Mesh Security Overview

***Bluetooth*® Informational Publication**

- **Version:** v2
- **Version Date:** 2025-10-13
- **Prepared By:** Mesh Working Group

If you are a SIG member and have any specific requested changes to this document, please submit an erratum in Jira:
https://bluetooth.atlassian.net/jira/software/c/projects/ES/issues

Abstract:

This Mesh Security Overview contains a high-level summary of the security toolbox provided by the Mesh Protocol specification and general recommendations on how to use the tools provided.

Version History

Version Number	Date (yyyy-mm-dd)	Comments
v1	2025-04-28	Approved by the Bluetooth SIG Board of Directors.
v2	2025-10-13	Approved by the Bluetooth SIG Board of Directors.

Acknowledgments

Name	Company
Robin Heydon	Qualcomm Technologies International Ltd.
Hannu Mallat	Silicon Labs
Jori Rintahaka	Silicon Labs
Szymon Slupik	Silvair, Inc.
Max Palumbo	Silicon Labs
Victor Zhodzishsky	Cypress Semiconductor
Omkar Kulkarni	Nordic Semiconductor ASA
Erik Anderlind	KiteSpring, Inc.

Contents

1 Introduction

The suite of Bluetooth® specifications for mesh networking opens up exciting new opportunities for building large, scalable networks. Expanding the horizons from tried-and-true, point-to-point Bluetooth connectivity to multipoint-to-multipoint connectivity using Bluetooth® Mesh Networking carries with it new security challenges. This Mesh Security Overview provides a high-level summary of the security features in the Mesh Protocol specification [1] and offers guidance for using these features to mitigate many of the common risks inherent in mesh networking.

A Bluetooth mesh network may consist of a multitude of devices that are used for different applications, such as lighting and sensor networks, deployed across an area that may have varying levels of physical security, such as indoors and outdoors. A mesh network can operate constantly for years, so the Mesh Protocol specification [1] provides the tools for planning, installing, and operating these networks securely for their entire lifetime.

The following is a summary of the key points in the security features of Bluetooth Mesh Networking, as in the Mesh Protocol specification [1]:

- All traffic within a Bluetooth mesh network is encrypted; the specification does not allow for sending unencrypted messages.
- Provisioning of a device (the process of adding a device to the network) is protected using Elliptic Curve Diffie-Hellman (ECDH) key agreement.
- Network traffic is protected using the Advanced Encryption Standard (AES) 128-bit Counter with Cipher Block Chaining-Message Authentication Code (CCM).
- Application traffic is encrypted separately using the AES-128-CCM with a set of keys that is different from the network traffic. This allows a division of concerns for distinct applications within the network.
- The AES-128-CCM requirements for mesh network communications are different from the requirements for communications between two devices used by the Bluetooth® Core Specification [3] Low Energy (LE) Secure Connections feature, such as nonce calculation.
- The way the ECDH shared secret is used for key derivation is different for mesh compared to the Bluetooth LE Security Manager ECDH used by the Bluetooth® Core Specification [3] LE Secure Connections feature.

Understanding mesh security concepts, potential security threats, and the means to counter those threats is useful to system architects, network deployment planners, and network operators.

This document assumes that the reader is familiar with the basics of Bluetooth Mesh Networking. An overview of the most important concepts and terms can be found in the Bluetooth Mesh Primer [5] as well as in the Mesh Protocol specification [1], Section 2.3. It is also assumed that the reader is familiar with the basic concepts in modern cryptography.

This document begins with an overview of mesh concepts in Section 2, followed by a description of provisioning security, network security, and application security in Sections 3 through 5. Network management procedures are discussed in Section 6. Readers already familiar with the Mesh Protocol specification [1] may choose to skip these sections.

Security threats in general are discussed in Section 7, and known vulnerabilities are listed in Section 8. Recommendations for mitigating the effect of threats are discussed in Section 9.

This is not a Bluetooth specification, therefore, the established Bluetooth SIG specification language conventions for use of the words ***shall***, ***shall not***, ***must***, ***should***, ***should not***, ***may***, and ***can*** do not apply to this document.

1.1 Specification versions

The specification that defines the protocol stack and security of communication for Bluetooth Mesh Networking was called Mesh Profile [4] for Version 1.0 and Version 1.0.1 and was renamed Mesh Protocol [1] for Version 1.1.

In the following, “Mesh Profile specification” refers only to Version 1.0 and Version 1.0.1.

2 Concepts

To understand the security risks and features described in this Mesh Security Overview, you will need to be familiar with the concepts described in this section. Additional details can be found in the Mesh Protocol specification [1].

2.1 Network

A mesh network is a collection of nodes that share a common address space, encryption keys, and other security-related parameters that are needed for communicating within the network.

A network can be partitioned into subnets by controlling the way encryption keys are made available to nodes. Only nodes that share the same keys are able to communicate with each other. An individual node can belong to one or more subnets.

2.2 Devices and nodes

An end product that supports Bluetooth Mesh Networking that has not been provisioned into a mesh network is known as a device. After a device has been provisioned into a mesh network, it is known as a node. A node is essentially any device that is addressable within the mesh network. A node consists of one or more individually addressable communication endpoints for mesh messages, known as elements. The number of elements on a node depends on what kind of device it is and what functionality it supports. A node that has just been provisioned has a range of consecutive addresses assigned to it, with one address per element. This same node also has a network key and a device key, so that it can be configured to perform tasks within the mesh network.

2.3 Low power nodes and friend nodes

A mesh node that constantly scans for messages can respond to incoming traffic with low latency. In contrast, a low power node (LPN) can remain in an inactive, power-saving, sleep state for extended periods of time. While the LPN is in a sleep state, a friend node will receive messages on behalf of the LPN. When the LPN wakes up, it will poll the friend for any messages it queued for delivery to the LPN.

While this arrangement helps the LPN to conserve power, it means that an LPN's response latency can be quite high. Sleep periods of several days are allowed by the Mesh Protocol specification [1]. This affects the potential execution duration for network management operations that involve LPNs. The length of an LPN's sleep period is a device vendor choice, and not configurable by the network administrator.

2.4 Provisioner

A provisioner adds devices to a mesh network. The provisioner functionality is often implemented on a smartphone or a similar portable device. A mesh network can have multiple physical provisioners. If multiple provisioners are used, these devices need to be able to coordinate with each other to decide how network resources are allocated, but the coordination does not need to happen in real time.

The device that performs provisioning does not have to be part of the mesh network at all times. If the provisioner is moved outside of the radio range of the mesh network, the network continues to function normally, with the same level of security, while the provisioner is absent.

2.5 Configuration manager

A newly provisioned node has only a minimum of information and cannot perform any useful tasks within the mesh network until it is configured. Usually, the provisioner is also a node in the mesh network, and it performs the configuration after it provisions the device. However, the configuration does not have to be performed by a provisioner. For example, a mesh network can have a node that performs continuous configuration and management of the network but does not perform provisioning. A node that can configure other nodes is known as a configuration manager.

2.6 Models

A model represents the functionality a node implements. The Mesh Model specification [2] defines standardized models for applications such as lighting and sensors. The Mesh Protocol specification [1] defines foundation models that control and configure mesh protocol related features on a node.

Model messages are exchanged between client models and server models, either using unicast (node-to-node) messaging or multicast messaging with the publish-subscribe paradigm.

A configuration manager sets up models on a node so that the node can communicate with other nodes in the network. This setup includes defining which addresses to use in sending and receiving messages and which application keys to use in message encryption.

Multiple instances of a model can exist on a node, but only a single instance can exist per element so that each instance of the model is reachable via a specific unicast address.

2.7 Keys

In Bluetooth Mesh Networking, messages are authenticated and encrypted using one or more of the following types of keys:

- Device key
- Network key
- Application key

Network keys are used to encrypt and authenticate the data transferred at the network layer of the mesh protocol stack. A configuration manager can partition a mesh network into multiple subnets by assigning different network keys to different sets of nodes, and only nodes that share the subnet key can communicate in the subnet.

Application keys are used to encrypt and authenticate model messages. A configuration manager manages the security of application communication between nodes by assigning different application keys to different sets of nodes. It also binds application keys to models, identifying which keys are to be used for specific functionalities.

A device key is a special key, formed during the provisioning process, that is used to encrypt and authenticate communication between a configuration manager and an individual node. Each device key is unique to a particular device.

Figure 2.1 shows how network and application keys can be used in a network.

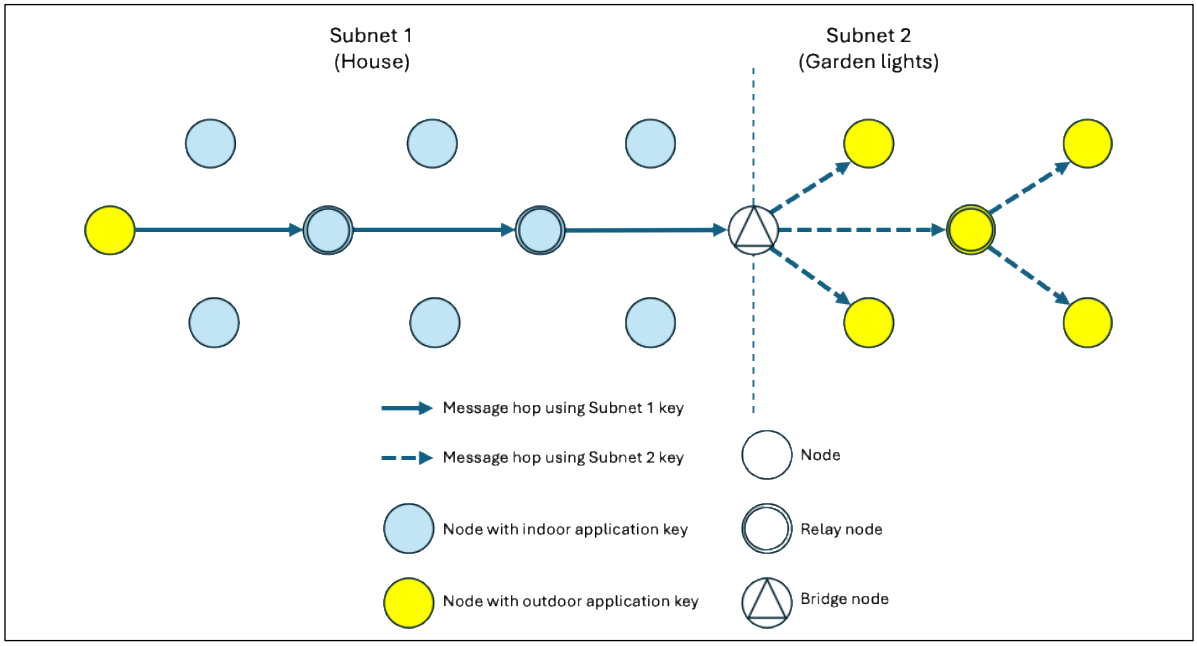

Figure 2.1: Example of network and application keys in a network

The application key for controlling outdoor lighting is deployed to the lights in the garden and to a light switch inside the house, while the application key for controlling indoor lighting is only deployed to the lights and switches that are inside the house.

