Getting started with STiRoT (ST Immutable Root of Trust) for STM32H7Sxx MCUs

Introduction

When deploying a device in an untrusted environment, it is important to consider the potential threats that may compromise the device security.

To mitigate these risks, it is recommended to allow only authentic firmware to run on the device.

Firmware updates are common for connected devices to fix bugs, introduce new features, or deploy new security improvements. However, it is important to install these updates securely to ensure the integrity of the device. Without proper security measures, firmware updates can result in serious consequences such as firmware cloning, malicious software downloads, or device corruption. To protect sensitive data and critical operations, security solutions must be designed leveraging cryptography and memory protections.

Cryptography ensures the authenticity and confidentiality of firmware update, while memory protection mechanisms prevent unauthorized access to the device.

This application note gives an overview of the STiRoT solution integrated in STM32H7Sxx microcontrollers, based on the Arm® Cortex®-M7 processor, with its associated ecosystem. How to use it, step by step, is described at [3].

Table 1. Applicable products

Type	Products
Microcontroller	STM32H7S7A8, STM32H7S7I8, STM32H7S7L8, STM32H7S7L8U, STM32H7S7Z8 STM32H7S3A8, STM32H7S3I8, STM32H7S3L8, STM32H7S3L8U STM32H7S3R8, STM32H7S3V8, STM32H7S3Z8

1 General information

This document applies to STM32H7Sxx Arm® Cortex®-M7-based microcontrollers.

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

Table 2. **Glossary**

Acronym	Definition
BL	Bootloader
HDPL	Hardware protection level
HUK	Hardware unique key
iLoader	Immutable loader software
MCE	Memory cipher engine
MCU	Microcontroller unit
MPU	Memory protection unit
OB	Option bytes
OBKeys	Option bytes keys
RSS	Root security services
SBSFU	Secure boot and secure firmware update
SESIP	Security evaluation standard for IOT platform
STiRoT	ST immutable (unchangeable) Root of Trust
uRoT	Updatable Root of Trust
TLV	Type length value
TPC	STM32 Trusted Package Creator

2 References

Table 3. List of documents

Reference	Name/address	Title
[1]	AN6008	*Introduction to Debug Authentication (DA) for STM32 MCUs* application note
[2]	RM0477	*STM32H7Rx/Sx Arm®-based 32-bit MCUs* reference manual
[3]	https://wiki.st.com/stm32mcu/wiki/Security:How_to_start_with_STiRoT_on_STM32H7S	*How to start with STiRoT on STM32H7S* wiki article
[4]	https://wiki.st.com/stm32mcu/wiki/Security:OEMiRoT_OEMuRoT_for_STM32H7S	*OEMiRoT/OEMuRoT for STM32H7S* wiki article
[5]	https://wiki.st.com/stm32mcu/wiki/Security:STiRoT_for_STM32H7S	*STiRoT for STM32H7S* wiki article
[6]	AN6022	Introduction to SFSP versions for STM32H7Rx/7Sx MCUs

Table 4. External references

Reference	Name/address	Title
[7]	https://www.trustedfirmware.org/projects/mcuboot/index.html	MCUBoot (1)

1. *This URL belongs to a third-party. It is active at document publication. However, STMicroelectronics shall not be liable for any change, move, or inactivation of the URL or the referenced material.*

3 STiRoT presentation

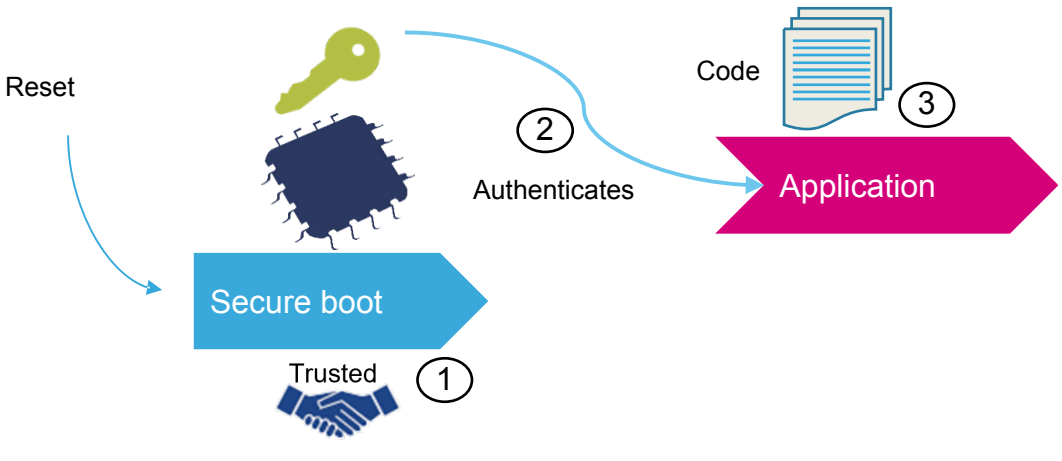
3.1 Overview

STiRoT stands for **ST I**mmutable (unchangeable) **R**oot **o**f **T**rust, and acts as the first boot stage. STiRoT targets a SESIP level 3 certified implementation.

STiRoT is embedded in an immutable area of the STM32H7Sxx and provides two services:

