ModusToolbox™

Secure Over-the-Air Firmware Upgrade

Document Number. 002-16561 Rev. *D

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Contents

1 Introduction

The SOTAFU SDK feature uses elliptic curve cryptography to implement a secure method to upgrade the firmware of devices that are based on Cypress WICED Bluetooth chips. The Elliptic Curve Digital Signature Algorithm (ECDSA) is used to sign the image which is verified by the firmware after the download and before the new image is allowed to be executed on the chip.

The device memory is split into active and passive partitions. The new firmware image is downloaded to the passive partition and then verified. If the verification is successful, then the partitions are swapped and the new image is executed during the next device startup. This two-step process ensures that interrupted or unverified downloads do not cause interruption of the service.

The SOTAFU also verifies that the image for the correct product is has been downloaded. The application image embeds four bytes of Product Version information which includes two bytes of a product ID, one byte with the major version and one byte with the minor version. The Product Version information is verified during the upgrade.

The digital signature is prepared by calculating an SHA256 hash of the firmware image file and then signed using the ECDSA algorithm with secp256r1 curve.

Verification of the new image will fail unless the following conditions are met:

- ECDSA verification must pass. This guarantees the image integrity and validates that the image was created by the authorized party which had access to the private key.
- The product ID which is included in the new verified image must match that of the existing application. This requirement prevents the use of a correctly signed application built for a different product.
- The major version number which is included in the new verified image must be the same or greater than that of the existing application. The major version should be increased when security vulnerabilities are fixed. This requirement prevents replacing the application with a version that has known security problems.

Note: The minor version number can be higher or lower than that of the current application to allow the customer to upgrade or downgrade the application without introducing the security risk.

If verification fails, the new image will not be activated and the existing application will continue to run without interruption.

Details of the OTA Firmware Upgrade protocol are discussed in the WICED Firmware Upgrade Library document *[1]* included in the WICED Bluetooth SDK.

2 IoT Resources

Cypress provides a wealth of data at http://www.cypress.com/internet-things-iot to help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. Cypress provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. Customers can acquire technical documentation and software from the Cypress Support Community website (http://community.cypress.com/).

3 Preparing the Secure Firmware Image

The WICED Bluetooth SDK provides tools to create a signed firmware image. Some changes are also required to the application code to support the SOTAFU feature. Detailed information is provided in the following sections. Below is a summary of the required steps to make the application SOTAFU compatible.

1. Create a public and private key pair (see Section 4). Save the private key in a safe location. When an updated version of the application is developed, it should be signed with the same private key.
2. Modify the application project to include a file with the public key and to link with the library supporting an over-the-air upgrade and memory access (see Section 5).
3. Modify the application source code as follows (see Section 6).
    a. Define the application Product Version information.
    b. Add the OTA Secure Upgrade service to the GATT database.
    c. Process commands and data received from the peer during SOTAFU process.
4. Build the application. See the Kit Guide for your device included with your WICED Bluetooth SDK for the description of the build process.
5. Sign the built image using the private key generated in step 1 (see Section 1).

Below is a summary of the steps required to prepare the next version of the application.

1. If security vulnerability is being fixed, increment the major version and optionally set the minor version to zero (0). Otherwise, increment the minor version.
2. Build the new version of the application.
3. Sign the application using the same private key that was used to sign the original version of the application.

Note: The WICED Bluetooth SDK contains source code and binaries for applications to generate ECDSA keys, sign and verify the image. See the *wiced_btsdk* project in the ModusToolbox IDE Project Explorer pane, under the *wiced_btsdk\tools\btsdk-utils\ecdsa25* directory.

4 Keys Generation

The WICED Bluetooth SDK includes a utility to create the private and public keys (*ecdsa_genkey.exe*). The application can be used to generate a public-private key pair or to create a public key based on a previously generated private key. In the latter case pass the name of the file with the binary private key as a parameter.

When application is executed, it generates:

- *ecdsa256_key.pri.bin* – Contains binary representation of the private key.
- *ecdsa256_key.pub.bin* – Contains binary representation of the public key.
- *ecdsa256_key_plus.pub.bin* – Contains binary representation of the public key concatenated with two bytes containing the number of zero bits in the key. This can be used when public key is stored in the OTP area and can be tempered with.
- *ecdsa256_pub.c* – Source code containing the public key that can be included in the WICED application project that will perform SOTAFU.

5 Project Modification

Copy the *ecdsa256_pub.c* file generated as per Keys Generation to the application source folder. The ModusToolbox build system will automatically build all sources found under the application folder.