Devices that know only the indoor lighting application key are not able to control the outdoor lights, and the devices that know only the outdoor lighting application key are not able to control the indoor lights either. However, indoor lighting nodes can still act as mesh infrastructure that relays messages for any application, because relaying is based on a shared network key and not on a shared application key. Relaying is needed whenever a node has a need to communicate beyond its immediate radio range.

In this example, the relayed messages that originate from the light switch traverse the indoor subnet hop by hop before reaching a node that acts as a subnet bridge. The subnet bridge is configured to forward messages from the indoor subnet to the outdoor subnet by re-encrypting messages using the network key of the destination subnet. In this manner, the light switch inside the house can reach the garden lights even though they are in a different subnet. The subnet bridge does not need to have access to any application key, unless the subnet bridge performs some application functionality.

3 Provisioning

Provisioning is the process of authenticating new devices, assigning them mesh addresses, and giving them the necessary security credentials to allow further configuration. A device participating in a provisioning process together with the provisioner is called a provisionee. Usually, a configuration manager (which often is the provisioner itself) will configure the provisionee immediately after it is provisioned, but it is possible to configure the provisionee later. The provisioner does not need to be present at all times during the operation of the mesh network; it only needs to be present when new devices are added to the network.

3.1 Out-of-band data

Provisioning can involve using provisionee-specific, out-of-band (OOB) data items; namely, out-of-band elliptic curve (EC) public keys (see Section 3.4) and out-of-band authentication data (see Section 3.5) to enhance provisioning security. Either data item can be used independently of each other.

Whether out-of-band data can be used during provisioning depends on the implementation of both the provisioner and the provisionee.

How using out-of-band data can improve the security of the provisioning process is summarized in the following tables. When out-of-band authentication data is used, providing sufficient entropy for the provisioning process can enhance security.

	In-band EC public key	**Out-of-band EC public key**
Authentication with No OOB	Vulnerable to active MITM	No known MITM vulnerabilities
Authentication with Static OOB	No known MITM vulnerabilities if sufficient entropy is provided	No known MITM vulnerabilities
Authentication with Output OOB	No known MITM vulnerabilities if sufficient entropy is provided	No known MITM vulnerabilities
Authentication with Input OOB	No known MITM vulnerabilities if sufficient entropy is provided	No known MITM vulnerabilities

Table 3.1: Man-in-the-middle attack protection for the Mesh Protocol specification, Version 1.1, provisioning process

	In-band EC public key	**Out-of-band EC public key**
Authentication with No OOB	Vulnerable to active MITM	No known MITM vulnerabilities
Authentication with Static OOB	Vulnerable to active MITM	No known MITM vulnerabilities
Authentication with Output OOB	Vulnerable to active MITM	No known MITM vulnerabilities
Authentication with Input OOB	Vulnerable to active MITM	No known MITM vulnerabilities

Table 3.2: Man-in-the-middle attack protection for the Mesh Profile specification provisioning process

3.2 Device discovery

Before a device can be provisioned, the device needs to be discovered by a provisioner. To enable discovery, a mesh-enabled device transmits messages that the provisioner can receive, which announce that the device is ready to be provisioned.

In advertisement-based provisioning, discovery messages are advertising packets known as unprovisioned device beacons. For Generic Attribute Profile- (GATT) based provisioning, these messages are GATT service advertisements for the Mesh Provisioning Service.

In both cases, a universally unique identifier (UUID) – a unique, 128-bit number – can be allocated to the device during manufacturing and created using any available technique. Another option is to have the device randomly generate a device UUID on the first power-up and skip this step in production to increase the ease of the manufacturing process.

The discovery message also includes information on whether any out-of-band provisioning information is available. Out-of-band data is not transmitted over Bluetooth®. For example, a device may be delivered with a near field communication (NFC) tag that contains additional out-of-band information, and the device advertises that it has an NFC tag along with its device UUID. A provisioner that receives such an advertising packet and supports reading NFC tags could then prompt the user to read this NFC tag to assist in secure provisioning of the device to the network. Similarly, a device may have a quick response (QR) code or similar printed information, and the advertising packets sent by the device can be used to prompt the user to find and scan the QR code.

As a result of successful device discovery, the provisioner will establish a provisioning session with the device, which takes on the role of the provisionee for the duration of the process.

3.3 Security algorithm negotiation

Each device that is added to a mesh network will support one or more security algorithms, and the provisioner has to determine which security algorithm to use to provision the provisionee after the provisionee's security-related capabilities have been discovered (see Sections 5.4.1.2 and 5.4.1.3 in the Mesh Protocol specification [1]).

The security algorithm negotiation is performed to address two different security concerns:

- Security algorithms that are considered secure today may be found to have flaws in the future. Therefore, it is essential that there is a way to negotiate and select the strongest available security algorithm among those supported.

  The ability to negotiate the algorithm could allow later versions of the Mesh Protocol specification [1] to remove support for compromised security algorithms and allow provisioners and provisionees to highlight the security implications of using such an algorithm.

  The original provisioning algorithm in the Mesh Profile specification [4] was shown to contain vulnerabilities related to the choice of authentication mechanism (see Section 8) that were corrected by adding a new provisioning algorithm in the Mesh Protocol specification [1], Version 1.1. For the time being, the original algorithm is retained in the specification for backward compatibility. However, for provisioners to support the update to the Mesh Protocol specification [1], Version 1.1, they will use the new algorithm when it is available on a provisionee.

- A network is only as strong as the weakest link. A single node that is added using a lower-security authentication scheme could provide a route for an attack on the whole network.

  Some deployments require using only a high-security authentication scheme when provisioning devices, such as a corporate office building. Other deployments can have reduced requirements and accept using a lower-security authentication scheme during provisioning, such as a vacation home in the wilderness. Provisioners can enforce a minimum level of device authentication scheme that is supported for a deployment. A provisioner application in a lower-security environment may accept an authentication scheme with lower security than a provisioner application used in a high-security environment.

3.4 Exchanging public keys

When the security algorithm negotiation is done, the provisioner and the provisionee will exchange elliptic curve (EC) public keys to create an ECDH shared secret for later use. The EC keys in Bluetooth mesh use the same P-256 curve as Bluetooth LE Secure Connections.

The public key the provisioner sends to the provisionee is an ephemeral key, generated on-demand, and is delivered to the provisionee over the provisioning session.

For the provisionee public key there are two alternatives:

- If the provisionee public key is available out-of-band, then the provisioner can retrieve the key using an out-of-band mechanism. One such mechanism is certificate-based provisioning (see Section 3.8).
- If the provisionee public key is not available out-of-band, then the provisionee will generate an ephemeral key and send it to the provisioner over the provisioning session.

Delivering the provisionee public key out-of-band helps to avert man-in-the-middle attacks during provisioning.

3.5 Authenticating devices

In Mesh Protocol specification [1], Version 1.1, up to 256 bits of authentication data can be used in provisioning a device. If used, this authentication data has to be exchanged out-of-band (that is, not over the provisioning protocol itself). For example, a number is displayed on the provisioner, and the user is directed to enter the same number in the provisionee.

There are three types of out-of-band authentication data exchange:

- Input: The provisioner generates a random value and outputs it, and the user inputs the same value in the provisionee.
- Output: The provisionee generates a random value and outputs it, and the user inputs the same value in the provisioner.
- Static out-of-band: The provisionee communicates the value, which may be a freshly generated random value or a value preprogrammed on the device using other technology, such as an NFC or QR code.

Input and output authentication data can be numeric (composed of decimal digits) or alphanumeric (composed of decimal digits and uppercase letters), depending on what input and output methods are available on the devices, while static authentication data is always binary (composed of octets).

The provisioner selects the type of authentication, the input or output authentication action (how the data is input on the provisionee by the user or output by the provisionee to the user), the size of data, and the provisioning algorithm to use based on its own capabilities and the discovered capabilities of the provisionee. For example, numeric input is possible but alphanumeric input is impossible if the provisionee has a numeric keypad.

Both the provisioner and the provisionee verify the other's knowledge of the correct authentication data by ensuring the other device can correctly calculate a hash value that requires authentication data.

Using authentication data helps to defend the provisioning process from man-in-the-middle attacks. The amount of entropy provided by the authentication data depends on the type of authentication, the chosen authentication action, and the size of data supported by the action.

