Secure OTA Firmware Update

Application Note for 32-bit NuMicro® Family

Document Information

Abstract	Introduce a secure system firmware upgrade architecture through OTA.
Apply to	NuMicro® M2351 series

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 Introduction

This document introduces a secure OTA firmware update mechanism based on the trusted boot(*), new firmware verification and host authentication. OTA (Over-the-Air Technology) is a technique for deploying new firmware to a terminal device by transmitting firmware information through a wireless interface.

In the Nuvoton Secure Microcontroller Platform (NuSMP), the NuBL2 – Trusted Bootloader is the first executed firmware responsible for trusted boot, memory partition and OTA update service (including firmware download). User's secure firmware can use the NuBL2's OTA update service to download and update secure/non-secure firmware.

In the following chapters, Chapter 2 introduces the firmware update architecture, includes all the firmware features in the OTA client side and verification mechanism. Chapter 3 demonstrates a firmware update sample, and describes how to set up the OTA client, the OTA server and the wireless AP (Access Point). Chapter 4 introduces an offline firmware update method.

(*): Please refer to the M2351 Trusted Boot application note for detailed information about trusted boot.

2 Firmware Update Architecture

Figure 2-1 shows the firmware update architecture in OTA client.

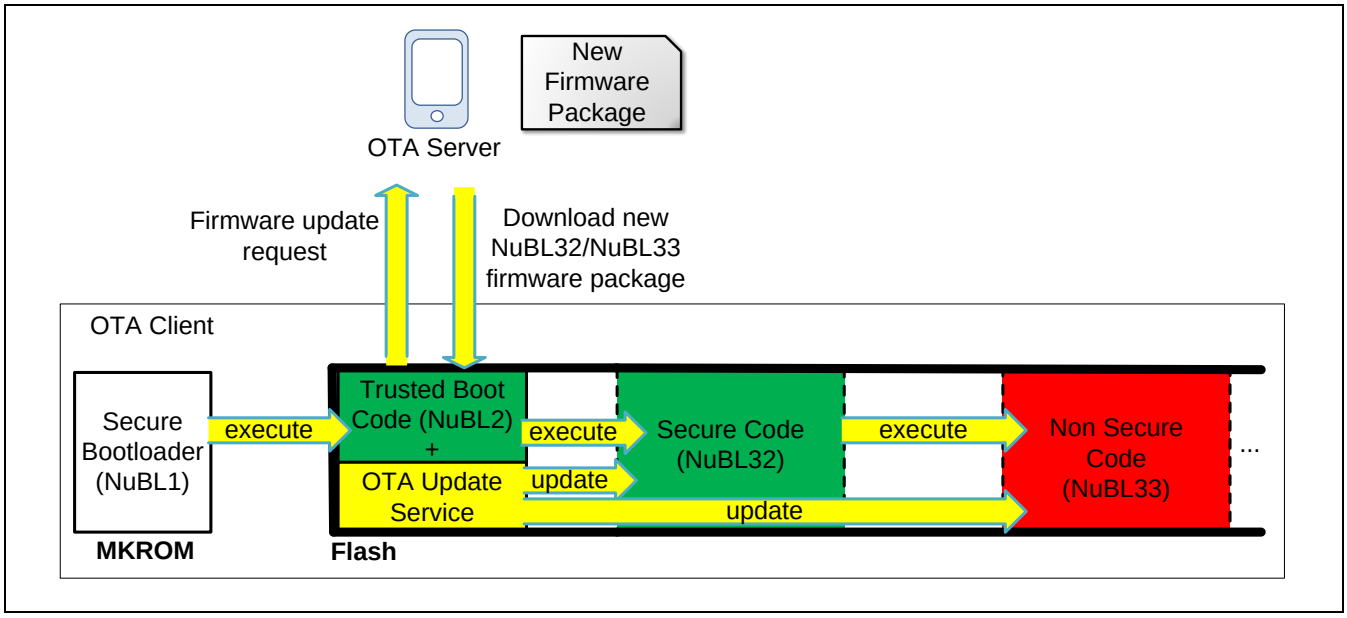

Figure 2-1 Firmware Update Architecture in Client

The firmware update architecture includes the OTA server and the OTA client. The OTA server can transfer new firmware to the client. The OTA client downloads new firmware and updates the NuBL32/NuBL33 firmware. All firmware in OTA client are described detailedly in Section 2.1.

The OTA client executes in the trusted execution environment. Each firmware is verified before executing. After power on, NuBL1(the root of trust) starts and verifies the NuBL2. Once passed verification, the NuBL2 gets execution and verifies the NuBL32/NuBL33. Finally, NuBL32 is executed to initial secure service and then jump to NuBL33.

When a new version of NuBL32/NuBL33 firmware package has been released to the OTA server, the NuBL32 can call OTA update service of NuBL2 in OTA client to request firmware update. The OTA update service is responsible to download the new firmware package and then update the NuBL32 and NuBL33 in the OTA client.

The firmware package file contains the firmware's signature, firmware information and the encrypted firmware image. Section 3.4 will describe how to use the CryptoTool to generate firmware package file.

2.1 Firmware Features

The firmware features in the OTA client are as below.

- NuBL1
  - Secure Bootloader, built in Mask ROM:
    - Performs the trust boot verification to NuBL2.
- NuBL2
  - Trusted Bootloader, the first executing firmware in secure Flash:
    - Performs the trust boot verification to NuBL32 and NuBL33.
    - Provides OTA update service for downloading new NuBL32/NuBL33 firmware package, decrypts firmware, and updates original NuBL32/NuBL33 firmware in OTA client.
- NuBL32
  - A secure code in the trusted execution environment:
    - Performs NuBL33 initialization then jumps to execute NuBL33.
    - Provides secure services for NuBL33.
    - Calls OTA update service for requesting firmware update.
- NuBL33
  - A non-secure code used for general application.

2.2 Authentication Mechanism

Figure 2-2 shows the client and server authentication diagram.

