Getting started with STiRoT (ST immutable Root of Trust) for STM32 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® STM32 microcontrollers, with its associated tools ecosystem. How to use it, step by step, is described at Section 1: General information.

Table 1. Applicable products

Type	Products
Microcontroller	STM32G564CB, STM32G564MB, STM32G564HE, STM32G564CE, STM32G564CC, STM32G564MC, STM32G564ME, STM32G564PB, STM32G564PC, STM32G564PE, STM32G564QB, STM32G564QC, STM32G564QE, STM32G564QEU, STM32G564RB, STM32G564RC, STM32G564RE, STM32G564REU, STM32G564VB, STM32G564VC, STM32G564VE, STM32V873IJ, STM32V873J-DK, STM32V873RJ, STM32V873XJ, STM32V873VJ, STM32V873ZJ, STM32V873ZJ

1 General information

This document applies to the STM32G5 series Arm® Cortex®-M33-based microcontrollers and STM32V8 series Arm® Cortex®-M85-based microcontrollers.

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

Arm and TrustZone are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

arm

Table 2. Terms and abbreviations

Acronym	Definition
DA	Debug authentication
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
WRP	Write protection

Table 3. List of documents

Reference	Name/address	Title
[1]	AN6008	*Introduction to Debug Authentication (DA) for STM32 MCUs* application note
[2]	RM0499	*STM32V86xxx/87xxx Arm®-based 32-bit MCUs* reference manual
[3]	RM0514	*STM32G56x/55x/54x/53x series Arm®-based 32 MCUs* reference manual
[4]	Security	https://wiki.st.com/stm32mcu/wiki/Category:Security

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

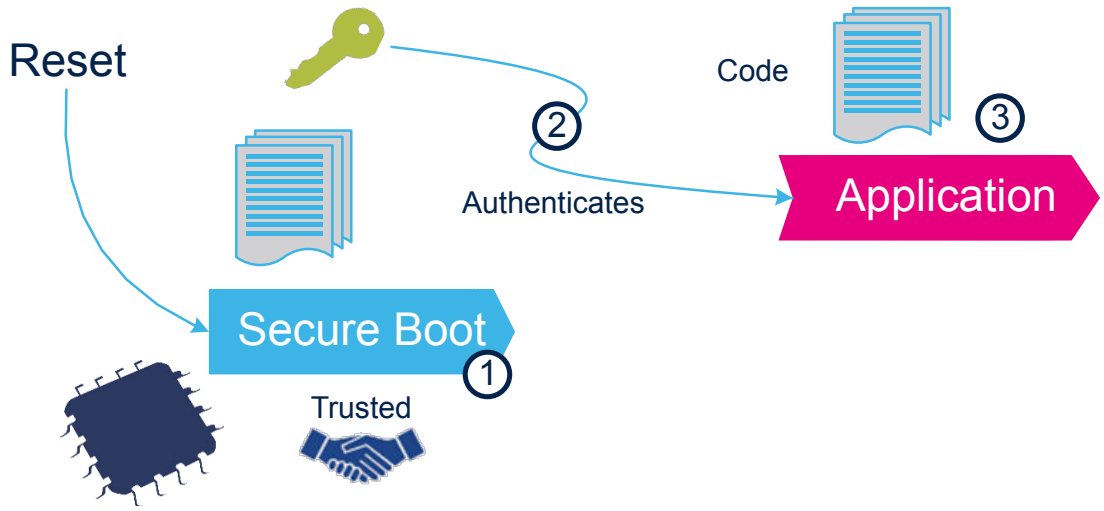
2.1 Overview

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 STM32 microcontrollers and provides two services:

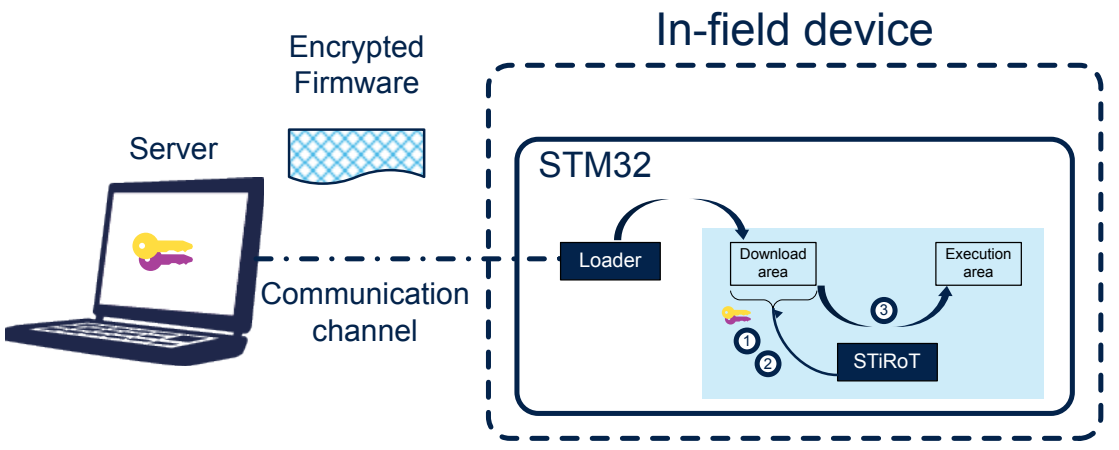
- The Secure Boot (root of trust services) is an immutable code, which is always executed after a system reset (1). It activates runtime protections and then, it verifies the authenticity and integrity of the application (2) code before every execution (3).