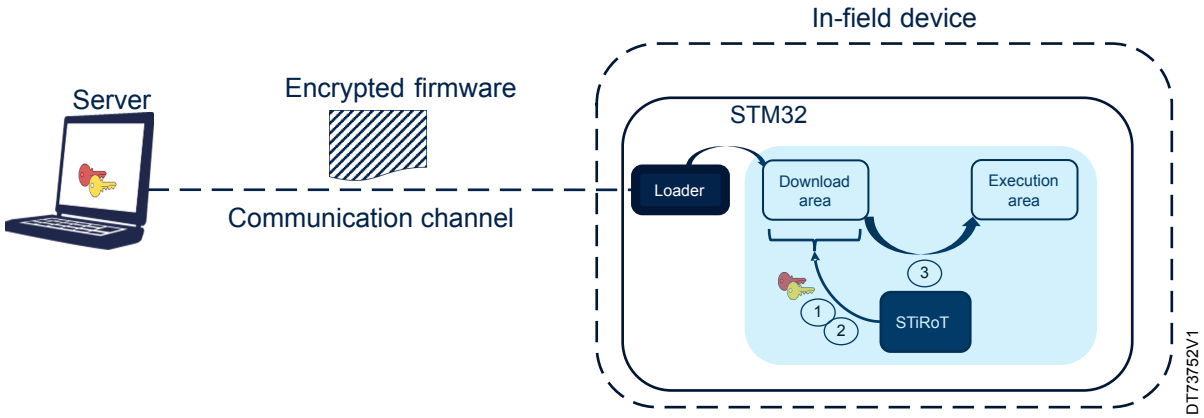
- The secure boot (root of trust services) is an immutable code, which is always executed after a system reset (1). It activates runtime protections, then it verifies the authenticity and integrity of the application (2) code before every execution (3).

Figure 1. Secure boot



- The secure firmware update application is an immutable code that detects when a new firmware image is available. It verifies the image authenticity (1), integrity (2) and finally installs the decrypted firmware (3).

Figure 2. Secure firmware update



The detailed startup sequence is described in Appendix A STiRoT start-up sequence as well as the most efficient way of working during the development phase is described in Application development phase.

3.2 Protection measures and security strategy

Cryptography ensures integrity, authentication, and confidentiality. However, the use of cryptography alone is not enough. A set of hardware and software security measures are supported by the STiRoT. The goal is to protect critical operations and sensitive data (such as a secret key). SESIP level 3 certification is targeted.

3.3 STiRoT design

STM32H7Sxx hardware integrates only 64 Kbytes of user flash. On this series, the user application is stored in an external flash memory and executed from an external flash memory, external RAM, or internal RAM.

STiRoT, which is an immutable code, is designed to be independent from external flash memory selection. Therefore, the control of the authenticity and integrity of the user application and its execution are done by STiRoT from the internal RAM. STiRoT runs in isolation level 1 (HDPL1).

An iLoader project is provided to do the interface between the internal RAM and the external flash memory. This project can be adapted to the external flash memory selected by the customer. The iLoader binary is programmed in user flash memory during the provisioning process (refer to Section 4 for more details). Note that the user flash memory is limited to 64 Kbytes. iLoader runs in isolation level 3 (HDPL3) and has therefore no access to the STiRoT configuration data (including the encryption and authentication keys) stored in HDPL1 OBKeys.

Depending on its state machine, STiRoT requests iLoader:

- To copy the user application code image from the external flash memory to the internal RAM. The purpose is to detect if a new user application image must be installed or to verify if an existing user application image can be executed.
- To copy the user application code from the internal RAM memory to the external flash memory at the end of the installation process. During the copy, the user application code image is reencrypted by MCE peripheral. The encryption key is configured and locked inside MCE by STiRoT.

A reset is generated after each iLoader action to continue the secure firmware update and the secure boot process.

3.4 STiRoT activation

On the STM32H7Sxx devices, STiRoT activation is done by:

1. Configuring the `IROT_SELECT` option byte to boot on STiRoT (refer to [2] for more details).
2. Defining STiRoT configuration.
   The main parameters for STiRoT configuration are:
   - Location, size of each area hosting user application code image in external flash memory and in internal RAM.
   - Authentication and encryption keys used by STiRoT.

   Section 5 provides details on all configuration parameters.

Both operations are part of the provisioning process.

STiRoT is activated in two different use cases:

- One boot stage: STiRoT integrated inside the STM32H7Sxx manages directly the user application.
- Two boot stages: STiRoT integrated inside the STM32H7Sxx manages an updatable boot stage (uRoT) located in the external flash memory, which manages the user application. The updatable boot stage can be customized to fit customer needs.

3.4.1 One boot stage

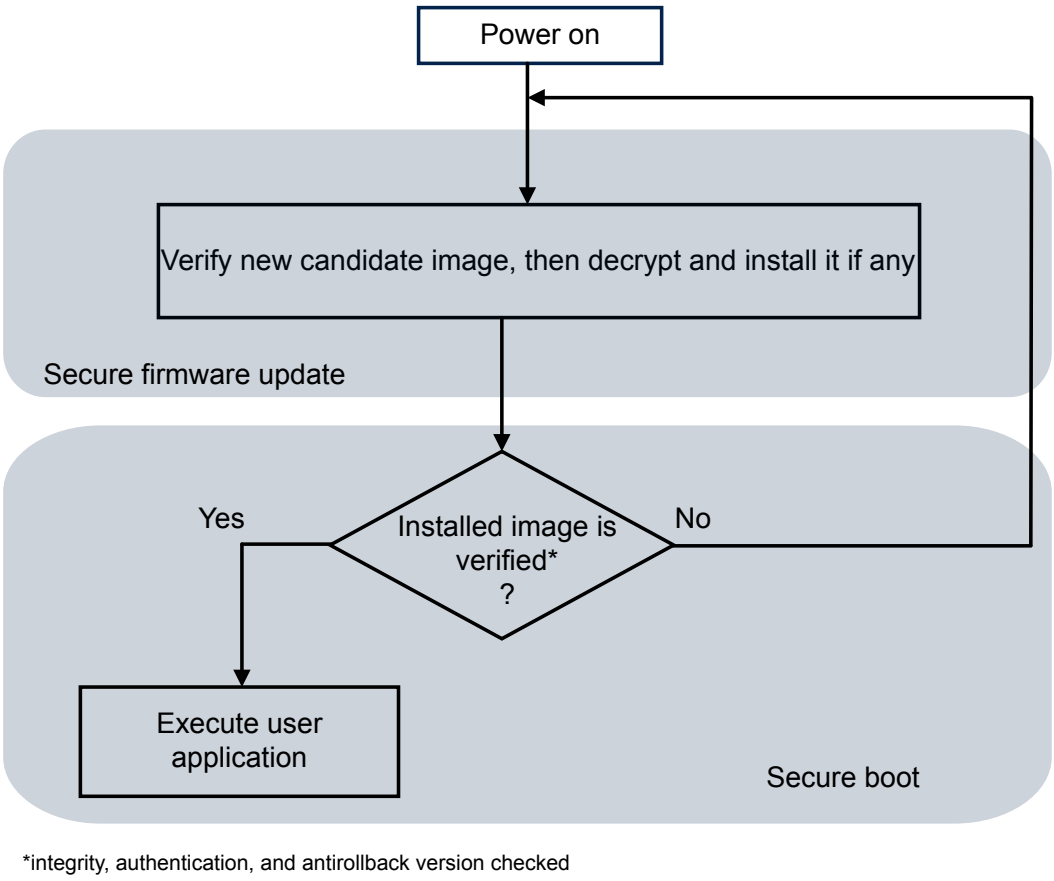
At reset, STiRoT is executed in temporal isolation 1 (HDPL1). Refer to [2] for more details on temporal isolation levels.

