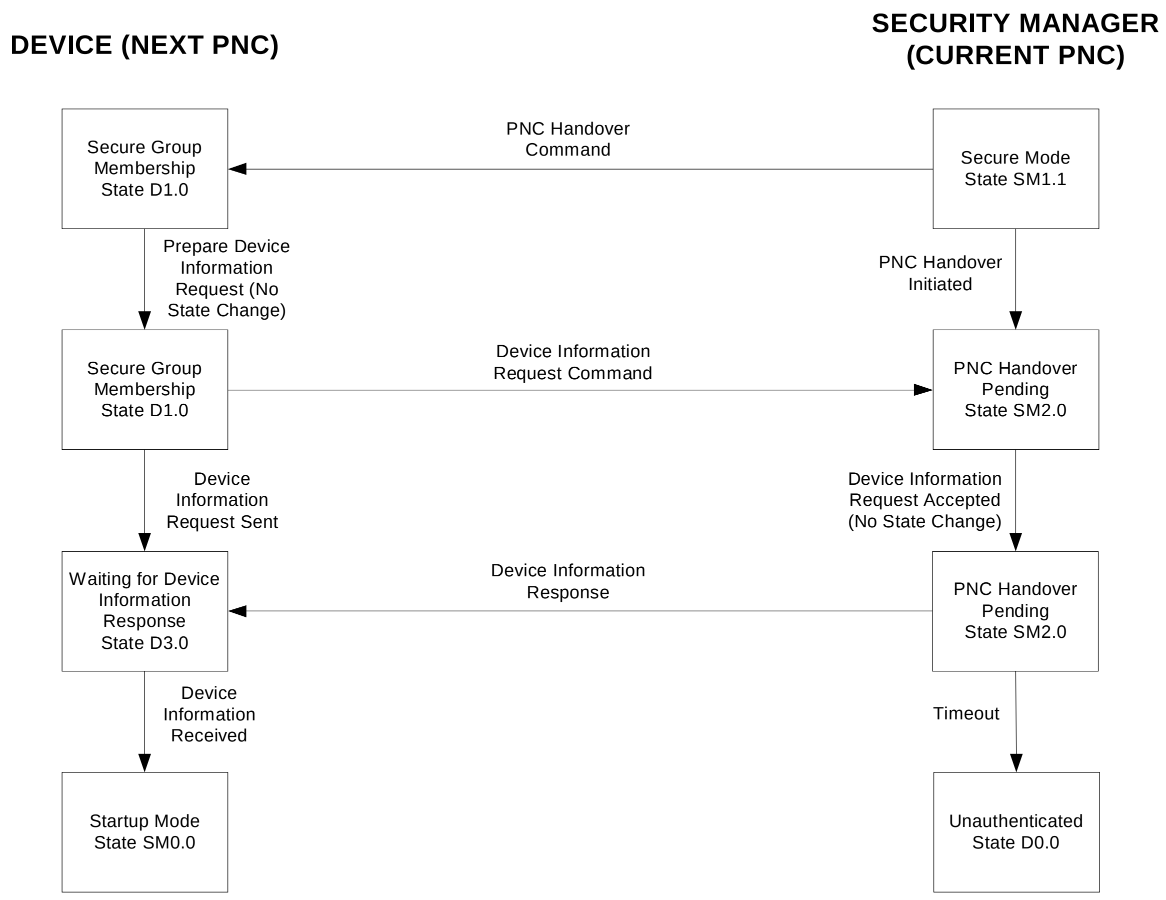

DEVICE (NEXT PNC)
SECURITY MANAGER (CURRENT PNC)
Secure Group Membership State D1.0
PNC Handover Command
Secure Mode State SM1.1
Prepare Device Information Request (No State Change)
PNC Handover Initiated
Secure Group Membership State D1.0
Device Information Request Command
PNC Handover Pending State SM2.0
Device Information Request Sent
Device Information Request Accepted (No State Change)
Waiting for Device Information Response State D3.0
Device Information Response
PNC Handover Pending State SM2.0
Device Information Received
Timeout
Startup Mode State SM0.0
Unauthenticated State D0.0

Figure 17—Successful PNC handover

1.10.5 Data Protection Protocol

Table 18—Setup for data protection protocol

Symbol	Initial Owner	
	Sending Device	Receiving Device
data	✓	–
seed_G - OR - Enc_G AND Int_G, SSID_G, TimeToken	✓	✓

The cryptographic functionality required to implement this protocol is:

Table 19—Capabilities for data protection protocol

Functionality	Required	
	Sending Device	Receiving Device
Symmetric decryption	–	✓
Symmetric encryption	✓	–
Symmetric message authentication code	✓	✓

The sending device may be acting as a normal device or a security manager for the particular key. In either case, the key is mutually shared between all members of the group.

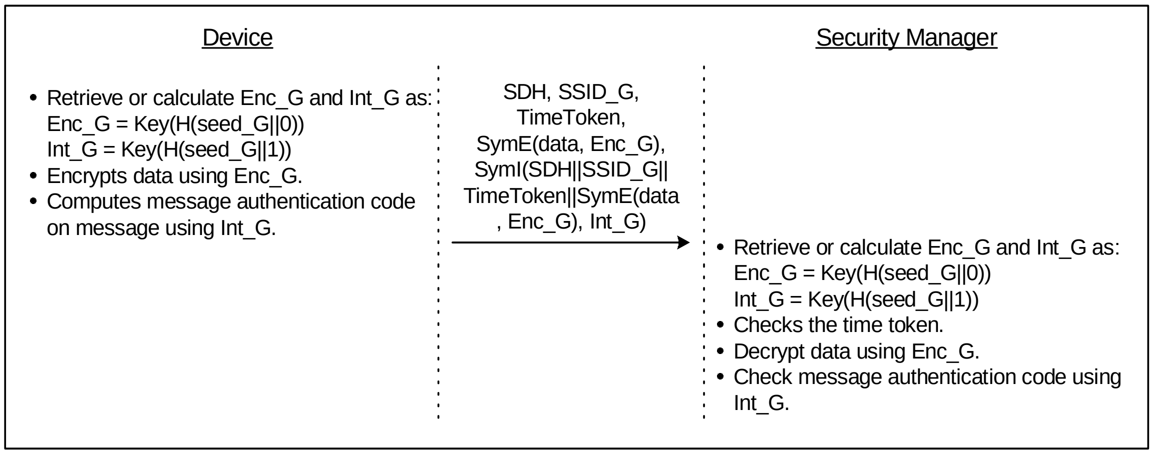

Device
Security Manager
• Retrieve or calculate Enc_G and Int_G as: Enc_G = Key(H(seed_G||0)) Int_G = Key(H(seed_G||1))
• Encrypts data using Enc_G.
• Computes message authentication code on message using Int_G.
SDH, SSID_G, TimeToken, SymE(data, Enc_G), SymI(SDH||SSID_G||TimeToken||SymE(data, Enc_G), Int_G)
• Retrieve or calculate Enc_G and Int_G as: Enc_G = Key(H(seed_G||0)) Int_G = Key(H(seed_G||1))
• Checks the time token.
• Decrypt data using Enc_G.
• Check message authentication code using Int_G.

Figure 18—Data protection protocol

1.10.5.1 Device States

Figure 19 shows the states and state transitions that apply to the device when it receives secure data packets.