Figure 1. Secure Boot



- The Secure Firmware Update application is an immutable code that detects that a new firmware image is available. It 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 startup 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

STiRoT activation is done by:

1. Configuring the `BOOT_UBE` option byte (refer to [3] or [2] for more details) 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 STM32 microcontrollers manages directly the user application.
- **Two boot stages:** The STiRoT integrated inside the STM32 microcontrollers 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. STM32G5 boot time performance (data from STM32G564)

Configuration	Boot time		
-	64 Kbytes	128 Kbytes	256 Kbytes
-	After system reset (1)		
Execution at 64 MHz	20.3 ms	33.6 ms	60.5 ms
Execution at 200 MHz	7.5 ms	11.8 ms	20.4 ms
Execution at 250 MHz	6.3 ms	9.7 ms	16.6 ms
-	Wake-up from standby mode(2)		
Execution at 64 MHz	12.5 ms	12.5 ms	12.5 ms
Execution at 200 MHz	5 ms	5 ms	5 ms
Execution at 250 MHz	4.3 ms	4.3 ms	4.3 ms

1. *STiRoT authenticates firmware using a reference hash method.*
2. *STiRoT using Fast wake-up does not authenticate firmware out of standby mode*

Table 6. STM32V8 boot time performance (data from STM32V873)

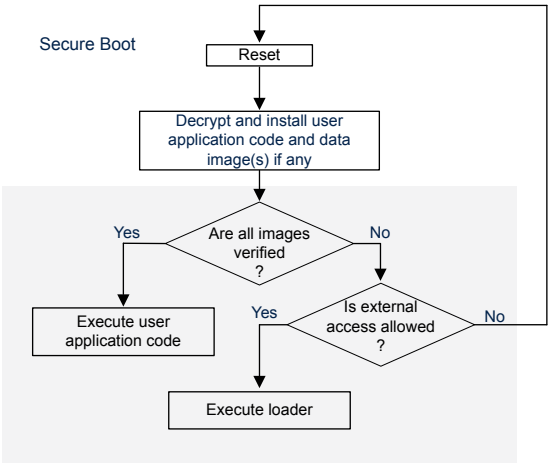
Configuration	Boot time		
Firmware area size	256 Kbytes	512 Kbytes	1 Mbyte/s
-	After system reset (1)		
Execution at 256MHz	17 ms	29 ms	52 ms
Execution at 450MHz	11.5 ms	18 ms	31 ms
Execution at 800MHz	8 ms	12 ms	19 ms
-	Wake-up from standby mode(2)		
Execution at 256MHz	4 ms	4 ms	4 ms
Execution at 450MHz	3.8 ms	3.8 ms	3.8 ms
Execution at 800MHz	3.6 ms	3.6 ms	3.6 ms

1. *STiRoT authenticates firmware using a reference hash method.*
2. *STiRoT using Fast wake-up does not authenticate firmware out of standby mode*

2.3.1 One boot stage

At reset, STiRoT is executed in temporal isolation 1 (HDPL1). Refer to [2] or [3] 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

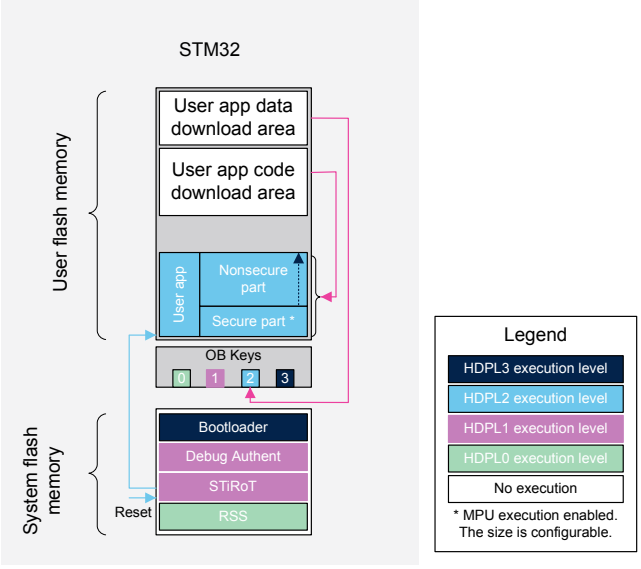
The user application can be fully secure or divided in a secure and nonsecure application. In both cases, the image stored in the download area is a single image. The secure part of the user application must contain at least the vector table and the reset handler. If this is not the case, the application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



DT81814V1

An example of one boot stage application is provided with STM32Cube MCU Package for STM32G5 series and STM32V8 series .

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

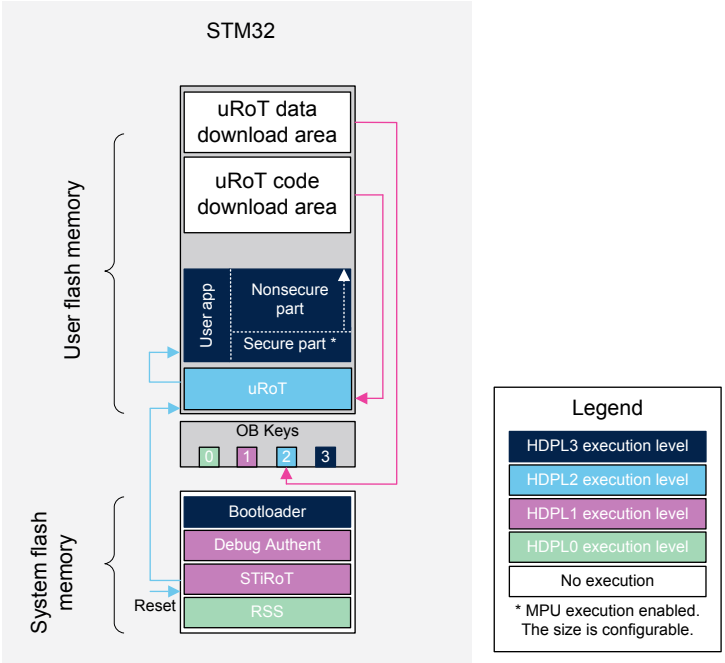
2.3.2 Two boot stages

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] or [3] or 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. STiRoT - User application executed after uRoT



An example of a two boot stages application is provided with the STM32Cube MCU Package for STM32G5 series and STM32V8 series .

