AN4992
Application note

Overview secure firmware install (SFI)

Introduction

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, the Secure Firmware Install (SFI), permitting to program OEM firmware into STM32 internal Flash memory in a secure way (with confidentiality, authentication and integrity checks).

SFI solution is introduced on the microcontrollers listed in *Table 1*.

STM32 Series supports protection mechanisms permitting to protect 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	Order code
Microcontrollers	STM32H75xxI	All order codes supported (refer to datasheet ordering information section).
	STM32H7B3xI	
	STM32L462CE	STM32L462CEU6F[1] Referred hereafter as STM32L462CE.
	STM32L5xxxx	Entire STM32L5 Series. Referred hereafter as STM32L5.
	STM32H733xx	All order codes supported (refer to datasheets ordering information section).
	STM32H735xx	
	STM32WL5xxx	Entire STM32WL5x line. Referred hereafter as STM32WL5.

1. This is the only supported order code. This code is not listed in the datasheet ordering section. Contact ST sales representative (special order).

Contents

List of figures

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

[AN3156] USB DFU protocol used in the STM32 bootloader

[AN4286] SPI protocol used in the STM32 bootloader

[AN4221] I2C protocol used in the STM32 bootloader

[AN3154] CAN protocol used in the STM32 bootloader

[AN5054] Secure programming using STM32CubeProgrammer

[RM0394] STM32L41xxx/42xxx/43xxx/44xxx/45xxx/46xxx advanced Arm®-based 32-bit MCUs[a]

[RM0433] STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs[a]

[RM0438] STM32L552xx and STM32L562xx advanced Arm®-based 32-bit MCUs[a]

[RM0455] STM32H7A3/7B3 advanced Arm®-based 32-bit MCUs[a]

[RM0468] STM32H723/733, STM32H725/735 advanced Arm®-based 32-bit MCUs[a]

[RM0453] STM32WL5x advanced Arm®-based 32-bit MCUs with sub-GHz radio solution[a]

[UM2237] STM32CubeProgrammer software description

[UM2238] STM32 Trusted Package Creator tool software description

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.*

[a] 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
HSM	Hardware security module
MAC	Message authentication code
MCU	Microcontroller unit
OB	Option bytes
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]*)
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 in untrusted production environment (such as OEM contract manufacturer). SFI is implemented in a secure bootloader.

The SFI process prevents the OEM firmware code from:

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

This mechanism consists in having the whole OEM firmware and the option bytes encrypted with an AES secret key, thanks to STM32 Trusted Package Creator tool(1), during OEM firmware development.

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

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

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

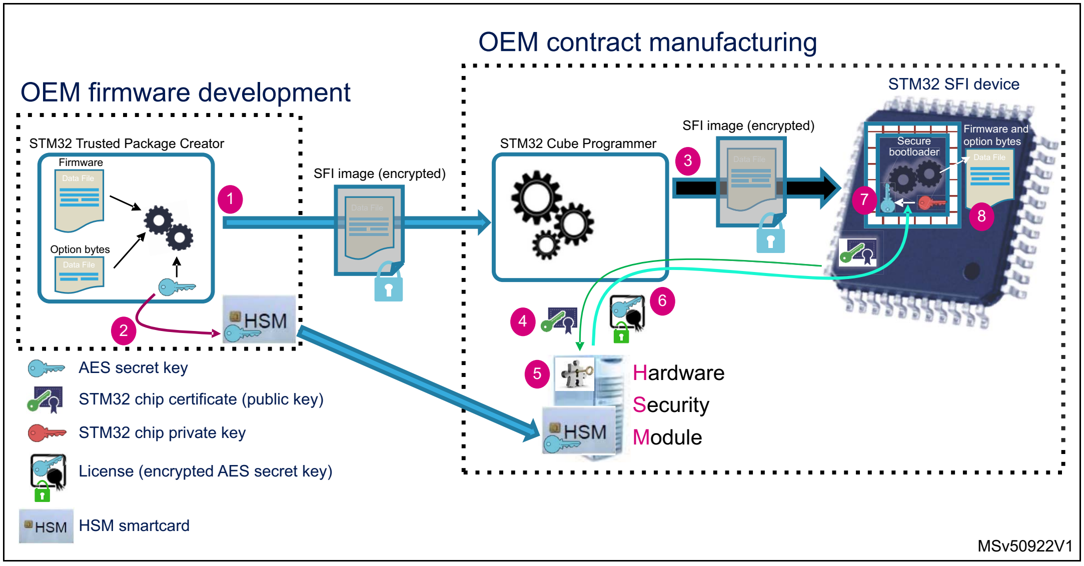
- Securely storing OEM AES secret key
- Checking STM32 device certificate(4) that is used to authenticate STM32 device(5)
- Generating and providing the license(6) to the secure bootloader to securely install the encrypted firmware on STM32 device.
- Counting number of produced STM32 devices.