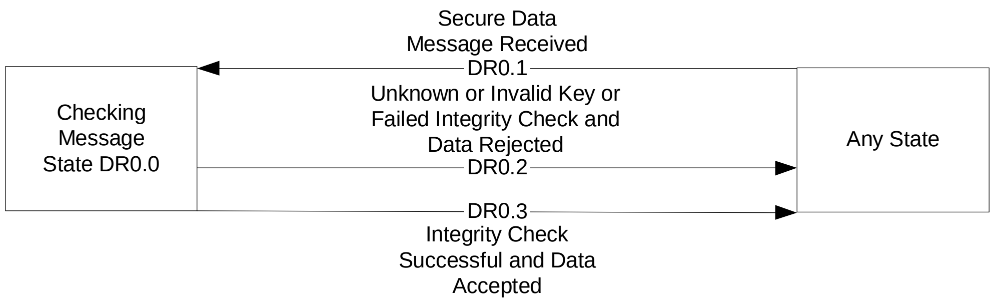

Secure Data Message Received
DR0.1
Unknown or Invalid Key or Failed Integrity Check and Data Rejected
DR0.2
DR0.3
Integrity Check Successful and Data Accepted
Checking Message State DR0.0
Any State

Figure 19—Secure data reception state diagram – device perspective

The following table describes the device states when receiving secure data frames.

Table 20—Device secure data reception states

State	Name	Description	Action
DR0.0	Checking Message	A processing-only state in which the device processes the data message to determine what to do with it. The device checks that a valid time token is in the data and that the MAC verifies with a valid key. • If any of these checks fail, the data is discarded and the device returns to the previous state. • If all of the checks pass, the data is accepted as secure data and the device returns to the previous state.	Device rejects all commands.

1.10.5.2 Device State Transitions

Figure 19 shows the device states and transitions relating to secure data reception. The following table describes the device state transitions while receiving secure data frames.

Table 21—Device secure data reception state transitions

State	Transition	Description
DR0.1	Any State to Checking Message	At any time while a device is associated with a security manager and possesses payload protection keys, the device may receive a secure data message. If the message is properly formatted and contains a valid time token and a known key, the device begins processing of the data and performs the transition to the checking message state.
DR0.2	Checking Message to Any State (1)	When a device in the checking message state determines that the message authentication code on the data is not valid, the data is rejected by the device and the device transitions back to the previous state.
DR0.3	Checking Message to Any State (2)	When a device in the checking message state determines that the message authentication code on the data is valid, the data is decrypted and accepted by the device and the device transitions back to the previous state.

1.10.5.3 Security Manager States

Secure data reception state diagram – security manager perspective shows the states and state transitions that apply to the security manager when it receives secure data packets:

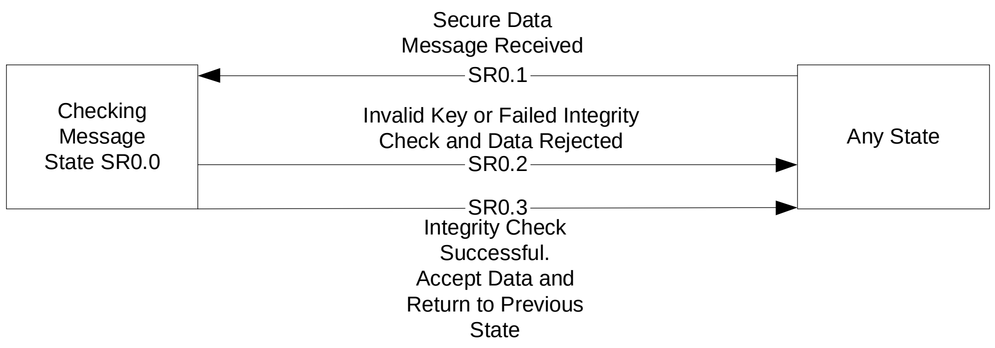

Secure Data Message Received
SR0.1
Checking Message State SR0.0
Invalid Key or Failed Integrity Check and Data Rejected
SR0.2
Any State
SR0.3
Integrity Check Successful. Accept Data and Return to Previous State

Figure 20—Secure data reception state diagram – security manager perspective

Figure 20 shows the states and the state transitions for the security manager role while receiving secure data frames. The following table describes the security manager states while receiving secure data frames.

Table 22—Security manager secure data reception states

State	Name	Description	Action
SR0.0	Checking Message	A processing-only state in which the security manager processes the data message to determine what to do with it. The device checks that the correct time token is in the data and that the MAC verifies with the current key. • If any of these checks fail, the data is discarded and the security manager returns to the previous state. • If all of the checks pass, the data is accepted as secure data and the device returns to the previous state.	Security manager rejects all commands.

1.10.5.4 Security Manager State Transitions

Figure 20 shows the security manager states and transitions relating to secure data reception. This section describes the processes and causes of the state transitions.

Table 23—Security manager secure data reception state transitions

State	Transition	Description
SR0.1	Any State to Checking Message	At any time when there are active payload protection keys, the security manager may receive a secure data message. If the message is properly formatted and contains the correct time token and the current key, the security manager begins processing of the data and performs the transition to the checking message state.
SR0.2	Checking Message to Any State (1)	When the security manager is in the checking message state and determines that the message authentication code on the data is not valid, the data is rejected by the security manager and the security manager transitions back to the previous state.
SR0.3	Checking Message to Any State (2)	When a security manager in the checking message state determines that the message authentication code on the data is valid, the data is decrypted and accepted by the security manager and the security manager transitions back to the previous state.

2. Security Suite Specifications

This clause specifies the security suites that may be used when security is implemented in the piconet. A security suite defines the algorithms and operations that shall be performed when that security suite is selected by the security manager in a security relationship. Within a piconet, the PNC shall choose a security suite that shall be used by all devices for authentication and payload protection for the piconet. Devices that engage in peer-to-peer security relationships may use a different security suite than that being used for piconet protection.

2.1 Modes for security suites

When a security suite is selected, devices may perform secure operations in one of two modes: mode 2 or mode 3. These modes are defined in sub-clause 9.4{xref}. For any given mode, there may be sub-suites defined that complete the specification of the security operations for the authentication protocol. For example, a security suite in Mode 3 may support different certificate formats. The remaining processes are the same regardless of the mode or sub-suite that the devices are operating in. Security suites are not defined for modes 0 and 1.

2.1.1 Determining the suite and mode

The security manager in any security relationship selects an OID that corresponds to the security suite, mode and sub-suite that it will use. In the associate response command or in a subsequent probe command, the security manager may send the desired OID for the security suite, mode and sub-suite to a device to assist that device in selecting a public key for the authentication process. If the device sends a PublicKeyObject in an authentication request that does not correspond to the OID selected by the security manager, the security manager shall reject the authentication request and return a failed authentication response command indicating that the public key was not accepted. If the PublicKeyObject is of the correct form, the security manager explicitly specifies the OID for the security relationship in the challenge response command.

2.1.2 Public-key Verification in Mode 2

In Mode 2, a transmitted PublicKeyObject shall be a raw public key, as defined by the security suite. While in this mode, the DME is responsible for determining if the key is to be trusted and if subsequent authentication requests are to proceed. The DME indicates acceptance of a public key by inserting a hash of the Device ID and public key into the MAC PIB. When a DEV receives an MLME-Authenticate.Request, it proceeds only if the MAC PIB indicates the DME has accepted the Device ID and public key. The method used by the MLME to verify the validity of a public key and device address are specified by the security suite. The mechanism that the DME uses to determine the authorization rights of the devices is out of scope.