2.4 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 a 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.
- STiRoT does not support a potential mode for the tampers. It resets in loop until power-off if a tamper event in potential mode occurs during its execution (or during Standby mode or VBAT mode).
- When a tamper event occurs during STiRoT execution (or Standby mode or VBAT mode), then STiRoT saves the content of the TAMP status register, and write a magic value in a specific location (see hardware diversity annex). The application can check the presence of this magic value and stored status register in this specific location to identify which tamper events occurred, then decides the appropriated action and clears the magic value.
- Internal tamper 9 (fault generation for cryptographic peripherals (SAES, PKA, AES, RNG) and internal tamper 15 (system fault detection) are activated by default during STiRoT execution and remains activated when jumping into the user application. The usage of these tampers in STiRoT can be deactivated through STiRoT OBKeys configuration (see Section 4: OBKeys configuration file).
- When the fast wake-up is allowed through STiRoT configuration parameters, the STiRoT relies on the hardware SBF (standby flash) to skip firmware image verification. The SBF flag remains set when the application is entered after a wake-up from standby mode. This allows tha application to handle its own context restoration. To maintain a secure execution environment, the user application must clear the SBF flasg after it is processed.
- 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.
- One SRAM (see hardware diversity annex) is used by STiRoT and is therefore erased at each reset.
- SWAP_BANK activation by option byte is not supported by STiRoT.

Table 7. STM32G564xx and STM32V873xx specificities

Description	STM32G564xx	STM32V873xx
Tamper magic number (if a tamper fault occurs, it equals 0x003981BB)	TAMP->BKP30R	4 bytes at 0x3C00 1FF8
Tamper status register copy (indicates which tampers have occurred)	TAMP->BKP31R	4 bytes at 0x3C00 1FFC
SRAM used during SFSP execution	SRAM2	SRAM6_FIX
SRAM erasure in case of reset	Always during STiRoT execution	Always during STiRoT execution
SRAM ECC management	Activation through STiRoT configuration	Activation through STiRoT configuration
Clock configuration	0: 64 MHz 1: 200 MHz 2: 250 MHz	0: 256 MHz 1: 450 MHz 2: 800 MHz

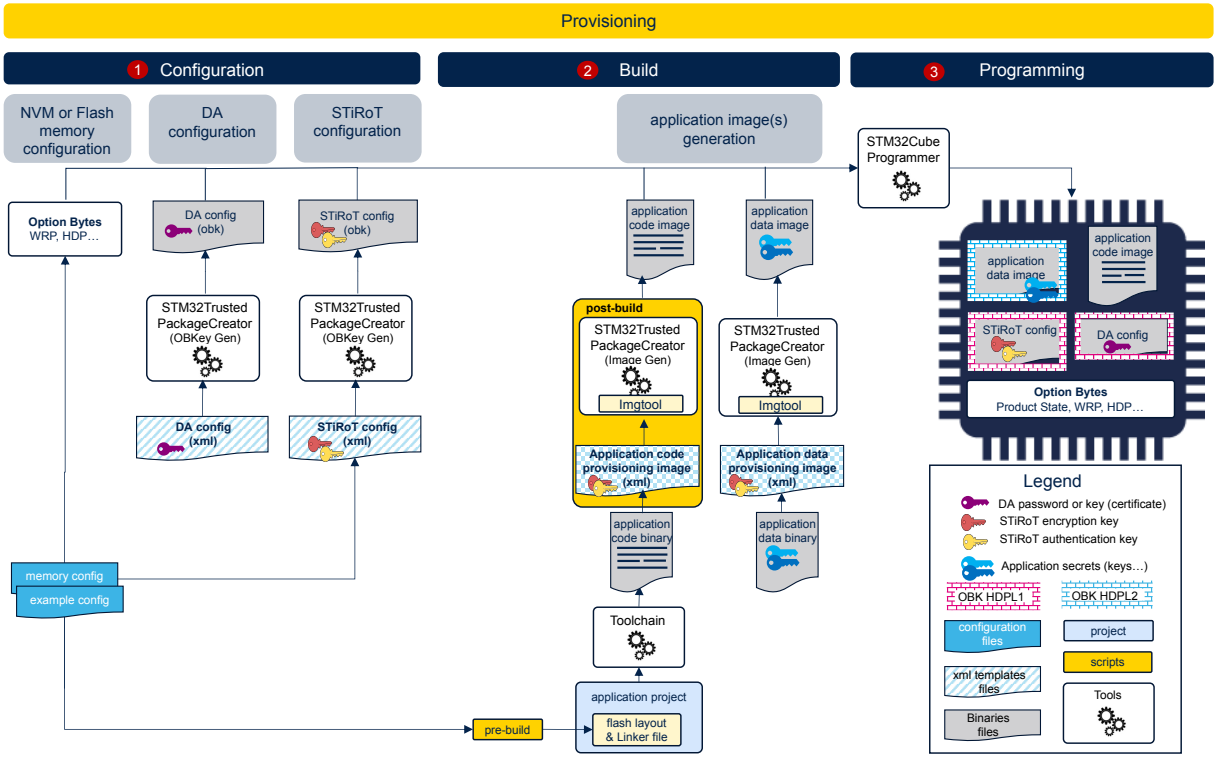
3 Provisioning process

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

1. Configuration of the STiRoT. At this stage, the flash layout, managed images (firmware only, or firmware and data), and associated authentication and encryption keys are configured. The configuration also defines whether bootloader access through an external jump is allowed when no valid image is installed, and whether ECC, internal tamper detection, and fast wake-up are enabled.
2. Generation of one or more images.
3. Programming of the option bytes, OBKeys, and images in the device.

Note: A script is provided in the STM32CubeG5 MCU package (`STM32CubeG5_Installation_path\examples\advanced\rot\stirot\NUCLEO-G564RE\provisioning` folder) and in the STM32CubeV8 MCU package (`STM32CubeV8_Installation_path\examples\advanced\rot\stirot\NUCLEO-V873XJ\provisioning` folder). It guides the user all along the provisioning process. Refer to Section 1: General information 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 hidden in STM32 Trusted Package Creator because scripts update them automatically. Configure user-customizable parameters in the `.ini` file or in `stirot_config.xml` for more advanced parameters.

