Introduction to secure firmware install (SFI) for STM32 MCUs

Introduction

This application note supports the secure firmware install (SFI) feature available on the STM32 MCUs listed in Table 1 .

Outsourcing of product manufacturing enables original equipment manufacturers (OEMs) to reduce their direct costs and concentrate on high added-value activities, such as research and development, sales and marketing.

However, contract manufacturing puts the OEM's proprietary assets at risk, and since the contract manufacturer (CM) manipulates the OEM's intellectual property (IP), it might be disclosed to other customers, or appropriated.

To meet the new market security requests and protect customers against any leakage of their IPs, STMicroelectronics introduces a new security concept, secure firmware install (SFI), permitting programming of OEM firmware into STM32 MCU internal flash memory in a secure way (with confidentiality, authentication and integrity checks).

Moreover, ST provides the STM32TRUSTEE that eases the OEM to secure its platform. STM32TRUSTEE involves adding STM32TRUSTEE-SM in SFI. STM32TRUSTEE is only available on STM32H5 series. In this document, each time there is a reference to STM32TRUSTEE, it applies only on STM32H5.

STM32 series devices support protection mechanisms that permit the protection of critical operations (such as cryptography algorithms) and critical data (such as secret keys) against unexpected access.

This application note gives an overview of the STM32 SFI solution with its associated tools ecosystem, and explains how to use it to protect OEM firmware during the CM product manufacturing stage.

Table 1. Applicable products

Type	Part numbers / lines / series	Referred as
Microcontrollers	Entire STM32H523/533, STM32H562, STM32H563/573 lines	STM32H5
	STM32H730xx(1), STM32H733xx(1), STM32H735xx(1)	STM32H7
	STM32H75xxx(2)	
	STM32H7B0xx(3), STM32H7B3xx(3)	
	Entire STM32H7R3/7S3, STM32H7R7/7S7 lines	STM32H7RS
	Entire STM32L5 series.	STM32L5
	Entire STM32U375/385 lines.	STM32U3
	Entire STM32U5 series.	STM32U5
	STM32WBA52(4), STM32WBA54/55(4) lines	STM32WBA5
	STM32WBA62(5), STM32WBA63(5), STM32WBA64/65(5) lines	STM32WBA6
	Entire STM32WL5x line.	STM32WL5

1. *Supported on specific order codes, see [ES0491] .*
2. *All order codes supported (refer to datasheet ordering information section).*
3. *Supported on specific order codes, see [ES0478] .*
4. *SFI only available on STM32WBA5xxG devices with 128-Kbyte SRAM, Rev B.*
5. *SFI only available on STM32WBA6xxI devices with 512-Kbyte SRAM.*

1 Preamble

1.1 Related documents

Refer to the following documents available from www.st.com (unless an NDA applies):

[AN3155]	USART protocol used in the STM32 bootloader
[AN3154]	CAN protocol used in the STM32 bootloader
[AN3156]	USB DFU protocol used in the STM32 bootloader
[AN4221]	I2C protocol used in the STM32 bootloader
[AN4286]	SPI protocol used in the STM32 bootloader
[AN5054]	Secure programming using STM32CubeProgrammer
[AN5927]	I3C protocol used in the STM32 bootloader
[ES0478]	STM32H7A3xI/G, STM32H7B0xB and STM32H7B3xI device errata
[ES0491]	STM32H72xx/73xx device errata
[RM0399]	STM32H745/755 and STM32H747/757 advanced Arm®-based 32-bit MCUs
[RM0433]	STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®based 32-bit MCUs
[RM0438]	STM32L552xx and STM32L562xx advanced Arm®-based 32-bit MCUs
[RM0453]	STM32WL5x advanced Arm®-based 32-bit MCUs with sub-GHz radio solution
[RM0455]	STM32H7A3/7B3 advanced Arm®-based 32-bit MCUs
[RM0456]	STM32U5 series Arm®-based 32-bit MCUs
[RM0468]	STM32H723/733, STM32H725/735 advanced Arm®-based 32-bit MCUs
[RM0477]	STM32H7Rx/Sx Arm®-based 32-bit MCUs
[RM0481]	STM32H563/H573 and STM32H562 Arm®-based 32-bit MCUs
[RM0487]	STM32U3 series Arm®-based 32-bit MCUs
[RM0493]	Multiprotocol wireless Bluetooth® Low-Energy and IEEE802.15.4, STM32WBA5xxx Arm®-based 32-bit MCUs
[RM0515]	STM32WBA6xxx advanced Arm®-based 32 MCUs
[UM2237]	STM32CubeProgrammer software description
[UM2238]	STM32 Trusted Package Creator tool software description
[WikiSFI]	https://wiki.st.com/stm32mcu/wiki/Security:SFI_overview
[WikiSTM32H5]	https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:SFI_Step-by-step_on_STM32H5

Note: Programming tool manufacturers who want to support SFI/SMI/SSP solutions from STMicroelectronics and integrate them into their production line equipment should contact ST sales office for additional information under NDA.

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

1.2 Glossary

Table 2. **Glossary terms**

Term	Definition
AES	Advanced encryption standard
AES GCM	AES Galois counter mode
CM	Contract manufacturer
FT	Flash memory programming tool
HDP	Hide data protection
HDPL	HDP level
ST HSM	STMicroelectronics hardware security module
MAC	Message authentication code
MCU	Microcontroller unit
OB	Option bytes
OBK	Option bytes key
OCTOSPI	Octo-SPI interface
OEM	Original equipment manufacturer
OTFDEC	On-the-fly decryption engine
RDP	Readout protection
Secure boot	Root of trust, check STM32 security protection
Secure bootloader	Standard ST bootloader with additional security features
SFI	Secure firmware install
SFIx	SFI on external flash memory
STM32 TPC	STM32 Trusted Package Creator (see [UM2238])
STM32TRUSTEE-SM	STM32TRUSTEE Secure Manager
TZEN	TrustZone® enable option byte
user flash memory	Flash memory embedded within STM32 microcontrollers (internal flash memory)
WRP	Write protection

2 STM32 secure firmware install (SFI)

2.1 SFI principles overview

SFI is a secure mechanism implemented in STM32 microcontrollers that allows secure and counted installation of OEM firmware and STM32TRUSTEE-SM in untrusted production environment (such as OEM contract manufacturer). SFI is implemented in a secure bootloader.

The SFI process prevents the OEM firmware and STM32TRUSTEE-SM code from:

- Being accessed by the contract manufacturer.
- Being extracted or disclosed.

This mechanism consists in having the following content encrypted with an AES secret key, thanks to the STM32 Trusted Package Creator tool(1) , during OEM firmware development in the OEM trusted office:

- The whole OEM firmware
- STM32TRUSTEE-SM
- OBK (in case of STM32H5 and STM32H7RS)
- The option bytes.

OEM must use STM32 Trusted Package Creator tool to program ST HSM with its own AES secret key(2) , its own nonce, and a maximum installation counter.

In order to properly protect the firmware confidentiality, the AES secret key must be different every time the OEM encrypts a new firmware or a new release of the same firmware. As a consequence, ST HSM must be changed.

Then, the OEM contract manufacturer have to use STM32CubeProgrammer to initiate SFI process and send encrypted SFI image(3) to STM32 device.

A hardware security module (ST HSM) is in charge of:

- Securely storing OEM AES secret key
- Checking STM32 device certificate(4) that is used to authenticate STM32 device(5)
- Generating and providing the license(6) to the secure bootloader to securely install the encrypted firmware on STM32 device.
- Counting number of produced STM32 devices.

The applicable STM32 microcontrollers are provisioned by STMicroelectronics with device dedicated public/private keys (unique key pair per device). The device keys can be accessed only through the embedded secure bootloader that retrieve AES secret key(7) by decrypting license using device private key.

Thanks to STM32 security features and cryptographic algorithm, the STM32 supports the programming in internal flash memory of:

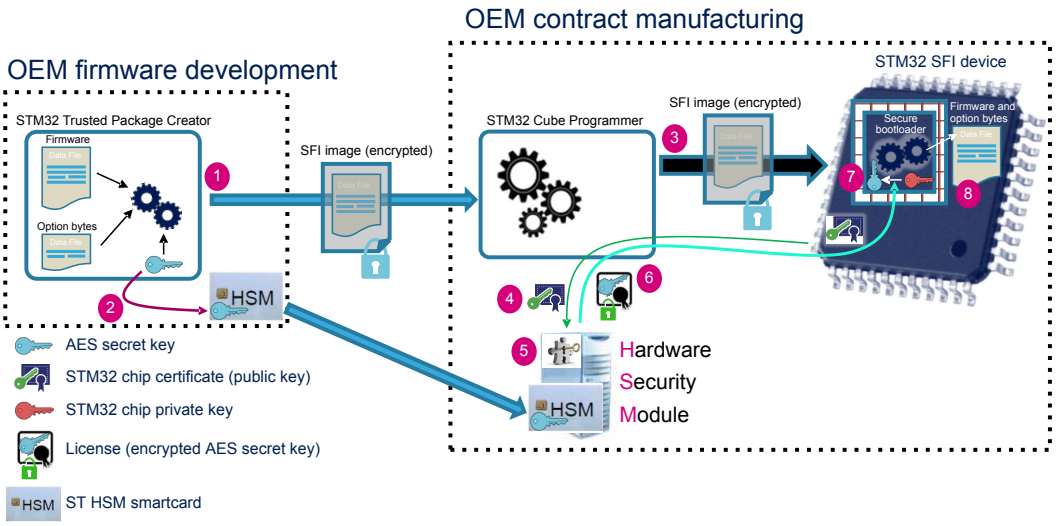
- Secure OEM firmware
- STM32TRUSTEE-SM
- OBK (in case of STM32H5 and STM32H7RS)

and ensures OEM firmware protection (confidentiality, authenticity and integrity) during OEM CM manufacturing stage. The secure firmware install solution securely receives and decrypts the firmware, OBK (in case of STM32H5 and STM32H7RS), and option bytes inside STM32 internal flash memory(8) and optionally external flash memory. Section 2.1.1 focuses on the way SFI process securely installs firmware and data within the internal flash memory, whereas Section 2.1.2 focuses on the way SFI process securely installs firmware and data within the external flash memory.

More information is available in [WikiSFI].

2.1.1 SFI and internal flash memory

Figure 1. SFI process overview



1. *SFI image (encrypted) available from STM32 Trusted Package Creator.*
2. *Program ST HSM with AES secret key.*
3. *SFI process launch.*
4. *Device certificate.*
5. *STM32 device authentication.*
6. *Provide license.*
7. *Retrieve AES secret key.*
8. *OEM firmware, STM32TRUSTEE-SM, OBK (in case of STM32H5 and STM32H7RS), and option bytes programming.*

The secure bootloader is a standard ST bootloader with additional security features.

If the STM32 microcontroller is reset during retrieving AES secret key (7) , all sensitive data are erased before restarting initial SFI procedure.

During SFI process, the secure bootloader never allows any other code to access user flash memory or SRAM.

2.1.2 SFI and external flash memory

This section applies to STM32 products with OTFDEC:

- STM32L562xx
- STM32U585xx/STM32U5Axxx/STM32U5Gxxx
- STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx
- STM32H573xx/STM32H533xx

When speaking of external flash memory, it matters to clearly identify the firmware and data that reside in external flash memory from the firmware and data that reside in user flash memory. Firmware and data in user flash memory are named hereafter internal firmware and data.

The firmware and data that reside in external flash memory are referenced as external firmware and external data throughout the next sections.

The internal firmware must enable the read/fetch of data/code within external flash memory, using OTFDEC and OCTOSPI peripherals.

Note: SFI cannot handle internal flash memory in a first sequence and external in a separate independent one: when SFI handles external firmware and data, it must first handle internal firmware and data that in turn enable decryption at runtime of external firmware and data.

External firmware and data on-the-fly decryption is handled by the OTFDEC peripheral. This peripheral can encrypt and decrypt on-the-fly external firmware and data stored in external flash memory connected to STM32 microcontrollers through the OCTOSPI interface. The OTFDEC can handle up to 4 regions of external flash memory, each one with its own dedicated Key. The OTFDEC uses standard AES CTR 128-bit algorithm for encryption and decryption operations. Refer to the OTFDEC section of the STM32 microcontroller reference manual to get more insight.

OEM can ensure the confidentiality of external firmware and data within external flash memory through 3 different use cases that are depicted within next sections.

2.1.2.1 *External flash memory encryption without secure bootloader*

The cryptographic engine responsible for the on-the-fly external flash memory decryption (OTFDEC) supports AES standard cryptographic algorithm. Thanks to this standard algorithm, OEM can encrypt external firmware and data on host before programming external flash memory, without using secure bootloader.

If external flash memory programming is done within a non-trusted facility, OEM must encrypt the external firmware and data before sending them to the non-trusted facility.

OEM internal firmware must handle the configuration of the OTFDEC peripheral with the AES key for external flash memory decryption. OEM must implement this part within the secure internal firmware in order to guaranty the confidentiality of the external flash memory AES encryption keys.

OEM internal firmware must also handle external flash memory drivers (through OCTOSPI) in order to get access to the external firmware and data.

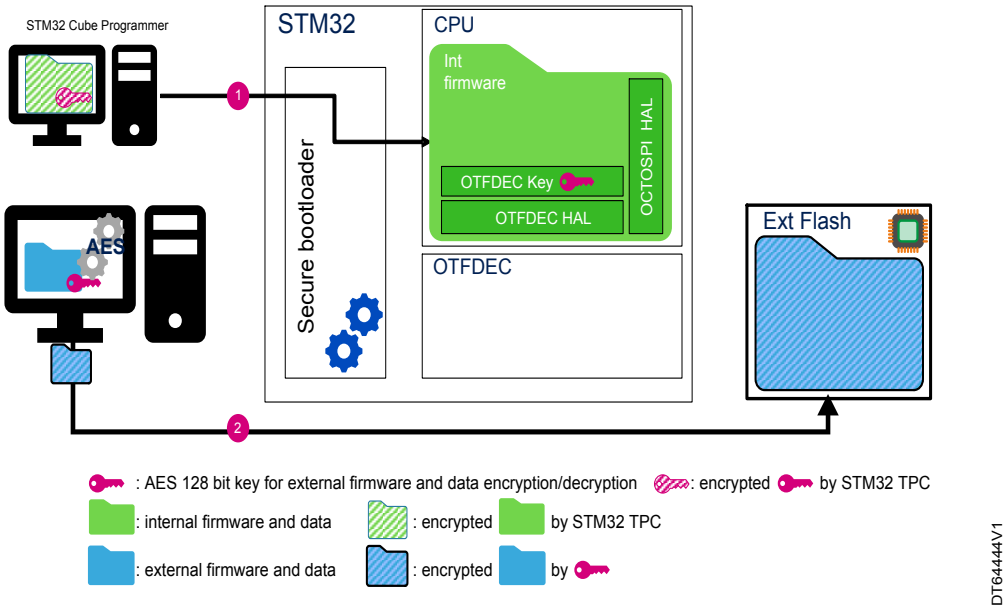
Since OEM programs external flash memory on host, OEM must not encrypt external firmware and data thanks to STM32 Trusted Package Creator. However, OEM must send the external firmware and data AES encryption keys within the SFI image. Then, when building the SFI image thanks to STM32 Trusted Package Creator, OEM must create the SFI image with at least:

- Internal firmware and data (include external flash memory drivers).
- External firmware and data AES key.