The startup sequence is the following:

First, StiRoT checks if any new candidate image needs to be installed. For that, the "image magic" tag (see also Table 10) must be detected in the download slot. In this case, STiRoT checks its version as a version downgrade prevention, its authenticity, and its integrity before installing it after decryption.

Then STiRoT controls the integrity, the authenticity, and the antirollback version of the installed image. In case of success, STiRoT executes the user application in HDPL2. In the case of a verification failure, a reset is triggered in loop. The debug authentication process must be executed to force the download of a new image. Refer to [1] for more details.

Figure 3. STiRoT startup sequence



*integrity, authentication, and antirollback version checked

Figure 4 illustrates the secure firmware update process. This demonstrates the necessary procedures for installing a new user application code image from the download slot to the installation slot in the external flash memory:

1. In the CLOSED or LOCKED state, RSS is the first software executed. If the `IROT_SELECT` option byte is configured to boot on STiRoT, RSS jumps into STiRoT.
2. STiRoT requests iLoader to load the new candidate image from the external flash memory to the internal RAM. iLoader is programmed in the user flash memory and can be customized if a different external flash memory is used. If no new candidate image is detected, the secure firmware update process is ended and the secure boot process is started.
3. iLoader copies the new image from the download slot, which is located in the external flash memory, to the internal RAM, then generates a reset.
4. At reset, STiRoT decrypts the new user application image in the internal RAM, then controls its integrity, its authenticity, and its antirollback version. If the control is successful, STIRoT requests iLoader to save the new user application code image in the external flash memory.
5. iLoader copies the new image from internal RAM to the installation slot located in the external flash memory, then generates a reset. During the copy, the user application code image is reencrypted by MCE_1 peripheral. The encryption key has been configured and locked inside MCE_1 by STiRoT.

Figure 4. Secure firmware update

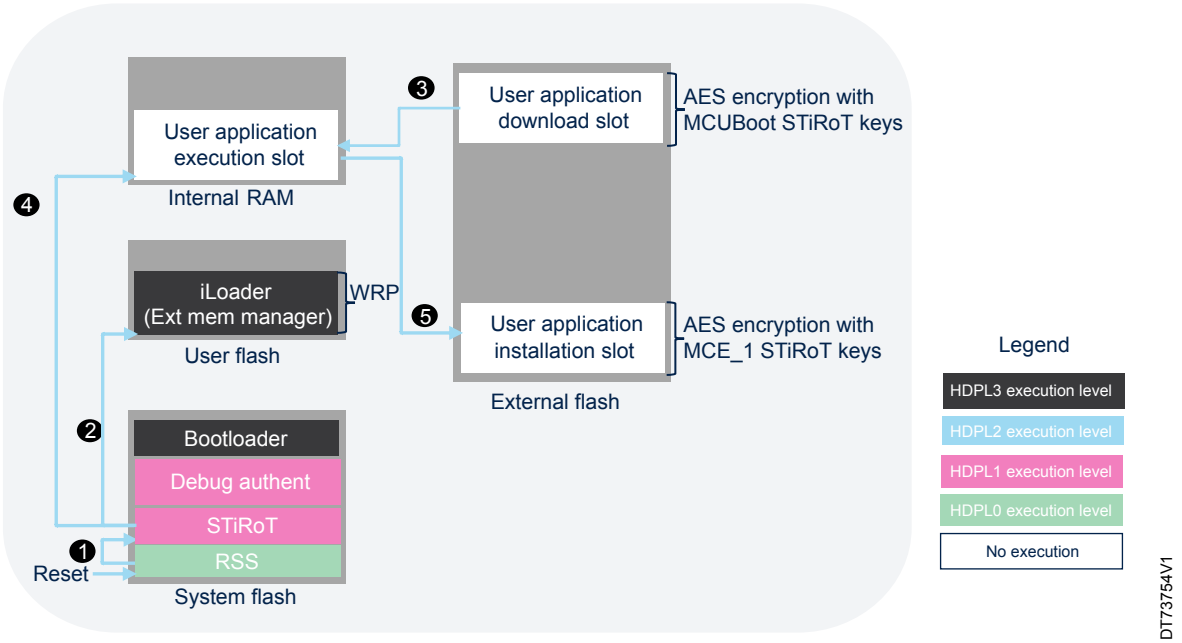
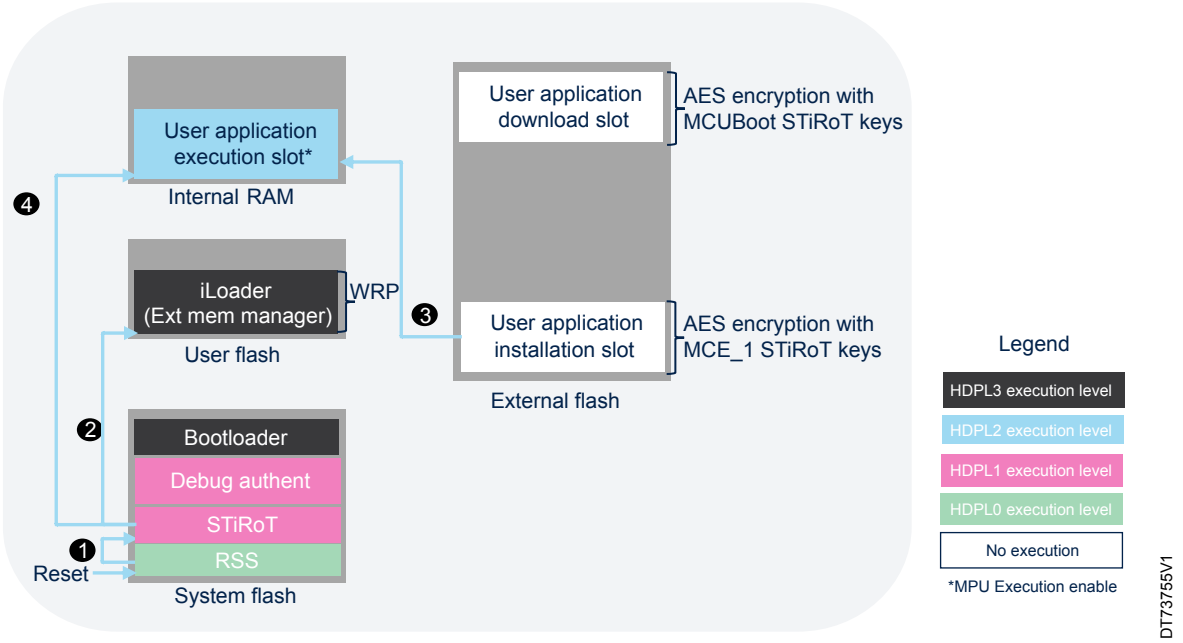


