AN99209

PSoC® 4 BLE and PRoC™ BLE: Bluetooth LE 4.2 Features

Author: utsv@cypress.com;sasd@cypress.com;ankc@cypress.com

Associated Part Family:CYBL11X7X and CY8C4XX8-BL5XX

Related Application Notes: For a complete list, click here.

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/AN99209

AN99209 provides an overview of Bluetooth Low Energy (BLE) 4.2 features, their benefits, and applications. It also explains how applications can use these features using Cypress's PSoC 4/PRoC BLE 4.2 devices.

Contents

1 Introduction

On 2nd December 2014, the Bluetooth Special Interest Group (SIG) released the Bluetooth Core Specification version 4.2. This version introduced three major features: LE Data Packet Length Extension, Link Layer Privacy (also referred to as Privacy 1.2), and LE Secure Connections. These features make Bluetooth Low Energy (BLE) or Bluetooth Smart devices smarter, faster, and more secure; ideal for Internet of Things (IoT). You can download the specification here.

Cypress's PRoC BLE (CYBL11X7X) and PSoC 4 BLE (CY8C4XX8-BL5XX) devices are fully compliant with the Bluetooth 4.2 Specification and support all the three new features. These devices also have a DMA Controller to enable data transfer without CPU intervention. They also include 256 KB of flash and 32 KB of RAM to enable over-the-air (OTA) firmware updates without external memory. Legacy BLE devices will support the LE Secure Connection feature by upgrading the BLE Component in PSoC Creator to version 3.0 or higher. Table 1 summarizes the supported Bluetooth 4.2 features of the different Cypress devices.

Table 1. Support for New Bluetooth Features on Cypress's Devices

Features	Devices with Bluetooth 4.1	Devices with Bluetooth 4.2
LE Data Packet Length Extension	No	Yes

Link Layer Privacy	No	Yes
LE Secure Connections	Yes	Yes

This application note explains the basics of the new Bluetooth 4.2 features, their benefits, and how to use them in your application with Cypress's PRoC BLE and PSoC 4 BLE devices. This application note assumes that you have a basic understanding of the BLE architecture and terms.

- If you are new to either BLE or PSoC, see the application note AN91627 - Getting Started with PSoC® 4 BLE or AN94020 - Getting Started with PRoC™ BLE.
- To understand the BLE Component in PSoC Creator and to learn how to develop applications based on standard BLE services, see the application note AN91184 - PSoC 4 BLE Designing BLE Applications.
- For the Bluetooth specifications, visit the Bluetooth SIG website.

2 PSoC/PRoC Resources

Cypress provides a wealth of data at www.cypress.com to help you select the right PSoC (Programmable System on Chip) and PRoC (Programmable Radio on Chip) device, and quickly and effectively integrate the device into your design. For a comprehensive list of resources, see KBA86521 - How to Design with PSoC 3, PSoC 4, and PSoC 5LP. The following is an abbreviated list for PSoC 4 BLE and PRoC BLE:

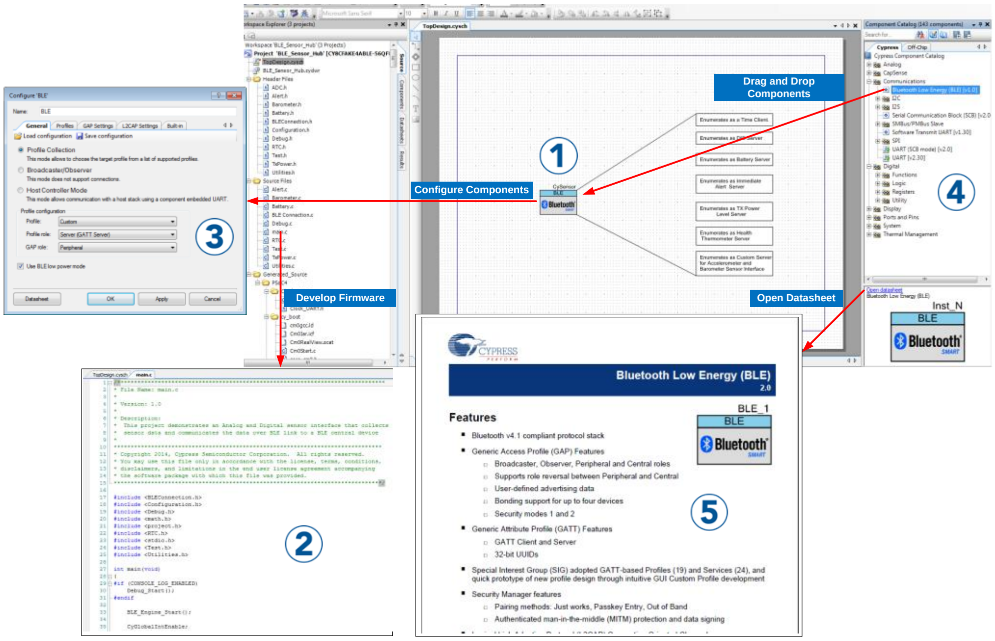
- **Overview:** Bluetooth® Low Energy Portfolio, Cypress Wireless/RF Roadmap
- **Product Selectors:** PSoC 4 BLE or PRoC BLE. In addition, PSoC Creator includes a device selection tool.
- **Datasheets** describe and provide electrical specifications for PSoC 4 BLE and PRoC BLE device families.
- **CapSense Design Guides:** Learn how to design capacitive touch-sensing applications with the PSoC 4, PSoC 4 BLE, and PRoC BLE families of devices.
- **Application Notes and Code Examples** cover a broad range of topics, from basic to advanced level. Many of the application notes include code examples.
- **Technical Reference Manuals (TRMs)** provide detailed descriptions of the architecture and registers in the PSoC 4 BLE and PRoC BLE devices.
- **Development Kits:**
  - CY8CKIT-042-BLE, BLE Pioneer Kit enables customers to evaluate and develop BLE applications using PSoC 4 BLE and PRoC BLE devices.
  - CY8CKIT-143A, PSoC 4 BLE 256KB Module with Bluetooth 4.2 Radio enables customers to evaluate the CY8C4XX8-BL5XX family of devices.
  - CY5676A, PRoC BLE 256KB Module with Bluetooth 4.2 Radio enables customers to evaluate the CYBL11X7X family of devices.
  - CY5677, CySmart BLE 4.2 USB Dongle enables customers to test and debug BLE 4.2 features with the CySmart PC tool.
  - CY5682, PRoC BLE Touch Mouse RDK provides a production-ready implementation of a BLE touch mouse.
  - CY5672, PRoC BLE Remote Control RDK provides a production-ready implementation of a BLE remote control.
- The **MiniProg3** device provides an interface for flash programming and debugging.
- **CySmart** is a BLE Host emulation tool for Windows PCs. The tool provides an easy-to-use GUI to enable customers to test and debug their BLE Peripheral applications.
- **CySmart – Mobile Apps** are Android™/iOS® apps developed by Cypress. These can be used to connect and test various BLE products including the BLE development kits from Cypress.
- **Cypress's Custom BLE Profiles and Services:** Cypress has defined several custom BLE profiles and services. These enable customers to send data over BLE for features that are not supported by Bluetooth SIG-specified standard BLE profiles and services.

3 PSoC Creator

PSoC Creator is a free Windows-based Integrated Design Environment (IDE). It enables concurrent hardware and firmware design of systems based on PSoC 3, PSoC 4, PSoC 4 BLE, PRoC BLE, and PSoC 5LP. As shown in Figure 1 - with PSoC Creator, you can:

1. Drag and drop Components to build your hardware system design in the main design workspace.
2. Co-design your application firmware with the PSoC hardware.
3. Configure the Components using configuration tools.
4. Explore the library of more than 100 Components.
5. Review the Component datasheets

Figure 1. PSoC Creator Schematic Entry and Components


Drag and Drop Components
Configure Components
Develop Firmware
Open Datasheet
1
2
3
4
5
CySensor
BLE
Bluetooth SMART
Enumerates as a Time Client
Enumerates as DIS Server
Enumerates as Battery Server
Enumerates as Immediate Alert Server
Enumerates as TX Power Level Server
Enumerates as Health Thermometer Server
Enumerates as Custom Server for Accelerometer and Barometer Sensor Interface
Inst_N
BLE_1
Datasheet
OK
Apply
Cancel

Configure 'BLE'

Name: BLE

General | Profiles | GAP Settings | L2CAP Settings | Built-in

Load configuration | Save configuration

● Profile Collection
This mode allows to choose the target profile from a list of supported profiles.

○ Broadcaster/Observer
This mode does not support connections.

○ Host Controller Mode
This mode allows communication with a host stack using a component embedded UART.

Profile configuration
Profile: Custom
Profile role: Server (GATT Server)
GAP role: Peripheral

☑ Use BLE low power mode

```
/*******************************************************************************
* File Name: main.c
*
* Version: 1.0
*
* Description:
*  This project demonstrates an Analog and Digital sensor interface that collects
*  sensor data and communicates the data over BLE link to a BLE central device
*
********************************************************************************
* Copyright 2014, Cypress Semiconductor Corporation.  All rights reserved.
* You may use this file only in accordance with the license, terms, conditions,
* disclaimers, and limitations in the end user license agreement accompanying
* the software package with which this file was provided.
*******************************************************************************/

#include <BLEConnection.h>
#include <Configuration.h>
#include <Debug.h>
#include <math.h>
#include <project.h>
#include <RTC.h>
#include <stdio.h>
#include <Test.h>
#include <Utilities.h>

int main(void)
{
#if (CONSOLE_LOG_ENABLED)
    Debug_Start();
#endif

    BLE_Engine_Start();

    CyGlobalIntEnable;
```

Bluetooth Low Energy (BLE)
2.0

Features

- Bluetooth v4.1 compliant protocol stack
- Generic Access Profile (GAP) Features
  - Broadcaster, Observer, Peripheral and Central roles
  - Supports role reversal between Peripheral and Central
  - User-defined advertising data
  - Bonding support for up to four devices
  - Security modes 1 and 2
- Generic Attribute Profile (GATT) Features
  - GATT Client and Server
  - 32-bit UUIDs
- Special Interest Group (SIG) adopted GATT-based Profiles (19) and Services (24), and quick prototype of new profile design through intuitive GUI Custom Profile development
- Security Manager features
  - Pairing methods: Just works, Passkey Entry, Out of Band
  - Authenticated man-in-the-middle (MITM) protection and data signing

3.1 PSoC Creator Help

Visit the PSoC Creator home page to download and install the latest version of PSoC Creator. Then launch PSoC Creator and navigate to the following items:

- **Quick Start Guide**: Choose **Help** > **Documentation** > **Quick Start Guide**. This guide gives you the basics for developing PSoC Creator projects.
- **Simple Component Code Examples**: Choose **File** > **Code Example**. These code examples demonstrate how to configure and use PSoC Creator Components.
- **System Reference Guide**: Choose **Help** > **System Reference Guides**. This guide lists and describes the system functions provided by PSoC Creator.
- **Component Datasheets**: Right-click a Component and select “Open Datasheet.” Visit the PSoC 4/PRoC BLE Component Datasheets page for a list of all PSoC 4/PRoC BLE Component datasheets.
- **Document Manager**: PSoC Creator provides a document manager to help you to easily find and review document resources. To open the document manager, choose the menu item **Help** > **Document Manager**.

3.2 Code Examples

PSoC Creator includes a large number of code examples. These projects are available from the PSoC Creator Start Page, as shown in Figure 2.

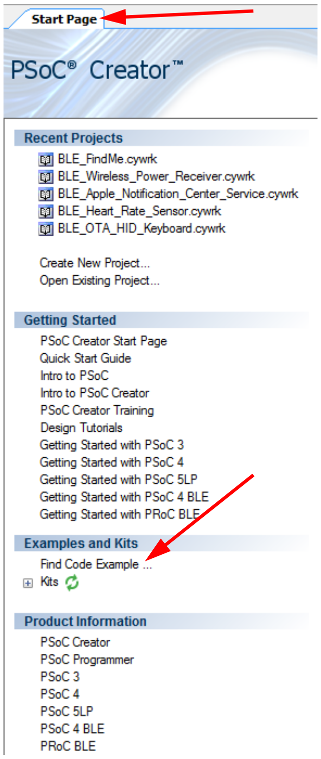
Code examples can speed up your design process by starting you off with a complete design, instead of a blank page. The code examples also show how PSoC Creator Components can be used for various applications.

In the **Find Code Example** dialog shown in Figure 3, you have several options:

- Filter for examples based on architecture or device family i.e. PSoC 4, PSoC 4 BLE, PRoC BLE, and so on; category; or keyword.
- Select from the menu of examples offered based on the **Filter Options**. There are more than 30 BLE code examples for your reference, as shown in Figure 3.
- Review the datasheet for the selection (on the **Documentation** tab)
- Review the code example for the selection. You can copy and paste code from this window to your project, which can help speed up code development.
- Or create a new project (and a new workspace if needed) based on the selection. This can speed up your design process by starting you off with a complete basic design. You can then adapt that design to your application.

In addition to the PSoC Creator code examples, you can also find BLE example projects in Cypress's GitHub repository.

Figure 2. Code Examples in PSoC Creator



Recent Projects

- BLE_FindMe.cywrk
- BLE_Wireless_Power_Receiver.cywrk
- BLE_Apple_Notification_Center_Service.cywrk
- BLE_Heart_Rate_Sensor.cywrk
- BLE_OTA_HID_Keyboard.cywrk

Create New Project...
Open Existing Project...