Figure 2 below shortly depicts SFI of an internal firmware that manages external firmware and data. The sequence part that addresses internal flash memory is the same than the one depicted in section Section 2.1.1: SFI and internal flash memory.

Figure 2 shows that the secure programming of internal flash memory(1) and the encryption plus programming of external firmware and data(2) are done in two separated flows within SFI. The first flow uses secure bootloader, the second one uses host for programming respectively internal flash memory and external flash memory.

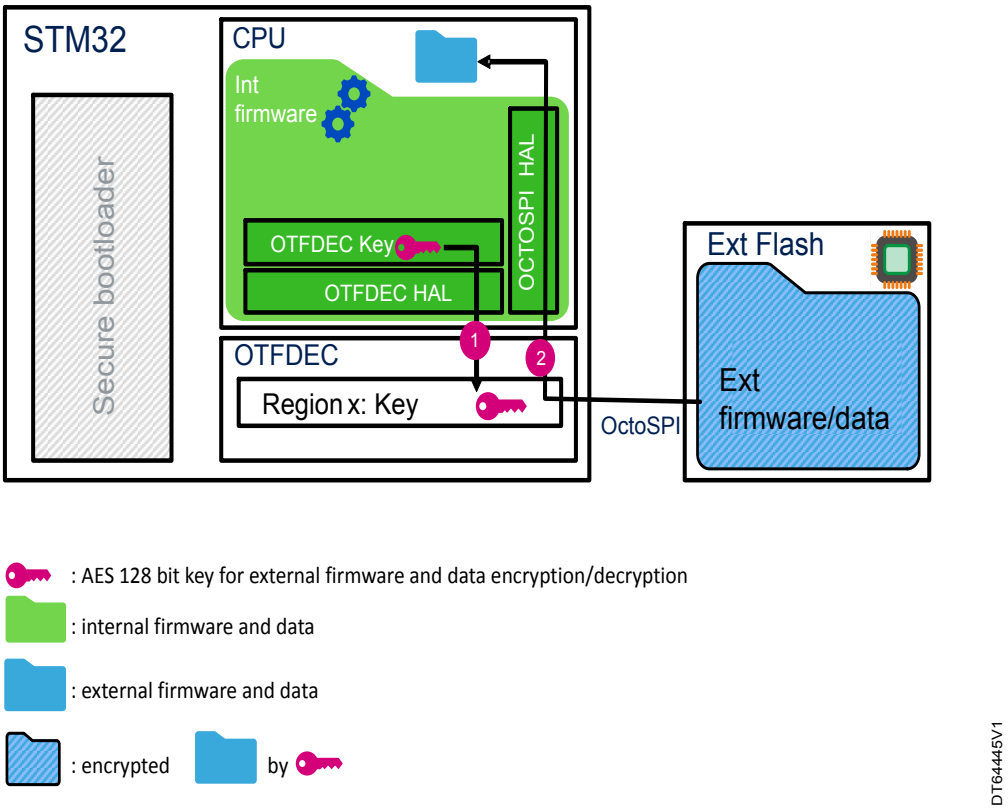
Figure 2. SFI and external flash memory encryption without secure bootloader



1. *Secure programming of internal flash memory.*
2. *Encryption plus programming of external firmware and data.*

Figure 3 below depicts the execution of the same internal firmware in order to manage external firmware and data decryption.

Figure 3. Internal firmware and external flash memory handling



1. *AES external firmware and data key programming in OTFDEC peripheral.*
2. *On-the-fly external flash memory decryption.*

At runtime, during secure boot, the secure internal firmware first copies the AES firmware and data key within the OTFDEC peripheral and activates the OTFDEC region tied to this key(1). Then the CPU can seamlessly read/fetch data/code from external flash memory once the OCTOSPI driver has been initialized(2).

2.1.2.2 *External flash memory encryption with secure bootloader and global key*

In the next two sections, this document focuses on SFI use cases with encryption of the external firmware and data by the secure bootloader. The STM32 receives encrypted external firmware and data, decrypts them with the SFI OEM key, and re-encrypts them with an external flash memory AES key common to all devices to be programmed or with a unique external flash memory AES key per device. This section focuses on the first scenario: a key common to all devices.

The STM32 secure bootloader uses the OTFDEC peripheral to encrypt external firmware and data, the STM32 secure bootloader stores the encryption result within SRAM. Then an external flash memory loader takes the previous result and program it within the external flash memory.

The external flash memory programming is under the responsibility of OEM, meaning that OEM must use an external flash memory loader that fits its external flash memory specificities. ST does not provide such external flash memory loader, except for demonstration purpose on ST boards supporting external flash memory.

Consequently, OEM must implement in internal flash memory the firmware parts dedicated to external flash memory handling (external flash memory keys and drivers).

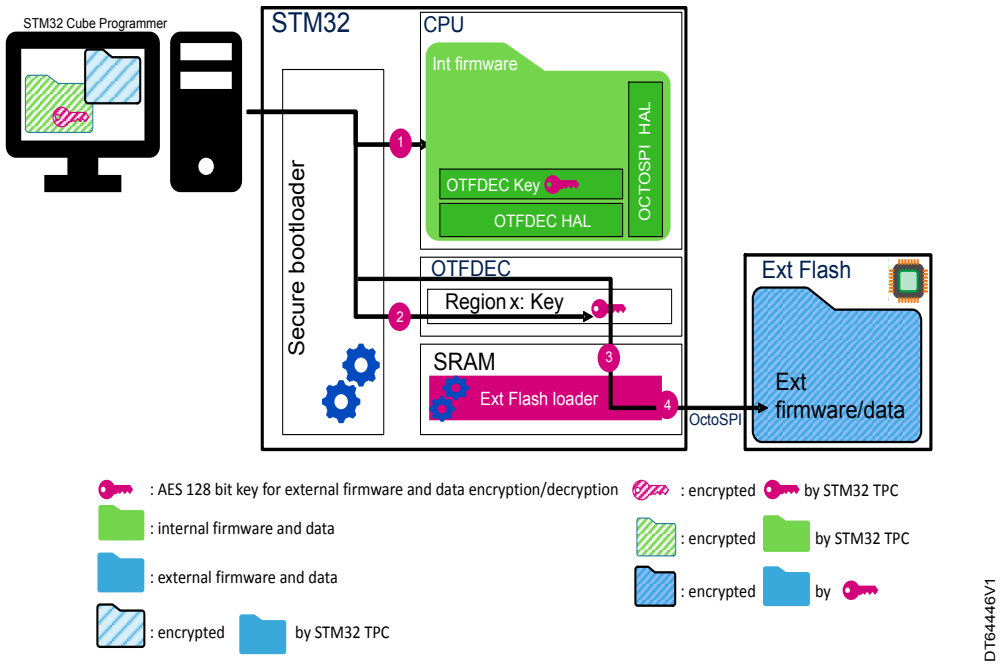
OEM must create SFI image with:

- Internal firmware and data (include external flash memory drivers).
- External firmware and data AES key.
- External firmware and data.

OEM must take care to specify, in STM32 TPC, the same address in internal flash memory for external firmware and data AES key than the one its internal firmware uses for external flash memory on-the-fly decryption with OTFDEC peripheral.

The picture below shortly depicts an SFI sequence where STM32 secure bootloader handles both internal firmware installation and external firmware installation with the help of external flash memory loader.

Figure 4. External flash memory encryption with secure bootloader and global AES Key

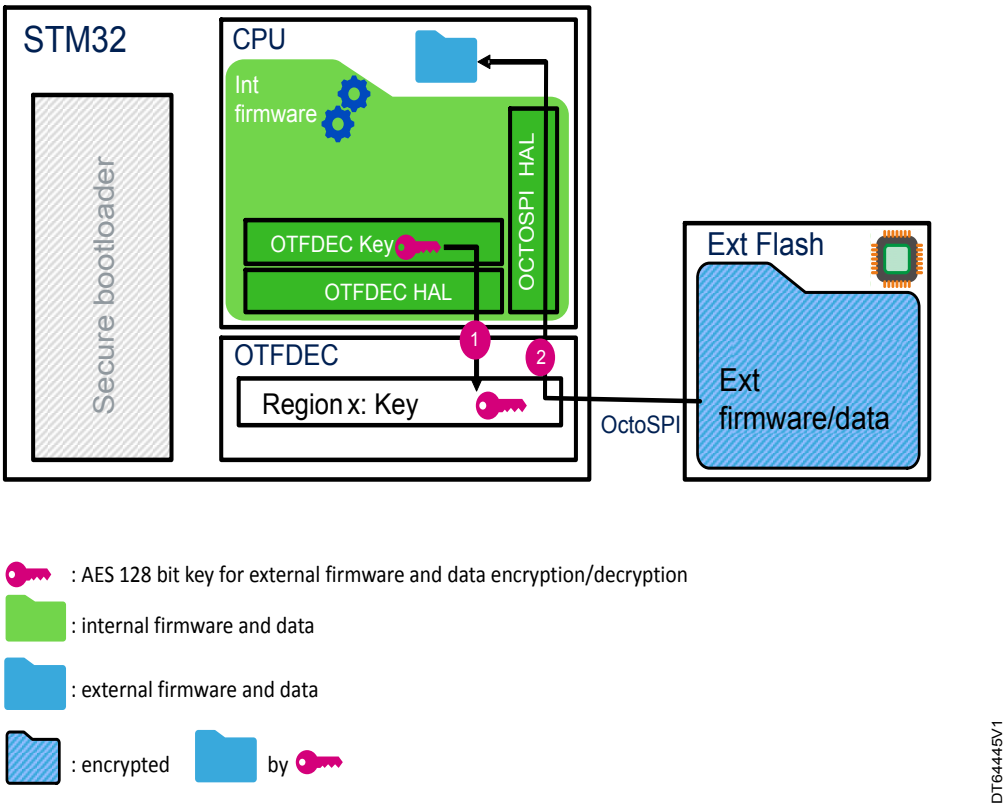


1. *Internal flash memory programming.*
2. *AES external firmware and data key programming in OTFDEC peripheral.*
3. *External flash memory chunk encryption.*
4. *External flash memory programming.*

Figure 4 above shows that internal firmware and external firmware and data AES key must be installed first(1), then the previously mentioned key is copied within the OTFDEC peripheral(2) that is used to encrypt the external firmware and data for external flash memory(3). At last, OEM external flash memory loader programs the encrypted external firmware and data into the external flash memory(4) .

Figure 5 below depicts the execution of the same internal firmware that, among other tasks, manages external firmware and data decryption.

Figure 5. Internal firmware and external flash memory handling (using global key)



1. *AES external firmware and data key programming in OTFDEC peripheral.*
2. *On-the-fly external flash memory decryption.*

At runtime, during secure boot, the secure internal firmware first copies the AES firmware and data key within the OTFDEC peripheral and activates the OTFDEC region tied to this key(1). Then the CPU can seamlessly read/fetch data/code from external flash memory once the OCTOSPI driver has been initialized(2).

2.1.2.3 *External flash memory encryption with secure bootloader and unique key*

This section focuses on the scenario where the secure bootloader encrypts the external firmware and data with a unique key per device. During SFI, the STM32CubeProgrammer first performs secure programming of the internal flash memory(1) then it requests the STM32 to randomly generate a key dedicated to the external flash memory encryption(2). The STM32 uses the TRNG peripheral to generate this key. After key generation the secure bootloader programs this last key(3) in the internal flash memory.

The STM32 uses the OTFDEC peripheral to encrypt the external firmware and data(4)(5). The STM32 stores the encryption result within SRAM. Then an external flash memory loader takes the previous result to program it within the external flash memory(6).

The external flash memory programming is under the responsibility of OEM, meaning that OEM must use an external flash memory loader that fits its external flash memory specificities. ST does not provide such external flash memory loader, except for demonstration purpose on ST boards supporting external flash memory.

Consequently, OEM must implement in internal flash memory the firmware parts dedicated to external flash memory handling (external flash memory keys and drivers).

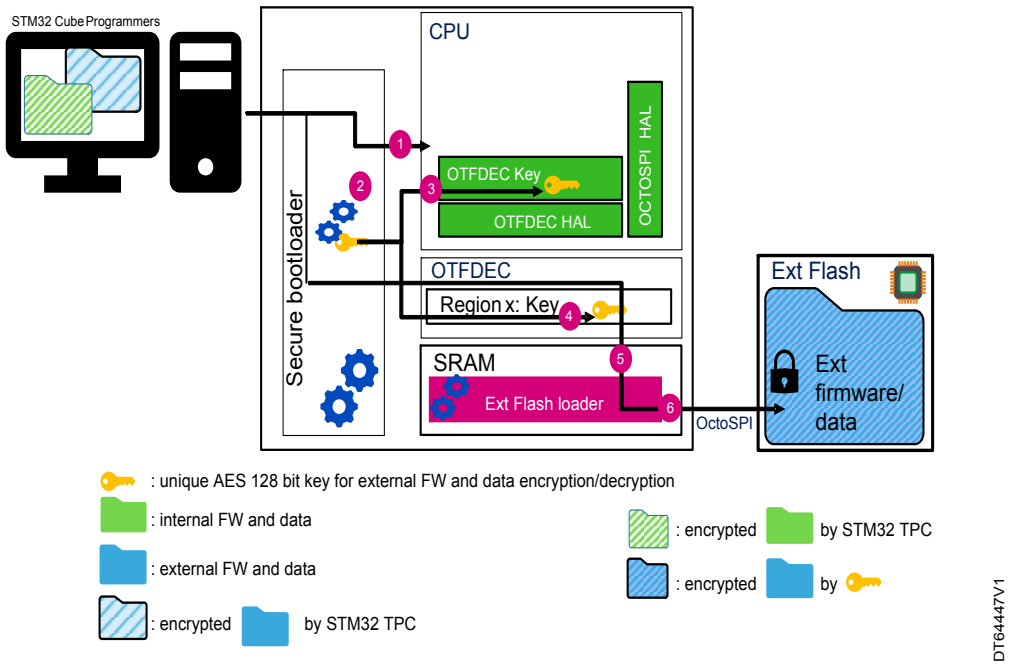
OEM must create the SFI image with:

- Internal firmware and data (include external flash memory drivers).
- External firmware and data AES key.
- External firmware and data.