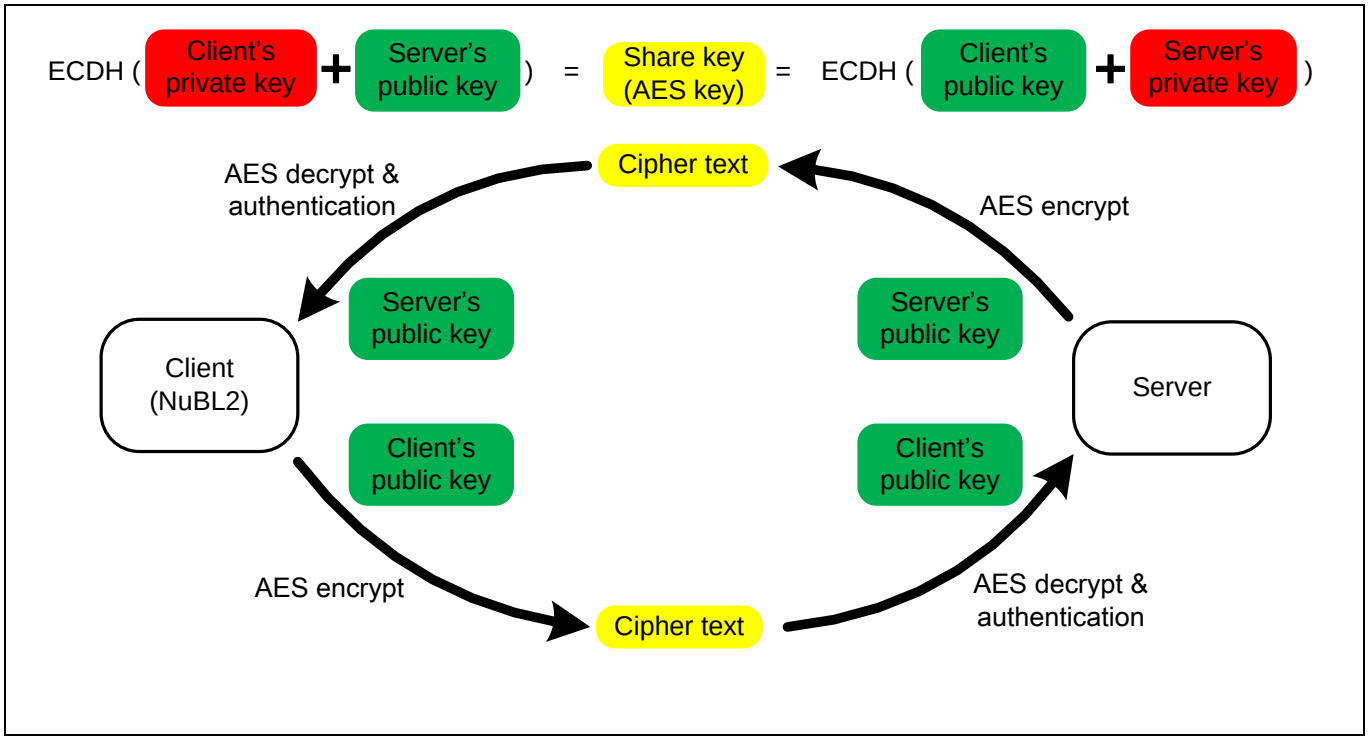

Figure 2-2 Client and Server Authentication Diagram

The client and server need to perform authentication process before downloading new firmware. The authentication takes the ECDH key exchange algorithm (Elliptic-curve Diffie–Hellman key Exchange) to identify each other.

Both the NuBL2 (in client) and server need generate the private/public key pair, and exchange public key to each other before connecting. Then client and server can calculate the same ECDH shared secret key based on its own private key and public key of the other side. This shared secret key is used to encrypt/decrypt transfer data between client and server. After connecting, the NuBL2 will receive an encrypted public key from server. The NuBL2 can authenticate the server by comparing the received public key with pre-exchanged public key. Similarly, server will receive an encrypted public key from client and compare it with pre-exchanged public key to identify the client.

The same method described above is used to verify the NuBL32/NuBL33 firmware download. Therefore, the server also needs to get NuBL32/NuBL33 public key before being connected.

3 SecureOTADemo Sample Code

The SecureOTADemo sample code is located in the M2351 BSP under bsp\SampleCode\MKROM\SecureOTADemo. This sample code contains two parts: the OTA client sample code and the Android App as the OTA server. The client sample code should run on a NuMaker-M2351 board with ESP8266 Wi-Fi module. The server sample program, OTA_Update_App.apk, need be installed on an Android mobile device. Both the OTA client and the OTA server connect to same wireless AP (Access Point) to demonstrate the firmware update for System firmware (NuBL32) and application firmware (NuBL33). The visual difference between before and after the firmware update is that the two LED lights on the M2351 NuMaker board change from off to blinking. Both the system firmware (NuBL32) and the application firmware (NuBL33) will be upgraded from the version number 1 to 2.

Figure 3-1 shows the Flash configuration in the client sample code. NuBL2 and NuBL32 are in the secure region Flash. NuBL33 is in the non-secure region Flash.

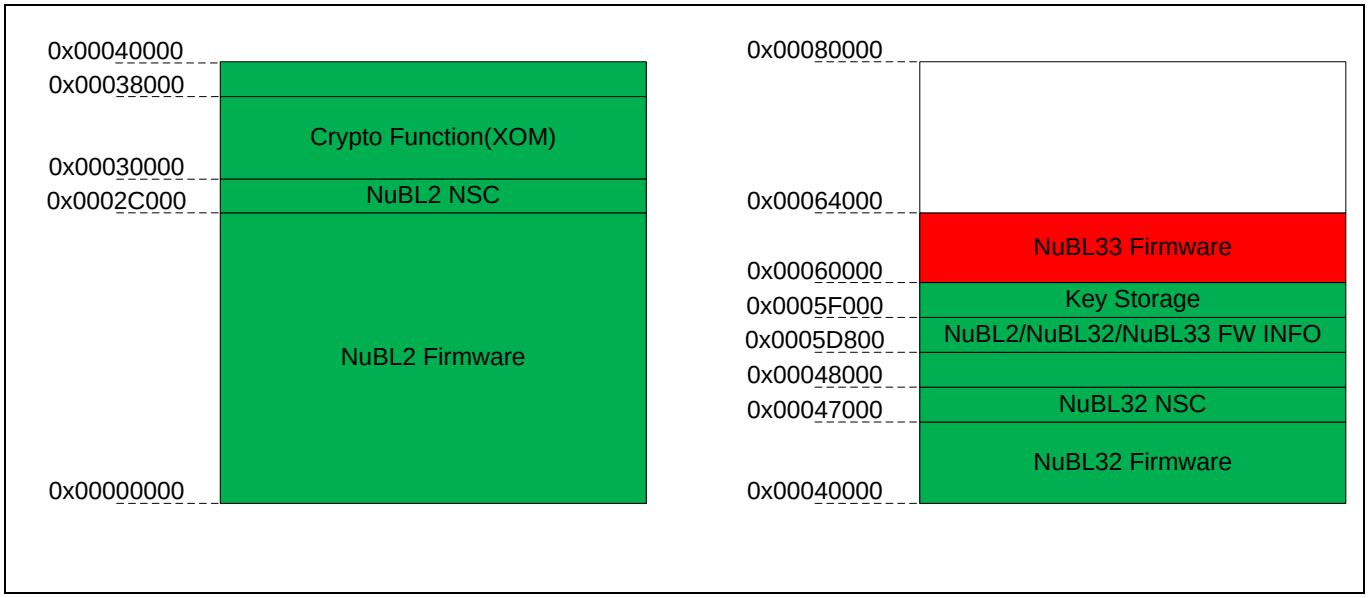

Figure 3-1 Flash Memory Map of Client

3.1 NuBL2 Firmware Update Flow

Figure 3-2 shows the on the fly firmware update flowchart of NuBL2. This update method does not store the received new NuBL32/NuBL33 firmware package, and directly performs firmware update in the connected state.

The “OTA Update Status” in Figure 3-2 is one word data (e.g. 0 or 1) and programmed in secure region Flash. NuBL2 uses the “OTA Update Status” to record firmware update progress. If the value of “OTA Update Status” is 0, it means NuBL2 needs to execute trust boot flow. If the value of “OTA Update Status” is 1, it means NuBL2 needs to execute firmware update flow.

