STM32H7Sxx security guidance for SESIP 3 Certification

Introduction

This document describes how to prepare STM32H7Sxx microcontrollers to make a secure system solution compliant with the SESIP 3 standard using the STM32CubeH7RS MCU Package.

The STM32H7S78-DK board integrating the STM32H7S7Z8 microcontroller is used as the hardware vehicle to implement and test a secure application executed through the STM32H7S7Z8 microcontroller immutable RoT but it does not bring any additional security mechanism.

The security guidance described in this document applies to any boards based on STM32H7Sxx microcontrollers.

1 General information

The STM32CubeH7RS MCU Package runs on STM32H7Sxx 32-bit microcontrollers based on the Arm® Cortex®-M processor.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

The following table presents the definition of acronyms that are relevant for a better understanding of this document.

Table 1. List of acronyms

Acronym	Description
CLI	Command-line interface
GUI	Graphic user interface
HDP	Secure hide protection
AHK	Application hardware key, generated during the HDLP0 provisioning process
HW	Hardware
MCE	Memory cipher engine
MPU	Memory protection unit
PSA	Platform security architecture. Framework for securing devices.
RoT	Root of Trust
STiROT	ST immutable application inside the STM32H7S7Z8 microcontroller managing secure boot and secure firmware update of an application installed in the STM32H7S7Z8 microcontroller user flash memory
User application	Application located inside the user flash started by STiROT after verifying its integrity and authenticity. Not part of TOE.
SESIP	Security evaluation standard for IoT platforms
iLoader	Software responsible for transferring the firmware image between internal RAM and external flash memories
SW	Software
TOE	Target of evaluation
DA	Debug Authentication

2 Reference documents

Table 2. List of reference documents

Name	Title/Description
RM0477	Reference manual *STM32H7Rx/Sx Arm®-based 32-bit MCUs* (RM0477) revision 6
AN4992	Application note *Introduction to secure firmware install (SFI) for STM32 MCUs* (AN4992) revision 16
UM2237	User manual *STM32CubeProgrammer software description* (UM2237) revision 24
TN1504	Technical note *Security Target for STM32H7Sxx compliant with SESIP3* (TN1504) revision 3
[IEE1149]	EEE 1149.1–2013
[ADI5]	Arm Debug Interface Architecture Specification ADIv5.0 to ADIv5.2
[Debug Authentication]	Arm document Authenticated Debug Access Control - DEN0101 V00BET1

3 Preparative procedures

This chapter describes the procedures to prepare the environment and the product before starting to use the product or before testing the product:

- Secure acceptance: procedures to check the product to be tested.
- Secure preparation of the operational environment: procedures to set up the environment needed to manage and test the product.
- Secure installation: procedure to program and configure the product to be tested.
- Tera Term connection preparation procedure: procedure to configure the Tera Term tool before starting to test the product.

3.1 Secure acceptance

Secure acceptance is the process in which the user securely receives the TOE and verifies the integrity and authenticity of all its components.

The TOE is distributed as an MCU.

To ensure that MCU is not manipulated during TOE delivery, the integrator must verify that the user flash memory is virgin (reading 0xFF everywhere with the STM32CubeProgrammer).

Note that it is the responsibility of the integrator to choose the correct STM32CubeH7RS MCU Package version.

How to accept the STM32H7Sxx microcontroller: by reading, with STM32CubeProgrammer (for more details, refer to UM2237), all items depicted below:

- Device ID: 485, meaning STM32H7Sxx
  - Address: `0x5C00 1000`
  - Type: `half-word`
  - Value: `0xX485`
- Revision: v1.2
  - Address: `0x5C00 1002`
  - Type: `half-word`
  - Value: `0x1003`
- Product configuration:
  - Address `0x5200 281C`
  - Type: `word`
  - Value:
    - Bit 8 (SAES, CRYP, PKA available): `0b0`
    - Bits 16, 17, 18 (MCE available): `0b000`
- Immutable firmware versions:
  - TOE in configuration TOE_WITH_STIROT:
    - STIROT version: v1.1.0
      - Address: `0x1FF1FDCC`
      - Type: `word`
      - Value: `0xFF010100`
    - Debug Authentication version: v1.0.0
      - Address: `0x1FF1FDC8`
      - Type: `word`
      - Value: `0xFF010000`
    - Security library version: v1.1.0
      - Address: `0x1FF1FDB0`
      - Type: `word`
      - Value: `0xFF010100`
    - RSS version: v1.1.0
      - Address: `0x1FF1FDAC`
      - Type: `word`
      - Value: `0xFF010100`
  - TOE in configuration TOE_WITHOUT_STIROT:
    - Debug Authentication version: v1.0.0
      - Address: `0x1FF1FDC8`
      - Type: `word`
      - Value: `0xFF010000`
    - Security library version: v1.1.0
      - Address: `0x1FF1FDB0`
      - Type: `word`
      - Value: `0xFF010100`
    - RSS version: v1.1.0
      - Address: `0x1FF1FDAC`
      - Type: `word`
      - Value: `0xFF010100`

3.2 Secure installation and secure preparation of the operational environment (AGD_PRE.1.2C)

Installation and secure preparation of the TOE correspond to generating the configuration files and loading them into the MCU memory. In the case of the STM32H7S78-DK development board, this can be performed using the STM32TRustedPackageCreator tool and then the STM32CubeProgrammer tool connected to the target via USB. Examples of configuration files and scripts are provided in the STM32CubeH7RS MCU Package, which can be downloaded from the STMicroelectronics website.

This section describes the hardware and software setup procedures.

3.2.1 Hardware setup procedure

To set up the hardware environment, the STM32H7S78-DK development board must be connected to a personal computer via a USB cable. This connection with the PC allows the user to:

- Flashing the board
- Interacting with the board via a UART console
- Debugging when the protections are disabled

The ST-LINK firmware programmed on the development board must be the V3J10M3 version.

3.2.2 Software setup procedure

This section lists the minimum requirements for the developer to set up the SDK on a Windows® 10 host, run the sample scenario and customize applications delivered in the STM32CubeH7RS MCU Package.

STM32CubeH7RS MCU Package

Download the STM32CubeH7RS MCU Package on the Windows® host hard disk, for example at `C:\data`, or any other path that is short enough and without any space.

Development toolchains and compilers

Software components part of the certified products are all integrated into the STM32H7Sxx microcontroller but a set of IDE project examples are delivered inside the STM32CubeH7RS MCU Package. Refer to the STM32CubeH7RS release note to get details about the development toolchains and compilers supported by the STM32CubeH7RS MCU Package releases are available on the STMicroelectronics website.

Software tools for programming STM32 microcontrollers

The STM32CubeProgrammer (STM32CubeProg) is an all-in-one multi-OS software tool for programming STM32 microcontrollers. It provides an easy-to-use and efficient environment for reading, writing, and verifying device memory through both the debug interface (JTAG and SWD) and the bootloader interface (UART and USB).

The STM32CubeProgrammer offers a wide range of features to program STM32 microcontroller internal memories (such as flash memory, RAM, OTP, and OB keys) as well as external memories. The STM32CubeProgrammer also allows option programming and upload, programming content verification, and microcontroller programming automation through scripting.

The STM32CubeProgrammer is delivered in GUI (graphical user interface) and CLI (command-line interface) versions. The STM32CubeProgrammer tool version to use is version v2.16.0.

For more details about the STM32CubeProgrammer, refer to UM2237.

Software tools for package creation

STM32TrustedPackageCreator is an all-in-one multi-OS software tool for package creation. For STM32H7Sxx, it provides an easy-to-use and efficient environment for generating:

- STiRoT and Debug Authentication configuration files (`.obk`) based on the XML template provided in the STM32CubeH7RS MCU Package
- Firmware and data images based on firmware and data binaries with the addition of a header and security parameters. The integrator must use STM32TrustedPackageCreator to build its firmware and data images in the format expected by the TOE.
- Debug Authentication certificate chain

STM32TrustedPackageCreator is a component of the STM32CubeProgrammer delivery pack and is installed by the STM32CubeProgrammer installer.

Terminal emulator

A terminal emulator software is needed to run the user application delivered as an example inside the STM32CubeH7RS MCU Package. It allows displaying the menu and interacting with the user application to download a new version of the user application.

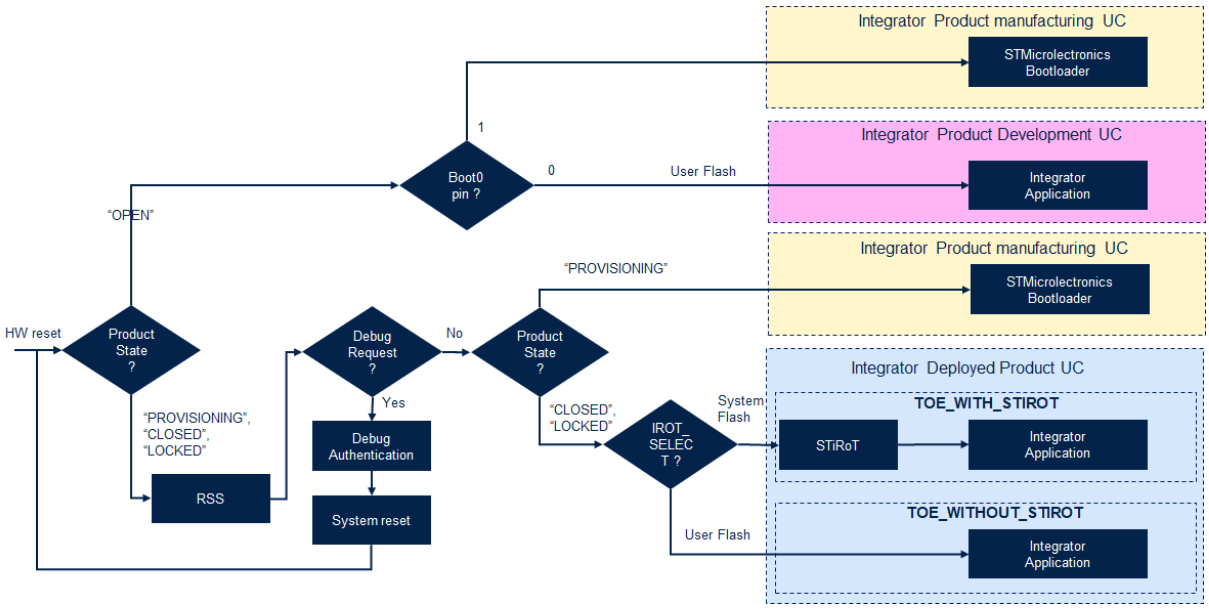
The example in this document is based on Tera Term, an open-source free software terminal emulator that can be downloaded from the https://osdn.net/projects/ttssh2/ webpage. Any other similar tool can be used instead.

3.3 Secure installation

As described in TN1504 and illustrated in the figure below, the TOE has been certified in two different TOE configurations:

- TOE_WITH_STIROT: Configuration using all TOE functionalities including the STiRoT functionalities
- TOE_WITHOUT_STIROT: Configuration using all TOE functionalities except the STiRoT functionalities

Figure 1. TOE configurations



An example of provisioning scripts as well as an example of user application including postbuild scripts are provided in the STM32CubeH5 MCU Package to guide the user during the STM32H7Sxx product preparation.

Secure installation of "TOE_WITH_STIROT"

The STM32H7Sxx product preparation is done in four steps. To get a complete installation with security fully activated, the four steps must be done as the STM32H7Sxx static security configuration is programmed only at the very last step:

- Step 0: Preliminary stage: At this step, the selection of the product state and the method of Debug Authentication (password or certificate) as well as the generation of iLoader is requested.
- Step 1: STiRoT and DA configuration. At this step, STiRoT and DA configuration files are generated using STM32TrustedPackageCreator. Depending on the STiRoT configuration chosen, project files (linker files) as well as the static security protection script are automatically updated.
- Step 2: Firmware image generation
  - The firmware image is generated with the postbuild command executed at the end of the compilation of the user application project.
- Step 3: Provisioning. At this step:
  - The static security protection script is executed.
  - iLoader is programmed in internal flash flash memory.
  - Firmware image is downloaded in external flash memory.
  - The product state is changed to “PROVISIONING” and configuration data are flashed into OB keys.
  - The final product state (“CLOSED” or “LOCKED”) is set. To protect the complete product including the user application, the final product state must be “CLOSED” or “LOCKED”.

For this TOE configuration, the static security protection script includes the configuration of the following option bytes:

- Write protection (WRPSx): write protection on the iLoader software binary.
- Memory boot (IROT_SELECT): enabled, boot in system flash memory
- ECC management (ECC_ON_SRAM): enabled or disabled
- Product state (PRODUCT_STATE): “CLOSED” or “LOCKED”

Refer to Section 5.1: Secure installation of "TOE_WITH_STIROT" step by step for the description of the three steps of the secure installation procedure.

Secure installation of "TOE_WITHOUT_STIROT"

The STM32H7Sxx product preparation is done in three steps. To get a complete installation with security fully activated, the three steps must be done as the STM32H7Sxx static security configuration is programmed only at the very last step:

- Step 1: DA configuration files generation. Depending on the integrator's user application, additional configuration files could be generated at this step.
- Step 2: Option bytes programming. The static security protection must be configured.
- Step 3: Image flashing and OB keys provisioning. At this step:
  - The product state is changed to “PROVISIONING” and configuration data are flashed into OB keys.
  - The final product (“CLOSED”, or “LOCKED”) is set. To protect the complete product including the user application, the final product state must be “CLOSED” or “LOCKED”.

For this TOE configuration, the static security protection script includes the configuration of the following option bytes:

- Memory boot (ROT_SELECT): disabled, boot in user flash memory
- Product state (PRODUCT_STATE): “CLOSED” or “LOCKED”

Refer to Section 5.2: Secure installation of “TOE_WITHOUT_STIROT“ step by step for the description of the three steps of the secure installation procedure.

4 Operational user guidance

4.1 User roles

The following user roles are distinguished for this TOE:

- Integrator

The integrator is the person to receive the TOE and perform the preparative procedures as described in Section 3: Preparative procedures, and integrates the TOE into a full IoT solution. The user operational guidance is described in Section 4.2: Operational guidance for the integrator role.