For example, when input out-of-band authentication is used with alphanumeric data and the data size is limited to five alphanumeric characters, only approximately 25 bits of entropy is possible whereas using eight alphanumeric characters allows for roughly 41 bits of entropy. On the other hand, when static out-of-band authentication is used, the data will be binary, and up to 256 bits of entropy can be achieved.

3.6 Adding devices to a network

After a provisionee has been authenticated, both sides individually derive the device key for the new node (see Section 5.2). The device key is not transmitted over the air.

Additionally, it is necessary to share a minimum amount of provisioning data to allow the provisionee to be further configured after it becomes a node and a part of the mesh network. This data, secured using a session key, includes a network key and address assignment so that the node can communicate in the network. After the provisioner shares this data with the provisionee, the provisionee converts into a fully functional mesh node.

3.7 Remote provisioning

The Mesh Protocol specification [1], Version 1.1, introduced the concept of remote provisioning. In remote provisioning, the provisioner and provisionee are not located within direct radio range of each other but further apart, and the existing mesh network is used to relay provisioning messages between the two.

The provisioning messages are encrypted as application messages when they are relayed over the mesh network but will be sent and received as provisioning protocol messages over the final hop, which takes place over a provisioning session established between the provisionee and a node that acts as a remote provisioning server for the provisioner. The provisionee is not aware if it is communicating directly with a provisioner or not.

The following table summarizes out-of-band data compatibility with remote provisioning. Lack of direct physical access to both the provisioner and the provisionee limits the authentication method choice.

	No out-of-band EC public key	**Out-of-band EC public key**
Authentication with no OOB	Compatible with remote provisioning	Compatible with remote provisioning
Authentication with static OOB	Compatible with remote provisioning	Compatible with remote provisioning

	No out-of-band EC public key	Out-of-band EC public key
Authentication with output OOB	Incompatible with remote provisioning (requires physical access to a provisionee)	Incompatible with remote provisioning (requires physical access to a provisionee)
Authentication with input OOB	Incompatible with remote provisioning (requires physical access to a provisionee)	Incompatible with remote provisioning (requires physical access to a provisionee)

Table 3.3: Remote provisioning compatibility of out-of-band data options

3.8 Certificate-based provisioning

The Mesh Protocol specification [1], Version 1.1, defines a mechanism called certificate-based provisioning for acquiring provisionee public keys out of band. Certificate-based provisioning consists of two parts:

- Definition of how to represent provisionee public keys together with device identification data as standard X.509 certificates, which are cryptographically signed by certification authorities (CAs)
- Definition of how to acquire these device certificates in a standardized manner, either from servers on the Internet, or directly from the provisionees themselves

Certificate-based provisioning is an optional feature of the Mesh Protocol specification [1], Version 1.1. Devices indicate their support for certificate-based provisioning to provisioners in the advertisements they broadcast during device discovery.

3.8.1 Device certificates

Certificate-based provisioning leverages the well-known public key infrastructure (PKI) built around X.509 certificates (see ITU-T X.509 [6], RFC 5280 [7], RFC 6818 [8], RFC 6960 [10], RFC 8399 [11], and RFC 9654 [12]).

The provisionee public key and device UUID, along with other information that provides device identity, such as manufacturer information, are placed in a device certificate that is cryptographically signed by a CA, either directly with the CA root certificate or as a chain of signatures leading to the CA root certificate. In the latter case, there will be intermediate certificates between the device certificate and the CA root certificate.

When the device UUID is recorded within the certificate, the device UUID is assigned a known value at production. In this case, assigning a random value on the first power-up is not possible (see Section 3.2).

When a provisioner acquires a certificate associated with a provisionee, it can check the integrity of the device certificate data and any intermediate certificates by inspecting the cryptographic signature of a certificate against the public key recorded in the certificate above it in the chain until the CA root certificate is reached. This hinders man-in-the-middle attacks and other tampering with provisionee public key data.

The provisioner is also able to check that the certificate originates from a trusted CA by checking that a known, trusted CA root certificate is reached at the end of the signature chain. The provisioner can have a predefined set of trusted CA root certificates and simply ignore any devices with device certificates signed by CAs that are not trusted.

Furthermore, device certificates may be revoked for devices that suffer from security vulnerabilities, and the CA that signed the revoked certificates can make the revocation information available to provisioners by standard PKI means, such as providing certificate revocation list (CRL) distribution points or via the Online Certificate Status Protocol (OCSP) (see RFC 6960 [9] and RFC 9654 [12]). Revocation information can be used to block access to devices with known security issues. The exact mechanism for maintaining and distributing up-to-date revocation information is not part of the Mesh Protocol specification [1] and is the responsibility of the CA or vendor to define.

3.8.2 Acquiring certificates

A device vendor may place device certificates on a server which is accessible from the public Internet. In this case, the provisionee will provide the provisioner with the server URI over the provisioning session, on an NFC tag, or on a QR tag. The provisioner may then fetch the device certificate and possibly any out-of-band intermediate certificates from the Internet server pointed to by the URI using standard HTTP over Transport Layer Security (TLS).

Another option for the device vendor is to place the device certificate and possibly any intermediate certificates on the provisionee itself and provide the certificates to the provisioner over the provisioning session. While the provisionee public key is transmitted over Bluetooth, it is materially different from simple in-band delivery of a raw ephemeral public key. In this case, the provisionee's public key is contained in a X.509 certificate instead of being transmitted as raw data.

Checking the certificate signature chain will help the provisioner detect any tampering done by a man-in-the-middle attack over the provisioning session. Furthermore, the provisioner can make decisions based on the certificate contents, such as the identity of the entity to which the certificate was issued.

Trusted CA root certificates are not transmitted over either mechanism. The provisioner has to manage the set of trusted CA root certificates in a way that does not create provisioning vulnerabilities. For example, the operating system for the platform the provisioner is running on may provide a set of up-to-date CA root certificates.

4 Network layer security

Whenever a mesh node sends a message into or receives a message from a mesh network, it uses keys derived from a network key to authenticate and encrypt the network data. Encryption and authentication are performed using the same AES-128-CCM security algorithm as in the Bluetooth® Core Specification [3].

The AES-128-CCM security algorithm uses a 128-bit encryption key together with a 104-bit nonce value to encrypt and authenticate a message. The network layer has security mechanisms for generating unique nonces for messages, encrypting and authenticating as much message information as possible, and obfuscating the identifying information in messages for additional privacy.

The network can be partitioned into multiple domains, called subnets, by assigning different network keys to different sets of nodes.

4.1 Derived keys

Because a network key is critical to the operation of a mesh network, the value of the network key is not used directly. Instead, values derived from the network key with key derivation functions (KDFs) are used. Each derived value is used for a specific purpose. The network key KDFs are based on the AES-128 cipher-based message authentication code (CMAC) hash algorithm.

A KDF is used to defend the network key if a derived key is compromised. For example, if an encryption key (derived from the network key) was compromised as a result of an attacker extracting the encryption key from a device, then the compromise would not have identified the network key and other values derived from that network key. This is important because some derived values are public and are transmitted unencrypted on a mesh network.

All derived values are refreshed on a node when the underlying network key is refreshed.

4.2 Unique per-packet nonce

The AES-CCM algorithm uses a key, a plaintext message, and a nonce to generate an encrypted message and a message authentication code. In RFC 3610 [13], the AES-CCM algorithm requires a unique nonce for a given key. Additionally, each encrypted message must use a distinct nonce as required by RFC 3610.

The uniqueness of the network layer nonce for a given message results from a combination of the unicast address of the message source, the sequence number of the message, and the network initialization vector (IV) Index.

The network's configuration manager assigns the unicast addresses for a node's elements when it provisions the node. This allows the network's configuration managers to know which unicast addresses are not yet in use and then assign unique addresses to the provisioned node.

Every node is responsible for keeping an up-to-date sequence number counter for each of its elements. The sequence number counter on an element has to be incremented by 1 for each new message that the element transmits, so that each message of an element has a unique sequence number.

The sequence number is a 24-bit value, and therefore an element can transmit more than 16 million messages before the sequence number exhausts the allocated space. For example, if an element sends a message once every 5 seconds, then these sequence numbers would run out after 2.66 years.

However, if an element sends a message every 100 ms, a high frequency for a mesh node, then an element would exhaust the available sequence numbers after only 19 days.

The sequence number value is limited to 24 bits because it is included in the network headers of the message and the space available in Bluetooth LE nonconnectable advertisements is constrained. To enable nodes to send messages for a longer period than a few days, a network IV Index is used in conjunction with the sequence number.

The IV Index is a 32-bit number that is shared by all nodes within a mesh network. This number can be incremented through an IV Index Update procedure (see Section 6.2), which is triggered by any node that is running out of sequence numbers. The IV Index Update procedure completion allows nodes to reset their element sequence number counters to zero and therefore avoid running out of sequence numbers. The updated IV Index can take up to 96 hours to propagate throughout the network, because low power nodes infrequently receive mesh messages. The node that sends messages with a high frequency could continue to operate for a few million years by requesting an IV Index Update at least 96 hours before it runs out of sequence numbers every 15 days.

4.3 Encryption and authentication of network data

The network data is encrypted using the AES-128-CCM algorithm. The destination address of the message and the message payload are encrypted using the encryption key, which is derived from the network key and the per-packet nonce. The network data is also authenticated using the same encryption key and the same per-packet nonce. This allows the encryption and authentication of the message to be performed in a single operation using a single AES-CCM algorithm. The authentication of the message is achieved using a 32-bit or 64-bit message authentication check value at the end of the network message, depending on whether the message payload contains encrypted application data or a transport control message. The values used to construct the nonce are also indirectly authenticated by the same operation. While application data is only authenticated by a 32-bit value at the network layer, it will be further authenticated by another 32-bit or 64-bit message authentication check value at the end of the access message (see Section 5.4).