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, STM32 support secure OEM firmware programming in internal Flash memory and ensure OEM firmware protection (confidentiality, authenticity and integrity) during OEM CM manufacturing stage. The secure firmware install solution securely receives and decrypts the firmware 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.

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 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. Firmware 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 applied to STM32L562xx and STM32H7B3xI/STM32H733xx/STM32H735xx (with OTFDEC)

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 and enhanced 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.

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.

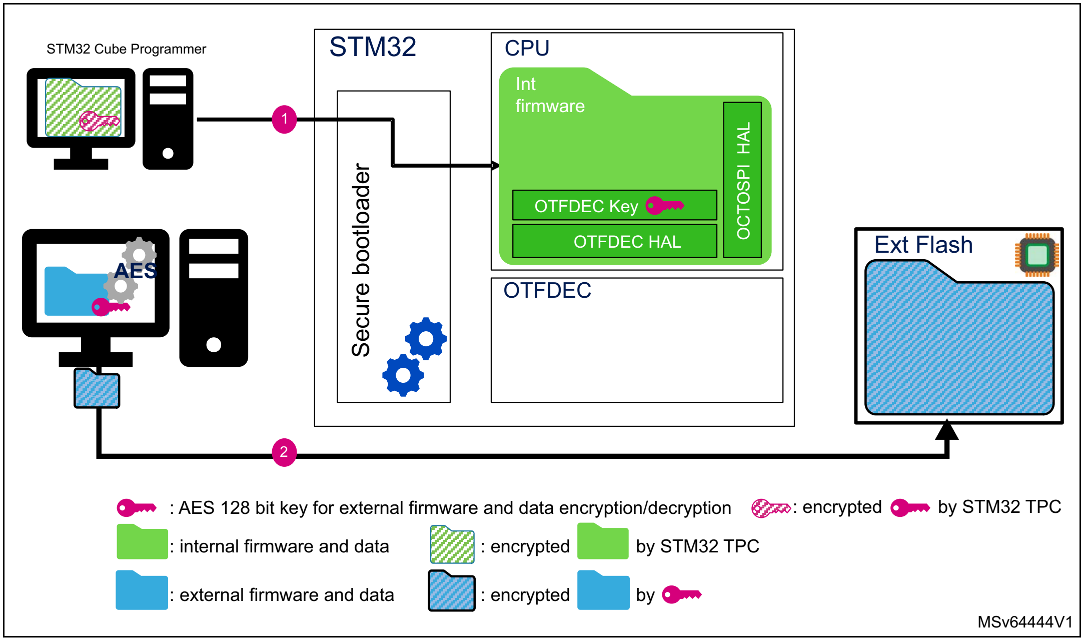
Since OEM programs external Flash memory on host, OEM must not encrypt external firmware and data thanks to STM32 Trusted Package Creator. However, OEM must send the external firmware and data AES encryption keys within the SFI image. Then, when building the SFI image thanks to STM32 Trusted Package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