The STM32CubeProgrammer requests the STM32 secure bootloader to randomly generate a unique set of keys during the SFI sequence. Then OEM must take care to specify, within STM32 TPC, the same address in internal flash memory for key generation than the one its internal firmware uses for external flash memory on-the-fly decryption with OTFDEC peripheral.

The picture below shortly depicts an SFI sequence where the STM32 secure bootloader handles both internal firmware installation and external firmware installation with a unique external flash memory AES key and the help of an external flash memory loader.

Figure 6. External flash memory encryption with secure bootloader and unique AES Key

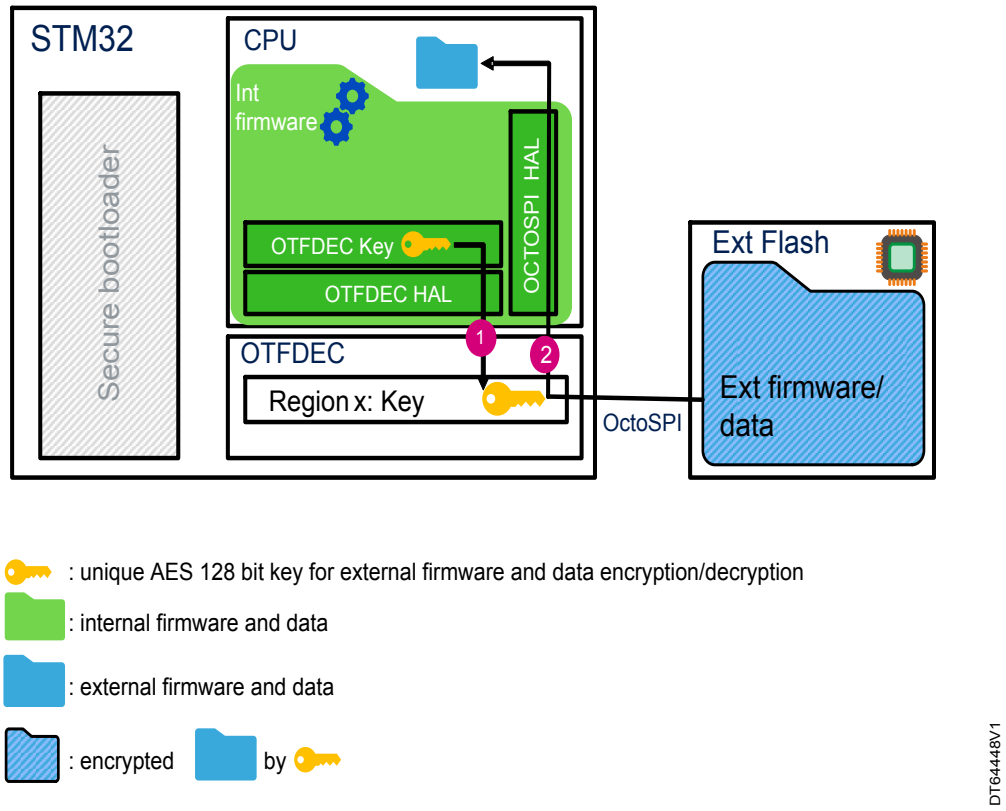


1. *Internal flash memory programming.*
2. *AES external firmware and data key generation.*
3. *AES external firmware and data key programming in internal flash memory.*
4. *AES external firmware and data key programming in OTFDEC peripheral.*
5. *External flash memory chunk encryption.*

6. External flash memory programming.

Figure 7 below depicts the way the internal firmware manages the OTFDEC and external flash memory unique key to decrypt on-the-fly external firmware and data.

Figure 7. Internal firmware and external flash memory handling (using unique key)



1. AES external firmware and data key programming in OTFDEC peripheral.

2. On-the-fly external flash memory decryption.

At runtime, during secure boot, the secure internal firmware first copies the AES firmware and data key within the OTFDEC peripheral and activates the OTFDEC region tied to this key(1). Then the CPU can seamlessly read/ fetch data/code from external flash memory once the OCTOSPI driver has been initialized(2).

2.2 SFI security features

The SFI security features are the following:

- Only genuine STMicroelectronics STM32 microcontrollers can install the protected firmware.
- The number of STM32 devices on which the firmware has been installed can be counted.
- Authenticity, integrity and confidentiality of the OEM internal firmware and option bytes are checked and user flash memory is programmed with decrypted firmware, OBK (in case of STM32H5 and STM32H7RS), and option bytes. If applicable, for the confidentiality of the OEM external firmware, the STM32 receives encrypted OEM external firmware, decrypts this firmware, and re-encrypts with a device unique or global key before programming in external flash memory.

2.3 SFI constraints

Due to SFI design, SFI requires the installed application to run with:

- A minimum RDP/Product State level (see 4.2 and 4.3)
- TrustZone enabled for product with ARM Cortex-M TrustZone.

3 STM32 secure bootloader

The STM32 secure bootloader, implementing SFI, is programmed by STMicroelectronics during STM32 manufacturing. Its main task is to manage protocol communication between STM32CubeProgrammer and STM32 device in order to:

- identify STM32 device
- exchange device certificate and license
- download SFI encrypted image inside STM32

3.1 STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx

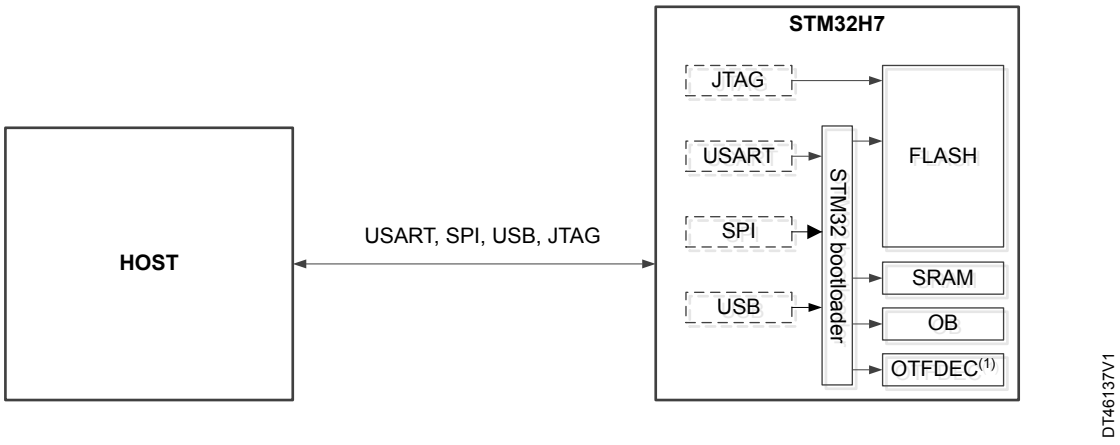
3.1.1 Secure bootloader overview

On the STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx, the secure bootloader is stored in the internal boot ROM (system memory) and supports following interfaces: USART, SPI, USB-DFU and JTAG.

The STM32H75xxx/ STM32H7B0xx /STM32H7B3xx/ STM32H730xx /STM32H733xx/STM32H735xx secure bootloader permits to run the SFI process several times after complete erase of the internal user flash memory, if erase allowed by installed application.

For more details on STM32 bootloader protocols, refer to [AN3155] (USART protocol), [AN4286] (SPI protocol) and [AN3156] (USB DFU protocol). The documents are available on www.st.com (unless an NDA applies).

Figure 8. STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx secure bootloader



1. *OTFDEC peripheral is only available on STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/ STM32H735xx devices.*

3.1.2 User flash memory mapping

On STM32H75xxx/ STM32H7B0xx /STM32H7B3xx/ STM32H730xx /STM32H733xx/STM32H735xx, since the secure bootloader is stored in the system memory, the product is delivered in RDP level0 and all the user flash memory is available for OEM.

The OEM firmware can be built to start at the beginning of the user flash memory (starting address 0x0800 0000).

3.1.3 External flash memory mapping

As explained before, the STM32H7B0xx /STM32H7B3xx/ STM32H730xx /STM32H733xx/STM32H735xx can decrypt external firmware and data on-the-fly. This can be done if they reside on external flash memory connected to the STM32H7B0xx /STM32H7B3xx/ STM32H730xx /STM32H733xx/STM32H735xx through the OCTOSPI interface. Consequently, the external firmware must be built with addresses within the range 0x90000000 to 0x9FFFFFFF (for OCTOSPI1) or within the range 0x70000000 to 0x7FFFFFFF (for OCTOSPI2).

3.1.4 Secure boot path

After successful SFI process and if a secure area is set, the STM32H75xxx/STM32H7B0xx/STM32H7B3xx/ STM32H730xx/STM32H733xx/STM32H735xx always boot in the secure area closest to the configured boot address.

More information is available in [RM0399] for STM32H755xx/STM32H757xx.

More information is available in [RM0433] for STM32H750xx/STM32H753xx.

More information is available in [RM0455] for STM32H7B0xx/STM32H7B3xx.

More information is available in [RM0468] for STM32H730xx/STM32H733xx/STM32H735xx.

3.2 STM32H7RS

Note: Hereafter, "STM32" refers to STM32H7RS (applicable to whole Section 3.2).

The STM32 SFI process can only perform secure programming in user flash memory firmware when all necessary data are loaded in SRAM (no interaction between host and STM32 device during SFI process) and product state is set to provisioning at the beginning of the SFI procedure.

3.2.1 Secure bootloader overview

On STM32, the secure bootloader is split in two parts.

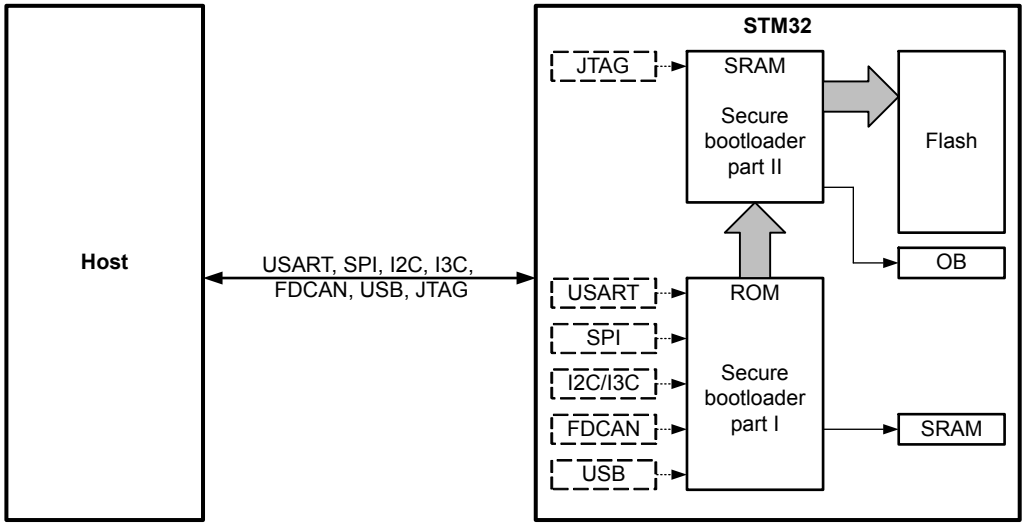
The first part is stored in boot ROM (system memory). It is responsible for loading the second part of the bootloader in SRAM through the supported interfaces. The supported interfaces are USART, SPI, I2C, I3C, FDCAN, USB and JTAG.

The second part runs in SRAM and is responsible for executing the SFI process, applying the SFI protocol and parsing all the data previously loaded in SRAM.

The STM32 secure bootloader permits to run the SFI process several times after complete erase of user flash memory, if the erase is allowed by the previously installed application.

For more details on the STM32 bootloader protocols, refer to [AN3155] (USART protocol), [AN4286] (SPI protocol), [AN4221] (I2C), [AN5927] (I3C), [AN3154] (FDCAN) and [AN3156] (USB). The documents are available on www.st.com (unless an NDA applies).

Figure 9. STM32H7RS secure bootloader



3.2.2 User flash memory mapping

On STM32, the secure bootloader is stored in system memory and SRAM, the product is delivered in Open product state and all the user flash memory is available for OEM.

More information is available in [RM0477].

3.3 STM32L5, STM32U3, STM32U5, STM32WBA5, STM32WBA6, and STM32H5

Note: Hereafter, "STM32" refers to STM32L5, STM32U3, STM32U5, STM32WBA5 , STM32WBA6, and STM32H5 series (applicable to whole Section 3.3).

The STM32 SFI process can only perform secure programming in user flash memory firmware using Arm® TrustZone®, i.e with TZEN bit from FLASH_OPTR register set to 1.

3.3.1 Secure bootloader overview

On STM32, the secure bootloader is split in two parts.