In many non-Bluetooth Mesh Networking specifications, the destination address is not included in the information that is encrypted because this increases the costs of routing messages within a mesh network. The Mesh Protocol specification [1] requires that whenever a message is retransmitted the network message be fully decrypted and authenticated, that the time to live (TTL) be decremented, and that the message then be fully re-encrypted and reauthenticated. Encrypting the destination address limits the ability of an eavesdropper to do traffic analysis based on the destination address.

4.4 Obfuscation of network information

In addition to the encryption and authentication of the network message, the parts of the packet that are used to generate the nonce are obfuscated. The purpose of obfuscation of the nonce data is to keep a passive observer from performing traffic analysis of mesh messages by observing the message headers.

For example, if obfuscation is not performed and a passive observer listens to a message sent from node 0x0001 and another message sent a very short time later from node 0x0002, then it is possible to surmise that the first message was sent to node 0x0002. If the destination of the message is known, then that information can be used to analyze traffic patterns in the network.

The obfuscation is performed using an AES encryption operation that takes a few octets from the start of the data packet as input along with a privacy key that is derived from the network key and uses the output of the encryption operation to perform an exclusive-OR with the header information.

This obfuscation does not improve confidentiality because the obfuscated data is not securely encrypted or authenticated. However, it provides the benefit of making traffic analysis more difficult.

4.5 Optimizations to reduce processing

A Bluetooth mesh network can be deployed near other Bluetooth mesh networks. For example, the mesh networks of neighboring homes will be observable from nodes in a home network. It would be inefficient for a node to attempt decrypting every network message that it receives to check that the message is encrypted and authenticated using one of the network keys known to the node. To reduce this overhead, every network message includes a network identification (NID) value that is sent in plaintext.

When a node receives a message, it will first check the NID value against the list of known NIDs for its known network keys. If there are foreign mesh networks in the area, then there is a high likelihood that the NIDs on those messages are different and the messages can be immediately discarded.

4.6 Subnets

Any node that is part of a mesh network, supports the Relay feature, and has a network key can relay messages for any other node that is part of the same mesh network.

Within a mesh network, one or more subnets can be created to isolate a set of nodes for security or administrative purposes. The extent of a subnet is determined by the number of nodes that know its network key. Subnets may be created on demand by creating a new key and distributing it to selected nodes. Within a single mesh network, as many as 4,096 subnets can be defined. One of the subnets is the primary subnet, which is used for network management operations. The network key of the primary subnet is called the primary network key.

Subnets can be used to isolate nodes that are not fully trusted from other nodes on a mesh network. A common use case is to provide a guest temporary access to a limited set of devices. To facilitate this, a new “guest key” is generated and distributed to the guest and the nodes that the guest will be given access to. The guest access is revoked by discarding the guest key and removing it from the nodes.

Subnets are also useful for isolating nodes that may be exposed to physical hijacking and hardware key extraction techniques from other nodes in the network. For example, garden lights might be added to a subnet of a home “lights” network and not be given the primary network key. If a garden light is hijacked and the network key of the subnet is extracted from its memory, then that key will not give the attacker access to lights in the primary subnet of the home network.

A node can have and use multiple network keys at the same time, allowing it to operate in multiple subnets of a mesh network, but communicating in a subnet over relay nodes implies that relays also have the network key of the subnet. Subnet bridges are special nodes that can re-encrypt data sent using one subnet’s encryption key with another subnet’s encryption key, based on configuration set up by the network’s configuration manager. Therefore, a subnet bridge allows messages from one subnet’s nodes to reach nodes in another subnet in a controlled manner, without having to distribute multiple network keys to source and destination nodes and all the relays in between.

5 Application security

An application key is used to provide an additional layer of security for application data within the context of an application security domain. Nodes that have the application key can understand each other's messages, while nodes that do not have the application key do not understand.

A device key provides security at the same protocol layer as an application key within the context of the communication between a configuration manager and a particular node.

5.1 Application key

Mesh application communication is based on the concept of models that represent some functionality a node implements. Models communicate with each other across a mesh network by sending and receiving application messages that interact with the model functionality. For example, a wall dimmer's ability to adjust light level in a room is represented by the Light Lightness Client model, and the requests it sends to set the light intensity level are application messages that are received by the Light Lightness Server model in light sources that have the ability to dim.

An application key is a key that is given to nodes within a mesh network for a given application security domain. Having an application key determines the set of nodes that can send and receive model messages for a particular application such as lighting. Application keys can be used in various ways to set up access control. Different application keys can be used to separate application domains or to implement fine-grained, role-based access control among users and administrators.

For example, a doorbell and its push button would have the doorbell application key and its corresponding doorbell network key but not the door lock application key and its corresponding door lock network key. If an attacker extracted the doorbell application key and the underlying network key from the doorbell push button, this would enable a "denial of silence attack". The attacker could ring the bell at any time until the compromised keys are replaced by new ones in the network or the doorbell node is reconfigured to use other keys. It would not, however, give access to the lock functionality for which the door lock application key is used. Even if the underlying network key were the same for both applications, the attacker would not have access to the lock functionality, so they would be able to execute a denial-of-service attack on the lock but not open the door.

A configuration manager binds specific application keys to specific models on each node. Models then use the application keys bound to them in sending and receiving application messages for the models' specific functionalities.

A node that implements multiple functionalities can include models for more than one application and can have more than one application key. For example, a room control panel that could implement both dimming and air conditioning functionalities can have application keys for both. It would still use a specific application key for a specific purpose, as defined by the key-to-model bindings.

5.2 Device key

The device key is a key that is used to authenticate and encrypt communications between a configuration manager and an individual node. Each node in a mesh network has a device key that is unique to that node. During provisioning, the device key is calculated using the shared secret that is derived from the ECDH key agreement between the provisioner and the device that is being provisioned. The device key is not transmitted over the air during provisioning or by any mesh messaging afterwards. Only the provisioner will have the device key unless the provisioner shares that key with configuration manager nodes, specifically for the purposes of configuration.

The device key is used to authenticate and encrypt some of the foundation models' communications between a configuration manager and a single node. A node's device key is not shared with other nodes in the network. Therefore, no other node can interpret communication encrypted with a particular device key. The foundation model messages encrypted with the device key are used to read information about the node, configure how the node communicates with other nodes, and distribute security credentials. For this reason, the device key is one of the most security-critical pieces of information any single node has.

The device key is implicitly bound to many of the foundation models defined in the Mesh Protocol specification [1]. It cannot be explicitly bound to any model.

5.3 Unique per-packet nonce

As with the network layer, the application layer uses a unique nonce for each message sent by a node. The uniqueness of the application layer nonce for a given message results from a combination of the unicast address of the message source, the sequence number of the message, and the network IV index, in the same manner as for the network layer nonce. An octet specifying nonce type at the start of each nonce keeps the values of the network layer nonce and the application layer nonce different from each other for the same message.

5.4 Encrypting and authenticating application data

As with the network layer, the application data is encrypted and authenticated using AES-128-CCM. The application data can be authenticated with either a 32-bit or 64-bit message integrity check value. Unsegmented messages can only use a 32-bit value because of the limited amount of data that can be sent in a single non-connectable advertisement. Segmented messages can use either a 32-bit value or a 64-bit value, depending on whether stronger security is considered necessary.

5.5 Optimizations to reduce processing

The application key is used as-is to encrypt, decrypt, and authenticate application data. The only value derived from the application key is a plaintext application key identifier (AID), which is a hint for a receiving node that functions similarly to the NID's function for a network key. AID value is not derived for device keys. A configuration manager can instead use the message source address to identify which device key to use to decrypt a received message.

6 Network management

The Mesh Protocol specification [1] defines procedures for network management. This section provides an overview of those procedures.

6.1 Mesh beacons

Many network management procedures are signaled using mesh beacons, which are messages that are broadcast periodically by the nodes in the network. These include secure network beacons and mesh private beacons; the difference between the two is explained in Section 6.4.

The mesh beacons contain three values:

- The current IV Index
- A flag to indicate whether an IV Update procedure is in progress
- A flag to indicate whether a Key Refresh procedure is in progress for the key used to transmit the beacon

6.2 Signaling network IV Index updates

As described in Section 4.2, the IV Index is a 32-bit value that is used in message encryption. All nodes in the network share the same IV Index. From time to time, nodes that are close to exhausting the sequence numbers available to them are responsible for triggering the IV Index Update procedure to have a fresh set of sequence numbers. An IV Index Update procedure started by one node propagates in the network via mesh beacons.

Mesh beacons include the current IV Index of the network, as known by the node that sent the beacon, and a flag indicating the current state of the IV Update procedure. When a node receives a secure network beacon with an updated IV Index, it will update its own state and then start to broadcast this updated information in the beacons that it transmits. Therefore, these beacons allow the network security state to be shared between nodes in the mesh network.

All nodes within a mesh network are expected to receive messages with the current IV Index and with the IV Index that was used most recently. This is necessary to allow nodes that have already processed an IV Index update to receive messages sent by nodes that have not yet processed that IV Index update. An IV Index Update procedure may take a long period of time. For example, a Low Power node may only wake up once every 96 hours, so it may be sending messages using the older IV Index for up to four days after most nodes in the network move to a new IV Index. Only nodes that have the primary network key (key index 0x000) are allowed to start an IV Index Update procedure. Procedure initiation attempts by nodes that are part of another subnet are ignored.

6.3 Signaling network key updates

A mesh network performs network key updates, and associated application key updates, using the Key Refresh procedure. Performing such updates periodically and whenever a device could have been compromised can improve security; for example, when a device is replaced and then placed into an insecure area such as a recycling center.

The Key Refresh procedure occurs in three phases:

- During the first phase, the new keys are distributed to all nodes that have not been excluded from the Key Refresh procedure.
- During the second phase, nodes are notified to transmit using these new keys.
- During the last phase, nodes are notified to revoke the old keys.