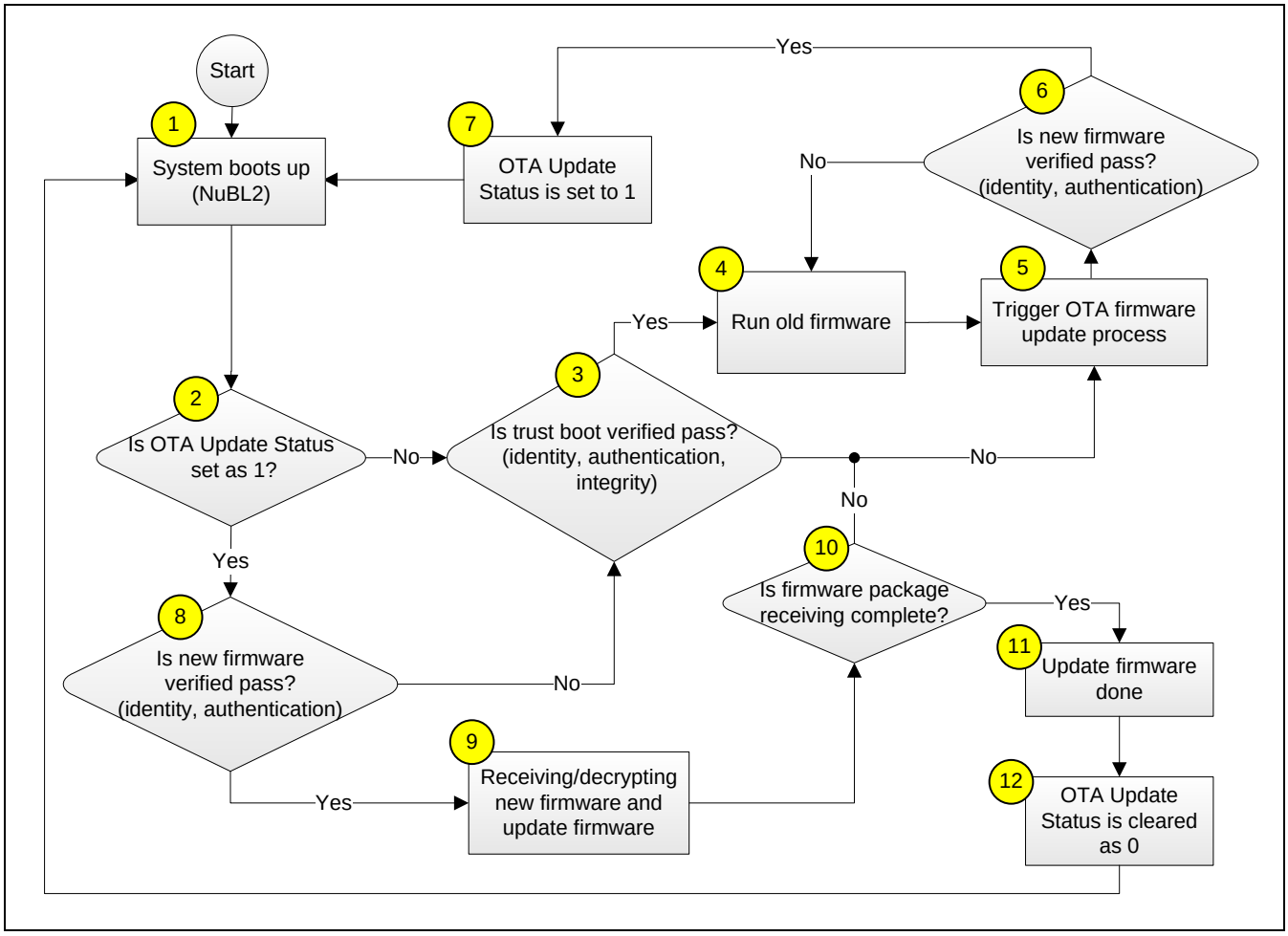

Figure 3-2 NuBL2 Firmware Update Flowchart

3.2 NuBL2 Firmware Sample Code

The following lists firmware upgrade macro definitions.

```
ota_api.h
#define OTA_UPGRADE_FROM_SD 0 /* 1: Enable OTA update form SD card method */
                              /* 0: Enable OTA update on the fly method */
```

```
NuBL_common.h
/*---------------------------------------------------------------------------------------*/
/* FLASH memory layout definitions                                                       */
/*---------------------------------------------------------------------------------------*/
#define NUBL2_FW_BASE           0x00000000ul    /* 0K, Secure code */
#define NUBL32_FW_BASE          0x00040000ul    /* 256K, Secure code; SRAM 32~48K */
#define NUBL33_FW_BASE          0x00060000ul    /* 384K, Non-secure code start address */

#define NUBL2_LIB_BASE          0x0002C000ul    /* 176K (0x2C000) */
#define NUBL32_LIB_BASE         0x00047000ul    /* 284K, non-Secure callable code */

#define NUBL1_IB_BASE           (NUBL33_FW_BASE - (1024*2)) /* 382K, 0x5F800 */

#define KEY_STORAGE_BASE        (NUBL1_IB_BASE - (1024*2))  /* 380K, 0x5F000 (encrypt
[NuBL32, NuBL33 and Host(for cmd) public keys]) */
#define NUBL2_FW_INFO_BASE      (NUBL1_IB_BASE - (1024*4))  /* 378K, 0x5E800 */
#define NUBL32_FW_INFO_BASE     (NUBL1_IB_BASE - (1024*6))  /* 376K, 0x5E000 */
#define NUBL33_FW_INFO_BASE     (NUBL1_IB_BASE - (1024*8))  /* 374K, 0x5D800 */
```

```
ota.h
#define FW_INFO_SIZE     (0x150UL) /* Size of FW INFO */

#define OTA_STATUS_BASE (0x48000) /* Base address of OTA status for OTA update from SD
card method */

#define SYS_FW_OTA_STATUS_BASE (0x48000 + FMC_FLASH_PAGE_SIZE) /* Base address of system
(NuBL32) firmware OTA status for OTA update on the fly method */

#define APP_FW_OTA_STATUS_BASE (0x48000 + (FMC_FLASH_PAGE_SIZE * 2)) /* Base address of
application (NuBL33) firmware OTA status for OTA update on the fly method */

#define SYS_FW_BASE (NUBL32_FW_BASE) /* base address of current system (NuBL32) firmware
*/
#define APP_FW_BASE (NUBL33_FW_BASE) /* base address of current application (NuBL33)
firmware*/
```