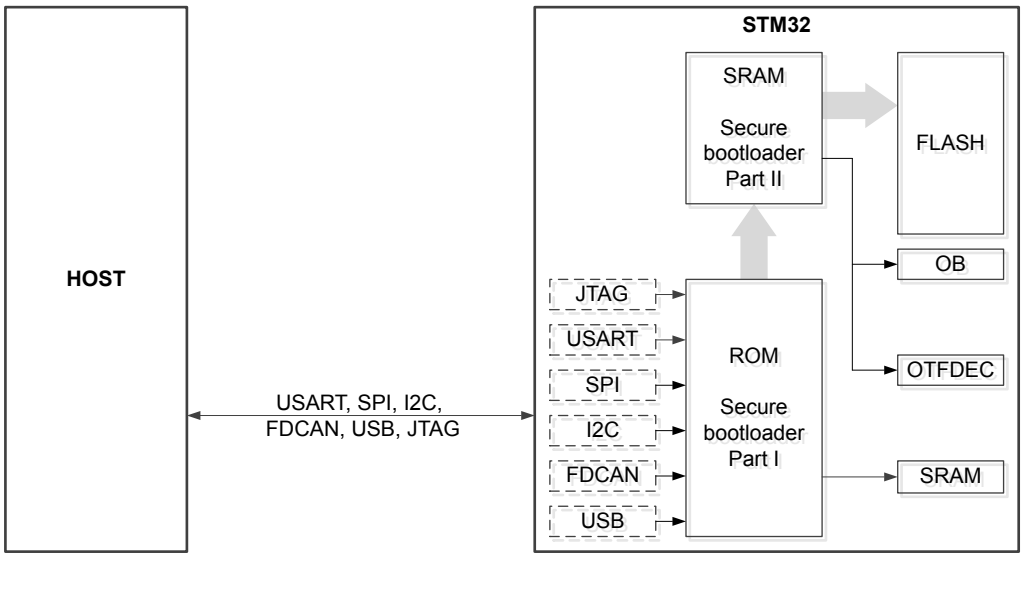
The first part is stored in boot ROM (system memory). It is responsible for loading the second part of the bootloader in SRAM through the supported interfaces. The supported interfaces are USART, SPI, I2C, FDCAN, USB and JTAG.

The second part runs in SRAM and is responsible for executing the SFI process and applying the SFI protocol thanks to the commands received through the above mentioned supported interfaces.

The STM32 secure bootloader permits to run the SFI process several times after complete erase of user flash memory, if the erase is allowed by the previously installed application.

For more details on the STM32 bootloader protocols, refer to [AN3155] (USART protocol), [AN4286] (SPI protocol), [AN4221] (I2C), [AN3154] (FDCAN) and [AN3156] (USB). The documents are available on www.st.com (unless an NDA applies).

Figure 10. STM32 secure bootloader



3.3.2 User flash memory mapping

On STM32, the secure bootloader is stored in system memory and SRAM, the products are delivered in RDP level0 and all the user flash memory is available for OEM.

3.3.2.1 No STM32TRUSTEE

The OEM internal firmware can be built to start at the beginning of the user flash memory (starting address 0x0800 0000). After a SFI, the SRAM is fully available for OEM.

3.3.2.2 STM32TRUSTEE (only on STM32H5)

Only a part of the user flash memory is available for OEM firmware (see 4.3.2)

3.3.3 External flash memory mapping

As explained before, the STM32L562xx, STM32U585xx, STM32U5Axxx, STM32U5Gxxx, and STM32H5 can decrypt external firmware and data on-the-fly. This can be done if they reside on external flash memory connected to the STM32 through the OCTOSPI interface. Consequently, the external firmware must be built with addresses within the range 0x90000000 to 0x9FFFFFFF (OCTOSPI1) or within the range 0x70000000 to 0x7FFFFFFF (only for STM32U585xx with OCTOSPI2).

The STM32L562xx, STM32U585xx, STM32U5Axxx, STM32U5Gxxx, and STM32H5 OCTOSPI peripheral can drive external flash memory up to 256 Mbytes. The external flash available memory is dependent from the external flash connected by the user to the STM32L562xx, STM32U585xx, STM32U5Axxx, STM32U5Gxxx or STM32H5.

3.3.4 Secure boot path

In order to get STM32 booting on secure user flash memory (at SECBOOTADD0 from FLASH_SECBOOTADD0R Option Byte register) after successful SFI, it is recommended to set to 0 nSWBOOT0 and set nBOOT0 to 1 from FLASH_OPTR register.

More information is available in:

- [RM0438] for STM32L5 series
- [RM0487] for STM32U3 series
- [RM0456] for STM32U5 series
- [RM0493] for STM32WBA5 series
- [RM0515] for STM32WBA6 series
- [RM0481] for STM32H5 series.

3.4 STM32WL5

3.4.1 Secure bootloader overview

The secure bootloader is split in two parts.

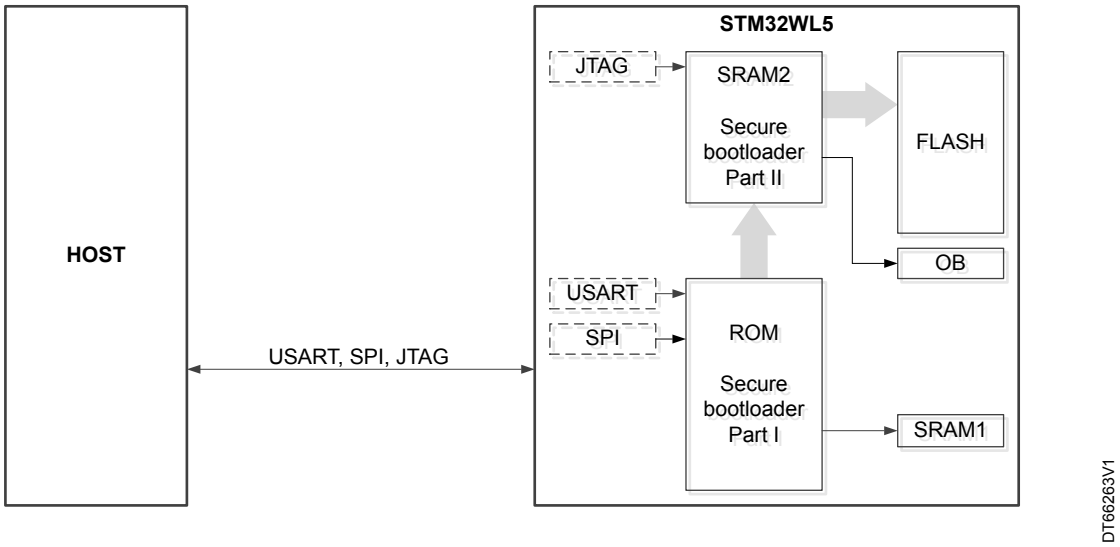
The first part is stored in boot ROM (system memory). It is responsible for loading the second part of the bootloader in SRAM2 through the supported COM ports. The supported COM ports are USART, SPI and JTAG.

The second part runs in SRAM2 and is responsible for executing the SFI process and applying the SFI protocol thanks to the commands received through the above mentioned supported COM ports.

The secure bootloader permits to run the SFI process several times. For this, the STM32WL5 has been reprogrammed to its initial OB configuration (as delivered by STMicroelectronics).

For more details on the STM32 bootloader protocols, refer to [AN3155] (USART protocol) and [AN4286] (SPI protocol).

Figure 11. STM32WL5 secure bootloader



3.4.2 User flash memory mapping

The secure bootloader is stored in system memory and SRAM2, the products are delivered in RDP level0 and all the user flash memory is available for OEM.

The OEM internal firmware can be built to start at the beginning of the user flash memory (starting address 0x0800 0000). After a SFI, the SRAM2 is fully available for OEM.

3.4.3 Secure boot path

In order to get the STM32WL5 booting on secure user flash memory after successful SFI, it is recommended to set nSWBOOT0 to 0 and nBOOT0 to 1 in FLASH_OPTR register.

More information is available in [RM0453].

4 SFI image preparation

The SFI image can be prepared by using STMicroelectronics STM32 Trusted Package Creator available both in CLI (command line interface) and GUI (graphical user interface) modes. This tool can be downloaded free of charge from www.st.com.

Note: STM32 Trusted Package Creator tool is included inside STM32CubeProgrammer tool package (STM32CubeProg) available on www.st.com.

It allows the generation of SFI images for STM32 microcontrollers.

Figure 12 shows STM32 TPC screen to be used for building SFI option bytes configuration file (.csv) that is needed for SFI image creation (examples of SFI OB configuration file available in a dedicated STM32CubeProgrammer directory). More information is available in [UM2238].

For STM32 supporting OEM password keys, values 0x00000000 (default) and 0xFFFFFFFF do not enable OEM password.

Figure 12. STM32 Trusted Package Creator - SFI option bytes

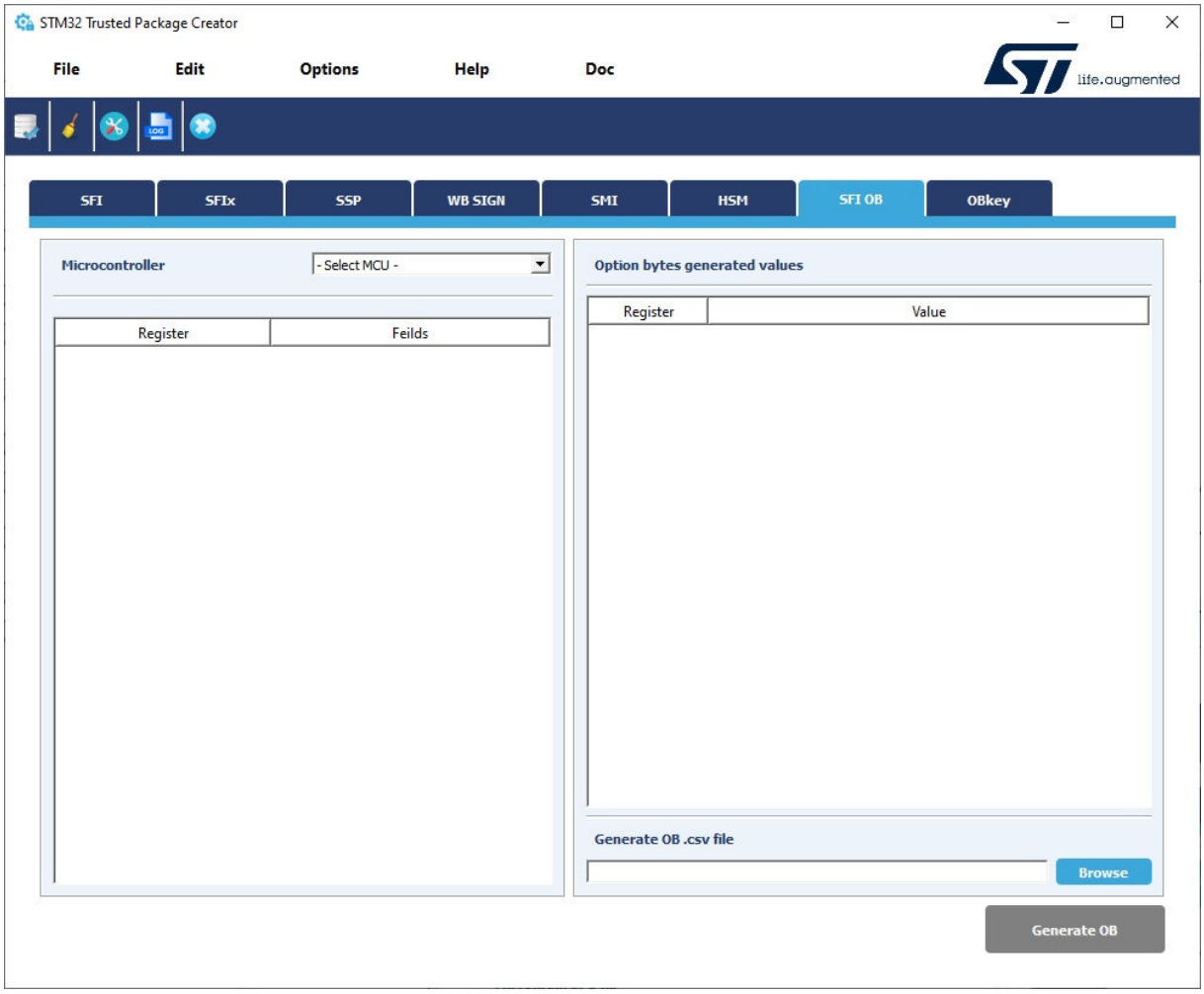
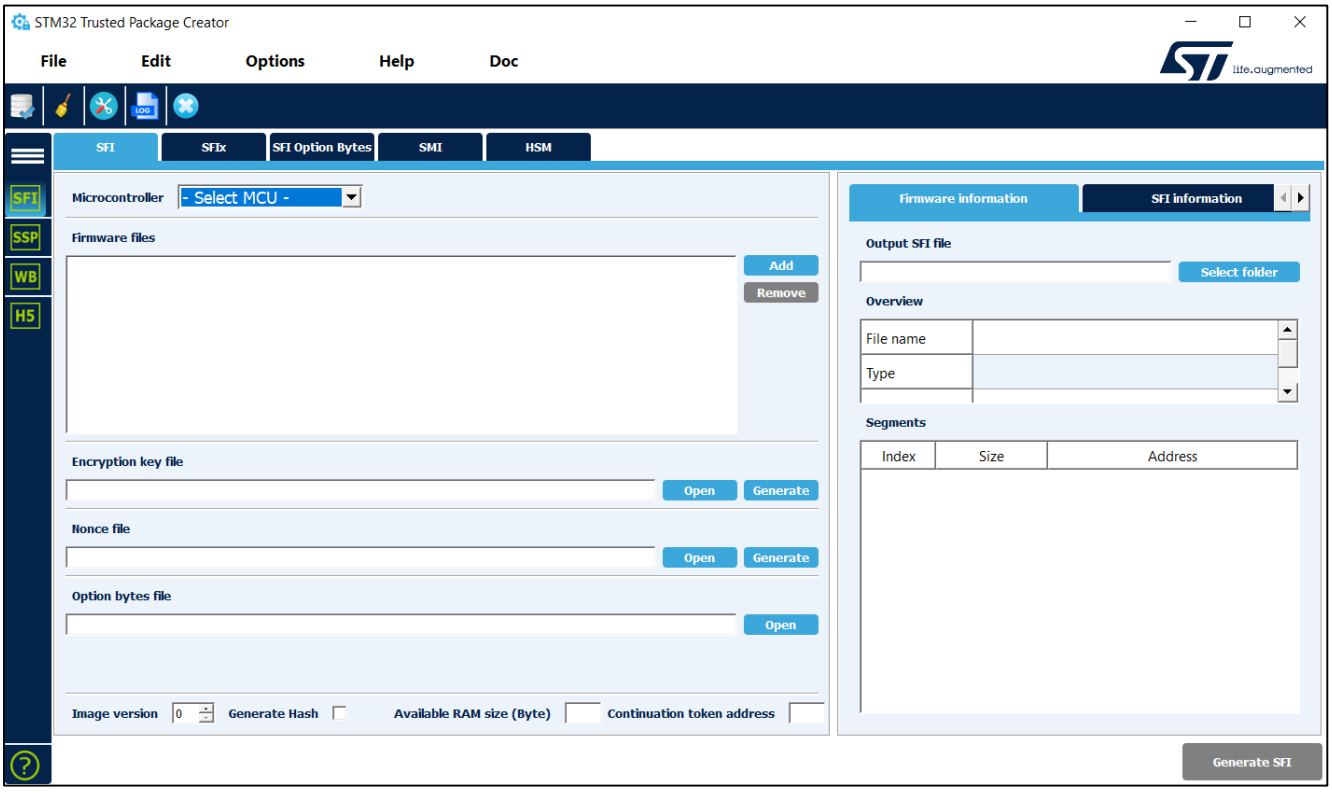


Figure 13 shows STM32 TPC screen to be used to create SFI image.

