GRPLT Lite Config Tool Customized Firmware Encryption Application Note

Version: 1.2

Release Date: 2022-06-27

Shenzhen Goodix Technology Co., Ltd.

Trademarks and Permissions

Disclaimer

Information contained in this document is intended for your convenience only and is subject to change without prior notice. It is your responsibility to ensure its application complies with technical specifications.

Shenzhen Goodix Technology Co., Ltd. (hereafter referred to as "Goodix") makes no representation or guarantee for this information, express or implied, oral or written, statutory or otherwise, including but not limited to representation or guarantee for its application, quality, performance, merchantability or fitness for a particular purpose. Goodix shall assume no responsibility for this information and relevant consequences arising out of the use of such information.

Without written consent of Goodix, it is prohibited to use Goodix products as critical components in any life support system. Under the protection of Goodix intellectual property rights, no license may be transferred implicitly or by any other means.

Shenzhen Goodix Technology Co., Ltd.

Headquarters: 2F. & 13F., Tower B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 FAX: +86-755-33338099

Website: www.goodix.com

Preface

Purpose

This document introduces how to customize encryption algorithms using the firmware encryption example project in the GR551x Software Development Kit (SDK) to perform customized firmware encryption and start corresponding encryption process, helping users better understand the customized firmware encryption application and related process of GRPLT Lite Config Tool.

Audience

This document is intended for:

- GR551x user
- GR551x developer
- GR551x tester
- Technical writer

Release Notes

This document is the third release of *GRPLT Lite Config Tool Customized Firmware Encryption Application Note*, corresponding to GR551x System-on-Chip (SoC) series.

Revision History

Version	Date	Description
1.0	2021-06-28	Initial release
1.1	2022-02-20	Updated some software interface screenshots.
1.2	2022-06-27	Updated the software name.

Contents

1 Introduction

Compared with Goodix encryption, GRPLT Lite Config Tool customized firmware encryption allows users to adopt customized algorithms and write the keys generated by customized algorithms into GR551x System-on-Chips (SoCs), to independently control encryption logics. During customized firmware encryption application, users can deduce the correctness of the keys according to the encryption algorithms, to ensure smooth operation of the encryption firmware as well as the security of user products.

Before getting started, you can refer to the following documents.

Table 1-1 Reference documents

Name	Description
GRPLT Lite Config Tool User Manual	Introduces installation and operational instructions for GRPLT Lite Config Tool.
GR551x DFU Application Note	Introduces the principles and methods of Device Firmware Upgrade (DFU) for GR551x SoCs.

2 Applying Customized Firmware Encryption

GR551x Software Development Kit (SDK) provides an encryption example project (ble_enc_app_template), in which encryption command response and Golden Unit (GU) serial port interaction logics are implemented. You can apply customized firmware encryption with simple operations.

Note:

- Currently, GRPLT Lite Config Tool customized firmware encryption functionality supports GR551x SDK V1.6.02 only. For other SDK versions, customized firmware encryption may fail because the configurations in the SDK do not match the configurations to be programmed.
- GU refers to a calibrated Bluetooth Low Energy (Bluetooth LE) module.

1. Generate customized encryption firmware. For details, refer to "Section 2.1 Modifying Encryption Algorithm" and "Section 2.2 Compiling Customized Encryption Example Project".
2. Start the customized firmware encryption process. For details, refer to "Section 2.3 Starting Customized Encryption Process".

2.1 Modifying Encryption Algorithm

The custom_enc_info() function in the encryption example project passes in the 16-byte chip_uid to generate 32-byte encryption information through customized encryption algorithms, so enc_key is written into the User Region. You need to modify the corresponding encryption algorithm function in the ble_enc_app_template encryption example project according to the region in which the encryption information will be written.

- To write the encryption information into the User Region of eFuse (first 32 bytes): Modify custom_enc_info() in *enc_key.c*.
- To write the encryption information into Non-Volatile Data Storage (NVDS) or other area of Flash: Modify custom_enc_info() in *enc_key.c* and custom_enc_process() in *custom_enc.c*.