The integrator is responsible for personalizing the product data and configuring the security of their product following the guidelines provided in this document.

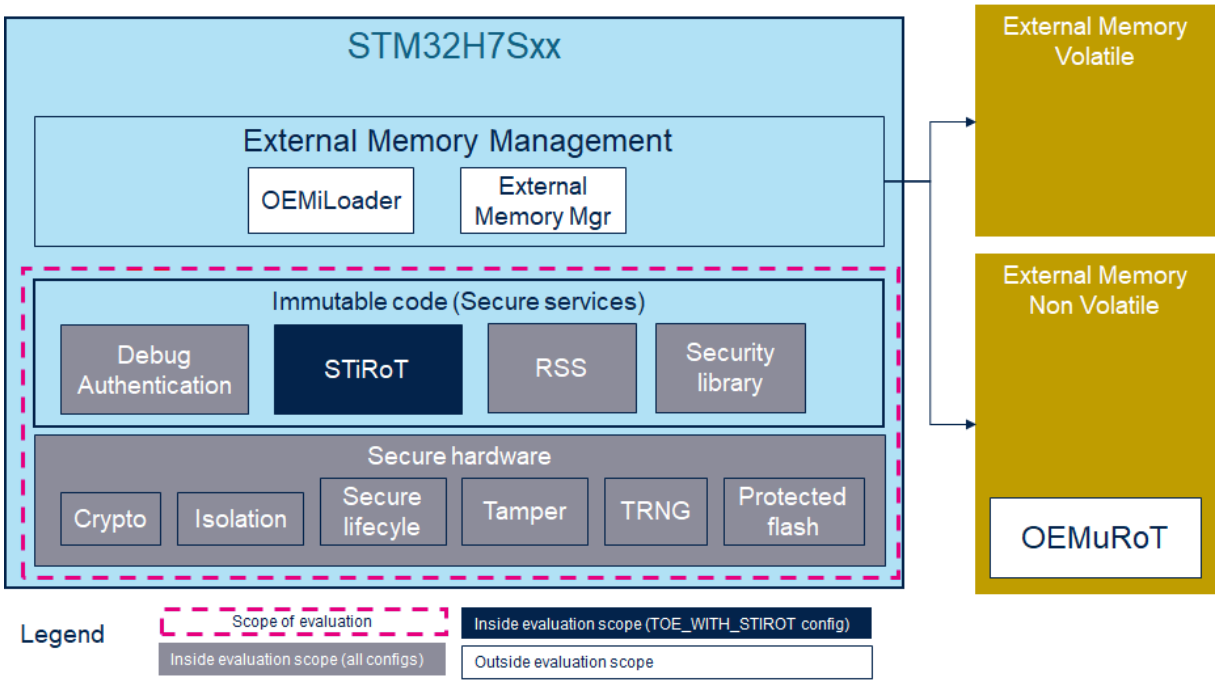
The integrator has full access to the STM32H7Sxx chip security features. The STM32H7Sxx chip is delivered in a virgin state without any security features activated that are integrated into its board. It has full access to the tools needed to program the TOE.

4.2 Operational guidance for the integrator role

4.2.1 User-accessible functions and privileges (AGD_OPE.1.1C)

The main task of the integrator is to integrate the TOE into a full solution. To this end, the system integrator has access to interfaces that are unavailable to other users, as described in Section 4.2.2: Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C). The integrator cannot change any parts inside the TOE but must configure the TOE to make it functional in its certified configurations. The integrator can change parts outside the TOE without compromising the security of the TOE.

Figure 2. TOE scope



Follow the procedures described in Section 3.1: Secure acceptance to check the TOE and follow the procedures described in Section 3.2 to configure the TOE. As mentioned in TN1504 and Section 3.2, the TOE is certified in two configurations ("TOE_WITH_STIROT" and "TOE_WITHOUT_STIROT"). The integrator must first define the TOE-certified configuration compatible with its product security architecture and compatible with its product security requirements. The TOE personalization depends on the TOE-certified configuration selected by the integrator. Once the TOE is personalized in one of its certified configurations, the integrator can still change parts outside the TOE and can still update the product state configuration. This section describes the TOE personalization that the integrator must do. It clarifies if there is any impact on the certification of the TOE or any impact on the functional level of the TOE.

The integrator must follow the guidelines described in this section. A failure to do so means that the TOE is not used in the certified configuration or might not be functional.

TOE personalization when using TOE configuration “TOE_WITH_STIROT”

The integrator has the privilege and responsibility of personalizing the TOE to make it functional. As described in Section 5.1: Secure installation of "TOE_WITH_STIROT" step by step, the TOE personalization is done in different parts:

- In HDPL1 OB keys,
- In STM32H7Sxx option bytes,
- And in the application image itself.

Personalization in HDPL1 OB keys:

- Debug Authentication configuration: The Integrator has the flexibility to activate the Debug Authentication features of the TOE with a certificate or password method. To do so, the Integrator must provision Debug Authentication root parameters (password or Debug Authentication key and permissions capabilities) inside the TOE (refer to Section 5.1: Secure installation of "TOE_WITH_STIROT" step by step to get details on permissions capabilities, details on debug-reopening, regression, and force download capabilities). The Integrator also has the flexibility to not activate the Debug Authentication capabilities (meaning no Debug Authentication configuration provisioned inside the TOE). In the certified configuration, the Debug Authentication features of the TOE must be configured with the certificate method.
- STiROT cryptographic keys: The Integrator has the privilege and the responsibility of configuring cryptographic keys used by the TOE to authenticate and decrypt any new firmware or any new data candidate images. Any failure in this responsibility does not compromise the security of the TOE and is still the certified product, however, it can result in a nonfunctional TOE (meaning it is not possible to boot an installed image or to install/update any new firmware or any new data candidate).
- Application firmware slots configuration: The Integrator has the privilege and the responsibility to change the location and the size of the firmware slots:
  - Firmware slot in RAM: Memory area where a verified application is executed and where a new candidate image is copied for installation.
  - Firmware installation slot in an external flash: Memory area where a verified application code is installed.
  - Firmware download slot in an external flash: Memory area in external flash memory where a new candidate firmware image must be downloaded to be automatically detected by the STiROT.

  Those application firmware slot configurations must comply with the memory alignment constraints as described in Section 5.1.6: Details on `STiROT_Config.obk`. Any failure in this responsibility does not compromise the security of the TOE and is still the certified product, however, it can result in a nonfunctional TOE (meaning it is not possible to boot an installed image or to install/update any new firmware candidate).
- Product state minimum: TOE must be configured to be functional only if the product state defined in the STM32H7Sxx option bytes is higher than the product state defined in the HDPL1 OB keys. The Integrator has the privilege and the responsibility to define with which minimum product state the TOE is "allowed" to be executed. Any failure (meaning consistency issue) in this responsibility does not compromise the security of the TOE and is still the certified product, however, it results in a nonfunctional TOE (meaning it is not possible to boot any installed image even if it is a correct image)

- ECC_ON_SRAM security configuration: TOE can be configured to be functional only if “Error Code Correction on SRAM” is activated. The Integrator has the privilege and the responsibility to define in which configuration the TOE is “allowed” to be executed. As described in Section 5.1: Secure installation of "TOE_WITH_STIROT" step by step, this configuration must be consistent with the configuration defined in the STM32H7Sxx option bytes. Any consistency issues between the option bytes and HDPL1 OB keys result in a nonfunctional TOE (meaning it is not possible to boot any installed image even if it is a correct image). The TOE is certified in both configurations (ECC_ON_SRAM activated or deactivated), however, the most robust configuration of the TOE is when ECC_ON_SRAM security protection is activated.
- High-speed boot (64 or 380 MHz) capability: TOE can be configured to run the STiROT at 64 or 380 MHz (the best speed supporting the full range of temperature in all hardware configurations). The STiROT is first started at 64 and then switched to 380 MHz if required in the HDPL1 OB keys configuration information. The TOE is certified in the three configurations.
- iLoader size and position in internal flash memory: iLoader is responsible for transferring the firmware image between the internal RAM and the external flash selected by the integrator. Write protection (WRP) on internal flash memory is automatically set for the iLoader code area based on the iLoader linker file. As described in Section 5.1: Secure installation of "TOE_WITH_STIROT" step by step, the configuration in the HDPL1 OB keys must be consistent with the configuration defined in the STM32H7Sxx option bytes. Any failure (meaning consistency issue between option bytes and HDPL1 OB keys) in this responsibility does not compromise the security of the TOE and is still the certified product, however, it results in a nonfunctional TOE (meaning it is not possible to boot any installed image even if it is a correct image).

To personalize all TOE information provisioned in HDPL1 OB keys, the integrator must generate the STiRoT and the DA configuration binary data files (`.obk`) and program it in the OB keys as defined in Section 3.3: Secure installation.

Personalization in STM32H7Sxx option bytes

- Internal flash memory write protection (WRPs option bytes): WRPs configuration aims to make the iLoader code immutable. The Integrator must follow the procedures described in Section 3.3: Secure installation to configure WRPs based on the iLoader linker file. As described in Section 5.1: Secure installation of "TOE_WITH_STIROT" step by step, the configuration in the HDPL1 OB keys must be consistent with the configuration defined in the STM32H7Sxx option bytes. Any failure (meaning consistency issue between option bytes and HDPL1 OB keys) in this responsibility does not compromise the security of the TOE and is still the certified product, however, it results in a nonfunctional TOE (meaning it is not possible to boot any installed image even if it is a correct image).
- Product state: The TOE is certified in the product state set at least to “Closed”. The Debug Authentication process gives the flexibility to perform a full regression (meaning product state regression after erasing all memories) or gives the flexibility to reopen the debug interface in certain domains, without erasing the associated memories, according to the Debug Authentication configuration provisioned in the TOE (as described in the TOE configuration chapter above). In case the product state goes back to “Open”, a full regression (meaning STM32H7Sxx MCU goes back to the virgin state) of the product is performed, and the secure installation must be executed again. The Integrator has the privilege and responsibility to set a correct product state of the TOE at the end of the TOE preparation procedure as described in Section 3.3: Secure installation. Any failure in setting a correct product state compromises the security of the TOE and is not the certified configuration.
- Memory boot configuration (IROT_SELECT option byte): As described in Section 3.3: Secure installation, the TOE is certified in two configurations (TOE_WITH_STIROT and TOE_WITHOUT_STIROT). The Integrator has the privilege and the responsibility to select the TOE configuration compatible with its product security architecture and compatible with its product security requirements. To be in the configuration TOE_WITH_STIROT, the Integrator must follow the procedures described in Section 3.3: Secure installation. Any failure in this responsibility impacts the SFRs (refer to TN1504 to get details about the nonsupported SFRs when not using the TOE_WITH_STIROT configuration) supported in the certified TOE and impacts the security of the product using the TOE as the application of the Integrator is started without TOE checking it.
- ECC_ON_SRAM configuration: The Integrator must follow the procedures described in Section 3.3: Secure installation to configure the ECC_ON_SRAM option byte with the same value as the one defined in “Personalization in HDPL1 OB keys”. Any failure in this responsibility does not compromise the security of the TOE and is still the certified product, however, it can result in a nonfunctional TOE (meaning it is not possible to boot an installed image or to install/update any new firmware candidate).

The authenticity and integrity of all TOE personalization information and confidentiality of the Debug Authentication information and the cryptographic keys must be ensured by the Integrator until it is provisioned inside the TOE and until the TOE security is correctly activated. Once the TOE security is fully activated, the authenticity, integrity, and confidentiality of TOE assets are ensured by STM32H7Sxx IC security protections. However, if the customer cannot rely on a trusted environment (such as trusted manufacturing) to provision the data and to activate the STM32H7Sxx IC security protection, then the Secure Firmware Installation (SFI) solution (refer to AN4992) embedded inside STM32H7Sxx might be used. Any failure in this responsibility can result in the creation of malicious firmware. The Integrator must therefore implement appropriate security measures for the environment to protect the keys involved in the signature of the application binary and the keys or the password involved in the Debug Authentication certificate.

TOE personalization when using TOE configuration “TOE_WITHOUT_STIROT”

The integrator has the privilege and responsibility of personalizing the TOE to make it functional. As described in Section 5.2: Secure installation of “TOE_WITHOUT_STIROT“ step by step, the TOE personalization is done in different parts:

- In HDPL1 OB keys
- In STM32H7Sxx option bytes

Personalization in HDPL1 OB keys

- Debug Authentication configuration: The Integrator has the flexibility to activate the Debug Authentication features of the TOE with a certificate or password method. To do so, the Integrator must provision Debug Authentication root parameters (password or Debug Authentication key and permissions capabilities) inside the TOE (refer to Section 5.2: Secure installation of “TOE_WITHOUT_STIROT“ step by step to get details on permissions capabilities, get details on debug-reopening, regression capabilities). The Integrator also has the flexibility to not activate the Debug Authentication capabilities (meaning no Debug Authentication configuration provisioned inside the TOE). In the certified configuration, the Debug Authentication features of the TOE must be configured with the certificate method.

To personalize all TOE information provisioned in HDPL1 OB keys, the integrator must generate the DA configuration binary data files (`.obk`) and program it in the OB keys as defined in Section 3.3: Secure installation. Depending on the user application, additional configuration files can be generated and programmed in OB keys.

