AN1233: Zigbee Security

This document introduces some basic security concepts, including network layer security, trust centers, and application support layer security features. It then discusses the types of standard security protocols available in EmberZNet PRO. Coding requirements for implementing security are reviewed in summary. Finally, information on implementing Zigbee Smart Energy security is provided.

KEY POINTS

- Presents the Zigbee stack architecture.
- Introduces basic security concepts such as network layer security, trust center networks, and APS layer security.
- Defines the different keys used for securing data in different ways.
- Describes Zigbee Smart Energy (ZSE) Security.

1. Introduction

Security is a major concern in the Zigbee architecture. Although Zigbee uses the basic security elements in IEEE 802.15.4 (for example, Advanced Encryption Standard (AES) encryption and Counter with CBC-MAC (CCM) security modes), it expands upon this with:

- 128-bit AES encryption algorithms
- Strong, U.S. National Institute of Standards and Technology (NIST)-approved security
- Defined key types (link, network)
- Defined key setup and maintenance
- Keys can be hardwired into an application
- CCM* (Unified/simpler mode of operation)
- Trust centers
- Security that can be customized for the application

As the following figure (from Zigbee document 05-3474-21: Zigbee Specification) illustrates, the security services provider block interactions with both the application and network layers.

Zigbee now supports a single defined security mode called **Standard Security**. Various policies exist within that mode to control how devices behave or interact on the network. Earlier versions of the Zigbee standard utilized modes known as **Residential Security** and **High Security**. These have been deprecated.

Note: IEEE 802.15.4 MAC-level security is **not** used by Zigbee and is therefore not supported by EmberZNet PRO and not described here. Zigbee implements message security at the network and application layers.

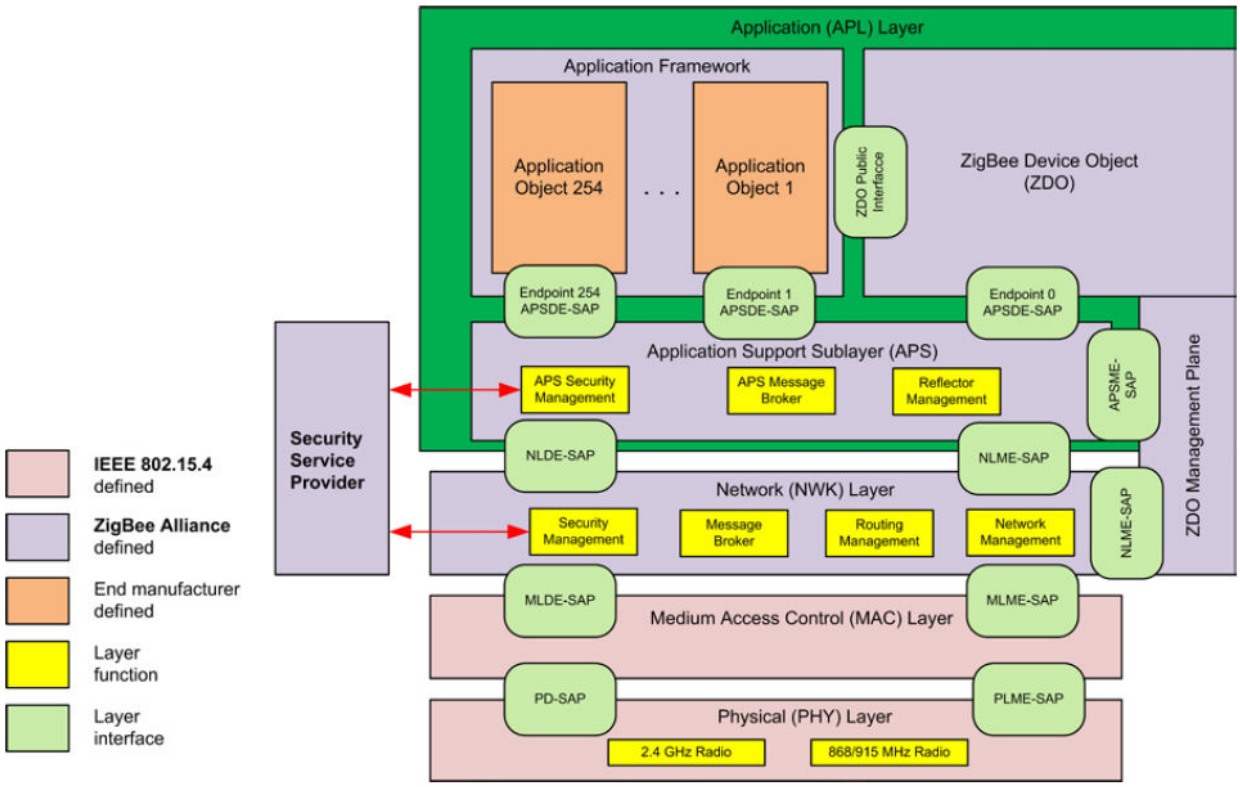


Figure 1.1. Zigbee Stack Architecture

This document first describes some basic security concepts, including network layer security, trust centers, and application support layer security features. It then discusses the types of standard security protocols available in EmberZNet PRO. Coding requirements for implementing security are reviewed in summary. Finally, information on implementing Zigbee Smart Energy security is provided. Details may be found in document *AN714: Smart Energy ECC-Enabled Device Setup*.

Those already familiar with Zigbee security concepts can jump to section 4. Zigbee Smart Energy (ZSE) Security.

2. Concepts

2.1 Network Layer Security

This section describes how Zigbee implements security at the network layer, which applies to standard security. Network security provides security independent of the applications that may be running on a Zigbee node. All Zigbee-certified devices must use network layer security. It provides the basic access control for controlling what nodes are allowed to participate in a particular Zigbee network. For application-controlled security, see section 2.4 APS Layer Security.

2.1.1 The Network Key

Network security uses a network-wide key for encryption and decryption. All devices that are authorized to join the network have a copy of the key and use it to encrypt and decrypt all network messages. The network key also has a sequence number associated with it to identify a particular instance of the key. When the network key is updated, the sequence number is incremented to allow devices to identify which instance of the network key has been used to secure the packet data. The sequence number ranges from 0 to 255. When the sequence number reaches 255, it wraps back to 0.

Note: All Zigbee keys are 128-bits in length.

All devices that are part of a secured Zigbee network have a copy of the network key.

2.1.2 Hop-by-Hop Security

It is important to note that network security in Zigbee is done on a hop-by-hop basis. Each router that relays an encrypted packet first verifies that it is a valid encrypted packet before any more processing is done. A router authenticates the packet by executing the Zigbee decryption mechanism and verifying the packet integrity. It then re-encrypts the packet with its own network parameters (such as source address and frame counter) before sending the message to the next hop. Without this protection, an attacker could replay a message into the network that would be routed through several devices, thereby consuming network resources. Using hop-by-hop security allows a router to block attempts to inject bad traffic into the network.

2.1.3 Packet Security

A packet secured at the network layer is composed of the elements shown in the following figure.

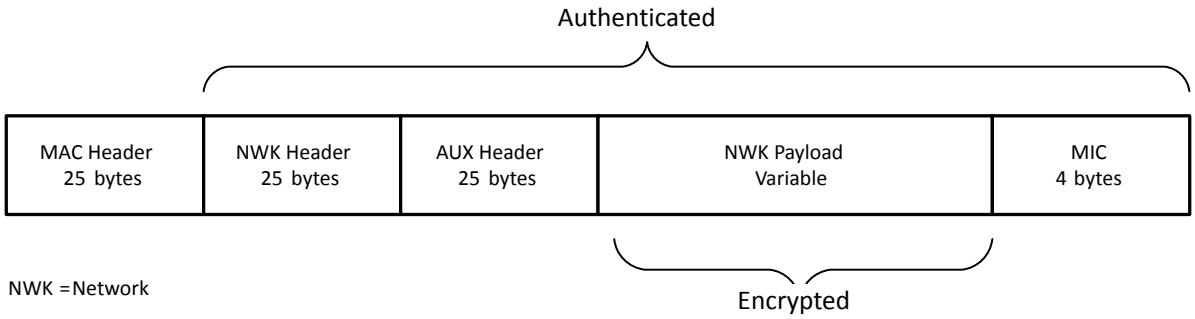


Figure 2.1. Anatomy of a Packet Secured at the Network Layer

2.1.4 Auxiliary Header

The auxiliary header contains data about the security of the packet that a receiving node uses to correctly authenticate and decrypt the packet. This data includes the type of key used, the sequence number (if it is the network key), the IEEE address of the device that secured the data, and the frame counter.

