Getting started with STiRoT (ST immutable Root of Trust) for STM32H5 MCUs

Introduction

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

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

Firmware updates are common for connected devices to fix bugs, introduce new features, or deploy new security improvement. 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 download, 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 Arm® TrustZone® STM32H5 microcontrollers, based on the Arm® Cortex®-M33 processor, with its associated tools ecosystem. How to use it, step by step, is described at [3].

Table 1. Applicable products

Type	Products
Microcontroller	STM32H573RI, STM32H573OI, STM32H573VI, STM32H573ZI, STM32H573II, STM32H573AI, STM32H573MI, STM32H573I-DK, STM32H573IIU, STM32H573IDIE STM32H533ZE, STM32H533VE, STM32H533RE, STM32H533HE, STM32H533CE

1 General information

This document applies to the STM32H5 series Arm® Cortex®-M33-based microcontrollers.

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

Table 2. Terms and abbreviations

Acronym	Definition
BL	Bootloader
HDPL	Hardware protection level
MCU	Microcontroller unit
MPU	Memory protection unit
OB	Option bytes
OBKeys	Option bytes keys
SBSFU	Secure boot and secure firmware update
SESIP	Security evaluation standard for IOT platform
STiRoT	ST immutable root of trust
uRoT	Updatable root of trust
SDP	Secure data provisioning
TLV	Type length value
TPC	STM32 Trusted Package Creator

Reference documents

Table 3. List of documents

Reference	Name/address	Title
[1]	AN6008	*Introduction to Debug Authentication (DA) for STM32 MCUs* application note
[2]	RM0481	*STM32H56x/STM32H573xx and STM32H523xx/STM32H533xx Arm®-based 32-bit MCUs* reference manual
[3]	https://wiki.st.com/stm32mcu/wiki/Category:How_to_start_with_STiROT_on_STM32H5	*How to start with STiRoT on STM32H5* wiki article(1)
[4]	https://wiki.st.com/stm32mcu/wiki/Security:How_to_start_with_STiRoT_on_STM32H573	*How to start with STiRoT on STM32H573* wiki article(1)

1. *This URL is active at document publication. However, STMicroelectronics shall not be liable for any change, move, or inactivation of the URL or the referenced material.*

Table 4. External references

Reference	Name/address	Title
[5]	http://mcuboot.com	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.*

2 STiRoT presentation

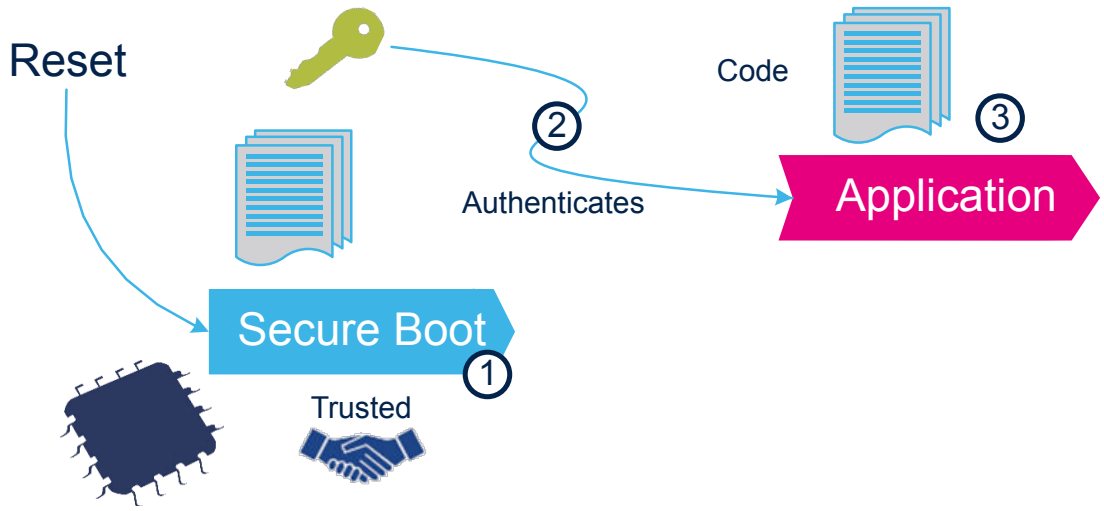
2.1 Overview

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 STM32H5 and provides two services:

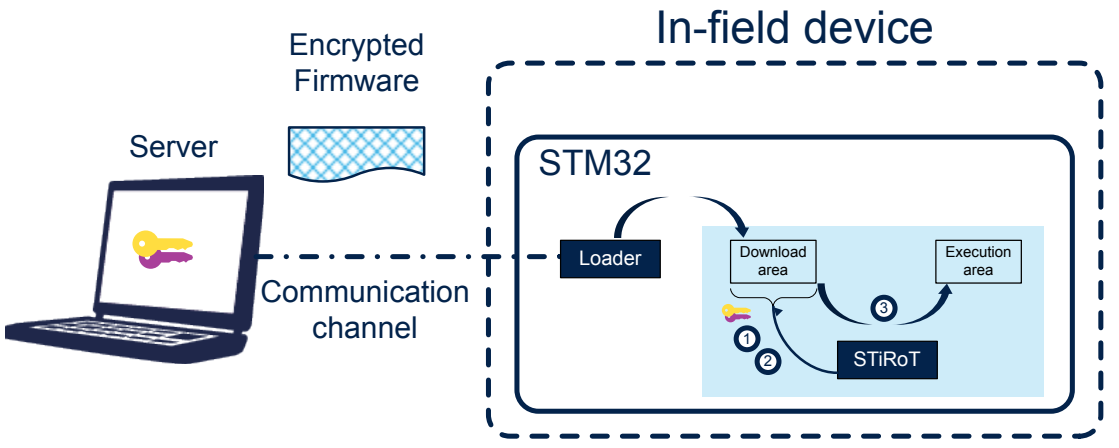
- The Secure Boot (root of trust services) is an immutable code, which is always executed after a system reset (1). It activates runtime protections and then, it verifies the authenticity and integrity of the application (2) code before every execution (3).

Figure 1. Secure Boot



- The Secure Firmware Update application is an immutable code that detects that a new firmware image is available. It checks its authenticity (1), then checks the integrity (2) of the code before installing it after decryption (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 development phase in Appendix B: Application development phase.

2.2 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 for protecting critical operations and sensitive data (such as a secret key). The SESIP level 3 certification is targeted.

The following cryptographic scheme is implemented in STiRoT:

- ECDSA-P256 asymmetric cryptography for image authentication.
- AES-CTR-128 symmetric cryptography with key ECIES-P256 encrypted for image confidentiality.
- SHA256 cryptography for image integrity check.
- All cryptography operations are hardware-accelerated.

2.3 STiRoT activation

On STM32H5, STiRoT activation is done by:

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

   Section 4 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:** The STiRoT integrated inside the STM32H5 manages directly the user application.
- **Two boot stages:** The STiRoT integrated inside the STM32H5 manages an updatable boot stage (uRoT) located in the user flash memory which manages the user application. The updatable boot stage can be customized to fit customer needs.

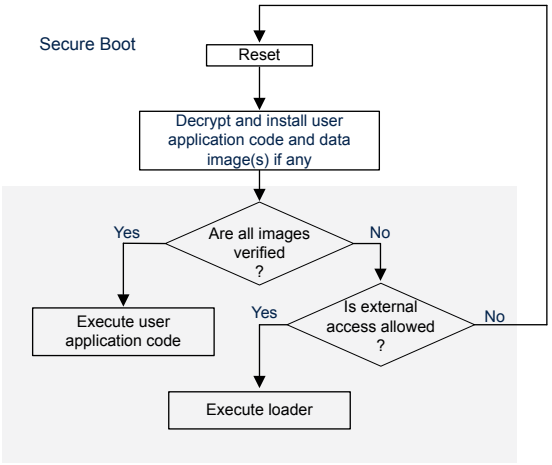
Table 5. Boot time performance

Configuration	Boot time		
-	**256 Kbytes**	**512 Kbytes**	**1 Mbytes**
Execution @ 64MHz	41,5 ms	70 ms	128 ms
Execution @ 200MHz	18 ms	27,5ms	46ms
Execution @ 250MHz	16ms	23,5ms	38ms

2.3.1 One 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 user application code and data images, STiRoT execute the user application in secure mode (HDPL2). In case of verification failure, STiRoT jumps into the bootloader to provide the end user a way to download a new user application code and/or data images if the external access is configured as allowed.

Figure 3. STiRoT behavior in case of verification failure



DT73292V1

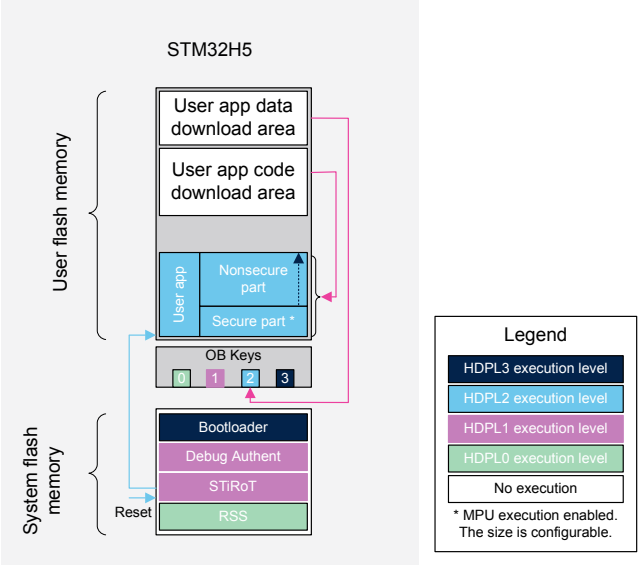
The user application can be fully secure or divided in a secure and nonsecure application. In both cases, the image stored in the download area is a single image. The secure part of the user application must contain at least the vector table and the reset handler.If this is not the case, the application does not start as STiRoT executed it in secure mode.

The secrets (data) associated to the user application are installed by STiRoT in the protected HDPL2 OBKeys. When no secrets are required by the application, the data image management can be deactivated through STiRoT configuration.

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

The following figure gives an example of the memory mapping when STiRoT is activated in a one boot stage configuration.