2.1.3 Public-key Verification in Mode 3

In Mode 3, a transmitted PublicKeyObject shall be a public-key certificate, as defined by the security suite, mode, and sub-suite. The public-key certificate contains information that binds the public key to the desired identity using the public key of a certificate authority. For any given security suite there may be more than one type of certificate that can be used in mode 3. The certificate type is uniquely identified by selection of a sub-suite. For each sub-suite, an OID is selected to define the sub-suite and the security suite shall define the operations that shall be performed on the certificate within the MLME.

2.2 Security suite selections

2.2.1 OID selections

This clause specifies a list of approved security suites that may be used by compliant implementations. Each security suite is identified by a globally unique OID. All of the OIDs in this document are built off of the following arc:

```
id-802-15-3-security-suites OBJECT IDENTIFIER ::= {
     iso(1) std(0) iso8802(8802) ieee802dot15(15)
     ieee802dot15dot3 (3) securitysuites(1)}
```

[[The following 2 tables represent what the tables would look like if the ECIES security suite text as proposed in this document is selected. If a different security suite is selected, these tables should be changed to reflect the new security suite.]]

The following table specifies the OID for each of the security suites specified in this clause

Table 24—Security suites

Security Suite Name	OID Name	OID Number	DER Encoding
ECIES 256-prime-1	ecies-sec-suite-1	id-802-15-3-security-suites 1	0x060728C4620F030101
NTRUEncrypt 251-1	ntruencrypt-sec-suite-1	id-802-15-3-security-suites 3	0x060728C4620F030102

The following table specifies the OID for each sub-suite defined for the security suites.

Table 25—OIDs for sub-suites

Sub-suite Name	OID Name	OID Number	DER Encoding
ECIES Raw 1	ecies-raw-1	ecies-sec-suite-1 1	0x060828C4620F03010101
ECIES X509 1	ecies-x509-1	ecies-sec-suite-1 2	0x060828C4620F03010102
ECIES Implicit 1	ecies-implicit-1	ecies-sec-suite-1 3	0x060828C4620F03010103
NTRUEncrypt Raw 1	ntruencrypt-raw-1	ntruencrypt-sec-suite-1 1	0x060828C4620F03010201

The value of the OID field for any command is the DER encoding of the OID.

2.2.2 Mode 2 mandatory to implement

[[The cipher suite name in this paragraph reflects the choice of ECIES Raw 1 as the mandatory to implement for mode 2.]]

In order to ensure that all compliant devices are able to interoperate when using security, all devices shall implement the ECIES 256-prime-1 security suite with the ECIES Raw 1 sub-suite. The security suite and mode are defined in clause 2.3. All other defined security suites and their respective sub-suite definitions may be implemented by a compliant device.

2.2.3 Security services

All of the currently defined sub-suites for the security suites provide the full range of security functionality defined in sub-clause 9.2{xref}. The following table summarizes the security services provided by these sub-suites.

Table 26—Supported security services

Security Services	Provided
Mutual Authentication	✓
Access Control List	✓
Verification of Public-Key	✓
Key Establishment	✓
Key Transport	✓
Beacon Integrity Protection	✓
Freshness Protection	✓
Command Integrity Protection	✓
ACK Integrity Protection	✓
Data Integrity Protection	✓
Data Encryption	✓

2.2.4 Symmetric cryptography building blocks

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

2.2.4.1 XOR and XOR encryption

The XOR operation used in this security suite is the bitwise exclusive-or of two bit strings of equal length. When one of the bit strings is a shared secret, this operation is defined as XOR encryption.

2.2.4.2 Bit ordering

In this clause 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.

2.2.4.3 Bitwise truncation

The bitwise truncation to *n*-bits operation used in this security suite shall be performed as taking the *n* left-most bits of the bit string as described in the FIPS publication #HMAC [{xref}HMAC].

2.2.4.4 SHA-256 cryptographic hash

The SHA-256 cryptographic hash algorithm used in this security suite shall be performed as specified in the FIPS 180-2 draft standard [{xref}FIP180].

2.2.4.5 HMAC keyed hashing for message authentication

The keyed hash message authentication code (HMAC) used in this security suite shall be performed as specified in the FIPS publication #HMAC [{xref}HMAC]. This message authentication code algorithm is parameterized by a 128-bit key, the SHA-256 hash function as specified in clause 2.2.4.4 and is truncated to 128 bits as specified in clause 2.2.4.3.

2.2.4.6 CBC encryption mode

The cipher-block chaining (CBC) encryption mode for block ciphers used in this security suite shall be preceded by a random IV and performed as specified in NIST Special Publication 800-38A [{xref}MODES].

2.2.4.7 AES encryption and decryption

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, 128-bit block size and the CBC encryption mode as specified in clause 2.2.4.6.

2.2.5 Symmetric cryptography implementation

All of the sub-suites defined for ECIES 256-prime-1 perform all symmetric operations in the same manner.

2.2.5.1 Symmetric cryptography data formats

The following table specifies the length and meaning of the symmetric cryptography related security suite specific data elements from clause 7{xref}. The operations performed to obtain the variable data values are specified in a separate sub-clause.

Table 27—Symmetric cryptography frame object formats

Notation	Length	Value	Description
AuthResponseType	2	See 7.4.1.2{xref}	The auth response type specifies the result of an HMAC computation as specified in sub-clause 2.2.4.5. The value is the entry for HMAC-SHA-256 in 7.4.1.2{xref}.

Table 27—Symmetric cryptography frame object formats

AuthResponseLength	2	16	The length of an HMAC computation as specified in sub-clause 2.2.4.5.
AuthResponse	16	Variable	The result of an HMAC computation as specified in sub-clause 2.2.4.5.
ChallengeResponseType	2	See 7.4.1.4{xref}	The challenge response type specifies the result of an HMAC computation as specified in sub-clause 2.2.4.5.The value is the entry for HMAC-SHA-256 in 7.4.1.4{xref}.
ChallengeResponseLength	2	16	The length of an HMAC computation as specified in sub-clause 2.2.4.5.
ChallengeResponse	16	Variable	The result of an HMAC computation as specified in sub-clause 2.2.4.5.
KeyPurpose	1	See 7.4.1.5{xref}	The type of key requested in key request protocols. Only seeds are transmitted in this security suite. The value is the entry for Seed in 2.2.4.7.
EncryptedKeyType	2	See 7.4.1.6{xref}	The encrypted key type specifies the result of an AES encryption as specified in sub-clause 2.2.4.7. The value is the entry for AES-128 in 7.4.1.6{xref}.
EncryptedKeyLength	2	32	The length of an encrypted 128-bit seed encrypted using AES encryption as specified in sub-clause 2.2.4.7.
EncryptedKey	32	Variable	The result of the encryption of the 128-bit seed using AES encryption as specified in sub-clause 2.2.4.7.

2.2.5.2 Symmetric cryptographic operations

The following table specifies the symmetric cryptography related operations on all secure frames defined in clause 7{xref}.