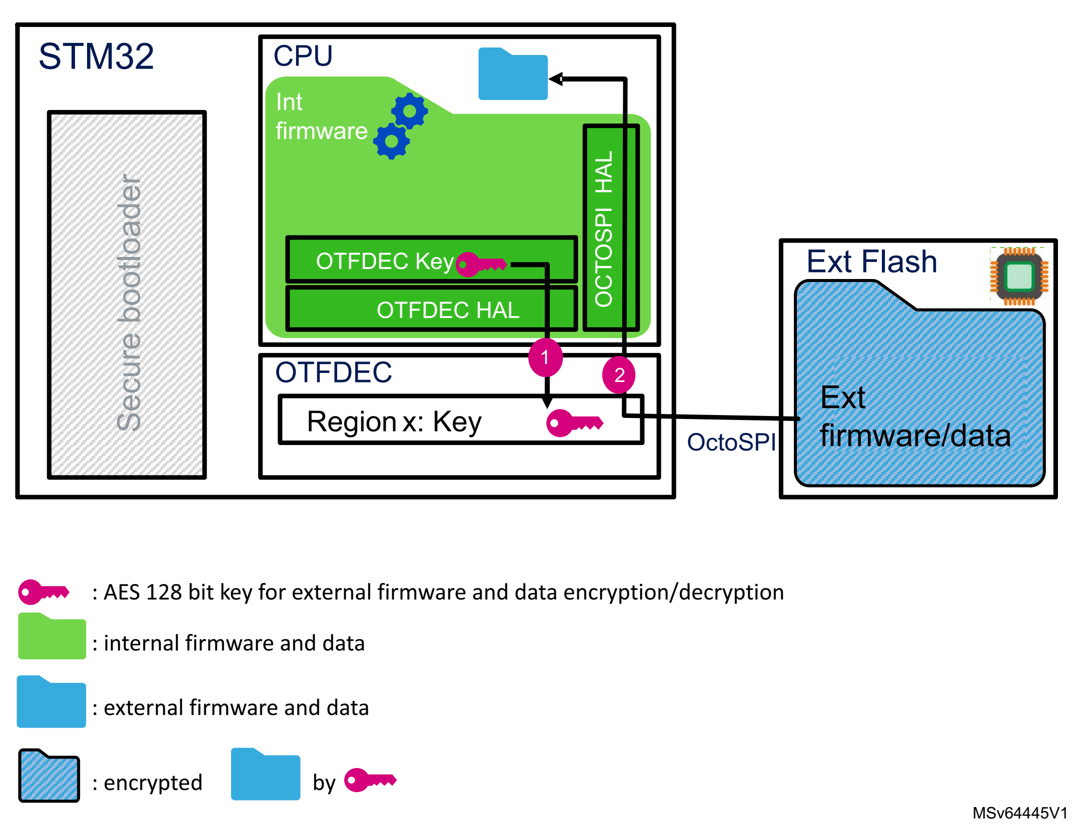
Figure 2. SFI and external Flash memory encryption without secure bootloader



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)].

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).

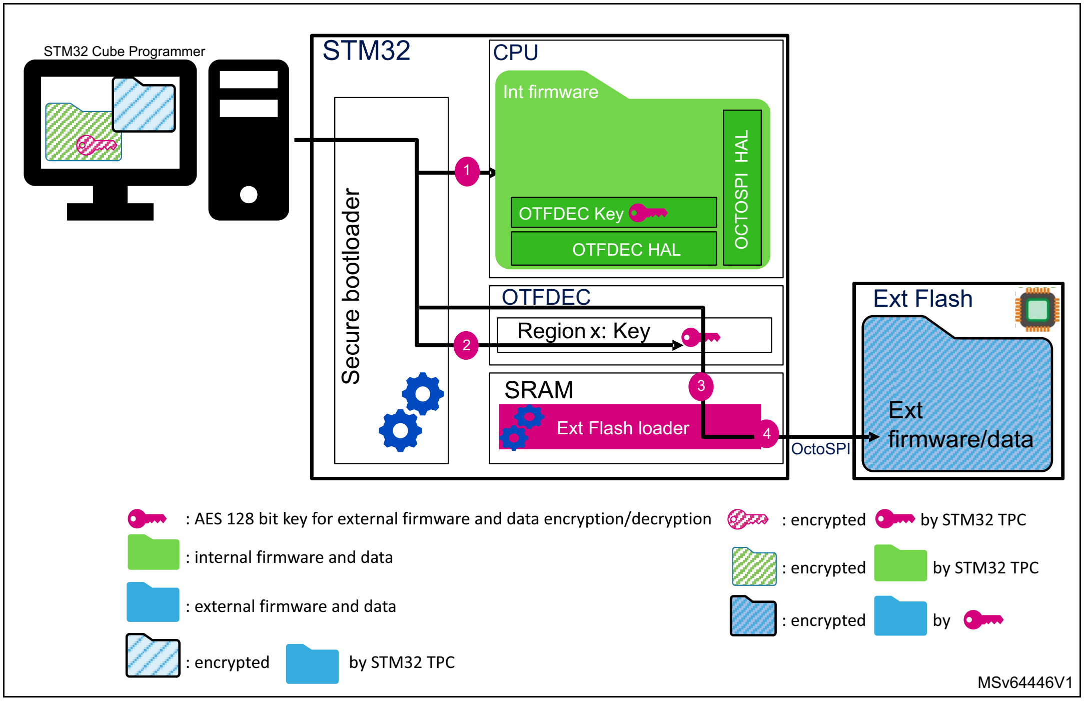
OEM must create SFI image with:

- 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

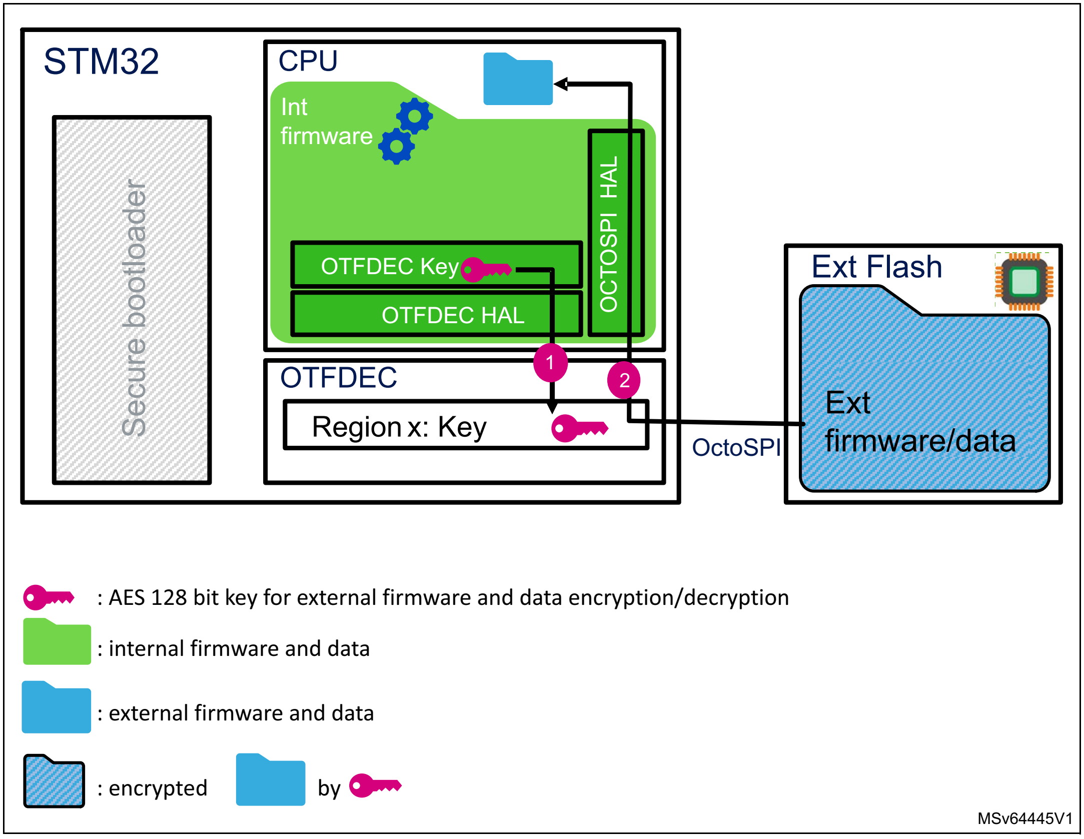


1. Internal Flash memory programming.
2. AES external firmware and data key programming in OTFDEC peripheral.
3. External Flash memory chunk encryption.
4. External Flash memory programming.

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 8 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].

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 STM32CubeProgrammer performs secure programming of the internal Flash memory with this last key[3].

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).

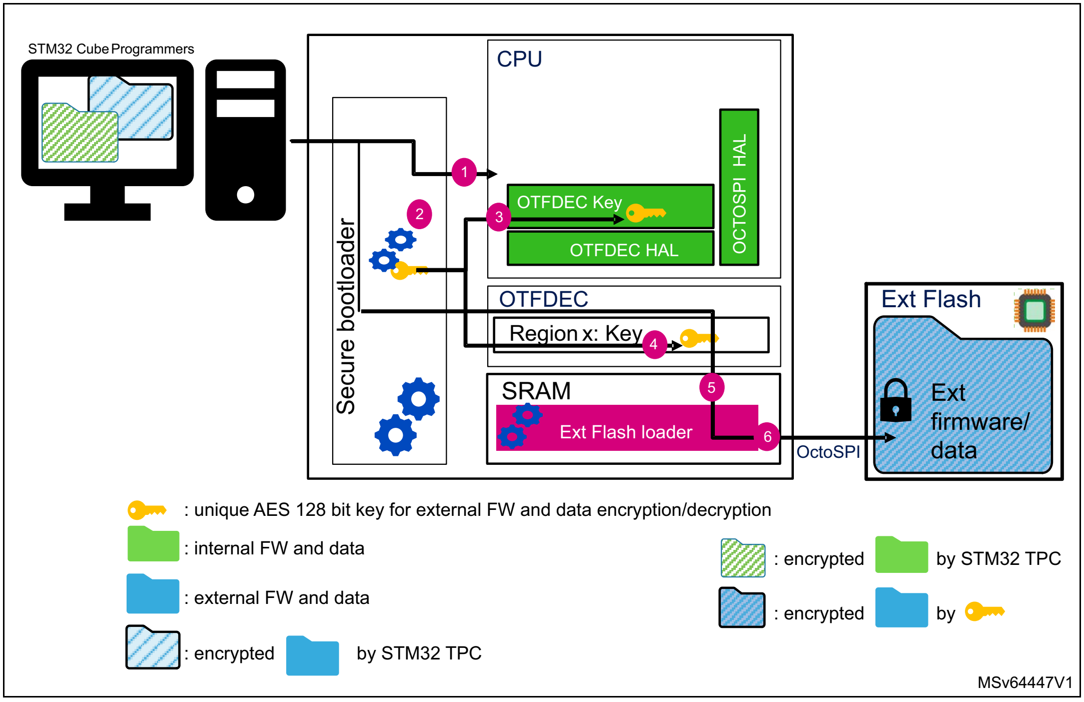
OEM must create the SFI image with:

- Internal firmware and data (include external Flash memory drivers).
- External firmware and data 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.

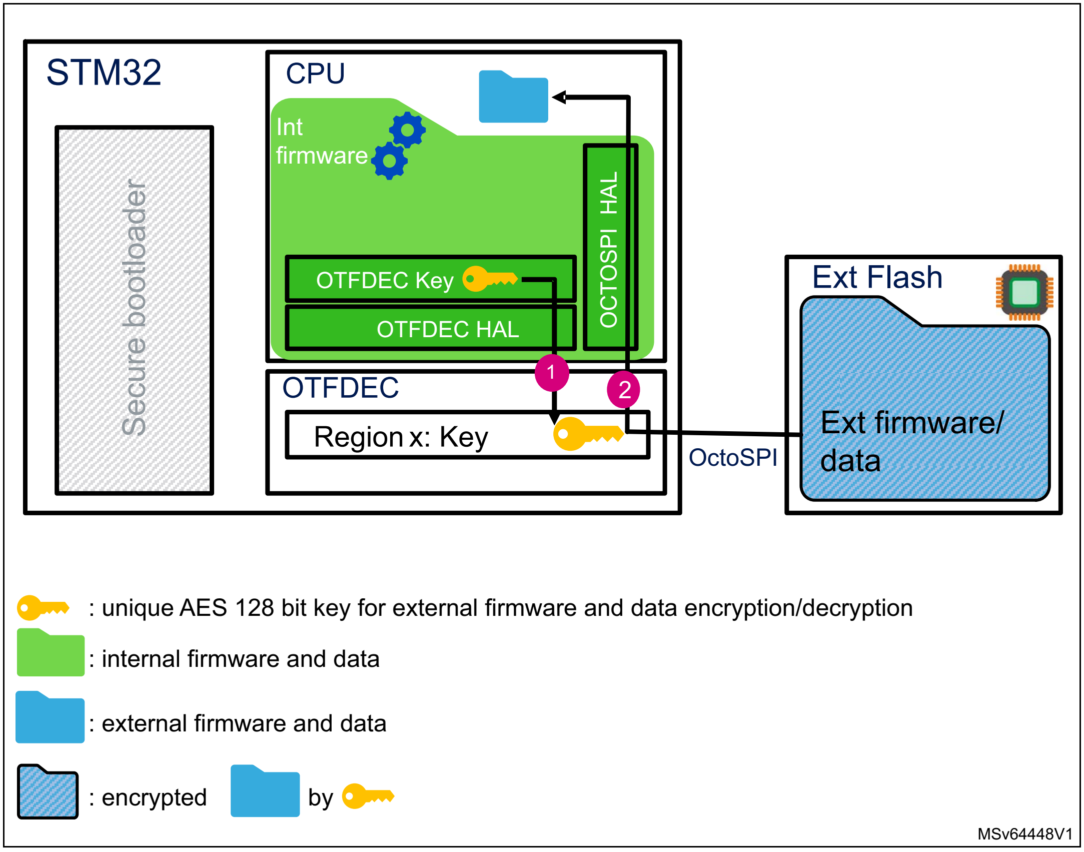
Figure 6. External Flash memory encryption with secure bootloader and unique AES Key



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 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.