The following is a NuBL2 sample code for the main function on the OTA client side. Users can use one of two OTA update methods by configuring the OTA_UPGRADE_FROM_SD definition in the ota_api.h. Users can configure OTA_UPGRADE_FROM_SD as 0 to use update firmware on the fly method.

```
/*---------------------------------------------------------------------------------------------*/
/*  MAIN function                                                                              */
/*---------------------------------------------------------------------------------------------*/
/* update on the fly */
int main(void)
{
    uint32_t    count;
    uint32_t u32NeedReset = 0;
    uint32_t u32Cfg;
    uint8_t u8FailNuBL3x; /* Bit0 = 1: NuBL32, Bit1 = 1: NuBL33 */

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Init UART for printf */
    UART_Init();

    printf("\n\n[HCLK %d Hz] (%s, %s)\n", SystemCoreClock, __DATE__, __TIME__);
    printf("+------------------------------+\n");
    printf("|    M2351 NuBL2 Sample Code    |\n");
    printf("+------------------------------+\n\n");
    CLK_SysTickDelay(200000);

    /* Create FW INFO */
    NuBLTrustBootInit();

    SYS_UnlockReg();
    FMC_Open();
    FMC_ENABLE_AP_UPDATE();
    /* Enable WDT and check if system reset by WDT */
    WdtEnableAndCheck();

    printf("check OTA status:%d\n", FMC_Read(OTA_STATUS_BASE));
```

```
    /* check OTA status for entry update mode */
    if (FMC_Read(OTA_STATUS_BASE) == 1)
    {
        /* init wifi module and connect to OTA server */
        OTA_Init(__HSI, (ISP_INFO_T *)&g_NuBL2ISPInfo);
        if (OTA_TaskProcess() == 0)
        {
            SYS_UnlockReg();
            FMC_Open();
            FMC_ENABLE_AP_UPDATE();
            /* upgrade done, clear OTA status */
            FMC_Erase(OTA_STATUS_BASE);

            printf("update done. Check OTA status: %d\n", FMC_Read(OTA_STATUS_BASE));
            if ((FMC_Read(OTA_STATUS_BASE) == 1))
            {
                printf("[ERR]clear OTA status error\n");
            }
            goto reset;
        }
    }

    /* normal boot */
    /* verify application firmware identity and integrity */
    NuBL2_Init();
    /* verify system firmware identity and integrity */
    if(NuBL2_ExecuteVerifyNuBL3x(NULL, 0, AUTH_METHOD) != 0)
    {
        /* if verify failed, do OTA update right now */
        printf("\n\nNuBL2 verifies NuBL32 FAIL.\n\n");
        /* set this flag for reset NuBL32 firmware version to 0. */
        u8FailNuBL3x = BIT0;

        goto _VERIFY_FAIL;
    }
    else
    {
        printf("\n\nNuBL2 verifies NuBL32 PASS.\n\n");
    }

    if(NuBL2_ExecuteVerifyNuBL3x(NULL, 1, AUTH_METHOD) != 0)
```

```
    {
        /* if verify failed, do OTA update right now */
        printf("\n\nNuBL2 verifies NuBL33 FAIL.\n\n");
        /* set this flag for reset NuBL33 firmware version to 0. */
        u8FailNuBL3x = BIT1;

        goto _VERIFY_FAIL;
    }
    else
    {
        printf("\n\nNuBL2 verifies NuBL33 PASS.\n\n");
    }
    __set_PRIMASK(1); /* Disable all interrupt */
    FMC_SetVectorPageAddr(NUBL32_FW_BASE);

    /* Reset CPU only to reset to new vector page */
    SYS_ResetCPU();
    while(1) {}

_VERIFY_FAIL:
    /* if verify failed, modified BL3x firmware version to 0, and do OTA update after
reboot */
    if (u8FailNuBL3x&BIT0)
    {
        FW_INFO_T FwInfo;

        /* Clear NuBL3x firmware version to 0 */
        FwInfo.mData.au32ExtInfo[0] = 0;
        /* NuBL32 */
        if (NuBL2_UpdateNuBL3xFwInfo((uint32_t *)&FwInfo, sizeof(FW_INFO_T), 0,
NUBL32_FW_INFO_BASE) != 0)
        {
            printf("\nClear NuBL32 F/W version failed.\n");
        }
    }

    if ((u8FailNuBL3x&BIT1))
    {
        FW_INFO_T FwInfo;

        /* Clear NuBL3x firmware version to 0 */
```

```
        FwInfo.mData.au32ExtInfo[0] = 0;
        /* NuBL33 */
        if (NuBL2_UpdateNuBL3xFwInfo((uint32_t *)&FwInfo, sizeof(FW_INFO_T), 1,
NUBL33_FW_INFO_BASE) != 0)
        {
            printf("\nClear NuBL33 F/W version failed.\n");
        }
    }

    SYS_UnlockReg();
    FMC_Open();
    FMC_ENABLE_AP_UPDATE();

    /* update OTA status for upgrade right now. */
    FMC_Write(OTA_STATUS_BASE, 0x1UL);

    /* init OTA */
    OTA_Init(__HSI, (ISP_INFO_T *)&g_NuBL2ISPInfo);
    if (OTA_TaskProcess() == 0)
    {
        /* check OTA status and re-boot for update firmware */
        printf("OTA update for NuBL3x verify failed: SYS:%d, APP:%d\n",
FMC_Read(SYS_FW_OTA_STATUS_BASE), FMC_Read(APP_FW_OTA_STATUS_BASE));
        if ((FMC_Read(SYS_FW_OTA_STATUS_BASE) == 1)||FMC_Read(APP_FW_OTA_STATUS_BASE) ==
1)
        {
            u32NeedReset = TRUE;
        }
    }
    goto reset;
    while(1) {}

reset:
    while(!(UART_IS_TX_EMPTY(DEBUG_PORT)));

    /* Reset CPU only to reset to new vector page */
    SYS_ResetChip();

    while(1) {}
}
```

3.3 OTA Porting Layer

Figure 3-3 shows the OTA porting layer.

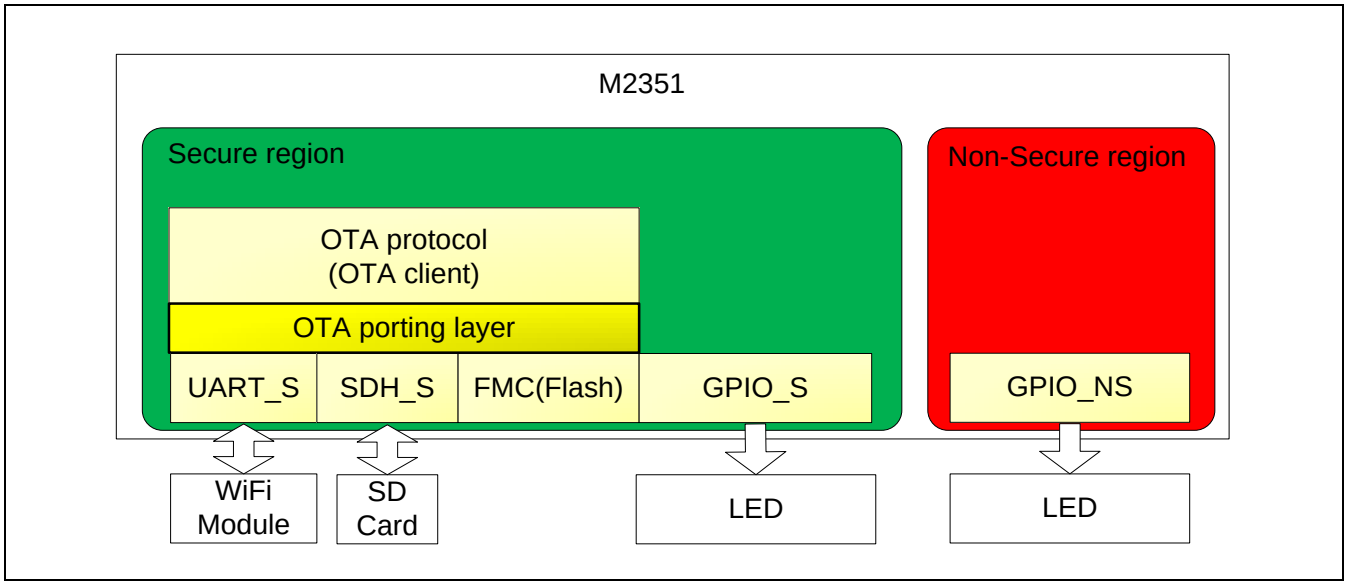

Figure 3-3 OTA Porting Layer

An OTA porting layer is at the bottom of the OTA protocol. The porting layer is connected to the peripherals of the MCU, such as FMC, secure UART, and secure SDH. It provides the flexibility for porting the OTA protocol to another MCU or wireless interface. The OTA protocol in this implementation is based on M2351 SecureISP Library of MaskROM Library. Users can refer to the AN_2023_M2351_MaskROM_Library_EN_Rev1.00 to know the OTA handshake flow and supported APIs of SecureISP Library..

The functions defined in the OTA porting layer are listed in Table 3-1. These functions can be classified into three groups for handling different hardware components including wireless module, FMC and SD card. Users can extend the defined functions of porting layer according to their requirement. When changing the hardware component, the related functions need to be ported. For example, if the wireless module is changed from Wi-Fi to BT, the functions for wireless module need to be implemented to control BT module.