Figure 5 illustrates the secure boot process by showing all the steps required to boot the user application code image located in the installation slot of the external flash memory:

1. In the CLOSED or LOCKED state, RSS jumps into STiRoT if the `IROT_SELECT` option byte is configured to boot on STiRoT.
2. STiRoT requests iLoader to load the installed image in the internal RAM.
3. iLoader copies the installed image from the installation slot, which is located in the external flash memory, to the internal RAM, then generates a reset. The image is automatically decrypted by MCE_1 using the AES key configured and locked inside MCE_1 by STiRoT.
4. At reset, STiRoT controls the integrity and the authenticity of the new image in the internal RAM. If successful, STiRoT jumps into the user application code.

Figure 5. **STiRoT secure boot**



An example of one boot-stage application is provided with the STM32CubeH7RS MCU package.

Note: To ensure the security of the device, the MPU is configured to allow only the user application code area to be executed when STiRoT jumps into the user application, minimizing the risk of unauthorized code execution. It is the user application responsibility to reconfigure the MPU to fit with its security needs.

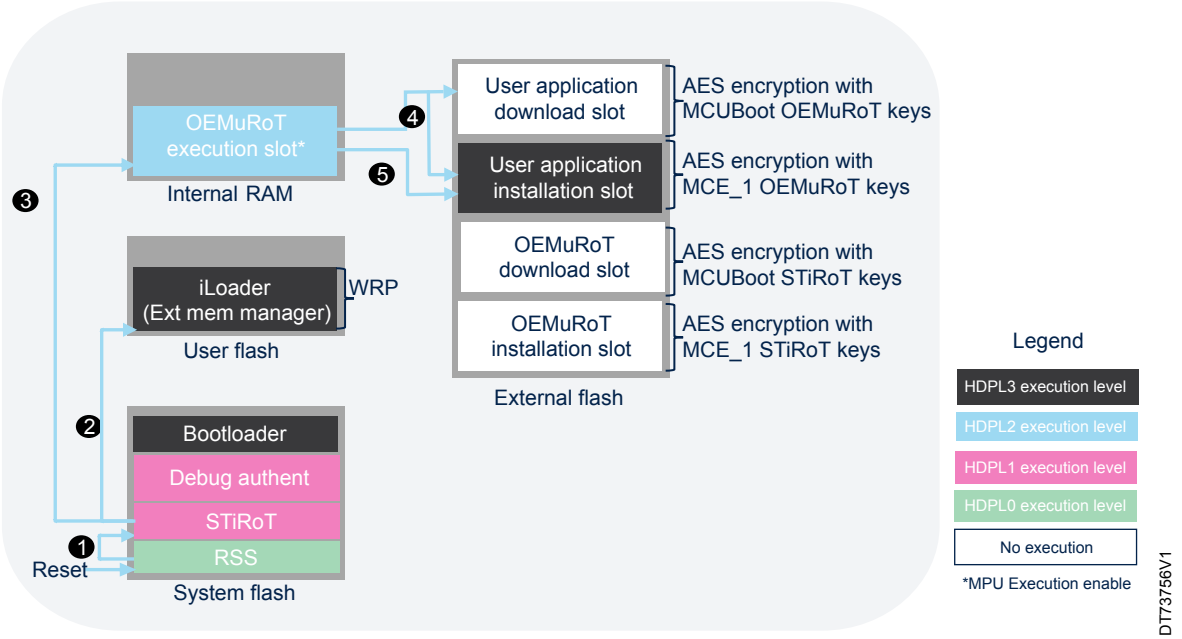
3.4.2 Two boot stages

The OEMuRoT (updatable Root of Trust) located in the user flash memory acts as a second boot stage. At reset, STiRoT is executed in temporal isolation 1, HDPL1. Refer to [2] for more details on temporal isolation levels. After a successful verification of the authenticity and the integrity of the OEMuRoT code image, STiRoT executes OEMuRoT in HDPL2 and in RAM memory. Then, OEMuRoT verifies the user application before executing it from external flash or external RAM memories. In case of a verification failure, the debug authentication process must be executed to force the download of a new image (Refer to [1]).