Personalization in STM32H7Sxx option bytes

- Product state: The TOE is certified in the product state set at least to “Closed”. The Debug Authentication process gives the flexibility to perform a full regression (meaning product state regression after erasing all memories) or gives the flexibility to reopen the debug interface in certain domains, without erasing the associated memories, according to the Debug Authentication configuration provisioned in the TOE (as described in the TOE configuration chapter above). In case the product state goes back to “Open”, a full regression (meaning STM32H7Sxx MCU goes back to the virgin state) of the product is performed, and the secure installation must be executed again. The Integrator has the privilege and responsibility to set a correct product state of the TOE at the end of the TOE preparation procedure as described in Section 3.3: Secure installation. Any failure in setting a correct product state compromises the security of the TOE and is not the certified configuration.
- Memory boot configuration (IROT_SELECT option byte): As described in Section 3.3: Secure installation, the TOE is certified in two configurations (TOE_WITH_STIROT and TOE_WITHOUT_STIROT). The Integrator has the privilege and the responsibility to select the TOE configuration compatible with its product security architecture and compatible with its product security requirements. To be in the configuration TOE_WITHOUT_STIROT, the Integrator must follow the procedures described in Section 3.3: Secure installation. Any failure in this responsibility impacts the SFRs (refer to TN1504 to get details about the nonsupported SFRs when not using the TOE_WITH_STIROT configuration) supported in the certified TOE. As the Integrator selects the TOE configuration “TOE_WITHOUT_STIROT”, he takes the responsibility to implement his immutable RoT application in the user flash if his product security requires such a type of security feature. The certified TOE only ensures the “boot” (after reset) of the Integrator's code located in the user flash without verifying it before executing it. However, the TOE still manages securely the Debug Authentication features.

The authenticity and integrity of all TOE personalization information and confidentiality of the Debug Authentication must be ensured by the Integrator until it is provisioned inside the TOE and until the TOE security is correctly activated. Once the TOE security is fully activated, the authenticity, integrity, and confidentiality of TOE assets are ensured by STM32H7Sxx IC security protections. However, if the customer cannot rely on a trusted environment (such as trusted manufacturing) to provision the data and to activate the STM32H7Sxx IC security protection, then the Secure Firmware Installation (SFI) solution (refer to AN4992) embedded inside STM32H7Sxx might be used. Any failure in this responsibility can result in the creation of malicious firmware. The Integrator must therefore implement appropriate security measures for the environment to protect the keys involved in the signature of the application binary and the keys involved in the Debug Authentication certificate.

User application development

In the TOE_WITH_STIROT configuration, the Integrator has the privilege and the responsibility to develop the user application. At each boot, the integrity and the authenticity of the user application are verified before booting the user application.

In both TOE_WITH_STIROT and TOE_WITHOUT_STIROT configurations, it is the Integrator's responsibility to activate all the required security protections during the user application execution.

iLoader development

In the TOE_WITH_STIROT configuration, the Integrator has the privilege and the responsibility to develop the iLoader. iLoader is responsible for transferring the firmware image between the internal RAM and the external flash selected by the integrator. On STiROT request, iLoader:

- Copies the “secondary” slot from external memory to RAM if an image is detected at the beginning of the installation process.
- Copies the firmware image from RAM to the “primary slot” in external memory at the end of the installation process (after decryption and authenticity and integrity controls).
- Copies the “primary” slot from external memory to RAM at each boot to verify the authenticity and integrity before its execution.

Security protections such as MPU protection to allow execution only in the iLoader code area, write protection (WRP) to avoid any modification of the iLoader code area and MCE configuration to encode the *firmware installation slot* in an external flash with specific keys are under the STiROT responsibility. This encryption key stored in the HDPL1 OB keys is specific to each product. It cannot be accessed from iLoader or a firmware application.

It is the integrator’s responsibility to adapt this code to the selected integrator’s external flash and to keep all required protections activated.

TOE hardware cryptographic accelerators

The TOE is certified with hardware-accelerated cryptography that is protected against side-channel, fault injection, and timing analysis attacks.

To achieve the required resistance against hardware attacks, the Integrator must use the following hardware-accelerated cryptography function, detailed in the corresponding sections of the RM0477 reference manual.

With the TOE, the Integrator can do cryptographic key generation for ECDH, ECIES, and ECDSA algorithms. In FIPS PUB 186-4 specification section B.4 NIST proposes two methods for the generation of the ECC private key ("extra random bits" or "testing candidates"). The Integrator must select one of those two methods when using the random number generation of the TOE to compute ECC private keys.

The platform also provides cryptographic hash operations without special side-channel attack resistance. It is recommended not to use these algorithms directly from the hardware peripheral when manipulating sensitive information.

TOE random number generation

For the device to generate random numbers as specified in NIST SP800-90B, the integrator must use the TRNG peripheral with the configuration A. Refer to the “True random number generator (RNG)” section, “validation conditions” subsection of the RM0477 reference manual for details.

4.2.2 Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C)

The integrator can access different interfaces to develop its product:

- Physical chip interface
- Firmware download slot interface
- Security library interface
- JTAG interface
- Cryptographic functions interface

Physical chip interface

Case 1 (TOE_WITH_STIROT configuration): The hardware life cycle state is set to at least “Closed” and the IROT_SELECT option byte is enabled to boot on STiROT in the system flash memory

After each product power-on or reset, the TOE boots on the STiRoT component located inside the immutable system flash memory of the STM32H7Sxx that manages the secure initialization of the user application in the user flash memory.

Method of use:

- Power on the system.
- Reset the STM32H7Sxx.
- “Running” the user application generates a reset (ArmV7M reset instruction or operation).

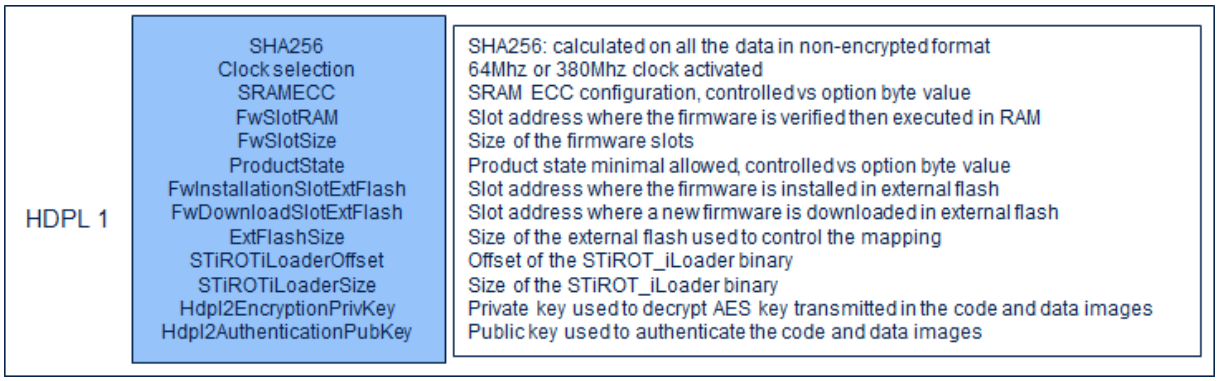
Parameters:

- Not applicable

Actions:

- Execute in HDPL1 mode the STiROT component located at a fixed address in the immutable system flash memory using STiROT provisioning data located in the HDPL1 OB keys area (overview given in Figure 3). After decryption with AHK, the integrity of STiROT provisioning data is controlled, then the value of the parameters is used to manage the secure initialization of the platform and to locate the image slots in the memory. The STiROT application executes first the Secure Boot function and then the Secure Firmware Update functions. The Secure Boot function manages the secure initialization of the platform. The Secure Firmware Update checks in the *Firmware download slot* if there are any candidate firmware images to be analyzed.

Figure 3. STiRoT provisioning data



- Errors:
  - STM32H7Sxx option byte value violations or STiRoT provisioning data error: In case the STM32H7Sxx option byte values are not correctly configured to ensure TOE security or in case the STiRoT provisioning data are not consistent, the TOE secure boot procedure detects the problem and blocks the STiRoT secure boot procedure execution (reset is generated).

Case 2 (TOE_WITHOUT_STIROT configuration): The hardware life cycle state is set to at least “Closed” and the IROT_SELECT option byte is disabled to boot in the user flash memory.

After each product power-on or reset, the TOE boots on the user application located in the user flash memory of the STM32H7Sxx.

Method of use:

- Power on the system.
- Reset the STM32H7Sxx.
- “Running” the user application generates a reset (ArmV7M reset instruction or operation).

Parameters:

- Not applicable

Actions:

- Execute in HDPL1 mode the code located at the boot address configured in the option bytes.

Errors:

- The user application is not started if its vector table is not located at `0x8000000`.

Firmware download slot interface (TOE_WITH_STIROT configuration)

The *Firmware download slot* is used to implement the remote firmware update functionality of the firmware image by triggering the Secure Firmware image upgrade process implemented inside the STiROT component. It is a memory area in the external flash where a new candidate of the firmware image is placed by writing into it, using the nonsecure bootloader application located inside the system flash memory via one of the supported physical interfaces or using another external loader application implemented in the user flash. After any product reset, on STiROT request, iLoader transfers the image in RAM if the magic 16 bytes are present at the slot area end location. The TOE attempts to interpret the data as a candidate firmware image and requests iLoader to apply it to the *Firmware installation* slot in case it is correctly verified. If a candidate firmware image is analyzed as not valid (authenticity and integrity), then the image is deleted from the *Firmware download slot*.

Method of use:

- The *firmware download slot* region is located at the *FwDownloadAreaExtFlash* address (defined in ST-iRoT provisioning data), as described in Figure 4. To use the *firmware download slot*, data must be written in the correct image format in the *firmware download slot* area and the magic 16 bytes must be written in the slot area end location as described in Figure 5.

Figure 4. Memory layout

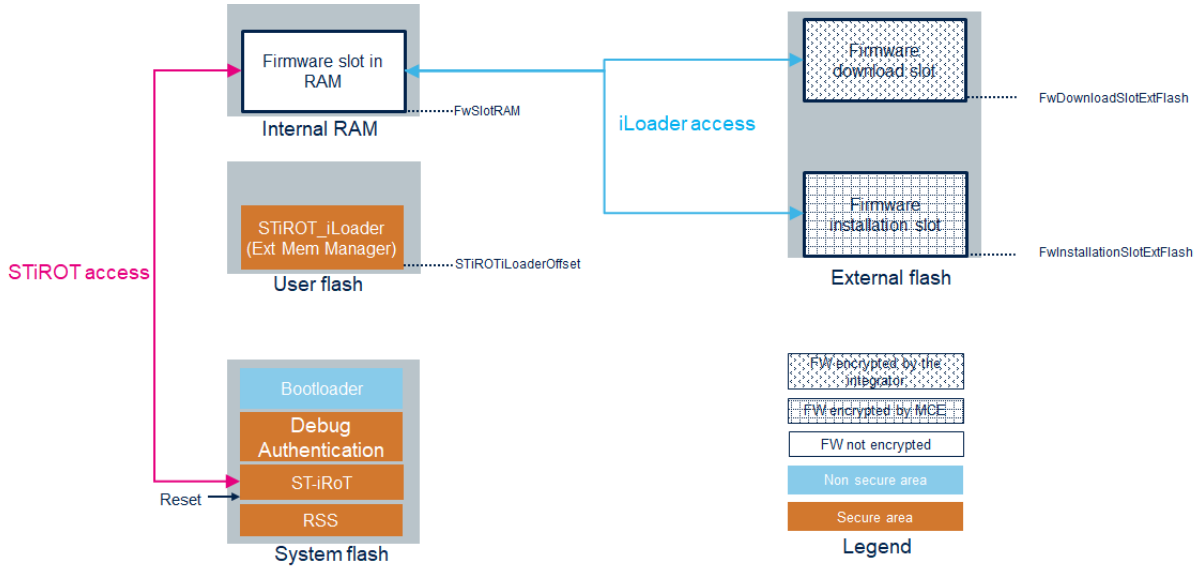
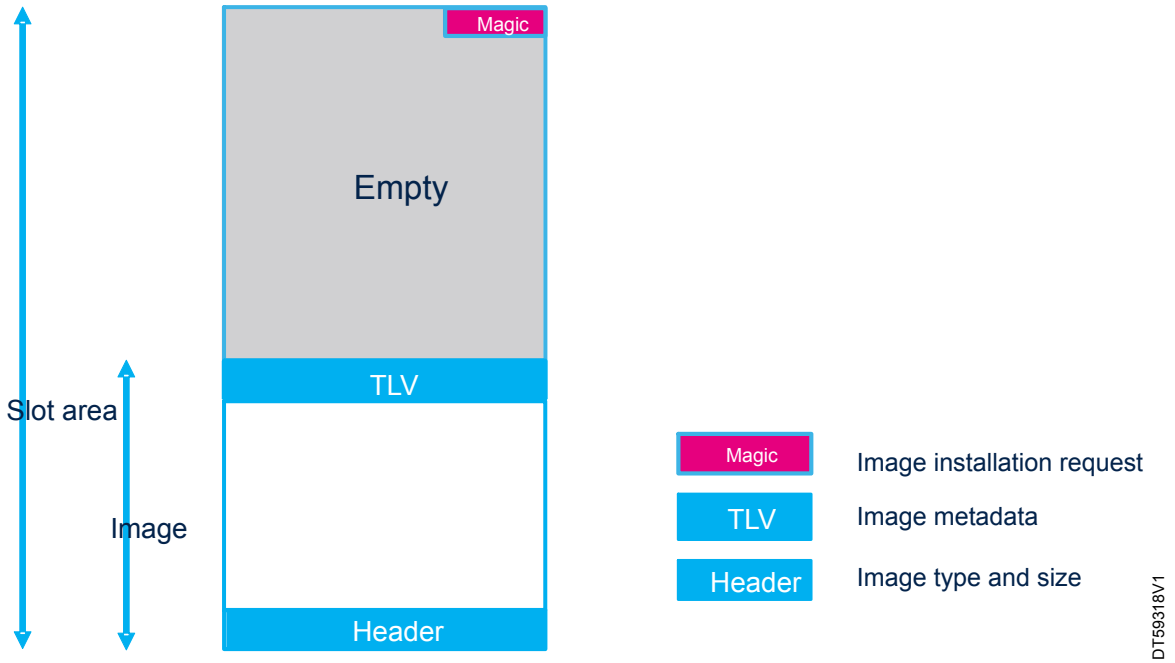


Figure 5. **Image format**



Parameters:

- The candidate firmware image is written in the firmware download slot.

Actions:

- At each product reset, the TOE (STiROT component) requests iLoader to check if a new firmware image is loaded by the nonsecure system bootloader or by a standalone external loader application in the *firmware download slot*. The new firmware image must be programmed at the beginning of the *firmware download slot* and must comply with the image format (image header, image payload, and image TLV) as defined by the `STiROT` application. A PC tool STM32TrustedPackageCreator is delivered to generate data image binary with the right format. When a new firmware image is detected, the *STiROT component* launches the update procedure of the firmware image in the *firmware installation slot* (that verifies the data before updating the firmware).