Table 28—Symmetric cryptographic operations

Operation	Specification
Integrity Key Derivation	Integrity keys are generated from a seed by first calculating the SHA-256 hash as specified in clause 2.2.4.4 on the seed concatenated with the byte 0x00 and then setting the key to be the truncation of the result to 128 bits as specified in clause 2.2.4.3.
Encryption Key Derivation	Encryption keys are generated from a seed by first calculating the SHA-256 hash as specified in clause 2.2.4.4 on the seed concatenated with the byte 0x01 and then setting the key to be the truncation of the result to 128 bits as specified in clause 2.2.4.3.
Challenge response generation	The challenge response is generated by computing the HMAC message authentication code as specified in clause 2.2.4.5 on the entire authentication protocol up to that point using the management integrity key.

Table 28—Symmetric cryptographic operations

Authentication response generation	The authentication response is generated by computing the HMAC message authentication code as specified in clause 2.2.4.5 on the entire authentication protocol up to that point using the management integrity key.
Beacon message authentication code generation	The message authentication code included in the beacon is computed as the HMAC message authentication code as specified in clause 2.2.4.5 on the entire beacon up to the integrity code information element using the piconet integrity key.
Command message authentication code generation	The message authentication code included in command frames is computed as the HMAC message authentication code as specified in clause 2.2.4.5 on the entire command up to the message authentication code using the selected integrity key.
ACK message authentication code generation	The message authentication code included in ACK frames is computed as the HMAC message authentication code as specified in clause 2.2.4.5 on the entire ACK up to the message authentication code using the selected integrity key.
Data message authentication code generation	The message authentication code included in data frames is computed as the HMAC message authentication code as specified in clause 2.2.4.5 on the entire data frame up to the message authentication code after encryption has been performed using the payload integrity key.
Seed encryption operation	The seed for key transport is encrypted using AES as specified in clause 2.2.4.7 using the management key encryption key.
Data encryption generation	Data in a data frame is encrypted using AES as specified in clause 2.2.4.7 using the payload encryption key.

2.3 ECIES 256-prime-1 security suite

[[All of the text in sub-clause 2.3 describing ECIES 256-prime-1 is the NTRU submission text for the ECC security suite.]]

The following sub-clauses define the security operations that are performed for the security suite ECIES 256-prime-1. The symmetric operations performed in this security suite are those specified in sub-clause 2.2.5. The public key and authentication operations are specified in the following sub-clause 2.3.1.

2.3.1 Public-key and authentication building blocks

The following cryptographic primitives and data elements are defined for use in all sub-suites of ECIES 256-prime-1.

2.3.1.1 Elliptic curve point representation

All transmitted elliptic curve points shall be transmitted in uncompressed form according to the definition of uncompressed in ANSI X9.63 [{xref}X963]. Additionally, when elliptic curve points are used as inputs into key derivation functions or any other function that utilizes a particular point representation, the points shall be represented in uncompressed form.

2.3.1.2 Elliptic curve ansip256r1

All elliptic curve points and operations used in this security suite shall be on the curve ansip256r1 as specified in ANSI X9.63 [{xref}X963].

2.3.1.3 Elliptic curve key pair

An elliptic curve key pair consists of the private key, which is an integer that is smaller than the order of the base point and the public key, which is a point on the elliptic curve as specified in ANSI X9.63 [{xref}X963]. All elliptic curve public keys shall be on the elliptic curve specified in sub-clause 2.3.1.2. Key validation is optional during all public-key operations.

2.3.1.4 ECIES encryption and decryption

The elliptic curve integrated encryption scheme (ECIES) encryption algorithm used in this security suite shall be performed as specified in ANSI X9.63 [{xref}X963] with the key validation procedures and checking for the identity left as optional. This encryption algorithm is parameterized by the XOR encryption algorithm as specified in clause 2.2.4.1, the standard Diffie-Hellman primitive as specified in ANSI X9.63 [{xref}X963], SHA-256 as specified in sub-clause 2.2.4.4, HMAC as specified in sub-clause 2.2.4.5 and point representation as specified in clause 2.3.1.1. The string SharedData1 that is used as input into the ECIES primitive shall be set to be the ephemeral public key generated by the encryptor, encoded as an octet string according to the conversion primitive in section 4.3.6 of ANSI X9.63 [{xref}X963]. The string SharedData2 shall be the empty string.

2.3.1.5 ECC X.509 certificate

The X.509 digital certificate format and verification used in this security suite shall be as specified by the PKIX RFC 2459 [{xref}PKIX]. These certificates shall contain an ECC public key as specified in clause 2.3.1.3. These certificates shall be signed using the ECDSA algorithm as specified in clause 2.3.1.6. The ASN.1 encoding for the public key and signature shall be as specified in SEC 1 [{xref}SEC1]. The subject field of the X.509 certificate shall be NULL and the subjectAltName shall consist of the PrintableString encoding of the hexadecimal representation of the 48-bit IEEE MAC address of the device.

2.3.1.6 ECDSA digital signatures and verification

The ECDSA digital signature algorithm used in this security suite shall be performed as specified in ANSI X9.62 [{xref}X962]. The ECDSA signature algorithm operations shall be performed on the elliptic curve specified in clause 2.3.1.2. Public-key validation and checking the point at infinity shall be left as optional operations.

2.3.1.7 ECC Implicit certificate

The ECC implicit certificate format and verification used in this security suite shall be as specified in clause 4. This certificate shall be associated with an ECC public key as specified in clause 2.3.1.3. This is parameterized by the following choices.

1) Each entity shall be a DEV;
2) Each entity's identifier shall be its 48-bit IEEE MAC address {xref}; the parameter entlen shall have the integer value 48;
3) Each entity shall use the cryptographic hash function as specified in Clause 2.2.4.4;
4) The format of the implicit certificate ImplCert shall be specified as follows: ImplCert=(PublicReconstrKey || Subject || Issuer), where
    a ImplCert shall be the representation of the string ICU as specified in the implicit certificate generation protocol;

b PublicReconstrKey shall be the representation of the public-key reconstruction data BEU as specified in the implicit certificate generation protocol, which is an elliptic curve point as specified in clause 2.3.1.3;

c Subject shall be the identifier of the entity U that is bound to the public-key reconstruction data BEU during the execution of the implicit certificate generation protocol;

d Issuer shall be the identifier of the entity CA that creates the implicit certificate during the execution of the implicit certificate generation protocol (the so-called Certificate Authority).

5) The format of the string IU as specified in Step 6 of the actions of the CA in the implicit certificate generation protocol is specified as follows: IU=(Subject || Issuer), where Subject and Issuer are as specified above.

2.3.2 ECIES Raw 1 sub-suite

ECIES Raw 1 is a Mode 2 sub-suite of the ECIES 256-prime-1 security suite. The cryptographic building blocks for ECIES Raw 1 are the same as for the ECIES 256-prime-1 security suite. The OID for this sub-suite is specified in Table 25. The following sub-clauses specify the public-key and authentication related objects for this sub-suite.

2.3.2.1 Public-key and authentication data formats

The following table specifies the length and meaning of the public-key cryptography and authentication related security suite specific data elements from clause 7{xref}. The operations performed to obtain the variable data values are specified in a separate sub-clause.