2.1.5 Authentication and Encryption

Zigbee uses a 128-bit symmetric key to encrypt all transmissions at the network layer using AES-128. The network and auxiliary headers are sent in the clear but authenticated, while the network payload is authenticated and encrypted. AES-128 is used to create a hash of the entire network portion of the message (header and payload), which is appended to the end of the message. This hash is known as the Message Integrity Code (MIC) and is used to authenticate the message by insuring it has not been modified. A receiving device hashes the message and verifies the calculated MIC against the value appended to the message. Alterations to the message invalidate the MIC and the receiving node will discard the message entirely.

Note: Zigbee uses a 4-byte MIC.

2.1.6 The Network Security Frame Counter

A frame counter is included in the auxiliary headers as a means of protecting against replay attacks. All devices have their own outgoing frame counter and they maintain a list of their neighbor's and children's frame counters. Every time a device sends a packet, it increments its outgoing frame counter. A receiving device verifies that the frame counter of the sending device has increased from the last value that it saw. If it has not increased, the packet is silently discarded. If the receiving device is not the final destination, the packet is decrypted and modified to include the routing device's frame counter. The packet is then re-encrypted and sent along to the next hop.

The frame counter is 32 bits and may not wrap to zero. The network key can be updated before the frame counter reaches its maximum value. When that occurs, the frame counter may be reset to zero if the local device’s value is above 0x80000000.

2.1.7 Unencrypted Network Data

All normal network datagrams are required to have network security and a valid frame counter. The only exception is during joining, when devices do not yet have the network key. In that case a joining device's messages are relayed through its parent until it is fully joined and authenticated. Any other messages that are received without network layer security are silently discarded.

2.2 Trust Center Networks

Authentication in a secure network may be controlled by means of a central authority known as a trust center. All devices entering the network are temporarily joined to the network until the trust center is contacted and decides whether or not to allow the new device into the network. The parent of the newly joined device acts as a relay between the trust center and the joining device. Only authentication messages can be sent to or from the device until it is fully joined and authenticated.

The trust center has the option of doing one of three things when a device joins:

- Send a copy of the current network key, which the parent relays to the joining device.
- Send the parent a command to remove the device from the network, thereby disallowing it from joining.
- Ignore the request. Parents will silently remove the device from the network if it does not receive a network key within 2 seconds.

Once the node has the network key, it is considered fully joined and authenticated, and may communicate with any device on the network.

A network operating with a trust center always needs a trust center to authenticate any new devices. Normal messages between two devices do not require the trust center to get involved.

2.3 Distributed Trust Center Networks

Networks may be formed without a centralized authentication. These networks are called **Distributed Trust Center Networks**. In this case, any router may authorize and authenticate new devices that wish to join.

These networks offer a simpler mechanism for adding devices to the network at the slight expense of a less secure network.

The decision to use a Distributed Trust Center Network or a Trust Center Network is done at the time the network is formed. There is no way to change this decision after the network has been started.

2.4 APS Layer Security

This section describes how Zigbee implements security at the Application Support (APS) layer.

2.4.1 End-to-End Security

APS security is intended to provide a way to send messages securely within a Zigbee network such that no other device can decrypt the data except the source and destination. This is different than network security, which provides only hop-by-hop security. In that case every device that is part of the network and hears the packet being relayed to its destination and decrypts it.

APS security uses a shared key that only the source and destination know about, thus providing end-to-end security.

Both APS layer and network layer encryption can be used simultaneously to encrypt the contents of a message. In that case APS layer security is applied first, and then network layer security.

A packet secured at the APS Layer is composed of the elements shown in the following figure.

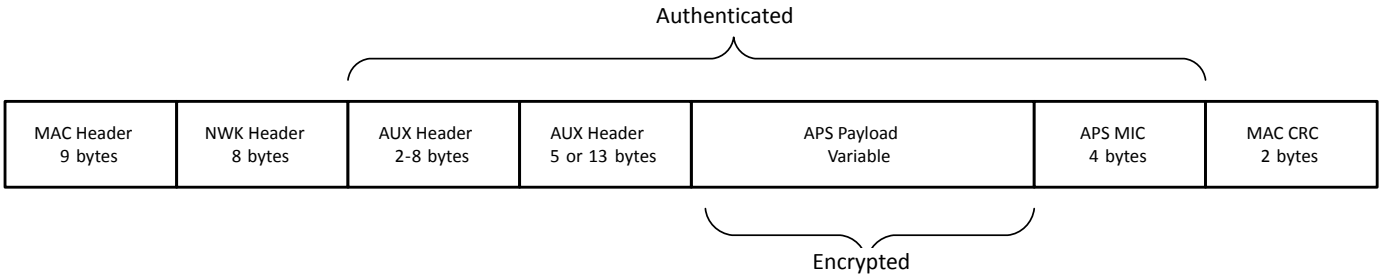


Figure 2.2. APS Packet Security

2.4.2 Link Keys

APS security uses a peer-to-peer key known as the link key. Both devices must have already established this key with one another before sending APS-secured data. There are two types of link keys: trust center link keys and application link keys.

Trust Center Link Keys

The **trust center link key** is a special link key in which one of the partner devices is the trust center. The stack uses this key to send and receive APS command messages to and from the trust center. The application may also use this key to send APS-encrypted data messages.

All devices in a Zigbee network must have link keys. In a Trust Center Network, the devices must have a Trust Center Link Key. In a Distributed Trust Center Network, this key is called a **Distributed Trust Center link key**.

Application Link Keys

Application link keys are shared keys that may be established between any two nodes in the network, where neither device is a trust center. They may be used to add additional security to messages being sent to or from the application running on a node. Devices can have a different application link key for each device with which they communicate.

A device may preconfigure an application link key or request a link key between itself and another device. In the latter case it issues a request to the trust center encrypted with its trust center link key. The trust center acts as a trusted third party to both devices, so they can securely establish communications with one another. This is discussed further in section 3.2.4 Application Link Keys. The process for establishing an application link key is illustrated in the following figure.

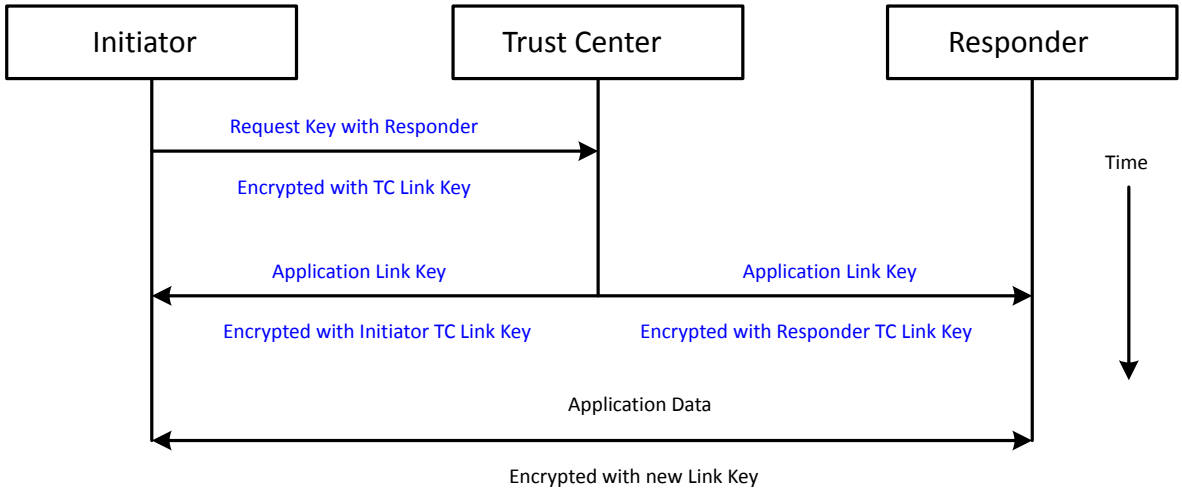


Figure 2.3. Establishing an Application Key

2.4.3 Unencrypted APS Data

APS layer security operates independently of network layer security. It is required for certain security messages (APS commands) sent to and from the trust center by the Zigbee stack.