2.2 Compiling Customized Encryption Example Project

After modifying encryption algorithms, you can compile the example project. After compilation, a customized encryption firmware file *ble_enc_app_template_fw.bin* will be generated in `SDK_Folder\projects\peripheral_app\ble_enc_app_template\project\Keil_5\build`.

Note:

SDK_Folder is the root directory of GR551x SDK.

2.3 Starting Customized Encryption Process

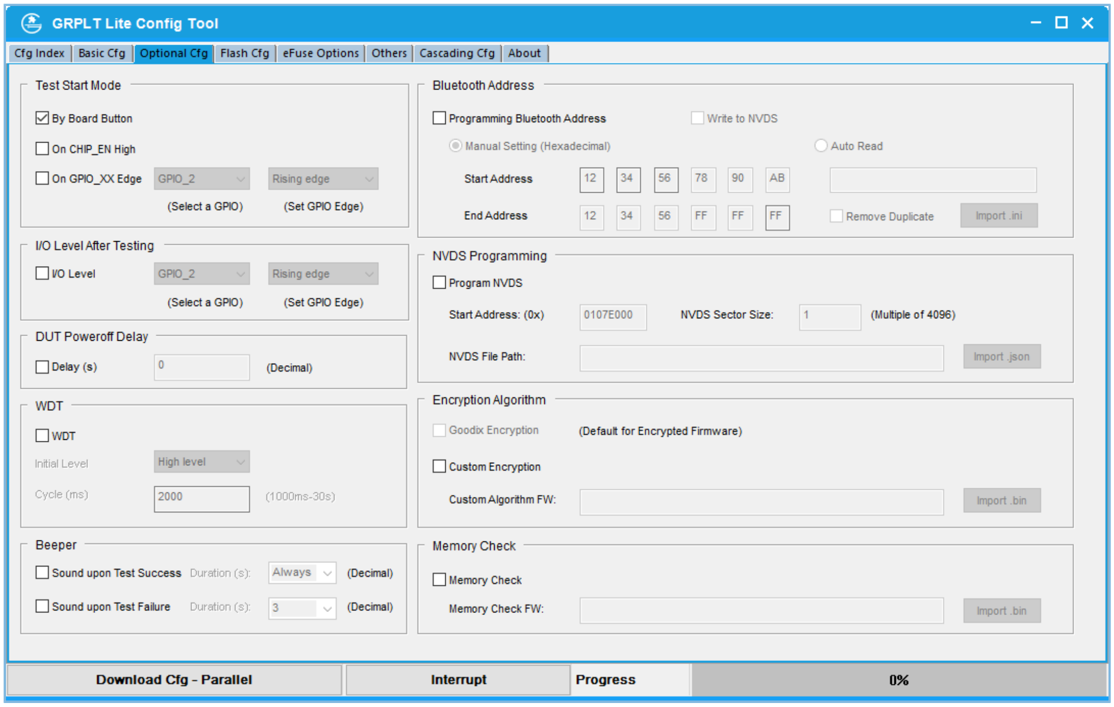
1. Download the customized encryption firmware.

   (1). Copy *ble_enc_app_template_fw.bin* to the corresponding folder in the GRPLT Lite Config Tool software package.

Note:

For the specific folder to which the firmware will be copied, refer to *GRPLT Lite Config Tool User Manual*.

(2). Run *GRPLT Lite Config Tool.exe*; click **Optional Cfg** > **Encryption Algorithm** and select **Custom Encryption**; click **Import .bin** to import *ble_enc_app_template_fw.bin* that is copied to the software package.