Errors:

- The candidate firmware image is not installed in the *firmware installation slot* and is erased from the *firmware download slot* in the case of the following errors:
  - Firmware image size is not consistent
  - Flash reading errors (double ECC errors)
  - Version check failure: The firmware image version is lower than the previous valid image installed.
  - Firmware image integrity failure.
  - Firmware image signature failure: firmware image not authentic
- The candidate firmware image is not installed in the *firmware installation slot* and TOE is reset:
  - Flash writing or erasing errors might be reported by the flash driver used to write data in the firmware installation slot.

Security library interface

The security library provides runtime services to STiROT, Debug Authentication, or the integrator's Root of Trust firmware to jump to the user application, iLoader, or BL (nonsecure system bootloader).

Jump to application

Function name: RSSLIB_JumpHDPLvl2

Function prototype: uint32_t RSSLIB_JumpHDPLvl2 (uint32_t VectorTableAddr, uint32_t MPUIndex)

Method of use:

- In the TOE_WITH_STIROT configuration and at each reset, STiROT controls the integrity and the authenticity of the user application and its associated secret. In case of success, STiROT jumps to the user application by calling the function pointer through the RSSLIB_pFunc_TypeDef structure located at the address `0x1FF1FD4C`.
- In the TOE_WITHOUT_STIROT configuration and at each reset, it is the Integrator's responsibility to implement its own Root of Trust firmware, which can jump to the user application by calling the function pointer through the RSSLIB_pFunc_TypeDef structure located at the address `0x1FF1FD4C`.

Parameters:

- VectorTableAddr: Input parameter, address of the next vector table to apply. The vector table format is the one used by the Cortex®-M7 core.
- MPUIndex: Input parameter, MPU region index. The Caller function must define but keep disabled the corresponding MPU region before calling RSSLIB_JumpHDPLvl2. The function enables the MPU region before jumping to the Reset Handler of the vector table. The vector table Reset Handler function must belong to the MPU region.

Actions:

- Close the flash memory HDPL1 and OB keys L1 areas by incrementing HDPL to 2.
- Then jump to the Reset Handler embedded within the vector table which address is passed as an input parameter

After closing HDPL1, RSSLIB_Sec_JumpHDPLvl2 enables the MPU region provided as an input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table and jumps to the Reset Handler function supported by the vector table too.

On successful execution, the function does not return and does not push LR onto the stack.

Errors:

In case of failure (bad input parameter value), RSSLIB_JumpHDPLvl2 returns `0xF5F5F5F5`.

Jump to BL or iLoader

Function name: RSSLIB_JumpHDPlvl3

Function prototype: uint32_t RSSLIB_JumpHDPlvl3 (uint32_t VectorTableAddr, uint32_t MPUIndex)

Method of use:

- In the TOE_WITH_STIROT configuration: At reset, STiRoT jumps to iLoader by calling the function pointer through the RSSLIB_pFunc_TypeDef structure located at the address `0x1FF1FD4C`.
- In the TOE_WITH_STIROT and TOE_WITHOUT_STIROT configurations: On request and after authentication process execution, Debug Authentication jumps to the nonsecure system bootloader located in the system flash memory by calling the function pointer through the RSSLIB_pFunc_TypeDef structure located at the address `0x1FF1FD4C`.

Parameters:

- VectorTableAddr: Input parameter, address of the next vector table to apply. The vector table format is the one used by the Cortex®-M7 core. The vector table format is the one used by the Cortex®®-M7 core.
- MPUIndex: Input parameter, MPU region index.

The Caller function must define but keep disable the corresponding MPU region before calling RSSLIB_JumpHDPLvl3. The function enables the MPU region before jumping to the Reset Handler of the vector table. The vector table Reset Handler function must belong to the MPU region.

Actions:

- Close the flash memory HDPL1 area by incrementing HDPL to 3.
- Then jump to the Reset Handler embedded within the vector table which address is passed as an input parameter.

After closing HDPL1, RSSLIB_JumpHDPlvl3 enables the MPU region provided as an input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table and jumps to the Reset Handler function supported by the vector table too.

On successful execution, the function does not return and does not push LR onto the stack.

Errors:

In case of failure (bad input parameter value), RSSLIB_JumpHDPlvl3 returns `0xF5F5F5F5`.

JTAG interface

Standard JTAG with SWD interface allows programming data inside the TOE and allows debugging of the TOE and integrator application. It is used according to [IEE1149] and [ADI5].

Debug Authentication is accessed through RSS at boot when writing “STDA” under reset in a dedicated 32-bit register (DBGMCU) via JTAG or SWD. When the product state is CLOSED, access to Cortex®®-M7 is closed and only this DBGMCU register can be accessed.

According to the product state, the Debug Authentication through JTAG can perform several actions:

- Open debug (for a different HDPL) when the product state is CLOSED
- Perform a full regression (erasing all the flash memory and OB keys)
- In addition, a platform power off and battery removal erase the nonvolatile memories
- Jump into a nonsecure system bootloader

The Debug Authentication protocol is based on Arm® PSA ADAC, which uses a certificate chain and a challenge-response principle to trigger the requested action. Refer to [Debug Authentication] to get more details on the Debug Authentication protocol.

Method of use:

- Write “STDA” in the DBGMCU register under reset then inject the certificate chain and token through DBGMCU to trigger an action.

Parameters:

- Commands can be sent to Debug Authentication to trigger actions through DBGMCU and JTAG.
- Root and leaf certificates (containing the key and permissions) and tokens (containing the signed challenge and the requested action). Refer to the Arm® document [Debug Authentication].

Actions:

- Discovery command: Allows knowing the capabilities of the Debug Authentication
- Authentication start: Triggers the beginning of the authentication procedure. The first step is the generation of a random challenge by the DEVICE, which is sent to the HOST. Then the HOST sends certificates (root and leaf) which are verified (the root certificate is linked to the hash of the public key stored in a 256-bit field in HDPL1 OB keys) and sends the token. Finally, according to the permissions defined in the DEVICE (the maximum capabilities of the Debug Authentication are defined in a 128-bit field in HDPL1 OB keys) and to the permissions defined in the certificate chain, the requested action is triggered or not. The concerned actions are:
  - Open debug (for the different HDPLs)
  - Perform a full regression (erasing all the flash memory and OB keys)
  - Jump into a nonsecure system bootloader
- Lock debug command: Allows closing debug after being opened by an authentication procedure

Errors:

The action is not executed, and an error message is returned to the HOST in case of the following errors:

- If the requested action does not match the allowed permissions or the authorized product state
- If the root certificate does not match the public key stored in HDPL1 OB keys
- If the signed random challenge is not the right one

Cryptographic functions interface

In each cryptographic peripheral protected against physical attacks, the Integrator can use a set of registers to access cryptographic operations. Register access can be restricted to secure code only. The Integrator must use an SAES peripheral when he intends to perform an AES operation protected against physical attacks. When using the PKA peripheral for both ECC and RSA cryptographic operations, the Integrator automatically leverages protection against physical attacks. Among cryptographic operations are random number generations, as specified in NIST SP800-90B.

The platform also provides cryptographic hash operations without special side-channel attack resistance. It is recommended not to use these algorithms directly from the hardware peripherals when manipulating sensitive information. The Integrator can use a set of registers to access the HASH peripheral. Register access can be restricted to secure code only.

Method of use:

- Reset the STM32H7Sxx.
- In the TOE_WITH_STIROT configuration and at each reset, STiRoT controls the integrity and the authenticity of the user application and its associated secret before booting the user application.
- In the TOE_WITHOUT_STIROT configuration and at each reset, the user application code located at the boot address is automatically executed.
- In both cases, the user application code uses SAES and PKA peripherals freely after the RNG peripheral is properly configured and clocked (in RCC).

Parameters:

- The Integrator can access the registers and memory of cryptographic peripherals protected against physical attacks (SAES, PKA, RNG), as defined in RM0477.
- The Integrator can access the registers and memory of the HASH peripheral as defined in RM0477.

Actions:

- The Integrator's code can freely use SAES, PKA, and RNG registers and memory, as defined in RM0477.
- Except if sensitive information is manipulated, the integrator's code can freely use HASH registers and memory, as defined in RM0477.

Errors:

- Peripherals SAES, PKA, RNG, and HASH have precise error events described in RM0477.
- When SAES, RNG, or PKA detects a fault injection, an input tamper event is raised in the TAMP peripheral. Depending on the Integrator's code, such an event can block the TOE application until the product is reset. In the certified configuration, when the STiROT application detects a hardware fault in RNG, SAES, or PKA the TOE is blocked until it is powered off/on.

4.2.3 Security-relevant events (AGD_OPE.1.4C)

TOE_WITH_STIROT configuration:

Once configured, TOE detects any unauthorized access and any unexpected configuration as described here:

- TOE access violation during TOE execution: any code execution attempt outside the TOE executable region is detected and notified by a MemFault exception on the CPU. This fault triggers a reset of the TOE.
- TOE access violation during application execution: TOE applies temporal isolation (HDPL) and switches from HDPL1 to HDPL3 when jumping into the nonsecure system bootloader, or switches to HDPL2 when jumping into a firmware image. At this point, any access attempt inside HDPL1 is detected and managed following read as 0 write ignore access policy.
- Firmware image authenticity or integrity violation: in case a corrupted image (authenticity or integrity failure) is detected during the TOE secure boot procedure launched after any product reset, the TOE does not start to execute the corrupted images but starts to execute the immutable nonsecure system bootloader. Using the nonsecure system bootloader, a new valid image can be downloaded in the firmware download slot. Once downloaded, this new image is verified and installed. In case the firmware image is corrupted during the application execution, then the problem is detected at the next product reset.
- STM32H7Sxx option bytes values violation: in case the STM32H7Sxx option bytes values are not correctly configured to ensure the TOE security, the TOE secure boot procedure after reset detects the problem and blocks the TOE secure boot procedure execution: A reset is generated. To unlock the product, a full regression must be executed through the Debug Authentication process.
- JTAG access violation: Once TOE security is fully configured, the product cannot be debugged via the JTAG interface anymore. The debug reopening is controlled through the Debug Authentication process. Once authenticated, the permission mask configured inside the STM32H7Sxx MCU and the permission masks set inside the certificate chain allow or not the debug reopening in the selected mode:
  - Debug HDPL1
  - Debug HDPL2
  - Debug HDPL3

An intrusion signal is raised as soon we connect JTAG, blocking access to all protected memories (flash, protected SRAMs, and backup registers).

- Tampering attempt: STM32H7Sxx anti-tamper mechanisms are activated in the TOE for internal tamper events internal security assets and cryptographic IP faults. The product is reset in case of any tampering attempt detected by the TOE.
- Life cycle regression:
  - With the life cycle state set to at least “Closed”, with the Debug Authentication configuration provisioned, and with the required certificate it is possible to do a full regression to the “Open” life cycle state. A flash and OB key mass erasure is performed first.
  - Any wrong certificate injection on the JTAG/SWD interface is detected and triggers a reset of the TOE.

TOE_WITHOUT_STIROT configuration:

- JTAG access violation: Once TOE security is fully configured, the product cannot be debugged via the JTAG interface anymore. The debug reopening is controlled through the Debug Authentication process. Once authenticated, the permission mask configured inside the STM32H7Sxx MCU and the permission masks set inside the certificate chain allow or not the debug reopening in the selected mode:
  - Debug HDPL1
  - Debug HDPL2
  - Debug HDPL3

An intrusion signal is raised as soon as we connect JTAG, blocking access to all protected memories (flash, protected SRAMs, and backup registers).

- Life cycle regression:
  - With the life cycle state set to at least “Closed”, with the Debug Authentication configuration provisioned, and with the required certificate it is possible to do a full regression to the “Open” life cycle state. A flash and OB key mass erasure is performed first
  - Any wrong certificate injection on the JTAG/SWD interface is detected and triggers a reset of the TOE.

4.2.4 Security measures (AGD_OPE.1.6C)

To verify the correct version of all platform components, the following measures must be taken:

- Follow all guidelines described in Section 3.1: Secure acceptance must be followed.

To achieve the correct usage of the TOE, the following measures must be taken:

- Follow all guidelines described and referenced in Section 3.2: Secure installation and secure preparation of the operational environment (AGD_PRE.1.2C).
- Follow all guidelines described in Section 4.2.1: User-accessible functions and privileges (AGD_OPE.1.1C) and Section 4.2.2: Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C) regarding the implementation of the required user drivers.
- Once the integrator finishes its user application development and wants to start to validate the complete product with the security fully activated, he must configure the TOE to use its certified configurations as described in Section 3.3: Secure installation to validate the user application in the final security configuration.
- Once the integrator finishes its user application development and wants to start production, the integrator must securely provision the TOE personalization data as described in Section 3.3: Secure installation and following the rules, as described in Section 4.2: Operational guidance for the integrator role, to ensure their authenticity, integrity, and confidentiality.
- Once the integrator finishes the production of a final user application, he must set the STM32H7Sxx hardware static protections as stated in Section 3.3: Secure installation. To protect the complete product including the user application, the final product state must be “CLOSED” or “LOCKED”.
- The integrator must protect the integrity and confidentiality of the private cryptographic keys used to build new authentic firmware images and part of the Debug Authentication configuration.
- The persons responsible for the application of the procedures described in Section 3: Preparative procedures and the persons involved in the delivery and protection of the product must have the required skills and must be aware of security issues.
- In the case that any part of the preparative procedures of the TOE or any part of the preparative procedures of the integrated IoT solution is executed by a party other than the integrator, the integrator must guarantee that sufficient guidance is provided to this party.
- The integrator must protect the integrity of all the TOE personalization data until they are provisioned and well-protected inside the TOE of each device. Moreover, the integrator must protect the confidentiality of the private cryptographic keys and private Debug Authentication keys that are included in the TOE personalization data.