Unlike network security, APS security for application messages is optional. Application messages are not automatically encrypted at the APS layer and are not ignored on the receiving side if they do not have APS encryption. Individual applications may choose whether to accept or reject messages that do not have APS layer security. For example, the Smart Energy profile describes what Zigbee Cluster messages are required to have security.

3. Standard Security

3.1 Overview

Standard security, introduced in the Zigbee 2007 specification along with Zigbee PRO, is the security model being used in all Zigbee application profiles and in Zigbee 3.0. It is the only security model supported by the EmberZNet PRO stack libraries.

Standard security uses network keys and link keys to encrypt data at the network and application layers, respectively. The application support (APS) layer security allows the trust center to securely transport the network key to joining or rejoining nodes, and it optionally allows applications to add additional security to their messages. Network (NWK) layer security is used to secure all traffic sent on a Zigbee network, with the exception of basic MAC layer communication such as association, data requests (polling), and MAC ACKs.

3.2 Use of Keys in Standard Security

Standard security defines different keys used for securing data in different ways. All keys are 128-bit symmetric and may or may not be used for encrypting/decrypting packets.

3.2.1 Network Key

This is the network-wide key used to secure transmissions at the network layer. Standard security requires the use of a shared network key among all devices in the network. The trust center may periodically update and switch to a new network key. The trust center may use either a broadcast update or a unicast update. In the broadcast case, the trust center first broadcasts a new network key encrypted with the old network key. In the unicast case, the trust center sends a new network key to each device encrypted with that device's corresponding Trust Center Link Key.

In both cases, the trust center later tells all devices to switch to using the new network key. The new network key has a sequence number that is one higher than the last sequence number.

3.2.2 Trust Center Link Key

This key (known simply as the link key) is used for secure end-to-end communications between two nodes, one of which is the trust center. The trust center link key is used in these cases:

1. Encrypting the initial transfer of the network key to a joining node.
2. Encrypting an updated copy of the network key to a rejoining node that does not have the current network key.
3. Routers sending or receiving APS security messages to or from the trust center. These may be updates informing the trust center of a joining or rejoining node, or a command sent by the trust center to a router to perform some security function.
4. Application unicast messages that enable APS encryption, where either the sending or receiving device is the trust center.

The trust center has the option of deciding how to manage the trust center link keys. It may choose unique keys for each device in the network, keys derived from a common piece of shared data (the IEEE address of the device), or a global key that is the same for all devices in the network. Trust center link keys may also be negotiated at the application layer using a key establishment protocol like Certificate-Based Key Establishment (CBKE).

3.2.3 Installation Code Keys

Zigbee 3.0 now has installation code keys, which were previously only available to Smart Energy networks. All Zigbee 3.0 certified devices are required to have them, but use in the network is decided by the trust center. Smart Energy networks are required to always use them.

An installation code key is just a preconfigured trust center link key used to enter the Zigbee network and obtain the current network key. Because this unique key must be known to both the joining device and the trust center at the time of network entry, a piece of shareable data known as the "installation code" (sometimes also referred to as the "install code") is used to derive the key at both sides. The code can be any arbitrary value of 6, 8, 12, or 16 bytes, followed by a 16-bit CRC (least significant byte first) over those bytes.

This code is then used as the input to a Matyas-Meyer-Oseas (MMO) hash function (as specified in Zigbee Document 053474, the Zigbee Specification), with a digest size (hash length) equal to 128 bits. The 128-bit (16-byte) result of this AES-MMO hash function is used as the value for the preconfigured trust center link key for that device, and the trust center can then install a key table entry with that key and the EUI64 of the joining device, which then allows the authentication to take place successfully during joining, and the joining device can successfully receive and decrypt the network key delivery.

As part of this process, the installation code and the joining device's EUI64 must be conveyed out-of-band (outside of the target Zigbee network, since the new node is not yet joined) to the network's trust center to allow the proper link key table entry to be created. This communication may involve the device installer contacting the network administrator (the party responsible for the trust center, such as a utility company) by telephone or Internet to provide the necessary information.

3.2.4 Application Link Keys

Standard security supports devices establishing application link keys with other devices. These keys are separate from the trust center link key and not required for normal operation. They are used for APS-level encryption between two devices in the network, neither of which is the trust center.

Application link keys must be established separately from the trust center link key. Devices may not establish an application link key with the trust center. However the trust center link key can be used to APS-encrypt application messages to the trust center, or from the trust center to a device on the network.

Application link keys can be established in one of two ways:

1. Manual configuration by the application specifying the key associated with a destination device.
2. Through a request that the trust center generate a key and send it to both devices.

The application can manually configure a key by calling into the stack and setting one up. The partner device must also configure the application link key and negotiate with the other device when they can start using that key.

Application link keys can also be established using the trust center. The EmberZNet PRO stack supports two methods for this. The first is the Zigbee standard method, discussed in section 2.4.2 Link Keys and illustrated in Figure 2.3 Establishing an Application Key on page 6, where one device requests an application link key with another device by contacting the trust center. The trust center then immediately responds and sends a randomly generated application link key back to the requesting device and to the partner device. The drawback with having only one device request a key is that the other device may be asleep, offline, or have insufficient capacity to hold another key.

The second method, shown in the following figure, is **not standardized in Zigbee** and will be non-interoperable with other vendors' devices. It also requires that all devices in the exchange are configured to use this method, including the trust center. It is more reliable in that it helps ensure that the partner device is online and able to receive an application link key. In this case, both devices must request an application link key from the trust center. The trust center stores the first request for an application link key for a period of time defined by the trust center application. During that time, the partner must send in its own application link key request with the first device as its partner. If that occurs, then the trust center generates a random application link key and sends it back to both devices. Requiring both devices to request an application link key greatly reduces the chance that a device or its partner will not receive the key.

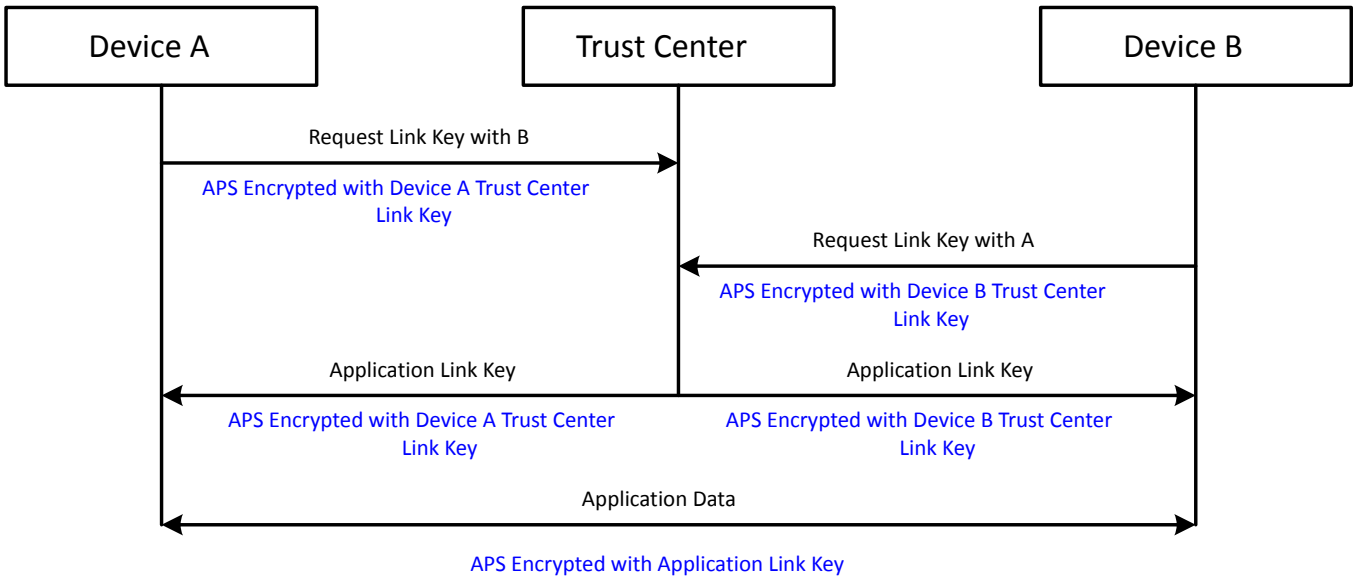


Figure 3.1. Requesting an Application Link Key with Another Device on the Network

EmberZNet PRO supports a configurable table for storing application link keys.