Function Name	Description	Adapter Component
OTA_API_SendFrame	Send OTA command to another device.	Wireless module (Wi-Fi or BT module)
OTA_API_RecvCallBack	The callback interface for a received command.	Wireless module (Wi-Fi or BT module)
OTA_API_Init	Hardware initialize function (optional).	Wireless module (Wi-Fi or BT module), SD card

OTA_API_GetFwUpgradeDone	Get flag of firmware upgrade done.	Wireless module (Wi-Fi or BT module)
OTA_API_SetResetFlag	Set Reset flag for transfer task. System executes the reset operation after transfer task has finished the current task.	Wireless module (Wi-Fi or BT module)
OTA_API_TaskProcess	Transfer task routine.	Wireless module (Wi-Fi or BT module)
OTA_API_TransferConnClose	Disconnect transfer connection.	Wireless module (Wi-Fi or BT module)
OTA_API_GetFlashPageSize	Get the Flash page size.	FMC (Flash)
OTA_API_EraseFlash	Erase Flash data.	FMC (Flash)
OTA_API_WriteFlash	Write Flash data.	FMC (Flash)
OTA_API_SDInit	Initialize the SD Host peripheral that includes configuring the multi-function pin, peripheral clock and enable the NVIC interrupt.	SD card
OTA_API_SDFwPackWriteOpen	Open new NuBL32/NuBL33 firmware package file from the SD card.	SD card
OTA_API_SDWrite	Write NuBL32/NuBL33 firmware package data into the SD card.	SD card
OTA_API_SDClose	Close NuBL32/NuBL33 firmware package file in the SD card.	SD card
OTA_API_SDFwPackReadOpen	Open existed NuBL32/NuBL33 firmware package file in the SD card.	SD card
OTA_API_SDRead	Read NuBL32/NuBL33 firmware package data from the SD card.	SD card

Table 3-1 Functions of OTA Porting Layer

3.4 Firmware Package Generation

Use the Nuvoton CryptoTool tool to generate the firmware package file used in OTA update. The firmware package file contains the firmware's signature, firmware information and the

encrypted firmware image.

Take the NuBL33 firmware package generation as an example. As shown in Figure 3-4, the steps are as follows.

1. Open CryptoTool and click the "ECDH" tab. Fill in the NuBL33's private key and the NuBL2's public key according to the field. Then click the "Calculate Share Key" button to calculate the ECDH Share Key.
2. Copy the ECDH Share Key and fill in the appropriate fields in the "Packer" tab. In order to make the cipher text of the firmware more difficult to be cracked, the AES CFB (Cipher feedback) mode is adopted. Therefore, the Initialization Vector must be filled. Then fill in the file locations of firmware information, Firmware 1, and Firmware 2, and specify the file name and storage path of the generated firmware package with "Output Pack File". After clicking the "Pack Firmware" button, the firmware package will be generated to the specified directory.

Two different firmware can be used in this Packer function. Because the NuBL32 is a secure firmware, usually a non-secure callable library is created for non-secure firmware. Therefore, the "Firmware 1" field in the "Packer" tab can be filled in the file path of the NuBL32 firmware and the "Firmware 2" field in the "Packer" tab can be filled in the file path of the non-secure callable library.

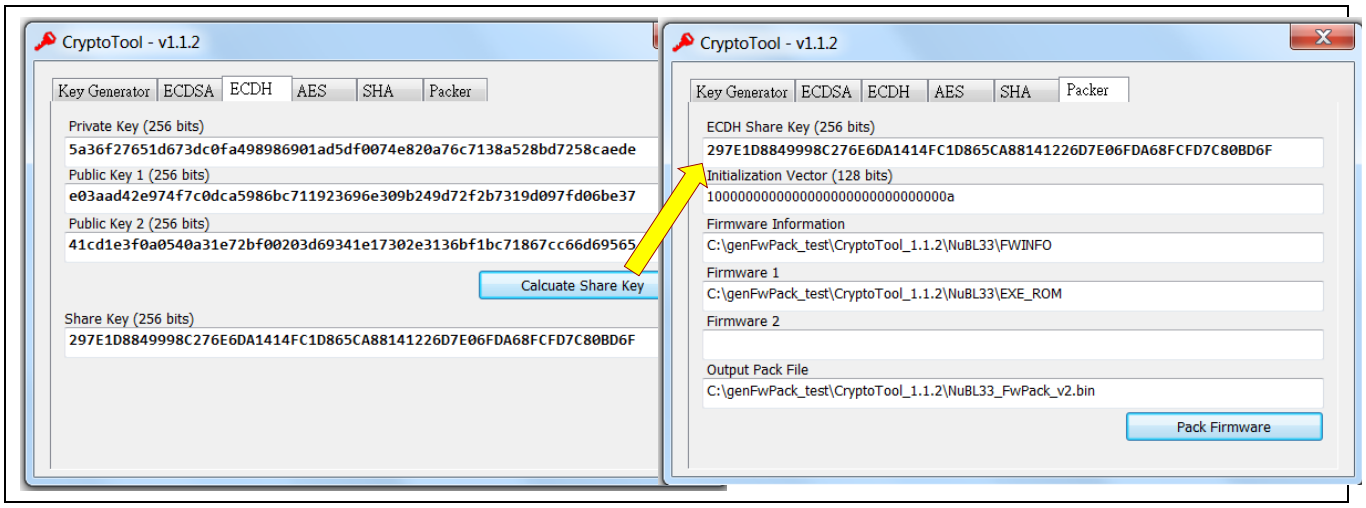

Figure 3-4 Firmware Package Generation by CryptoTool

3.5 Demo Environment Setup and Operation

A wireless AP is necessary for the demonstration. It is recommended to turn off the firewall and not to connect to external network during the demonstration.

1. Demo environment setup

   (1) Set up the OTA server

   i. Generate new firmware package files of NuBL32 and NuBL33 for OTA update. Use the NuBL32v2 and NuBL33v2 projects under bsp\SampleCode\MKROM\SecureOTADemo, and compile the bin file of NuBL32 firmware version 2 and NuBL33 firmware version 2. Next, use the CryptoTool utility to generate the corresponding firmware package file.

   ii. Install the OTA_Update_App.apk on the Android mobile device. Users can find the OTA_Update_App.apk file in the M2351 BSP under bsp\SampleCode\MKROM\SecureOTADemo\androidApp.

   iii. Put three files into the "Download" directory of the mobile device.

   The three files are NuBL32_FwPack_v2.bin, NuBL33_FwPack_v2.bin and license.txt. The NuBL32_FwPack_v2.bin is the firmware package of NuBL32 firmware version 2, and NuBL33_FwPack_v2.bin is the firmware package of NuBL33 firmware version 2. The license.txt is the public key of NuBL2. Since the ECC key algorithm is used, there will be two public keys, and the format in the file is a public key for each line. As shown in Figure 3-5, users can get the private and public keys of NuBL2, NuBL32 and NuBL33 in the /Keil\FwSign.ini file of their respective project.