Getting Started

PSoC Creator Start Page
Quick Start Guide
Intro to PSoC
Intro to PSoC Creator
PSoC Creator Training
Design Tutorials
Getting Started with PSoC 3
Getting Started with PSoC 4
Getting Started with PSoC 5LP
Getting Started with PSoC 4 BLE
Getting Started with PRoC BLE

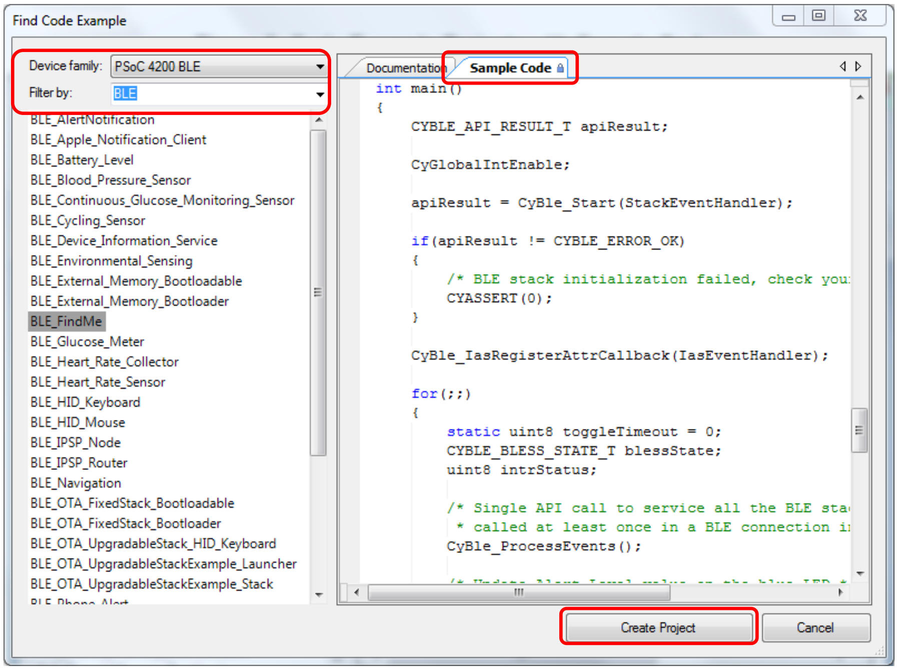
Examples and Kits

Find Code Example ...
⊞ Kits

Product Information

PSoC Creator
PSoC Programmer
PSoC 3
PSoC 4
PSoC 5LP
PSoC 4 BLE
PRoC BLE

Figure 3. Code Examples with Sample Code



Device family: PSoC 4200 BLE
Filter by: BLE

BLE_AlertNotification
BLE_Apple_Notification_Client
BLE_Battery_Level
BLE_Blood_Pressure_Sensor
BLE_Continuous_Glucose_Monitoring_Sensor
BLE_Cycling_Sensor
BLE_Device_Information_Service
BLE_Environmental_Sensing
BLE_External_Memory_Bootloadable
BLE_External_Memory_Bootloader
BLE_FindMe
BLE_Glucose_Meter
BLE_Heart_Rate_Collector
BLE_Heart_Rate_Sensor
BLE_HID_Keyboard
BLE_HID_Mouse
BLE_IPSP_Node
BLE_IPSP_Router
BLE_Navigation
BLE_OTA_FixedStack_Bootloadable
BLE_OTA_FixedStack_Bootloader
BLE_OTA_UpgradableStack_HID_Keyboard
BLE_OTA_UpgradableStackExample_Launcher
BLE_OTA_UpgradableStackExample_Stack
BLE_Phone_Alert

```
int main()
{
    CYBLE_API_RESULT_T apiResult;

    CyGlobalIntEnable;

    apiResult = CyBle_Start(StackEventHandler);

    if(apiResult != CYBLE_ERROR_OK)
    {
        /* BLE stack initialization failed, check you
        CYASSERT(0);
    }

    CyBle_IasRegisterAttrCallback(IasEventHandler);

    for(;;)
    {
        static uint8 toggleTimeout = 0;
        CYBLE_BLESS_STATE_T blessState;
        uint8 intrStatus;

        /* Single API call to service all the BLE sta
         * called at least once in a BLE connection i
        CyBle_ProcessEvents();
```

4 Introduction to BLE 4.2 Features

Bluetooth 4.2 introduced three major features: LE Data Packet Length Extension, Link Layer (LL) Privacy, and Low Energy Secure Connections. The following sections explain the new features in detail.

5 LE Data Packet Length Extension

The Link Layer (LL) is the part of the BLE protocol stack that takes care of advertising, scanning, creating, and maintaining connections.

The Link Layer packet format is shown in the Figure 4. Each packet consists of four fields: the Preamble, the Access Address, the Protocol Data Unit (PDU), and the Cyclic Redundancy Check (CRC). The packets transmitted during advertising, scanning, or connection creation procedures use the Advertising Channel PDU. The packets transmitted to exchange data with connected devices use the Data Channel PDU.

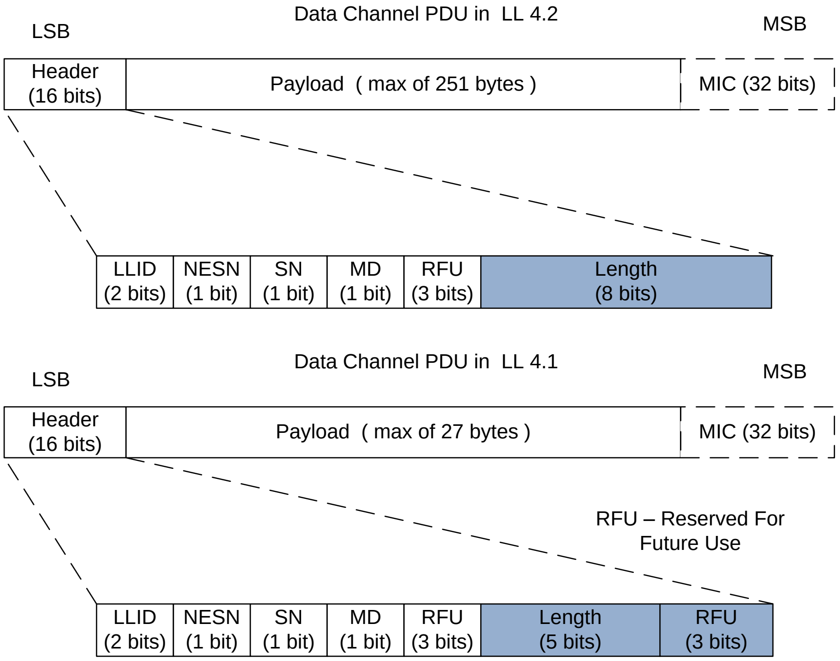
Figure 4. Link Layer Packet Format

Preamble (1 byte)	Access Address (4 bytes)	Data Protocol Data Unit (PDU) (2 to 257 bytes in BLE 4.2 & 2 to 33 bytes in BLE 4.1)	CRC (3 bytes)

A Data Channel PDU includes a 16-bit Header, a variable-size Payload field, and an optional Message Integrity Check (MIC) field. In the Bluetooth 4.2 Specification, the maximum size of the Payload field in the Data Channel PDU was increased from 27 bytes to 251 bytes, thus increasing the capacity of the Data Channel by approximately 10 times (see Higher Throughput).

Figure 5 shows the difference between the Data Channel PDU in Bluetooth 4.2 and Bluetooth 4.1.

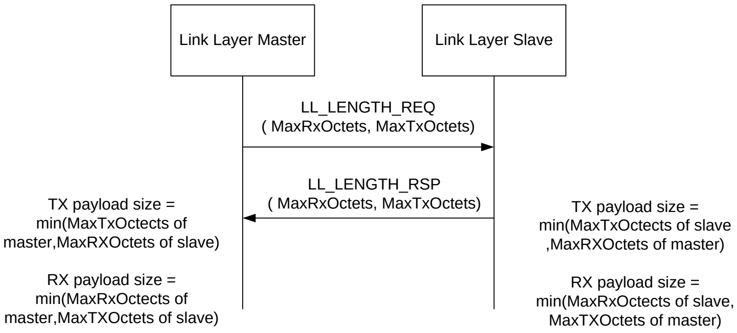
Figure 5. LE Data Channel PDU in Bluetooth Specification Versions 4.2 and 4.1



The Length field in the Header specifies the number of bytes of data following the Header. The size of the Length field in the Header is increased from 5 bits in Bluetooth 4.1 to 8 bits in Bluetooth 4.2, thereby increasing the range of the Length field value from 31 to 255. The Message Integrity Check (MIC) used in encrypted packets is 4 bytes long. Therefore, the maximum possible payload size is 251 bytes in Bluetooth 4.2 and 27 bytes in Bluetooth 4.1. To learn about the other fields in the Header, read the Bluetooth Core Specification Version 4.2, Volume 6, Part-B, Section 2.4.

The Link Layer uses the Data Length Update procedure to negotiate the payload size used in transmit and receive directions as shown in Figure 6. Two newly added Link Layer control PDUs, LL_LENGTH_REQ and LL_LENGTH_RSP are used to exchange the maximum payload size supported by the devices. The maximum payload size that can be transmitted by a device is called MaxTxOctets, while the maximum payload size that can be received by a device is called MaxRxOctets. The actual transmit (TX) and receive (RX) payload sizes are decided based on the Data Length Update procedure. The actual TX payload size is the minimum of the local MaxTxOctets and the peer MaxRxOctets parameters; similarly, the actual RX payload size is the minimum of the local MaxRxOctets and the peer MaxTxOctets parameters. The Link Layer uses the default 27-bytes payload size until the Data Length Update procedure is completed. A device that does not support this feature responds with the LL_UNKNOWN_RSP PDU when it receives the LL_LENGTH_REQ PDU; the Link Layer will then use the default 27-bytes payload size.

Figure 6. Data Length Update Procedure



5.1 Benefits

The LE Data Packet Length Extension feature enables applications to get higher throughput, lower power consumption, and asymmetric bandwidth. These benefits are available only when the following conditions are met:

- Both the BLE devices support LE Data Packet Length Extension
- Higher-layer protocols use greater than the default (23 bytes) Maximum Transmission Unit (MTU) size

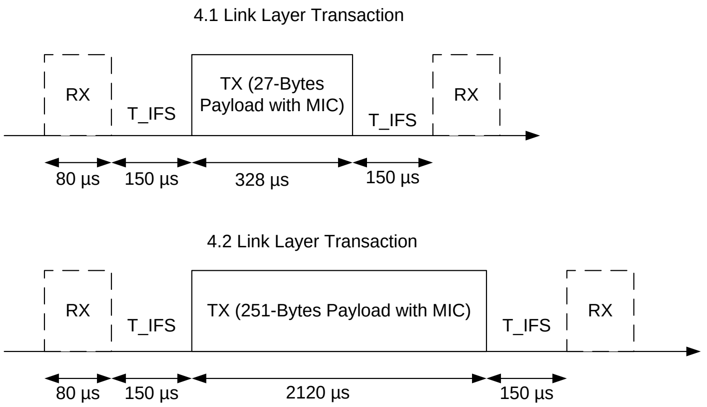
5.1.1 Higher Throughput

The LE Data Packet Length Extension feature enables you to get approximately 2.6 times higher throughput through the Link Layer. Figure 4 shows the Link Layer packet format and Figure 5 shows the various fields of a Data Channel PDU. The MIC field is only appended in the encrypted packets for a Data Channel PDU with a non-zero payload size.

The shortest packet transmission time is 80 µs for a payload size of 0 bytes. The longest packet transmission time is 328 µs for a payload size of 27 bytes in Bluetooth 4.1 and 2120 µs for a payload size of 251 bytes in Bluetooth 4.2.

Figure 7 shows typical Link Layer transactions while transmitting data for Bluetooth 4.1 and Bluetooth 4.2. A single data packet transaction involves an RX of an empty packet (payload size=0), an Inter Frame Spacing (T_IFS) of 150 µs, TX of a maximum payload size packet, and T_IFS until the next RX operation. At this point, the procedure is repeated. The second RX in the figure is where the first TX packet is acknowledged and is the beginning of a new transaction.

Figure 7. Link Layer Transaction


4.1 Link Layer Transaction
RX
T_IFS
TX (27-Bytes Payload with MIC)
T_IFS
RX
80 µs
150 µs
328 µs
150 µs
4.2 Link Layer Transaction
RX
T_IFS
TX (251-Bytes Payload with MIC)
T_IFS
RX
80 µs
150 µs
2120 µs
150 µs

The Link Layer throughput (i.e., the throughout available to the higher layers of the BLE protocol stack) is defined as:

Throughput = Payload Size / Time for Single Transaction

In Bluetooth 4.1, the payload size is 27 bytes (216 bits) and the total time taken for single transaction is 708 µs that gives the theoretical throughput of 298 kbps.

In Bluetooth 4.2, the payload size is 251 bytes (2008 bits) and the total time is 2500 µs that gives the theoretical throughput of 784 kbps. This gives approximately 2.6 times throughput for a Bluetooth 4.2 device as compared to a Bluetooth 4.1 device.

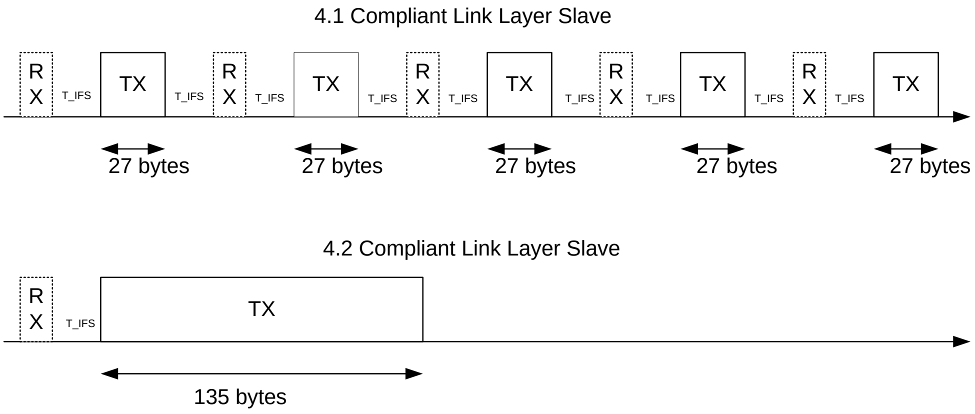
5.1.2 Low Power Consumption

Under the ideal conditions with no air-interference, the LE Data Packet Length Extension feature helps reduce the power consumption due to a more efficient use of the bandwidth.