The list of all the parameters is the following:

Table 8. STiRoT configuration

Parameter	Hidden	Description	Additional controls / constraints
Number of images managed by STiRoT	Yes	• 1: Firmware image only. Configuration selected for any application that does not manage secret information. • 2: Firmware and data images. For example, in the two boot stages use case, the data image contains the cryptographic keys managed by the uRoT application.	This parameter is inherited from the memory INI file.
System clock selection (see hardware diversity annex)	No	• 0: initial clock configuration • 1: The best speed clock supporting the full range of temperature • 2: max speed clock For details on supported frequencies, see hardware diversity annex.	-
Is the application full secure?	Yes	Yes: the firmware booted is fully secure. No: the firmware booted is made of a secure part and a nonsecure part.	-
Jump into ST bootloader when no valid image	No	0: Jump not allowed. There is no access to user flash memory provided by STiRoT. 1: Jump allowed when no valid image	Firmware must be programmed during the provisioning stage and the download function must be implemented and controlled by the firmware.
Application image installation slot offset	Yes	Offset from begin of user flash memory. It must be a multiple of the sector size	No overlap is allowed between images and between execution and download areas. This parameter is inherited from the memory INI file.
Application image download slot offset	Yes	Offset from begin of user flash memory. It must be a multiple of the sector size.	No overlap is allowed between images and between execution and download areas. This parameter is inherited from the memory INI file.
Application image slot size	Yes	Size of the firmware. It must be a multiple of the sector size.	No overlap is allowed between images and between execution and download areas.

Parameter	Hidden	Description	Additional controls / constraints
			Firmware slots definition must not exceed the user flash capacity. This parameter is inherited from the memory INI file.
Data image installation slot offset in HDPL2 OBKeys	Yes	The offset is from the beginning of the HDPL2 OBKeys area.	The range allowed is from the start of HDPL2 OBKeys area till the end - 16 bytes (the last 16 bytes are reserved for debug information): there is no overlap with the HDPL3 OBKeys. This parameter is disabled for the “one image configuration”. This parameter is inherited from the memory INI file.
Data image installation slot size in HDPL2 OBKeys	Yes	Size of the data area in HDPL2 OBKeys	From the start of HDPL2 OBKeys area till end - 16 bytes (the last 16 bytes are reserved for debug information) Parameter disabled for the “1 image” configuration. This parameter is inherited from the memory INI file.
Data image download slot offset	Yes	Offset from begin of user flash memory. It must be a multiple of the sector size.	No overlap is allowed between images and between execution and download areas. This parameter is inherited from the memory INI file.
Data image download slot size	Yes	It must be a multiple of the sector size.	This parameter is inherited from the memory INI file.
Size of the secure area inside the application image installation slot	Yes	Size of the secure part of the application image. It must be a multiple of the sector size.	In no isolation case, this size should be equal to the firmware area size
SRAM ECC management activation (see hardware diversity annex)	No	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 no hardware attack is possible.	-
Product state minimal allowed	Yes	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”.	This parameter is inherited from the provisioning.ini file
Encryption key	No	The key is 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 (`appli_code_fwu_image.xml` `appli_code_provisioning_image.xml` `appli_data_fwu_image.xml` `appli_data_provisioning_image.xml`)

Parameter	Hidden	Description	Additional controls / constraints
Authentication key	No	The key is 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 (`appli_code_fwu_image.xml` `appli_code_provisioning_image.xml` `appli_data_fwu_image.xml` `appli_data_provisioning_image.xml`)
Activate internal tampers	No	Yes (default value): Activate the system (iTamper15) and crypto (iTamper9) internal tampers. No: tampers are not activated	-
Enable fast wake-up	No	Yes (default value): Enable fast wake-up from standby mode: bypass images control. No: do not bypass images control	-
Output file	No	Name of the output file	File name

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 information (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:

- Provisioning image: Image in clear to be programmed in the execution area during initial device programming,
- FWU image: Encrypted image to be programmed in the download area. It is installed during the secure firmware update process.

5.1 Provisioning application image generation

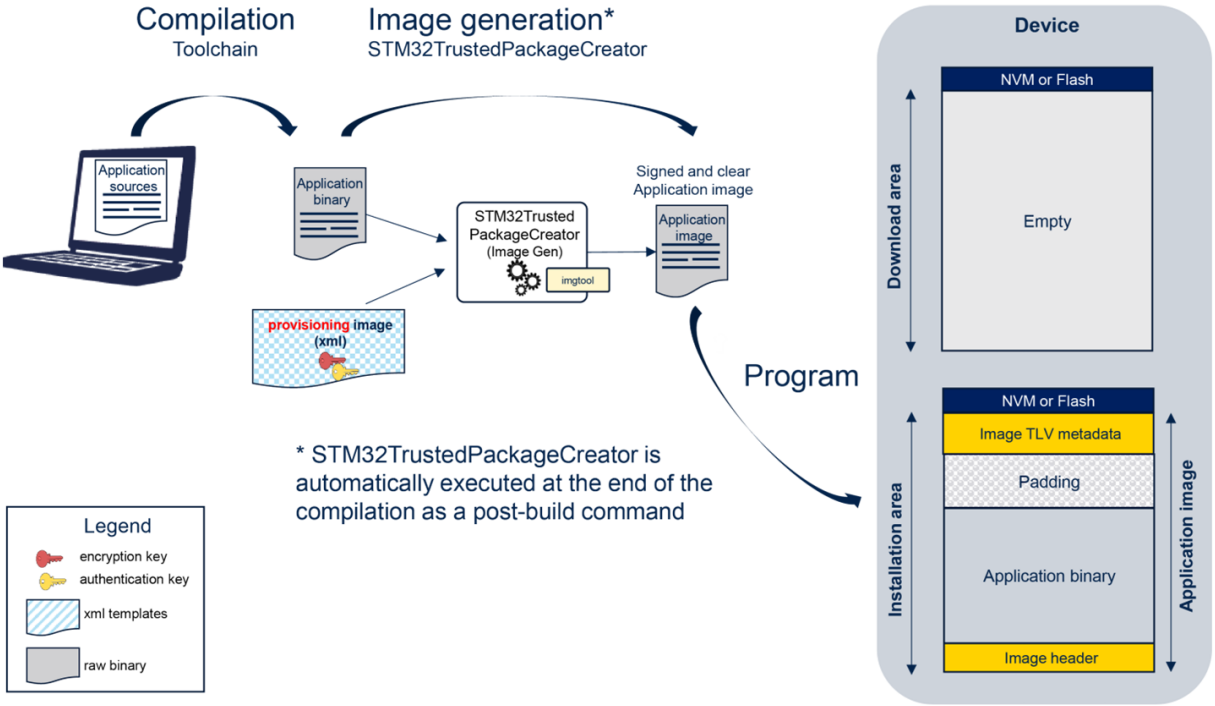
`appli_code_provisioning_image.xml`contains all the parameters driving the image generation such as:

- The authentication keys.
- The version.

`appli_code_provisioning_image.xml` can be edited with STM32 Trusted Package Creator to modify the version information. The keys are directly inherited from the `stirot_config.xml file`.

The following figure shows the provisioning application image generation:

Figure 7. Provisioning application image generation



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

5.2 FWU application image generation

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

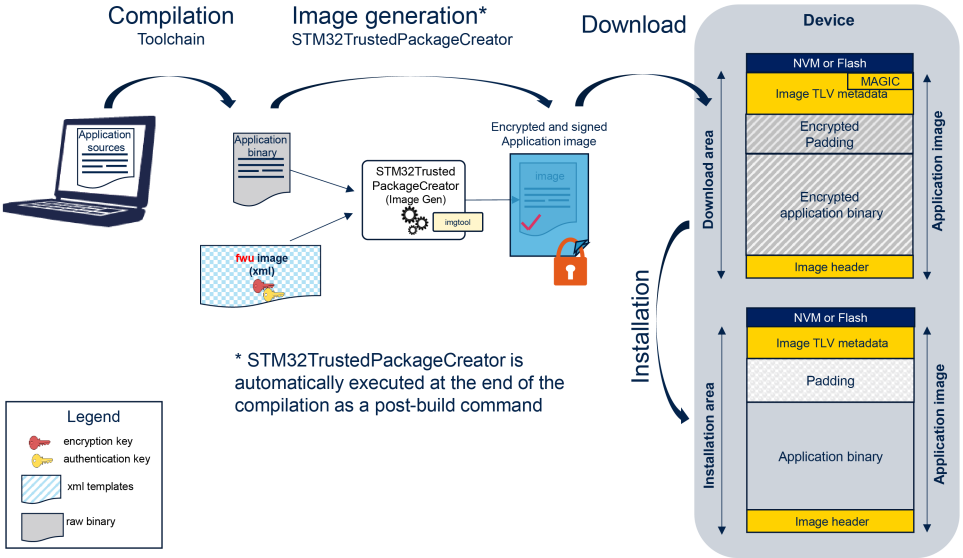
- The authentication and encryption keys.
- The version and the dependency information.

`appli_code_fwu_image.xml` can be edited with STM32 Trusted Package Creator 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.

The following figure shows the FWU application image generation:

Figure 8. FWU application image generation



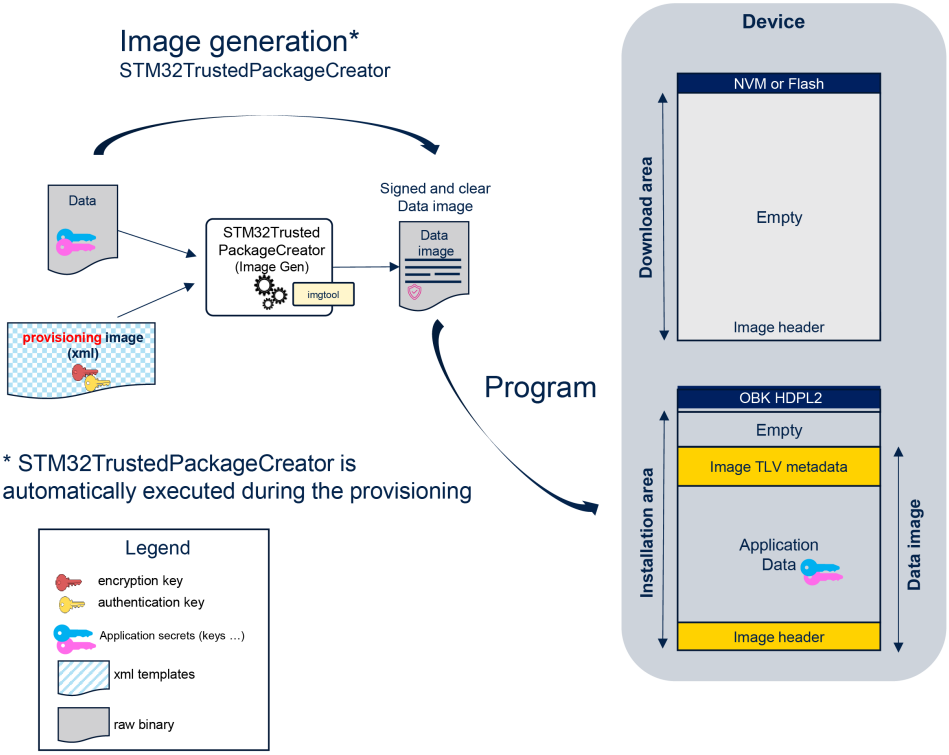
The list of all parameters is detailed in the appendix Appendix F: FWU application image XML file.

5.3 Provisioning application data generation

The `appli_data_provisioning_image.xml` file contains all the parameters driving the image generation such as the authentication and the version information. `appli_data_provisioning_image.xml` can be edited with STM32 Trusted Package Creator to modify the version information. The keys are directly inherited from the `stirot_config.xml` file.