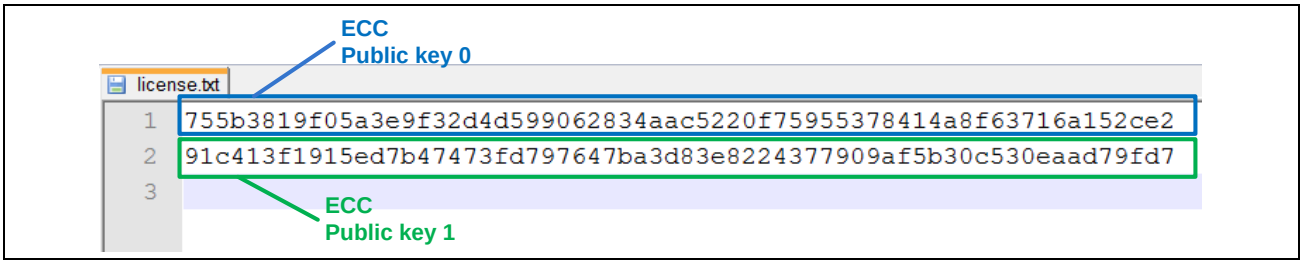

Figure 3-5 Public Keys in License File

(2) Turn on the wireless AP. The OTA server can get a local area network IP address after establishing a connection with the wireless AP.

(3) Set up the OTA client.

The steps to set up the OTA client are as follows.

i. Get the IP Address of the server. This network IP address of the OTA server can be found in the network settings of the mobile device. Figure 3-6 shows the IP address interface of the mobile device.

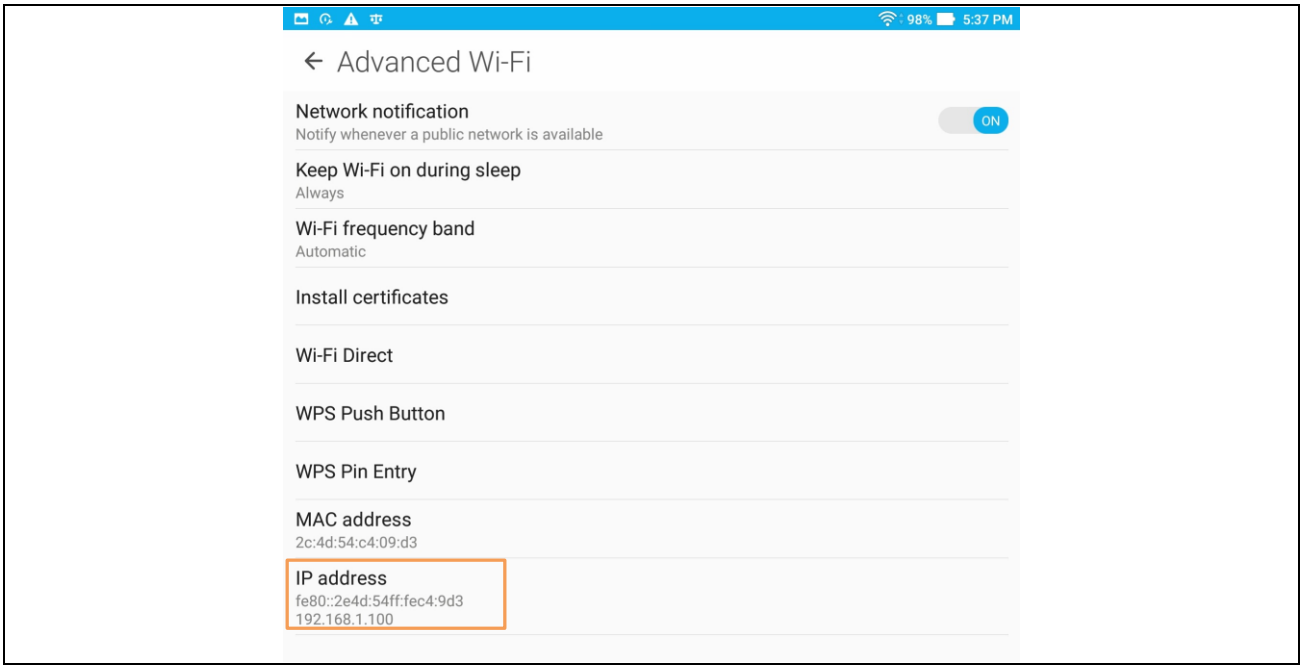

Figure 3-6 IP Address of OTA Server

ii. Configure the network connection of the OTA client. Users can modify the following definitions in the ota_transfer.h under bsp\SampleCode\MKROM\SecureOTADemo\NuBL2\ota.

These settings are the name and password of the wireless AP and the IP address of the OTA server in the local area network.

```
ota_transfer.h
#define WIFINAME            "VIZIO_AP"        /* Name of wireless AP */
#define WIFIPASS            "123456789"       /* Password of wireless AP */
#define WIFIIP              "192.168.1.100"   /* IP of server */
```

iii. Set the OTA_DEMO_WITH_APP definition to 1 in the ota.h under bsp\SampleCode\MKROM\SecureOTADemo\NuBL2\ota.

```
ota.h
#define OTA_DEMO_WITH_APP 1
```

Set this definition to 1 to reduce the manual steps of the OTA demonstration. During the OTA update process, the OTA client needs to reboot the system when both the OTA client and the Android App verify their identity. After the client confirmed that there is a new firmware that can be updated, the OTA client needs to restart the system and switch to NuBL2 to perform OTA update process. The Android App will be notified before the system on the client side reboots. Then users need to manually restart the Android App to continue the OTA task. After

this definition is enabled, the new firmware version will be checked and updated by NuBL2 when the system is powered on.

iv. Open and rebuild the NuBL2, NuBL32 and NuBL33 projects. Then users download these firmware imagesto OTP client.

2. Demo operation

(1) Open the OTA Update App.

i. In the main screen, click the top "SELECT FILE" button, and select NuBL32_FwPack_v2.bin.

ii. Click the second "SELECT FILE" button, and select NuBL33_FwPack_v2.bin.

iii. Click the third "SELECT FILE" button, and select license.txt. Figure 3-7 shows the main screen of the Android App.