Figure 4. STM32H5 - STiRoT - User application executed after STiRoT



DT73293V1

An example of one boot stage application is provided with STM32CubeH5 MCU Package.

Note: When required, this is the user application responsibility to move to HDPL3 and to switch in nonsecure mode.

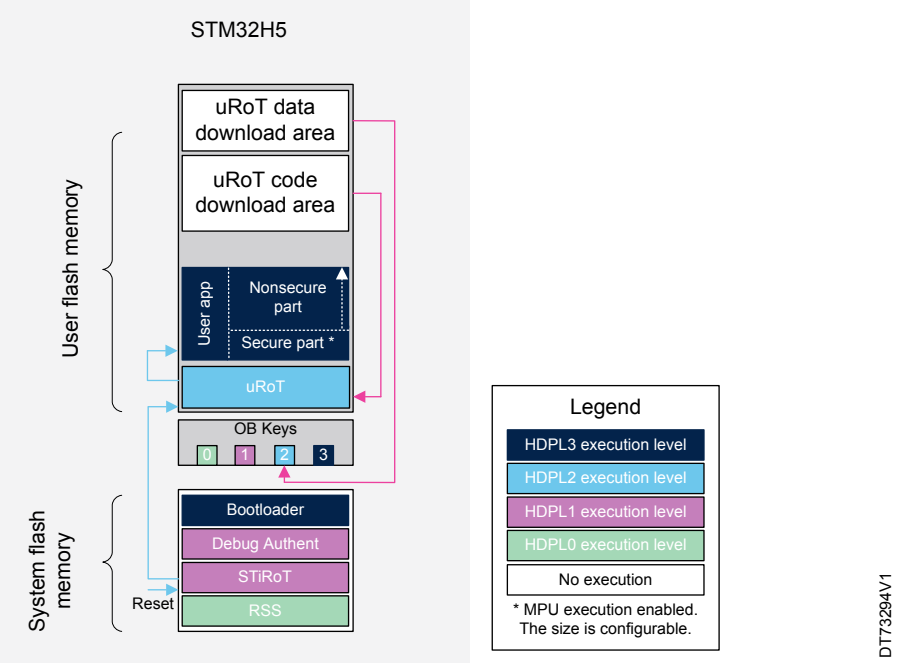
2.3.2 Two boot stages

The uRoT (updatable Root of Trust) located in the user flash memory act 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 uRoT code and data images, STiRoT executes uRoT in HDPL2 and in secure mode. Then, uRoT verify the user application before executing it. In case of verification failure of uRoT code and/or data image, STiRoT jumps into the bootloader to provide the end user a way to download a new uRoT code and/or data images if the external access is configured as allowed. See Figure 3. STiRoT behavior in case of verification failure in case of verification failure.

The uRoT is a fully secure application. The secret keys managed by uRoT and hosted in the data image are installed by STiRoT in the protected HDPL2 OBKeys.

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, minimizing the risk of unauthorized code execution. It is the uRoT responsibility to reconfigure the secure MPU to fit with its security needs. In this case, the security of the user application depends on the uRoT strategy.

Figure 5. STM32H5 - STiRoT - User application executed after uRoT



An example of a two boot stages application is provided with the STM32H5 series MCU Package.

2.4 STiRoT constraints

The list of constraints to be taken into account 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 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 by default in confirmed mode during STiRoT execution and remains activated when jumping into the user application. They can be deactivated through STiRoT OBKeys configuration. If a tamper event occurs during STiRoT execution, a reset is performed and the content of the TAMP status register is written in the last 32-bits of the BKPSRAM and a 32-bits magic value (0x003981bb) is placed just before. Based on this information, the application can take the appropriate action, and then clear the magic value.
- In Standby and VBAT modes, tampers configured in potential mode are not supported by STiRoT.
- MPU is configured to only allow 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 with its security needs.
- SRAM2 is used by STiRoT and is therefore erased at each reset.
- SWAP_BANK activation by option byte is not supported by STiRoT.