Table 29—Public-key frame object formats

Notation	Length	Value	Description
PublicKeyObjectType	2	See 7.4.1.2{xref}	An ECC public key as specified in clause 2.3.1.3. The value is the entry for ECC Raw 256 in table 7.4.1.2{xref}.
PublicKeyObjectLength	2	65	The length of an ECC public key as specified in clause 2.3.1.3.
PublicKeyObject	65	Variable	The particular instance of the ECC public key.
OIDLength	1	10	The length of the DER encoding of the OID ecies-raw-1 as specified in Table 25.
OID	10	OID Value	The DER encoding of the OID ecies-raw-1 as specified in Table 25.
ChallengeType	2	See table 7.4.1.3{xref}	The challenge type is an ECIES encryption of a 16-octet challenge as specified in clause 2.3.1.4. The value is the entry for ECIES 256 encryption in 7.4.1.3{xref}.
ChallengeLength	2	65 + 16 + 16 = 97	The length of an ECIES encryption of a 16-octet challenge as specified in clause 2.3.1.4.
Challenge	97	Variable	The result of the ECIES encryption of the 16-octet challenge as specified in clause 2.3.1.4.

2.3.2.2 Public-key and authentication cryptographic operations

The following table specifies the public key cryptography and authentication related operations for the authentication protocol frames defined in clause 7{xref}.:

Table 30—Authentication related operations

Use	Operation
Verification of Public-Key	The ID and public key received during the authentication protocol is verified by generating the SHA-256 hash of the device address concatenated with the public key of the device and comparing it to the hash of the ID and public key stored in the MAC PIB. If the hash is not in the PIB, the public key is passed to the DME to establish trust by other means.
Challenge generation	The challenges generated during the authentication protocol are computed by performing an ECIES encryption as specified in sub-clause 2.3.1.4 on a fresh, randomly generated 16-byte challenge using the other device's public key.
Challenge decryption	The challenge decryption operation is performed using ECIES decryption as specified in sub-clause 2.3.1.4 on the received challenge. The decryptor shall verify that the length of the output is 16-bytes and, if not, reject the challenge.
Seed generation (for authentication protocol)	The 32-byte seed for the authentication protocol consists of the decrypted challenge from the security manager, concatenated with the decrypted challenge of the DEV.

2.3.3 ECIES X509 1 sub-suite

ECIES X509 1 is a mode 3 sub-suite of the ECIES 256-prime-1 security suite. The cryptographic building blocks for ECIES X509 1 sub-suite are selected from the public-key cryptographic building blocks defined for the ECIES 256-prime 1 security suite. The OID for this sub-suite is specified in Table 25. The following sub-clauses specify the public-key and authentication related objects for this sub-suite.

2.3.3.1 Public-key and authentication data formats

The following table specifies the length and meaning of the public-key cryptography and authentication related security suite specific data elements from clause 7{xref}. The operations performed to obtain the variable data values are specified in a separate sub-clause.

Table 31—Public-key frame object formats

Notation	Length	Value	Description
PublicKeyObjectType	2	See 7.4.1.2{xref}	An ECC X.509 certificate as specified in clause 2.3.1.5. The value is the entry for ECC X.509 256 in table 7.4.1.2{xref}.
PublicKeyObjectLength	2	Variable	The length of the particular instance of the X.509 certificate.

Table 31—Public-key frame object formats

PublicKeyObject	Variable	Variable	The particular instance of the X.509 certificate.
OIDLength	1	10	The length of the DER encoding of the OID ecies-x509-1 as specified in Table 25.
OID	10	OID Value	The DER encoding of the OID ecies-x509-1 as specified in Table 25.
ChallengeType	2	See table 7.4.1.3{xref}	The challenge type is an ECIES encryption of a 16-octet challenge as specified in clause 2.3.1.4. The value is the entry for ECIES 256 encrypted seed in 7.4.1.3{xref}.
ChallengeLength	2	65 + 16 + 16 = 97	The length of an ECIES encryption of a 16-octet challenge as specified in clause 2.3.1.4.
Challenge	97	Variable	The result of the ECIES encryption of the 16-octet challenge as specified in clause 2.3.1.4.

2.3.3.2 Public key and authentication cryptographic operations

The following table specifies the public key cryptography and authentication related operations for the authentication protocol frames defined in clause 7{xref}.:

Table 32—Authentication related operations

Use	Operation
Verification of Public-Key	The X.509 certificate received during the authentication protocol is verified by retrieving the appropriate CA key and verifying the ECDSA signature as specified in clause 2.3.1.6 on the certificate. The device shall verify that the MAC address in the subjectAltName of the certificate matches the MAC address that the certificate was received from. The device shall extract the public key for use in the authentication protocol. There are several other checks that should be performed by the device if possible to ensure the security properties of the certificate including a CRL check, validity period verification and the key use field check. In addition, the device shall check that the device in the certificate is included in the ACL.
Challenge generation	The challenges generated during the authentication protocol are computed by performing an ECIES encryption as specified in sub-clause 2.3.1.4 on a fresh, randomly generated 16-byte challenge using the other device’s public key.
Challenge decryption	The challenge decryption operation is performed using ECIES decryption as specified in sub-clause 2.3.1.4 on the received challenge. In addition, the decryptor shall verify that the length of the output is 16-bytes and, if not, reject the challenge.
Seed generation (for authentication protocol)	The 32-byte seed for the authentication protocol consists of the decrypted challenge from the security manager, concatenated with the decrypted challenge of the DEV.

2.3.4 ECIES Implicit 1 sub-suite

ECIES Implicit 1 is a mode 3 sub-suite of the ECIES 256-prime-1 security suite. The cryptographic building blocks for ECIES Implicit 1 sub-suite are selected from the public-key cryptographic building blocks defined for the ECIES 256-prime 1 security suite. The OID for this sub-suite is specified in table 25. The following sub-clauses specify the public-key and authentication related objects for this sub-suite.

2.3.4.1 Public-key and authentication data formats

The following table specifies the length and meaning of the public-key cryptography and authentication related security suite specific data elements from clause 7{xref}. The operations performed to obtain the variable data values are specified in a separate sub-clause.

Table 33—Public-key frame object formats

Notation	Length	Value	Description
PublicKeyObjectType	2	See 7.4.1.2{xref}	An ECC implicit certificate as specified in clause 2.3.1.7. The value is the entry for ECC Implicit 256 in table 7.4.1.2{xref}.
PublicKeyObjectLength	2	65+6+6= 77	The length of the implicit certificate.
PublicKeyObject	77	Variable	The particular instance of the implicit certificate.
OIDLength	1	10	The length of the DER encoding of the OID ecies-implicit-1 as specified in table 2{xref}.
OID	10	OID Value	The DER encoding of the OID ecies-implicit-1 as specified in table 2{xref}.
ChallengeType	2	See table 7.4.1.3{xref}	The challenge type is an ECIES encryption of a 16-octet challenge as specified in clause 2.3.1.4. The value is the entry for ECIES 256 encrypted seed in 7.4.1.3{xref}.
ChallengeLength	2	65 + 16 + 16 = 97	The length of an ECIES encryption of a 16-octet challenge as specified in clause 2.3.1.4.
Challenge	97	Variable	The result of the ECIES encryption of the 16-octet challenge as specified in clause 2.3.1.4.