3.3 Joining a Network

A device initiates the process of joining a Zigbee standard security network by first using MAC association to join to a suitable parent device. If the association is successful, the device is joined but unauthenticated, as it does not possess the network key.

After sending the success response to the MAC association request, the router sends the trust center an Update Device message indicating that a new node wishes to join a Zigbee network. The trust center can then decide whether or not to allow the device to join. If the device is not allowed to join, a Remove Device request is sent to the parent, as shown in the following figure. If the device is allowed to join, the trust center's behavior depends upon whether the device has a preconfigured link key.

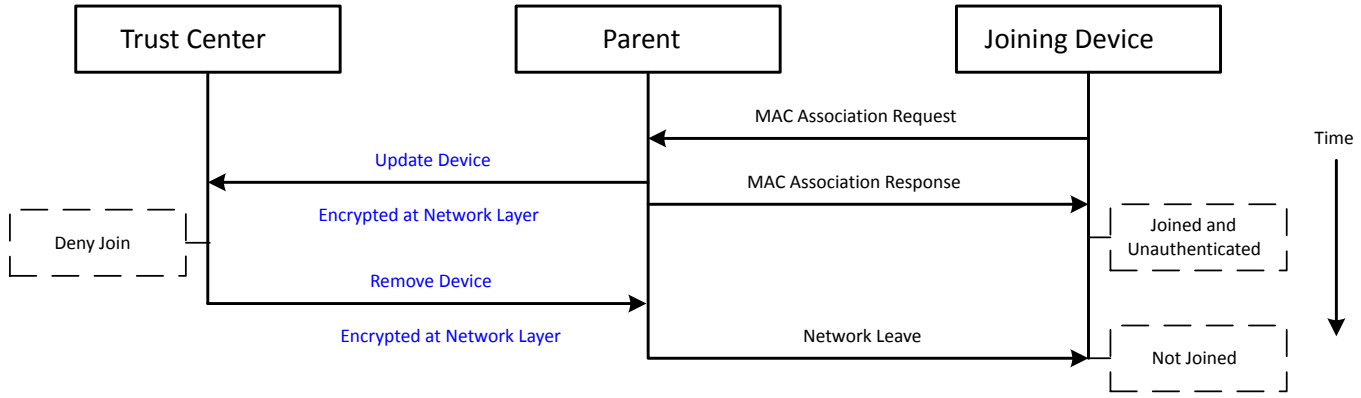


Figure 3.2. A Device that is Denied Access to Join the Network

3.3.1 Preconfigured Link Keys

The trust center dictates the policy of how to handle new devices and determines whether a device should have a preconfigured link key. If a new device does not have a preconfigured link key, it will be unable to join the network.

The trust center has the option of choosing whether that key is the well-known default link key (ZigBeeAlliance09) or a pre-shared installation code key. The following figure illustrates the joining process with a preconfigured key.

To allow a device onto the network, the trust center transmits the network key encrypted with the device's preconfigured link key.

Zigbee 3.0 and all Zigbee application profiles require a preconfigured link key for joining.

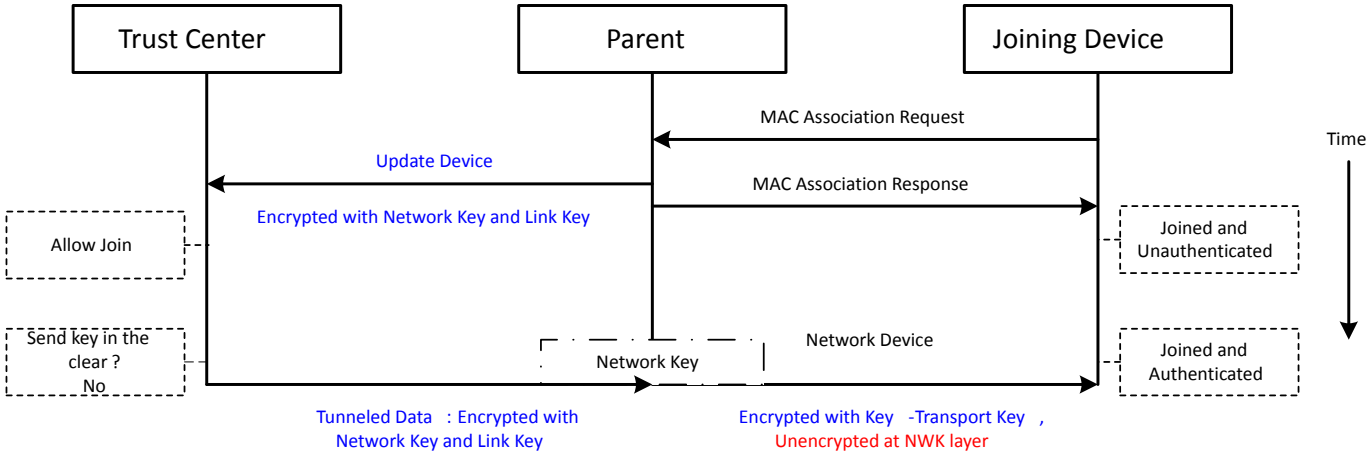


Figure 3.3. Joining Using a Preconfigured Trust Center Link Key

3.3.2 Decision for Using Well-Known Key or Installation Code

The choice of whether the trust center shall use a well-known key or an installation code is based on a balance between ease of use and security.

Using a well-known key enables devices to join more easily without much user interaction. However, encrypting the network key with a well-known key provides a moment of vulnerability until that well-known key is replaced with a new key.

Using an installation code provides security for the initial exchange of the network key to the device, at the cost of added interaction between the user and the trust center. A user must somehow transfer the key from the device to the trust center. This is accomplished by a mechanism outside of the Zigbee network, such as typing in the code into the trust center GUI from a label listing the code on the joining device.

The main application running on the network will help dictate whether ease of use versus better security is more important. The Zigbee standard allows either.

3.3.3 Requesting a New Link Key after Joining

Zigbee 3.0, devices are required to request an updated trust center link key after successfully joining. This will replace their existing preconfigured key and be used as long as they are joined to that network. Even if the device joined used an installation code key, that key will be replaced. The following figure illustrates how Zigbee 3.0 devices update the Trust Center Link Key.