The following figure shows the provisioning application data generation:

Figure 9. Provisioning application data generation



The list of all parameters is detailed in Appendix G: Provisioning application data XML file.

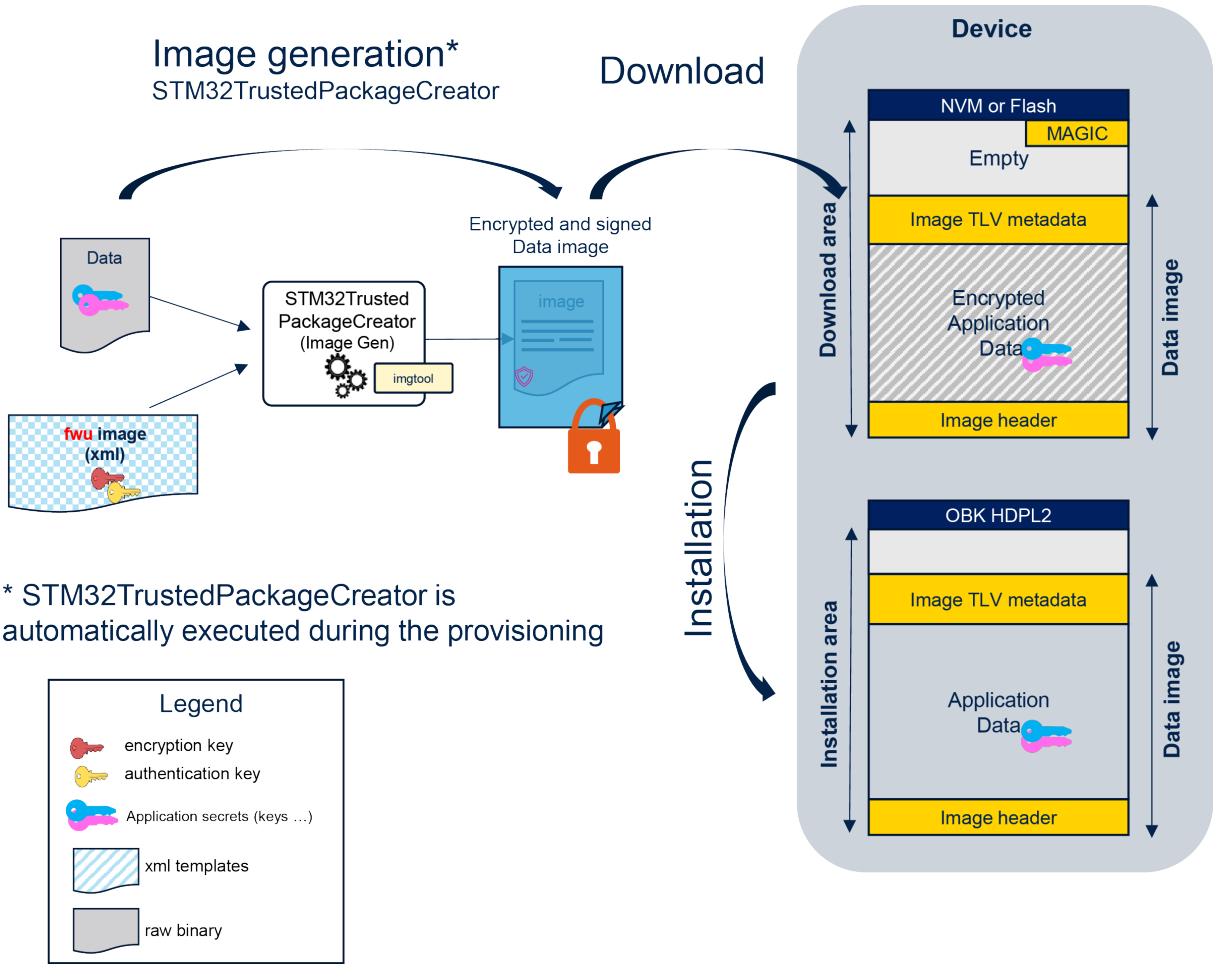
5.4 FWU application data generation

The `appli_data_fwu_image.xml` file contains all the parameters driving the image generation such as the authentication and encryption keys, and the version information. `appli_data_fwu_image.xml` can be edited with STM32 Trusted Package Creator 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 firmware image version is required: installation of both images is synchronized in this case

The following figure shows the FWU application data generation:

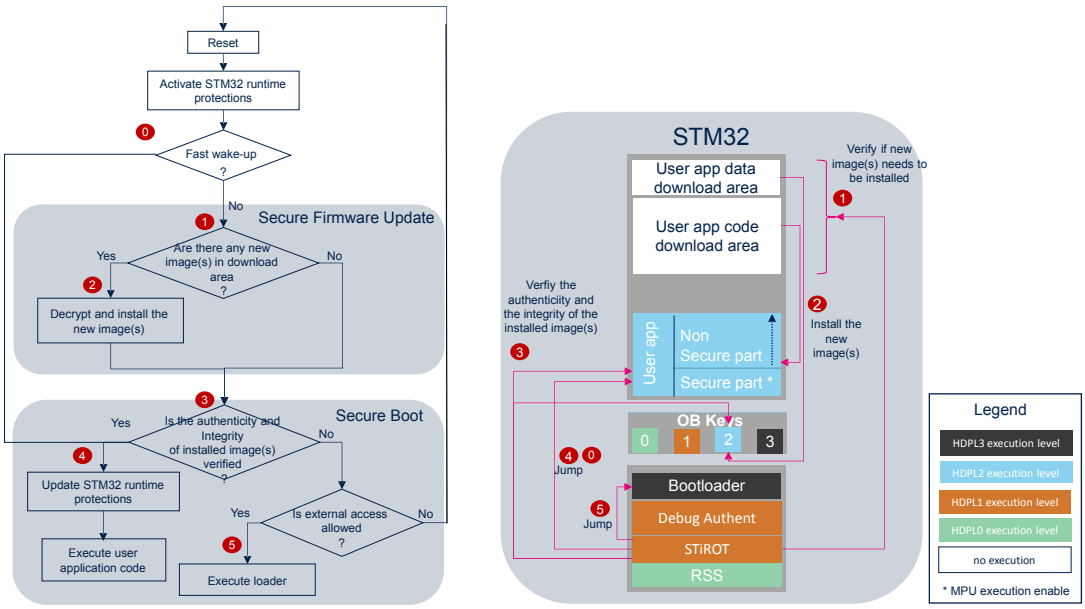
Figure 10. FWU application data generation



The list of all parameters is detailed in Appendix H: FWU application data XML file.

