

TLE9893_2QTW62S_SECURE_ACCESS_USING_AES

About this document

Scope and purpose

The aim of this guide is to present the scope, the implementation, the algorithm and a demonstration of the **TLE9893_2QTW62S_SECURE_ACCESS_USING_AES** example code for the TLE989x Infineon Embedded Power ICs based on Arm® Cortex® M3. This example code can be found in the Keil µVision Pack Installer.

The full functionalities and characteristics of the embedded power devices are described in the datasheets and user's manual. Please refer to these documents for more detailed information. Furthermore, a low level (line-by-line) description of the code is not the aim of this document, although occasionally some codeblocks might be reported if necessary to the comprehension.

Note: The following information is given as a hint for the implementation of the system only and shall not be regarded as a description or warranty of a certain functionality, condition or quality of the referred devices or presented software example.

Intended audience

Design engineers, system engineers, embedded power designers

Table of contents

| | |
|--|-----------|
| About this document..... | 1 |
| Table of contents..... | 1 |
| 1 Introduction | 2 |
| 2 Hardware | 4 |
| 3 Implementation | 6 |
| 3.1 Get the example via the Pack Installer for Keil..... | 6 |
| 3.2 Configuration..... | 6 |
| 3.3 Sample code flow | 7 |
| 3.4 Sample code for AES user API | 9 |
| References..... | 10 |
| Revision history..... | 11 |

1 Introduction

Figure 1 shows the basic secure access mechanism implemented and the user api call sequence. The tool (via UART) must send random seed along with the secure access request service Id. The device will use the cryptolibrary and performs the aes encryption and sends positive response. The tool shall send the key request with encrypted data. The device will verify the received data and provides either +ve or -ve response. Figure 2 shows the UART output.