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 STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx

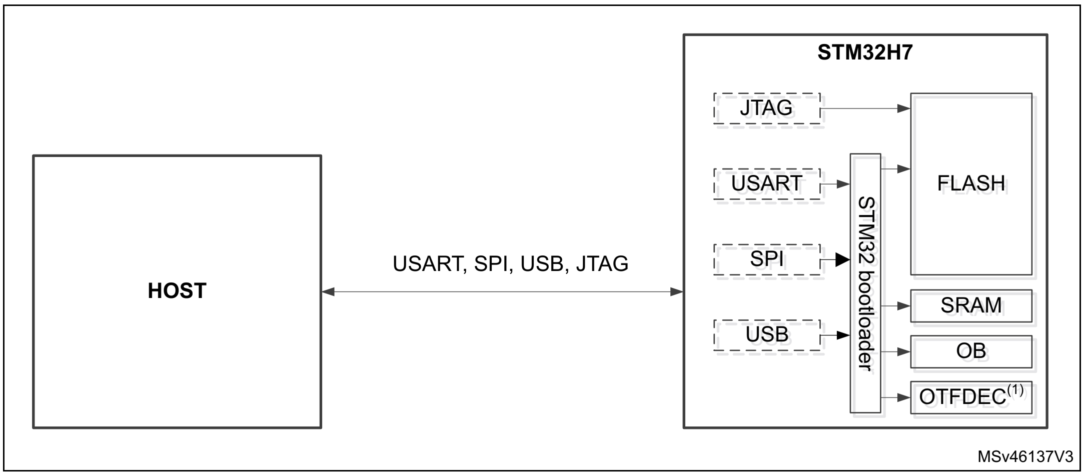
3.1.1 Secure bootloader overview

On the STM32H75xxI/STM32H7B3xI/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 STM32H75xxI/STM32H7B3xI/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. STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx secure bootloader



1. OTFDEC peripheral is only available on STM32H7B3xI/STM32H733xx/STM32H735xx devices.

3.1.2 User Flash memory mapping

On STM32H75xxI/STM32H7B3xI/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 STM32H7B3xI/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 STM32H7B3xI/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 STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx always boot in the secure area closest to the configured boot address.

More information is available in *[RM0433]* for STM32H75xxI.

More information is available in *[RM0455]* for STM32H7B3xI.

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

3.2 STM32L462CE

3.2.1 Secure bootloader overview

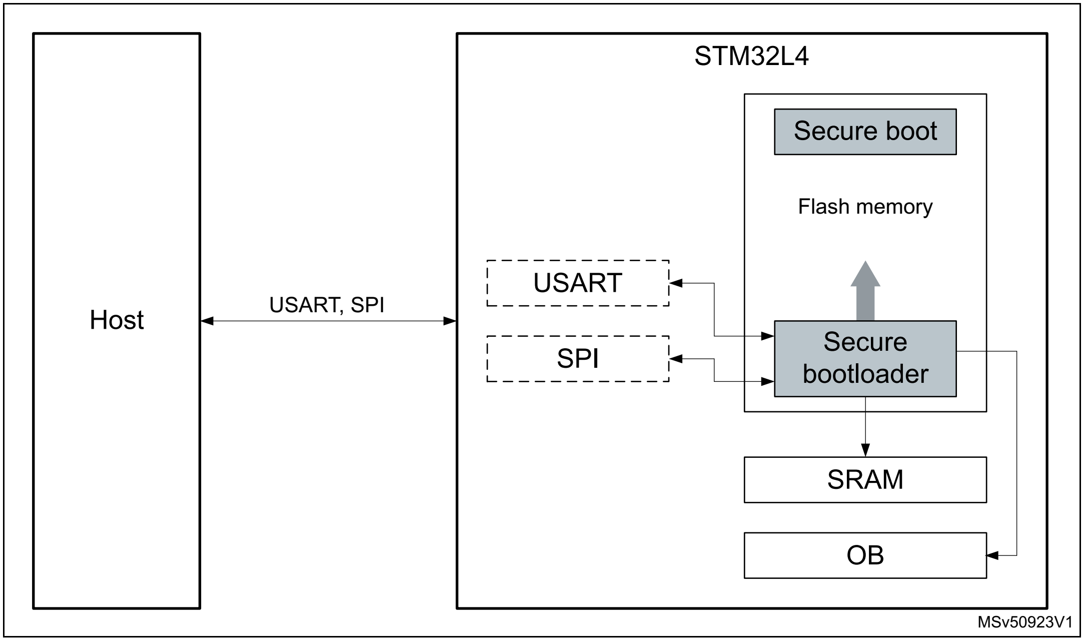
On the STM32L462CE[a], the secure bootloader is stored in internal user Flash memory and supports USART and SPI serial interfaces.

After successful SFI process (no error detected during firmware and OB installation), the secure bootloader is erased from user Flash memory and the SFI cannot be launched again afterwards. In case of any error during SFI process (authentication, integrity, power down,...), it is possible to run again SFI after an automatic cleaning process.