In Bluetooth 4.2, a lower number of transactions are required to transfer a given amount of data compared to Bluetooth 4.1. This reduces the time for which the radio is active and allows the device to remain in a low-power mode for a longer duration, thereby reducing the average current consumption.

Figure 8 illustrates this mechanism assuming a 135-byte Link Layer payload. In Bluetooth 4.1, the 135-byte payload is split into 27-byte payloads and sent over five transactions. In Bluetooth 4.2, the 135-byte payload is sent in a single transaction.

Figure 8. 135 Bytes Data Transfer Case



5.1.3 Asymmetric Bandwidth

Asymmetric bandwidth means that the TX and RX bandwidths are not same. It is useful in applications like over-the-air (OTA) firmware upgrade that require higher bandwidth in the RX direction and lower bandwidth in the TX direction. By a proper selection of values for the local MaxTxOctets and MaxRxOctets parameters, you can achieve asymmetric bandwidth. For example, setting the MaxTxOctets to 251 bytes and the MaxRxOctets to 27 bytes provides more bandwidth in the transmit direction. Similarly, setting the MaxTxOctets to 27 bytes and the MaxRxOctets to 251 bytes provides more bandwidth in the receive direction.

5.2 Applications

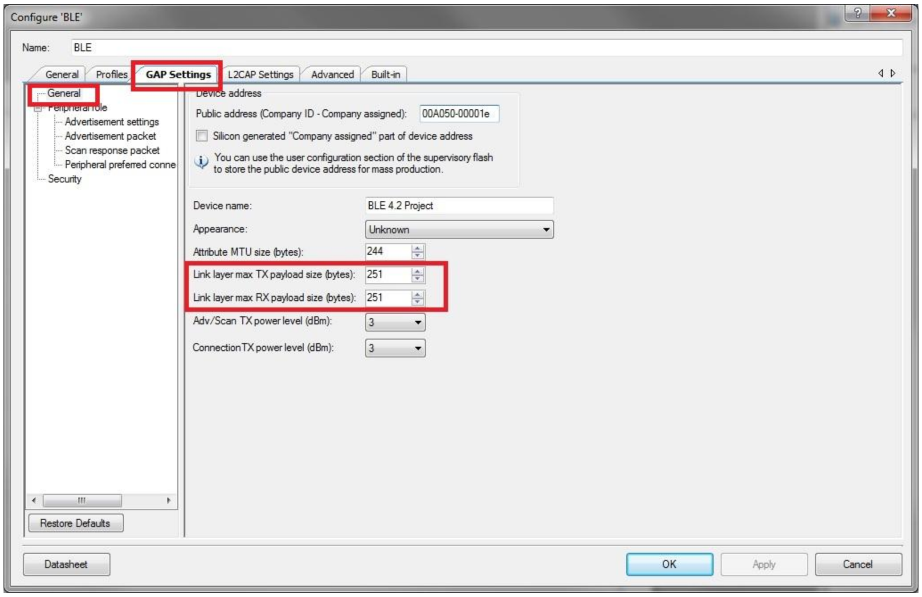
- Audio-over-BLE can make use of the higher bandwidth to reduce the processing power needed to compress the data.
- Over-the-air (OTA) firmware upgrade can be completed in a shorter time with a lower power consumption.
- Internet Protocol Support Profile (IPSP) packets can be exchanged faster, resulting in faster discovery and transactions.
- Data from multiple sensors can be logged faster with the increase in the data payload size.

5.3 Developing Applications Using LE Data Packet Length Extension with PSoC Creator

5.3.1 Component Configuration

To use the LE Data Packet Length Extension feature, you have to configure the BLE Component with the appropriate values for the MaxTxOctets using the **Link Layer max TX payload size (bytes)** parameter and the MaxRxOctets using the **Link Layer max RX payload size (bytes)** parameter in the **GAP Settings** > **General** section as shown in Figure 9.

Figure 9. Configuring LE Data Packet Length Extension Parameters in BLE Component



Name: BLE

Device address

Public address (Company ID - Company assigned): 00A050-00001e

□ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: BLE 4.2 Project

Appearance: Unknown

Attribute MTU size (bytes): 244

Link layer max TX payload size (bytes): 251

Link layer max RX payload size (bytes): 251

Adv/Scan TX power level (dBm): 3

Connection TX power level (dBm): 3

5.3.2 Application handling

The BLE Stack automatically negotiates the TX and the RX payload sizes with the peer device immediately after a connection is established, based on the component configuration. Table 2 summarizes the BLE Stack events, a description of the event, and the actions taken to make use of the LE Data Packet Length Extension feature.

Table 2 BLE Stack Events and Action for the LE Data Packet Length Extension Feature

BLE Stack Event Name	Event Description	Event Handler Action
CYBLE_EVT_GAP_DATA_LENGTH_CHANGE	Reports the negotiated TX and the RX length	Informative event.

The negotiated maximum transmit and receive payload sizes for the connection are reported to the application through the `CYBLE_EVT_GAP_DATA_LENGTH_CHANGE` BLE stack event. The application firmware can use this event to know the change in maximum TX and RX payload sizes for the connection.

Table 3 provides the list of the new APIs (along with the description) to support the LE Data Packet Length Extension feature.

Table 3. New APIs for the LE Data Packet Length Extension Feature

API	Description
`CyBle_GapSetDataLength`	Sets the new TX payload size to initiate a new data length update procedure.

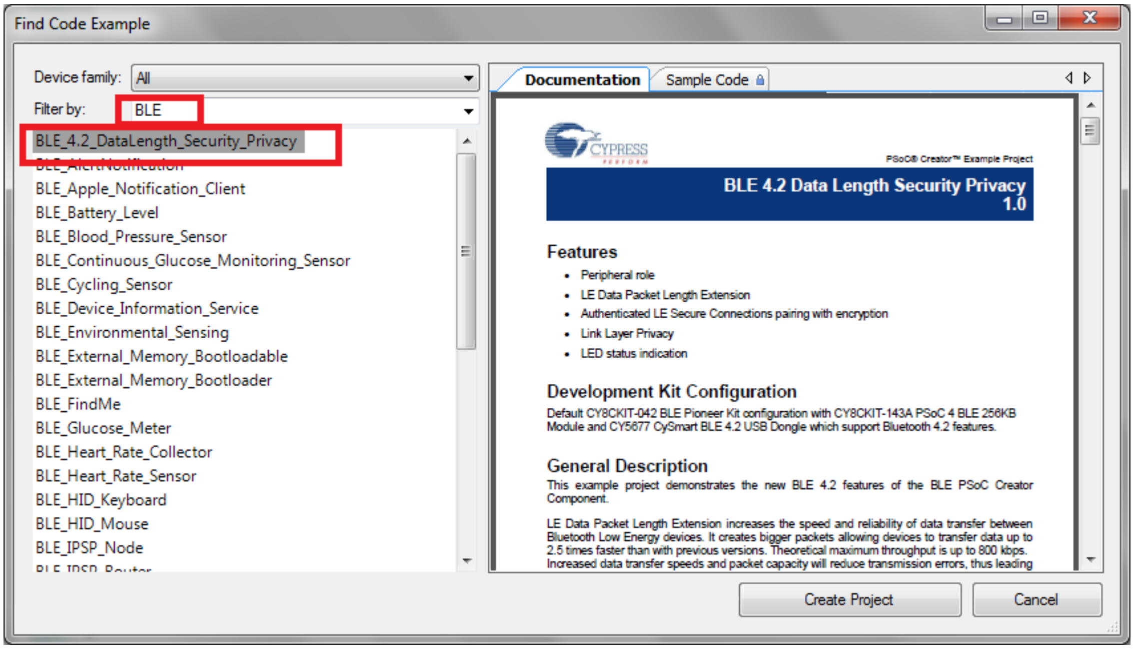
After connection, the MaxTxOctets parameter can be changed anytime by using the CyBle_GapSetDataLength API. This API initiates a fresh Data Length Update procedure and the actual transmit and receive payload sizes are reported to the application after the procedure is complete via the `CYBLE_EVT_GAP_DATA_LENGTH_CHANGE` event.

See the BLE Component datasheet for more details on BLE protocol stack events and APIs.

5.3.3 Example Project

The **BLE_4.2_DataLength_Security_Privacy** example project available with PSoC Creator uses the LE Data Packet Length Extension feature. The example project can be accessed using **PSoC Creator** > **File** > **Code Example** with the filter item set to **BLE** as shown in Figure 10.

Figure 10. BLE 4.2 Example Project



Find Code Example

Device family: All

Filter by: BLE

BLE_4.2_DataLength_Security_Privacy
BLE_AlertNotification
BLE_Apple_Notification_Client
BLE_Battery_Level
BLE_Blood_Pressure_Sensor
BLE_Continuous_Glucose_Monitoring_Sensor
BLE_Cycling_Sensor
BLE_Device_Information_Service
BLE_Environmental_Sensing
BLE_External_Memory_Bootloadable
BLE_External_Memory_Bootloader
BLE_FindMe
BLE_Glucose_Meter
BLE_Heart_Rate_Collector
BLE_Heart_Rate_Sensor
BLE_HID_Keyboard
BLE_HID_Mouse
BLE_IPSP_Node
BLE_IPSP_Router

PSoC® Creator™ Example Project

BLE 4.2 Data Length Security Privacy 1.0

Features

- Peripheral role
- LE Data Packet Length Extension
- Authenticated LE Secure Connections pairing with encryption
- Link Layer Privacy
- LED status indication

Development Kit Configuration

Default CY8CKIT-042 BLE Pioneer Kit configuration with CY8CKIT-143A PSoC 4 BLE 256KB Module and CY5677 CySmart BLE 4.2 USB Dongle which support Bluetooth 4.2 features.

General Description

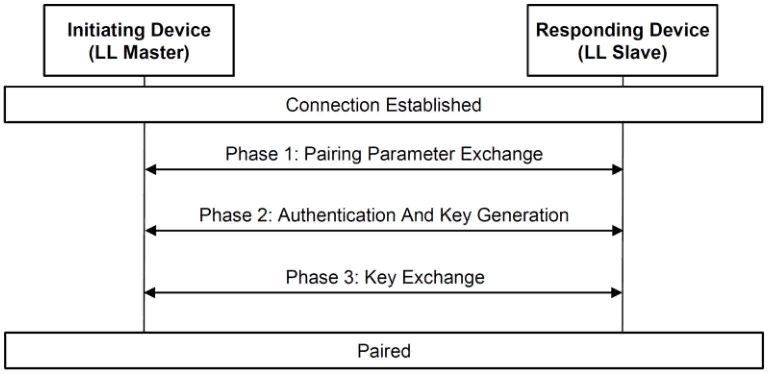
This example project demonstrates the new BLE 4.2 features of the BLE PSoC Creator Component.

LE Data Packet Length Extension increases the speed and reliability of data transfer between Bluetooth Low Energy devices. It creates bigger packets allowing devices to transfer data up to 2.5 times faster than with previous versions. Theoretical maximum throughput is up to 800 kbps. Increased data transfer speeds and packet capacity will reduce transmission errors, thus leading

6 Low Energy Secure Connections

Pairing is the process of authentication and key-exchange between two BLE devices. Pairing is a three phase process as shown in the Figure 11. LE Secure Connections is an enhanced security feature introduced in Bluetooth 4.2. It uses a Federal Information Processing Standards (FIPS) compliant algorithm called Elliptic Curve Diffie-Hellman (ECDH) for key generation, and a new procedure for key exchange. An Association Model in BLE is a model that defines the method of pairing based on the input and output capabilities of the two BLE devices. Bluetooth 4.2 introduces a new association model called Numeric Comparison (NC).

Figure 11. Pairing Process



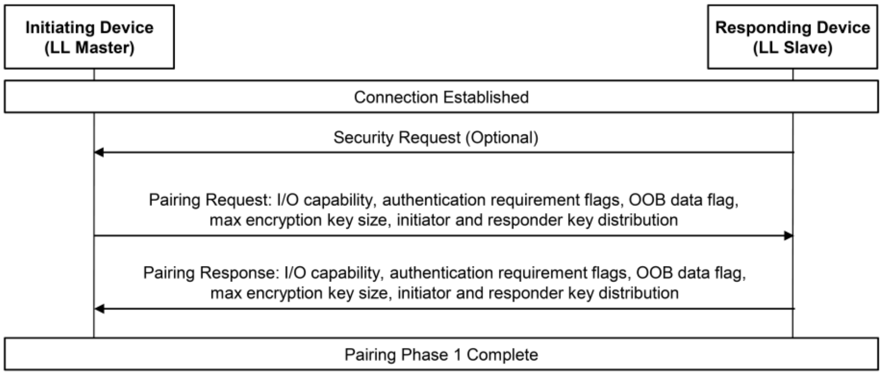
The following sections describe the three phases of the pairing process and how the LE Secure Connections feature affects each phase.

6.1 Update to Pairing Process

6.1.1 Pairing Phase 1

In phase 1, the initiating device and the responding device exchange pairing parameters like input/output capabilities, authentication requirement flags, encryption key sizes and Out-Of-Band (OOB) data availability. Phase 1 of the pairing process, as shown in Figure 12, is common for LE legacy pairing and LE Secure Connections.