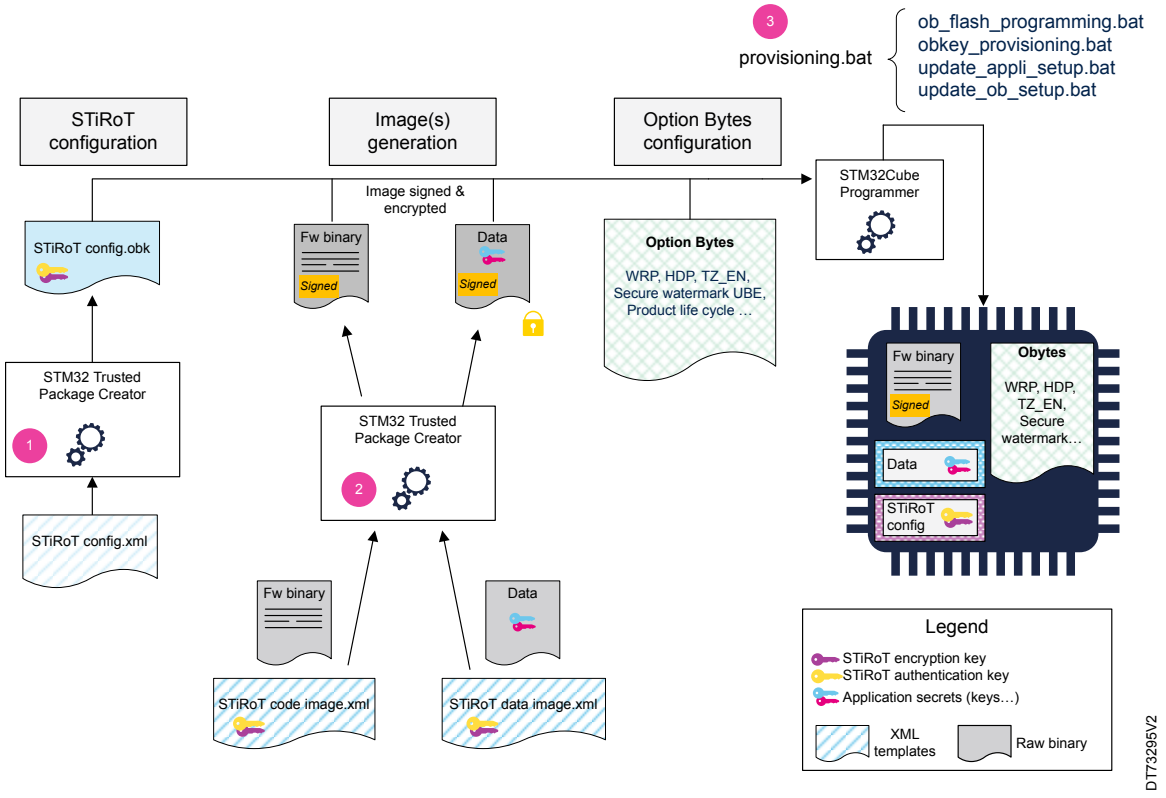
3 Provisioning process

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

1. Configuration of the STiRoT. At this stage, the number of images managed (firmware image only, or firmware and data images), the location of the images, and the cryptographic keys are defined.
2. Generation of the image(s).
3. Programming of the option bytes, OBKeys, and image(s) in the device.

Note: A set of scripts is provided in the STM32CubeH5 FW package (`Firmware/Projects/Board/ROT_Provisioning/STiRoT` folder). It guides the user all along the provisioning process. Refer to [4] for more information.

Figure 6. STiRoT provisioning process



4 OBKeys configuration file

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 input.

STiRoT configuration provides the possibility to:

- Define the user flash memory mapping: select the location and the size of each area.
- Configure the authentication and encryption keys.
- Activate the optional management of a data image.
- Allow the jump in the bootloader (external access) when no valid image is installed.

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 the following:

Table 6. STiRoT Configuration

Parameter	Hidden	Description	Additional controls/constraints
Number of images managed by STiRoT	No	• 1: Firmware image only. Configuration selected for any application which does not manage secret information. • 2: Firmware and data images. As example, in the two boot stages use case, the data image contains the cryptographic keys managed by uRoT application.	Slot definition of data image (size and offset) will be available trough STM32 Trusted Package Creator as soon as two images configuration is selected.
High speed clock activation	Yes	• 0: 64 MHz, initial clock configuration • 1: 200 MHz (default value), best speed clock supporting the full range of temperature • 2: 250 MHz, max speed clock	-
Is the firmware full secure	No	• Yes: The firmware booting is fully secure. Refer to the example `STiRoT_Appli` provided in STM32H5 series Cube. • No: The firmware booting is made of a secure part and a nonsecure part. Refer to the example `STiRoT_Appli_TrustZone` provided in STM32H5 series Cube.	The size of the secure area inside the firmware execution area must be specified as soon as the firmware is not fully secure. In this configuration, the image generation triggered by the post build command of the nonsecure project is preceded by an assembly command of the secure and nonsecure binaries.
Jump into ST bootloader	Yes	• Yes (default value): STiRoT jump into the ST bootloader when no valid firmware image is installed. • No: Jump into the ST bootloader is not allowed.	When a jump is not allowed, a reset in loop is generated when no valid firmware image is installed.
Firmware execution area offset	No	The offset is from the beginning of the user flash memory. It must be aligned on a sector start address.	No overlap is allowed between the execution areas and the download areas. A firmware execution area cannot be mapped between the two download areas.
Firmware download area offset	No	The offset is from the beginning of the user flash memory. It must be aligned on a sector start address.	No overlap is allowed between the execution areas and the download areas. A firmware execution area cannot be mapped between the two download areas. Both firmware and data download areas can overlap. In this case, the installation of a data and firmware image in a single execution using the dependency feature is not possible.