A configuration manager uses foundation model messages to distribute the new keys; when a node receives a message containing new key material, it will automatically move from normal operation to the first phase of the Key Refresh procedure. At this phase, the node will still transmit using the old keys but will receive with both the old and the new keys.

Mesh beacons are used to signal the move to the second phase after new key distribution is complete. The beacons are sent by the configuration manager using the credentials of the new keys with a flag set to indicate that receiving with the old keys is still acceptable but transmission has to be done with the new keys. The nodes that know the new keys receive these mesh beacons and then retransmit the beacons using the new keys, propagating the phase information throughout the network.

The nodes that have been excluded from the Key Refresh procedure can only observe that a new beacon has appeared. Those nodes cannot directly identify the network from the new mesh beacon, and they will not know whether the new network is the same as the previous network or if it is a new, separate network created in the same area. The nodes might try to correlate the IV Index value carried in the beacons with the IV Index value they know, but beacon data obfuscation in mesh private beacons (see Section 6.4) will hinder this as well.

Because all the information in mesh messages is encrypted or obfuscated, the excluded node cannot process messages that are sent using the new keys that they do not have.

After the configuration manager determines all nodes have been moved into the second phase, for example, by explicitly querying each node involved in the Key Refresh procedure for the phase the node is currently in, the third phase is initiated. The third phase again uses the mesh beacons sent in the context of the new keys, but this time with a flag indicating that the old keys are revoked. All excluded nodes are gone from the network and all mesh beacons that are authenticated by the old network key are now ignored.

Executing a Key Refresh procedure can take a long time, especially if the network is large or if Low Power nodes with high-response latency are involved.

6.4 Beacon privacy

Many mesh networks are static in nature, such as building automation networks. However, mesh networks that are roaming are also possible; for example, an in-vehicle network. The fact that a secure network beacon consists of fields that only change as a result of a network management operation poses a problem. The movement of such roaming networks may be passively tracked if the nodes send unchanging secure network beacons for longer periods of time. This enables listeners to identify devices simply by comparing received data to previously captured data, without having to attempt decrypting the data. Furthermore, secure network beacons contain beacon data in plaintext, making it easy for a passive attacker to monitor IV index update and key refresh processes in the network.

To help keep the data of users and owners of such roaming mesh networks private, the Mesh Protocol specification [1], Version 1.1, adds the concept of mesh private beacons. In private beacons, the data in the beacon is obfuscated with the help of random data that is periodically changed.

Similar to network header obfuscation, beacon data obfuscation is performed using an AES encryption operation that encrypts a random number using a beacon privacy key. The beacon privacy key is derived from the network key and uses the output of the encryption operation to perform an exclusive-OR with the beacon data. The random number used in obfuscation is transmitted in the beacon, so that a receiver is able to deobfuscate the beacon data using the same beacon privacy key and the same random number. The node that sends beacons generates a new random number every time beacon data changes, or after a configurable time interval if the beacon data remains constant.

Furthermore, using a private Bluetooth® address when transmitting private beacons is required to help hinder tracking on the roaming mesh network by passive eavesdropping of mesh beacons. The node that sends beacons updates its private Bluetooth address each time the random number that obfuscates beacon data changes.

Mesh private beacons can be beneficial in static networks as well, given that beacon data obfuscation makes it more difficult for attackers to extract beacon data and track network management operations.

Other than the privacy-related differences in the beacon data, mesh private beacons function the same as secure network beacons do.

6.5 Complete rekeying of devices

A network owner may take over a network from a previous owner (e.g., as part of a building lease or purchase). In such a case, refreshing all network and application keys in the network is not enough. Anyone with the knowledge of the old keys as well as the device keys for at least some nodes in the network may listen to the configuration messages deploying new keys to nodes and gain knowledge of them.

The Mesh Protocol specification [1] solves this conundrum by introducing the Device Key Refresh procedure as part of the Remote Provisioning feature. The Device Key Refresh procedure allows a regeneration of a node’s device key without having to factory reset the node back to an unprovisioned device state. The node keeps all its configuration, including already deployed keys, and changes only its device key. As in provisioning an unprovisioned device, the device key is not transmitted over the air but is calculated from data exchanged over the provisioning protocol. The only difference is in the Device Key Refresh procedure the provisioning protocol is run using the already existing mesh network as remote provisioning, not over a provisioning bearer to a nearby unprovisioned device.

After the device key on a node is refreshed, configuration requests to deploy new network and application keys to the node are encrypted with a key not known to the previous owner of the network.

7 Security threats

This section discusses the security threats that are taken into consideration and mitigations of the threats implemented in the security architecture of Bluetooth® mesh networks.

7.1 Replay attack

As with most networking technologies, the protection against replay attacks on a mesh network is of utmost importance. In a replay attack, the attacker records a message and then retransmits that same message to gain unauthorized access. For example, an attacker might attempt to retransmit an Unlock House message.

To help protect against replay attacks, each new network protocol data unit (PDU) sent over a mesh network is given a new sequence number. The sequence numbers are monotonically incremented for each new network PDU. Elements within the same node may or may not share sequence number space. When a node receives a message from a peer node, it checks to see if the sequence number in that message is greater than the previous sequence number it received from that peer. If the sequence number in the Access or Transport Control message is not greater than the previous sequence number received from that peer, then the message is not processed.

In a segmented message, a longer application message is split into multiple smaller segments that are sent to the peer node in individual network layer messages. The original message is then reassembled. The sequence number is incremented for each segment that is sent. When the last segment needed to reassemble the message is received, its sequence number is checked to verify that the reassembled Access or Transport Control message can be processed.

The sequence number is a 24-bit value, and therefore the available sequence numbers can be exhausted. However, the sequence number is logically extended by a 32-bit IV Index that can be updated every few days. When the IV Index is updated, the sequence numbers on all nodes in the mesh network are reset to zero. This has the advantage that a node only needs to remember peer nodes that have sent messages to it recently, and the peer nodes that no longer send messages can be forgotten after the IV Index is updated.

To track the sequence numbers of the mesh messages that have been received from various peers, a node maintains an internal list (called a replay protection list) of the sequence numbers from the latest messages processed for each peer source address. Whenever a node processes a message from another node, it will update the corresponding replay protection list entry so that the list is continuously kept up to date.

The number of peer source addresses that can be tracked for replay protection depends on the size of this list. When this list is full, a node cannot receive messages from new source addresses that are not already tracked in this list unless entries in this list are freed up because of completing the IV Index update.

The replay protection list is saved in a persistent memory to provide replay protection across power cycles.

7.2 Bit-flipping attack

A common attack used in other networking technologies is the bit-flipping attack. In a bit-flipping attack, an attacker takes a message that it has passively received and can cause a valid node in the network to do something useful for the attacker by flipping an appropriate pattern of bits. This type of attack is observed most often when the authentication of the message is weak.

The Mesh Protocol specification [1] requires a minimum of 64 bits of message integrity check values on every message sent, with a minimum of 32 bits at the network layer and a minimum of 32 bits at the application layer. The ability to perform a bit-flipping attack is significantly reduced because of the combined strength of these check values.

7.3 Eavesdropping attack

During an eavesdropping attack, a passive observer attempts to do traffic analysis based on information in the packet. Because mesh messages are broadcast using advertising packets for Bluetooth LE, any device that can receive those advertising packets can also receive packets from mesh nodes. To help protect the data within mesh messages from passive eavesdroppers, all mesh messages are encrypted twice using the AES-CCM algorithm. The first encryption takes place at the application layer, where the application message is encrypted using the application key. The second encryption takes place at the network layer, where the application message, along with the destination address, is encrypted using the network key. Also, the source address of the message is obfuscated to help hide information identifying the sender of the message. For more information, see Sections 4.3 and 4.4.

7.4 Capture now and decrypt later

In another variant of eavesdropping, an attacker records traffic they cannot decrypt right away but expect to be able to decrypt later; for example, as a result of having ample time to brute-force a key offline, finding a new vulnerability in a security algorithm, or acquiring cryptographic keys by some other attack (such as a trash can attack, see Section 7.10). If an attacker is able to decrypt the captured traffic, then its information is recognizable and an attacker can analyze it further.

Regular key refresh will limit the extent of such an attack by restricting the time frame when an acquired key is in use. Partitioning the network into subnets will limit the effect of acquiring a single network key. Using multiple application keys for specific purposes will limit the effect of acquiring a single application key. The choices on how to use each key type can be made independently from each other.

7.5 Man-in-the-middle attacks

A mesh network is at its most vulnerable while provisioning a new device onto the network. During provisioning, it is possible for a “man-in-the-middle” to pretend to be the device that is being added to the mesh network and generate two shared secrets, one with the provisioner and one with the new device that is under attack.

To help protect against a man-in-the-middle attack, two mechanisms are provided: out-of-band public keys and out-of-band authentication data.

When the provisionee’s public key is acquired by the provisioner out-of-band, it becomes much more difficult for an attacker to feed their own public key to the provisioner. If the provisionee’s public key is enveloped in a X.509 certificate, the attacker would need to fabricate a valid certificate issuer signature; and if the provisionee’s public key is transmitted over a communication channel other than Bluetooth then the attacker would need to be able to be a man-in-the-middle on that channel as well.

Likewise, when authentication data is acquired by the provisioner out-of-band, it again becomes much more difficult for an attacker to feed their own authentication data to the provisioner. The attacker could try to mount a brute-force attack and this can be made infeasible if the provisionee's authentication data contains enough entropy. If possible, refresh authentication data on every provisioning attempt.

When a device is provisioned using remote provisioning, input and output authentication might not be possible. In that case, static out-of-band authentication data can still be used. The Mesh Protocol specification [1], Version 1.1, provisioning algorithm needs be selected if out-of-band authentication data is used, as the previous provisioning algorithm contains known vulnerabilities related to authentication data processing.