The configuration of the Debug Authentication feature is part of the steps described in Section 3.3: Secure installation: DA configuration files generation, OB keys provisioning. To achieve the correct configuration of the debug functionality, all the measures described in this section for the correct usage of TOE must be followed.

4.2.5 Modes of operation (AGD_OPE.1.5C)

Configuration “TOE_WITH_STIROT”

The TOE operates after the product reset by executing the TOE immutable Root of Trust, the only interface is the external flash memory slot where a new firmware image can be downloaded. In case a new image to install is available then TOE verifies it and installs it. In case there is no new image to be installed, TOE verifies the installed image. If the firmware-installed image is valid, then the TOE immutable Root of Trust starts the firmware image.

In case the firmware image is not valid, which means no valid firmware image installed and no new image in the *firmware download slot* to be installed, the TOE generates a reset.

In case the TOE option bytes are not correctly configured to ensure the TOE security, the TOE secure boot procedure, after reset, detects the problem, blocks the TOE secure boot procedure execution, and generates a reset. To unlock the product, as the life cycle state is at least set to “Closed”, a full regression must be performed through the Debug Authentication process that fully erases the flash memories. Then the preparation procedure as described in Section 3.3: Secure installation must be followed.

In case the TOE detects any violation, as described in Section 4.2.3: Security-relevant events (AGD_OPE.1.4C), the TOE generates a reset.

Configuration “TOE_WITHOUT_STIROT”

The TOE operates after the product reset by executing the application located at `0x80000000` in the user flash memory.

It is the Integrator's responsibility to implement the user application, to control the correct configuration of the TOE option bytes to ensure TOE security and to define how errors are handled during its execution.

To unlock the product, as the life cycle state is at least set to “Closed”, a full regression must be performed through the Debug Authentication process that fully erases the flash memories. Then the preparation procedure as described in Section 3.3: Secure installation must be followed.

In case the TOE detects any violation, as described in Section 4.2.3: Security-relevant events (AGD_OPE.1.4C), the TOE generates a reset.

5 Annex

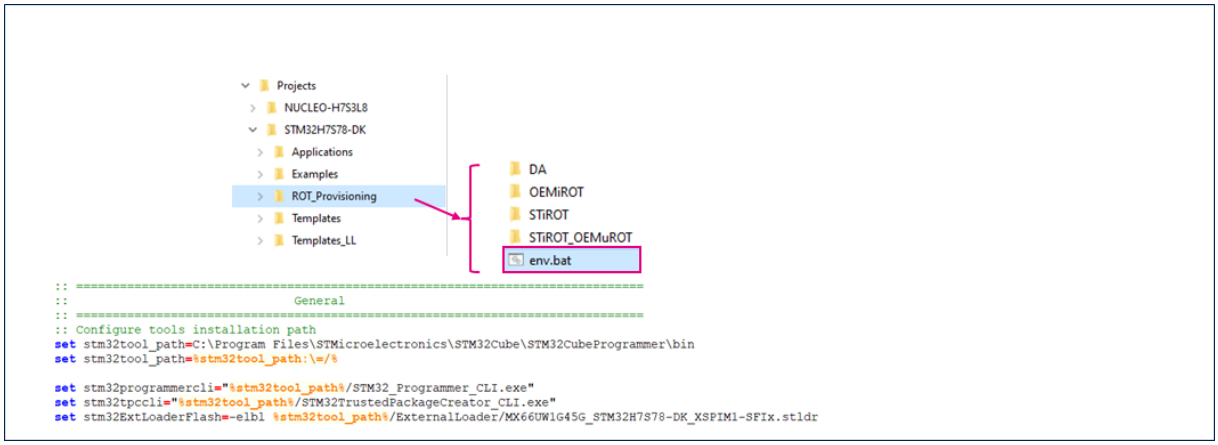
5.1 Secure installation of "TOE_WITH_STIROT" step by step

The STM32H7Sxx product preparation is done in four steps:

- Step 0: Preliminary stage, product state, and method of Debug Authentication selection, iLoader compilation
- Step 1: STiRoT and DA configuration file generation
- Step 2: Code and data image generation
- Step 3: Product provisioning

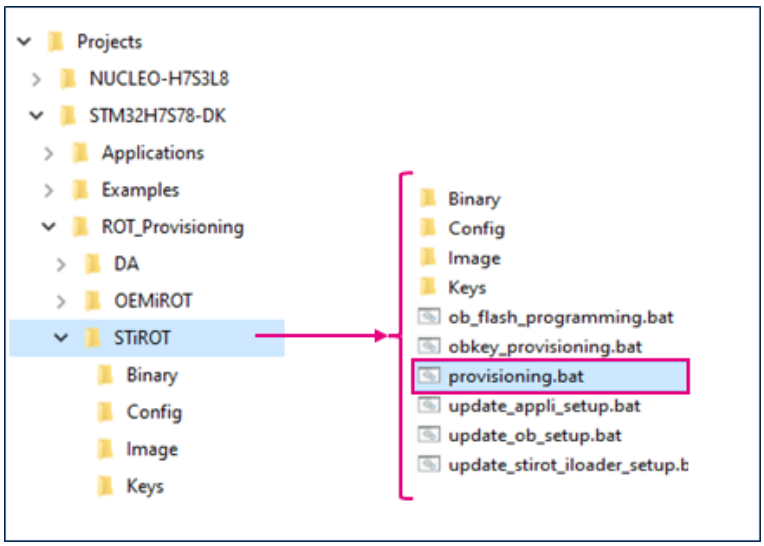
First, the path to access *STM32TrustedPackageCreator* and *STM32CubeProgrammer* on your PC must be updated in `env.bat`.

Figure 6. `env.bat` file



Then, to start the product preparation, launch the `provisioning.bat` script from the STM32CubeH7RS MCU Package.

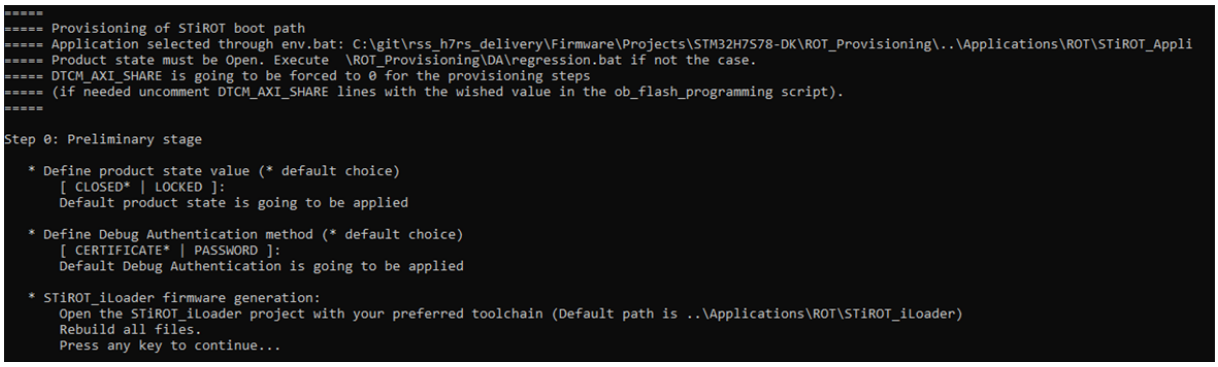
Figure 7. Launch `provisioning.bat`



5.1.1 Step 0: Preliminary stage, product state, and method of Debug Authentication selection

At startup, the product state as well as the Debug Authentication method must be selected. Then iLoader must be compiled:

Figure 8. `provisioning.bat` file execution

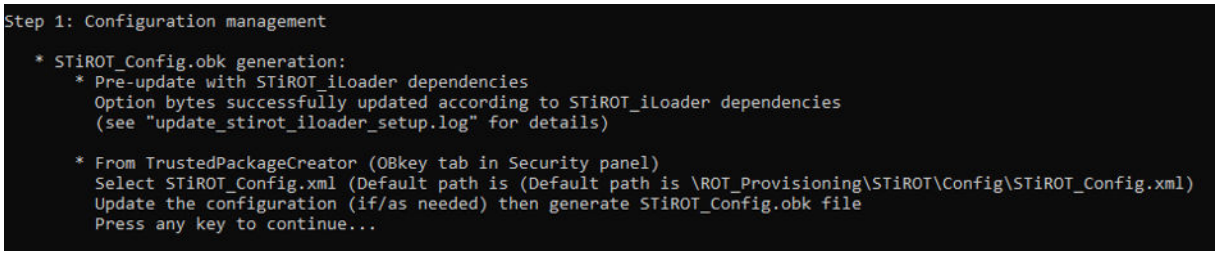


Once the iLoader binary file is generated, press a key to execute the next operation: STiROT and DA configuration.

5.1.2 Step 1: STiRoT and DA configuration files generation

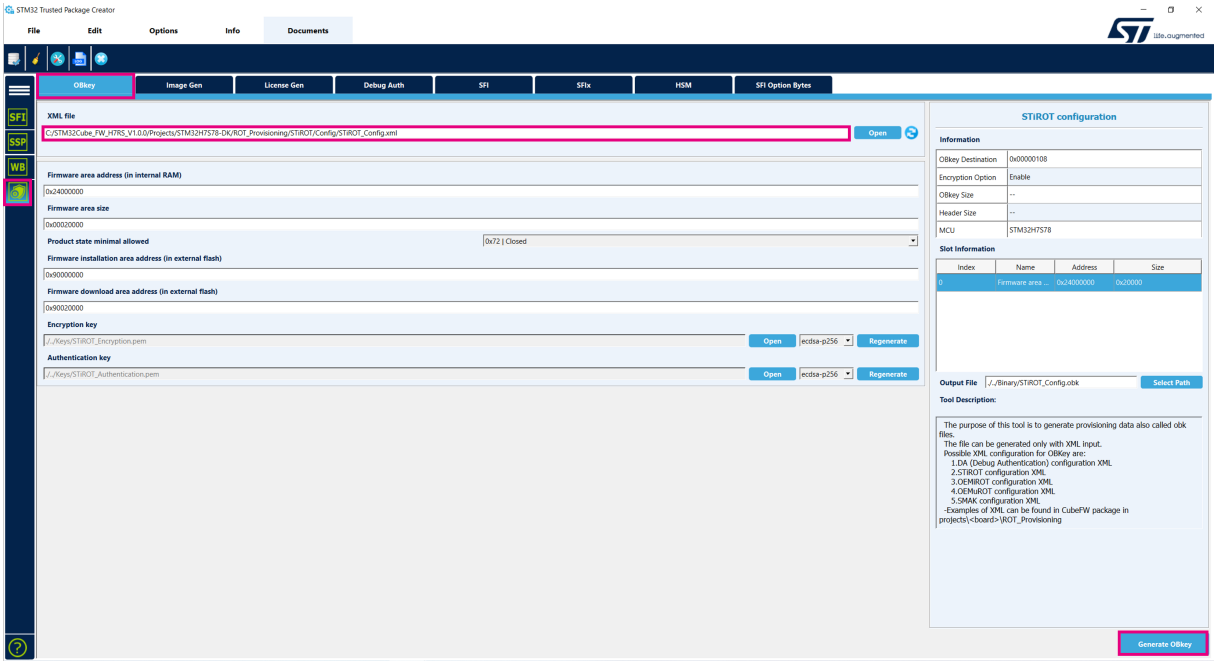
First, the iLoader dependencies are updated inside `STiROT_Config.obk`:

Figure 9. `provisioning.bat` file execution



A default configuration file (`STiROT_Config.obk`) is provided as an example but it is possible to modify this configuration using *STM32TrustedPackageCreator* and `STiROT_Config.xml` as input.

Figure 10. Generation of `STiROT_Config.obk` file using STM32TrustedPackageCreator



The integrator can modify the cryptographic keys used by the TOE to authenticate and decrypt any new firmware, configure the number of images managed as well as the associated areas, and specify whether the application is fully secure or made of a combination of secure and nonsecure applications.

Additional parameters exist but are hidden by default. Modifying the XML file by switching the <Hidden> </Hidden> property of a parameter from 1 to 0 allows the display and the modification of this parameter through *STM32TrustedPackageCreator*. These additional parameters are:

- Clock selection (64 or 380 MHz)
- SRAM ECC management activation

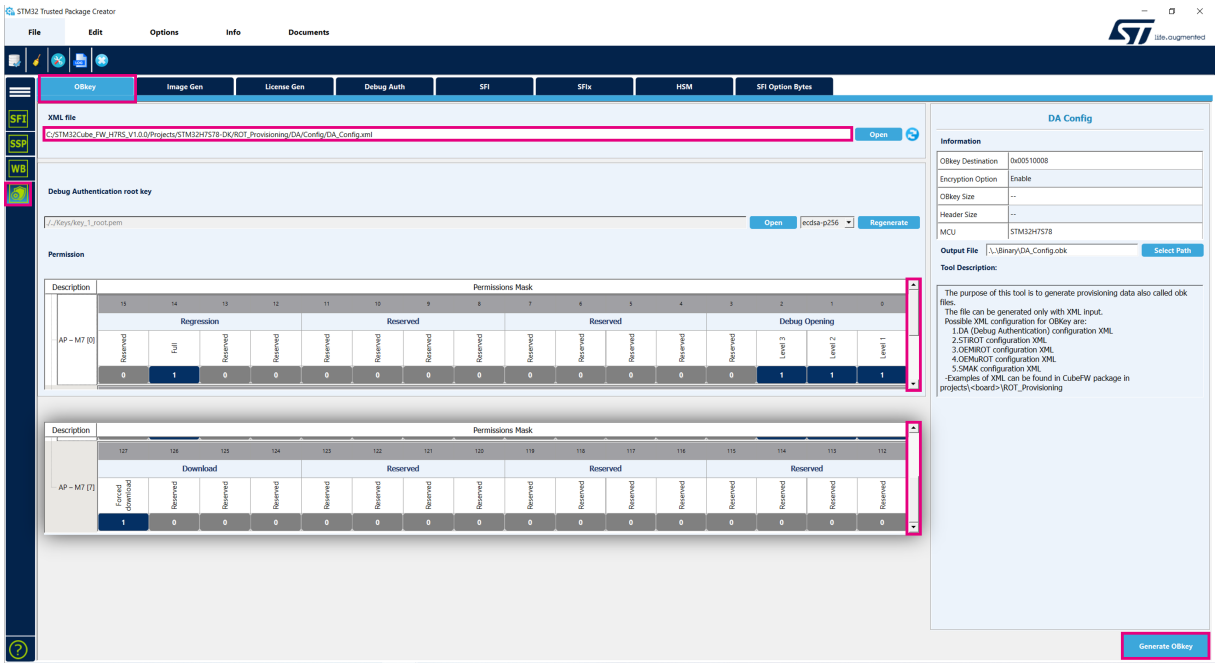
Once `STiROT_Config.obk` is generated or if the default configuration is kept, press a key to execute the next operation, the `DA_Config.obk` generation.