Appendix A STiRoT startup sequence

Figure 11. **STiRoT startup sequence**



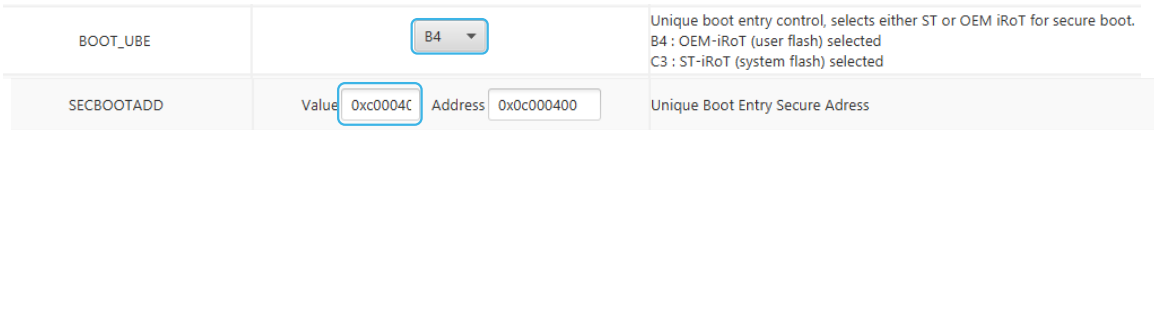
The step 0 is optional and can be disabled through STiRoT configuration. When the fast wake-up is allowed, firmware image control is bypassed on wake-up to accelerate resume time. Software countermeasures ensure the same level of security of the secure boot.

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 user flash start address + 0x400 using STM32CubeProgrammer (STM32CubeProg). By default, the firmware execution offset is `0x0`.

Figure 12. 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 I: 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 9. 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 – (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	Reserved
84	4	uint32_t	SRAM ECC management activation (see hardware diversity annex)
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	1	uint8_t	Internal tampers activation
256	1	uint8_t	Fast wakeup activation
257	11	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 10. 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 Provisioning application image XML file

Table 11. Provisioning application image XML file

Parameter	Updatable	Description
Authentication key	Automatic	Private key used to sign the image. This parameter is inherited from the stirot_config.xml file.
Encryption key	Automatic	Public key used for image encryption. This parameter is inherited from the stirot_config.xml file.
Symmetric key length	No	AES-CTR symmetric key (depends on crypto scheme)
Public key format	No	Authentication public key fully provided in image metadata (must match SHA of provisioned key).
Image magic	No	Magic value (32 bits) at start of image header
No encryption (clear)	No	The image is clear in the installation slot but includes encryption metadata for reencryption if swapped to the download slot.
Endianness	No	Little endian
Application image slot size	Automatic	Application image slot size, inherited from stirot_config.xml file
Header size	No	Size of image header, 0x400 bytes.
Padding header	No	Padding the header with 0xFF to fulfill the header size.
Version	Yes	Firmware image version in x.y.z format (x ≤ 255, y ≤ 255, z ≤ 65535)
Security version	Yes	Value used for antirollback. Can be 'auto' (aligned to image version) or manually set (32-bit). Note that the antirollback feature is inactive if the security version remains unchanged at each update of the image.
Installation mode	No	STiRoT implement the overwrite installation method. The maximum allowed firmware image size is the firmware area size.
Align	No	Used to align the swap info / copy done / image OK flags in the trailer.
Firmware download area offset	Automatic	Used to generate.hex binary file format with destination address. This parameter is inherited from the stirot_config.xml file
Firmware binary input file	Automatic	Location of the firmware binary file This parameter is updated during the provisioning process based on info from the project INI file
Image output file	Automatic	Location of the firmware image generated. This parameter is updated during the provisioning process based on info from the project INI file

Appendix F FWU application image XML file

Table 12. FWU application image XML file

Parameter	Updatable	Description
Authentication key	Automatic	Private key used to sign the image. This parameter is inherited from the stirot_config.xml file.
Encryption key	Automatic	Public key used for image encryption. This parameter is inherited from the stirot_config.xml file.
Symmetric key length	No	AES-CTR symmetric key (depends on crypto scheme)
Public key format	No	Authentication public key fully provided in image metadata (must match SHA of provisioned key).
Image magic	No	Magic value (32 bits) at start of image header
Endianness	No	Little endian
Application image slot size	Automatic	Application image slot size, inherited from stirot_config.xml file
Header size	No	Size of image header, 0x400 bytes.
Padding header	No	Padding the header with 0xFF to fulfill the header size.
Version	Yes	Firmware image version in x.y.z format (x ≤ 255, y ≤ 255, z ≤ 65535)
Security version	Yes	Value used for antirollback. Can be 'auto' (aligned to image version) or manually set (32-bit). Note that the antirollback feature is inactive if the security version remains unchanged at each update of the image.
Dependency with data image	Yes	Index and version of other image required for synchronized installation. Format is 'Index, version': • Index: '2' for application data image • Version: x.y.z (x ≤ 255, y ≤ 255, z ≤ 65535). As long as the dependency condition is not satisfied, the candidate image remains pending in the download slot. This parameter can be removed by setting the <enable> property to 0.
Installation mode	No	STiRoT implement the overwrite installation method. The maximum allowed firmware image size is the firmware area size.
Align	No	Used to align the swap info / copy done / image OK flags in the trailer.
Firmware binary input file	Automatic	Location of the firmware binary file This parameter is updated during the provisioning process based on info from the project INI file
Image output file	Automatic	Location of the firmware image generated. This parameter is updated during the provisioning process based on info from the project INI file

