BLE Mesh Security Overview

Application Note *v1.0*

INTRODUCTION

In July of 2017, the Bluetooth SIG released *Mesh Profile Specification v1.0* describing a Mesh Profile running on top of any device which is v4.0 or newer. This means that BLE mesh doesn't require a new radio. The goal of this document is to provide an introductory overview of Bluetooth Low Energy (BLE) mesh security.

OVERVIEW

BLE security in BLE mesh is ***mandatory and cannot be disabled***. All BLE mesh messages are encrypted. BLE mesh uses the following three types of security keys:

- Device Key (DevKey) – Used for provisioning and only known by the provisioning node.
- Network Keys (NetKey) – The network encryption key and Privacy key are derived from the NetKey.
- Application Keys (AppKey) – Used to encrypt/decrypt application messages.

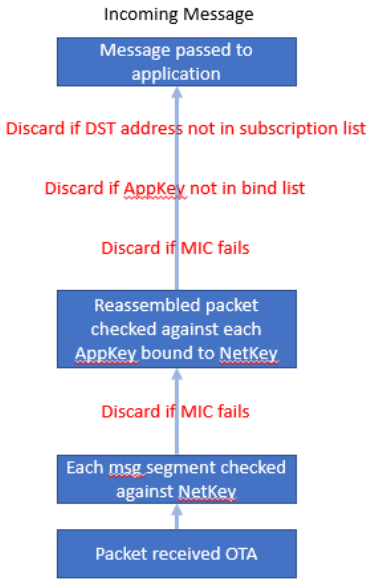

Figure 1: BLE mesh security keys

This setup allows for multiple levels of security but also provides great flexibility. For example, imagine a scenario where you need to control building lighting but also access control to doors. Within a single network and with a single network key, you could have multiple applications. You may need a light switch to turn on a light bulb; that light bulb, being powered, can act as a relay node for the network to propagate lighting messages through the network. However, that light bulb relay node could also be used to relay door access

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

messages without the light bulb being able to decrypt the message intended for doors. The access control messages can only be decrypted by the nodes with the access control application key.

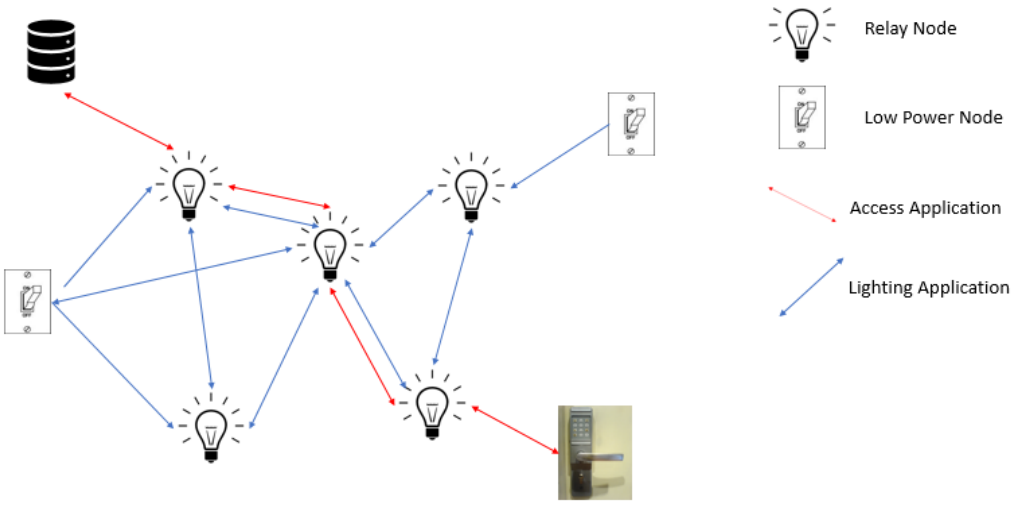

Figure 2: Site with single network key and two application keys

This implies that a mesh node can have multiple network and application keys. The mesh specification allows 4096 of each to be shared and are referred to by an index number 0 to 4095. Each message that is sent contains the index number information so that the receiving end can do a lookup to extract the appropriate key for decrypting and authenticating the message.

PROVISONING

An unprovisioned device, when powered up, sends unprovisioned beacon adverts containing a unique and unchanging device UUID that identifies it. A provisioner, on receipt of that advert, establishes a logical communication link (using adverts or a BLE connection) using the UUID as the device identifier. Then, using Elliptical Curve Diffie-Hellman, it negotiates a shared secret that becomes a long-term resource kept in the provisioner (along with the device UUID). That shared secret is then used to create a session key which is used to allocate one or more node addresses to the device and provide the primary network key.