2.3.4.2 Public key and authentication cryptographic operations

The following table specifies the public key cryptography and authentication related operations for the authentication protocol frames defined in clause 7{xref}.:

Table 34—Authentication related operations

Use	Operation
Verification of Public-Key	The public key in the received implicit certificate is implicitly verified by retrieving the appropriate CA key, generating the public key by performing computations using the implicit certificate, identifying information and CA key, and verifying that the authentication protocol succeeds as specified in clause 2.3.1.7. The device shall also verify that the expected MAC address of the device matches the MAC address in the certificate. There are several other checks that should be performed by the device to ensure the security properties of the certificate including a CRL check and validity period verification. In addition, the device shall check that the device in the certificate is included in the ACL.
Challenge generation	The challenges generated during the authentication protocol are computed by performing an ECIES encryption as specified in sub-clause 11.3.1.4{xref} on a fresh, randomly generated 16-byte challenge using the other device’s public key.
Challenge decryption	The challenge decryption operation is performed using ECIES decryption as specified in sub-clause 11.3.1.4{xref}on the received challenge.
Seed generation (for authentication protocol)	The 32-byte seed for the authentication protocol consists of the decrypted challenge from the security manager, concatenated with the decrypted challenge of the DEV.

2.4 NTRUEncrypt 251-1

The following sub-clauses define the security operations that are performed for the security suite NTRUEncrypt 251-1. The symmetric operations performed in this security suite are those specified in sub-clause 2.2.5. The public key and authentication operations are specified in the following sub-clause 2.4.1

2.4.1 Public-key and authentication building blocks

The following cryptographic primitives and data elements are defined for use in all sub-suites of NTRUEncrypt 251-1.

2.4.1.1 NTRUEncrypt Parameter Set ees251ep1

All NTRUEncrypt objects and cryptographic operations used in this security suite shall use the parameter set ees251ep1 as specified in EESS #1 [{xref}EESS#1]. All transmitted NTRUEncrypt polynomials shall be sent in uncompressed form as specified in EESS #1 [{xref}EESS#1].

2.4.1.2 NTRUEncrypt key pair

An NTRUEncrypt key pair consists of the private key, which is a small polynomial and the public key, which is a large polynomial as specified in EESS #1[{xref}EESS#1]. All NTRUEncrypt public keys shall use the parameter set specified in sub-clause 2.4.1.1.

2.4.1.3 NTRUEncrypt encryption and decryption

The NTRUEncrypt encryption algorithm used in this security suite shall be performed as specified in EESS #1[{xref}EESS#1]. All encryption and decryption operations shall use the parameter set specified in 2.4.1.1.

2.4.2 NTRUEncrypt Raw 1 sub-suite

NTRUEncrypt Raw 1 is a mode 2 sub-suite of the NTRUEncrypt 251-1 security suite. The cryptographic building blocks for the NTRUEncrypt Raw 1 sub-suite are selected from the public-key cryptographic building blocks defined for the NTRUEncrypt 251-1 security suite. The OID for this sub-suite is specified in Table 25. The following sub-clauses specify the public-key and authentication related objects for this sub-suite.

2.4.2.1 Public-key and authentication data formats

The following table specifies the length and meaning of the public-key cryptography and authentication related security suite specific data elements from clause 7{xref}. The operations performed to obtain the variable data values are specified in a separate sub-clause.

Table 35—Public-key frame object formats

Notation	Length	Value	Description
PublicKeyObjectType	2	See 7.4.1.1{xref}	An NTRUEncrypt public key as specified in clause 2.4.1.2. The value is the entry for NTRUEncrypt Raw 251 in table 7.4.1.1{xref}.
PublicKeyObjectLength	2	251	The length of the particular instance of the NTRUEncrypt public key.
PublicKeyObject	251	Variable	The particular instance of the NTRUEncrypt public key.
OIDLength	1	10	The length of the DER encoding of the OID ntruencrypt-raw-1 as specified in Table 25.
OID	10	OID Value	The DER encoding of the OID ecies-raw-1 as specified in Table 25.
ChallengeType	2	See table 7.4.1.3{xref}	The challenge type is an NTRUEncrypt encryption of a 21-octet challenge as specified in clause 2.4.1.3. The value is the entry for NTRUEncrypt 251 encrypted seed in 7.4.1.3{xref}.
ChallengeLength	2	251	The length of an NTRUEncrypt encryption of a 21-octet challenge as specified in clause 2.4.1.3.
Challenge	251	Variable	The result of the NTRUEncrypt encryption of the 21-octet challenge as specified in clause 2.4.1.3.

2.4.2.2 Public key and authentication cryptographic operations

The following table specifies the public key cryptography and authentication related operations for the authentication protocol frames defined in clause 7{xref}.:

Table 36—Authentication related operations

Use	Operation
Verification of Public-Key	The ID and public-key received during the authentication protocol is verified by generating the SHA-256 hash of the device address concatenated with the public key of the device and comparing it to the hash of the ID and public key stored in the MAC PIB. If the hash is not in the PIB, the public key is passed to the DME to establish trust by other means.
Challenge generation	The challenges generated during the authentication protocol are computed by performing an NTRUEncrypt encryption as specified in sub-clause 2.4.1.3 on a fresh, randomly generated 21-byte challenge using the other device's public key.
Challenge decryption	The challenge decryption operation is performed using NTRUEncrypt decryption as specified in sub-clause 2.4.1.3 on the received challenge.
Seed generation (for authentication protocol)	The 42-byte seed for the authentication protocol consists of the decrypted challenge from the security manager, concatenated with the decrypted challenge of the DEV.

3. Security Considerations

3.1 Claimed Security Services

Each of the protocols defined in clause 9{xref} is designed to offer specific security services, detailed in the following sub-clauses.

3.1.1 Authentication and Key Establishment Protocol

The following table specifies the security services provided by the authentication and key establishment protocol specified in clause 10.4.1{xref} along with a description of the method employed to provide the security service:

Security Service	Method Provided
Verification by the security manager that the authenticating device possesses its private key	The ownership of the private key is demonstrated by the device through the proper generation of the integrity key and the computation and transmission of the integrity code on the challenge response command.

Verification by the authenticating device that the security manager possesses its private key	The ownership of the private key is demonstrated by the security manager through the proper generation of the integrity key and the computation and transmission of the integrity code on the authentication response command.
Verification by the security manager of the linkage of the following items to the current run of the protocol: — current session ID — current security suite — public key of each participating entity — identity of each participating entity — challenge by each participating entity	The linkage of the items to the current run of the protocol is demonstrated by the device by the correctly formed integrity code computed on the items.
Verification by the device of the linkage of the following items to the current run of the protocol: — current session ID — current security suite — public key of each participating entity — identity of each participating entity — challenge by each participating entity — device's proof of ownership of private key — current group payload protection seed	The linkage of the items to the current run of the protocol is demonstrated by the security manager with the correctly formed integrity code that is computed on the items.
The device obtains two-party management keys for transfer of protected commands between the security manager and the authenticating device	The successful operations performed on the challenge from the security manager and the combination of that challenge with the challenge from the device establishes the two-party management keys.
The security manager obtains two-party management keys for transfer of protected commands between the security manager and the authenticating device	The successful operations performed on the challenge from the device and the combination of that challenge with the challenge from the security manager establishes the two-party management keys.
Both devices initialize freshness information for messages sent using new keys by security manager and by device	At the conclusion of the protocol, each device stores the initialized sequence numbers for the security manager and device.