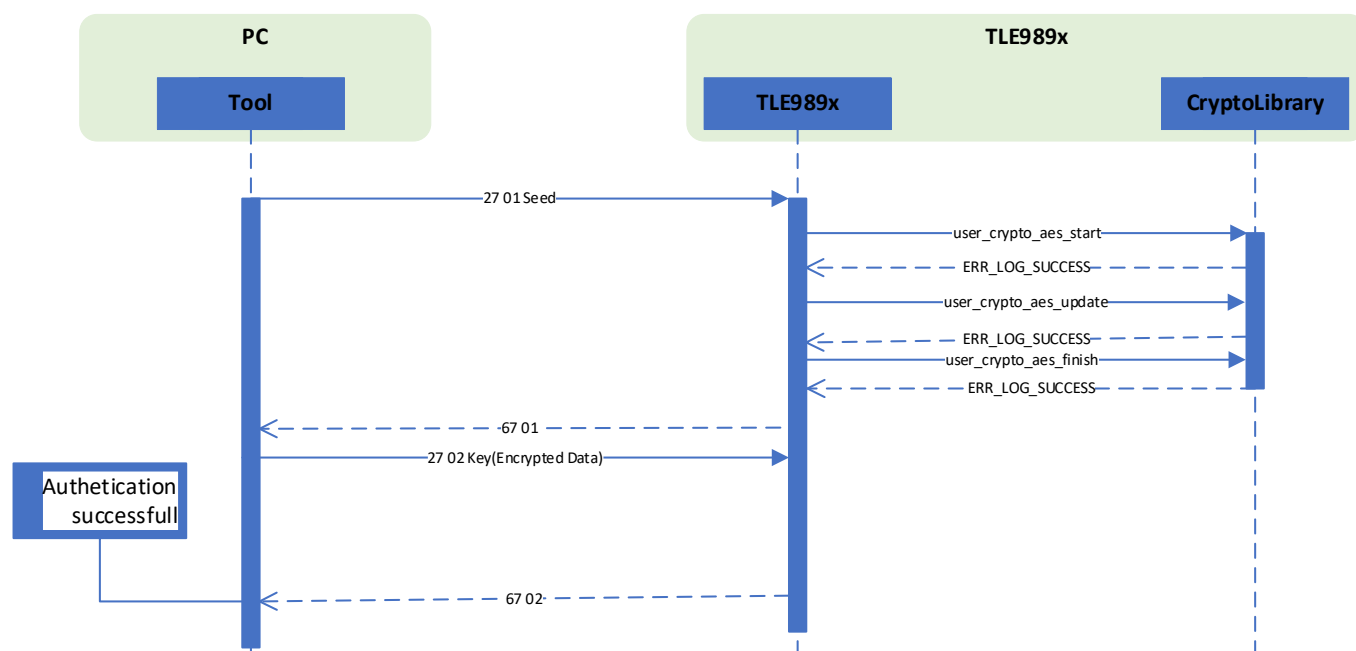
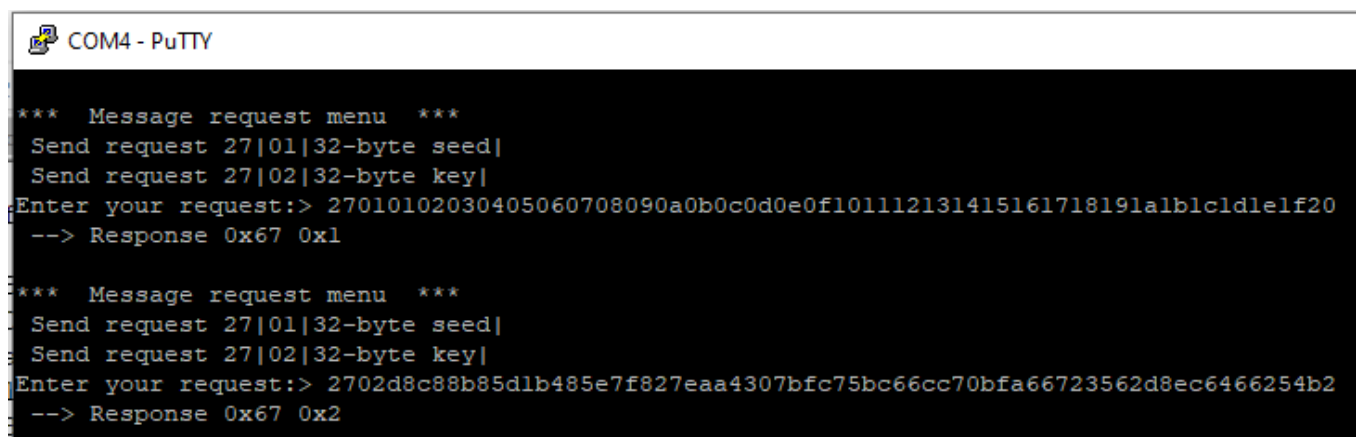


Figure 1 Secure access mechanism



```

COM4 - PuTTY

*** Message request menu ***
Send request 27|01|32-byte seed|
Send request 27|02|32-byte key|
Enter your request:> 27010102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20
--> Response 0x67 0x1

*** Message request menu ***
Send request 27|01|32-byte seed|
Send request 27|02|32-byte key|
Enter your request:> 2702d8c88b85d1b485e7f827eaa4307bfc75bc66cc70bfa66723562d8ec6466254b2
--> Response 0x67 0x2
  
```

Figure 2 UART Output

Example:

Seed request: 27010102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20

Key request: 2702d8c88b85d1b485e7f827eaa4307bfc75bc66cc70bfa66723562d8ec6466254b2

■ Service id
 ■ subfunction
 ■ 32-byte hex encoded seed or encrypted key

Note:

- Encrypted key is AES cipher text of the seed (using the symmetric key stored in the device). The length of default symmetric key is 16 bytes (128 bits).