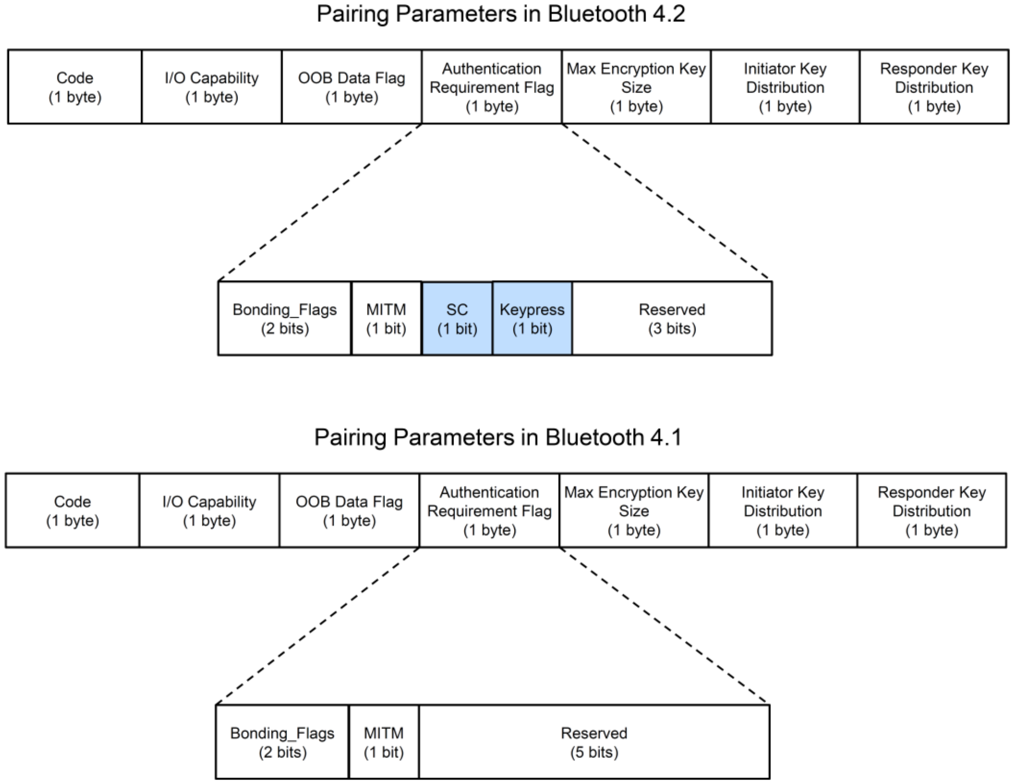
Figure 12. LE Pairing Phase 1


Initiating Device
(LL Master)
Responding Device
(LL Slave)
Connection Established
Security Request (Optional)
Pairing Request: I/O capability, authentication requirement flags, OOB data flag,
max encryption key size, initiator and responder key distribution
Pairing Response: I/O capability, authentication requirement flags, OOB data flag,
max encryption key size, initiator and responder key distribution
Pairing Phase 1 Complete

The initiating device (a Link Layer Master) starts the exchange of parameters using the Pairing Request command. The responding device (a Link Layer Slave) responds with a Pairing Response command. The responding device can also initiate the pairing process using a Security Request command.

Authentication requirement flags are part of the pairing parameters. In Bluetooth 4.2 these flags are updated to add two new fields: Secure Connections (SC) and Keypress. Figure 13 shows the pairing parameters and authentication requirement flag in Bluetooth 4.1 and 4.2.

Figure 13. Pairing Parameters


Pairing Parameters in Bluetooth 4.2

Code (1 byte)
I/O Capability (1 byte)
OOB Data Flag (1 byte)
Authentication Requirement Flag (1 byte)
Max Encryption Key Size (1 byte)
Initiator Key Distribution (1 byte)
Responder Key Distribution (1 byte)
Bonding_Flags (2 bits)
MITM (1 bit)
SC (1 bit)
Keypress (1 bit)
Reserved (3 bits)

Pairing Parameters in Bluetooth 4.1

Code (1 byte)
I/O Capability (1 byte)
OOB Data Flag (1 byte)
Authentication Requirement Flag (1 byte)
Max Encryption Key Size (1 byte)
Initiator Key Distribution (1 byte)
Responder Key Distribution (1 byte)
Bonding_Flags (2 bits)
MITM (1 bit)
Reserved (5 bits)

The pairing method or Association Model for phase 2 is determined by the parameters exchanged during phase 1. There are four pairing methods or Association Models that can be used in phase 2:

- Just Works
- Numeric Comparison (Only for LE Secure Connections)
- Passkey Entry
- Out Of Band (OOB)

Table 4. Determining Association Model in LE Secure Connections

		Initiating Device			
		OOB Data Flag Set	OOB Data Flag Not Set	MITM Set	MITM Not Set
Responding Device	OOB Data Flag Set	Use OOB	Use OOB		
	OOB Data Flag Not Set	Use OOB	Check MITM		
	MITM Set			Use I/O Capabilities	Use I/O Capabilities
	MITM Not Set			Use I/O Capabilities	Use Just Works

Table 4 shows how the Association Model for phase 2 is determined in LE Secure Connections, based on the pairing parameters exchanged during phase 1. In LE Secure Connections, when both the initiating and the responding devices have Display and Yes/No I/O capabilities, or Display and Keyboard I/O capabilities, the Numeric Comparison Association Model is used. Read the Bluetooth Core Specification Version 4.2, Volume 3, Part H, Section 2 for details about how Association Models are determined based on I/O capabilities and when one or both of the BLE devices only support LE legacy pairing.

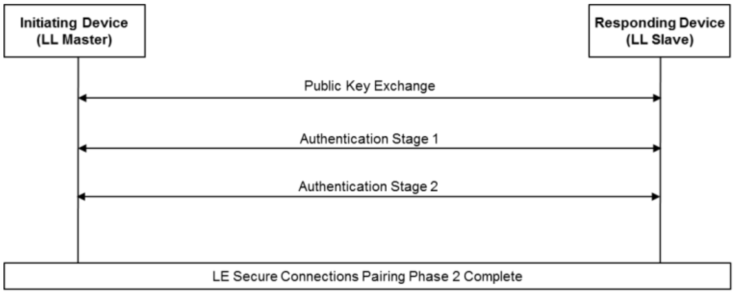
6.1.2 Pairing Phase 2

Phase 2 of the pairing process involves authentication for protection against Man-In-The-Middle (MITM) attacks and generation of the keys used to encrypt the BLE link.

In LE legacy pairing a Temporary Key (0 for the Just Works model, a 6-digit or 20-bit value for the Passkey Entry model, and 128 bits for the OOB model) is used to derive the key that encrypts the BLE link. The Temporary Key is the only random data not exchanged over the BLE link. For more details on LE legacy pairing read the Bluetooth Core Specification Version 4.2, Volume 3, Part H, section 2.3.5.

Phase 2 in LE Secure Connections consists of three stages as illustrated in Figure 14.

Figure 14. Phase 2 in LE Secure Communication Pairing


Initiating Device (LL Master)
Responding Device (LL Slave)
Public Key Exchange
Authentication Stage 1
Authentication Stage 2
LE Secure Connections Pairing Phase 2 Complete

In the public key exchange stage, the initiating device and the responding device exchange their public keys and start computing the Diffie-Hellman Key. The Diffie-Hellman key is generated using the Elliptic-Curve Diffie-Hellman function, P256. P256 takes a device-specific secret key and the public key of the peer device as an input. The Diffie-Hellman key is never exchanged over the air and provides 256 bits of randomness. Refer Protection against Passive Eavesdropping to understand how the ECDH algorithm enables both the initiating and responding devices to compute a common Diffie-Hellman Key without exchanging the secret key over the BLE link.

In authentication stage 1 the BLE devices authenticate each other to prevent Man-In-The-Middle (MITM) attacks. Authentication stage 1 is different for each association model. Refer Protection Against Man-In-The-Middle (MITM) Attacks to understand how authentication stage 1 for Numeric Comparison, Passkey Entry and OOB Association Models provides a high degree of protection against MITM attacks. Any failure in the authentication stage 1 results in the termination of pairing process.

In authentication stage 2 the BLE devices compute the Long Term Key (LTK) that will used for encrypting the link. The Diffie-Hellman key, Bluetooth device address, 128-bit random numbers generated in authentication stage 1 and the I/O capabilities of the device are used to generate the LTK.

For more details on authentication stage 1 and stage 2 in LE Secure Connections read the Bluetooth Core Specification Version 4.2, Volume 3, Part H, Section 2.3.5.6.

6.1.3 Pairing Phase 3

In pairing phase 3 the BLE link is encrypted using the STK generated in phase 2 in case of LE legacy pairing or using the LTK generated in phase 2 in case of LE Secure Connections pairing. The following keys are then distributed over the encrypted link:

1. Identity Resolving Key (IRK) - A 128-bit key used to generate and resolve random addresses
2. Connection Signature Resolving Key (CSRK) - A 128-bit key used to sign data and verify signatures on the receiving device
3. Long Term Key (LTK) - A 128-bit key used to generate the contributory session key for an encrypted connection. Refer to the Bluetooth Core Specification Version 4.2, Volume 6, Part B, Section 5.1.3 for details.
4. Encrypted Diversifier (EDIV) - A 16-bit stored value used to identify the LTK distributed during LE legacy pairing. A new EDIV is generated each time a unique LTK is distributed
5. Random Number (Rand) - A 64-bit stored valued used to identify the LTK distributed during LE legacy pairing. A new Rand is generated each time a unique LTK is distributed

LTK, EDIV and Rand are only distributed in the case of LE legacy pairing.

6.2 Benefits

6.2.1 Better Security

LE Secure Connections provide better security against MITM attacks and passive eavesdropping as compared to LE legacy pairing.

A passive eavesdropper is a third device that monitors the communication between two BLE devices. A passive eavesdropper needs to grabs the keys distributed over the BLE link to monitor encrypted communication. In LE Secure Connections, the 256-bit Diffie-Hellman key is never exchanged over-the-air for any Association Model. Therefore, the passive eavesdropper will not be able to calculate the Long Term Key (LTK) used for encryption and will not be able to monitor encrypted BLE communication. In Bluetooth 4.1, protection against passive eavesdropping is only available via the OOB association model and is weaker due to the usage of a 128-bit Temporary Key. Refer Protection against Passive Eavesdropping to understand how two BLE devices compute a common Diffie-Hellman key in LE Secure Connections.

MITM attack involves a third device that spoofs the peer device for each of the devices intended to be in connection. It can then alter the data exchanged between the two devices by acting as the peer device. The Numeric Comparison, Passkey Entry and OOB Association Models provide a high degree of protection against MITM attacks. Refer Protection Against Man-In-The-Middle (MITM) Attacks to understand how these Association Models provide protection against MITM attacks. Numeric Comparison enables faster pairing than Passkey Entry and does not require a separate communication link like OOB.

Table 5 summarizes the level of protection in LE legacy pairing and LE Secure Connections for different pairing methods.

Table 5. LE Legacy Pairing and LE Secure Connections Pairing Comparison

Feature	LE Legacy Pairing Methods	LE Secure Connections Pairing Methods
MITM protection	Passkey and OOB	Numeric Comparison, Passkey Entry, OOB
Passive eavesdropping protection	OOB	All

6.2.2 Secure Connection Only Mode

This mode forces strict pairing using only LE Secure Connections. If either of the devices has the strict pairing flag set, phase 2 of the pairing process will only happen if both sides support LE Secure Connections and the authentication requirements are met. This is very useful in applications where a very high level of security is needed.

6.3 Applications

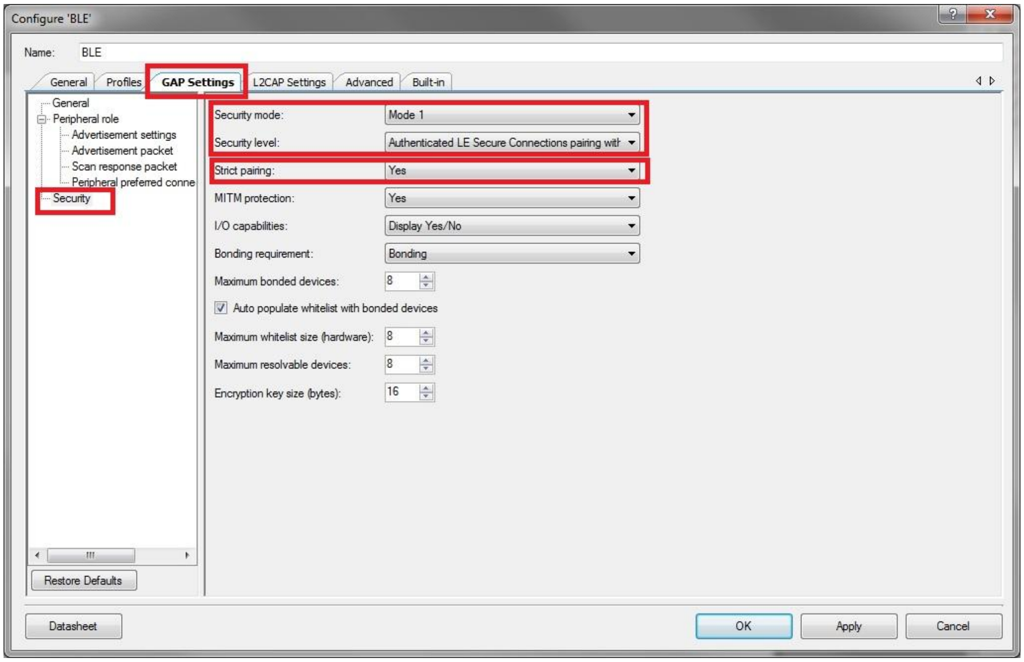
Door locks, banking cards, and others applications that require a high level of security can make use of Bluetooth 4.2’s enhanced security features.