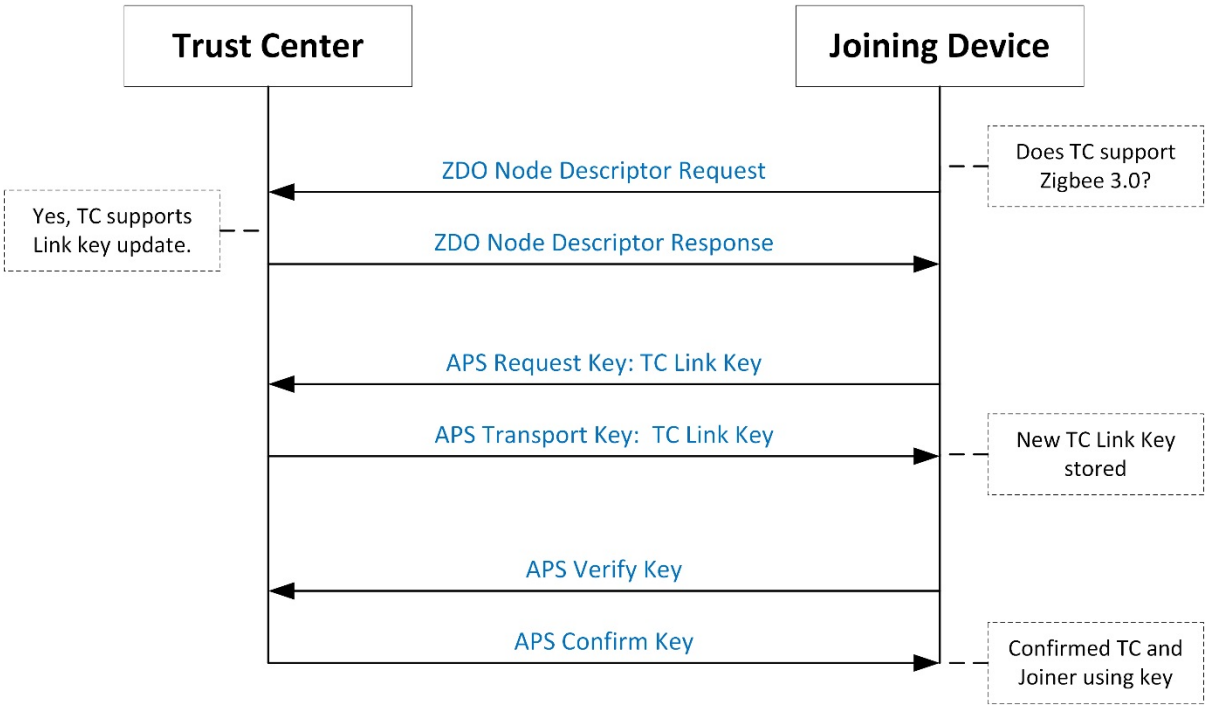


Figure 3.4. Zigbee 3.0 Devices updating the Trust Center Link Key

This replacement of the key is only possible if both the Trust Center and Joining Device support Zigbee 3.0. If one of them does not, then the original joining key will be kept.

3.4 Network Key Updates

The network key encrypts all transmissions at the network layer. As a result, a local device constantly increases its local network key frame counter. Before any device in the network reaches a frame counter of all F's, the trust center should update the network key. Since it is not possible for the trust center to know the frame counter value of every device in the network at any given time, or even to inspect the frame counters of incoming messages, an approach that relies on specific frame counter thresholds is not practical. Thus, a preventative maintenance approach relying on periodic updates at long intervals, similar to what is described below, is recommended.

The recommended model is for the trust center to periodically update the network key to help minimize the risk associated with a particular instance of the network key being compromised. This helps to ensure that a device that has left a secured Zigbee Network is not able to rejoin later.

Key updates may either be broadcast or unicast. The trust center decides which mechanism it wants to use. Both mechanisms are described in the sections that follow.

3.4.1 Broadcast Network Key Update

When the key update is broadcast the message is encrypted using the current network key. Devices that hear the broadcast do not immediately use the key, but simply store it. Later, a Key Switch is broadcast by the trust center to tell all nodes to start using the new key. At a minimum, the trust center should allow adequate time (approximately 9 seconds) for the broadcast of the new key to propagate throughout the network before switching. In addition, a trust center must keep in mind that sleeping end devices may miss the initial broadcast unless they poll frequently.

The broadcast mechanism is very simple because it does not require knowing the identity of all devices on the network. It also only involves sending two messages, an updated key message and a switch message.

3.4.2 Unicast Network Key Update

For a unicast update the trust center will send an individual key update to each device on the network. The trust center must have been previously maintaining a list of all authorized devices on the network in order to do this.

The update message is unicast to each device with APS encryption using each device's specific trust center link key. Later a key switch is broadcast by the trust center to tell all nodes to start using the new key.

3.4.3 Missing a Network Key Update

It is possible that any device may miss a key update. This may happen because it was sleeping, was powered off, or dropped off the network for an extended period of time. If this occurs, a device may try to perform a trust center rejoin. The trust center can then decide whether to allow the node back on the network.

The EmberZNet PRO stack can detect the condition where an encrypted packet arrives secured with a newer network security key. It will automatically perform a trust center rejoin to its current network to attempt to acquire the latest network key.

3.5 Network Rejoin

Rejoining is a way for a node to reconnect to a network of which it was previously part. Rejoining is necessary in two different circumstances:

1. Mobile or sleepy devices that may no longer be able to communicate with their parent.
2. Devices that have missed the network key update and need an updated copy of the network key.
3. Devices that have missed a PAN ID update and need to discover the network's new PAN ID.

When a device tries to rejoin, it may or may not have the current network key. Without the correct network key, the device's request to rejoin is silently ignored by nearby routers.

Therefore, a device has two choices when rejoining: a secured rejoin or a trust center rejoin. Note that neither of these rejoin cases requires the MAC Permit Association (also known as "permit joining") flag to be set on any devices in the target network. A router/coordinator device will always provisionally accept a NWK layer Rejoin command that is either a trust center rejoin or a secured rejoin with the active network key. For the trust center rejoin case it is then the trust center's responsibility to grant or deny access once it is notified of the rejoining device.

3.5.1 Secured Rejoining

A secured rejoin is the simpler case and a device seeking to rejoin the network should try this method first. If it has the current network key, the device will be able to communicate on the network again very quickly. A secured rejoin is only necessary when a sleepy or mobile end device has lost its parent.

As illustrated in the following figure, the device sends its rejoin request encrypted with its copy of the network key. If a router is nearby and is using the same network key, the rejoin response is sent back to the device encrypted. The device is now joined and authenticated on the network again. The parent that answered the rejoin request informs the trust center that the device rejoined, but no further action must be taken by the trust center.

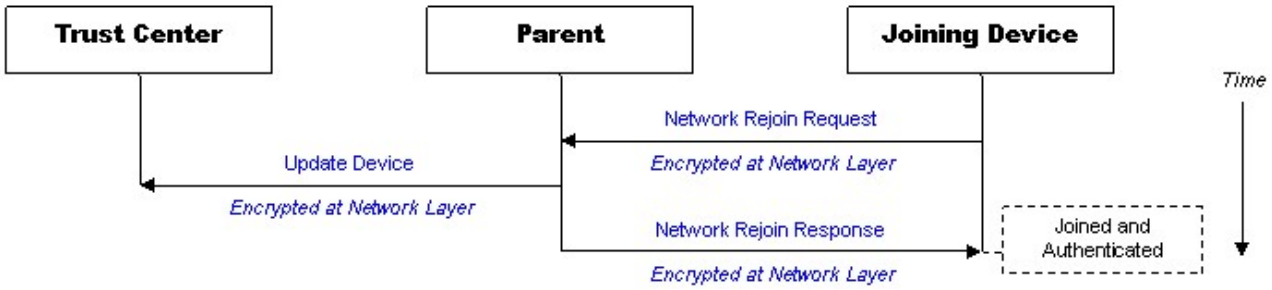


Figure 3.5. Secured Rejoin

If the secured rejoin fails and the device is using standard security, the application can try an unsecured rejoin.

3.5.2 Trust Center Rejoin

A trust center rejoin is necessary when neighboring devices have switched to a new network key and no longer use the same network key as the rejoining device. To succeed in the trust center rejoin, the device must have a trust center link key. The device sends the rejoin request unencrypted. A nearby router accepts the unencrypted rejoin request and responds to the device, allowing it to transition to the joined and unauthenticated state.

As illustrated in the following figure, the parent of the rejoining device sends an Update Device message to the trust center, informing it of the unsecured rejoin. The trust center has two choices: deny or accept the rejoin. If it accepts the rejoin, it must send an updated network key to the device. However, it secures this message using that device's trust center link key. The message is sent to the parent of the rejoining device encrypted at both the network and APS layers. The parent then relays this message without network encryption to the rejoining device. Once it has the network key, it will be in the joined and authenticated state and can communicate on the network again.