Figure 11. `DA_Config.obk` generation

```
* DA_Config.obk generation:
    From TrustedPackageCreator (OBkey tab in Security panel)
    Select DA_Config.xml(Default path is \ROT_Provisioning\DA\Config\DA_Config.xml)
    Update the configuration (if/as needed) then generate DA_Config.obk file
    Press any key to continue...
```

In the same way, a default configuration file (`DA_Config.obk`) is provided as an example but it is possible to modify this configuration using *STM32TrustedPackageCreator* and `DA_Config.xml` as input.

Figure 12. Generation of `DA_Config.obk` file using STM32TrustedPackageCreator



The integrator can modify the Debug Authentication key and each permission capability:

- Open the debug in HDPL1/2/3.
- Execute a full regression.
- Force the download of a new image through the system bootloader.

Once `DA_Config.obk` is generated or if the default configuration is kept, press a key to execute the next operation, which is the automatic script update.

Figure 13. Automatic script update

```
* ob_flash_programming script update ...
    Option bytes successfully updated according to STiROT_Config.xml
    (see "update_ob_setup.log" for details)

* STiROT_Appli project files (.icf) update ...
    (stm32H7S73xx_flash.icf successfully updated according to STiROT_Config.xml)
    (see "update_appli_setup.log" for details)
```

First, the option bytes provisioning script is automatically updated based on the `STiROT_Config.obk` and STiROT_iLoader icf files values. It concerns:

- WRP option bytes: updated based on information coming from the iLoader icf file (code section definition).
- ECC_ON_SRAM option byte: updated with the same value as the one defined in `STiROT_Config.obk`.
- List of sectors to be erased: updated based on the external flash area defined in `STiROT_Config.obk`.

Finally, the linker file of the user application projects is updated based on the firmware area in the internal RAM information defined in `STiROT_Config.obk`.

5.1.3 Step 2: Firmware image generation

Figure 14. Image generation

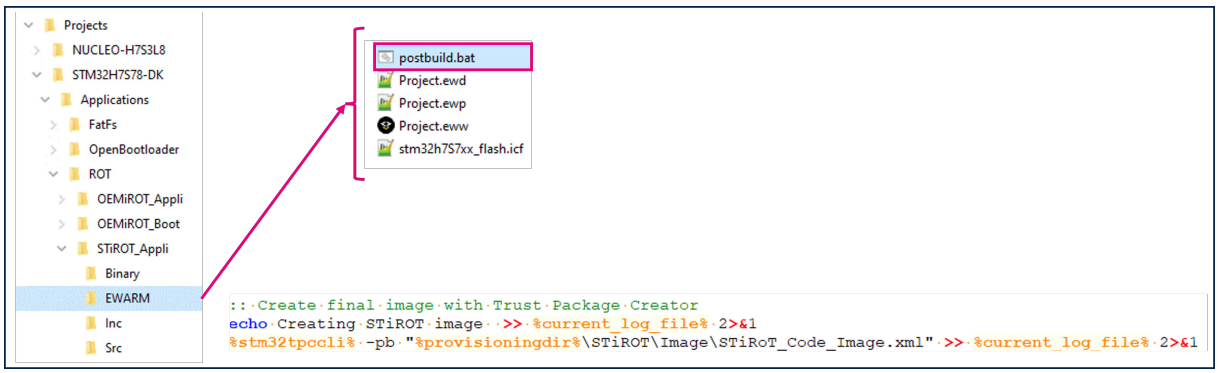
```
Step 2: Image generation

  * Code firmware image generation:
    Open the STiROT_Appli project with your preferred toolchain
    Rebuild all files. The appli_enc_sign.hex file is generated with the postbuild command
    Press any key to continue...
```

A user application example is provided in `C:\STM32Cube_FW_H7RS_V1.0.0\Projects\STM32H7S78-DK\Applications\ROT\STiROT_Appli`.

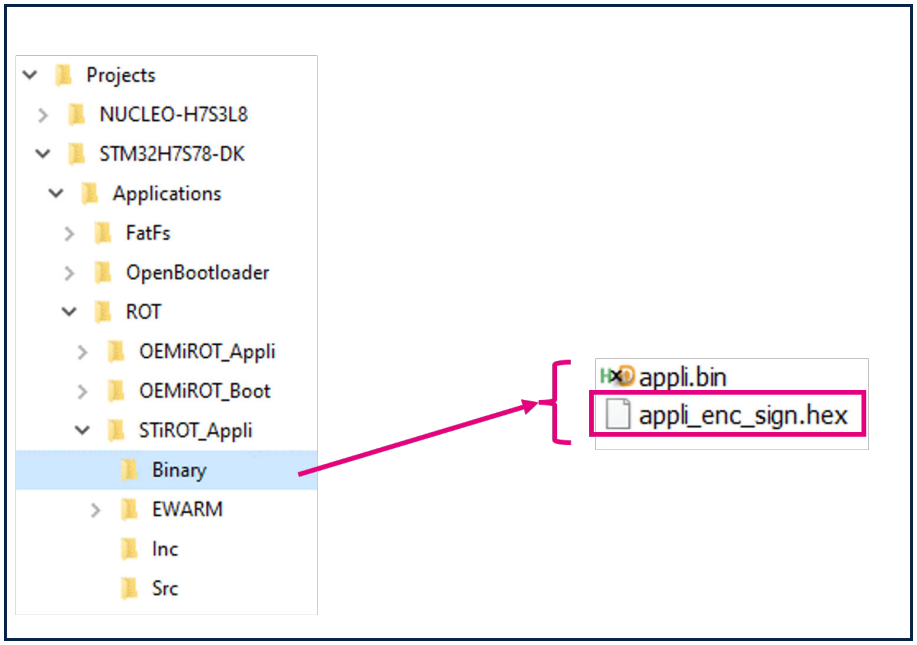
As some project files are updated to take account of changes in STiROT configuration, the project must be fully recompiled. At the end of the compilation, the postbuild command generates the user application image (`appli_enc_sign.hex`) based on the project binary, which is encrypted and signed, with the addition of a header and a TLV part as decrypted in Figure 15.

Figure 15. `postbuild.bat` execution



As a result, `appli_enc_sign.hex` is generated.

Figure 16. `appli_enc_sign.hex` generation



Once the firmware image is generated, press a key to execute the next operation, the product provisioning.

5.1.4 Step 3: Product provisioning

Figure 17. `provisioning.bat` launching

```
Step 3: Provisioning

   * Setting the product state PROVISIONING

   * Provisionning the .obk files ...
       Successful obk provisioning
       (see "obkey_provisioning.log" for details)
```

Figure 18. Product provisioned

```
   * Setting the product state PROVISIONING

   * Provisionning the .obk files ...
       Successful obk provisioning
       (see "obkey_provisioning.log" for details)

   * Programming the option bytes and flashing the image ...
       Successful option bytes programming and image flashing
       (see "ob_flash_programming.log" for details)

   * Setting the iROT_SELECT to STiROT

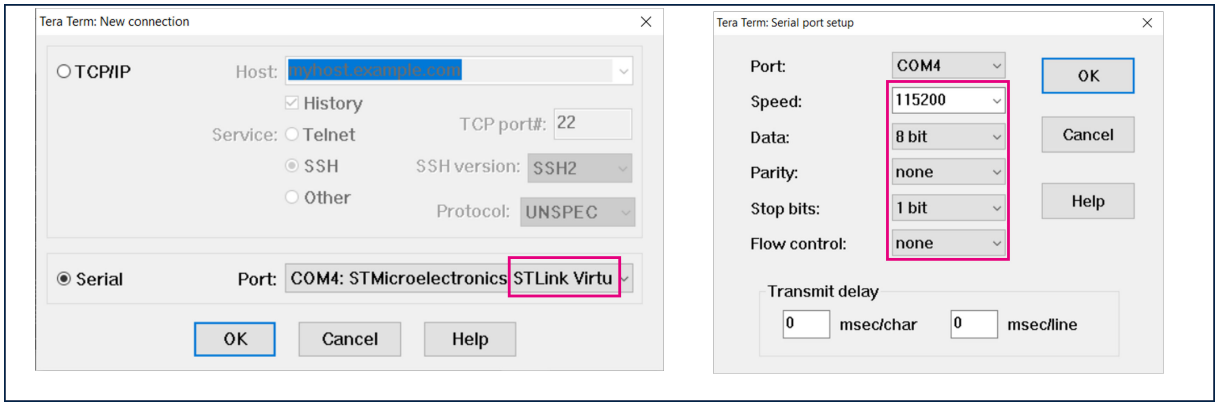
   * Setting the final product state CLOSED


=====
===== The board is correctly configured.
===== Connect UART console (115200 baudrate) to get application menu.
===== Reset the board to start the application.
=====
```

5.1.5 STiROT user application example execution

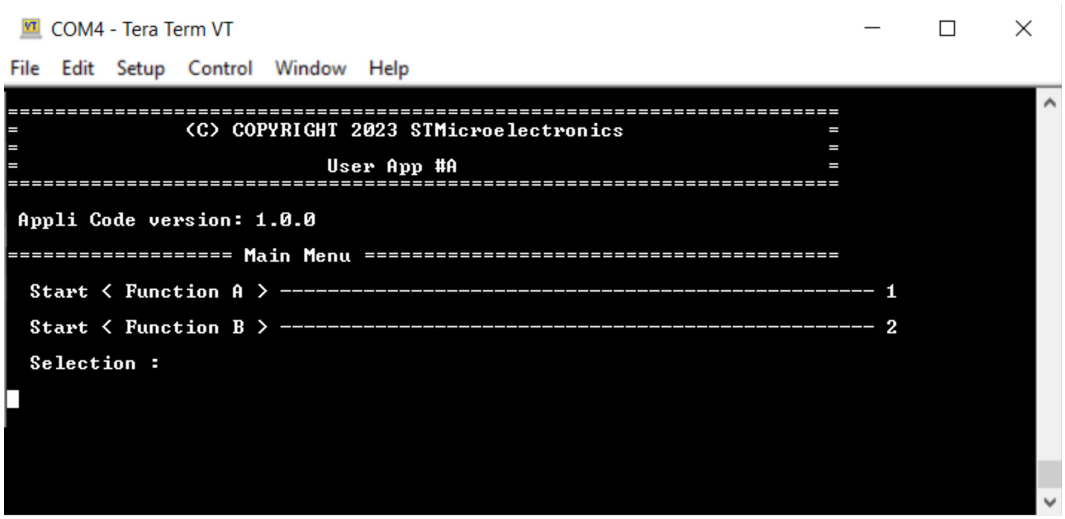
At startup, the STiROT user application example displays a menu through the serial link.
Tera Term, an open-source free software terminal emulator, can be used to display it.
The Virtual COM port used for the connection with the device must be configured as follows:

Figure 19. Virtual COM port configuration



At each reset (the black button on the board), STiROT controls the authenticity and the integrity of the user application before starting its execution. The following menu is displayed.

Figure 20. User application menu



5.1.6 Details on `STiROT_Config.obk`

Table 3 lists all the constraints/controls for each parameter.

Table 3. Parameter description

Parameter	Hidden	Description	Additional controls/constraints
Clock selection	Yes	• 0 (default value): 64 MHz, default configuration • 1: 380 MHz, the best speed supporting the full range of temperature	-
SRAM ECC management activation	Yes	• Yes (default value): a reset is generated in case of ECC detection. ECC is a mechanism to prevent external attacks. • No: ECC is not managed. This is a less secure configuration but it can be selected if a hardware attack is not possible.	This information is duplicated from option bytes to control that the option byte is well configured.
Firmware execution area address in RAM	No	Firmware execution area address in RAM	It must be aligned on 16 Kbytes.
Firmware area size	No	Size of the firmware.	It must be a multiple of 16 Kbytes. Firmware slot size must not exceed the internal RAM capacity.
Product state	No	Minimal allowed product state	This information is duplicated from option bytes to control that the option bytes are well configured.
Firmware installation area address in external flash	No	Firmware installation area address in external flash	It must be aligned on 16 Kbytes. No allowed overlap between execution areas and download areas.
Firmware download area address in external flash	No	Firmware download area address in external flash.	It must be aligned on 16 Kbytes. No allowed overlap between execution and download areas.
External flash size	Yes	External flash size.	Information used to control the mapping of the firmware areas in the device
STiROTLoader offset	Yes	Offset from the beginning of the user flash where is located the STiROT_Loader	WRP protection is automatically set in the STiROT_Loader area.
STiROTLoader size	Yes	Size of the STiROT_Loader	WRP protection is automatically set on the STiROT_Loader area.
Encryption key	No	Key used to encrypt the firmware and data images	When this key is regenerated, the firmware image must be processed with the TPC “Image Gen” tab (STiRoT_Code_Image.xml).
Authentication key	No	Key used to authenticate the firmware and data images	When this key is regenerated, the TPCfirmware image must be processed with the TPC “Image Gen” tab (STiRoT_Code_Image.xml).