Figure 6 illustrates the main steps performed to execute a user application in a two boot stage configuration:

1. In the CLOSED or LOCKED state, RSS is the first software executed. If the `IROT_SELECT` option byte is configured to boot on STiRoT, RSS jumps into STiRoT.
2. After executing the secure firmware update process, STiRoT requests iLoader to load the OEMuRoT image from the installation slot into the internal RAM.
3. Then STiRoT controls its integrity and its authenticity. If successful, STiRoT executes OEMuRoT in internal RAM.
4. OEMuRoT checks if a new user application is stored in the download slot. If any, OEMuRoT decrypts it and controls its integrity, its authenticity, and its antirollback version. If successful, the image is copied in the user application installation slot. The image is reencrypted by the MCE_1 with OEMuRoT encryption keys.
5. OEMuRoT controls the integrity and the authenticity of the user application from the installation slot. If successful, OEMuRoT executes it from external flash memory. The user application can also be executed from external or internal RAM. Refer to [4] for more details on OEMuRoT configuration capabilities.

Figure 6. STiRoT - User application executed after OEMuRoT



An example of a two boot stage application is provided with the STM32CubeH7RS MCU package.

Note: To ensure the security of the device, the MPU is configured to allow only the uRoT code area to be executed when STiRoT jumps into uRoT, thus minimizing the risk of unauthorized code execution. It is the uRoT responsibility to reconfigure the MPU to fit with its security needs. In this case, the security of the user application depends on the uRoT strategy.

3.5 List of features

The main features of the STiRoT are:

- Hardware-accelerated cryptography operations with PKA, HASH, SAES and MCE peripherals :
  - ECDSA-P256 asymmetric cryptography for image authentication.
  - SHA256 cryptography for image integrity check.
  - AES-CTR-128 cryptography for image encryption with symmetric key encrypted in ECIES-P256 provided in the image itself. Refer to Section 6 for more details.
  - AES-ECB-128 cryptography for image encryption with a MCE_1 key after the installation in an external memory flash. MCE_1 key is derived from HUK.
- Accelerated boot thanks to the image SHA256 reference management.
  Image verification mainly consists of verifying the SHA256 of the image (integrity check), and then verifying the signature of this SHA256 (authentication check). If successful, the SHA256 is stored as a reference for the next boot. At the next boot, the signature verification is skipped if the SHA256 of the image matches the reference stored in HDPL1 OBKeys. This feature brings a performance optimization.
- The antirollback version checks based on the image counter stored in OBKeys.
- Image installation is resistant to asynchronous power down and reset.
- Integration of the full entropy TRNG source (RNG hardware peripheral) for random delays generation.
- Memory configuration including:
  - Download slot in the external flash memory. The image is encrypted in AES-CTR-128 with a symmetric key. This key is encrypted in ECIES P256 with the MCUboot key.
  - Installation slot in the external flash memory. This slot is encrypted in AES-ECB-128 with the MCE_1 based on a key derived from HUK.
  - Execution slot in the internal RAM.
  - iLoader area in the internal flash memory.
- Clock selection: STiRoT can be executed at the 64 MHz (default configuration) or 380 MHz. 380 MHz is the best speed configuration that supports the full range of temperatures (V_{CC} must be adapted).
- Integration of the hardware security peripherals and mechanisms to implement a root of trust. Product life cycle, IROT_SELECT, MPU, WRP, HDP, and TAMPER are combined to achieve the highest security level.
- Image generation with the STM32 Trusted Package Creator tool delivered within STM32CubeProgrammer.

3.6 STiRoT constraints

The list of constraints to be considered when activating the STiRoT boot path is the following:

- The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to prevent malfunctions due to software or hardware failures in a disturbed environment. For application requesting such high safety level, it is recommended to activate IWDG (controlled by hardware) via option bytes when using STiRoT and regularly reload the IWDG on the application side.
- WWDG activation by option byte is not supported by STiRoT. Only IWDG can be activated through option byte.

- Internal tamper 9 (fault generation for cryptographic peripherals (SAES, PKA, AES, RNG)) and internal tamper 15 (system fault detection) are activated in confirmed mode during STiRoT execution and remain activated when jumping into the user application. Reset is performed in case of tamper event detection during STiRoT execution. If a tamper event occurs during STiRoT execution, the system performs a reset. The content of the TAMP status register and a magic value are saved at the end of the backup SRAM. Then the application can check for the presence of this magic value to detect if a tamper occurred, decide the appropriate action, and finally clear the magic value.

Table 5. BACKUP SRAM mapping

BACKUP SRAM Address	Size (Bytes)	Description
0x3880 0FFC	0x4	Tamper status register copy (indicates which tamper(s) have occurred)
0x3880 0FF8	0x4	Magic value (if a tamper fault occurred during STiRoT, it is set to 0x003981BB)

Note: It is applicable from the SFSP version 2.1.0 on STM32H7Sxx devices. For previous SFSP version, when a tamper fault occurs, the system resets in loop until a backup domain power-on reset is performed. Refer to [6] to get all the information for SFSP version tracking.

- In Standby and VBAT modes, tampers configured in potential mode are not supported by STiRoT.
- MPU is configured to allow only the user application code area to be executed when STiRoT jumps into the user application. It is the user application responsibility to reconfigure the MPU to fit its security needs.