In a variant of a man-in-the-middle attack, called a downgrade attack, the attacker attempts to disable security enhancements, such as the new provisioning algorithm available in the Mesh Protocol specification [1], Version 1.1. If the attacker succeeds in forcing either the provisioner or the provisionee to use less secure protocol options, then the attacker can attempt to use the known vulnerabilities. Downgrade attacks can be made more difficult with a provisioner policy that refuses to provision devices requesting authentication algorithms and methods that have known vulnerabilities.

7.6 Brute-force attacks on keys

The Mesh Protocol specification [1] helps protect against brute-force attacks on keys by using two mechanisms:

- The network key and the application key are used in series when encrypting and authenticating a packet, and therefore a successful attack on the network key is needed before a successful attack on the application key can begin.
- The time the devices in the network are vulnerable to a brute-force attack on a particular key is limited by the duration of time that key is in use.

The risk of a successful brute-force attack can be mitigated by performing periodic key refreshes. The way to mitigate the risk of a successful brute-force attack is to refresh both the network keys and the application keys with sufficient frequency that an attacker does not have the time to successfully test all variations on the key values. Note that device keys cannot be refreshed using the key refresh procedure that applies only to network and application keys; the Device Key Refresh procedure must be used instead.

A network key refresh assigns a new 128-bit random value to a network key, and therefore any partially successful attack on the previous network key or its derivations is irrelevant when considering how to attack the new network key as long as the key distribution is not compromised.

7.7 Network key compromise

In general, if a network key is compromised, a malicious node will be able to send, receive, and relay network layer messages without detection. In addition, a malicious node will be able to send transport control messages, attempt to initiate IV Index updates more frequently, or flood the mesh network with invalid application traffic. The malicious node could also send messages that appear to come from another node within the mesh network. Some of the attacks can be easy to spot (e.g., by lack of connectivity to a node or to a part of the network), while other attacks may be harder to discover.

However, the compromise of a network key does not break the encryption of application messages. An attacker cannot fabricate fake application model or foundation model messages without application or device keys.

7.7.1 Replay protection list exhaustion

An attacker that has access to a network key has the ability to keep legitimate messages from being processed by filling up the replay protection list with spoofed entries. While an attacker cannot spoof legitimate application messages without an application key, an attacker can spoof transport control messages and segment acknowledgment messages seemingly from every source address used in the network with the maximum sequence number of 0xFFFFFF. This causes nodes to ignore all messages in the network until an IV Index update is done.

Such an attack could be detected by noticing a connectivity degradation between the affected nodes and the rest of the network, such as unresponsiveness. Nodes under attack can recover by processing an IV index update after the attack is stopped (see Section 6.2).

7.7.2 Flooding the network

A node that has compromised the network key but not an application key could flood the network with traffic consisting of transport control messages or junk application messages that do not decrypt. This forces the receiving nodes to consume resources when processing the messages, so the traffic could be used as a denial-of-service attack that could target a particular set of nodes in the network. For example, the traffic could target a specific physical area or specific kinds of nodes such as Friend nodes.

Such an attack could be detected by degrading connectivity between the nodes and the rest of the network. Nodes under attack can recover after the traffic subsides.

7.7.3 Path manipulation

The directed forwarding feature provides a mesh network with the means to control message flows in the network with more refinement than a simple TTL counter. To achieve this, nodes that have the directed forwarding feature enabled generate and process transport control messages related to message path discovery.

An attacker that is in possession of a network key may attempt to disrupt path discovery by generating fake responses, altering existing paths to go through them to facilitate further attacks, or flooding the forwarding tables on the nodes with useless information.

An attack to disrupt path discovery could be detected by noticing problems in directed forwarding connectivity between different parts of the network. Nodes under attack would need to rebuild their forwarding tables to recover, but flooding traffic would work as usual so no node would be completely unreachable.

7.7.4 Fake friendships

Low Power nodes rely on their friendship with Friend nodes to receive messages while they are in energy-saving mode. A compromised network key could allow an attacker to either pretend to be a Low Power node, using up resources on Friend nodes that other Low Power nodes could be using, or pretend to be a Friend node and disrupt the message flow to a Low Power node. An attacker could also manipulate an existing friendship and control which messages a Friend node queues on behalf of the Low Power node or flush the queue before the Low Power node has retrieved the messages. It could also terminate an existing friendship.

Such an attack can be hard to detect since Low Power nodes can sleep for extended periods of time and typically receive few messages from the network. Low Power nodes under attack would need to reform friendships with well-behaving Friend nodes to recover.

7.7.5 Disruptions using heartbeat messages

Heartbeat messages are transport control messages used to help identify operational nodes in the mesh network. They can also be used to identify the topology of the mesh network.

When and where to send heartbeats is controlled by the configuration manager, but a compromised network key could allow an attacker to detect ongoing heartbeat signaling and send out fake heartbeat messages to provide invalid information to other nodes in the mesh network. Other nodes on the mesh network can be easily identified as under such an attack because the total number of heartbeat messages becomes higher than expected. Nodes under attack have the option to retry their heartbeat signaling after the malicious node has been excluded from the network to recover valid information.

7.7.6 Isolating a compromised node

It is possible to help protect against a node that is considered to be compromised by excluding the rogue device and performing a key refresh of all other nodes in the network. During a key refresh, the network keys that are used to encrypt and obfuscate the network layer traffic are updated using the device key, and the previous network keys are revoked on all nodes that have not been excluded from the Key Refresh procedure. After the old network keys are revoked, the attacking node becomes a node in a mesh network of one device, which isolates the problem. For this reason, it is highly recommended to perform a key refresh of trusted mesh nodes with reasonable frequency, such as at least once a week.

7.8 Device key compromise

New network and application keys are distributed to nodes by using the device keys. Therefore, even if network keys and application keys are compromised, the distribution of new network and application keys is not compromised as long as device keys themselves are not compromised.

Since each device key is specific to one device, partitioning the network into subnets can limit the number of nodes affected by a device key compromise. A successful attack on a given device key would only provide access to the subnet keys distributed to that particular node after the device key compromise.

Likewise, partitioning the applications into separate security domains can limit the number of applications affected by a device key compromise. A successful attack on a given device key would only provide access to the application keys distributed to that particular node after the device key compromise.

If it is known that a node’s device key is compromised, then that node can be removed from the mesh network by excluding it before the next key refresh.

7.9 Attacks on physically insecure devices

Many devices on a mesh network may be placed in locations that are physically insecure. Although devices within a building can be guarded and physically secured against malicious attack, devices on the outside of a building or around the edge of the property are vulnerable to physical attack.

To help defend against such attacks, subnets can be created to isolate devices that are physically insecure from other devices on the main network. Nodes that are physically exposed to attacks are not given the main network key but are organized as a perimeter subnet. Hijacking the subnet key does not give the attacker access to the main network.

For further protection against such attacks, separation is maintained between network layer security and application layer security. To be able to relay messages to and from a physically insecure node, a node in

the mesh network only needs to know the network key that is used by a physically insecure node. If the network key is compromised, then the impact is limited because no application keys are known.

Application domains also can be used to mitigate the effects of attacks on physically insecure devices. Nodes in physically insecure areas can have a different set of application keys from nodes located in physically secure areas.

If that separation is maintained, even if the physically insecure node is compromised and an attacker can send a message using the network key and application key extracted from the device, then the attacker will not be able to access functionality protected by different application keys.

7.10 Trash can attack

When a mesh node is replaced, for whatever reason, the device may be discarded into the trash and then thrown out of the building along with all other trash. This device could still have the security credentials for the network inside, which would make it vulnerable to what is known as a trash can attack. Although it is possible to send a reset message to a node to wipe out all this security information, the request to reset cannot protect against devices that have recently stopped working with the security credentials still stored in nonvolatile memory within the device. Depending on the storage technology, such as flash storage, removing the data completely programmatically could be difficult.

To help defend against trash can attacks, exclude the discarded devices from key refresh, and perform a key refresh on all remaining devices within the network. Therefore, before a discarded device leaves the secure building environment, it only has older keys that have already been revoked. The old keys are no longer in use, and therefore messages sent using the old keys will not be processed by nodes in the network.

Disposal procedures are often already in place in organizations for devices containing sensitive data such as hard drives. Amending these procedures to take Internet of things (IoT) device persistent storage characteristics into consideration could help properly prevent a trash can attack.

An implementation may make a trash can attack more difficult by storing the mesh-related data in an encrypted form or within a hardened subsystem such as a secure element.

7.11 Guest access attacks

Many homes and businesses invite guests into their buildings, and these guests are often given security information that allows them to get onto a building's network. As a security measure, the network administrator can set the guest credentials to expire automatically after a reasonable time frame and limit guest access so guests are not provided with any information that would allow them to reconfigure the network.

To enable guest access, separate network keys and application keys can be provided to the guests. The guest keys that are not updated using a Key Refresh procedure will automatically be revoked the next time that the network and application keys change. Importantly, no guest has access to the device keys needed to initiate a Key Refresh procedure. Furthermore, a guest cannot initiate an update of the IV Index because guests do not receive access to the primary network key. For more information about guest access, refer to the Mesh Protocol specification [1].

Using separate application keys for guests is also important, as this can limit not only the nodes that the guests can interact with but also the functionality that is exposed to the guest. The functionality of nodes can be bound to specific application keys. For example, a guest might be able to read the current temperature of a hotel room but not reprogram the temperature sensor to publish the temperature to a different address.

7.12 Privacy attacks

Knowing who or what is in a location can be very valuable information. Such information could enable illegal activities such as stalking an individual or monitoring which companies are flying to which locations in the hope of determining the next merger or acquisition. The Mesh Protocol specification [1] helps protect against such attacks by obfuscating messages sent by mesh nodes.