Figure 13. STM32 Trusted Package Creator - SFI image creation



4.1 SFI firmware image format

The SFI format is an encryption format for firmware created by STMicroelectronics. It uses AES-GCM algorithm with a 128-bit key to transform a firmware in Elf, Hex, Bin or Srec formats into an encrypted and authenticated firmware in SFI format.

An SFI firmware image is composed of a header plus several areas. The areas are usually contiguous firmware areas. The last area is the configuration area containing the option byte values to be programmed when the SFI is complete.

4.2 SFI firmware image creation procedure for STM32H7/L5/U3/U5/WBA5/WBA6

To obtain SFI firmware images in the correct format (see Section 4.1: SFI firmware image format), the OEM must follow the steps below to build an SFI image:

1. Build the firmware image(s) using regular development tools.
2. Build the option byte configuration file.
   This file contains the option bytes that the customer plans to apply on the STM32 and that are programmed thanks to SFI. Among option bytes, the user must define the intended RDP and TZEN values to be applied in order to enable the protection of user firmware and data against debug connection. Refer to applicable reference manual for more details on RDP level.
   Table 3 below sums up the minimum RDP level that the customer must apply within the option byte file in order to protect the user firmware and data within either internal flash memory or (if applicable) external flash memories.
3. Generate the binary image(s). A binary image does not need to be contiguous (that is it can cover multiple disjoint areas).

Table 3. Minimum RDP requirements

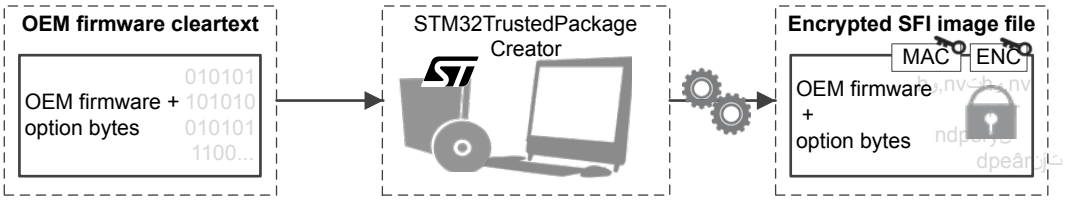
Microcontroller	Use Case	Min RDP level	TZEN (TrustZone enable)
STM32H75xxx	SFI on internal flash memory	1	Not applicable
STM32H7B0xx / STM32H7B3xx	SFI on internal flash memory and external flash memories	1	Not applicable
STM32H730xx / STM32H733xx/ STM32H735xx	SFI on internal flash memory and external flash memories	1	Not applicable
STM32L5 STM32U3 STM32U5 STM32WBA5 STM32WBA6	SFI on internal flash memory Protection of secure internal flash memory only	0.5	1 (enabled)
	SFI on internal flash memory Protection of secure and non-secure internal flash memory only	1	1 (enabled)
	SFI on internal and external flash memories(1) Protection on both memories	1	1 (enabled)
STM32WL5	SFI on internal flash memory Protection of secure internal flash memory only	0(2)	1 (enabled)
	SFI on internal flash memory Protection of secure and non-secure internal flash memory only	1	1 (enabled)

1. *Applicable to STM32L562xx, STM32U585xx, STM32U5Axxx and STM32U5Gxxx only (with OTFDEC).*
2. *With DDS = 1 and FSD = 0.*

Figure 14 shows the SFI image preparation procedure based on the STM32 Trusted Package Creator.

Note: STM32 Trusted Package Creator tool is included inside STM32CubeProgrammer tool package (STM32CubeProg) available on www.st.com .

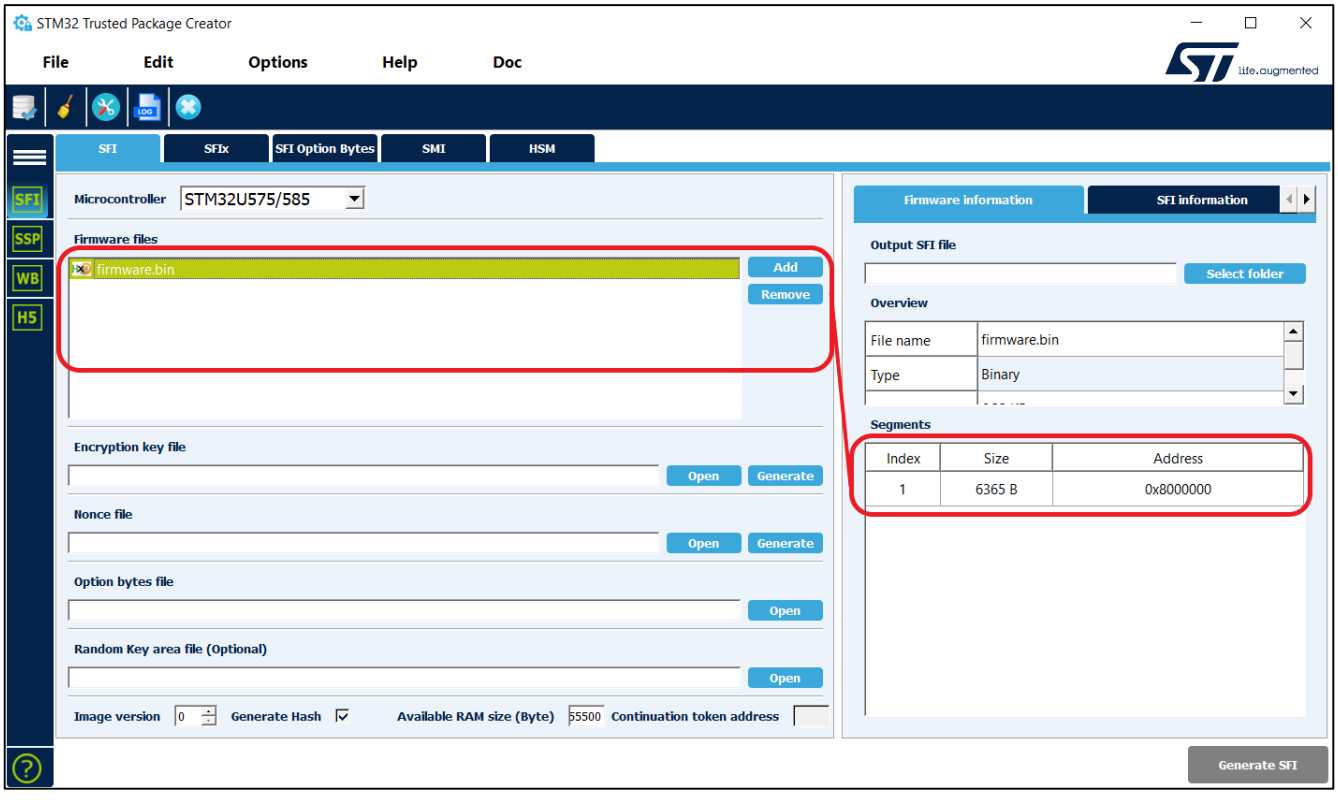
Figure 14. SFI image preparation procedure



4.2.1 Internal flash memory only

To successfully generate an SFI image from the supported input firmware formats, click the SFI GUI tab, select the targeted micro-controller and fill in the interface fields with valid values:

Figure 15. SFI image generation tab example



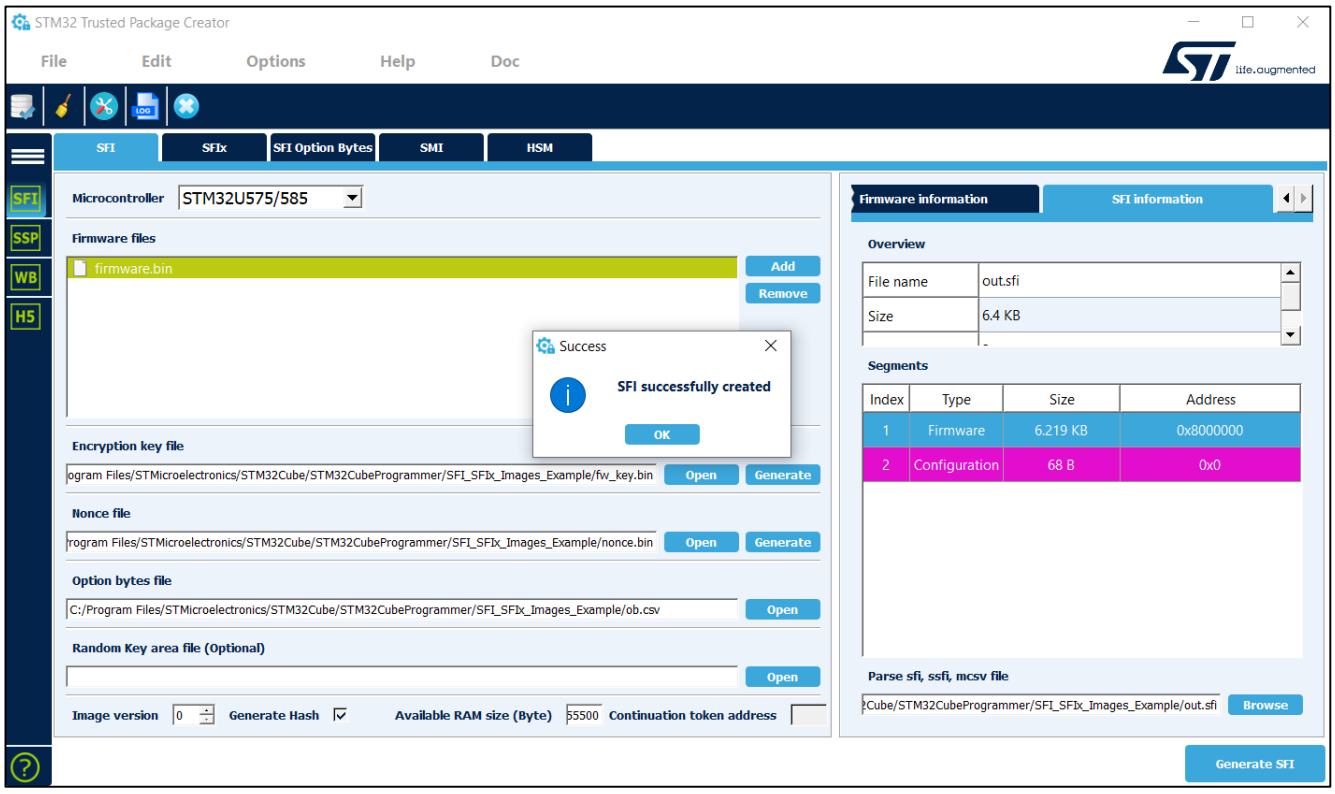
1. *Add the firmware file using the **Add** button located within the **Firmware files** area appended to the input firmware files list.*
2. *Make sure you are referring to the right valid firmware. Details are available in the **Firmware information** section once you have the Input firmware file is selected.*
3. *Select the **Encryption key** and **Nonce** files. They can be selected either by entering their absolute or relative paths, or by selecting them with the **Open** button.*
4. *Make sure that the size of the Encryption key file (16 bytes) and the Nonce file (12 bytes) are respected. In order to properly protect the firmware confidentiality, the AES secret key must be different every time the OEM encrypts a new firmware or a new release of the same firmware.*
5. *Select the **Option bytes** file in .csv format. This is the only format supported.*
6. *Enter the image version of the SFI to be generated. It must range from 0 to 255.*
7. *Select **Random key area file** using the **Open** button (this is optional). Key area requests STM32 to randomly generate data of a specified size and at a specified address filled by user within key area file in .kcsv format.*
8. *The **Generate Hash** option must be enabled on all STM32 devices supporting SFI.*
9. *Select MCU to fill available RAM size for the SFI operation.*
10. *Select the address where the continuation token is going to be stored. This field is only relevant for STM32H75xxx/ STM32H7B0xx /STM32H7B3xx/ STM32H730xx /STM32H733xx/STM32H735xx products, depending on firmware mapping, 0x081FF000 can be used as a nominal value (0x080FF000 as nominal value for STM32H733xx/STM32H735xx).*
11. *For STM32H75xxx/ STM32H7B0xx /STM32H7B3xx/ STM32H730xx /STM32H733xx/STM32H735xx only, select SMI files, if any (optional field).*
12. *Select the **Output** file folder path where the SFI image, out.sfi, is going to be generated.*

13. *Generate the SFI file by clicking the **Generate SFI** button. If all the fields are filled properly, the "SFI created successfully" message is displayed.*

More information is available in [UM2238].

Note: SMI files selection is only mandatory for secure module installation process. It is only supported on STM32H75xxx/ STM32H7B0xx /STM32H7B3xx/ STM32H730xx /STM32H733xx/STM32H735xx products.