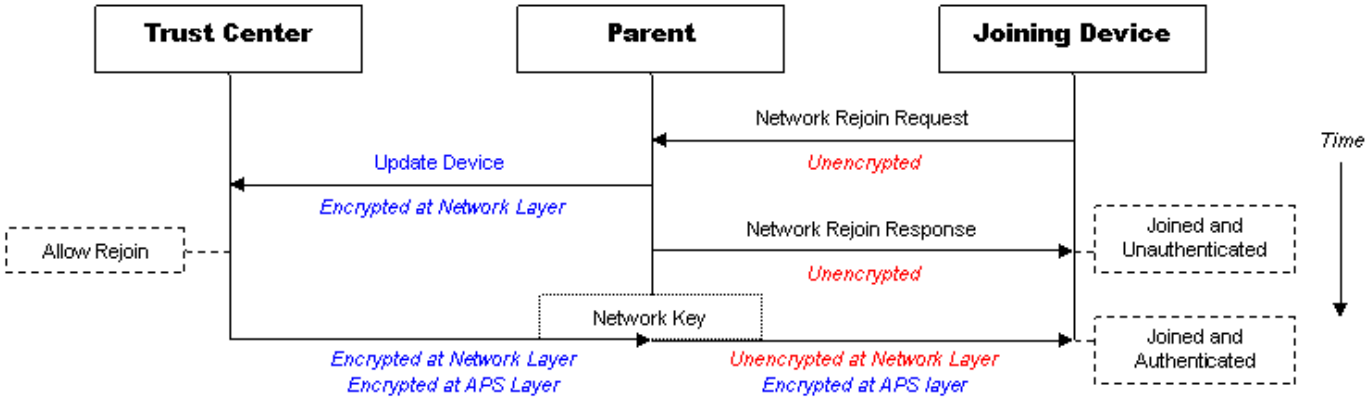


Figure 3.6. Trust Center Rejoin

Important Note on allowing Trust Center Rejoins

The decision by a Trust Center to allow a Trust Center Rejoin **must** consider whether or not a well-known link key is in use by the requesting device. If a Trust Center has only a well-known link key for the device (such as the ZigBeeAlliance09 key), it should silently ignore the rejoin request. The well-known link key is insecure and thus should not be used for transporting an important piece of data, namely the network key, during a rejoin. A well-known link key should only be used for initial joining and even then it should allowed only for a brief period of time.

Pre-Zigbee 3.0 networks did not update the Link Key after joining and thus it is highly recommended to reject any attempt to rejoin with the well-known link key. Smart Energy networks are not subject to this particular issue because they update their link key after joining. (See section 4. Zigbee Smart Energy (ZSE) Security, for more details.) Therefore, Smart Energy networks may accept Trust Center Rejoins.

The recommendation to *silently* reject the Trust Center Rejoin as opposed to *explicitly* reject the Trust Center rejoin is necessary to maintain compatibility with pre-Zigbee 3.0 devices. Pre-Zigbee 3.0 devices may try Trust Center Rejoins without considering whether an updated link key has been obtained previously. If an explicit rejection is done, the device may drop off the network entirely while a silent rejection would allow the device to try a Secure Rejoin later and succeed.

The Trust Center's behavior will ultimately determine the security of the Trust Center Rejoin attempt. This can be controlled via the `emberTrustCenterJoinHandler()` callback on the SOC or via the `ezspSetPolicy()` API configuring the EZSP_TRUST_CENTER_POLICY for an NCP. To silently ignore the Trust Center Rejoin on an SOC device, the `emberTrustCenterJoinHandler()` must return EMBER_NO_ACTION. To silently ignore the Trust Center Rejoin on an NCP, the host must set the EZSP_TRUST_CENTER_POLICY to EZSP_IGNORE_TRUST_CENTER_REJOINS.

An End Device is advised to restrict itself to only performing Secure Rejoins if it has only a well-known link key. This is the default in the End Device Support Plugin of the latest release of EmberZNet PRO. However, as noted above, it is the Trust Center's behavior that ultimately guarantees the safety of the network.

3.6 Trust Center Decision Process Summary

The following figure illustrates the decision tree for the trust center when a device joins the network. The parent of a joining or rejoining device sends an Update Device APS command to the trust center, indicating the event has taken place. The trust center application decides what to do based on that information. This figure describes the behavior for a Zigbee PRO device joining a Zigbee PRO network using standard security.

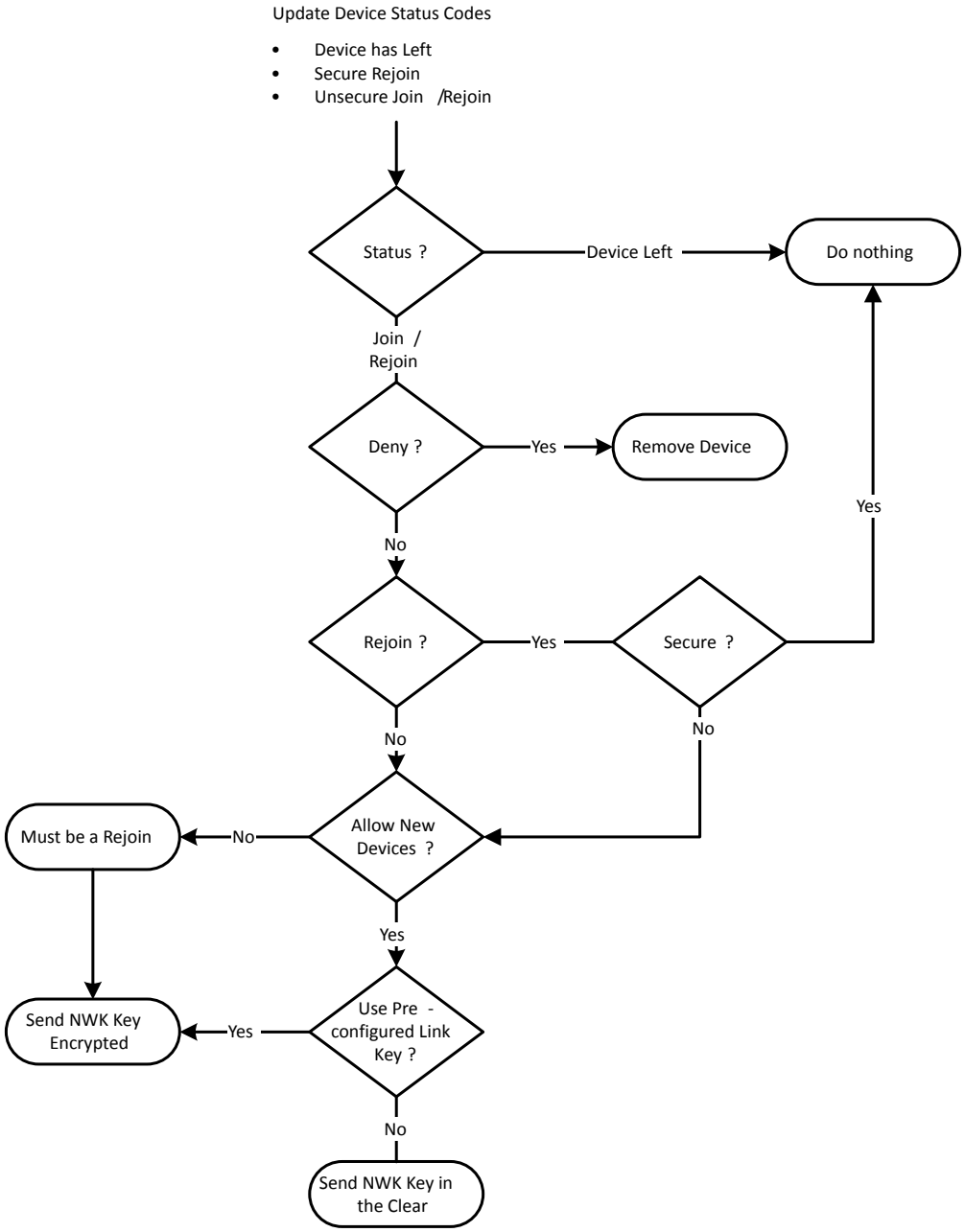


Figure 3.7. Decision Process for the Trust Center

The trust center can decide whether or not to allow devices into a Zigbee network and whether or not to send the key in-the-clear. The trust center's decision can be made based on any number of additional factors, such as a user event (button press), a time-based condition, IEEE address of the joining device, or some other condition (such as, the network is being commissioned).

When new devices join, the trust center decides whether the device should have a preconfigured key. The joining devices have no ability to inform the trust center through the Zigbee protocol about whether or not they have a preconfigured key.

3.6.1 Distributed Trust Center Mode

Normally a joining device is authenticated by the trust center through its parent. This is advantageous, as it allows one device to act as a gatekeeper and authenticate all devices that want to join the network. Security messages are relayed to the joining device through its parent until it becomes joined and authenticated.

However, this means that all routers must have a route to the trust center and vice versa. When applications are being developed or when commissioning a network, the trust center may not be reachable, and thus devices cannot join or perform a trust center rejoin.