While installations such as building lighting networks are static by nature, there are also vehicle-based, wearable, and other mobile-use networks that exist. An attacker tracking the movements of these kinds of networks is a privacy issue.

To help protect against invasions of privacy, the Mesh Protocol specification [1] obfuscates all the identifying information on a mesh payload apart from a single network identity octet. This obfuscation is set up to make it more difficult to determine the source address, sequence number, and TTL value through passive observation. The network identity octet is clear text. However, given that there are only 128 possible values out of the huge number of possible networks, inferring any network identity octet would be difficult in areas where many mobile mesh networks pass through. As the NID values are derived from network keys, periodically refreshing network keys will generate new NID values which further limits tracking possibilities.

Furthermore, the private beacon functionality obfuscates the broadcast beacons used to advertise network presence, network security state, and encryption key updates. The obfuscation makes use of random numbers to provide entropy to data which might otherwise remain static over long periods.

A mesh message is not reliant on the Bluetooth address that is included in advertising packets. Such addresses can quickly help to identify messages sent by the same device or, in the worst case, to identify the exact device. Because the mesh advertisement bearer ignores the Bluetooth address, a different random address can be used as a Bluetooth address periodically, or a different Bluetooth address can even be used for every message that is sent without any impact on network functionality.

7.13 Counterfeit or altered devices

An attacker may try to add a counterfeit device, or a modified device with legitimate origin, to a network. Such a device can look like and behave as an unmodified device from a legitimate vendor on the surface, while containing mechanisms an attacker can make use of, such as making cryptographic keys it receives available over a non-Bluetooth communication channel.

A variant of this attack involves a legitimate device with a modified NFC tag or QR code that leads the user into provisioning a hidden rogue device onto the network using the rogue device's authentication data. The hidden rogue device will act as a proxy to the legitimate device, so the user thinks there is nothing out of the ordinary.

Certificate-based provisioning can be used to provide the provisionee's device certificate to the provisioner, and the provisioner can decide whether it trusts the certificate issuer and uses the provisionee's public key that is recorded in the certificate or not.

A device certificate can contain a reference to the policy under which the certificate was issued by a CA, and this may help the provisioner to decide whether to provision the device or not.

7.14 Traffic analysis attacks

Network header information in mesh network PDUs is obfuscated instead of being fully encrypted. The obfuscation is set up to keep passive eavesdroppers from tracking message traffic patterns between nodes.

With enough captured data a successful attack could be performed on this obfuscation, which would lead to an attacker gaining knowledge of source addresses and other header information in mesh messages. The destination address of a message is encrypted, not obfuscated.

The success of this type of attack is reliant upon the network key or the IV Index not changing on the mesh network. For this reason, periodic network key updates are strongly recommended.

Because the obfuscation is performed using the privacy key – a specific derivation of a network key that is not used for any other purpose – even if the privacy key value was compromised it does not lead to compromising the contents of the mesh message payload as the payload encryption is done using a separate derived key.

7.15 Reused nonce attack

As described in Section 4.6, subnet bridges are nodes that relay messages between subnets, re-encrypting data that passes through them with a different network key than the message was initially sent with. An attacker could try to abuse a subnet bridge by sending a message that reuses an already used nonce value to encrypt and authenticate a different payload. If a subnet bridge would re-encrypt such a message without questioning, then it would be breaking the cardinal rule in cryptography to not reuse nonces. This could lead to the attacker gaining access to the network key used on the other side of the bridge. Subnets are often used to allow guest access, so an attacker could target a subnet bridge between a guest network and the primary network after gaining guest access, instead of trying to gain access to the primary network directly.

Subnet bridges have a requirement in the Mesh Protocol specification [1] that maintains a replay protection list for all messages sent to bridged subnets, and they need to pass the replay protection list check to bridge any messages. This helps subnet bridges avoid being unwitting participants in attacks that attempt to reuse nonce values.

8 Known specification vulnerabilities

This section lists the known vulnerabilities in the Mesh Profile specification [4] and provides links to the Bluetooth® SIG security notices that provide information on mitigating the effect of the vulnerabilities.

These vulnerabilities have been addressed through additional recommendations, errata, and new options available in Mesh Protocol specification [1], Version 1.1.

For an up-to-date list of the Bluetooth SIG security notices, see https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/reporting-security/.

The same page contains information on how to report security issues.

CVE ID	Bluetooth SIG security notice
CVE-2020-26556	Malleable commitment in Bluetooth Mesh Profile provisioning
CVE-2020-26557	Predictable AuthValue in Bluetooth Mesh Profile provisioning leads to MITM
CVE-2020-26559	Bluetooth Mesh Profile AuthValue leak
CVE-2020-26560	Impersonation attack in Bluetooth Mesh Profile provisioning

Table 8.1: List of known Mesh Profile specification vulnerabilities

9 Security recommendations

The following methods are recommended to help mitigate the security risks associated with mesh networks that are identified in this Mesh Security Overview.

The amount of work to maintain the network security is highly dependent on factors such as the physical security of nodes, size of the network, device lifecycles, and guest access; larger networks where devices are often added or removed will require more management activity. Automation of repeated tasks, such as running the Key Refresh procedure, can reduce the management workload.

9.1 Separation of application domains

An application key is considered to belong to a given security domain. Configuration managers can help by providing separate application keys for different security domains, even for a single application and a single subnet. Then, even if some of the nodes using an application are compromised, an attacker will gain access to only a small part of the whole network's functionality. If any two nodes do not strictly need to talk to each other, then they use different application keys. In the Mesh Protocol specification [1], there is a limit of 4,096 application keys that can be used simultaneously in a mesh network.

For example, even though meeting room lights and garage lights at an office building are similar nodes, maybe even identical nodes, they can use different application keys to encrypt their application payload. A compromise of the garage lights, a possibility given that they are in a less physically secure location than the meeting room lights, could allow an attacker to turn on and off the garage lights but not turn on and off the meeting room lights.

If the nodes do need to share some common state, such as having a main light switch that turns off all the lights at the office, then it is recommended to have a "pairwise" application key between the main switch and each group of lights. When the security domains are arranged in this way, even if an attacker gains access to the garage lights, they would not be able to control any other lights at the office.

9.2 Physical access considerations

Physical barriers, such as walls and fences, can be taken into consideration when defining security domains. For example, a network can be partitioned into subnets based on whether devices are placed indoors or outdoors. Devices that can be more easily accessed by outsiders can be given their own set of application and network keys. If possible, subnet bridging between the indoor and the outdoor subnets can be configured to flow only from indoors to outdoors. Then, if an attacker gains access to an outdoor device, they would still not be able to attack any of the indoor devices.

9.3 Storage of sensitive data

Device keys are one of the most valuable pieces of information in an entire mesh network. Even if some of the nodes, or some network and application keys, are compromised, the device keys allow for reestablishing a secure network after the compromised nodes are identified and excluded from the network.

If a smartphone application uses the secure-storage application programming interfaces (APIs) provided by the operating system to store all network keys, application keys, and device keys, then the smartphone is allowed to encrypt the security credentials whenever they are not in use. With a strong enough lock code even a stolen smartphone would not help a thief to gain access to a house.

A mesh node can store its security credentials using a dedicated hardware mechanism for secure key storage, including keys associated with device certificates. If a node uses public key out-of-band provisioning, then using a dedicated storage mechanism can keep the private key of this key pair safe. The same set up applies for static OOB authentication data that is programmed at the factory.

9.4 Man-in-the-middle prevention

The provisioning process is critical to the security of a mesh network, as the provisioner is the gatekeeper that lets devices participate in a network. An attacker that manages to insert themselves into the middle of a provisioning process will gain the same access to the network as the provisionee, including the device key of the provisionee. With the device key, the attacker can listen for and decrypt configuration messages that deploy more encryption keys into the provisionee, making more subnets and applications susceptible to attack. Also, by examining configuration messages that set up model communication parameters on the provisionee the attacker could gain some information on network structure.

To defend against man-in-the-middle attacks, use out-of-band public key or out-of-band authentication data during provisioning. When using out-of-band authentication data, refer to the Mesh Protocol specification [1] Section 5.4.2.3 for more details on which security algorithm to apply.

It is highly recommended that the maximum available entropy is used when generating the authentication data value. For example, if devices are programmed with static out-of-band authentication data in production, then fully using the 256 bits available for static out-of-band authentication data is recommended, instead of truncating the effective data length (e.g., to 128 bits).

Using freshly generated input or output out-of-band authentication data for every provisioning attempt helps to defend against an attacker attempting to brute-force the authentication mechanism.

To help defend against attempts to confuse the user, the Mesh Protocol specification [1] forbids reusing the same authentication data in different security procedures, such as Bluetooth LE pairing. This applies to all types of out-of-band authentication data, including static out-of-band authentication data that is pre-programmed to the device.

To defend against downgrade attacks, out-of-band public keys can be used to hinder the man-in-the-middle from inserting itself into the provisioning flow. Another possibility is to use out-of-band authentication data containing more than 128 bits of entropy, as the security algorithm used in Mesh Profile specification [4] is capped at 128 bits. Additionally, a provisioner can impose a policy to disallow the use of the security algorithm which has known vulnerabilities, disallow the use of in-band public keys for the provisionee, or disallow using out-of-band authentication data with devices that appear to support only low-entropy mechanisms for entering input and output authentication data.

If neither out-of-band public key nor out-of-band authentication are used, then the provisioning protocol does not defend against man-in-the-middle attacks. Other means (such as physical security) may still be used to prevent man-in-the-middle attacks.

Certificate-based provisioning (see Section 3.8) can be used to acquire the public key of the provisionee along with information identifying the device in a format that can be cryptographically verified. Other mechanisms defined by device vendors are also possible but not included in the Mesh Protocol specification [1].