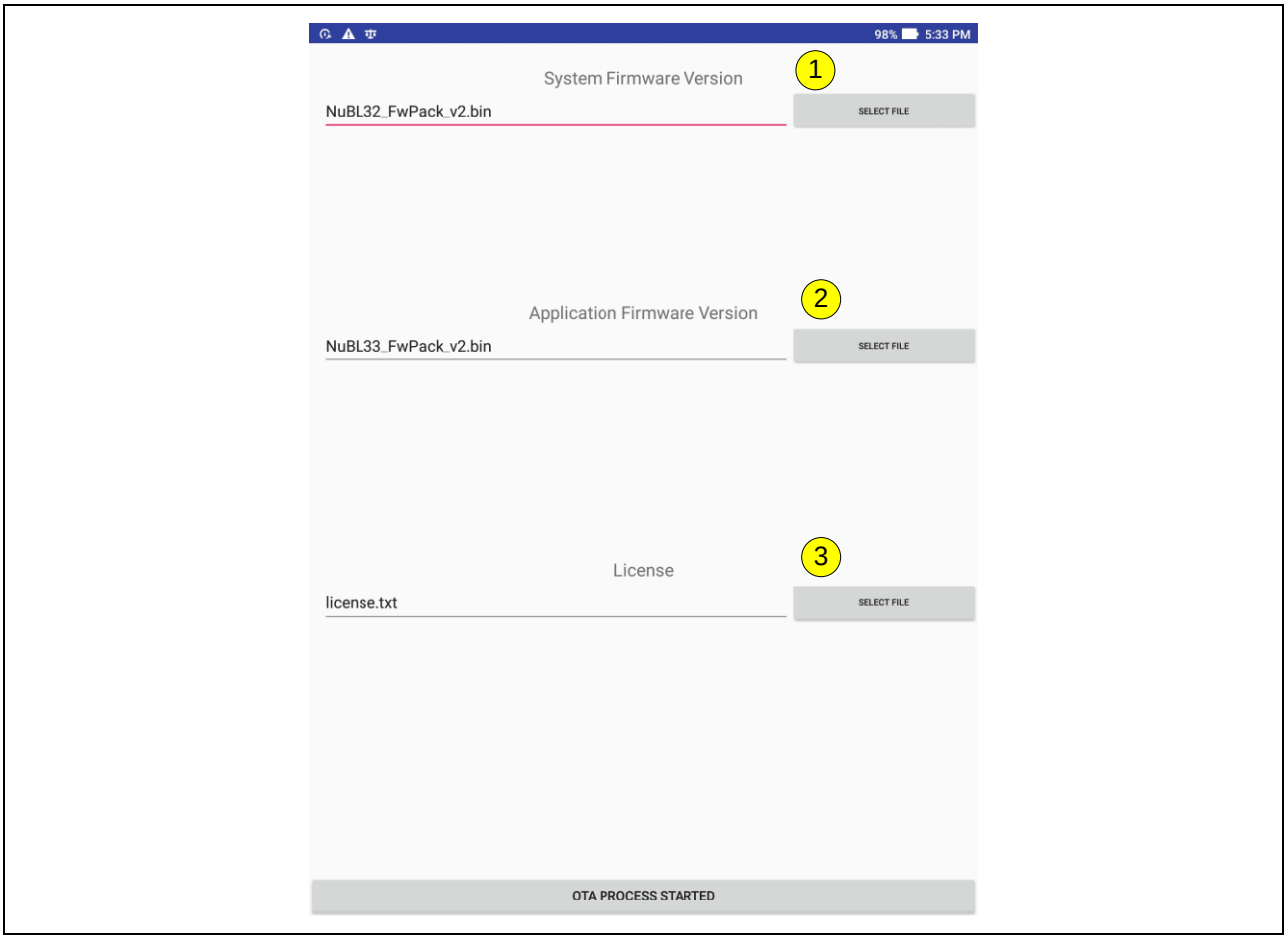

Figure 3-7 Select Files for OTA Server

iv. Wait for client to connect after clicking the "OTA PROCESS STRATED" button. After clicking the “OTA PROCESS STARTED” button, the “Open OTA Update

Server" prompt will be displayed and the OTA server is waiting for the client to connect now, as shown in Figure 3-8.

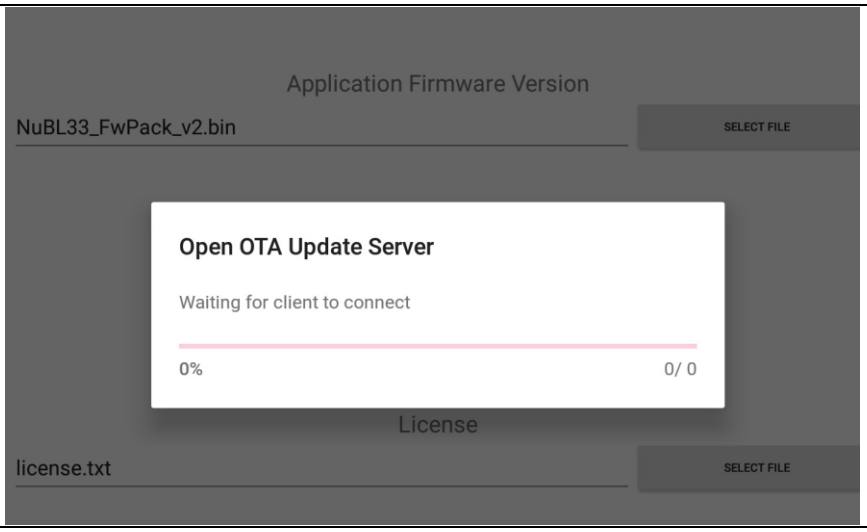

Figure 3-8 Open OTA Update Server

(2) Press the reset button on NuMaker-M2351 board. The OTA client will be connected to OTA server for OTA update after system reboots. Since the NuBL32 and NuBL33 firmware versions to be deployed by the OTA server are both 2, which is newer than the original OTA client, both NuBL32 and NuBL33 will be updated. Thus, users will see the prompt to update the NuBL32 firmware, as shown in Figure 3-9, and the prompt to update the NuBL33 firmware, as shown in Figure 3-10.

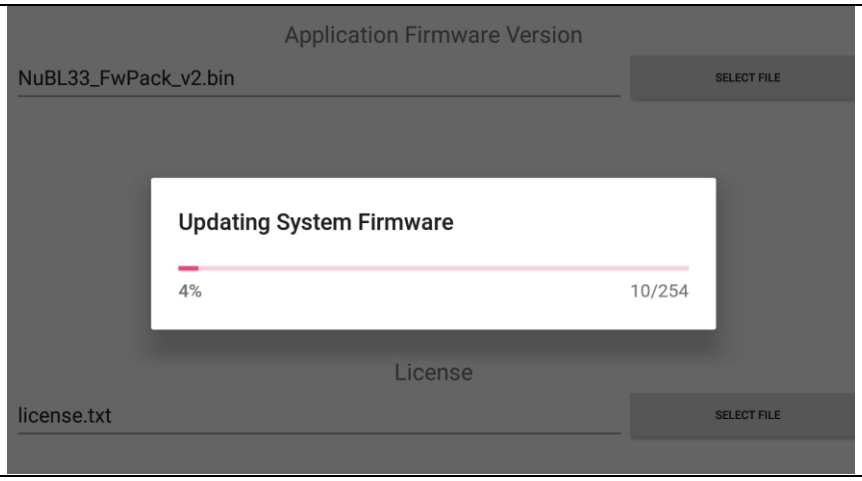

Figure 3-9 Updating System Firmware

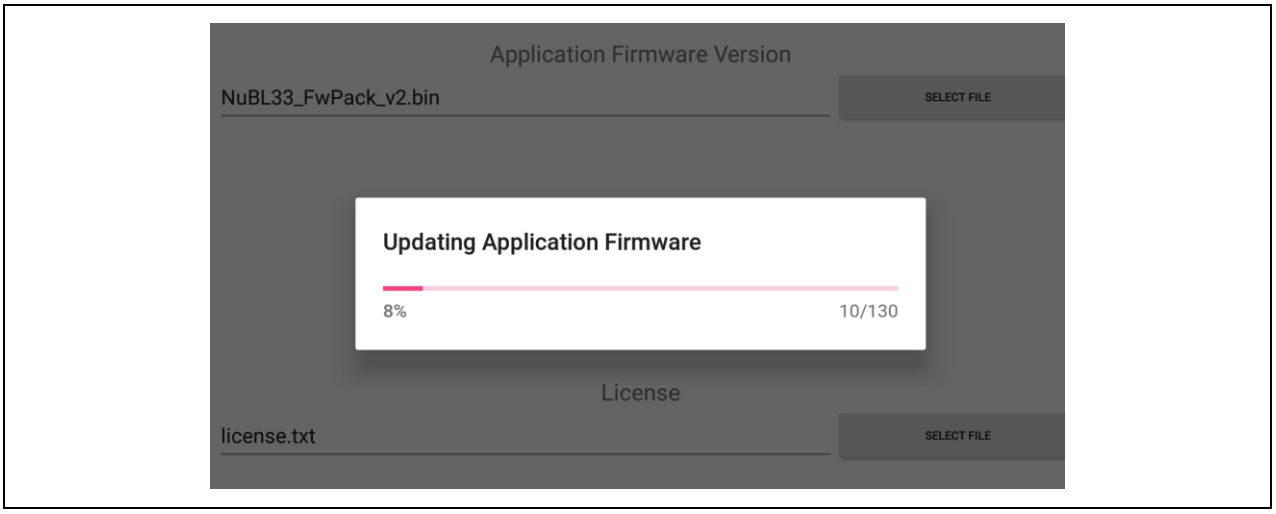

Figure 3-10 Updating Application Firmware

When the two new versions of the firmware are updated, the OTA server will prompt as shown in Figure 3-11. At this time, the OTA client will restart, and the firmware versions of NuBL32 and NuBL33 have been updated to version 2. Then, you can see the two LEDs on the NuMaker-M2351 board are flashing alternately.