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

[a] The only supported order code is STM32L462CEU6F. This code is not listed in the datasheet ordering section. Contact ST sales representative (special order).

Figure 9. STM32L462CE secure bootloader



3.2.2 User Flash memory mapping

The secure bootloader on STM32L462CE[a] uses the following part of the internal user Flash memory which is protected in RDP level1:

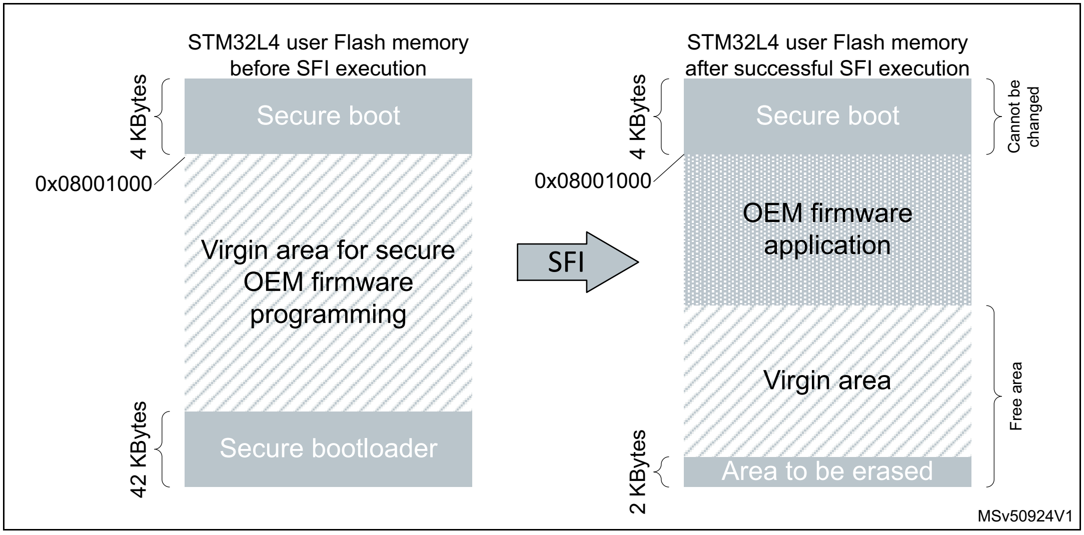
- 4 Kbytes at the beginning of the user Flash memory (secure boot)
- 42 Kbytes at the end of the user Flash memory (secure bootloader)

The STM32L462CE secure boot area is a non-modifiable code area which is executed on Reset. It checks the security configuration (RDP and WRP on secure boot) and that no error occurs during SFI installation.

After successful SFI execution, the last 42 Kbytes of the Flash memory are available for OEM application (40 Kbytes erased by SFI process and last 2 Kbytes can be erased by OEM firmware). Only the first 4 Kbytes (secure boot) are kept in internal user Flash memory and cannot be changed, as a consequence the OEM firmware must be built with an offset of 4 Kbytes (starting address 0x08001000).

[a] The only supported order code is STM32L462CEU6F. This code is not listed in the datasheet ordering section. Contact ST sales representative (special order).

Figure 10. STM32L462CE internal user Flash memory mapping with SFI



3.2.3 Secure boot path

In order to always boot in user Flash memory, it is highly recommended to keep the default option bytes configuration with nSWBOOT0 set to 0 and nBOOT0 set to 1.

More information is available in [RM0394].

3.3 STM32L5

The STM32L5 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 STM32L5 Series, the secure bootloader is split in two parts.