The EmberZNet PRO stack allows a network to use standard security features without a trust center. This is known as distributed trust center mode. This mode is Zigbee-compliant as of Zigbee 3.0. This mode has the advantage of permitting devices to join without requiring the parent node to send information to the trust center and await the response. In this mode, all routers mimic the behavior of a trust center by sending the security data directly to the joining node. Each router individually decides whether or not to let the device onto the network. This mode is useful to allow commissioning of a complete network and then establishment of a trust center for security.

In this mode, all devices use a single trust center link key is preconfigured. All devices inherit the distributed trust center setting from their parent when they join and also operate in that mode. Thus, only the device that forms the network (the coordinator) needs to be set up to run in distributed trust center mode.

3.7 Additional Requirements for a Trust Center

To function correctly in a Zigbee PRO network, a trust center also requires that:

1. The trust center application must act as a concentrator (either high or low RAM).
2. The trust center application must have support for source routing. It must record the source routes and properly handle requests by the stack for a particular source route.
3. The trust center application must use an address cache for security, in order to maintain a mapping of IEEE address to short ID.

Failure to satisfy all of the above requirements may result in failures when joining/rejoining devices to the network across multiple hops (through a target node that is neither the trust center nor one of its neighboring routers.)

3.7.1 Trust Center as a Concentrator

The trust center must act as a concentrator because Zigbee PRO Security requires two-way routes to and from the trust center in order to transmit all the security messages necessary to transition a device to the joined and authenticated state.

Routers running the EmberZNet PRO stack automatically add a route to the trust center through their parent (the device they joined to) immediately after they become joined and authenticated. This route assumes that the trust center is acting as a low RAM concentrator.

The trust center should periodically broadcast the many-to-one route message, so that all routers update their routing tables and repair broken routes to the trust center. This also allows it to notify routers if it is acting as a high RAM concentrator, thereby updating the default route.

3.7.2 Trust Center and Source Routing

The trust center must have support for source routing in the application. It should record the routes of incoming messages and store them in its own table. If the trust center is acting as a high RAM concentrator, it must keep track of all source routes. If the trust center is acting as a low RAM concentrator, then only the last couple of source routes must be recorded. The minimum number of entries in the source route table should be sized to support the maximum number of simultaneous security events that may occur at one time. These security events include rejoins, joins, and leaves.

In addition to storing the source routes, the trust center must also implement the proper hooks to respond to requests by the stack for a particular source route. Silicon Labs provides a source route library, which manages a source route table, and works with a trust center.

Note: For EZSP host-based designs, the source route table on the host cannot be used for routing security messages sent to devices joining or rejoining the network. This is because the APS security transactions must be handled by the stack (in the network coprocessor) without relying on application level interaction (at the host.)

3.7.3 Trust Center Address Cache

In order to properly decrypt APS-encrypted messages, the trust center must maintain a mapping of IEEE address to short ID.

For a high RAM concentrator, the trust center must keep track of all devices in the network.

For a low RAM concentrator, the trust center need only keep track of a couple of entries at a time and may overwrite old entries as needed. The size of the cache should be equal to the maximum number of simultaneous security events that can occur at one time. These security events include rejoins, joins, and leaves.

Silicon Labs provides sample code for implementing the trust center address cache mechanism on SoC platforms (See app/util/security/security-address-cache.c). The EZSP network coprocessor binaries contain the security address cache feature as part of the firmware, but the host application must enable this feature by setting the Trust Center Address Cache Size to a value greater than 0. (Refer to document *UG100: EZSP Reference Guide*, for details on how to do this in your EZSP host implementation.)

4. Zigbee Smart Energy (ZSE) Security

Due to the complexity of the security used in ZSE application development, this topic has been given its own section detailing the implementation and design requirements and considerations. Note that more detailed information about ZSE can be found in the Zigbee Smart Energy Profile Specification, available at the http://www.zigbee.org website.

4.1 Overview

The following sections explain the concepts used in ZSE to provide additional security beyond the Zigbee standard security.

4.1.1 Installation Codes

ZSE devices are required to have and use installation code keys to join the network. Trust centers are required to use them as part of the initial joining. For more information, see section 3.2.3 Installation Code Keys. You can find more information about this process in the “Out of Band PreConfigured Link Key Process” section of the Zigbee Smart Energy Profile Specification.

4.1.2 Certificates and Key Establishment

Once a device has joined to the ZSE network and obtained the network key from the trust center, the new device must then re-negotiate its trust center link key through certificate-based key establishment (CBKE). This takes the place of the Zigbee 3.0 link key request mechanism that a standard Zigbee 3.0 device uses.

The CBKE protocol ensures that the new link key is unrelated to the preconfigured key, ensures that the key that is random and unreproducible, and provides proof of identity by validating the authenticity of the certificates at both devices. The new link key derived from this CBKE process replaces the original, preconfigured trust center link key, such that the preconfigured key is not used again unless this new ZSE device is purged from the network and later needs to re-enter. ZSE networks require that a CBKE-based link key shall be used for unicast data communications on most ZSE clusters. (Refer to the “Cluster Usage of Security Keys” section of the Zigbee Smart Energy Profile Specification for details about which clusters require only Network layer security and which require both Network and APS layer security.)

CBKE is variation of Public Key-Key Establishment (PKKE, as opposed to SKKE, Symmetric Key-Key Establishment) between a pair of devices. PKKE is a process whereby a link key is established based on each party’s shared, static, public key and ephemeral, public key. Since these keys are public, they do not require secrecy in their storage and transmission. These keys by themselves (without the non-public certificate data) aren’t enough to recreate the key, so knowledge of these public keys doesn’t compromise the established link key. In CBKE, specifically, each device’s static, public key is transported as part of a device-implicit certificate signed by the sender’s certificate authority (CA), allowing the receiver to validate the device’s identity during key establishment; this differs from traditional PKKE, where certificates are manually created.

The digital certificates used in the CBKE process are programmed into each device at manufacturing time and are issued by the CA. For the process to complete successfully, both devices must contain certificates signed by the same CA. For ZSE networks using Smart Energy 1.x protocol versions, Certicom (www.certicom.com) is the only Zigbee-approved certificate issuer. Certicom offers certificates signed by either of the following CAs:

- Test SE CA – A special certificate authority used exclusively for non-commercial testing purposes. Certificates signed by this CA are free to generate through Certicom’s website.
- Production CA – The normal certificate authority used by Certicom to sign certificates for production-grade devices used in commercial deployments. These certificates require paid licensing terms with Certicom to generate and will not interoperate with test certificates signed by the Test SE CA.

There are actually two different certificate formats and two different elliptic curves used by the CBKE protocol. The original Smart Energy 1.0 specification used the sect163k1 elliptic curve, which has a symmetric key equivalent length of 80 bits. The NIST standard recommends a key strength of 128 bits for all devices deployed after 2010 (see NIST sp800-57-part1, Table 4: Recommended algorithms and minimum key sizes). Therefore, the Smart Energy 1.2 standard introduced a second elliptic curve, sect283k1, which has an equivalent strength of128 bits. Smart Energy labelled the certificate format and curve for the SE 1.0 release Crypto Suite 1, and labelled the certificate format for the SE 1.2 release Crypto Suite 2. Smart Energy 1.2 supports both Crypto Suite 1 and Crypto Suite 2.

The Crypto Suite 1 certificate data stored on each device consist of the fields described in the following table.

Table 4.1. ZSE Crypto Suite 1 Stored Certificate Fields

Size (bytes)	Name	Description
22	CA Public Key	Public key specific to the CA who signed the certificate. During CBKE, this is used to verify the authenticity of the CA.
48	Device-Implicit Certificate	Unique data signed by the CA (using the CA’s private key) and representing the digital certificate for this specific device. This is the portion of the certificate that is shared over the air during CBKE and contains the following subfields: • Reconstruction data for the device’s public key (22 bytes) • This device’s IEEE MAC address (EUI64), also known as the Subject for the certificate (8 bytes in MSB order) • Issuer ID for the CA who created this device-implicit certificate (8 bytes in MSB order) • Profile-specific data as defined by the Zigbee application profile using the certificate. The first 2 bytes represent the 16-bit Zigbee application profile ID (in most significant byte notation), such as 0x0109 for ZSE. (10 bytes)
21	Device Private Key	A unique, device-specific value chosen by the CA during certificate generation. During CBKE, this is used as an input (along with the Device Public Key) to an Elliptic-Curve Cryptography (ECC) algorithm.