6.4 Developing Applications Using LE Secure Connections with PSoC Creator

6.4.1 Component Configuration

The BLE Component provides an easy way to configure the security mode, security level, strict pairing requirements, MITM protection, I/O capabilities, bonding requirements, and the encryption key size. In order to use LE Secure Connections in your application, navigate to **Gap Settings** > **Security** and configure the parameters as **Security mode** to **Mode 1** and **Security level** to **Authenticated LE Secure Connections pairing with encryption**. The other parameters can be set per your application's requirements.

Figure 15. BLE Component Configuration: Security Settings



Security mode: Mode 1

Security level: Authenticated LE Secure Connections pairing with

Strict pairing: Yes

MITM protection: Yes

I/O capabilities: Display Yes/No

Bonding requirement: Bonding

Maximum bonded devices: 8

☑ Auto populate whitelist with bonded devices

Maximum whitelist size (hardware): 8

Maximum resolvable devices: 8

Encryption key size (bytes): 16

6.4.2 Application Handling

The Application firmware needs to manage the following tasks when using the LE Secure Connections feature.

- Generation of local public and private key pair
- Display of numbers for Numeric Comparison and display and entry of Passkey for Passkey Entry
- Transmission of Passkey Entry status
- Transmission of Passkey
- Transmission of acceptance or rejection of the numbers displayed for Numeric Comparison

Table 6 lists the BLE stack events, a description of each event, and the actions to be taken when using the LE Secure Connections feature.

Table 6. BLE Events and Corresponding Actions for LE Secure Connections

BLE Stack Event Name	Event Description	Event Handler Action
`CYBLE_EVT_GAP_NUMERIC_COMPARISON_REQUEST`	Provides the 6-digit number that needs to be displayed for Numeric Comparison	Displays the 6-digit number Sends accept or reject based on the number displayed on both the devices for Numeric Comparison
`CYBLE_EVT_GAP_KEYPRESS_NOTIFICATION`	Informs the Passkey Entry status on the peer side	Checks if the notification from the peer device matches the user input in this device Disconnects if there is a mismatch
`CYBLE_EVT_GAP_OOB_GENERATED_NOTIFICATION`	Informs that the OOB data generation is complete	Application specific action
`CYBLE_EVT_GAP_SMP_NEGOTIATED_AUTH_INFO`	Informs the negotiated pairing parameters	Application specific action

Table 7 provides the list of new APIs to be used in a LE Secure Connections-enabled application.

Table 7. New APIs for LE Secure Connections

API	Description
`CyBle_GapSetSecureConnectionsOnlyMode`	Enables or disables secure connection only mode
`CyBle_GapGenerateLocalP256Keys`	Generates the local public and private keys
`CyBle_GapAuthSendKeyPress`	Sends the keypress status
`CyBle_GapGenerateOobData`	Generates the OOB data

Read the BLE Component datasheet to learn more about the events and the APIs.

6.4.3 Example Project

The example project BLE_4.2_DataLength_Security_Privacy available with PSoC Creator uses LE Secure Connections. The example project can be accessed using **PSoC Creator** > **File** > **Code Example** with the filter item set to **BLE** as shown in Figure 10.

7 Link Layer (LL) Privacy

7.1 Introduction to Privacy

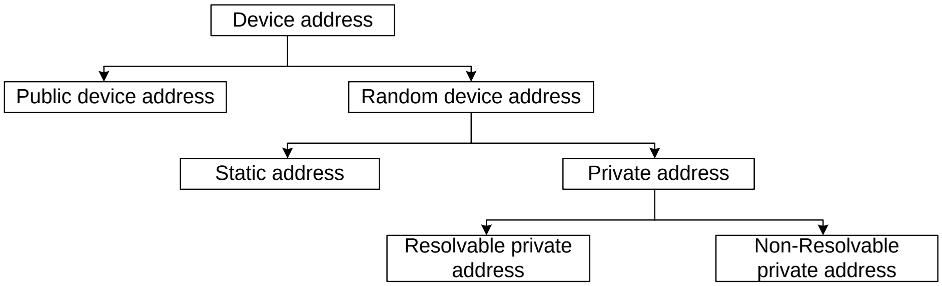
BLE devices are identified using a 48-bit device address. This device address is part of all the packets sent by the device in the advertising channels. A third device which listens on all three advertising channels can easily track the activities of a device by using its device address. Privacy is a feature that reduces the ability to track a BLE device by using a private address that is generated and changed at regular intervals.

7.2 Privacy Concepts

7.2.1 Bluetooth Address Types

Figure 16 shows the different address types possible for the BLE devices.

Figure 16. Bluetooth Address Types



The device address can be a Public Device Address or a Random Device Address. The Public Device Addresses are comprised of a 24-bit company ID (an Organizationally Unique Identifier or OUI based on the IEEE 802-2001 standard) and a 24-bit company-assigned number (unique for each device). There are two types of Random Device Addresses: Static Address and Private Address. The Static Address is a 48-bit randomly generated address with the two most significant bits of the 48-bit address set to 1. Public Device Addresses do not change over time. Static Addresses can change only when the device is power-cycled. A device using either of these addresses can be easily discovered and connected to by a peer device. Private Addresses changes at regular intervals to ensure that the BLE device cannot be tracked. A Non-Resolvable Private Address changes on every reconnection. Non-Resolvable Private Address cannot be resolved by the peer device and must be shared with the peer device during the preceding connection. Resolvable Private Addresses (RPA) can be changed at regular intervals, can be resolved and are used by Privacy enabled devices. In this application note we will discuss RPA in the context of the Privacy feature. Refer to the Bluetooth Core Specification Version 4.2 for more details on the different types of addresses.

Every Privacy-enabled BLE device has a unique address called the Identity Address. The Identity Address is the Public Address or Static Address of the BLE device. Every Privacy-enabled BLE device also has an Identity Resolving Key (IRK). The IRK is used by the BLE device to generate its RPA and is used by peer devices to resolve the RPA of the BLE device. Both the Identity Address and the IRK are exchanged during the third stage of the pairing process. Privacy-enabled BLE devices maintain a list that consists of the peer device's Identity Address, the local IRK used by the BLE device to generate its RPA, and the peer device's IRK used to resolve the peer device's RPA. This is called the Resolving List. Each entry in the Resolving List follows the format shown in Figure 17 below.

Figure 17. Resolving List

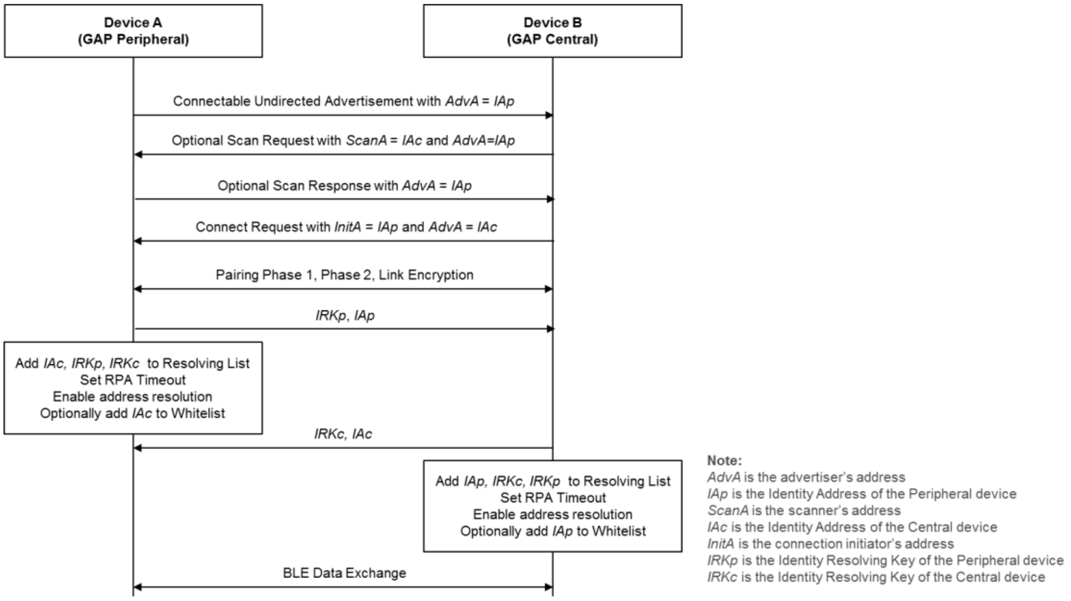
Identity address	Local IRK	Peer IRK

A Privacy-enabled BLE device periodically changes its RPA to avoid tracking. The BLE Stack configures the Link Layer with a value called RPA Timeout that specifies the time after which the Link Layer must generate a new RPA.

7.2.2 Privacy Flow

Figure 18 shows the initial connection establishment between two BLE devices that want to use the Privacy feature. The figure shows that one of the devices is in the Generic Access Profile (GAP) Peripheral role while the other device is in the GAP Central role. For details about GAP and various roles possible for a Bluetooth device read the Bluetooth Core Specification Version 4.2, Volume 3, Part C, Section 2.2.2.

Figure 18. Initial Connection Establishment Between Devices That Want To Use Privacy


Device A
(GAP Peripheral)
Device B
(GAP Central)
Connectable Undirected Advertisement with AdvA = IAp
Optional Scan Request with ScanA = IAc and AdvA=IAp
Optional Scan Response with AdvA = IAp
Connect Request with InitA = IAp and AdvA = IAc
Pairing Phase 1, Phase 2, Link Encryption
IRKp, IAp
Add IAc, IRKp, IRKc to Resolving List
Set RPA Timeout
Enable address resolution
Optionally add IAc to Whitelist
IRKc, IAc
Add IAp, IRKc, IRKp to Resolving List
Set RPA Timeout
Enable address resolution
Optionally add IAp to Whitelist
BLE Data Exchange

Note:
AdvA is the advertiser's address
IAp is the Identity Address of the Peripheral device
ScanA is the scanner's address
IAc is the Identity Address of the Central device
InitA is the connection initiator's address
IRKp is the Identity Resolving Key of the Peripheral device
IRKc is the Identity Resolving Key of the Central device

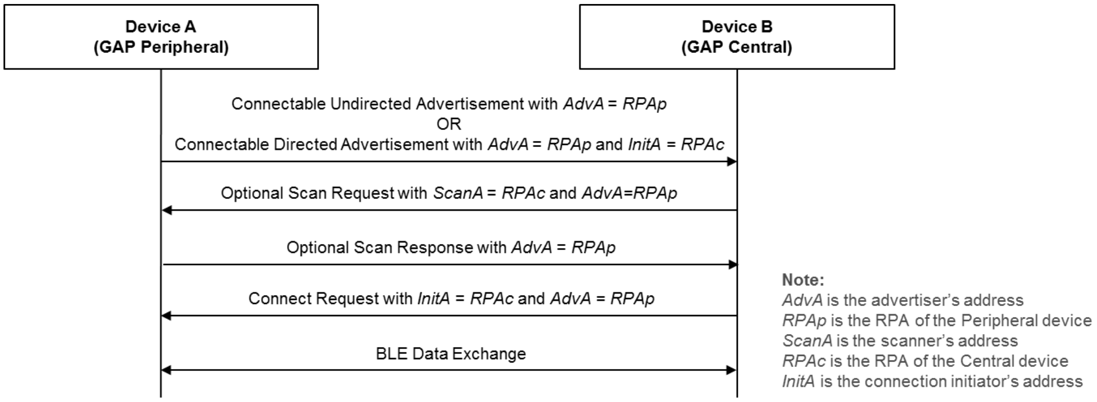
As seen above during initial connection establishment the GAP Peripheral and the GAP Central use the Identity Address (*IAp* for Peripheral and *IAc* for Central) in all packets sent over the advertisement channels. This is because initially both devices are not aware of the peer device's IRK will therefore not be able to resolve the peer's RPA. The packets sent over the advertisement channels include:

- Connectable Undirected Advertisement – These are advertisement packets that are not directed at a particular recipient i.e. at a particular GAP Central. These advertisements do not contain the address of the GAP Central and only carry the GAP Peripheral i.e. the advertiser's address.
- Optional Scan Request and Scan Response – Scan Request can be send by a GAP Central to request a peripheral for additional data before connecting to the peripheral, Scan Response is sent by the peripheral and includes the data requested for in the Scan Response. Both of these packets are optional.
- Connect Request – This is the request to initiate connection sent from the GAP Central to the GAP Peripheral

All a packets other than the above packets are sent on data channels and do not include a Bluetooth device address. The Connect Request packet is the last packet sent on the advertising channels. In addition to the Identity Addresses of the GAP Central and GAP Peripheral, the Connect Request packet also contains a random 32-bit number called the Access Address. The Access Address identifies a Link Layer connection between the two BLE devices. A new Access Address is generated during every connection establishment i.e. for every Connect Request. Every packet exchanged between the two BLE devices on the data channels use the Access Address to identify the connection between the two devices. This includes packets for pairing and data exchange.

In the third stage of pairing (refer 6.1 to understand the pairing process), the GAP Central and GAP Peripheral exchange IRKs and Identity Addresses as part of the payload in pairing related packets. The peer IRK and Identity Address are stored along with local IRK in the Resolving List by both devices. At this stage both devices have the information necessary to resolve RPAs therefore both devices set their RPA Timeouts and enable address resolution. Both devices may optionally add the Identity Address of the peer device to the Whitelist for device filtering. The Whitelist is a set of Bluetooth device addresses that the Link Layer of a BLE device uses to filter advertisers, scanners and connection initiators. Whitelist is used only when device filtering is enabled. The devices then exchange data on the BLE data channels. Because the devices now possess the information required to resolve peer RPAs in the Resolving List a reconnection procedure will use RPAs. Figure 19 shows a scenario where the devices were disconnected after initial connection establishment and are reconnecting.