Default symmetric key: ffffffffffffffffffffffffffffffff

The value of symmetric key (16 or 32 bytes) can be stored in NVM using the example SECURITY_WRITE_KEY_EXAMPLE_TLE989X.

- Complete access request string can be copy pasted to the Putty terminal and after pressing the “Enter” key sent command will be echoed to the terminal.

2 Hardware

This chapter shows how to run the TLE9893_2QTW62S_SECURE_ACCESS_USING_AES example with the TLE988x/TLE989x evaluation board. For this the project must be opened and compiled.

Figure 3 shows the TLE988x/TLE989x evaluation board. The application code must be loaded via a debugger (e.g. ULINK or J-Link) to the board. The board must be powered with 12V (red and black connections).

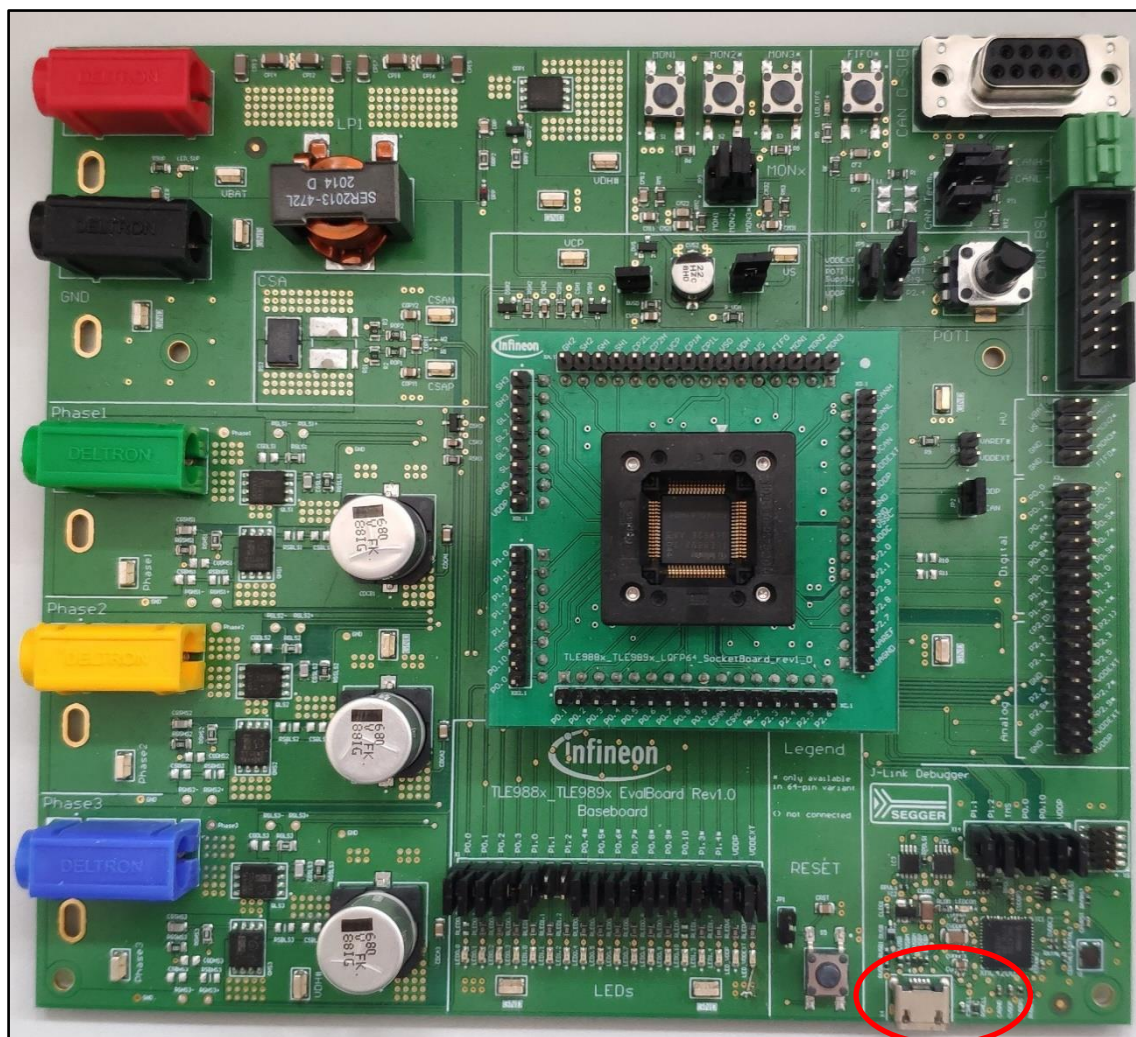


Figure 3 TLE988x/TLE989X evaluation board

Alternatively, a USB connection can be established to a local PC, which emulates a virtual COM port. The relevant COM device number can be identified via the Device Manager on Windows systems or the dmesg tool on Unix based operating systems.

In order to show the output on a command console, free tools like Putty or TeraTerm can be used. The UART1 in this example is configured with:

- a transmission baud rate of 115200,
- 8 data bits,
- 1 stop bit,
- no parity and no flow control.

3 Implementation

This chapter shows the process to follow to get a working secure access simple example.

3.1 Get the example via the Pack Installer for Keil

Open the Pack Installer within the Keil IDE.

Choose the appropriate device (here TLE9893_2QTW62S) on the left-hand side. On the right-hand side, select the tab Examples, where you can access the TLE98932QTW62S_SECURE_ACCESS_USING_AES example.

Clicking on “Copy” will copy the example on your computer and open it.

| Software Component | Sel. | Variant | Version | Description |
|--------------------|-------------------------------------|--------------|---------|--|