4 Provisioning process

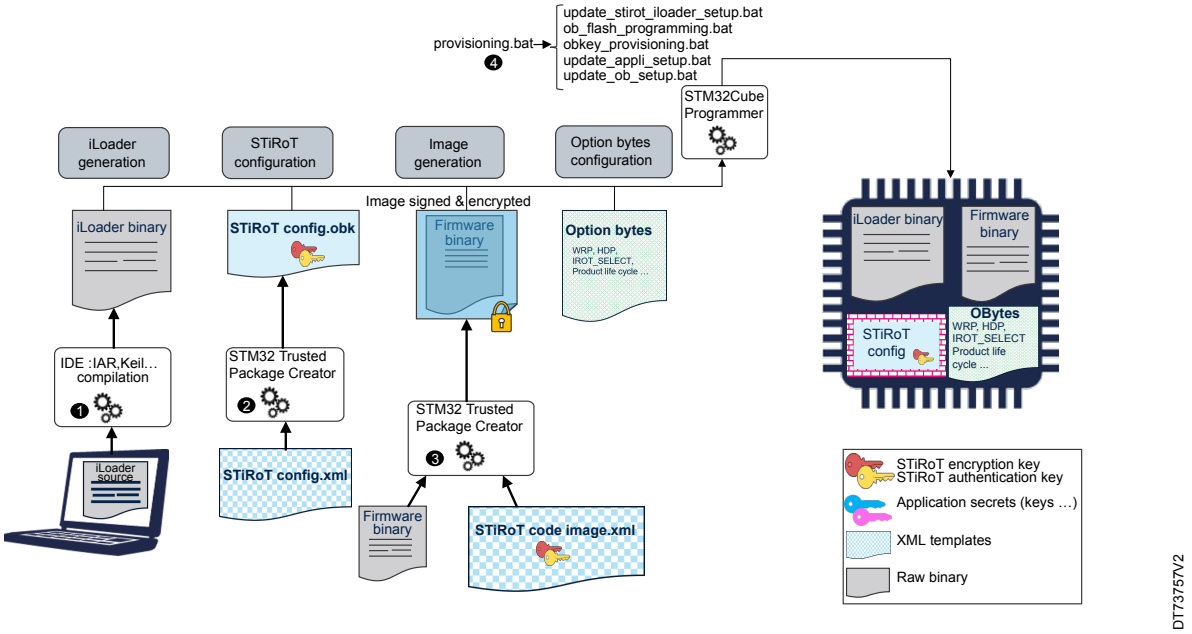
The product provisioning to activate and configure STiRoT is done following the four steps below when executing provisioning.bat.

1. Generation of iLoader binary. iLoader can be customized if a different external flash is used.
2. Configuration of the STiRoT
   At this stage, the location of the image, and the cryptographic keys are defined.
3. Generation of the code image
4. Programming of the OBKeys, the option bytes, the iLoader, and the image in the device

Note: A set of scripts is provided in the STM32CubeH7RS MCU package (`Firmware/Projects/STM32H7S78-DK/ROT_Provisioning/STiROT` folder). It guides the user all along the provisioning process.

Refer to [3] for more information.

Figure 7. STiRoT provisioning process



5 OBKeys configuration file

The STiRoT OBKeys configuration file (STiRoT_Config.obk) is generated using STM32 Trusted Package Creator with a template file listing all the different parameters (STiRoT_Config.xml) as inputs.

The STiRoT configuration provides the possibility to:

- Define the RAM and the external flash memory mapping: select the location and the size of each area.
- Configure the authentication and encryption keys.

Some parameters are considered as advanced configuration and are, therefore, hidden when the STiRoT configuration is displayed by STM32 Trusted Package Creator. If required, the `<Hidden> </Hidden>` property can be switched from 1 to 0 in the STiRoT_Config.xml file.

The list of all the parameters is provided in the following Table 6.

Table 6. STiRoT configuration

Parameter	Hidden	Description	Additional controls/constraints
Clock selection	Yes	0 (default value): 64 MHz, default configuration 1: 380 MHz, best speed configuration supporting the full range of temperature (V_{CC} must be adapted).	-
SRAM ECC management activation	Yes	Yes (default value): a reset is generated in case of SRAM ECC detection. SRAM ECC is a mechanism to detect memory tampering. No: ECC is not managed when SRAM tampering detection is not required.	Expected configuration of option bytes.
Firmware execution area address in RAM	No	Firmware execution area address in RAM	It must be aligned on 16 Kbytes (MPU configuration constraint).
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 product state allowed	Expected configuration of option bytes.
Firmware installation area address in external flash memory	No	Firmware installation area address in external flash memory	It must be aligned on 64 Kbytes (external flash constraint). No overlap is allowed between execution areas and download areas.
Firmware download area address in external flash memory	No	Firmware download area address in external flash memory	It must be aligned on 64 Kbytes (external flash constraint). No overlap is allowed between execution and download areas.
External flash memory size	Yes	External flash memory size	Information used to control the mapping of the firmware areas in the device
iLoader offset	Yes	Offset from the beginning of the user flash where the iLoader is located	WRP protection is automatically set during provisioning on the iLoader area.
iLoader size	Yes	Size of the iLoader	WRP protection is automatically set during provisioning on the iLoader area.
Encryption key	No	The key is used to encrypt the firmware image.	When this key is regenerated, both firmware and data images must be regenerated with the STM32 Trusted Package Creator “Image Gen” tab. (STiRoT_Code_Image.xml and STiRoT_Data_Image.xml)
Authentication key	No	The key is used to authenticate the firmware image.	When this key is regenerated, both firmware and data images must be regenerated with the STM32 Trusted Package Creator “Image Gen” tab. (STiRoT_Code_Image.xml and STiRoT_Data_Image.xml)

A detailed presentation of the format of the STiRoT_Config.obk file is provided in Appendix C: STiRoT Config.obk file format.

6 Image generation

STiRoT manages images, based on MCUboot format, that include:

- A header
- The encrypted firmware or data binary
- Metadata information (TLV format: tags length value) allowing the control of the image (SHA256, ECDSA-P256 signature, …)
- A magic number to trigger the installation at the end of the slot

Further information about the MCUboot open-source software is available at [7].

STM32 Trusted Package Creator is provided to generate the firmware image. This image is encrypted and signed using the keys configured in the OBKeys configuration file (STiRoT_Config.xml).