Test Start Mode
☑ By Board Button
□ On CHIP_EN High
□ On GPIO_XX Edge GPIO_2 (Select a GPIO) Rising edge (Set GPIO Edge)

I/O Level After Testing
□ I/O Level GPIO_2 (Select a GPIO) Rising edge (Set GPIO Edge)

DUT Poweroff Delay
□ Delay (s) 0 (Decimal)

WDT
□ WDT
Initial Level: High level
Cycle (ms): 2000 (1000ms-30s)

Beeper
□ Sound upon Test Success Duration (s): Always (Decimal)
□ Sound upon Test Failure Duration (s): 3 (Decimal)

Bluetooth Address
□ Programming Bluetooth Address □ Write to NVDS
● Manual Setting (Hexadecimal) ○ Auto Read
Start Address: 12 34 56 78 90 AB
End Address: 12 34 56 FF FF FF □ Remove Duplicate Import .ini

NVDS Programming
□ Program NVDS
Start Address: (0x) 0107E000 NVDS Sector Size: 1 (Multiple of 4096)
NVDS File Path: Import .json

Encryption Algorithm
□ Goodix Encryption (Default for Encrypted Firmware)
□ Custom Encryption
Custom Algorithm FW: Import .bin

Memory Check
□ Memory Check
Memory Check FW: Import .bin

Figure 2-1 To import the customized encryption firmware

(3). Click **Download Cfg - Parallel** in the lower-left corner of GRPLT Lite Config Tool to start downloading the configurations to the selected device under test (DUT).

Upon successful download, close GRPLT Lite Config Tool.

2. Run the encryption firmware and start offline programming.

Connect a PLT Lite board to a DUT, and start offline programming for the customized encryption firmware by pressing K2 or K5 on the PLT Lite board. For the difference between K2 and K5 buttons, refer to *GRPLT Lite Config Tool User Manual*.

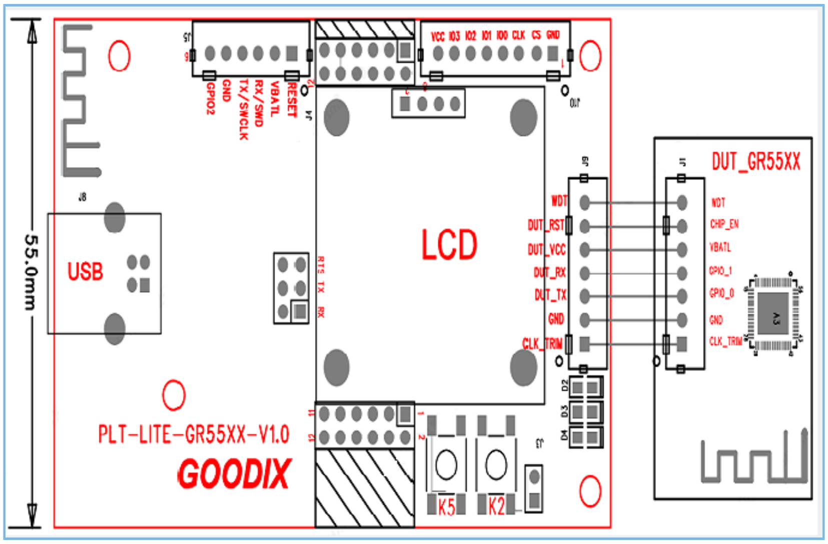


Figure 2-2 In-system programming (ISP) hardware connection

After detecting that the customized encryption firmware runs normally, GU sends the “ENC_DEAL” (0x0401) command to control the firmware to encrypt DUT information and write the encryption key into DUT.

Note:

- In device applications, you can deduce whether the encryption key is correct according to the encryption type. For example, ble_enc_app_template writes two copies of chip_uid into the User Region of eFuse to implement encryption. In device applications, first read the User Region of a GR551x SoC, to check whether there are two chip_uid copies in the User Region.
- For more information about “ENC_DEAL”, refer to “Chapter 4 Customized Encryption Command “ENC_DEAL””.
- DUT refers to a PCB soldered with a GR551x SoC in this document.

  - Both writing and reading back the encryption key succeed: DUT sends success status to GU through serial ports.
  - Writing fails: Return failure status.

For specific customized encryption process and related error messages, refer to “Chapter 3 Customized Encryption Programming Process and Related Error Messages”.