| ⊕ CMSIS | | | | Cortex Microcontroller Software Interface Components |
| ⊕ CMSIS Driver | | | | Unified Device Drivers compliant to CMSIS-Driver Specifications |
| ⊖ Compiler | | ARM Compiler | 1.6.0 | Compiler Extensions for ARM Compiler 5 and ARM Compiler 6 |
| Event Recorder | <input type="checkbox"/> | DAP | 1.4.0 | Event Recording and Component Viewer via Debug Access Port (DAP) |
| I/O | | | | Retarget Input/Output |
| File | <input type="checkbox"/> | File System | 1.2.0 | Use retargeting together with the File System component |
| STDERR | <input type="checkbox"/> | Breakpoint | 1.2.0 | Stop program execution at a breakpoint when using STDERR |
| STDIN | <input checked="" type="checkbox"/> | User | 1.2.0 | Retrieve STDIN from a user specified input source (USART, Keyboard or other) |
| STDOUT | <input checked="" type="checkbox"/> | User | 1.2.0 | Redirect STDOUT to a user defined output target (USART, Graphics Display or other) |
| TTY | <input type="checkbox"/> | Breakpoint | 1.2.0 | Stop program execution at a breakpoint when using TTY |
| ⊕ Device | | | | Startup, System Setup |
| ⊕ File System | | MDK-Plus | 6.13.8 | File Access on various storage devices |
| ⊕ Graphics | | MDK-Plus | 6.10.8 | User Interface on graphical LCD displays |
| ⊕ Network | | MDK-Plus | 7.14.0 | IPv4 Networking using Ethernet or Serial protocols |
| ⊕ USB | | MDK-Plus | 6.14.1 | USB Communication with various device classes |

Figure 4 RTE settings for stdout and stdin

In order to redirect the stdout functions - the printf call in the example, adjust the runtime environment setting for the compiler within the Keil IDE. Select the option “User” under Compiler -> I/O -> STDOUT (see Figure 4).

In order to redirect the stdin functions - the `stdin_getchar` call in the example, adjust the runtime environment setting for the compiler within the Keil IDE. Select the option "User" under Compiler -> I/O -> STDIN (see Figure 4).