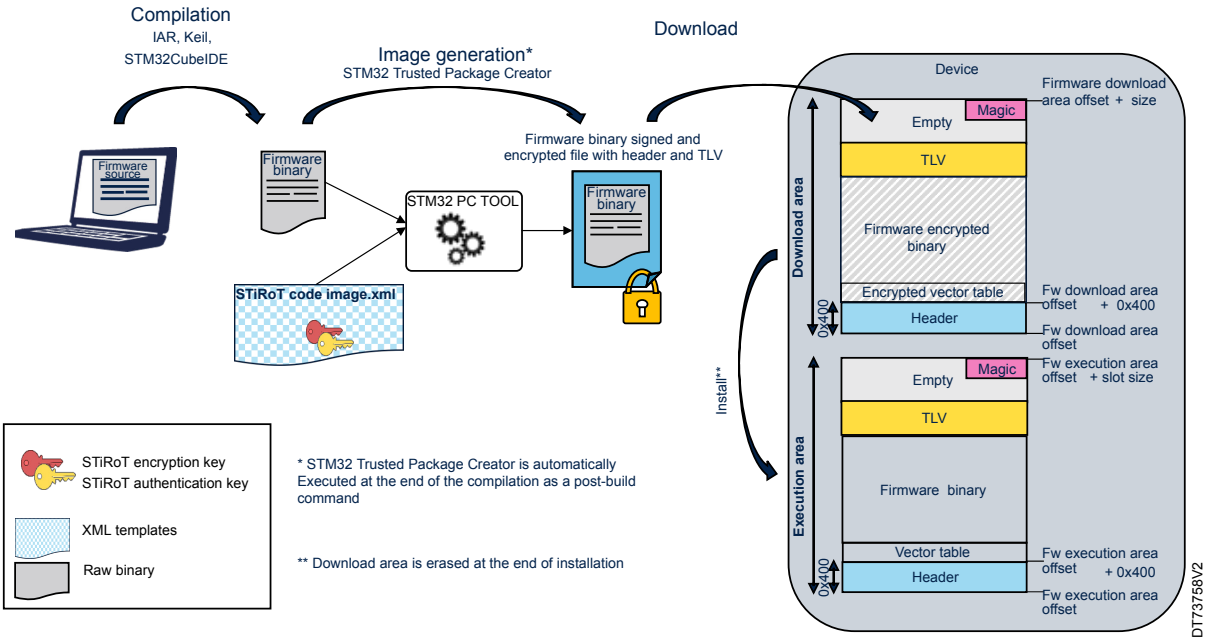
STiRoT_Code_Image.xml contains all the parameters driving the image generation such as:

- The authentication and encryption keys
- The version and the dependency information

STiRoT_Code_Image.xml can be edited with STM32 Trusted Package Creator to modify the version. The keys are directly inherited from the STiRoT_Config.xml file.

Figure 8 shows the firmware image generation.

Figure 8. Firmware image generation

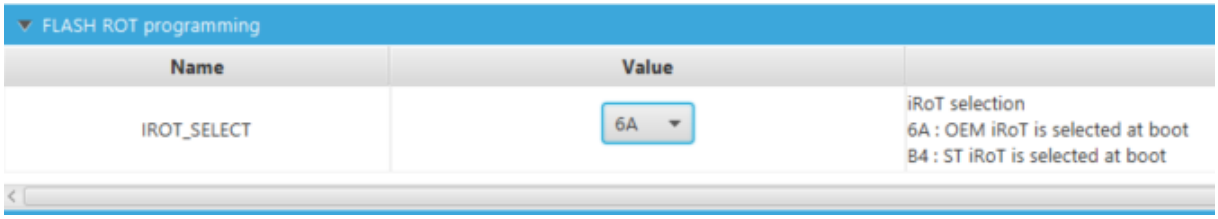


The list of all parameters is detailed in Appendix E: Code image XML file.

Appendix A Application development phase

The most efficient way to develop and debug an application is to develop it without activating the STiRoT secured boot. The `IROT_SELECT` option byte is configured to boot on user flash (`0x6A`) as shown in Figure 9.

Figure 9. Boot directly in user flash memory



As long as the device is in an open state, the user flash memory can be erased without performing a regression, and a debugger can be connected without performing a Debug Authentication. Refer to [1] for more details. Once validated, the secure boot path can be activated by configuring STiRoT through the provisioning process. The linker file must be updated to be compliant with the *STiRoT_Appli* template project provided with the STM32CubeH7RS MCU package

Refer to Appendix F: STiRoT execution status for more information on debug information when STiRoT is activated.

Appendix B Performance

Table 7 shows the boot time that depends on the size of the user application and the clock selected :

Table 7. Performance data

Configuration	Boot time	
	128 Kbytes	376 Kbytes
64 MHz	< 200 ms	< 296 ms
380 MHz	< 100 ms	< 120 ms

Appendix C STiRoT Config.obk file format

Note: Pay attention to the definition of endianness. `uint32_t 0x01234567` *is defined in the obk file as* `0x67452312`*.*

Table 8. STiRoT_Config.obk format

Offset	Size	Type	Description
0	1	uint8_t	sdp command header : destination OBKey index
1	1	uint8_t	sdp command header : destination OBKey HDP level
2	2	uint16_t	sdp command header : reserved
4	4	uint32_t	sdp command header : data size (number of bytes)
8	4	uint32_t	sdp command header : encryption disabled/enabled(0/1)
12	32	array	SHA256 of the following data content (from offset 44 to 267)
44	1	uint8_t	Reserved
45	1	uint8_t	Clock selection (64/380 MHz) (0/1)
46	1	uint8_t	Reserved
47	1	uint8_t	SRAM ECC management activation disabled/enabled (0/1)
48	4	uint32_t	Firmware area address (in internal RAM)
52	4	uint32_t	Firmware area size
56	16	uint8_t	Reserved
72	4	uint32_t	Product state minimal allowed: 0x72 (closed), 0x5C (locked)
76	4	uint32_t	Firmware installation area address (in external flash memory)
80	4	uint32_t	Firmware download area address (in external flash memory)
84	4	uint32_t	Reserved
88	4	uint32_t	External nonvolatile memory size
92	4	uint32_t	iLoader offset from the begin of the internal flash memory (`0x0000, 0x2000 ... 0xE000`)
96	4	uint32_t	iLoader size
100	7	uint8_t	Reserved
107	70	array	ECDSA-P256 encryption private key
177	91	array	ECDSA-P256 authentication public key

Appendix D STiRoT_Data.obk file format

STiRoT_Data.obk is part of the provisioning process to initialize the images version antirollback counters (zero value) and code image SHA256 reference (zero value).