3 Customized Encryption Programming Process and Related Error Messages

Select customized encryption and start offline programming. Then, customized encryption process starts correspondingly. If the customized encryption process is executed successfully, proceed with subsequent process; otherwise, the PLT Lite board shows corresponding error messages on the display and mass production programming stops.

In the offline mass production programming process, steps for customized encryption are provided below:

1. GU downloads the customized encryption firmware to DUT through serial ports. The PLT Lite board shows “StartDown ENC FW” on the display.

   Steps to download the customized encryption firmware:

   (1). GU checks the encryption firmware format.

   If GU detects that the pattern of img_info is incorrect or the load_addr of boot_info in DUT is inconsistent with that in the encryption firmware, the PLT Lite board shows **Down ENC FW Img Check Fail**, indicating that the encryption firmware is in a wrong format.

   (2). GU writes the encryption firmware into DUT according to DFU protocols.

Note:

For more information about DFU commands (PROGRAM_START, PROGRAM_FLASH, PROGRAM_END, and OPERATE_REG) mentioned in this chapter, refer to *GR551x DFU Application Note*.

GU sends “PROGRAM_START” (0x23) to DUT to write the firmware. If DUT fails to respond to the command, the PLT Lite board shows **Down ENC FW Start Error** on the display, indicating that writing the encryption firmware boot information fails.

(3). GU sends “PROGRAM_FLASH” (0x24) to DUT to write the header information of firmware data. If DUT fails to respond to the command, the PLT Lite board shows **Down ENC FW Program Error** on the display, indicating that encryption firmware programming fails.

(4). GU sends “PROGRAM_END” (0x25) to DUT to finish firmware writing. If DUT fails to respond to the command, the PLT Lite board shows **Down ENC FW End Fail** on the display, indicating that encryption firmware check fails.

Note:

If there is no response to any command (including commands related to operating Flash, resetting registers, updating img_info, encrypting response, and erasing Flash) when you download customized encryption firmware to DUT, **Down ENC Info Timeout** will be displayed, indicating that writing user encryption information times out.

---

2. After the encryption firmware is downloaded, GU sends “OPERATE_REG” (0x2C) to control the DUT registers and reset the SoC before encryption, to ensure the encryption firmware can be executed successfully.

   If DUT fails to respond to the command, **Down ENC Info Fail** will be reported, prompting that writing customized encryption information fails.

3. After the encryption firmware is downloaded and executed successfully, GU sends “ENC_DEAL” (0x0401) to write the customized encryption information into the DUT. DUT then checks whether the encryption process is executed successfully.

   - If no: DUT responds with "0x02" (indicating the writing fails) and reports **DownENC Info Error**, prompting that an error occurs in writing the customized encryption information.

   - If yes: Proceed with subsequent programming procedures.

4 Customized Encryption Command “ENC_DEAL”

GU starts writing customized encryption information into DUT by sending “ENC_DEAL”. DUT starts the encryption process after receiving the command, checks whether the command is executed successfully, and responds to GU.

4.1 Data Sent from GU

Table 4-1 Data sent from GU

Byte No.	Description	Valid Value	Remarks
0–1	Frame header	0x4744	Represented by 0x47 and 0x44 which are the ASCII code values of characters ‘G’ and ‘D’
2–3	Frame type	0x0401	Download the customized encryption firmware and start the corresponding encryption process.
4–5	Data length	0x0000	
6–7	Checksum	0x0000–0xFFFF	16-bit checksum for frame type and data length

4.2 Response Data from DUT

Table 4-2 Response data from DUT

Byte No.	Description		Valid Value	Remarks
0–1	Frame header		0x4744	Represented by 0x47 and 0x44 which are the ASCII code values of characters ‘G’ and ‘D’
2–3	Frame type		0x0401	Run the customized encryption firmware and execute the corresponding encryption process.
4–5	Data length		0x0002	
6	Data content	Response	0x03	0x03: DUT responds with 0x0401.
7		Execution result	0x01/0x02	0x01: Succeeded. 0x02: Failed.
8–9	Checksum		0x0000–0xFFFF	16-bit checksum for frame type, data length, and data content (response and execution result)