Parameter	Hidden	Description	Additional controls/constraints
Firmware area size	No	Must be a multiple of the sector size.	No overlap is allowed between the execution areas and the download areas. A firmware execution area cannot be mapped between the two download areas.
Data area offset in HDPL2 OBKeys	No	The offset is from the beginning of the HDPL2 OBKeys area (0xFFD0900).	• The range allowed is from 0xFFD0900 to 0xFFD0BF0: there is no overlap with the HDPL3 OBKeys. • This parameter is disabled for the "one image configuration".
Data slot size in HDPL2 OBKeys	No	The maximum is 0x2F0.	• The range allowed is from 0xFFD0900 to 0xFFD0BF0: there is no overlap with HDPL3 OBKeys. • This parameter is disabled for the "one image configuration".
Data download area offset	No	The offset is from the beginning of the user flash memory. It must be aligned on a sector start address.	Both firmware and data download areas can overlap. In this case, the installation of a data and firmware image in a single execution using the dependency feature is not possible.
Data download slot size	Yes	Must be a multiple of the sector size.	Must be 0x2000 for STM32H5 series product (parameter hidden).
Size of the secure area inside the firmware execution area	No	Must be a multiple of the sector size.	This parameter is disabled for the "firmware fully secure" configuration. STiRoT configures the MPU to allow execution only for the secure part of the firmware. Then, it is up to the secure application to reconfigure the MPU depending on its needs.
SRAM2 erasing in case of reset	Yes	• Yes (default value): the SRAM2 is automatically erased by hardware in case of reset. it ensure that all secrets are erased in case of reset. • No: SRAM2 is not erased. The configuration is less secured but it can be required when the application has no other possibility to manage persistent information in SRAM2.	This information is duplicated to ensure that the option bytes are correctly configured.
SRAM2 ECC management activation	Yes	• Yes (default value): a reset is generated in case of ECC detection. ECC is a mechanism to detect memory tampering. • No: ECC is not managed when SRAM tampering detection is not required.	This information is duplicated to ensure that the option bytes are correctly configured.
Product state minimal allowed	No	Open, Provisioning, Provisioned, TZ-Closed, Closed, Locked: STiRoT only execute the firmware if the product state programmed in the option byte is equal or greater than the “product state minimal allowed”.	If the user plans to use partial regression, then they must set the product state minimal to TZ-Closed (at maximum).
Encryption key	No	The key used to encrypt the firmware and data images.	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 used to authenticate the firmware and data images.	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)
Internal tamper 9 and 15 activation.	Yes	Yes (default value): Internal tamper 9 and 15 are activated during STiRoT execution. No: Internal tampers 9 and 15 are deactivated during STiRoT execution.	-

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

5 Images generation

STiRoT manages images at MCUboot format including:

- A header,
- The encrypted firmware or data binary,
- Metadata informations (TLV format: Tags Length Value) allowing the control of the image (SHA256, encryption key…),
- A magic number to trigger the installation.

Further information about the MCUboot open source software is available at [5].

STM32 Trusted Package Creator is provided to generate both firmware and data images. These images are encrypted and signed using the keys configured in the OBKeys configuration file.

Two types of images are generated:

- Initial image: Image in clear to be programmed in installation area during initial device programming,
- Update image: Encrypted image to be programmed in download area. It is installed during secure firmware update process.

5.1 Firmware image generation

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.

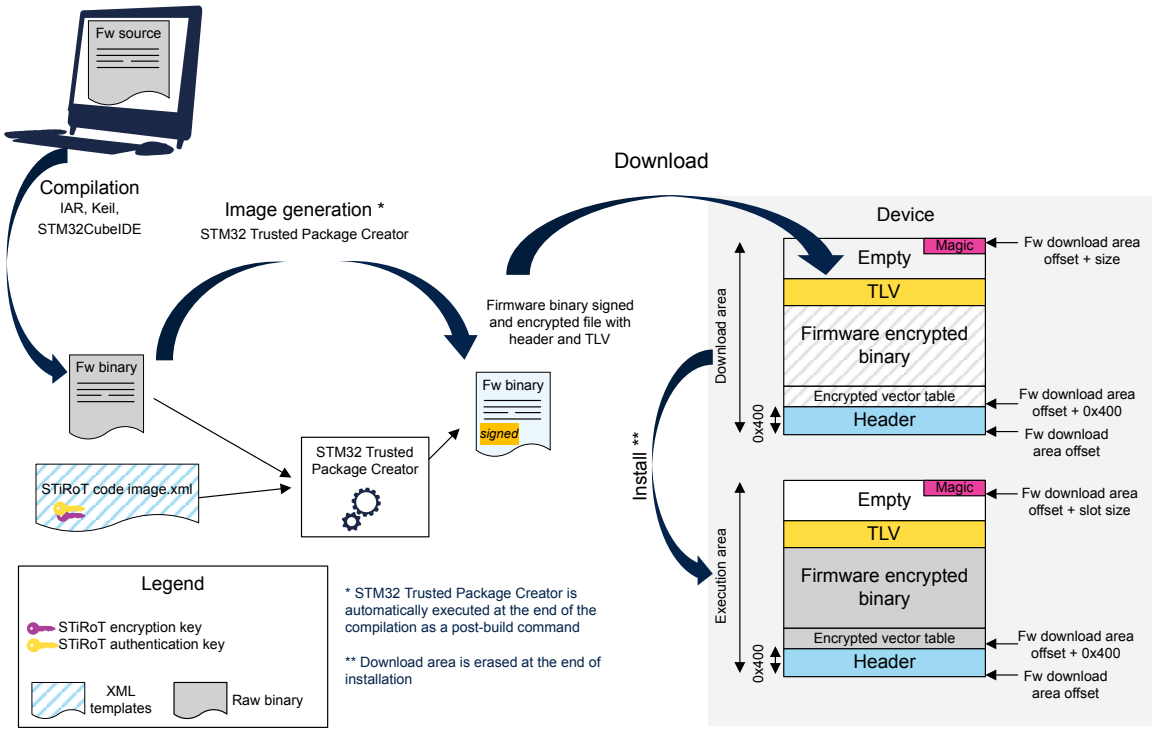
STiRoT_Code_Image.xml can be edited with STM32 Trusted Package Creator in order to modify the version and the dependency information. The keys are directly inherited from the STiRoT_Config.xml file.