An attacker might try to mount a downgrade attack by blocking access to the provisionee's out-of-band public key. For example, an attacker may attempt to block a provisioner's access to the server from where a provisionee's certificate would be retrieved. It is recommended for the provisioner to refrain from provisioning the device if the device is expected to have an out-of-band key but that key cannot be accessed.

9.5 Blocking counterfeit products

Certificate-based provisioning can be used to check that a provisionee has been issued a cryptographic certificate by a legitimate CA that processes certificate requests according to a policy that is acceptable to the mesh network operator. A CA policy can define a set of acceptable production-related processes that a vendor has to meet before the CA issues certificates for the vendor's devices.

The provisioner can check the association between a device UUID and a public key from a certificate. Additionally, the provisioner can choose to refuse to provision devices with UUIDs that are not covered by a certificate that has been verified.

9.6 Device removal actions

When a device is removed from a mesh network for whatever reason, the network keys and application keys of that mesh network could still be stored within that device. Therefore, it is highly recommended to exclude a device from future Key Refresh procedures upon removal from a mesh network, and refresh the keys to exclude the device from the network.

Disposing of the removed device is highly recommended only after Key Refresh procedures for all the network and application keys the device had access to have been completed.

9.7 Key refresh periods

Periodically refreshing the network keys and application keys reinforces the strength of the obfuscation of the network header and helps keep an attacker from performing a successful brute-force attack on the network keys, application keys, or derived keys within a mesh network. Furthermore, short key refresh periods can limit the damage caused by a compromised network or application key, as the number of messages encrypted with a particular key will be limited and messages sent after the refresh cannot be decrypted by the attacker unless they make another effort to compromise the new key. It is highly recommended that all network keys and application keys be refreshed, using the network composition and traffic profile to decide the interval, to mitigate potential attacks. A Key Refresh procedure can be automated at a configuration manager, so that it does not cause a maintenance burden for the network administrator.

9.8 IV update periods

Refreshing the IV Index periodically has multiple benefits. It updates secure network beacon contents, helps to keep the nonces used in encryption and authentication fresh, and removes stale entries associated with past IV Index values from the replay protection lists. Furthermore, devices in a nonprimary subnet that cannot initiate an IV Index update may run out of sequence numbers if the IV Index remains the same for extended periods of time. It is highly recommended that the IV Update procedure be started periodically, considering the network composition and traffic profile in deciding the interval. An IV Index Update procedure can be automated at a configuration manager, so that it does not cause a maintenance burden for the network administrator.

9.9 Avoidance of fixed network keys

It is strongly recommended that no devices be shipped preconfigured with a static network key. Although a set of mesh devices could be built and provisioned at the point of manufacture and packaged into a single box that can be sold as a working mesh network to an end consumer, this is strongly discouraged because it requires a fixed network key. Because the consumer would not have access to the device keys for these devices, they could not reconfigure these devices and therefore would not be able to follow security recommendations to refresh the keys and perform IV Index updates or to protect the network after a device is removed.

Instead of provisioning devices at the point of manufacture and doing nothing more, this type of mesh solution can be distributed with an associated provisioning database that allows a provisioner to quickly connect to each node and perform the Device Key Refresh procedure that allows the provisioner to completely rekey the nodes, including the device keys, as explained in Section 6.5. After rekeying, the provisioner can add more network and application keys to the devices as needed. It is highly recommended that, even if this arrangement makes rekeying possible, the initial set of keys is not fixed across a class of devices but is unique to each unit sold.

The above recommendation is acceptable for an initial “starter-kit” installation, but it is not in itself enough for adding additional nodes to a preexisting mesh network. Even if those nodes are shipped with a provisioning database that documents the network keys and unicast addresses allocated to those nodes, it would not be possible to merge these two networks automatically because the unicast addresses might overlap. In this case, the provisioner can execute the Node Address Refresh procedure that changes the device key and the node unicast address assignment at the same time.

A “starter-kit” installation is recommended to include a provisioning database with only the list of devices in the starter kit along with their device UUIDs. The devices are not preconfigured with a fixed network key or a fixed configuration. Also, provisioning software is recommended to be able to recognize the presence of a “starter-kit” provisioning database and automatically walk the user through configuring the devices.

9.10 Distribution of a provisioning database

It is possible to distribute the provisioning database among multiple provisioners and configuration managers within a mesh network. This provisioning database contains highly sensitive information, such as the device keys of nodes within the mesh network along with network keys and application keys, and therefore can be considered highly confidential.

Whenever the provisioning database is transferred between devices, it is highly recommended that the transfer be made with the highest security possible. In order not to degrade the security of the network, the security used for this transfer needs to be at least as strong as the security used within the mesh network itself.

Out-of-band authentication can be used to help check that only the intended receivers can decrypt and authenticate the provisioning database, especially if the encrypted and authenticated provisioning database is transferred over an insecure network; for example, if it is attached to email, stored on a network file sharing service, or downloaded from a web page.

9.11 Using authentication in a multi-provisioner environment

A device being provisioned responds to any provisioner that initiates a provisioning session with it. This may lead into a situation where the device is provisioned to a wrong network, intentionally or unintentionally. While this does not cause harm to the intended network, it may cause unnecessary work in diagnosing and correcting the problem with the “malfunctioning” device, causing issues with the user experience of the network.

While it is the provisioner that controls the overall provisioning process, the provisionee can request out-of-band authentication during the provisioning process. This can be used to help check that the devices are provisioned by the intended provisioner.

9.12 Certificate revocation for vulnerable devices

During certificate-based provisioning, the provisioner will need to verify the device certificate associated with the device being provisioned. If the certificate includes an extension pointing to revocation information, such as an OCSP Responder location URI, then the provisioner is expected to make use of it to determine certificate revocation status as part of the verification before commencing with the rest of the provisioning process.

If a critical security flaw is found (e.g., if sensitive data such as security credentials can be extracted from device storage because of a hardware vulnerability), then the device vendor is strongly recommended to ask the CA to revoke the certificates issued to the vulnerable devices. This will help in mitigating the impact of the security flaw at network installations as the revoked certificate will alert the provisioner that the device has an issue that needs to be checked.

9.13 Using privacy-enhanced beacons and service advertisements

Mesh private beacons, Mesh Proxy Service advertising with private network identity, and Mesh Proxy Service advertising with Private Node Identity all serve to obfuscate the data in PDUs that are used to indicate node and network presence to nearby mesh nodes. Such data could be used by a passive observer to track a mobile mesh network if it wasn’t obfuscated.

It is recommended to use these forms of beaconing and service advertisements instead of the ones detailed in Mesh Profile specification [4]. If both beacons and GATT service advertisements are used in a network, then it is recommended to enable privacy for both and not one or the other.

9.14 Certificate management

The Mesh Protocol specification [1] requires a provisioner to validate a provisionee’s certificate before using the public key contained in the certificate for provisioning. The validation procedure includes, but is not limited to, checking the cryptographic signatures across the whole certificate chain from the root certificate to the provisionee’s certificate. RFC 5280 [7] sets the basis for the validation procedure, which requires the provisioner to have a set of trusted root certificates.

As part of network configuration, the network operator is responsible for deciding which set of root certificates the provisioner trusts. When the provisioner comes with a preconfigured set of root certificates the choice is implicitly made by choosing to use that provisioner. It is recommended that mesh network operators familiarize themselves with the certificate policy and Certification Practice Statement documents of a CA before deciding to trust the root certificates originating from that CA.

It is recommended that provisioner vendors clearly document how the mesh network operator can manage the set of trusted root certificates or what those certificates are when there is a preconfigured set of root certificates.

Furthermore, it is recommended that device certificates issued by a CA clearly indicate the CA policy under which the certificate has been issued by including a certificate policies extension in the certificate. This allows the provisioner to easily check if the certificate has been issued under an acceptable policy.

10 Acronyms and abbreviations

Acronym/Abbreviation	Meaning
AES	Advanced Encryption Standard
AID	Application Identifier
API	application programming interface
CA	certification authority
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CMAC	cipher-based message authentication code
EC	elliptic curve
ECDH	Elliptic Curve Diffie-Hellman
GATT	Generic Attributes Profile
IoT	Internet of Things
IV Index	initialization vector Index
KDF	key derivation function
NFC	near field communication
NID	network identification
OCSP	Online Certificate Status Protocol
OOB	out-of-band
PDU	protocol data unit
QR code	quick response code
TTL	time to live
URI	uniform resource identifier
UUID	universally unique identifier

Table 10.1: Acronyms and abbreviations

11 References

[1] Mesh Protocol specification, Version 1.1 or later

[2] Mesh Model specification, Version 1.1 or later

[3] Bluetooth Core Specification, Version 4.2 or later

[4] Mesh Profile specification, Version 1.0 or later

[5] The Bluetooth® Mesh Primer, https://www.bluetooth.com/bluetooth-mesh-primer/

[6] ITU, “ITU-T X.509 – Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks”, October 2019, https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=14033

[7] IETF, “RFC 5280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, May 2008, https://tools.ietf.org/html/rfc5280

[8] IETF, “RFC 6818 – Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, January 2013, https://tools.ietf.org/html/rfc6818

[9] IETF, “RFC 6960 – X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, June 2013, https://tools.ietf.org/html/rfc6960

[10] IETF, “RFC 8398 – Internationalized Email Addresses in X.509 Certificates”, May 2018, https://tools.ietf.org/html/rfc8398

[11] IETF, “RFC 8399 – Internationalization Updates to RFC 5280”, May 2018, https://tools.ietf.org/html/rfc8399

[12] IETF, “RFC 9654 – Online Certificate Status Protocol (OCSP) Nonce Extension”, August 2024, https://tools.ietf.org/html/rfc9654

[13] IETF, “RFC 3610 – Counter with CBC-MAC (CCM)”, September 2003, https://tools.ietf.org/html/rfc3610