3.2 Configuration

In order to configure the UART module for the TLE9893_2QW62S_SECURE_ACCESS_USING_AES example, select the UART tab. Enable the UART1 module. Next, select the 8-bit UART mode with variable baudrate. The baudrate is set to 115200 in the blue box Baudrate Generator Settings. This is one of the common speed settings for the UART. In the pink box Transmission Settings, select the pin P1.1. In the green box Reception Settings, select the pin P1.2 and set enable receiver of serial port radio box. In the yellow box Interrupt, enable both receive interrupt radio boxes and rewrite an intuitive name for the interrupt service routine, here

uart_receive. See Figure 5 Config Wizard, module UART for more details

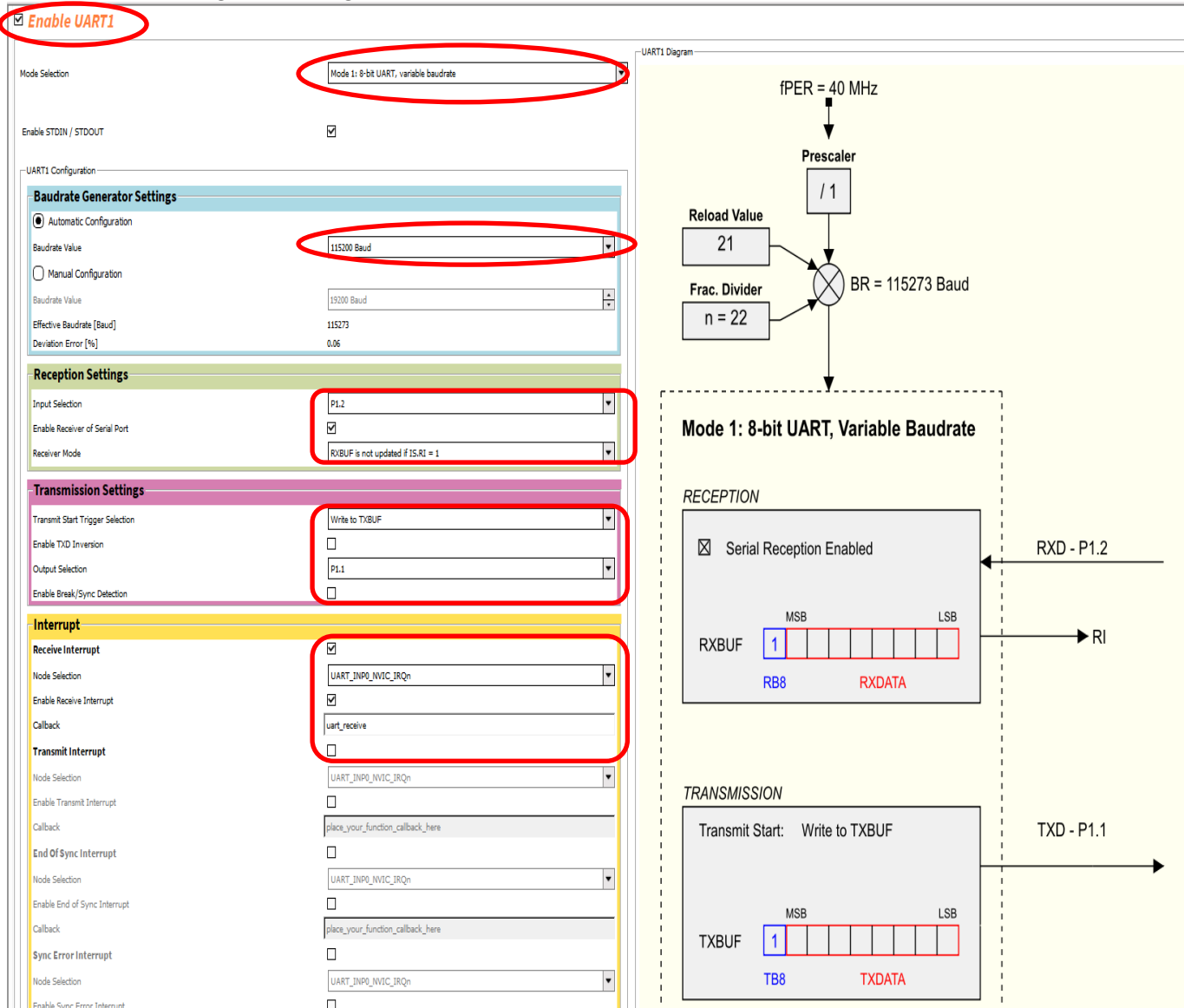


Figure 5 Config Wizard frequency configuration

Finally, save your configuration to take these changes into account (File -> Save).

3.3 Sample code flow

After setting the connection with PC using virtual COM via Putty for instance, upon reset device with send the uart message for sample commands format for secure access reuest. Then the sequence for secure access is as follows:

- First UART tool will send the secure access request id that is 0x27 0x01 (defined in UDS), along with 32 bytes of random seed data. The device validates the request ID and data length, and passes this information to the cryptolibrary to calculate the cipher text. If every thing goes well it sends the positive response, else it will send the Negative Response Code (NRC).
- In second step, UART tool will send the key authentication request i.e (0x27 0x02), with cipher text. Device verifies the key based on the received cipher text value, and provides the positive or negative response.

All received NRCs (Negative Response Code) are based on the standard UDS protocol.

Figure 6 depicts the code flow implemented in the sample example. This is based on the UDS protocol.