Modify the application *makefile*:

- Initialize the `OTA_FW_UPGRADE` and `OTA_SEC_FW_UPGRADE` makefile variables to 1.
- Include the Firmware Upgrade Library components:

```
ifeq ($(OTA_FW_UPGRADE),1)
COMPONENTS+=fw_upgrade_lib
SEARCH_LIBS_AND_INCLUDES+=$(CY_SHARED_PATH)/dev-kit/libraries/btsdk-ota
endif
```

The 'ota_fw_upgrade' sample application demonstrates these settings, and is available for download using the ModusToolbox New Application wizard.

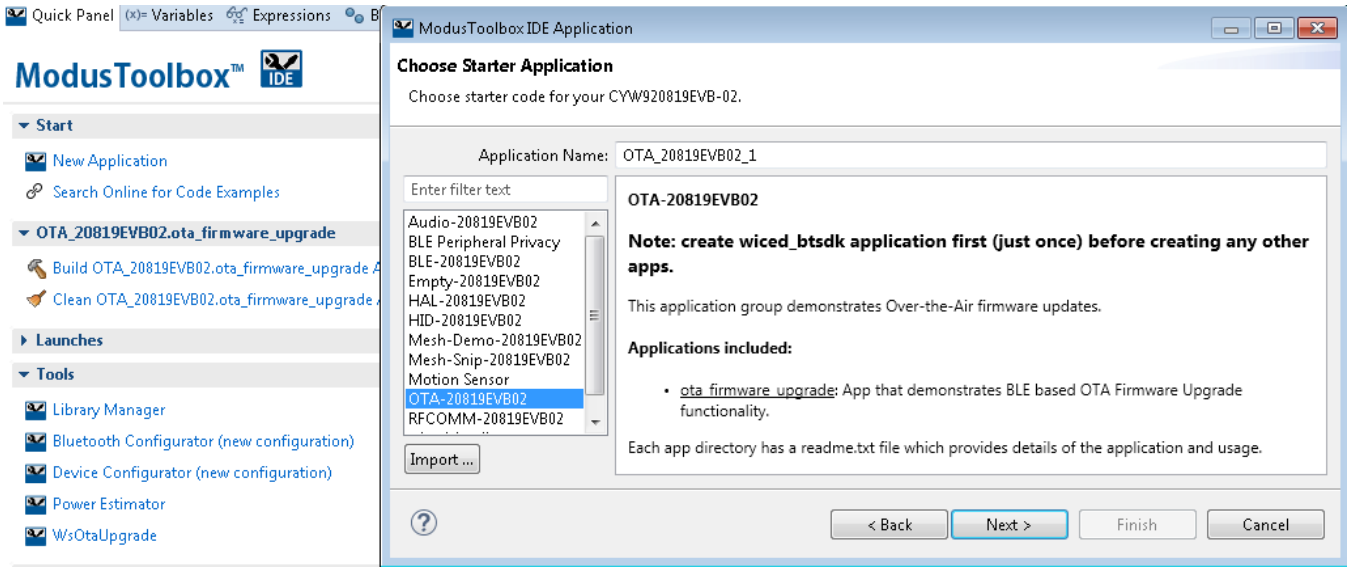

Figure 1. ModusToolbox New Application Wizard

6 Source Code to Support SOTAFU

Each application that supports SOTAFU should have:

- Unique product ID and version info (see Application Versioning).
- GATT database section that includes descriptions of the SOTAFU service and characteristics (see Changes to the GATT Database).
- Code to process SOTAFU commands and image data (see Firmware Upgrade Library Access).

6.1 Application Versioning

Each application contains three fields for version control. During the upgrade the application should verify that the download has the same product ID and that the major version did not decrease.

Below are some sample definitions:

```
#define HELLO_SENSOR_APP_ID   0x31AB
#define HELLO_SENSOR_APP_VERSION_MAJOR 1
#define HELLO_SENSOR_APP_VERSION_MINOR 1
PLACE_IN_APP_ID_AREA wiced_bt_application_id_t ble_remote_app_id =
{
    .product_id = HELLO_SENSOR_APP_ID,
    .major      = HELLO_SENSOR_APP_VERSION_MAJOR,
    .minor      = HELLO_SENSOR_APP_VERSION_MINOR
};
```

Do not change the product ID when a new version of an application is developed.