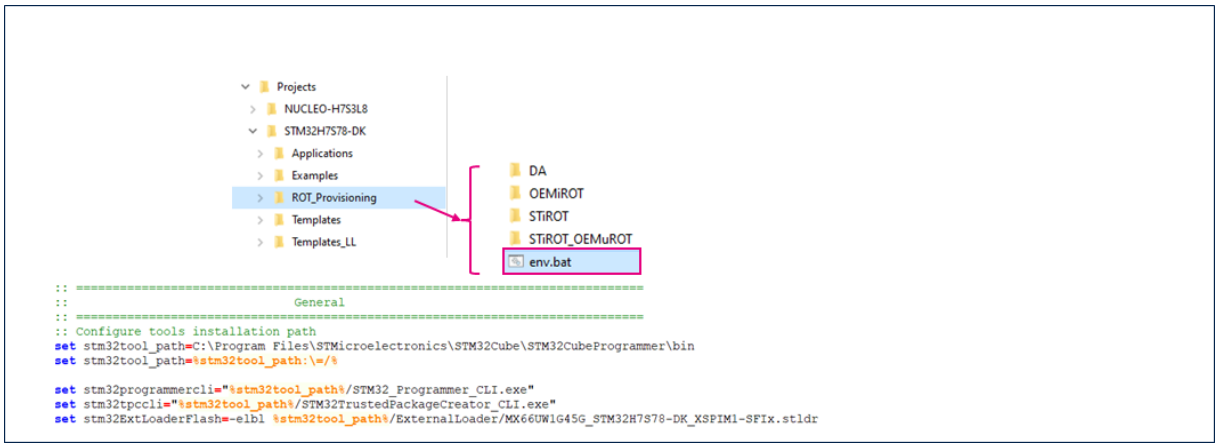
5.2 Secure installation of “TOE_WITHOUT_STIROT“ step by step

The STM32H7Sxx product preparation is done in three steps:

- Step 1: DA configuration file generation
- Step 2: Option bytes programming
- Step 3: Image flashing and OB keys provisioning

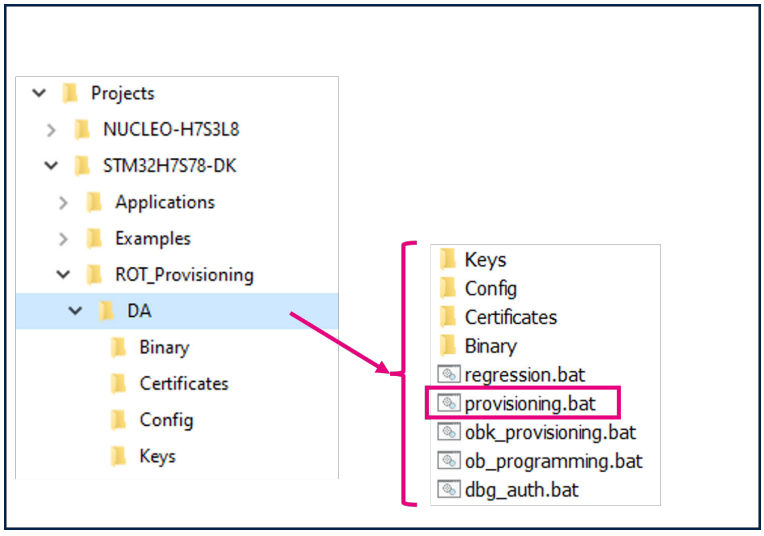
First, the path to access *STM32TrustedPackageCreator* and *STM32CubeProgrammer* on tour PC must be updated in `env.bat`.

Figure 21. `env.bat` file



Then, to start the product preparation, launch the `provisioning.bat` script from the STM32CubeH7RS MCU Package.

Figure 22. Launch `provisioning.bat`



5.2.1 Step 1: DA configuration file generation

At startup, the product state as well as the Debug Authentication method must be selected:

Figure 23. `provisioning.bat` execution

```
=====
===== Provisioning of Legacy DA
===== Product state must be Open. Execute  \ROT_Provisioning\DA\regression.bat if not the case.
===== DTCM_AXI_SHARE is going to be forced to 0 for the provisioning steps
===== (if needed uncomment DTCM_AXI_SHARE lines with the wished value in the ob_programming script).
=====

Step 1 : Configuration management

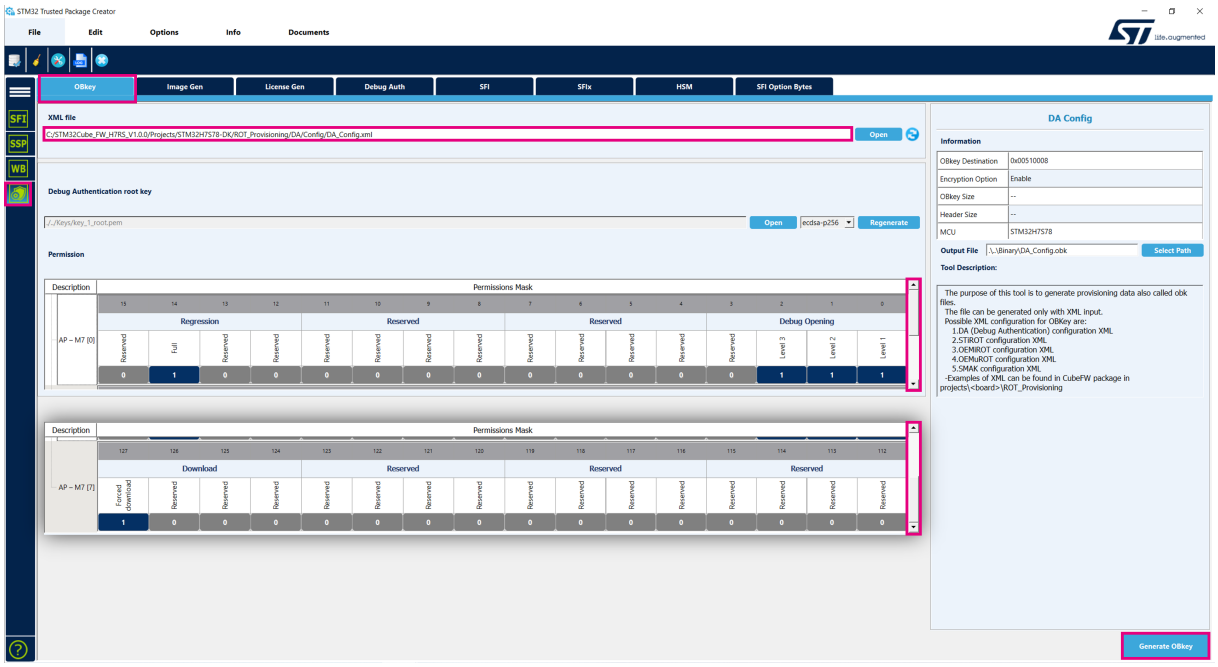
      [ CLOSED | LOCKED ]: CLOSED

   * Define Debug Authentication method
      [ CERTIFICATE | PASSWORD ]: CERTIFICATE

   * DA_Config.obk generation:
      From TrustedPackageCreator (OBkey tab in Security panel).
      Select DA_Config.xml (Default path is \ROT_Provisioning\DA\Config\DA_Config.xml)
      Update the configuration (if/as needed) then generate DA_Config.xml file
      Press any key to continue...
```

A default configuration file (`DA_Config.obk`) is provided as an example but it is possible to modify this configuration using *STM32TrustedPackageCreator* and `DA_Config.xml` as input.

Figure 24. Generation of `DA_Config.obk` file using STM32TrustedPackageCreator



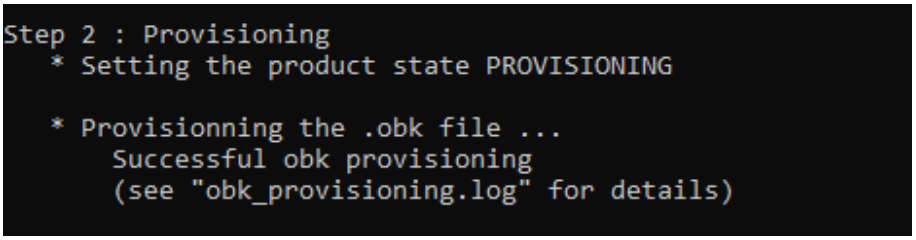
The integrator can modify the Debug Authentication key and each permission capability:

- Open the debug in HDPL1/2/3.
- Execute a full regression.
- Force the download of a new image.

Once `DA_Config.obk` is generated or if the default configuration is kept, press a key to execute the next operation, the option bytes programming.

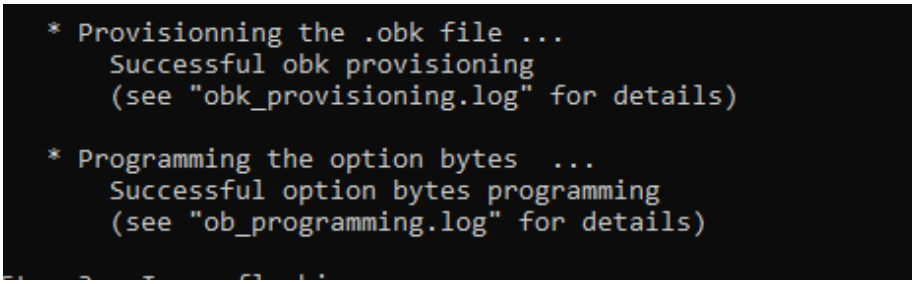
5.2.2 Step 2: Option bytes provisioning and option bytes programming

Figure 25. Option keys provisioning



Once the OB keys are provisioned, it goes to the next operation: OB bytes programming.

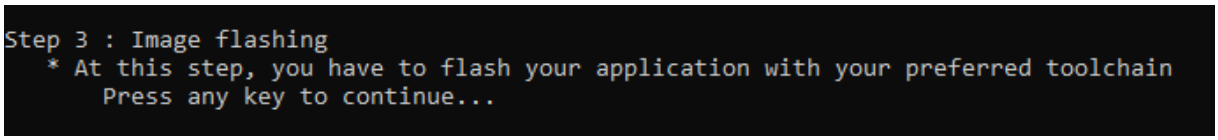
Figure 26. provisioning.bat execution



The WRPSx, HDP_AREA option bytes, are set disabled. Memory boot (IROT_SELECT) option byte is disabled meaning boot in user flash memory.

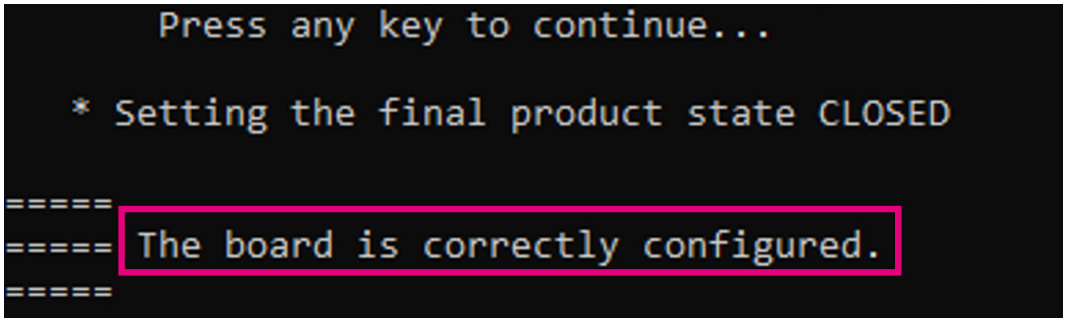
5.2.3 Step 3: Image flashing

Figure 27. `provisioning.bat` launching



Once the user application is programmed in user flash, press a key to execute the final operation: the product state selected for which the TOE is certified is set: "Closed", or "Locked" and the product is provisioned.

Figure 28. Product provisioned



5.3 Debug Authentication process

5.3.1 Certificate generation

Default certificates are provided in `C:\STM32Cube_FW_H7RS_V1.0.0\Projects\STM32H7S78-DK\ROT_Provisioning\DA\Certificates`.

When required, *STM32TrustedPackageCreator* can be used to modify the certificate chain made of:

- A root certificate
- An intermediate certificate
- A leaf certificate

Figure 29. DA certificate chained generation

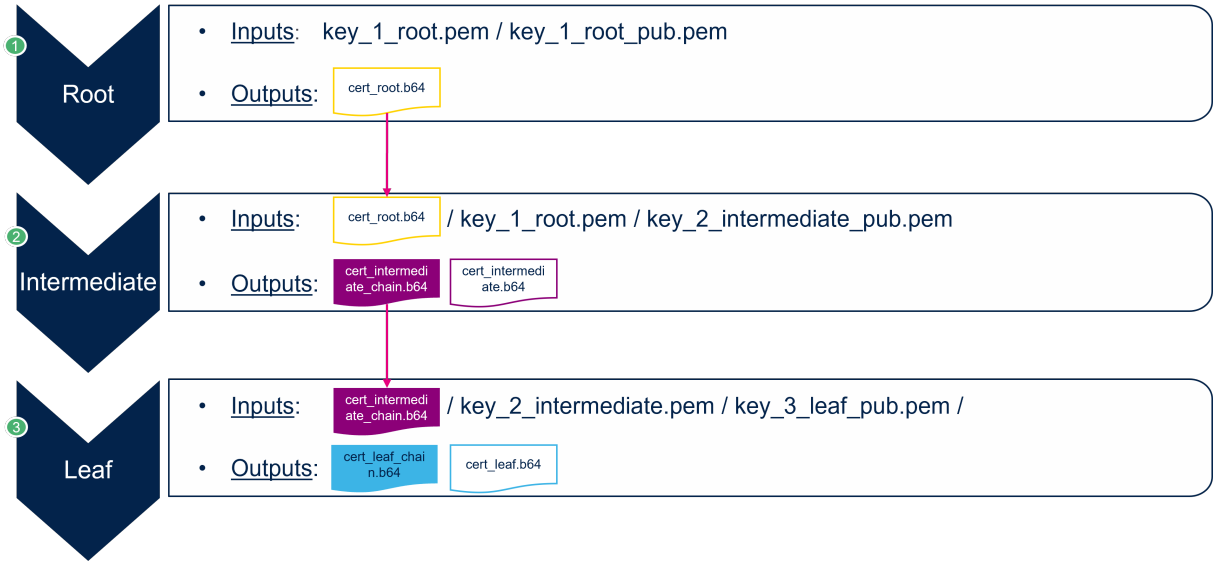


Figure 30. Root certificate

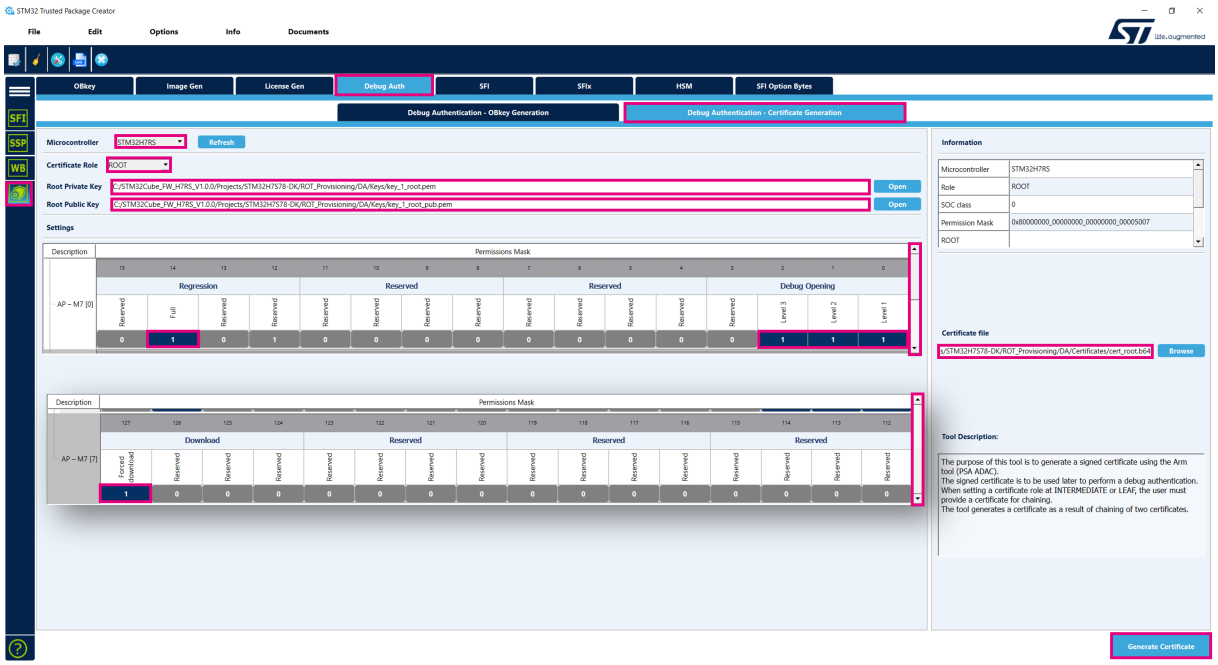


Figure 31. **Intermediate certificate**

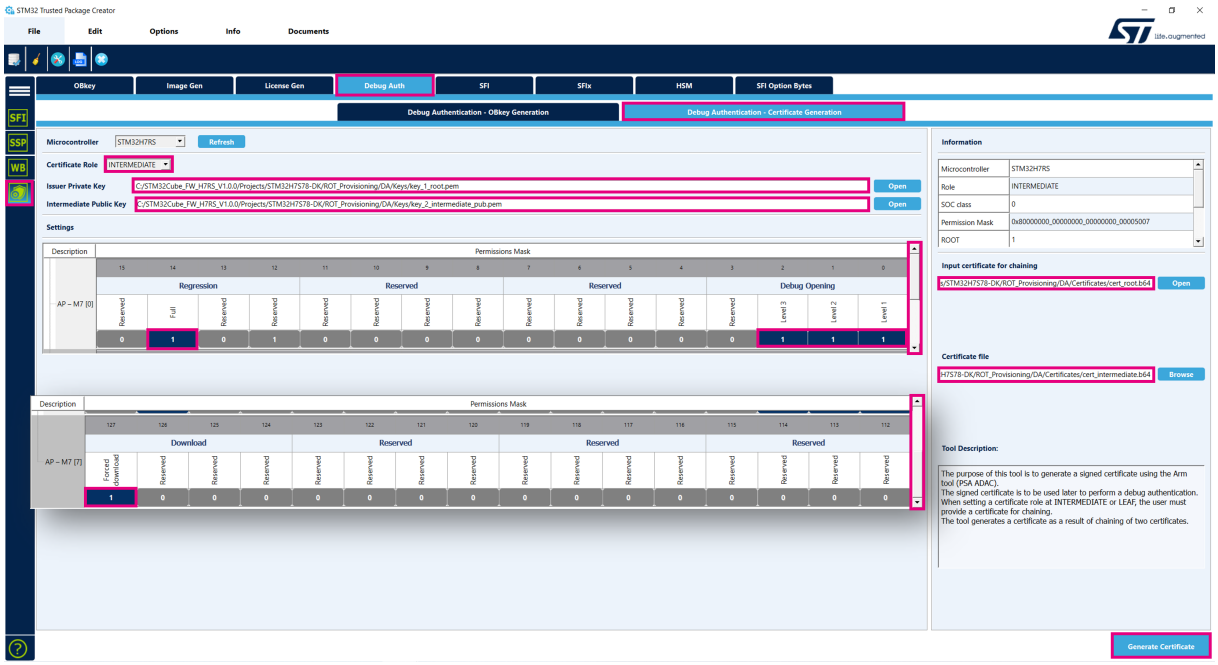
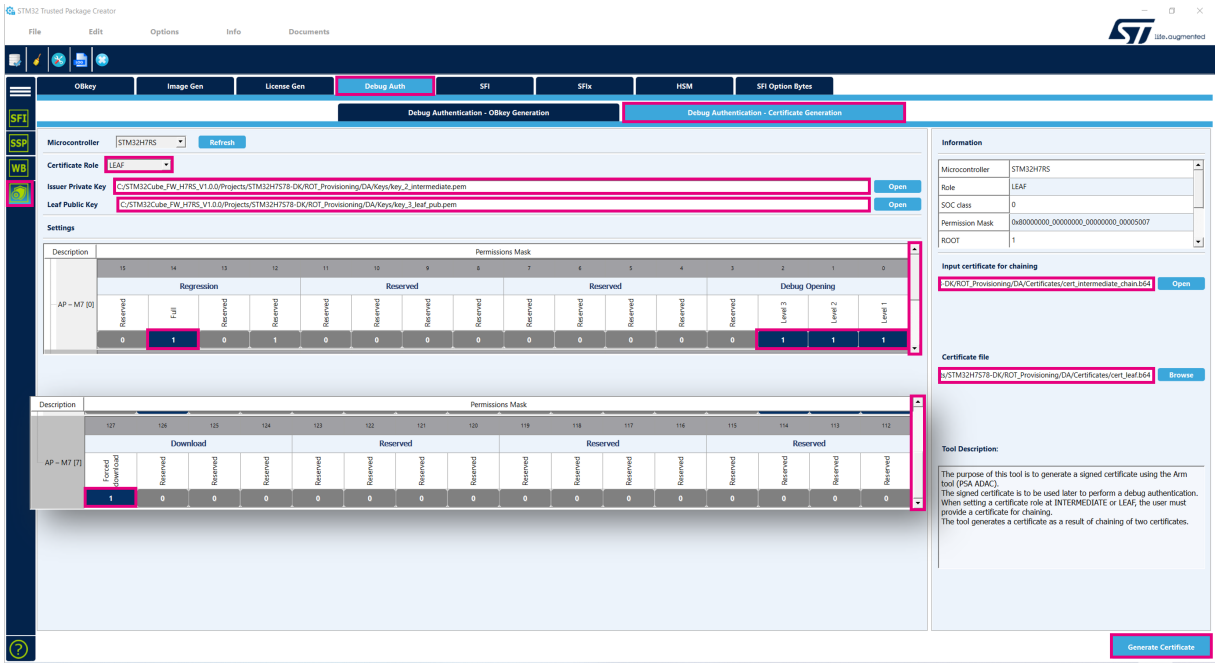


Figure 32. **Leaf certificate**



Each certificate brings additional limitations through the permission mask.

An action (debug opening, regression) is authorized only if the accumulation (logical and) of all permissions masks (certificate, `DA_Config.obk`) allows this operation.

5.3.2 Action execution

A `dbg_auth.bat` script is provided in `C:\STM32Cube_FW_H7RS_V1.0.0\Projects\STM32H7S78-DK\ROT_Provisioning\DA`.

First, the Debug Authentication method is selected.

```
Which debug authentication is used ? [ CERTIFICATE | PASSWORD ]: Certificate
```

When the password method is selected: a full regression is executed after the control of the password.

When the certificate method is selected, this script allows the user to select any action from the permission list.

Figure 33. Action selection from the permission list

```
Start Debug Authentication Sequence

Open SDM Lib
SDMOpen                         :   602 : open       : SDM API v1.0

SDMOpen                         :   603 : open       : SDM Library version v1.1.0

open_comms                      :   495 : open       : Asserting target reset

open_comms                      :   499 : open       : Writing magic number

open_comms                      :   509 : open       : De-asserting target reset

open_comms                      :   561 : open       : Communication with the target established successfully

response_packet_lock
discovery: permission if authorised...........:(a) ==> Full Regression
discovery: permission if authorised...........:(b) ==> Level 3 Intrusive Debug
discovery: permission if authorised...........:(c) ==> Level 2 Intrusive Debug
discovery: permission if authorised...........:(d) ==> Level 1 Intrusive Debug
discovery: permission if authorised...........:(e) ==> Forced download
SDMOpen                         :   602 : open       : SDM API v1.0

SDMOpen                         :   603 : open       : SDM Library version v1.1.0

open_comms                      :   495 : open       : Asserting target reset

open_comms                      :   499 : open       : Writing magic number

open_comms                      :   509 : open       : De-asserting target reset

open_comms                      :   561 : open       : Communication with the target established successfully

[00%]   discovery command
[10%]   sending discovery command
[20%]   receiving discovery
response_packet_lock
[40%]   loading credentials

Keys/key_3_leaf.pem

Certificates/cert_leaf_chain.b64
[50%]   sending challenge request
[60%]   receiving challenge
response_packet_lock
[70%]   signing token
Please select the possible permission/actions:

(a) Full Regression

(b) Level 3 Intrusive Debug

(c) Level 2 Intrusive Debug

(d) Level 1 Intrusive Debug

(e) Forced download
```

Revision history

Table 4. **Document revision history**

Date	Revision	Changes
14-Mar-2024	1	Initial release.
03-Apr-2024	2	Updated figures.
14-Jun-2024	3	Updated to add some information on regression.

Contents

1 General information 2
2 Reference documents 3
3 Preparative procedures 4
3.1 Secure acceptance 4
3.2 Secure installation and secure preparation of the operational environment (AGD_PRE.1.2C) 6
3.2.1 Hardware setup procedure 6
3.2.2 Software setup procedure 6
3.3 Secure installation 7
4 Operational user guidance 9
4.1 User roles 9
4.2 Operational guidance for the integrator role 9
4.2.1 User-accessible functions and privileges (AGD_OPE.1.1C) 9
4.2.2 Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C) 14
4.2.3 Security-relevant events (AGD_OPE.1.4C) 20
4.2.4 Security measures (AGD_OPE.1.6C) 21
4.2.5 Modes of operation (AGD_OPE.1.5C) 21
5 Annex 23
5.1 Secure installation of "TOE_WITH_STIROT" step by step 23
5.1.1 Step 0: Preliminary stage, product state, and method of Debug Authentication selection . 24
5.1.2 Step 1: STiRoT and DA configuration files generation 24
5.1.3 Step 2: Firmware image generation 27
5.1.4 Step 3: Product provisioning 28
5.1.5 STiROT user application example execution 29
5.1.6 Details on `STiROT_Config.obk` 30
5.2 Secure installation of “TOE_WITHOUT_STIROT“ step by step 31
5.2.1 Step 1: DA configuration file generation 32
5.2.2 Step 2: Option bytes provisioning and option bytes programming 33
5.2.3 Step 3: Image flashing 33
5.3 Debug Authentication process 34
5.3.1 Certificate generation 34
5.3.2 Action execution 36
Revision history 37
List of tables 39
List of figures 40

List of tables

Table 1. List of acronyms . . . 2
Table 2. List of reference documents . . . 3
Table 3. Parameter description . . . 30
Table 4. Document revision history . . . 37

List of figures

Figure 1. TOE configurations . . . 7
Figure 2. TOE scope . . . 9
Figure 3. STiRoT provisioning data . . . 14
Figure 4. Memory layout . . . 15
Figure 5. Image format . . . 16
Figure 6. `env.bat` file . . . 23
Figure 7. Launch `provisioning.bat` . . . 23
Figure 8. `provisioning.bat` file execution . . . 24
Figure 9. `provisioning.bat` file execution . . . 24
Figure 10. Generation of `STiRoT_Config.obk` file using STM32TrustedPackageCreator . . . 25
Figure 11. `DA_Config.obk` generation . . . 25
Figure 12. Generation of `DA_Config.obk` file using STM32TrustedPackageCreator . . . 26
Figure 13. Automatic script update . . . 26
Figure 14. Image generation . . . 27
Figure 15. `postbuild.bat` execution . . . 27
Figure 16. `appli_enc_sign.hex` generation . . . 28
Figure 17. `provisioning.bat` launching . . . 28
Figure 18. Product provisioned . . . 28
Figure 19. Virtual COM port configuration . . . 29
Figure 20. User application menu . . . 29
Figure 21. `env.bat` file . . . 31
Figure 22. Launch `provisioning.bat` . . . 31
Figure 23. `provisioning.bat` execution . . . 32
Figure 24. Generation of `DA_Config.obk` file using STM32TrustedPackageCreator . . . 32
Figure 25. Option keys provisioning . . . 33
Figure 26. provisioning.bat execution . . . 33
Figure 27. `provisioning.bat` launching . . . 33
Figure 28. Product provisioned . . . 33
Figure 29. DA certificate chained generation . . . 34
Figure 30. Root certificate . . . 34
Figure 31. Intermediate certificate . . . 35
Figure 32. Leaf certificate . . . 35
Figure 33. Action selection from the permission list . . . 36

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2024 STMicroelectronics – All rights reserved