Dependency can be enabled when the compatibility with a specific data image version is required: the installation of both images is synchronized in this case.

To generate a clear (non-encrypted) image, the encryption key must be removed from the list of parameters of the xml file.

The following figure shows the firmware image generation:

Figure 7. **Firmware image generation**



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

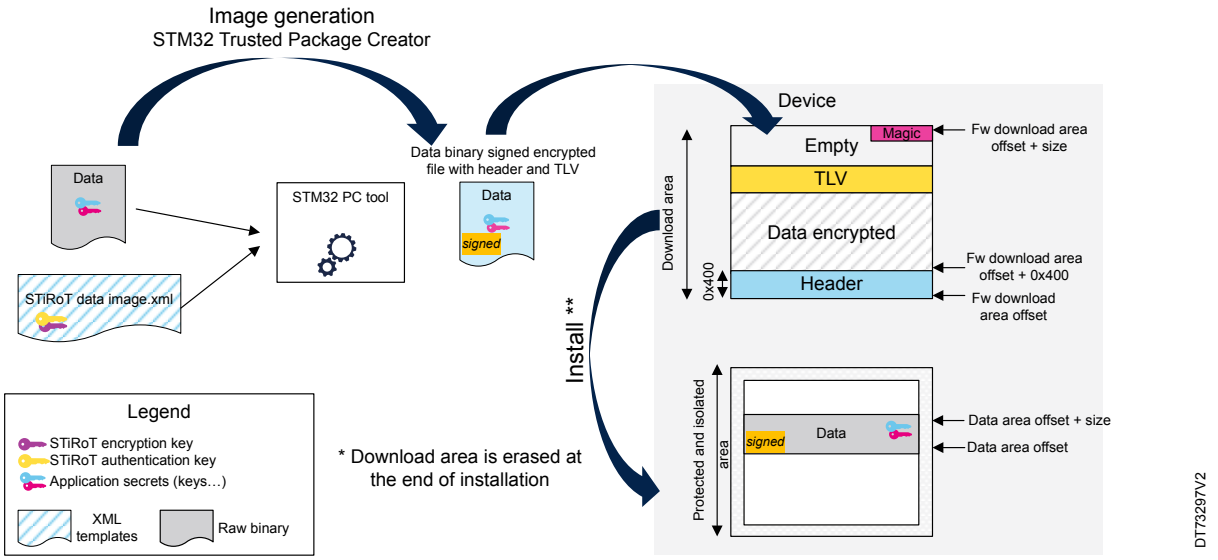
5.2 Data image generation

The STiRoT_Data_Image.xml file contains all the parameters driving the image generation such as the authentication and encryption keys, and the version information. STiRoT_Data_Image.xml can be edited with STM32 Trusted Package Creator in order modify the version and the dependency information. The keys are directly inherited from the STiRoT_Config.xml file.

Dependency can be enabled when the compatibility with a specific firmware image version is required: installation of both images is synchronized in this case

The following figure shows the data image generation:

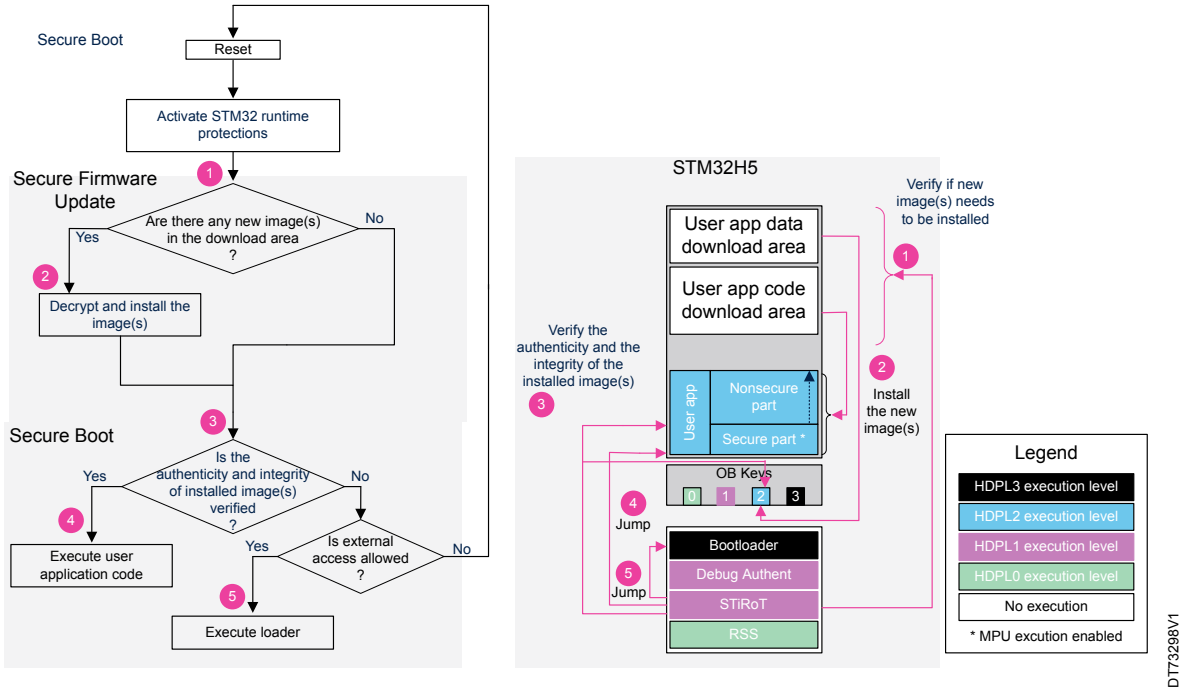
Figure 8. **Data image generation**



The list of all parameters is detailed in Appendix F: Data image XML file.

Appendix A STiRoT start-up sequence

Figure 9. STiRoT start-up sequence

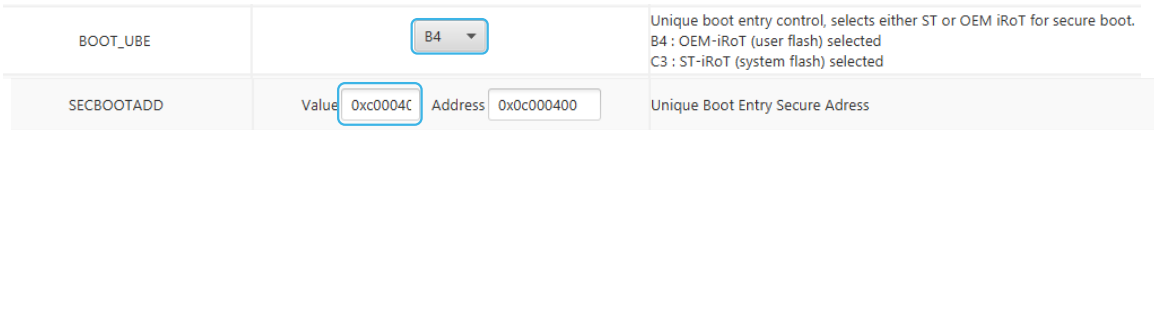


The step 5 is optional and can be disabled through STiRoT configuration. When not allowed, a reset is generated instead of a jump to the loader.

Appendix B Application development phase

The most efficient way to develop and debug an application is to boot directly on the user flash memory in the Open product state. To do this, set the UBE to `0xB4` and the `SECBOOTADD` to `0x0C000400` using STM32CubeProgrammer (STM32CubeProg). By default, the firmware execution offset is `0x0`.

Figure 10. Boot directly in user flash memory



As long as the device is in 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.

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

Appendix C STiRoT Config.obk file format

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

Table 7. STiRoT_Config.obk format

Offset	Size	Type	Description
0	4	uint32_t	sdp command header: destination address
4	4	uint32_t	sdp command header: data size address
8	4	uint32_t	sdp command header: encryption enabled/disabled (1/0)
12	32	array	SHA256 of the following data content (from offset 44 to 268)
44	1	uint8_t	Number of images managed by STiRoT
45	1	uint8_t	Clock selection (64/200/250 MHz) – (0/1/2)
46	1	uint8_t	Firmware fully secure (1/0)
47	1	uint8_t	Jump into ST bootloader enabled/disabled (1/0)
48	4	uint32_t	Firmware execution area offset
52	4	uint32_t	Firmware download area offset
56	4	uint32_t	Firmware area size
60	4	uint32_t	Data area offset in HDPL2 OBKeys
64	4	uint32_t	Data slot size in HDPL2 OBKeys
68	4	uint32_t	Data download area offset
72	4	uint32_t	Data download slot size
76	4	uint32_t	Size of the secure area inside the firmware execution area (meaningful only when the firmware is not fully secure).
80	4	uint32_t	SRAM2 erasing in case of reset
84	4	uint32_t	SRAM2 ECC management activation
88	4	uint32_t	Product state minimal allowed (`0x0000ED00`, Open / `0x00001700`, Provisioning / `0x00002E00`, Provisioned / `0x0000C600`, TZ-Closed / `0x00007200`, Closed / `0x00005C00`, Locked
92	70	array	Encryption private key (MCUboot format)
162	2	array	Reserved
164	91	array	Authentication public key (MCUboot format)
255	13	array	Reserved (global alignment of OBKeys data on 16 bytes)

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).