Be careful to endianness: `uint32_t 0x01234567` is represented in the obk file as `0x67452301`.

Table 9. STiRoT_Data.obk file format

Offset	Size	Type	Description
0	1	uint8_t	sdp command header : destination OBKey index
1	1	uint8_t	sdp command header : destination OBKey HDP level
2	2	uint16_t	sdp command header : reserved
4	4	uint32_t	sdp command header : data size
8	4	uint32_t	sdp command header : encryption disabled/enabled(0/1)sdp command header : data size address
12	32	Array	SHA256 of the following data content (from offset 44 to 267)
44	32	Array	SHA256 of the code image installed, automatically updated during the secure boot and secure firmware update process.
76	4	uint32_t	Version of the code image, automatically updated during the secure boot and secure firmware update process.
80	4	uint32_t	Version of the previous code image, automatically updated during the secure boot and secure firmware update process.
84	24	Array	Reserved
108	32	Array	*The data in the offset range [108 to 203] is a copy of the data in the offset range [12 to 107]. It is required to maintain system consistency*
140	32	Array	
172	4	uint32_t	
176	4	uint32_t	
180	24	Array	

Appendix E Code image XML file

Table 10. Firmware image generation

Parameter	Updatable	Description
Authentication key	Automatic	Private key inherited from STiRoT_Config.xml file.
Encryption key	Automatic	Public key inherited from STiRoT_Config.xml file.
Image magic	No	Magic value in image header identifying a firmware code image.
Endianness	No	Little endian.
Padding	No	Add an installation magic value at the end of the image to trigger image installation. Padding with `0xFF` when required.
Firmware area size	Automatic	Firmware slot size information inherited from STiRoT_Config.xml file.
Header size	No	`0x400` bytes.
Padding header	No	Padding the header with `0xFF` to fulfil the `0x400` bytes.
Dependency with data image	No	The dependency is disabled by setting the `<enable>` property to 0.
Write option	No	Image format that is compatible with the primary-only upgrade strategy implemented in STiRoT.
Version	Yes	x.y.z firmware image version.
Security counter	Yes	Security counter value on which antiroll-back feature is relying. Can be set to 'auto' to align image security counter to image version. It can also be set manually to a 32 bits value. Antiroll-back feature is disabled if the security counter value remains unchanged at each version of image.
Align	No	The size of the encrypted binary generated is a multiple of 16 bytes.
Firmware download area offset	Automatic	Used to generate .hex binary file format including the destination address. This parameter is inherited from the STiROT_Config.xml file with the addition of `0x90000000`.
Firmware binary input file	Automatic	Location of the firmware binary file. This parameter is updated during the provisioning process based on information from the env.bat file.
Image output file	Yes	Location of the encrypted image generated. If changed, provisioning scripts must be updated accordingly.

Appendix F STiRoT execution status

STiRoT is executed as a black box, but there is a way to verify the steps that are being executed in case the STiRoT execution stops due to an issue.

For each HDPL, four bytes are reserved for the status log inside the OBKeys area. In STiRoT, the information can be extracted from HDPL1 OBkeys using the debug authentication discovery command (refer to [1] for more information).

Table 11. Execution status

Executed step	Status coding	Description
STIROT_INIT_DONE	0x00000001UL	Device initialization is done. STiRoT boot path is selected. STiRoT is executed.
STIROT_CONFIG_DONE	0x00000002UL	STiRoT configuration according to STiRoT_Config.obk is checked to be correct in OBKeys HDPL1.
STIROT_SECURITY_DONE	0x00000004UL	Security is activated. OBKeys STiRoT configuration is checked to be correct.
STIROT_SLOT_PRIMARY_1_VALID	0x00000010UL	Authenticity, integrity, and antirollback version are checked for the user application firmware image in the execution slot.
STIROT_SLOT_SECONDARY_1_VALID	0x00000040UL	Authenticity and integrity are checked for the user application firmware image in the download slot.
STIROT_PARSING_DONE	0x00000100UL	Slots parsing is done.
STIROT_INSTALLATION_DONE	0x00000200UL	Images in the download slot (one or two images depending on the configuration) are installed.
STIROT_VALIDATION_DONE	0x00000400UL	Execution slot analysis is completed to verify which image is valid (integrity and authenticity).
STIROT_JUMP_BL	0x00001000UL	At least one image is not valid. It leads to bootloader execution.
STIROT_JUMP_APPLI	0x00002000UL	All images (one or two depending on the configuration) are valid → the application execution occurs.
STIROT_NOJUMP	0x00004000UL	At least one image is not valid.
STIROT_JUMP_OEM_ILOADER	0x00008000UL	An error occurred before jumping into iLoader.
STIROT_ILOADER_ERROR	0x00010000UL	An error is being returned by iLoader.

Note: The expected value for a boot without error is `0x00002717`.

Appendix G Application system clock

The system clock is set to 380 MHz to be functional with all hardware configurations (ECC_ON_SRAM enabled, no internal regulator) and with the full range of temperature (Tj up to 125 degrees Celsius). If the constraints related to power and temperature are met, and if the RAMECC security feature is disabled (STiROT_Config.xml), it is possible to increase the system clock up to 600 MHz. The system clock configuration must then be updated in the STiROT_Appli project. A compatible power configuration must be applied in the STiROT_iLoader project before any XSPI configuration.

Revision history

Table 12. **Document revision history**

Date	Revision	Changes
20-Mar-2024	1	Initial release.
06-Jun-2024	2	Updated: • Table 5. STiRoT configuration • Table 8. Firmware image generation • Figure 7. STiRoT provisioning process • Figure 8. Firmware image generation
27-Jan-2025	3	Added • Section 3.6: STiRoT constraints • Appendix F: Application system clock
24-Oct-2025	4	Updated: • Section 3.1: Overview • Section 3.6: STiRoT constraints • Appendix E: Code image XML file Added: • Appendix D: STiRoT_Data.obk file format

Contents

List of tables

List of figures