A provisioner is the only entity in a mesh that is aware of all the members of the network. It's also the only entity that knows how to program the publication address and the subscription lists of all the nodes so that all the nodes can operate as a well-choreographed collective. All the while, each node is never aware of the full picture of the mesh network.

ENCRYPTION

By the time the messages get on the air, they have been ***encrypted twice***. Once with an application or device key and the second time with a network key. Each key is 128-bit long and the encryption algorithm uses AES in CCM mode.

An outgoing message at the application layer is not encrypted for the simple reason that it does not have any knowledge of the keys (only the underlying mesh stack is aware of them). In addition, an application does not even know or care about its own node address. When an application sends the message, the destination address

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0600

(also known as the publication address, given Ble Mesh uses a publish/subscribe paradigm) is added and the encryption is performed by the underlying mesh stack.

Replay Attacks

An eavesdropper could conduct a replay attack by capturing messages over the air and replaying them later. This can fool the attacked device into accepting a message that then has a malicious result.

BLE mesh uses two fields in the protocol data unit (PDU) – the sequence number and initialisation vector index (IV Index). The IV index is shared by all nodes in a given network. Together these values make up the Nonce (a number which is only used once). Each time a message is encrypted and published in a BLE mesh it is given a new Nonce value. If the sequence and IV index values are less than or equal to the previous message (PDU), the message is discarded.

BlackListing

There is the situation where a device, such as a light bulb, no longer works and is thrown away because the LED is faulty. In that case, there is a *trash-can* attack threat because the bulb still contains valid network and appkeys. The specification in this case allows for all keys in all the mesh nodes to be updated *except for* the discarded node (which is not updated). The discarded node is blacklisted by the simple process of no longer having the correct key to encyrpt and decrypt.

This occurs because of a Key Refresh procedure (consult the specification for details, if necessary). The expectation is that this procedure is never exposed at the application level; it is transparently dealt with by the underlying mesh stack in the node – only a provisioner entity is required to initiate it. Because of this, the only exposer to the application layer is in the provisioner, and even that is simply to initiate a refresh.

Subnets

It is possible to have both multiple application keys and multiple network keys within a single BLE mesh network. In a large building mesh network, it may be beneficial to split the building into subnets. Each subnet confines messages, perhaps to prevent messages being relayed across multiple floors when not required, but also to add an additional layer of security for sensitive applications. Not only do you need the correct application key to get access to the message payload, you also need the correct network key.

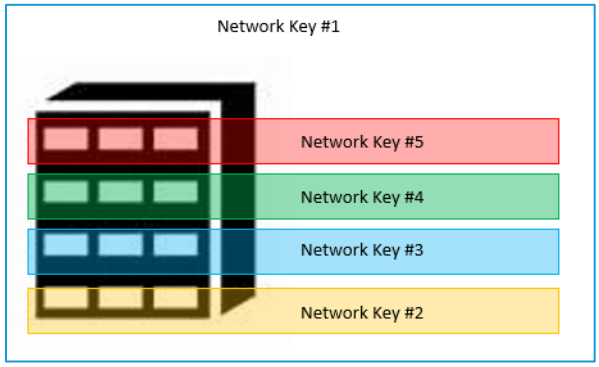

Figure 3: Network keys

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0600

LAIRD *SMART*BASIC

All the complexities of encrypting, decrypting, and managing mesh messages are handled by the *smart*BASIC firmware. You only have to create a *smart*BASIC application that defines the behavior of the application when receiving and sending those messages. The behavior is defined by registering a list of opcodes that a device processes as well as their handlers. This is all done using appropriate *smart*BASIC functions and events.

Critical information like node address, app keys, net keys, publication address, subscription list, key bindings, and many other configuration information is managed completely by the *smart*BASIC firmware; they are *only available after a device is provisioned into a network by a provisioner.*

REFERENCES

Bluetooth Mesh Specification v1.0 – https://www.bluetooth.com/specifications/mesh-specifications

About Bluetooth Mesh – https://blog.bluetooth.com/answers-to-your-questions-about-bluetooth-mesh

Laird Bluetooth Modules – https://www.lairdtech.com/product-categories/connectivity-solutions/bluetooth-modules

REVISION HISTORY

Version	Date	Notes	Contributor(s)	Approver
1.0	09 Jan 2018	Initial Release	Mark Duncombe	Jonathan Kaye

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0600