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

Table 8. STiRoT_Data.obk format

Offset	Size	Type	Description
0	4	uint32_t	sdp command header : destination address
4	4	uint32_t	sdp command header : data size (number of bytes)
8	4	uint32_t	sdp command header : encryption enabled/disabled (1/0)
12	32	array	SHA256 of the following data content (from offset 44 to 108)
44	4	uint32_t	Version of the code image, automatically updated during the secure boot and secure firmware update process.
48	4	uint32_t	Version of the previous code image, automatically updated during the secure boot and secure firmware update process.
52	4	uint32_t	Version of the data image, automatically updated during the secure boot and secure firmware update process.
56	4	uint32_t	Version of the previous data image, automatically updated during the secure boot and secure firmware update process.
60	32	array	SHA256 of the code image installed, automatically updated during the secure boot and secure firmware update process.
92	16	array	Reserved

Appendix E Code image XML file

Table 9. 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. This parameter can be disabled (adding `<enable>` property to 0) in order to generate a nonencrypted image
Image magic	No	Magic value 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	Yes	Dependency with a specific data image identified with its version. This parameter can be removed by setting the `<enable>` property to 1.
Write option	No	STiRoT implement the overwrite installation method.
Version	Yes	x.y.z firmware image version.
Security counter	No	Security counter value automatically generated based on version information.
Align	No	16-bytes alignment
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 `0x8000000`.
Firmware binary input file	Automatic	Location of the firmware binary file. This parameter is updated during the provisioning process based on info 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 Data image XML file

Table 10. Data image generation

Parameter	Updatable	Description
Authentication key	Automatic	Private key inherited from the STiRoT_Config.xml file.
Encryption key	Automatic	Public key inherited from the STiRoT_Config.xml file. This parameter can be disabled (adding `<enable>` property to 0) in order to generate a nonencrypted image
Image magic	No	Magic value identifying a data 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.
Data download slot size	No	Data download slot size information inherited from the STiRoT_Config.xml file.
Header Size	No	`0x400` bytes.
Padding Header	No	Padding the header with `0xFF` to fulfill the `0x400` bytes.
Dependency with firmware image	Yes	Dependency with a specific firmware image identified with its version. This parameter can be removed by setting the `<enable>` property to 1.
Write option	No	STiRoT implement the overwrite installation method.
Version	Yes	x.y.z data image version.
Security counter	No	Security counter value automatically generated based on version information.
Align	No	16-bytes alignment.
Data 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 `0x8000000`.
Data binary input file	Automatic	Location of the data 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 G STiRoT execution status

STiRoT is executed as a black box, but there is a way to verify the steps that have been executed and in case of an issue where the STiRoT execution has stopped.

For each HDPL OBKeys, there are four bytes reserved for the status log. This information can be extracted using the debug Authentication discovery command (refer to [3] for more information).

Table 11. Execution status

Executed step	Status coding	Description
STIROT_INIT_DONE	0x00000001UL	Device initialization done. STiRoT bootpath selected. STiRoT executed.
STIROT_CONFIG_DONE	0x00000002UL	STiRoT configuration according to STiRoT_Config.obk is verified to be correct in OBKeys HDPL1.
STIROT_SECURITY_DONE	0x00000004UL	Security activated. OBKeys STiRoT configuration verified to be correct.
STIROT_SLOT_PRIMARY_1 _VALID	0x00000010UL	Authenticity and integrity verified for the user application firmware image in the execution slot.
STIROT_SLOT_PRIMARY_2 _VALID	0x00000020UL	Authenticity and integrity verified for the data image in the HDPL2 OBKeys secure storage.
STIROT_SLOT_SECONDARY_1_ VALID	0x00000040UL	Authenticity and integrity verified for the user application firmware image in the download slot.
STIROT_SLOT_SECONDARY_2_ VALID	0x00000080UL	Authenticity and integrity verified for the data image in the download slot.
STIROT_PARSING_DONE	0x00000100UL	Slots parsing done.
STIROT_INSTALLATION_DONE	0x00000200UL	Image(s) in the download slot (one or two images depending on config) are installed.
STIROT_VALIDATION_DONE	0x00000400UL	Execution slots analysis completed to verify which image is valid (integrity and authenticity).
STIROT_JUMP_BL	0x00001000UL	At least one image is not valid -> bootloader execution.
STIROT_JUMP_APPLI	0x00002000UL	All images (one or two depending on config) are valid -> application execution.
STIROT_NOJUMP	0x00004000UL	At least one image is not valid, but bootloader execution is not allowed (option defined in STiRoT_Config.xml).

Appendix H Troubleshooting

Table 12. Troubleshooting

Problem	Possible solution
Provisioning errors in ob_flash_programming.bat script	STM32CubeProgrammer is still connected to the device.
Provisioning errors	Device is not in OPEN product state: regression.bat script must be executed.

Revision history

Table 13. **Document revision history**

Date	Revision	Changes
21-Sep-2023	1	Initial release.
19-Mar-2024	2	Terminology updated. Appendix added: Appendix G: Troubleshooting
20-Mar-2024	3	Topic updated: Section 1: General information
08-Jan-2025	4	Section 2.4: STiRoT constraints added.
03-Mar-2025	5	Appendix D: STiRoT Data.obk file format added.
13-Jun-2025	6	Updated: • Section 2.4: STiRoT constraints • Figure 6. STiRoT provisioning process • Section 4: OBKeys configuration file • Section 5: Images generation • Figure 7. Firmware image generation • Figure 8. Data image generation

Contents

List of tables

List of figures