Appendix G Provisioning application data XML file

Table 13. Provisioning application data XML file

Parameter	Updatable	Description
Image magic	No	Magic value (32 bits) at start of image header
Endianness	No	Little endian
Data image installation slot size in HDPL2 OBKeys	Automatic	Application data slot size, inherited from stirot_config.xml file
Header size	No	Size of image header, 0x20 bytes.
Padding header	No	Padding the header with 0xFF to fulfill the header size.
Version	Yes	Data image version in x.y.z format (x ≤ 255, y ≤ 255, z ≤ 65535)
Security version	Yes	Value used for antirollback. Can be 'auto' (aligned to image version) or manually set (32-bit). Note that the antirollback feature is inactive if the security version remains unchanged at each update of the image.
Installation mode	No	STiRoT implement the overwrite installation method. The maximum allowed data image size is the data area size.
Align	No	Used to align the swap info / copy done / image OK flags in the trailer.
Data binary input file	Yes	Location of the data binary file
Image output file	Yes	Location of the data image generated.

Appendix H FWU application data XML file

Table 14. FWU application data XML file

Parameter	Updatable	Description
Authentication key	Automatic	Private key used to sign the image. This parameter is inherited from the stirot_config.xml file.
Encryption key	Automatic	Public key used for image encryption. This parameter is inherited from the stirot_config.xml file.
Symmetric key length	No	AES-CTR symmetric key (depends on crypto scheme)
Public key format	No	Authentication public key fully provided in image metadata (must match SHA of provisioned key).
Image magic	No	Magic value (32 bits) at start of image header
Endianness	No	Little endian
Data image download slot size	Automatic	Application data slot size, inherited from stirot_config.xml file
Header size	No	Size of image header, 0x20 bytes.
Padding header	No	Padding the header with 0xFF to fulfill the header size.
Version	Yes	Data image version in x.y.z format (x ≤ 255, y ≤ 255, z ≤ 65535)
Security version	Yes	Value used for antirollback. Can be 'auto' (aligned to image version) or manually set (32-bit). Note that the antirollback feature is inactive if the security version remains unchanged at each update of the image.
Dependency with firmware image	Yes	Index and version of other image required for synchronized installation. Format is 'Index, version': • Index: '0' for application code image • Version: x.y.z (x ≤ 255, y ≤ 255, z ≤ 65535). As long as the dependency condition is not satisfied, the candidate image remains pending in the download slot. This parameter can be removed by setting the <enable> property to 0.
Installation mode	No	STiRoT implement the overwrite installation method. The maximum allowed data image size is the data area size.
Align	No	Used to align the swap info/copy done / image OK flags in the trailer.
Data binary input file	Yes	Location of the data binary file
Image output file	Yes	Location of the data image generated.

Appendix I 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 [1] for more information).

Table 15. 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 J Troubleshooting

Table 16. **Troubleshooting**

Problem	Possible solution
Provisioning errors	Device is not in OPEN product state: python provisioning.py da --regression must be executed.

Revision history

Table 17. **Document revision history**

Date	Revision	Changes
02-Nov-2026	1	Initial release.

Contents

1 General information 2
2 STiRoT presentation 3
2.1 Overview 3
2.2 Protection measures and security strategy 4
2.3 STiRoT activation 4
2.3.1 One boot stage 5
2.3.2 Two boot stages 7
2.4 STiRoT constraints 8
3 Provisioning process 9
4 OBKeys configuration file 10
5 Images generation 13
5.1 Provisioning application image generation 13
5.2 FWU application image generation 14
5.3 Provisioning application data generation 15
5.4 FWU application data generation 16
Appendix A STiRoT startup sequence 17
Appendix B Application development phase 18
Appendix C STiRoT Config.obk file format 19
Appendix D STiRoT Data.obk file format 20
Appendix E Provisioning application image XML file 21
Appendix F FWU application image XML file 22
Appendix G Provisioning application data XML file 23
Appendix H FWU application data XML file 24
Appendix I STiRoT execution status 25
Appendix J Troubleshooting 26
Revision history 27
List of tables 29
List of figures 30

List of tables

Table 1. Applicable products . . . 1
Table 2. Terms and abbreviations . . . 2
Table 3. List of documents . . . 2
Table 4. External references . . . 2
Table 5. STM32G5 boot time performance (data from STM32G564) . . . 5
Table 6. STM32V8 boot time performance (data from STM32V873) . . . 5
Table 7. STM32G564xx and STM32V873xx specificities . . . 8
Table 8. STiRoT configuration . . . 10
Table 9. STiRoT_Config.obk format . . . 19
Table 10. STiRoT_Data.obk format . . . 20
Table 11. Provisioning application image XML file . . . 21
Table 12. FWU application image XML file . . . 22
Table 13. Provisioning application data XML file . . . 23
Table 14. FWU application data XML file . . . 24
Table 15. Execution status . . . 25
Table 16. Troubleshooting . . . 26
Table 17. Document revision history . . . 27

List of figures

Figure 1. Secure Boot . . . 3
Figure 2. Secure Firmware Update . . . 3
Figure 3. STiRoT behavior in case of verification failure . . . 6
Figure 4. STiRoT - User application executed after STiRoT . . . 6
Figure 5. STiRoT - User application executed after uRoT . . . 7
Figure 6. STiRoT provisioning process . . . 9
Figure 7. Provisioning application image generation . . . 13
Figure 8. FWU application image generation . . . 14
Figure 9. Provisioning application data generation . . . 15
Figure 10. FWU application data generation . . . 16
Figure 11. STiRoT startup sequence . . . 17
Figure 12. Boot directly in user flash memory . . . 18

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.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The 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.

The 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 the 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.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

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.

© 2026 STMicroelectronics – All rights reserved