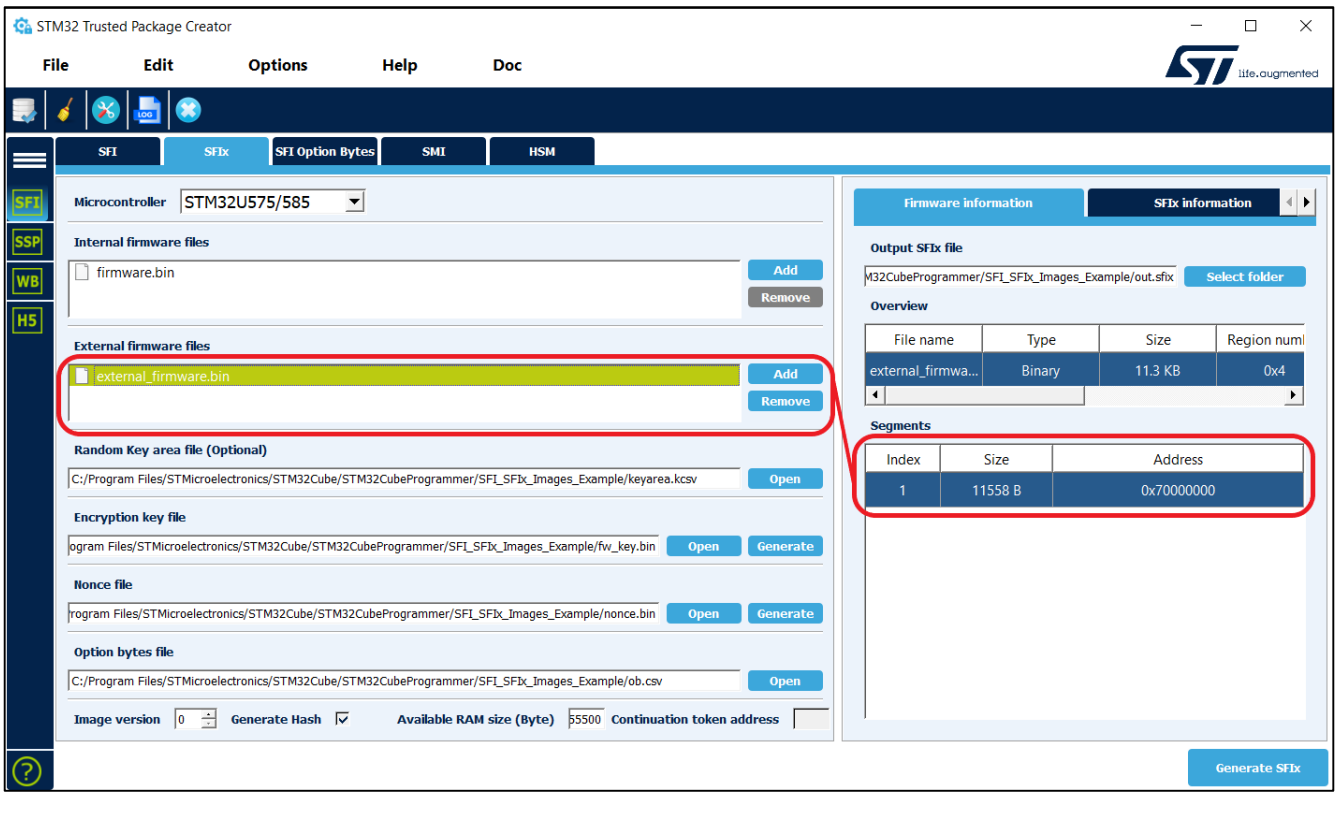
Figure 16. SFI image successful generation



4.2.2 Both internal and external flash memories

To successfully generate a SFI image including external FW and Data (also called SFIx image) from the supported input firmware formats, click the SFIx GUI tab, select the targeted micro-controller and fill in the interface fields with valid values:

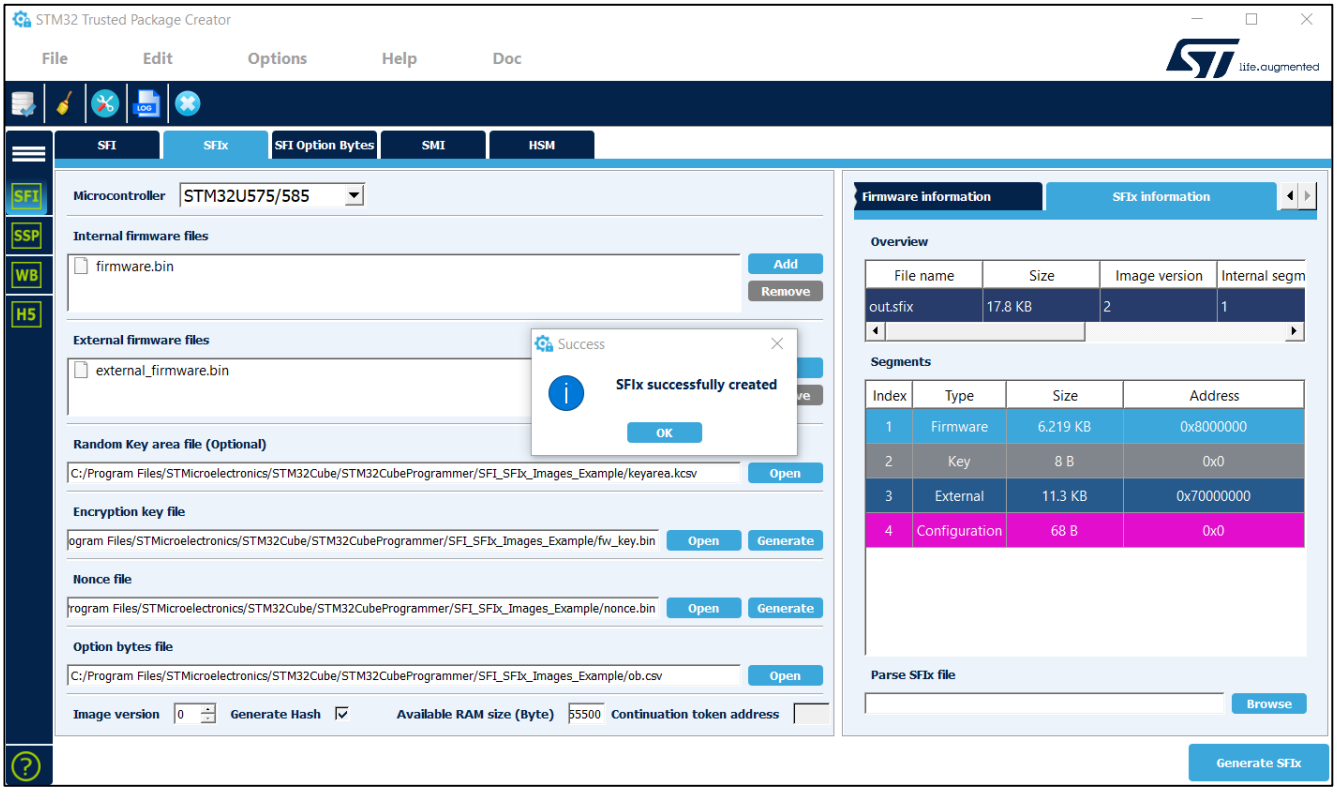
Figure 17. **SFIx image generation tab example**



1. *Add the internal firmware file(s) using the **Add** button located within the **Internal** firmware files area appended to the input firmware files list. If the file is in bin format, the destination address has to be filled. If OEM uses global AES external flash memory keys, then internal firmware files must include those keys at the right address.*
2. *Add the external firmware file(s) using the **Add** button located within the **external** firmware files area appended to the input firmware files list. When adding an external firmware, several additional fields have to be filled (destination address in external flash memory, OTFD region number, OTFD mode and global AES external flash key address).*
3. *Select key area file using the **Open** button (this is optional). Key area requests STM32 to randomly generate data of a specified size and at a specified address filled by user within key area file in .kcsv format.*
4. *Select the **Encryption key** and **Nonce** files. They can be selected either by entering their absolute or relative paths, or by selecting them with the Open button.*
5. *Make sure that the size of the Encryption key file (16 bytes) and the Nonce file (12 bytes) are respected. In order to properly protect the firmware confidentiality, the AES secret key must be different every time the OEM encrypts a new firmware or a new release of the same firmware.*
6. *Select the **Option bytes** file in .csv format. This is the only format supported.*
7. *Enter the image version of the SFIx image to be generated. It must range from 0 to 255.*
8. *The **Generate Hash** option must be enabled on all STM32 devices supporting SFI.*
9. *Select MCU to fill available RAM size for the SFIx operation.*
10. *Select the address where the continuation token is going to be stored. This field is only relevant for STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx products, depending on firmware mapping, 0x081FF000 can be used as a nominal value (0x080FF000 as nominal value for STM32H733xx/STM32H735xx).*
11. *Select the **Output** file folder path where the SFIx image, out.sfix, is going to be generated.*
12. *Generate the SFIx image file by clicking the "**Generate SFIx**" button. If all the fields are filled properly, the "SFIx created successfully" message is displayed.*

More information is available in [UM2238].

Figure 18. **SFIx image successful generation**



DT67585V2

4.2.3 OEM passwords for RDP regression (STM32U3, STM32U5, STM32WBA5 and STM32WBA6 series only)

On STM32U3, STM32U5, STM32WBA5, and STM32WBA6, it is possible to set the OEM1 and OEM2 passwords during the option bytes programming. To do so, the OEM must provide passwords different than 0x00000000 or 0xFFFFFFFF in the dedicated fields inside the Option-Bytes configuration file.

The OEM can also decide to diversify these passwords for each programmed chips thanks to the KEYDIVERS field in the Option-Bytes configuration file. This diversification is done during the secure programming of the SFI image using the KDF256 algorithm, with the OEM-Key registers and the chip's Device Authentication ID as inputs. Depending on part numbers, the following OEM password diversification schemes are available:

Figure 19. **OEM password diversification scheme for STM32U5, STM32WBA5 and STM32WBA6**

DT72202V1

Figure 20. **OEM password diversification scheme for STM32U3**

DT77218V1

Knowing the master keys, the OEM can read the Authentication ID from the DBG_MCU interface, compute the chip's diversified keys on his own and then trigger the desired RDP regression.

The OEMDIVERS field from the Option-Bytes can be configured in multiple ways:

- 0x00000000 means that no OEM password is diversified.
- 0x0000EABE means that only OEM1 password is diversified.
- 0xEABE0000 means that only OEM2 password is diversified.
- 0xEABEEABE means that both OEM1 and OEM2 passwords are diversified.

Caution: *On STM32U5, STM32WBA5 and STM32WBA6 the OEM passwords can be reset in RDP 0 only.*
On STM32U3, once programmed the OEM passwords can never be reset.

4.2.3.1 *KDF256 details*

The KDF256 algorithm is described by ANSI X9.63-2011.

KDF256 (OEMxKeyR1 || OEMxKeyR2 || AUTHID) is equivalent to SHA256 (OEMxKeyR1 || OEMxKeyR2 || AUTHID || 0x00000001) with OEMxKeyR1, OEMxKeyR2, AUTHID, and the counter all expressed in big endian.

KDF256 (OEMxKeyR1 || OEMxKeyR2 || OEMxKeyR3 || OEMxKeyR4 || AUTHID) is equivalent to SHA256 (OEMxKeyR1 || OEMxKeyR2 || OEMxKeyR3 || OEMxKeyR4 || AUTHID || 0x00000001) with OEMxKeyR1, OEMxKeyR2, OEMxKeyR3, OEMxKeyR4, AUTHID, and the counter all expressed in big endian.

4.2.3.2 *OEM passwords diversification example on STM32U5*

OEM1KEYR1 = 0x11111111

OEM1KEYR2 = 0x22222222

AUTHID = 0x10e0e13e

KDF256 (OEM1 || OEM2 || AUTHID) = SHA256 (11 11 11 11 22 22 22 22 10 0e e1 3e 00 00 00 01) = 98 ed c0 39 2e 06 c3 17 03 21 20 9c 8b 36 7e 52 a9 05 86 75 88 e4 33 9f 6f 47 44 86 89 64 a4 18

The first four bytes are used for the derived OEM1 password (0x98edc039) and the second group of four bytes is the derived OEM2 password (0x2e06c317).

So, KDF256 result is:

OEM1KEYR1_divers = 0x98edc039

OEM1KEYR2_divers = 0x2e06c317

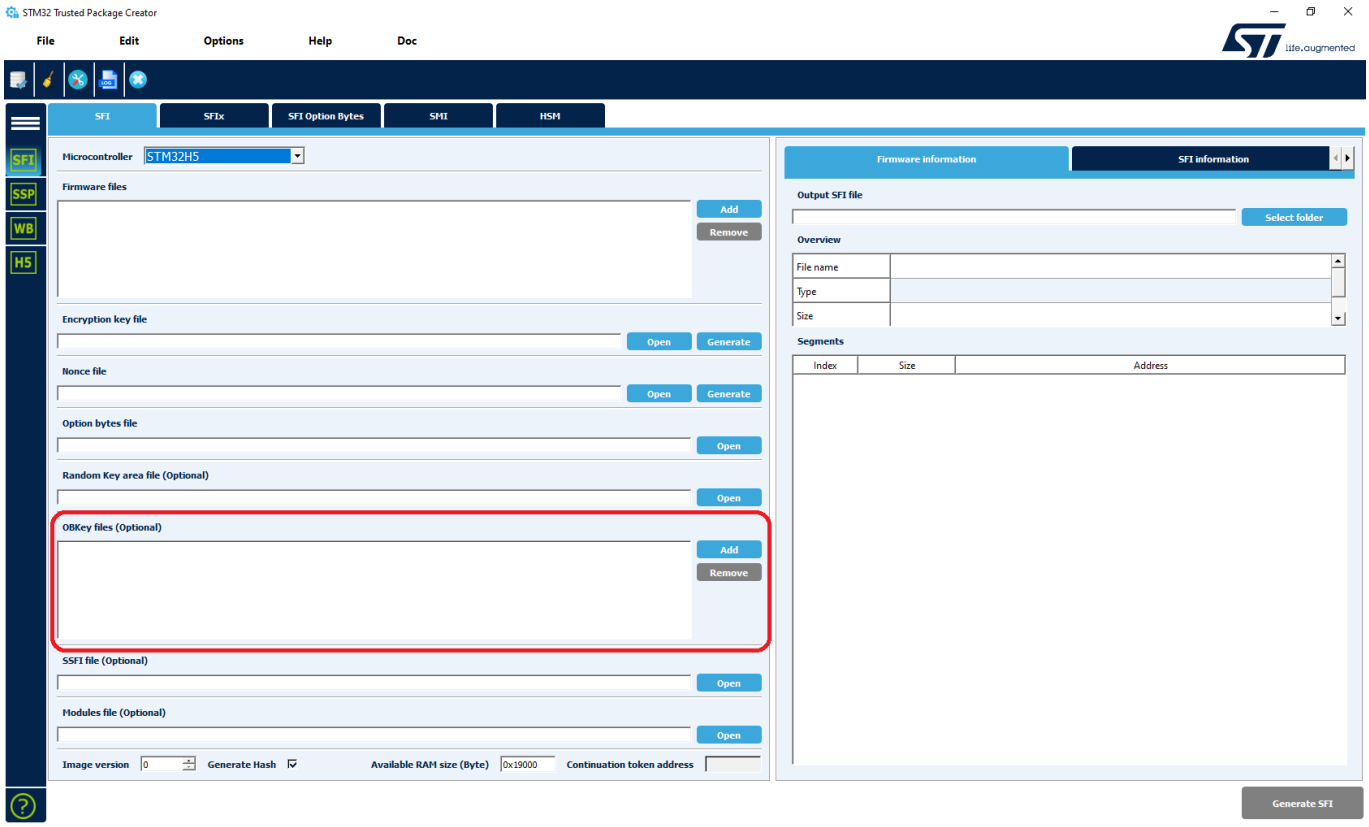
4.3 SFI firmware image creation for STM32H5

4.3.1 No STM32TRUSTEE

The steps are the same than in 4.2 with the following 2 modifications:

- Step2 build the option byte configuration file:
  There is no RDP option byte in STM32H5. The equivalent option byte is the PRODUCT_STATE.
  It must be set to minimum PROVISIONED (that is PROVISIONED, TZ_CLOSED, CLOSED or LOCKED)
  There is the same constraint as in 4.2 about the TZEN option byte value.
- OBK provision:
  It is new compared to 4.2
  Figure 21 depicts the TPC GUI to provision the OBK.