Figure 19. Reconnection Using RPA


Device A
(GAP Peripheral)
Device B
(GAP Central)
Connectable Undirected Advertisement with AdvA = RPAp
OR
Connectable Directed Advertisement with AdvA = RPAp and InitA = RPAc
Optional Scan Request with ScanA = RPAc and AdvA=RPAp
Optional Scan Response with AdvA = RPAp
Connect Request with InitA = RPAc and AdvA = RPAp
BLE Data Exchange

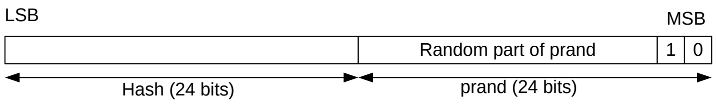
Note:
AdvA is the advertiser's address
RPAp is the RPA of the Peripheral device
ScanA is the scanner's address
RPAc is the RPA of the Central device
InitA is the connection initiator's address

As seen above the devices now use RPA in all the packets sent over the advertisement channels. Due to this the communication between the devices on the advertisement channels cannot be tracked by any third device. Note that the GAP Peripheral can use Connectable Undirected Advertisements or Connectable Directed Advertisements for reconnection. 7.2.5 explains why Privacy-enabled devices can only support Connectable Directed Advertisements with Bluetooth 4.2. Scan Request and Scan Response packets can only be used when the GAP Peripheral uses Connectable Undirected Advertisements. When the GAP Peripheral uses Connectable Directed Advertisements the GAP Central can only respond with a Connect Request. Note that Privacy prevents tracking of the BLE devices via communication on advertisement channels. Data channel packets are less susceptible to tracking because they use the random Access Address that is generated for every connection establishment. Privacy does not affect the data channel packets.

7.2.3 Resolvable Private Address (RPA) Generation

This section explains how Privacy-enabled BLE devices generate the RPA. The format of the Resolvable Private Address is described in the Bluetooth specification and is shown in Figure 20.

Figure 20. Resolvable Private Address Format



The device generates a 24-bit number (22 bits are random and 2 bits are fixed), referred to as *prand*. The device uses *prand* to generate a 24-bit *hash* using the hash function:

$hash = e(IRK_{local}, padding \mid prand)$, truncated to 24 bits where:

IRK_{local} = 128-bit local IRK

Padding = 104-bit zero padding to extend *prand* to 128 bits

prand = 24-bit random number (22 bits are random and the 2 most significant bits are set to 0b10)

The security function 'e' is a 128-bit AES encryption function implemented in the Link Layer hardware in the PSoC 4/ PRoC BLE devices. It generates a 128-bit *encryptedData* from a 128-bit *key* and a 128-bit *plaintextData*, as defined in FIPS-1971:

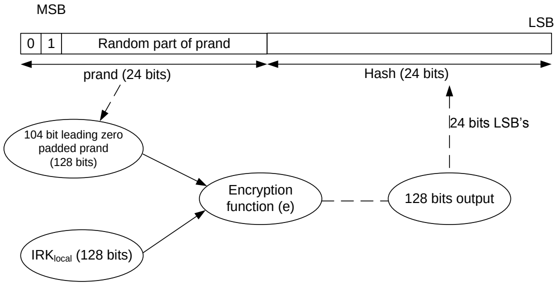
encryptedData = e(key, plaintextData)

The 24-bit *hash* and the 24-bit *prand* are concatenated to generate the random address *(randomAddress)* in the following manner:

randomAddress = hash | prand

The Resolvable Private Address generation is illustrated in Figure 21.

Figure 21. Resolvable Private Address Generation



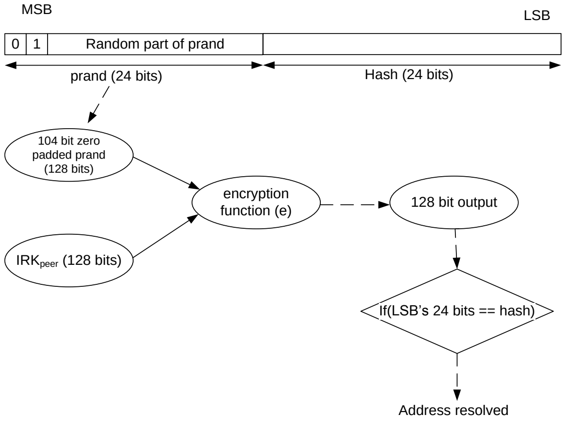
7.2.4 Resolvable Private Address Resolution

This section explains how Privacy-enabled BLE devices resolve the peer device's RPA. The RPA received from a peer device consists of a 24-bit random part (*prand*) and a 24-bit hash part (*hash*) as described in section 7.2.3. To resolve or decode the address, the least significant 24 bits of the RPA are extracted into the *hash* and the most significant 24 bits of the address are extracted into the *prand* of the peer device. A *localHash* value is then generated using the hash function described earlier with the input parameters set to the IRK received from the peer device and the input parameter *prand* value extracted from the RPA.

localHash = *e*(IRK_{peer}, *padding* | *prand*), truncated to 24 bits

The *localHash* value is compared with the *hash* value extracted from RPA. If the *localHash* value matches the extracted *hash* value, then the identity of the peer device is said to be resolved. Address resolution is shown in Figure 22.

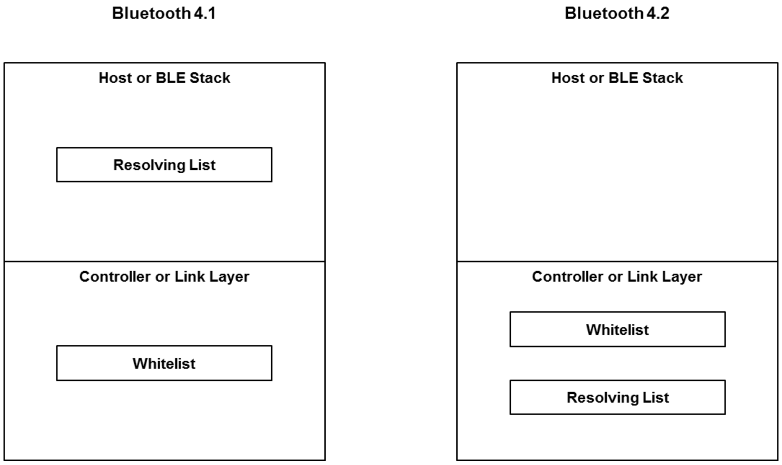
Figure 22. Resolvable Private Address Resolution



7.2.5 Privacy 1.1 vs. Privacy 1.2

The Privacy feature in Bluetooth 4.1 was called Privacy 1.1. In Bluetooth 4.2 the Privacy feature is called Privacy 1.2 or Link Layer Privacy. This is because address resolution i.e. RPA resolution is handled by the Link Layer in Bluetooth 4.2. As shown in Figure 23 both the Resolving List and the Whitelist are part of the Link Layer in Bluetooth 4.2.

Figure 23. Privacy 1.1 in Bluetooth 4.1 vs. Privacy 1.2 in Bluetooth 4.2



Connectable Directed Advertisements contain the RPA of both the advertiser and the recipient. With Link Layer Privacy in Bluetooth 4.2, the Link Layer can resolve RPAs using the Resolving List. The Link Layer therefore understands whether it is the intended recipient of the Connectable Directed Advertisement. Acceptable values for RPA Timeout are also different in Bluetooth 4.1 and Bluetooth 4.2. RPA Timeout can be any value between 1 second and 11.5 hours with a default value of 900 seconds in Bluetooth 4.2. In Bluetooth 4.1 RPA Timeout is fixed at 15 minutes.

7.3 Benefits

7.3.1 Faster Reconnection

Section 7.2.5 explained how Privacy-enabled devices can use Connectable Directed Advertising for reconnection. In Bluetooth 4.1 device filtering with Privacy was also not possible. The Link Layer did not have access to the Resolving List. The Link Layer was therefore unable to resolve RPAs to Identity Addresses and use Whitelist to filter Identity Addresses. In Bluetooth 4.2 the Resolving List is part of the Link Layer as shown in Figure 23. The Link Layer can therefore resolve RPAs to Identity Addresses using the Resolving List and then filter Identity Addresses using the Whitelist. Device filtering together with Connectable Directed Advertisements in Bluetooth 4.2 enable faster reconnection for Privacy-enabled BLE devices.

7.3.2 Power-Efficient GAP Central Devices

For Privacy-enabled GAP Central devices that are using device filtering, the Link Layer only sends Directed Advertisements and Scan Responses from peer devices that are in the Resolving List and the Whitelist to the BLE Stack. This reduces the processing required in the BLE Stack. BLE devices like Cypress's PSoC 4 BLE and PRoC BLE implement the BLE Stack as firmware running on an MCU and the Link Layer as a hardware block. Therefore reduction in the processing required to be done in the BLE Stack, also reduces MCU active time and consequently reduces system power consumption.

7.3.3 Power-Efficient GAP Peripheral Devices

For Privacy-enabled GAP Peripheral devices that are using device filtering, the Link Layer only sends Scan Requests and Connection Requests from peer devices that are in the Resolving List and the Whitelist to the BLE Stack. This reduces the processing required in the BLE Stack. BLE devices like Cypress's PSoC 4 BLE and PRoC BLE devices implement the BLE Stack as firmware running on an MCU and the Link Layer as a hardware block. Therefore reduction in the processing required to be done in the BLE Stack, also reduces MCU active time and consequently reduces system power consumption.

7.3.4 More Private than Bluetooth 4.1

As mentioned in Section 7.2.5, Privacy in Bluetooth 4.1 uses a fixed 15-minute RPA timeout while in Bluetooth 4.2 the RPA Timeout can be as low as 1 second. This results in more frequent RPA changes and makes the BLE device more difficult to track in Bluetooth 4.2.

7.4 Applications

There are many applications that benefit from the Link Layer Privacy feature.

With Privacy users can control the visibility of their wearable devices in public environments, like a gymnasium. Advertisements from a Privacy-enabled wearable, as it tries to connect to the user's smartphone or to the gymnasium equipment, cannot be tracked by a third device. This prevents users from being tracked through their wearables in public environments. Most wearables are GAP Peripherals and do not use the Privacy feature in Bluetooth 4.1 due to higher power consumption. With Link Layer Privacy in Bluetooth 4.2, wearables can use Privacy due to the reduction in power consumption described in 7.3.3.

Retail applications also benefit from Privacy. BLE beacons are used by service providers in retail environments to send offers via advertisements to a user's smartphone. With Link Layer Privacy smartphones can use device filtering (as described in section 7.3.1) to only receive offers from preferred service providers and ignore other service providers.

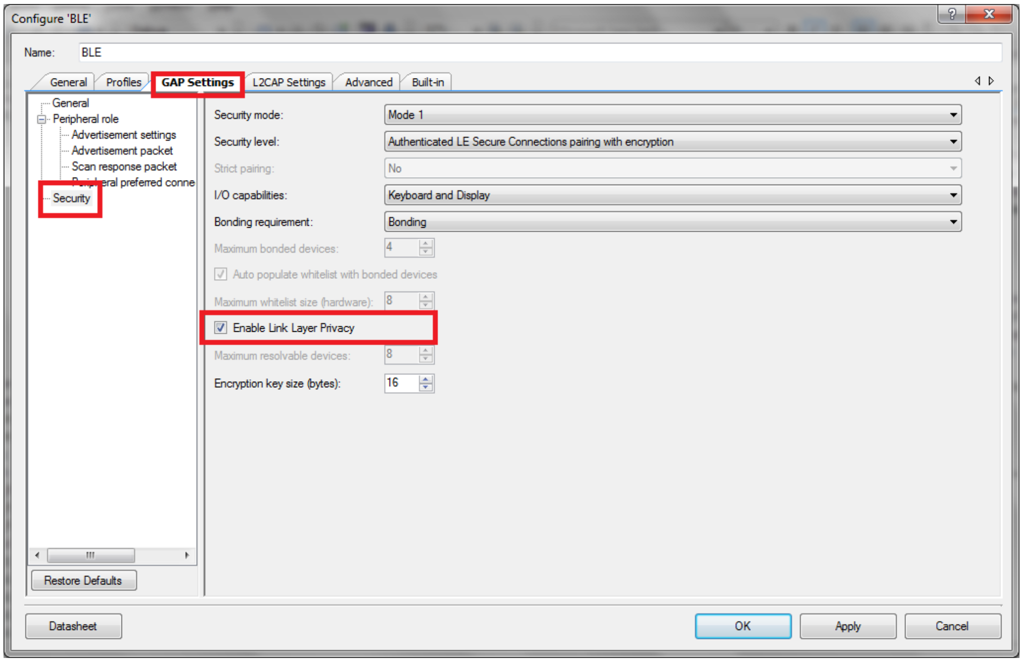
Link Layer Privacy makes it harder to track a user's smartphone in public environments due to frequently changing RPA. Smartphones typically act as GAP Central devices. Link Layer Privacy also reduces the power-consumption due to BLE communication for smartphones (as described in section 7.3.2)

7.5 Developing Applications Using Link Layer Privacy with PSoC Creator

7.5.1 Component Configuration

The Link Layer Privacy feature and the related APIs can be enabled or disabled using the **Enable Link Layer Privacy** option in the BLE Component configuration window under **GAP Settings** > **Security** as shown in Figure 24.