Increase the major version when a security vulnerability is fixed. Examples of security vulnerability include application crashes under certain conditions and unauthorized remote access to internal data. Image verification requires that the major version number be the same or greater than that of the existing version, and this prevents downgrading to a prior version that has vulnerability. Change the minor version if no critical problem was fixed.

6.2 Changes to the GATT Database

To include SOTAFU the application GATT database shall publish a special Cypress vendor-specific service. Definitions for the handles and UUIDs are included in the *wiced_bt_firmware_upgrade.h* header file.

Typically, the application will need to include the following snippet in the GATT database. The handles used by the service should be in the sorted order. The service shall be placed after the last application service.

The snippet below can be added as-is in the application code.

```
    // Handle 0xff00: Cypress vendor specific WICED Secure OTA Upgrade Service.
    PRIMARY_SERVICE_UUID128
        (HANDLE_OTA_FW_UPGRADE_SERVICE, UUID_OTA_SEC_FW_UPGRADE_SERVICE),

    // Handles 0xff01: characteristic Secure Firmware Upgrade Control Point, handle 0xff02
    // characteristic value.
    CHARACTERISTIC_UUID128_WRITABLE
    (
        HANDLE_OTA_FW_UPGRADE_CHARACTERISTIC_CONTROL_POINT,
        HANDLE_OTA_FW_UPGRADE_CONTROL_POINT,
        UUID_OTA_FW_UPGRADE_CHARACTERISTIC_CONTROL_POINT,
        LEGATTDB_CHAR_PROP_WRITE | LEGATTDB_CHAR_PROP_NOTIFY | LEGATTDB_CHAR_PROP_INDICATE,
        LEGATTDB_PERM_VARIABLE_LENGTH | LEGATTDB_PERM_WRITE_REQ | LEGATTDB_PERM_AUTH_WRITABLE
    ),

    // Handle 0xff03: Declare client characteristic configuration descriptor
    CHAR_DESCRIPTOR_UUID16_WRITABLE
    (
        HANDLE_OTA_FW_UPGRADE_CLIENT_CONFIGURATION_DESCRIPTOR,
        UUID_DESCRIPTOR_CLIENT_CHARACTERISTIC_CONFIGURATION,
        LEGATTDB_PERM_READABLE | LEGATTDB_PERM_WRITE_REQ | LEGATTDB_PERM_AUTH_WRITABLE
```

```
    ),

    // Handle 0xff04: characteristic OTA firmware upgrade data, handle 0xff05 characteristic
    // value. This characteristic is used to send portions of the FW image
    CHARACTERISTIC_UUID128_WRITABLE
    (
        HANDLE_OTA_FW_UPGRADE_CHARACTERISTIC_DATA,
        HANDLE_OTA_FW_UPGRADE_DATA,
        UUID_OTA_FW_UPGRADE_CHARACTERISTIC_DATA,
        LEGATTDB_CHAR_PROP_WRITE,
        LEGATTDB_PERM_VARIABLE_LENGTH | LEGATTDB_PERM_WRITE_REQ | LEGATTDB_PERM_AUTH_WRITABLE
    ),
```

6.3 Firmware Upgrade Library Access

The WICED Firmware Upgrade Library (provided in the *wiced_btsdk* project created and used by all WICED applications) implements functionality common to all applications required firmware upgrade.

The application needs to initialize the library, send notifications to the library when a connection to a peer device is established or dropped, and pass to the library packets received from the connected peer and destined for the OTA Firmware Upgrade service. This includes read, write, and indication confirmation messages received from the peer device.

Note: In the samples below the code assumes that OTA Firmware Upgrade service is the last service in the GATT database.

6.3.1 Library Initialization

The WICED Firmware Upgrade Library should be initialized any time during the application execution after the stack has been initialized and before the first GATT Write request is passed to the library.

The following snippet shows library initialization that is performed during device startup.

```
void app_init(void)
{
    /* Initialize WICED app */
    wiced_bt_app_init();

    if (!wiced_ota_fw_upgrade_init(&ecdsa256_public_key, NULL))
    {
        WICED_BT_TRACE("OTA upgrade Init failure!!!\n");
    }

    // Continue with the application initialization
```

6.3.2 Connection Handler

The application passes connection up/down events to the library whenever it receives notification from the stack.