Figure 21. **SFI image generation tab for STM32H5**



4.3.1.1 *Debug authentication*

Provision OBK to be able to perform debug authentication procedure that is used to:

- perform a full or partial regression
- open the debug for a given HDP level and security state (HDPL1_S, HDPL1_NS, HDPL2_S, HDPL2_NS, HDPL3_S, HDPL3_NS)

4.3.2 STM32TRUSTEE

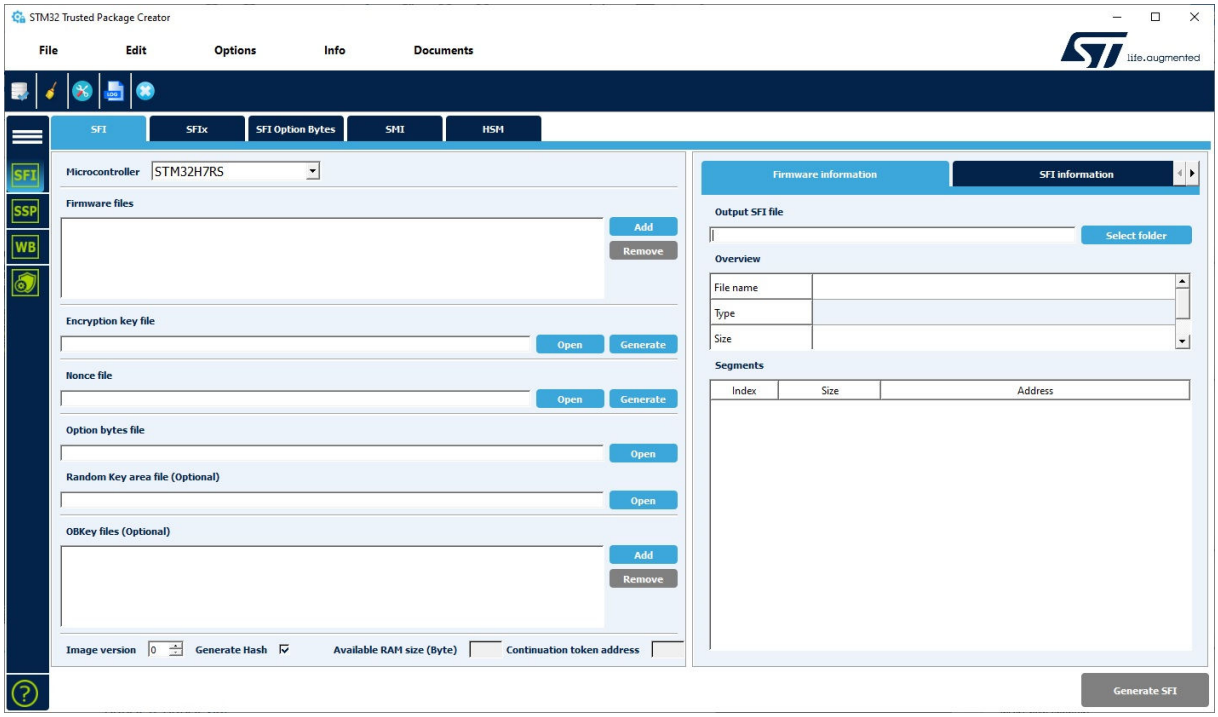
See [WikiSTM32H5].

4.4 SFI firmware image creation for STM32H7RS

4.4.1 Internal flash memory only

The steps are the same than in 4.2 plus the OBK to provision, SFIx is not supported on STM32H7RS:

Figure 22. **SFI image creation tab for STM32H7RS**



4.4.2 Minimum OB requirements

For FLASH_ROTSRP register, it is mandatory to follow these rules:

- OEM_PROVD must be set to 0xB4 (provisioned)
- DBG_AUTH must be set to 0x51, 0x8A or 0xB4 (ECDSA, password or Locked)
- IROT_SELECT must be set to whether 0x6A or 0xB4 (configuration ignored on STM32H7R3/H7R7)

4.4.3 OBK provisioning

The OBK must be provisioned to be able to perform the debug authentication procedure that is used to:

- perform a full regression
- open the debug for a given HDP level (HDPL1, HDPL2, HDPL3)
- force download (allow application update using BL)

Optionally, provision the OBK to configure firmware application. It is mandatory to provision debug authentication OBK first.

More information is available in [RM0477].

5 SFI ST HSM key provisioning

ST HSM smartcard configuration is done using STM32 Trusted Package Creator tool following the below steps:

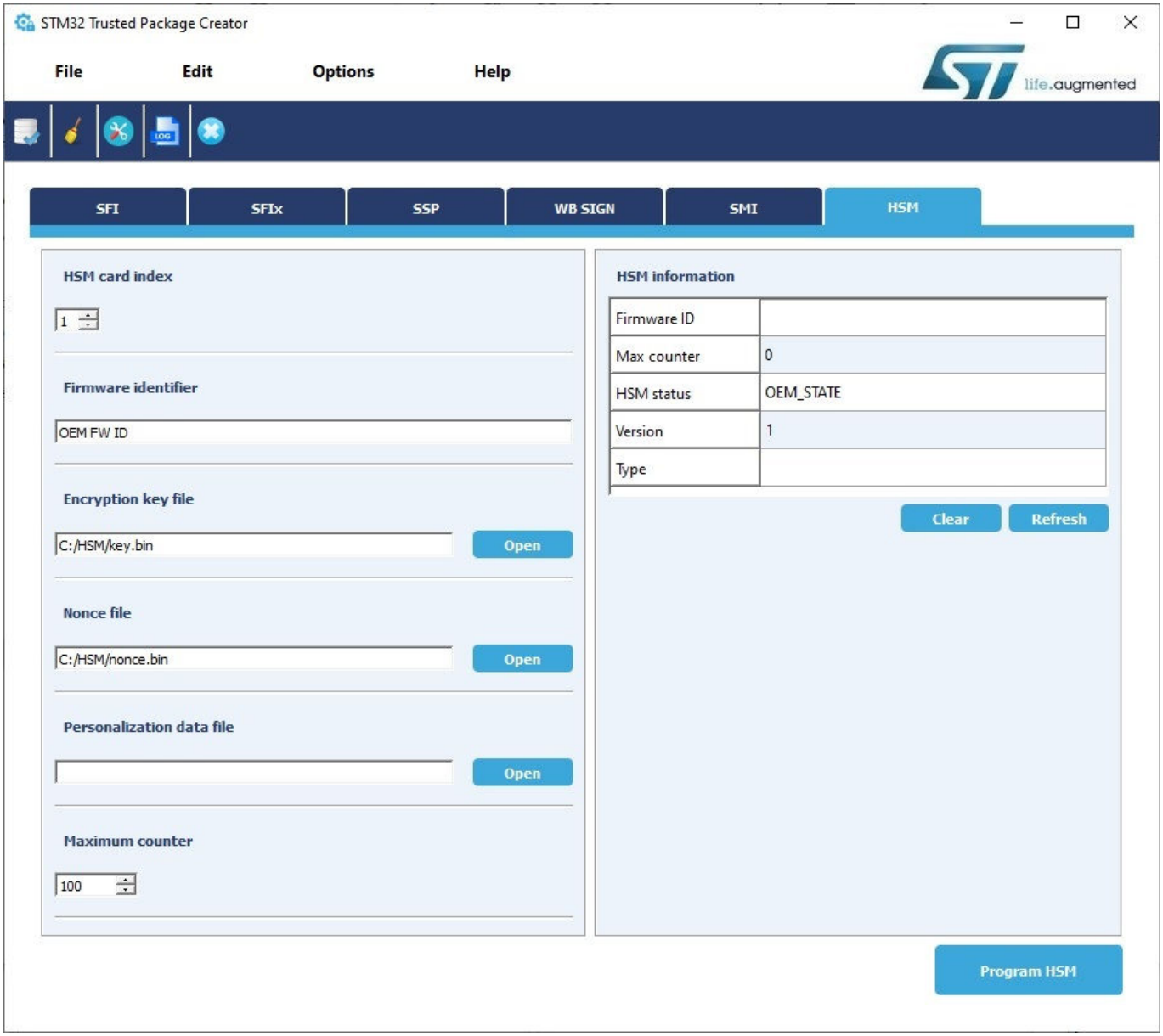
Note: ST HSM configuration can be done only once, the ST HSM is locked after successful programming.
Contact ST sales representative for ST HSM ordering information.

1. Insert a virgin ST HSM in smartcard reader.
2. Select **HSM** tab of STM32 Trusted Package Creator.
3. Add a firmware identifier (allows OEM to identify the correct ST HSM for a given firmware).
4. Open the **Encryption key** and **Nonce** files. They can be selected either by entering their absolute or relative paths, or by selecting them with the **Open** button. In order to properly protect the firmware confidentiality, the AES secret key must be different every time the OEM encrypts a new firmware or a new release of the same firmware.
5. On ST HSM V2, select the personalization data file according to specific STM32 MCU, productID and license type (stm32xx_productID_licenseType_....enc.bin). ProductID is defined in STM32 chip certificate and the different license types are detailed in Section 6.2.
   For security reasons, it is advised to use the most recent personalization data file available in the latest STM32CubeProgrammer version. More info on personalization data in [AN5054].
6. Configure the maximum installation counter.
7. Configure ST HSM by clicking the **Program HSM** button. A feedback window indicating that ST HSM is going to be completely locked is displayed, to confirm and continue the procedure click **Yes**. If all the fields are filled properly, the "HSM programmed successfully" message is displayed.

After key provisioning, ST HSM can be used to secure install protected firmware on **counter** number of STM32 devices.

More information on ST HSM provisioning is available in [AN5054].

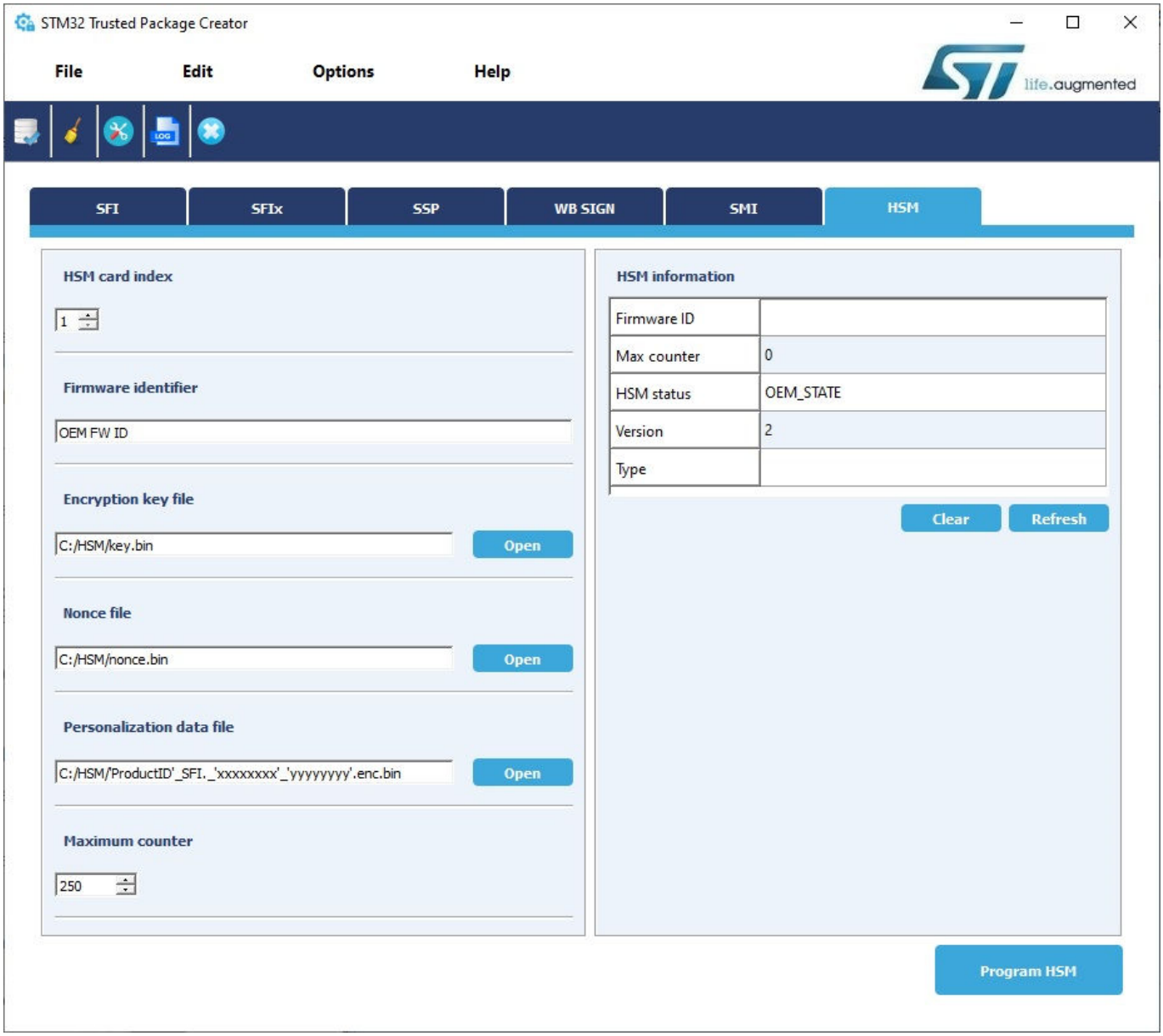
Figure 23. **ST HSM V1 key provisioning example**



Note: When using ST HSM version 1, the "Personalization data file" field is ignored when programming starts. It is only used with ST HSM version 2.