The Crypto Suite 2 certificate data stored on each device consist of the fields described in the following table.

Table 4.2. ZSE Crypto Suite 2 Stored Certificate Fields

Size (bytes)	Name	Description
1	Type	Type of certificate. 0 = implicit, no extensions.
8	SerialNo	Serial Number of the Certificate
1	Curve	Curve identifier (sect283k1 is 0x0D)
1	Hash	Hash identifier (AES-MMO is 0x08)
8	Issuer	8 byte identifier, 64-bit IEEE 802.15.4 address.
5	ValidFrom	40-bit Unix time from which the certificate is valid.
4	ValidTo	32-bit # seconds from the ValidFrom time for which the certificate is considered valued (0xFFFFFFFF = infinite)
8	SubjectID	8 byte identifie, 64-bit IEEE 802.15.4 address
1	KeyUsage	Bit flag indicating key usage (0x88 = digital signature or key agreement allowed)
37	PublicKey	37-byte compressed public key value from which the public key of the Subject is reconstructed.

This certificate data is used at runtime to establish a shared secret (the new link key) through ECC computations along an elliptic-curve. While the computational details of the CBKE process are beyond the scope of this document, the Zigbee Cluster Library (ZCL) messages exchanged as part of this process are illustrated in the following figure. Additional details about this process can be found in Appendix C of the Zigbee Smart Energy Profile Specification. All of these messages are encrypted at the Network layer without any APS layer encryption. The Initiator in this process is typically the new device that entered the network, while the Responder is typically the trust center or Energy Services Interface (ESI) for the Home Area Network (HAN).

Figure 4.1. Message Exchange for Certificate-Based Key Establishment

Once the key has been established, it can be used for future ZSE-related transactions among this pair of devices. If ZSE-related communications is desired between another pair of devices in the HAN, the two devices can request a mutual link key by requesting one from the trust center (who is a trusted party by virtue of CBKE having succeeded). For information about third-party application link key requests, refer to section 3.2.4 Application Link Keys.

4.1.3 Application Layer Requirements

To ensure ZSE end-product compliance, ZSE devices have a number of special design requirements including the following, which make them unique from other Zigbee devices:

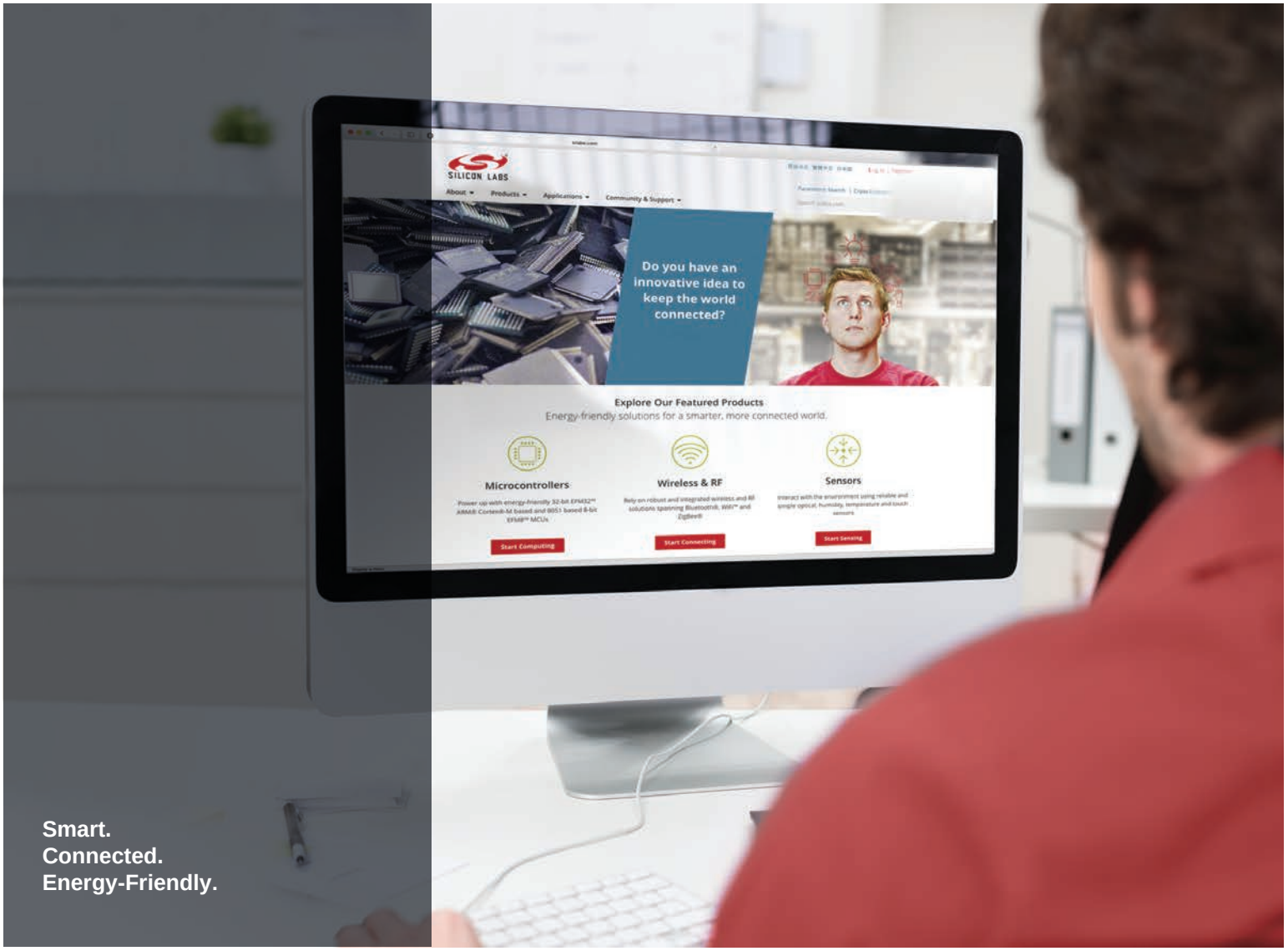
- Support for CBKE, including the underlying ECC algorithm used for key generation.
- Support for Elliptic-Curve Digital Signature Authentication (ECDSA), which is needed for validating firmware image data transferred in the ZSE Over-the-Air (OTA) Bootload cluster.
- Pre-installed certificates issued by the proper certificate authority.
- Pre-installed installation codes chosen by the manufacturer.
- Access to the installation code and EUI64 during network setup (to facilitate out-of-band communication of this data to the trust center).
- (Trust center only) Key table space sufficient to track the maximum number of ZSE devices that the HAN will support.
- (Trust center only) Support for inputting installation codes (for deriving link key values through an AES-MMO hash function) or link key values and EUI64 addresses (for creating key table entries for new devices).
- APS data messages for certain clusters (most ZSE clusters) require APS security with an application (or trust center) link key.

The Zigbee application framework ensures that the above requirements are satisfied (where applicable) when the Simplicity Studio AppBuilder tool is used to configure a ZSE device. For more information about Zigbee application framework-based development, refer to document *UG391: Zigbee Application Framework Developer's Guide*.

4.2 Additional Sources of Information

For more information regarding ZSE security concepts, refer to the following resources:

- *ZSE AMI Profile Specification* – Zigbee document #07-5356. This is the top-level application profile specification for ZSE and is available for public (non-member) download through the www.zigbee.org website.
- *AN708: Setting Smart Energy Certificates for Zigbee Devices*. This is an application note explains how to program the necessary manufacturing data (certificates and installation codes) into Silicon Labs chips using Silicon Labs programming tools and how to verify these data once they have been programmed into the device.
- *AN714: Smart Energy ECC-Enabled Device Setup Process*. This application note describes how to set up a device with the security resources required to support Smart Energy (SE) security. While these security resources are not necessary for testing SE networks, any devices designed to participate in or host a Zigbee-compliant, production-grade (non-test) SE network must implement these features.
- *UG391: Zigbee Application Framework Developer's Guide*. This document describes the command line interface (CLI) commands available for inspecting and altering the security configuration of a Silicon Labs Zigbee application framework-based application at runtime. It also explains any callbacks or plugins used by the Zigbee application framework to implement security behavior in the application.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com