3.1.2 Beacon Protection Protocol

The following table specifies the security services provided by the beacon protection protocol specified in clause 10.4.2{xref} along with a description of the method employed to provide the security service:

Security Service	Method Provided
Communication of current time token to the devices in the piconet	The PNC increments the time token for each superframe and protects it using the current group key. The integrity protection on the beacon and the storage of the previous time token allows each device to determine that the time token is fresh.
Indication of the identity of the PNC to the devices in the piconet	If PNC handover has not occurred, the device address of the current PNC appears in the beacon. If PNC handover has occurred, the device address of the new PNC appears in the beacon. The integrity protection on the beacon and the freshness from the time token allow each device to determine the identity of the current PNC.

3.1.3 Distribute Key Protocol

The following table specifies the security services provided by the distribute key protocol specified in clause 10.4.3{xref} along with a description of the method employed to provide the security service:

Security Service	Method Provided
Privacy protection on distributed key	The encryption of the key with the shared key encryption key ensures that the key remains private
Integrity protection on distributed key	The receiving device verifies that the integrity code verifies properly and that the freshness checks succeed.
Verification by the security manager that the device received the key	The security manager verifies that the integrity code verifies properly and that the freshness checks succeed.

3.1.4 Key Request Protocol

The following table specifies the security services provided by the key request protocol specified in clause 10.4.4{xref} along with a description of the method employed to provide the security service:

Security Service	Method Provided
Privacy protection on requested key	The encryption of the key with the shared key encryption key ensures that the key remains private
Integrity protection on the requested key	The receiving device verifies that the integrity code verifies properly and that the freshness checks succeed.

3.1.5 Data Transport Protocol

The following table specifies the security services provided by the data transport protocol specified in clause 10.4.5{xref} along with a description of the method employed to provide the security service:

Security Service	Method Provided
Privacy protection on data	The encryption of the data with the shared encryption key ensures that the key remains private
Integrity protection on data	The receiving device verifies that the integrity code verifies properly and that the freshness checks succeed.

3.2 Public Key and Identity Binding Method

The DME bears the responsibility for establishing a binding between a public key and a device's identity.

As part of the authentication process, a device provides its public key and device ID. The security manager checks if the public key and ID are represented in its access control list (ACL). If so, the protocol continues. If not, the security manager passes the public key and ID pair to its DME and returns a failure message to the device, indicating the cause of failure.

The DME may use any method to decide if the public key and ID pair is to be trusted. This method is out of scope, but may include:

— A digital certificate.
— An analog certificate. A device manufacturer may print the device ID and a hash of the public key on the bottom of the device for the user to verify.
— Low-power transmission. Two devices may be brought into close physical proximity so they can "whisper" public keys over their radios.
— Range. The user confirms the distance between the two devices.
— Open enrollment. While located in a secure environment like a free-standing house, devices may simply trust public keys they receive over the air.
— Pre-loading. A device manufacturer selling matched devices, like the components of a home-entertainment system, may pre-load the ACLs with the IDs and public keys the system needs.
— User action. The user could push a button on both devices simultaneously.

This range of options is allowed to enable cost-effective and user-friendly applications.

3.3 Protocol Security Analysis

3.3.1 Comparison with TLS

The following describes some of the differences between the TLS protocol and the protocols defined in clause 10. We first describe the differences between the TLS protocol and our design, and then the differences between the proposed security suites and ciphersuites supported in TLS.

- TLS provides integrity with HMAC and PRF using SHA-1 and MD5. These protocols instead allow other HMAC algorithms, hash algorithms or block cipher message authentication codes using symmetric algorithms.

  The security suite specification in this document specifies the use of HMAC with SHA-256.
- TLS requires certificates. This protocol does not.

In TLS, the typical use case is that a user's browser communicates to an e-commerce site and requires assurance that the site belongs to a legitimate business. Certificates are highly suitable for this situation. In 802.15.3, a DEV is obtaining a connection to an ad-hoc network. In this situation, certificates may or may not be used depending on the needs of the environment and the trust model for the public keys.

- TLS typically authenticates the server with an encrypted challenge-response, while client authentication is done by means of a signed challenge from the server. This protocol instead allows for the use of two-way encrypted challenge-response and other two-party mutual authentication mechanisms.

  This change is desirable for several reasons:

  - If both parties use similar challenge mechanisms, both parties can contribute to the freshness of the key that is established, preventing reuse of old shared secrets.
  - In cost-sensitive devices such as an 802.15.3 DEV, gates and cost can be saved if devices do not have to implement signing as well as key establishment techniques.

- TLS has cipher suite agreement in line. This protocol does not.

  For a group networking protocol such as this one, all devices must use the same cipher suite. A device that wants to join the network should not be able to compel all other devices to change cipher suite. Therefore, there is no requirement for cipher suite negotiation when a device is joining the group.

- In TLS, the client (device) commits to a public random value in step 1. In this protocol, there are no public random values.

  TLS requires the use of public random values to ensure that its Diffie-Hellman-based ciphersuites produce a different seed every time. The techniques specified in the algorithm suites ensure that a unique seed is produced every time.

- In TLS, the server (security manager) commits to a public random value in step 2. In this protocol, there are no public random values.

  See the previous comment.

- TLS offers optional compression. This protocol does not.

  TLS is an application-layer protocol. The data it produces is passed down through the protocol stack. However, because it is encrypted, it appears random, which means that it cannot be compressed by any process running further down the stack than TLS. TLS therefore has to offer compression. In contrast, 802.15.3 is a networking protocol. If data is to be compressed, it will be done at a higher level than the MAC layer.

- In TLS, the sequence number used for payload protection is not explicitly sent (it is stored internally and incremented). This protocol transmits the sequence number, cryptographically protected, in the messages.

  TLS uses a reliable communications channel between two applications, and can assume that packets arrive in order and that both parties are still involved in the communication. 802.15.3 is a standard for ad hoc networking and can make no such assumptions. In order to ensure all devices are able to decrypt and check integrity on the data, it is necessary to explicitly include the sequence number in the messages.

- TLS does not require additional key transport messages after the handshake (authentication) is complete.

In TLS, once a shared key is established between the two entities, no further exchange of cryptographic material is necessary. In this protocol, once the shared key has been established, the joining device must have the group key securely transmitted to it before it can participate in the group communications. For this reason, the current protocols require the additional key transport message.

- Key derivation from the seed is different in TLS from this protocol.

  TLS requires the use of two different hash functions, MD5 and SHA-1, to derive the key from the seed. This protocol relies on the security of a single hash function.

4. Implicit Certificate Specification

[[All of the text in clause 4 describing implicit certification is part of the NTRU submission text for the ECC security suite.]]

4.1 Implicit Certificate Scheme

This section specifies the ECQV implicit certificate scheme based on ECC supported in this standard.

Implicit certificate schemes are designed to be used by three entities - a Certification Authority CA, a certificate requester U, and a certificate processor V, where U wants to obtain an implicit certificate from CA in order to convey U's public key to V.

Here implicit certificate schemes are described in terms of a certificate generation protocol, a certificate processing operation, and associated setup and CA key deployment procedures. CA, U, and V shall use the schemes as follows, when they wish to communicate.

CA, U, and V should use the setup procedure to establish which options to use the scheme with. CA should use the key deployment procedure to select a key pair and U and V should obtain CA's public key - CA will use the key pair during the certificate generation protocol, U will use the public key during the certificate generation protocol, and V will use the public key during the certificate processing operation. When U wants to obtain an implicit certificate, U and CA should perform the certificate generation protocol to obtain a key pair known to U and an implicit certificate IC. Finally, when V wants to obtain U's public key, U should convey IC to V and V should apply the certificate processing to IC under CA's public key to obtain U's public key. V concludes that the public key is genuine, provided U provides evidence that it possesses the corresponding private key.