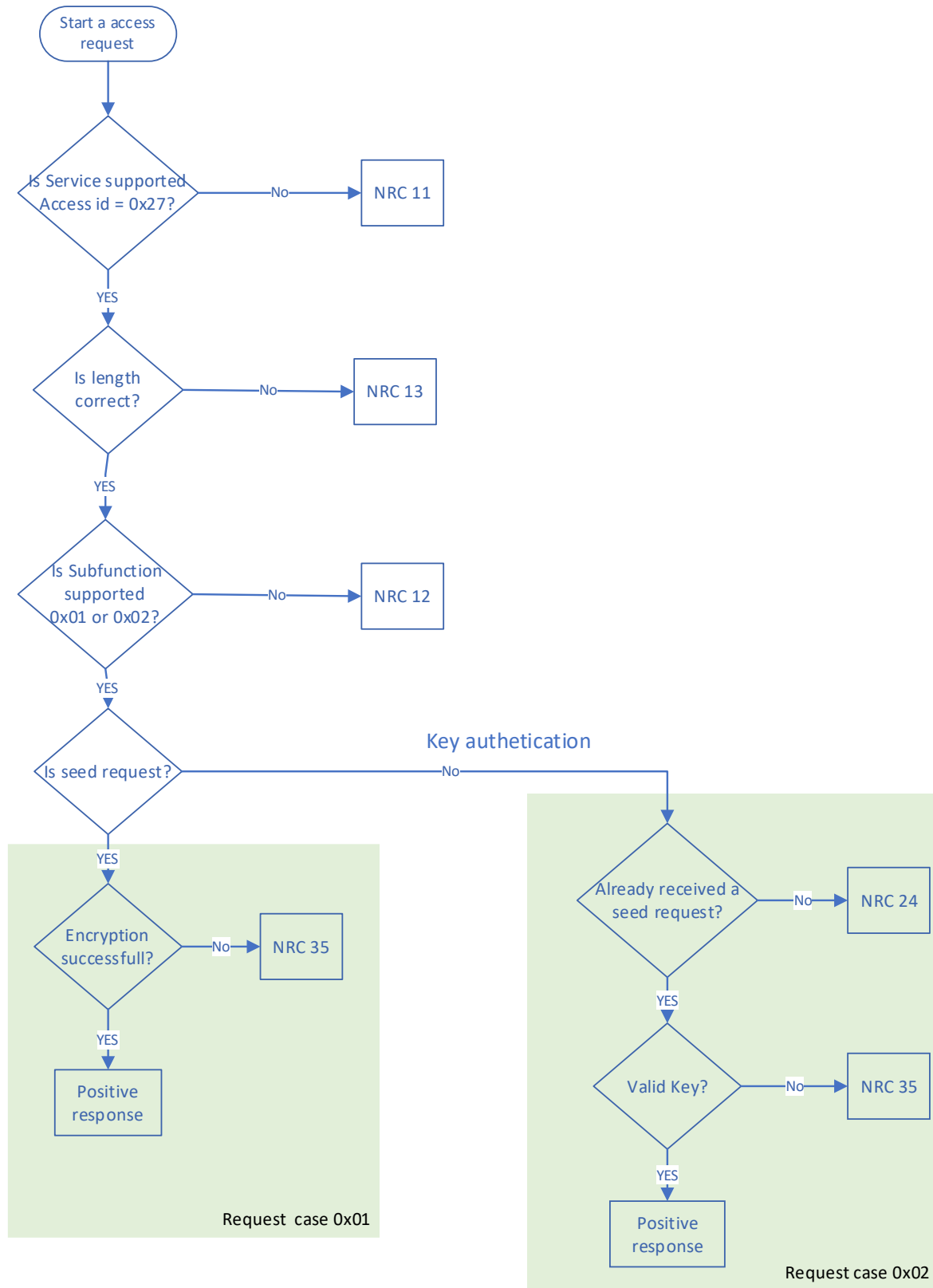


Figure 6 TLE9893_2QTW62S_SECURE_ACCESS_USING_AES application code

3.4 Sample code for AES user API

Figure 7 shows the application code of the user api usage.

The encryption method is set as `CRYPTO_ECB_ENCRYPT` and the key id iused here is slot 1. (Note: Key slot 0 is reserved)

Step 1: call `user_crypto_aes_start` user API function which initializes an ECB encryption operation.

Step 2: call `user_crypto_aes_update` user API function which updates an ongoing ECB encryption operation. The pointer to the minimum output length must be provided along with the input buffer.

Step 3: call `user_crypto_aes_finish` user API function which finishes an ongoing ECB encryption operation and updates the output buffer and length.

```

/**@brief this api initializes encryption */
s32_status = user_crypto_aes_start(fid, u32_keyId, NULL_PTR);

if (s32_status == ERR_LOG_SUCCESS){

    s32_status |= user_crypto_aes_update(sp_aesIO);
    u32_totalLength += *(sp_aesIO->out.length);
    u32_encodeOutLength = REQUEST_LENGTH - 2;

    sp_aesIO->inp.buffer = sp_aesIO->inp.buffer + AES_UPDATE_LENGTH;
    /**@brief reset the input length */
    sp_aesIO->inp.length = 0;
    sp_aesIO->out.buffer = (uint8 *) (sp_aesIO->out.buffer + u32_totalLength);
    sp_aesIO->out.length = &u32_encodeOutLength;
    /**@brief updates ongoing CBC or ECB encryption or decryption operation */
    s32_status |= user_crypto_aes_finish(sp_aesIO);

}
else{
    /**@brief finish api must be called even if any other APIs returns error */
    s32_status |= user_crypto_aes_finish(sp_aesIO);
}

```

Figure 7 TLE9893_2QTW62S_SECURE_ACCESS_USING_AES application code

References

See the code examples at www.infineon.com

Revision history

| Document version | Date of release | Description of changes |
|------------------|-----------------|------------------------|
| 1.0 | 2021-11-04 | Initial version |
| 1.1 | 2022-10-13 | Editorial changes |

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2022-10-13

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2022 Infineon Technologies AG.

All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.