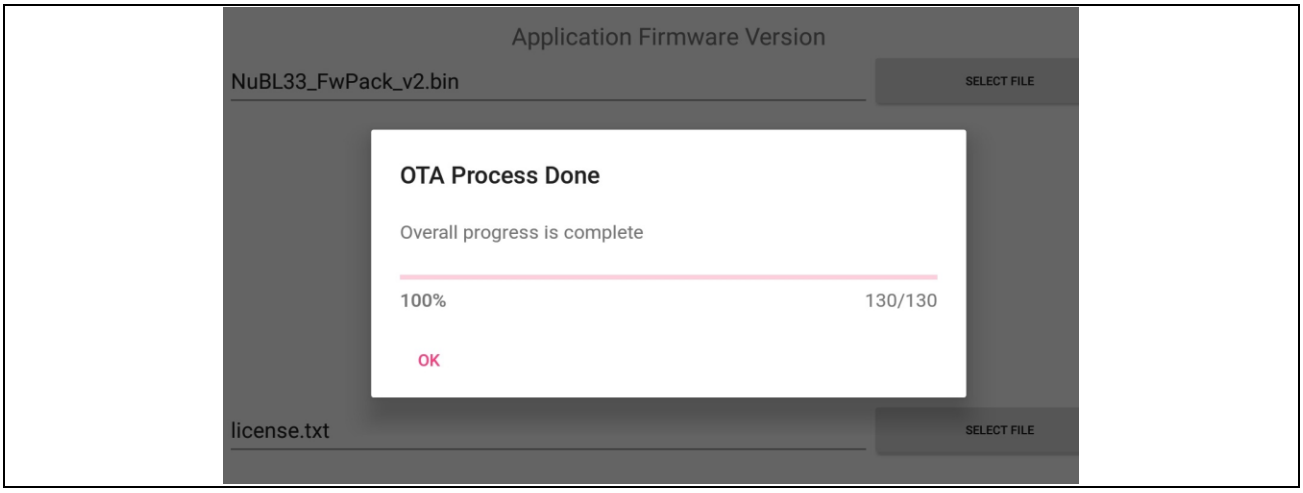

Figure 3-11 OTA Process Done

4 Offline Firmware Update Flow

Figure 4-1 shows the offline firmware update flowchart. This method downloads and writes the new firmware package into the SD card first, and then NuBL2 updates NuBL32/NuBL33 firmware by reading the firmware from the SD card. This method verifies integrity before updating firmware.

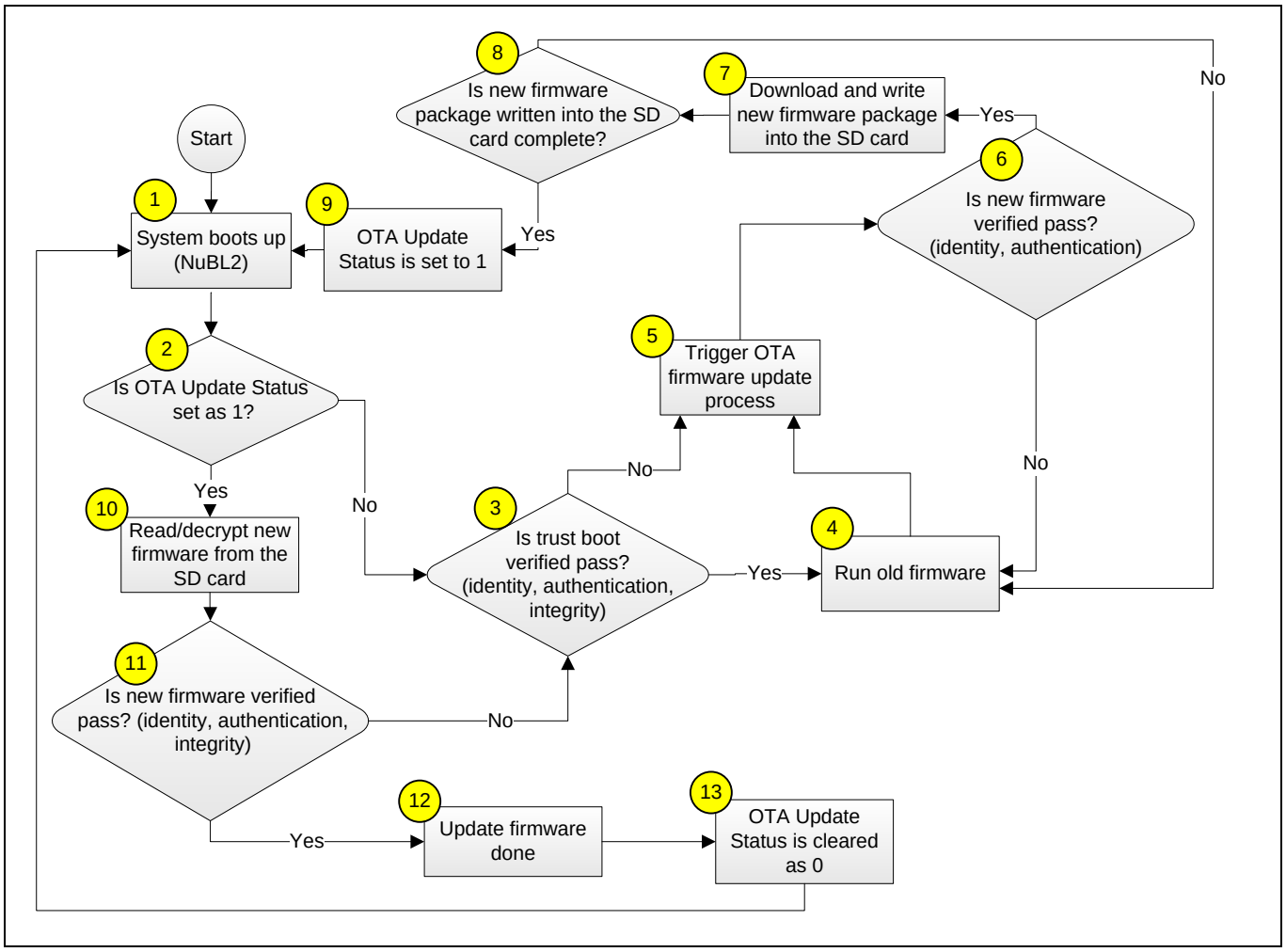

Figure 4-1 NuBL2 Offline Firmware Update Flowchart

5 Conclusion

The secure OTA firmware update mechanism consists of the following features:

- The client and server need to perform authentication process on network connection.
- The new firmware has been encrypted to archive data confidentiality.
- The authenticity of the new firmware is checked by verifying the ECDSA signature.
- The integrity of the new firmware is checked by verifying the firmware's hash value.

These features are based on the functions provided in the Trusted Bootloader (NuBL2). By using the OTA update service of NuBL2, both secure and non-secure firmware can be securely updated through the wireless interface.

Revision History

Date	Revision	Description
2018.10.02	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.