Figure 24. Enabling/Disabling Link Layer Privacy



Configure 'BLE'

Name: BLE

Security mode: Mode 1

Security level: Authenticated LE Secure Connections pairing with encryption

Strict pairing: No

I/O capabilities: Keyboard and Display

Bonding requirement: Bonding

Maximum bonded devices: 4

☑ Auto populate whitelist with bonded devices

Maximum whitelist size (hardware): 8

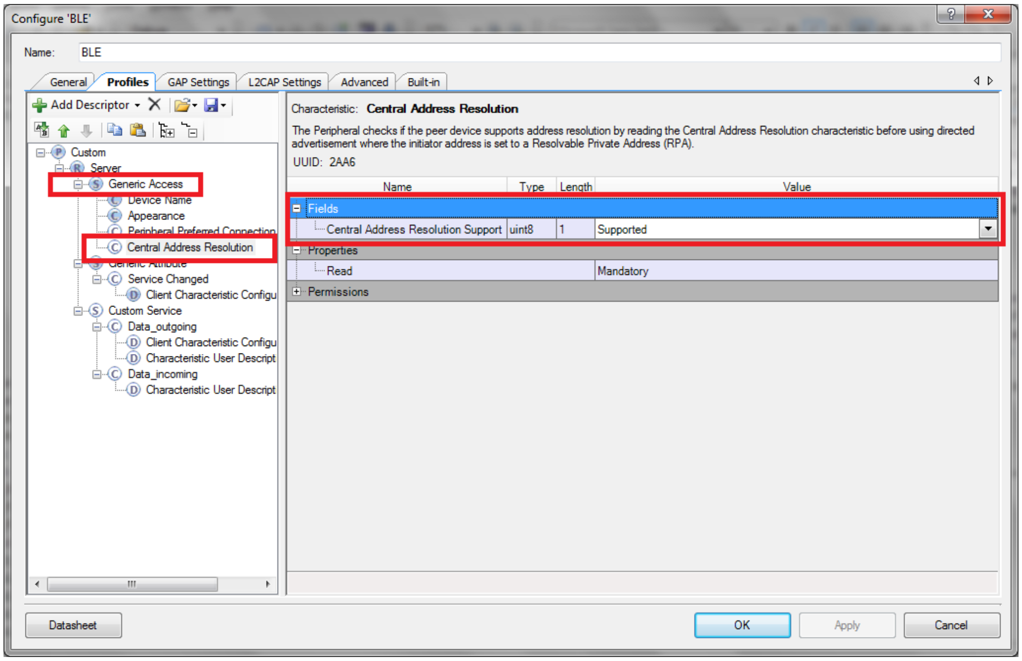
☑ Enable Link Layer Privacy

Maximum resolvable devices: 8

Encryption key size (bytes): 16

The Central Address Resolution characteristic in the GAP service is read by the GAP Peripheral device after initial connection establishment. The value of this characteristic tells the GAP Peripheral whether the GAP Central supports address resolution in the Link Layer. If address resolution is supported in then Link Layer then the GAP Peripheral can use Connectable Directed Advertisements for reconnection with the GAP Central. The value of the Central Address Resolution characteristic can be set using the drop-down menu under **Profiles** > **Generic Access** > **Central Address Resolution** as shown in Figure 25. A device supporting Link Layer Privacy should set the value of this characteristic to **Supported** to inform the peer device that Link Layer Privacy is supported.

Figure 25. Setting Central Address Resolution Characteristic Value



Configure 'BLE'

Name: BLE

Characteristic: **Central Address Resolution**

The Peripheral checks if the peer device supports address resolution by reading the Central Address Resolution characteristic before using directed advertisement where the initiator address is set to a Resolvable Private Address (RPA).

UUID: 2AA6

Name	Type	Length	Value
Fields			
Central Address Resolution Support	uint8	1	Supported
Properties			
Read			Mandatory
Permissions			

7.5.2 Application Handling

The application firmware needs to manage the following tasks in a privacy-enabled device firmware:

- Enabling/disabling address resolution
- Setting RPA Timeout
- Adding/removing devices to/from the Resolving List
- Setting proper Whitelist filter policy

Table 8 lists the BLE stack events, a description of the event, and the actions to be taken in the event handler.

Table 8. BLE Events and Actions for LL Privacy

BLE Stack Event Name	Event Description	Event Handler Action
`CYBLE_EVT_STACK_ON`	BLE stack initialization completed successfully	Populate the Resolving List Enable address resolution Set RPA Timeout Set your Whitelist filter policy
`CYBLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT`	Exchange of security keys with the peer device is complete	Copy the peer device IRK Update the Resolving List Update Whitelist filter policy
`CYBLE_EVT_GAPC_DIRECT_ADV_REPORT`	Directed Advertisement received after address resolution	Connect to the device

Table 9 provides a list of new APIs to be used in a LL Privacy-enabled application.

Table 9. New APIs for LL Privacy-Enabled Applications

API	Description
CyBle_GapAddDeviceToResolvingList	Adds a device to the Resolving List
CyBle_GapRemoveDeviceFromResolvingList	Removes a device from the Resolving List
CyBle_GapSetAddressResolutionEnable	Enables address resolution in the controller
CyBle_GapSetResolvablePvtAddressTimeOut	Sets the RPA Timeout
CyBle_GapReadResolvingList	Reads the current Resolving List
CyBle_GapReadPeerResolvableAddress	Reads the current RPA address of the peer device
CyBle_GapReadLocalResolvableAddress	Reads the current RPA address of the local device
CyBle_GapGetDevSecurityKeyInfo	Gets the local IRK value

7.5.3 Reading Central Address Resolution Characteristic

As mentioned in section 7.5.1, a Peripheral device should read the Central Address Resolution characteristic and send Connectable Directed Advertisements only when the Privacy-enabled Central device supports Link Layer Privacy. To read the Central Address Resolution characteristic, the CyBle_GattcReadUsingCharacteristicUuid API can be used with appropriate parameters as shown below.

```
CYBLE_GATTC_READ_BY_TYPE_REQ_T read_by_type_req;

read_by_type_req.range.startHandle = CYBLE_GATT_ATTR_HANDLE_START_RANGE;

read_by_type_req.range.endHandle = CYBLE_GATT_ATTR_HANDLE_END_RANGE;

read_by_type_req.uuid.uuid16 = CYBLE_UUID_CHAR_CENTRAL_ADDRESS_RESOLUTION;

read_by_type_req.uuidFormat = CYBLE_GATT_16_BIT_UUID_FORMAT;

CyBle_GattcReadUsingCharacteristicUuid(cyBle_connHandle,&read_by_type_req);
```

If the Central device has the Central Address Resolution characteristic, the value of the characteristic along with the attribute handle will be passed to the application in the CYBLE_EVT_GATTC_READ_BY_TYPE_RSP event. The application can check the value of the characteristic and decide whether to use Connectable Directed Advertisements for the next reconnection.

7.5.4 Adding/Removing Device to/from Resolving List

A new BLE device can be added to the Resolving List shown in Figure 17 using the CyBle_GapAddDeviceToResolvingList API with the local IRK, the peer IRK, and the peer identity address as parameters.

```
CYBLE_GAP_RESOLVING_DEVICE_INFO_T rpaInfo;

memcpy(&rpaInfo.bdAddr,&peer_ID_Addr[1],CYBLE_GAP_BD_ADDR_SIZE);
rpaInfo.type = smp_key->idAddrInfo[0];
memcpy(rpaInfo.localIrk,local_irk,CYBLE_GAP_SMP_IRK_SIZE);
memcpy(rpaInfo.peerIrk,smp_key->irkInfo,CYBLE_GAP_SMP_IRK_SIZE);

/* Add device to resolving list */
CyBle_GapAddDeviceToResolvingList(&rpaInfo);
```

Peer IRK and peer identity address are passed to the application in the CYBLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT event. The application must save these values. Local IRK can be obtained using the CyBle_GapGetDevSecurityKeyInfo API.

A device can be removed from the Resolving List by calling the CyBle_GapRemoveDeviceFromResolvingList API with the identity address of the device.

Read the BLE Component datasheet for more details about these events and APIs.

7.5.5 Example Project

For an example project using Link Layer Privacy, see the example project **BLE_4.2_DataLength_Security_Privacy** available with PSoC Creator. The example project can be accessed using **PSoC Creator** > **File** > **Code Example** with the filter item set to **BLE** as shown in Figure 10.

8 Summary

This application note explained the new features introduced in Bluetooth LE 4.2 Specification, their benefits, and how to develop applications using these new features.

9 Related Application Notes

AN94020 - Getting Started with PRoC™ BLE

AN91267 - Getting Started with PSoC® 4 BLE

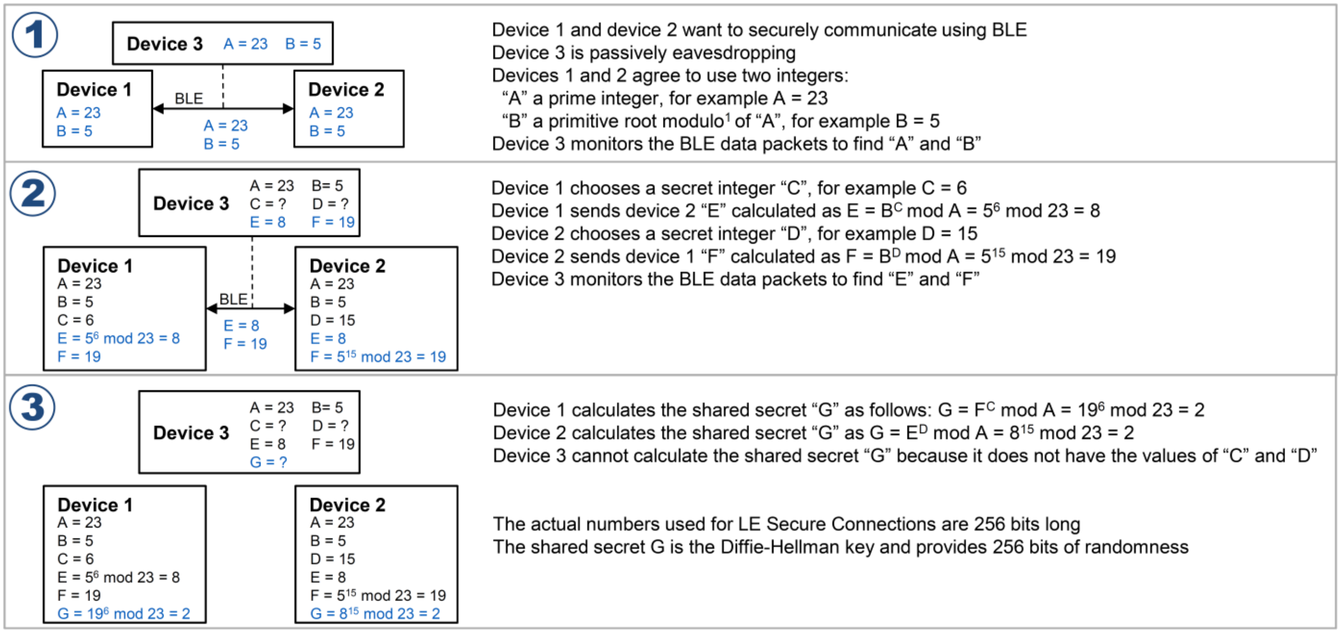
AN97060 - PSoC® 4 BLE and PRoC™ BLE - Over-The-Air (OTA) Device Firmware Upgrade (DFU) Guide

AN91184 - PSoC 4 BLE - Designing BLE Applications

A Protection against Passive Eavesdropping

Elliptic Curve Diffie-Hellman (ECDH) algorithm enables two BLE devices to establish a shared secret when using LE Secure Connections-based pairing. The shared secret or the Diffie-Hellman Key is used in Phase 2 (Authentication Stage 2) of the pairing process to generate the Long Term Key (LTK) that encrypts the BLE link. The Diffie-Hellman Key provides 256 bits of randomness and is not exchanged over the air as shown in Figure 26.

Figure 26. Establishing a Shared Secret Using ECDH



Device 1 and device 2 want to securely communicate using BLE
Device 3 is passively eavesdropping
Devices 1 and 2 agree to use two integers:
"A" a prime integer, for example A = 23
"B" a primitive root modulo[1] of "A", for example B = 5
Device 3 monitors the BLE data packets to find "A" and "B"

Device 1 chooses a secret integer "C", for example C = 6
Device 1 sends device 2 "E" calculated as $E = B^C \bmod A = 5^6 \bmod 23 = 8$
Device 2 chooses a secret integer "D", for example D = 15
Device 2 sends device 1 "F" calculated as $F = B^D \bmod A = 5^{15} \bmod 23 = 19$
Device 3 monitors the BLE data packets to find "E" and "F"

Device 1 calculates the shared secret "G" as follows: $G = F^C \bmod A = 19^6 \bmod 23 = 2$
Device 2 calculates the shared secret "G" as $G = E^D \bmod A = 8^{15} \bmod 23 = 2$
Device 3 cannot calculate the shared secret "G" because it does not have the values of "C" and "D"

The actual numbers used for LE Secure Connections are 256 bits long
The shared secret G is the Diffie-Hellman key and provides 256 bits of randomness

[1] To understand "primitive root modulo" refer to *Number Theory* by George E. Andrews (ISBN-10:0486682528)

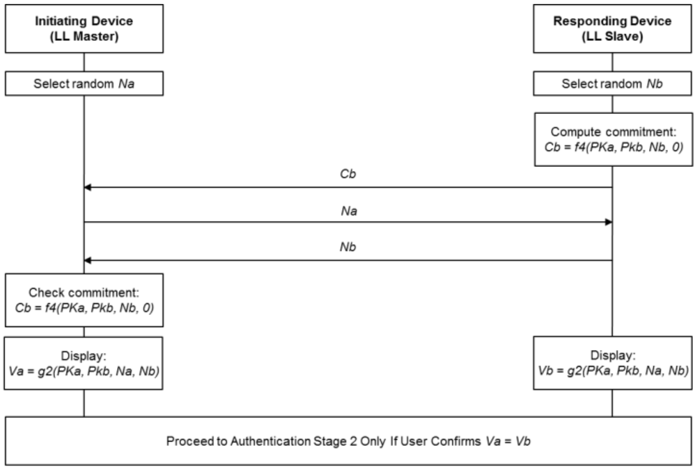
B Protection Against Man-In-The-Middle (MITM) Attacks

The sections below explain how Numeric Comparison, Passkey Entry and Out-Of-Band (OOB) Association Models in LE Secure Connections-based pairing protect against MITM attacks.