The following snippet of the code shows how the application should process connection events.

```
wiced_bt_gatt_status_t app_connection_status_event(wiced_bt_gatt_connection_status_t
*p_status)
{
    // Pass connection up/down event to the OTA FW upgrade library
    wiced_ota_fw_upgrade_connection_status_event(p_status);

    // Continue with the event processing
```

6.3.3 Read Handler

The following snippet of the code shows how the application should process GATT Read requests from the peer.

```
wiced_bt_gatt_status_t app_read_handler(uint16_t conn_id, wiced_bt_gatt_read_t * p_read)
{
    // if read request is for the OTA FW upgrade service, pass it to the library to process
    if (p_read->handle > HANDLE_OTA_FW_UPGRADE_SERVICE)
```

```
    {
        return wiced_ota_fw_upgrade_read_handler(conn_id, p_read);
    }

    // Continue with the event processing
```

6.3.4 Write Handler

The following snippet of the code shows how the application should process GATT Write requests from the peer.

```
wiced_bt_gatt_status_t app_write_handler(uint16_t conn_id, wiced_bt_gatt_write_t *
p_write_data)
{
    // if write request is for the OTA FW upgrade service, pass it to the library to process
    if (p_write_data->handle > HANDLE_OTA_FW_UPGRADE_SERVICE)
    {
        return wiced_ota_fw_upgrade_write_handler(conn_id, p_write_data);
    }

    // Continue with the event processing
```

6.3.5 Indication Confirmation Handler

The following snippet of the code shows how the application should process GATT Indication Confirmations received from the peer.

```
wiced_bt_gatt_status_t app_conf_handler(uint16_t conn_id, uint16_t handle)
{
    // if indication confirmation is for the OTA FW upgrade service, pass it to the library
    // to process
    if (handle > HANDLE_OTA_FW_UPGRADE_SERVICE)
    {
        return wiced_ota_fw_upgrade_indication_cfm_handler (conn_id, handle);
    }

    // Continue with the event processing
```

7 Build the SOTAFU Image

When a target application is built, a separate binary file for the over-the-air upgrade is created in the build directory. For example, when the **ota_ firmware_upgrade** is successfully built for the CYW920819EVB-02 platform, the *OTA_FirmwareUpgrade_CYW920819EVB-02.ota.bin* file is created in the output folder for the application.

8 Sign the SOTAFU Image

Execute the `ecdsa_sign` utility, passing the name of the OTA upgrade image file name as a parameter.

The command below is used to sign the *OTA_FirmwareUpgrade_CYW920819EVB-02.ota.bin* file created during the build process.

Note: The following procedure should be executed in a secure environment since a private key is used. The application uses the *ecdsa256_key.pri.bin* file.

```
ecdsa_sign OTA_FirmwareUpgrade_CYW920819EVB-02.ota.bin
hash = f8314ab0600f3b5c6477df534710cd944939686440bb72c33d859e8200421f89
r = 771b80bbee9fdf06e28c1ff2a86cd15ef5ddc9babababaa43d98018531bb3af223
s = c85975f5757749fa76983e3714b694bc00096f43ab6ad3990f238c30d006bc97
Signed file OTA_FirmwareUpgrade_CYW920819EVB-02.ota.bin.signed
```

The signed output file can be used to perform the upgrade. During execution the application prints out the hash of the image and the signature that can be used for debugging.

9 Upgrade Firmware

The *wiced_btsdk/tools/btsdk-peer-apps-ota* directory contains the source code and the executables to perform the upgrade in Windows or Android environments. Under Windows pair the computer with the device running the image, and execute the `WsOtaUpgrade` utility passing the name of the signed firmware image file as a parameter. For example:

```
<path>\WsOtaUpgrade.exe ota_firmware_upgrade-CYW920706WCDEVAL-rom-ram-Wiced-
release.ota.bin.signed
```

The application displays the progress of the operation and the status when the procedure is complete.

References

The references in this section may be used in conjunction with this document.

Document or Item Name	Number	Source Items
[1] WICED Firmware Upgrade Library	002-19289	WICED Bluetooth SDK

Document History

Document Title: Secure Over-the-Air Firmware Upgrade

Document Number: 002-16561

Revision	ECN	Submission Date	Description of Change
**	5861331	08/23/2017	Initial release
*A	6336320	10/08/2018	Replaced “WICED Studio” with “ModusToolbox”. Replaced “WICED Studio SDK” with “WICED SDK with PSoC 6 Support”. Updated Sales page.
*B	6489230	02/19/2019	Updated to include CYW20819 Replaced “WICED SDK with PSoC 6 Support” with “WICED Bluetooth SDK”
*C	6554890	04/24/2019	Removed Associated Part Family Updated for BT SDK release
*D	6701185	10/15/2019	Updated for ModusToolbox 2.0

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

PSoC® Solutions

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

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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