When the card is successfully programmed, a popup window message "HSM successfully programmed" appears, and the ST HSM is locked. Otherwise an error message is displayed.

Figure 24. **ST HSM V2 key provisioning example**



Note: When the card is successfully programmed, a popup window message "HSM successfully programmed" appears, and the ST HSM is locked. Otherwise an error message is displayed.

6 SFI image programming by OEMs or CMs

6.1 Secure firmware installation flow

Once the SFI image has been prepared using the STM32 Trusted Package Creator, the OEM sends it to the CMs.

CM production lines need to be equipped with a flash memory programming tool (FT). This tool is used to:

1. Download the SFI image prepared with STM32 trusted package creator, including the image header and the encrypted part of the SFI image.
2. For each STM32 device:
   a. STM32L5, STM32U3, STM32U5, STM32WBA5, STM32WBA6, STM32WL5, and STM32H5: load secure bootloader part II within SRAM.
   b. Request the device certificate from the secure bootloader.
   c. Request a dedicated license from the ST HSM. The ST HSM uses the device certificate to produce a dedicated license for a given firmware to be stored in this particular STM32 device. ST HSM can generate different license types according to the provisioned personalization data file in ST HSM (see Section 5 step 5). The different licenses types are detailed in Section 6.2.
   d. Ask the secure bootloader to process the license, decrypt and flash the SFI image in internal flash memory.
   e. Ask secure bootloader to decrypt and re-encrypt SFI image section dedicated to external flash memory if applicable.
   f. Ask external flash loader to program encrypted external flash section to external flash memory if applicable.
   g. Program STM32 option bytes.

STM32CubeProgrammer is the flash memory programming tool provided by STMicroelectronics. It is available in CLI mode and can be downloaded free of charge from www.st.com.

STM32CubeProgrammer is not certified for production lines, but can be used for testing SFI.

It supports the secure programming of SFI images for:

- STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx microcontrollers via USART, SPI, USB-DFU bootloader or JTAG interfaces.
- STM32H7RS microcontrollers via USART, SPI, I2C, I3C, FDCAN, USB or JTAG.
- STM32L5 and STM32U5 microcontrollers via USART, SPI, I2C, FDCAN, USB or JTAG.
- STM32U3 microcontrollers via USART, FDCAN, I2C, I3C, USB, SPI or JTAG.
- STM32WL5 microcontrollers via USART, SPI or JTAG.
- STM32WBA5 microcontrollers via USART, I2C, SPI or JTAG.
- STM32WBA6 microcontrollers via USART, I2C, USB, SPI or JTAG.
- STM32H5 microcontrollers via USART, FDCAN, I2C, I3C, SPI or JTAG.

STM32CubeProgrammer communicates with the chosen interface that triggers the secure bootloader in order to handle the OEMs' encrypted SFI image during SFI operations.

On STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx only, to start the SFI process, the bootloader activates the security via OB programming, which in turn activates the secure bootloader.

On STM32U5, STM32WBA5, and STM32WBA6, if OEM password keys have been set, it is necessary to reset them to start a SFI or SFIx. The value 0xFFFFFFFF must be written in OEM password keys to reset these keys. On STM32U3, if OEM password keys have been set, it is not possible to run SFI procedure on the sample anymore.

STM32CubeProgrammer example command using USART interface on STM32 devices supporting SFI and with ST HSM smartcard usage:

- sfi command example allowing secure installing of firmware "data.sfi" into STM32 user flash memory:

STM32_Programmer_CLI.exe -c port=COM1 -sfi "C:\SFI\data.sfi" hsm=1 slot=1

More information is available in [UM2237] and [AN5054].

6.2 License types

Here are the different license types generated by the ST HSM depending on the selected personalization data file:

Table 4. SFI licenses

Licenses		Platforms								
type	description	WL5	L5	U3	U5	H5	WBA5	WBA6	H7	H7RS
SFI.	ST HSM license for firmware install	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SMI.	ST HSM license for module install	No	No	No	No	No	No	No	Yes	No
SFIA.	ST HSM license for firmware install without user flash mass erase	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes

Caution: *When selecting SFIA license, it is used to load firmware or data into user's flash memory before launching SFI procedure.*

Note: If SFIA personalization data file is not available under STM32CubeProgrammer, contact ST sales representative.

In order to properly protect customer firmware confidentiality when using SFIA license, SFI installed firmware must control and restrict preloaded firmware or data during OEM firmware boot and execution.

By default, "SFI." license must be used so that user flash is erased or checked during SFI installation.

7 Known limitations

7.1 STM32H75xxx known limitations

SFI limitation

During a SFI involving a firmware which has to be written on last word of a flash memory bank, an error is raised by the secure bootloader and the firmware is not programmed.

Recommendation

If the firmware to program in internal flash memory exceeds the size of one memory bank, it must be splitted in two parts (from linker point of view) with the first part avoiding the last word of the bank.

7.2 STM32U5 known limitations

SFI limitation

For SFI on STM32U575I evaluation board (STM32U575I-EV), SWD frequency must not exceed 8000 kHz.

On the STM32U5G9J-DK2 board SFIx is unavailable.

7.3 STM32WBA5 known limitations

SFI limitation

SFI can only run on samples Revision B.

SFI can only run on products with 128 kB of RAM.

7.4 STM32WBA6 known limitations

SFI limitation

SFI can only run on products with 512kB of RAM.

7.5 STM32U3 known limitations

SFI limitation

OEM password keys cannot be reset once programmed. SFI can only run on sample with no OEM password keys programmed.

8 SFI packages

ST provides the 2 following packages:

- X-CUBE-RSSe
- X-CUBE-SFI

8.1 X-CUBE-RSSe

This package provides all the embedded SW components to run a SFI procedure on each product supporting SFI.

8.2 X-CUBE-SFI

This package provides an SFI example on each product supporting the SFI.

Revision history

Table 5. **Document revision history**

Date	Revision	Changes
20-Dec-2018	1	Initial release.
12-Mar-2019	2	Added Table 1. Applicable products with indication of the STM32L462CEU6F (special order) code.
11-Jun-2019	3	Document is declassified.
11-Sep-2019	4	Restored and updated missing Figure 8. STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx secure bootloader.
07-Oct-2019	5	Updated Table 2. Glossary terms. Updated Section 3.3.1: Secure bootloader overview. Updated Section 3.3.2: User flash memory mapping. Added Section 7: Known limitations.
20-Dec-2019	6	Added support for STM32L5 series: Updated Table 1. Applicable products. Updated Section 1.1: Related documents. Updated Section 1.2: Glossary. Updated Section 2.1: SFI principles overview. Added Section 2.1.2: SFI and external flash memory. Updated Section 2.2: SFI security features. Added Section 3.3: STM32L5, STM32U3, STM32U5, STM32WBA5, STM32WBA6, and STM32H5. Updated Section 4.2: SFI firmware image creation procedure for STM32H7/L5/U3/U5/WBA5/WBA6. Added Section 4.2.2: Both internal and external flash memories. Updated Section 6.1: Secure firmware installation flow.
14-Jan-2020	7	Introduced STM32H7B3xx. Updated Table 1. Applicable products. Updated title of Section 2.1.2: SFI and external flash memory. Updated Section 3.1: STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx. Updated Table 3. Minimum RDP requirements. Updated Section 4.2.1: Internal flash memory only. Updated Section 6.1: Secure firmware installation flow.
26-Aug-2020	8	Introduced STM32H733xx/STM32H735xx. Updated Table 1. Applicable products. Updated Section 1.1: Related documents. Updated title of Section 2.1.2: SFI and external flash memory. Updated Section 3.1: STM32H75xxx/STM32H7B0xx/STM32H7B3xx/STM32H730xx/STM32H733xx/STM32H735xx. Updated Table 3. Minimum RDP requirements. Updated Section 4.2.1: Internal flash memory only. Updated Section 6.1: Secure firmware installation flow.

Date	Revision	Changes
13-Nov-2020	9	Introduced STM32WL5. Updated Table 1. Applicable products. Updated Section 1.1: Related documents. Updated Section 2.1: SFI principles overview. Updated Section 2.2: SFI security features. Updated Section 3.3.1: Secure bootloader overview. Updated Section 3.3.1: Secure bootloader overview. Updated Section 3.3.1: Secure bootloader overview. Updated Section 3.4.1: Secure bootloader overview. Updated Table 3. Minimum RDP requirements. Added Section 3.4: STM32WL5. Updated Section 5: SFI ST HSM key provisioning. Updated Section 6.1: Secure firmware installation flow.
28-Jan-2021	10	All updates areSTM32L462CE related only. Updated 3.3.1: Secure bootloader overview. Updated 3.3.1: Secure bootloader overview. Updated Figure 11. STM32L462CE internal user flash memory mapping with SFI. Updated footnote in Table 3. Minimum RDP requirements. Updated Section 7.2: STM32L462CE known limitations.
07-Sep-2021	11	Introduced STM32U5. Updated Table 1. Applicable products. Updated Section 1.1: Related documents. Updated Section 2.1.2: SFI and external flash memory. Updated 3.3.1: Secure bootloader overview including Figure 10. STM32L462CE secure bootloader. Updated Section 3.3: STM32L5, STM32U3, STM32U5, STM32WBA5, STM32WBA6, and STM32H5. Updated Table 3. Minimum RDP requirements. Updated Section 5: SFI ST HSM key provisioning including Figure 23. ST HSM V1 key provisioning example and new Figure 24. ST HSM V2 key provisioning example. Updated Section 6.1: Secure firmware installation flow. Updated Section 7.2: STM32L462CE known limitations.
10-Mar-2022	12	Updated Table 1. Applicable products. Updated Section 1.1: Related documents. Updated Section 2.1: SFI principles overview. Updated Section 2.1.2.3: External flash memory encryption with secure bootloader and unique key. Updated Section 4: SFI image preparation introduction. Added Figure 12. STM32 Trusted Package Creator - SFI option bytes. Updated Figure 13. STM32 Trusted Package Creator - SFI image creation. Updated Section 4.2.1: Internal flash memory only. Updated Figure 15. SFI image generation tab example. Updated Figure 16. SFI image successful generation. Updated Section 4.2.2: Both internal and external flash memories. Updated Figure 17. SFIx image generation tab example. Updated Figure 18. SFIx image successful generation.

Date	Revision	Changes
28-Jun-2022	13	Added STM32H730xx and STM32H7B0xx in the whole document. Replaced STM32H75xxI by STM32H75xxx in the whole document. Replaced STM32H7B3xI by STM32H7B3xx in the whole document. Updated Section 3.1.4: Secure boot path. Updated Section 4: SFI image preparation introduction. Updated Section 4.2: SFI firmware image creation procedure for STM32H7/L5/U3/U5/WBA5/WBA6. Updated Section 5: SFI ST HSM key provisioning. Updated Section 6.1: Secure firmware installation flow. Added Section 7.2: STM32U5 known limitations.
14-Feb-2023	14	Extended the scope of the document to entire STM32U5 series (instead of only STM32U575/585 lines) with updates in: • Table 1. Applicable products. • [RM0456]. • Section 3.3: STM32L5, STM32U3, STM32U5, STM32WBA5, STM32WBA6, and STM32H5. • Section 3.3.3: External flash memory mapping. • Table 3. Minimum RDP requirements. • Section 6.1: Secure firmware installation flow. • Section 7.2: STM32U5 known limitations.
28-Sep-2023	15	Added STM32U5 in the whole document. Updated document title. Updated: • Section Introduction. • Section 1: Preamble. • Section 2: STM32 secure firmware install (SFI). • Section 3.3.2: User flash memory mapping. • Section 4: SFI image preparation. • Section 7.2: STM32U5 known limitations. Added: • Section 3.3.2.1: No STM32TRUSTEE. • Section 3.3.2.2: STM32TRUSTEE (only on STM32H5). • Section 4.2.3: OEM passwords for RDP regression (STM32U3, STM32U5, STM32WBA5 and STM32WBA6 series only). • Section 4.3: SFI firmware image creation for STM32H5 . • Figure 21. SFI image generation tab for STM32H5 .

Date	Revision	Changes
29-Feb-2024	16	Added STM32H7RS and STM32WBA5 in the whole document. Suppressed STM32L4 devices in the whole document. Updated Table 1. Applicable products. Updated Section 1.1: Related documents. Updated Section 2.1: SFI principles overview. Updated Section 2.1.1: SFI and internal flash memory. Updated Section 2.1.2: SFI and external flash memory. Updated Section 2.2: SFI security features. Updated Section 3.3: STM32L5, STM32U3, STM32U5, STM32WBA5, STM32WBA6, and STM32H5 introduction. Updated Figure 10. STM32 secure bootloader title. Updated Section 3.3.4: Secure boot path. Updated Section 4.2: SFI firmware image creation procedure for STM32H7/L5/U3/U5/WBA5/WBA6. Updated Section 4.3.1: No STM32TRUSTEE. Added Section 4.4: SFI firmware image creation for STM32H7RS. Updated Section 5: SFI ST HSM key provisioning. Updated Section 6: SFI image programming by OEMs or CMs. Added Section 6.2: License types. Added Section 7.3: STM32WBA5 known limitations.
06-Mar-2025	17	Added STM32U3 and STM32WBA6 in the whole document. Replaced HSM by ST HSM in the whole document. Updated Table 1. Applicable products. Updated Section 1.1: Related documents. Updated Section 1.2: Glossary. Updated Section 2.1: SFI principles overview. Updated Figure 1. SFI process overview. Updated Section 2.2: SFI security features. Updated introduction to Section 4.3: SFI firmware image creation for STM32H5 . Updated Figure 10. STM32 secure bootloader. Updated Section 3.3.4: Secure boot path. Updated introduction to Section 4.2: SFI firmware image creation procedure for STM32H7/L5/U3/U5/WBA5/WBA6. Updated Table 3. Minimum RDP requirements. Updated Section 4.2.3: OEM passwords for RDP regression (STM32U3, STM32U5, STM32WBA5 and STM32WBA6 series only). Added Section 4.2.3.1: KDF256 details. Added Section 4.2.3.2: OEM passwords diversification example on STM32U5. Updated intorduction to Section 4.3.1: No STM32TRUSTEE. Updated Section 6.1: Secure firmware installation flow. Updated Section 6.2: License types. Updated Table 4. SFI licenses. Added Section 7.4: STM32WBA6 known limitations. Added Section 7.5: STM32U3 known limitations. Added Section 8: SFI packages.

Contents

List of tables

List of figures