The specification of ECQV in this document relies heavily on the mathematical foundations and cryptographic components specified in Sections 2 and 3 of [SEC1].

The setup procedure for ECQV is specified in Section 8.1, the key deployment procedure is specified in Section 8.2, the certificate generation protocol is specified in Section 8.3, and the certificate processing operation is specified in Section 8.4.

4.1.1 Scheme Setup

CA, U, and V shall perform the following setup procedure to prepare for the use of ECQV.

1. An infrastructure shall have been established for the operation of the scheme - including a certificate format, certificate processing rules, and unique identifiers. An example of such an infrastructure is described by PKIX [PKIX-X509].

2. Each entity has an authentic copy of the system's elliptic curve domain parameters D=(p,a,b,G,n,h) or D=(m,f(x),a,b,G,n,h). These parameters shall have been generated using the parameter generation primitive in Sections 3.1.1.1 or the primitive specified in Section 3.1.2.1, both of [SEC1]. Furthermore, the parameters shall have been validated using the parameter validation primitives in Sections 3.1.1.2 or Section 3.1.2.2 of [SEC1].

3. The CA shall have decided which cryptographic hash function to use when generating implicit certificates. Let Hash denote the hash function chosen, and let hashlen denote the length in bits of the output value of this hash function. Each entity shall have an authentic copy of this hash function.

4. Each entity shall be bound to a unique identifier (e.g. distinguished names). All identifiers shall be bit strings of the same length entlen bits. Entity U's identifier will be denoted by the bit string U. Entity V's identifier will be denoted by the bit string V. Entity CA's identifier will be denoted by the bit string CA.

5. The CA shall be bound to a static public key pair associated with the system's elliptic curve domain parameters D. The binding process shall include the validation of the static public key as specified in Section 3.2.2 of [SEC1]. The key binding shall include the unique identifier CA for the entity involved.

6. Each entity shall have decided how to represent elliptic curve points as octet strings (i.e., compressed form, uncompressed form, or hybrid form).

4.1.2 Key Deployment

CA, U, and V shall use the following key deployment procedure to prepare for the use of ECQV.

1. CA shall establish a static elliptic curve key pair (WCA, wCA) associated with D to use with the certificate generation and processing protocols. The key pair shall be generated using the primitive specified in Section 3.2.1 of SEC 1.

2. U and V shall obtain in an authentic manner the elliptic curve public key WCA selected by CA.

4.1.3 Implicit Certificate Generation Protocol

This section specifies the protocol for generating implicit certificates (self-certified public keys).

Figure 12 illustrates the messaging involved in the use of the certificate generation protocol.

CA and U shall generate an implicit certificate for U using the keys and parameters established during the setup procedure and the key deployment procedure as follows:

Input: None.

Output for CA: An octet string ICU, which is U's implicit certificate, and an elliptic curve public key WU corresponding to the elliptic curve domain parameters D, which is U's public key.

Output for U: An octet string ICU, which is U's implicit certificate, and an elliptic curve key pair (WU, wU) corresponding to the elliptic curve domain parameters D, which is U's public key pair.

Actions: U proceeds as follows:

1. Select a random ephemeral elliptic curve key pair (QU, qU) associated with the elliptic curve domain parameters D. The key pair shall be generated using the primitive specified in Section 3.2.1 of SEC 1. (qU is referred to as U's private request value; QU as U's public request value.)

2. Convert QU to an octet string QEU as specified in Section 2.3.5 in SEC 1, and send it to CA.

3. Obtain the values ICU and se from CA. Compute the hash value H = Hash(ICU) using the established hash function. Derive an integer e from H following the conversion routine of Step 5 in Section 4.1.3 of [SEC1].

4. Verify the content of ICU according to the established infrastructure. This includes verifying the contents of the certificate, e.g., the subject's name and the validity period.

5. Derive BEU and IU from ICU according to the procedures of the established infrastructure.

6. Derive WCA from IU, according to the certificate format specified during the setup procedure.

7. Convert BEU to an elliptic curve point BU as specified in Section 2.3.4 of SEC 1 and convert se to an integer s as specified in Section 2.3.8 of SEC 1.

8. Compute the secret key wU := s + qU . e (mod n) and the public key WU := wU G.

9. Reconstruct the public key WU' = e BU + WCA.

10. If WU' = WU, accept the certificate and output the computed key pair (WU, wU) and the implicit certificate ICU; otherwise output reject.

Actions: CA proceeds as follows:

1. Verify the authenticity of the request received from U according to the procedures of the established infrastructure. The checks performed shall include, as a minimum, checking that U is indeed the origin of the request, and checking that U is authorized to obtain a certificate.

2. Receive QEU from U and convert it to an elliptic curve point QU as specified in Section 2.3.4 in SEC 1.

3. Select an ephemeral elliptic curve key pair (QCA, qCA) associated with the elliptic curve domain parameters D. The key pair shall be generated using the primitive specified in Section 3.2.1 of [SEC1].

4. Compute the elliptic curve point BU:= QU + QCA.

5. Convert the elliptic curve point BU to an octet string BEU as specified in Section 2.3.3 in SEC 1.

6. Construct the 'to-be-signed-certificate' data, which is an octet string IU. IU shall contain identification information according to the procedures of the established infrastructure and may also contain other information, such as the intended use of the public key, the serial number of the implicit certificate, and the validity period of the implicit certificate. The exact form of IU depends on the certificate format being used.

7. Construct according to the procedures of the established infrastructure U's implicit certificate, which is an octet string ICU. ICU should contain IU and BEU. The exact form of ICU depends on the certificate format being used.

8. Compute the hash value H = Hash(ICU) using the established hash function. Derive an integer e from H following the conversion routine of Step 5 in Section 4.1.3 of [SEC1].

9. Compute the integer s = qCA e+wCA (mod n). (s is referred to as the private-key reconstruction data.)

10. Convert s to an octet string se as specified in Section 2.3.7 of SEC 1, and send ICU and se to U.

11. Compute WU = e BU + WCA. Output the implicit certificate ICU and the elliptic curve public key WU.

4.1.4 Implicit Certificate Processing Operation

V shall process U's implicit certificate using the keys and parameters established during the setup procedure and the key deployment procedure as follows:

Input: U's purported implicit certificate ICU.

Output: U's purported public key WU.

Actions: V proceeds as follows:

1. Verify the content of ICU according to the established infrastructure. This includes verifying the contents of the certificate, such as the subject's name and the validity period.

2. Derive BEU and IU from ICU, according to the certificate format specified during the setup procedure.

3. Derive WCA from IU, according to the certificate format specified during the setup procedure.

4. Convert the octet string BEU to an elliptic curve point BU as specified in Section 2.3.4 of SEC 1.

5. Compute the hash value H = Hash(ICU) using the established hash function. Derive an integer e from H following the conversion routine of Step 5 in Section 4.1.3 of [SEC1].

6. Compute the public key WU = e BU + WCA.

After performing the certificate processing operation, V can conclude that WU is genuine, provided U evidences knowledge of the corresponding private key wU.