B.1 Numeric Comparison Association Model

Figure 27 shows authentication stage 1 of the pairing process when using the Numeric Comparison Association Model for LE Secure Connections.

Figure 27. Authentication Stage 1 for Numeric Comparison


Initiating Device
(LL Master)
Responding Device
(LL Slave)
Select random *Na*
Select random *Nb*
Compute commitment:
Cb = f4(PKa, Pkb, Nb, 0)
Cb
Na
Nb
Check commitment:
Cb = f4(PKa, Pkb, Nb, 0)
Display:
Va = g2(PKa, Pkb, Na, Nb)
Display:
Vb = g2(PKa, Pkb, Na, Nb)
Proceed to Authentication Stage 2 Only If User Confirms *Va = Vb*

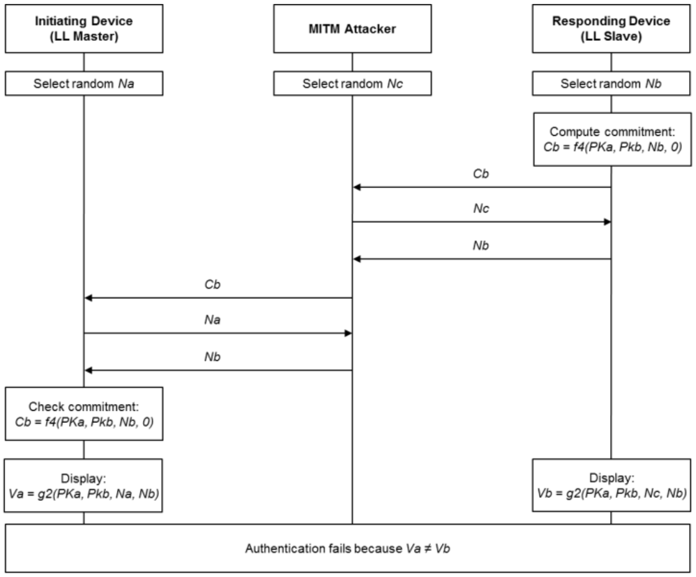
Note:
Na is a 128-bit random number generated by the initiating device
Nb is a 128-bit random number generated by the responding device
PKa and *PKb* are the public keys of the initiating and responding device. These are exchanged during the public key exchange stage of phase 2
Cb is the commitment value calculated by the responding device and checked by the initiating device
f4 and *g2* are AES-CMAC based encryption functions
Va and *Vb* are 6 digit numbers that are displayed to the user by the initiating and responding devices respectively

The Numeric Comparison Association Model uses the following authentication measures to protect against a MITM attack:

- The responding device must share a Commitment Value (*Cb*) calculated using the responding device’s random number (*Nb*) and the public keys of both BLE devices, before it receives the initiating device’s random number (*Na*)
- The initiating device must share its random number (*Na*) before it receives the responding device’s random number (*Nb*)
- The initiating device must check the Commitment Value (*Cb*) after it receives the responding device’s random number (*Nb*)

Figure 28 shows the scenario where the MITM attacker first exchanges random numbers with the responding device and then with the initiating device.

Figure 28. MITM Attacker Exchanging Keys with Responding Device First and Then with Initiating Device

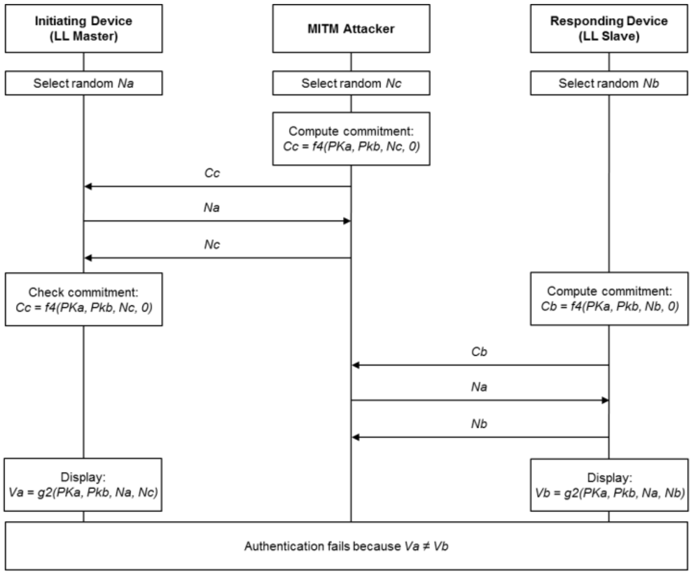

Initiating Device (LL Master)
MITM Attacker
Responding Device (LL Slave)
Select random *Na*
Select random *Nc*
Select random *Nb*
Compute commitment: *Cb = f4(PKa, Pkb, Nb, 0)*
Cb
Nc
Nb
Cb
Na
Nb
Check commitment: *Cb = f4(PKa, Pkb, Nb, 0)*
Display: *Va = g2(PKa, Pkb, Na, Nb)*
Display: *Vb = g2(PKa, Pkb, Nc, Nb)*
Authentication fails because *Va ≠ Vb*

Note:
Na is a 128-bit random number generated by the initiating device
Nb is a 128-bit random number generated by the responding device
Nc is the 128-bit random number of the MITM attacker
PKa and *PKb* are the public keys of the initiating and responding device. These are exchanged during the public key exchange stage of phase 2
Cb is the commitment value calculated by the responding device and checked by the initiating device
f4 and *g2* are AES-CMAC based encryption functions
Va and *Vb* are 6 digit numbers that are displayed to the user by the initiating and responding devices respectively

As evident from Figure 28 the attacker has to share its random number (*Nc*) with the responding device before getting access to the responding device's random number (*Nb*). This leads to different 6-digit values being displayed by the BLE devices and results in authentication failure when the user checks the displayed values.

Figure 29 shows the scenario where the MITM attacker first exchanges random numbers with the initiating device and then with the responding device.

Figure 29. MITM Attacker Exchanging Keys with Initiating Device First and Then with Responding Device


Initiating Device (LL Master)
MITM Attacker
Responding Device (LL Slave)
Select random *Na*
Select random *Nc*
Select random *Nb*
Compute commitment: *Cc = f4(PKa, Pkb, Nc, 0)*
Cc
Na
Nc
Check commitment: *Cc = f4(PKa, Pkb, Nc, 0)*
Compute commitment: *Cb = f4(PKa, Pkb, Nb, 0)*
Cb
Na
Nb
Display: *Va = g2(PKa, Pkb, Na, Nc)*
Display: *Vb = g2(PKa, Pkb, Na, Nb)*
Authentication fails because *Va ≠ Vb*

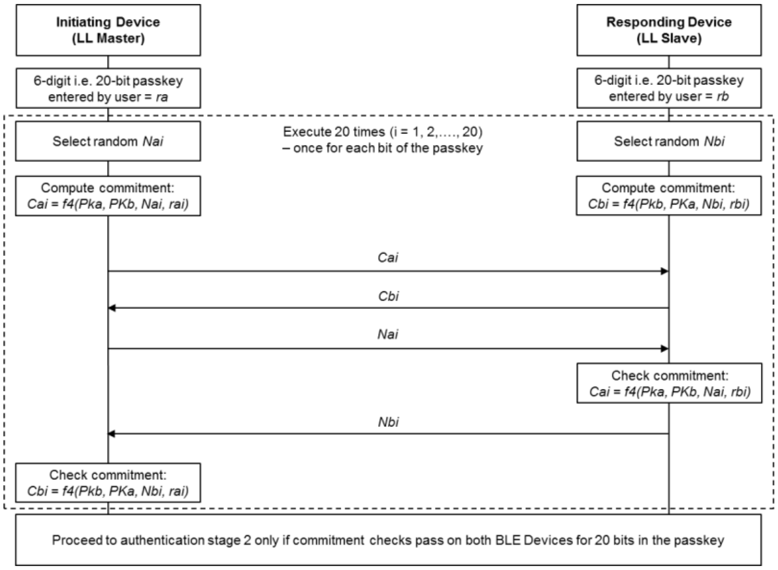
Note:
Na is a 128-bit random number generated by the initiating device
Nb is a 128-bit random number generated by the responding device
Nc is the 128-bit random number of the MITM attacker
PKa and *PKb* are the public keys of the initiating and responding device. These are exchanged during the public key exchange stage of phase 2
Cc and *Cb* are the commitment values of the attacker and the responding device respectively
f4 and *g2* are AES-CMAC based encryption functions
Va and *Vb* are 6 digit numbers that are displayed to the user by the initiating and responding devices respectively

As evident from Figure 29 the attacker has to share its Commitment Value (*Cc*) with the initiating device before it gets access to the initiating device's random number (*Na*). The initiating device checks the Commitment Value and thereby prevents the attacker from changing its random number. This leads to different 6-digit values being displayed by the BLE devices and results in authentication failure when the user checks the displayed values.

B.2 Passkey Entry Association Model

Figure 30 shows authentication stage 1 of the pairing process when using the Passkey Entry Association Model for LE Secure Connections.

Figure 30. Authentication Stage 1 for Numeric Comparison


Initiating Device
(LL Master)
Responding Device
(LL Slave)
6-digit i.e. 20-bit passkey entered by user = *ra*
6-digit i.e. 20-bit passkey entered by user = *rb*
Select random *Nai*
Execute 20 times (i = 1, 2,…., 20) – once for each bit of the passkey
Select random *Nbi*
Compute commitment: *Cai = f4(Pka, PKb, Nai, rai)*
Compute commitment: *Cbi = f4(Pkb, PKa, Nbi, rbi)*
Cai
Cbi
Nai
Check commitment: *Cai = f4(Pka, PKb, Nai, rbi)*
Nbi
Check commitment: *Cbi = f4(Pkb, PKa, Nbi, rai)*
Proceed to authentication stage 2 only if commitment checks pass on both BLE Devices for 20 bits in the passkey

Note:
ra=rb is the passkey entered by the user on both BLE devices
Nai and *Nbi* are 128-bit random numbers. New 128-bit random numbers are generated for each bit of the passkey (*Nai = Na1, Na2….Na20* and *Nbi = Nb1, Nb2…..Nb20*)
Rai and *rbi* represent the *i*th bit of the passkey where *i= 1,2….20)*
PKa and *PKb* are the public keys of the initiating and responding device. These are exchanged during the public key exchange stage of phase 2
Cbi is the commitment value calculated by the responding device and checked by the initiating device
Cai is the commitment value calculated by the initiating device and checked by the respondingdevice
f4 is an AES-CMAC based encryption function

The Passkey Entry Association Model has the following authentication measures to protect against an MITM attack:

1. Both BLE devices receive a common 6-digit or 20-bit Passkey (*ra* and *rb*) as user input
2. Steps 3 through 7 are repeated for each bit in the Passkey (i.e. for *i = 1, 2,….20*)
3. Both BLE devices select 128-bit random numbers (*Nai* and *Nbi*)
4. The initiating device shares a Commitment Value (*Cai*) calculated using the public keys of both BLE devices (*PKa* and *PKb*), a 128-bit random number (*Nai*) and the i^{th} bit of the Passkey (*rai*)
5. The responding device shares a Commitment Value (*Cbi*) calculated using the public keys of both BLE devices (*PKa* and *PKb*), a 128-bit random number (*Nbi*) and the i^{th} bit of the Passkey (*rbi*)
6. The initiating device shares its 128-bit random number (*Nai*) with the responding device
7. The responding device checks the Commitment Value (*Cai*) received from the initiating device using the initiating device's 128-bit random number (*Nai*). If the check passes, the responding device shares its 128-bit random number (*Nbi*) with the initiating device
8. The initiating device checks the Commitment Value (*Cbi*) received from the responding device using the responding device's 128-bit random number (*Nbi*). If the check passes the initiating device continues the pairing process

At the heart of the Passkey Entry Association Model is the gradual or bit-by-bit disclosure of the Passkey. An MITM attacker that engages with both the initiating and responding devices would only be able to get access to 2 bits of the Passkey before the BLE devices detect a failure in Commitment Value checks and abort the pairing process. The disadvantage of Passkey Entry Association Model when compared to Numeric Comparison Association Model is the time taken for the pairing process due to the 20 iterations of the authentication measures (once for each bit of the Passkey).

B.3 Out-Of-Band (OOB) Association Model

If the OOB communication is resistant to MITM attacks them the OOB Association Model is also resistant to MITM attacks. The key disadvantage of the OOB Association Model is that both BLE devices must have an OOB interface in addition to the BLE interface for comm. The OOB interface adds to the cost of the BLE device. For details on the OOB Association Model refer to the Bluetooth Core Specification Version 4.2, Volume 3, Part H, Section 2.3.5.6.

Document History

Document Title: AN99209 - PSoC® 4 BLE and PRoC™ BLE: Bluetooth LE 4.2 Features

Document Number: 001-99209

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5059016	ANKC	06/01/2016	New application note.
*A	5700414	AESATP12	04/19/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.