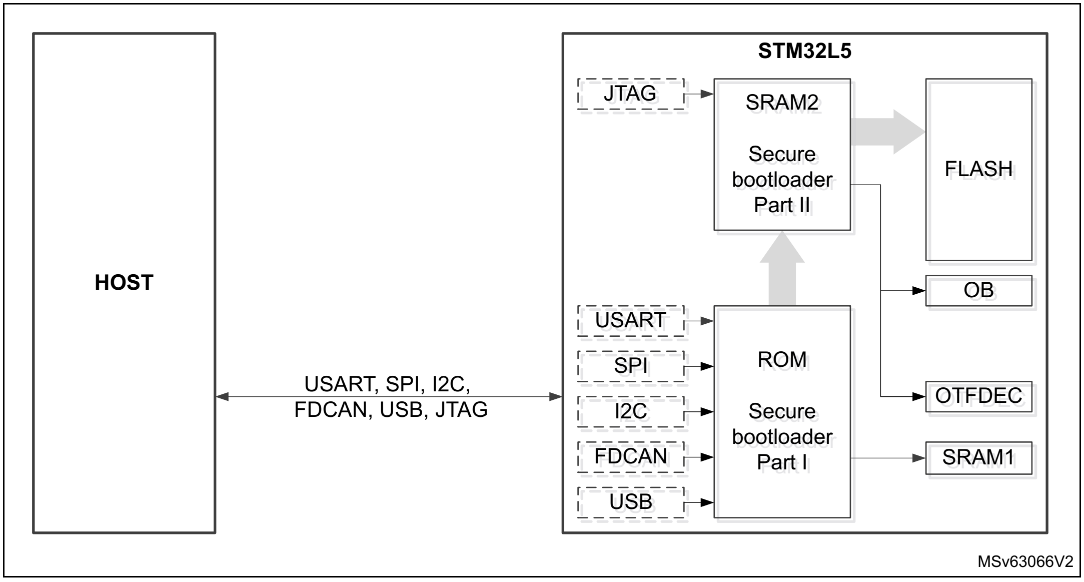
The first part is stored in boot ROM (system memory). It is responsible for loading the second part of the bootloader in SRAM2 through the supported COM ports. The supported COM ports are USART, SPI, I2C, FDCAN, USB and JTAG.

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 STM32L5 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 11. STM32L5 secure bootloader


HOST
USART, SPI, I2C, FDCAN, USB, JTAG
STM32L5
JTAG
SRAM2 Secure bootloader Part II
FLASH
OB
USART
SPI
I2C
FDCAN
USB
ROM Secure bootloader Part I
OTFDEC
SRAM1
MSv63066V2

3.3.2 User Flash memory mapping

On STM32L5 Series, 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.

3.3.3 External Flash memory mapping

As explained before, the STM32L5 can decrypt external firmware and data on-the-fly. This can be done if they reside on external Flash memory connected to the STM32L5 through the OCTOSPI interface. Consequently, the external firmware must be built with addresses within the range 0x90000000 to 0x9FFFFFFF.

The STM32L5 OCTOSPI peripheral can drive external Flash up to 256 Mbytes. The external Flash available memory is dependent from the external Flash connected by user to the STM32L5.

3.3.4 Secure boot path

In order to get STM32L5 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]*.

3.4 STM32WL5

3.4.1 Secure bootloader overview

The secure bootloader is split in two parts.

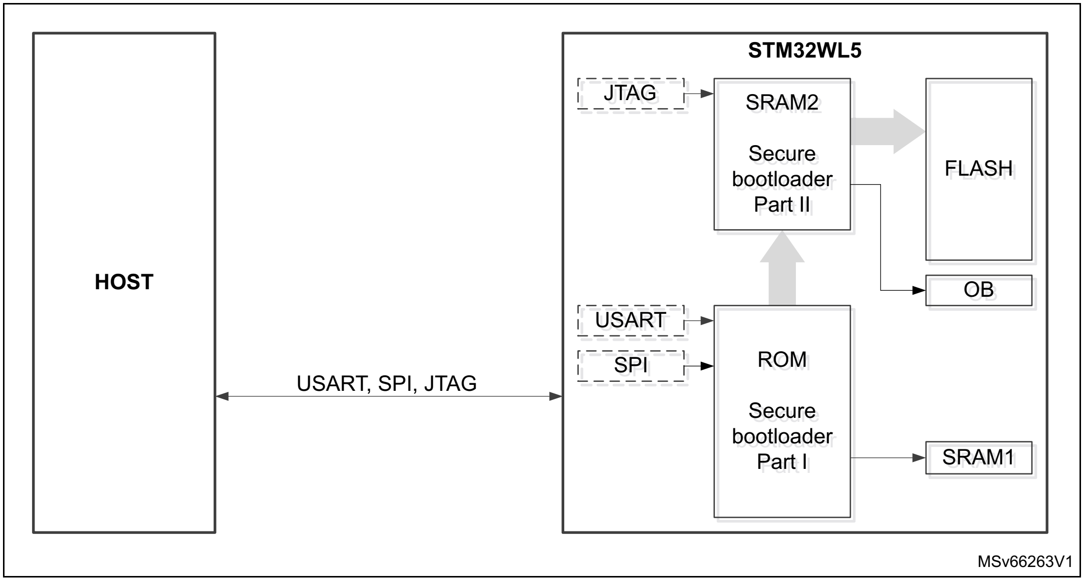
The first part is stored in boot ROM (system memory). It is responsible for loading the second part of the bootloader in SRAM2 through the supported COM ports. The supported COM ports are USART, SPI and JTAG.

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 12. 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.

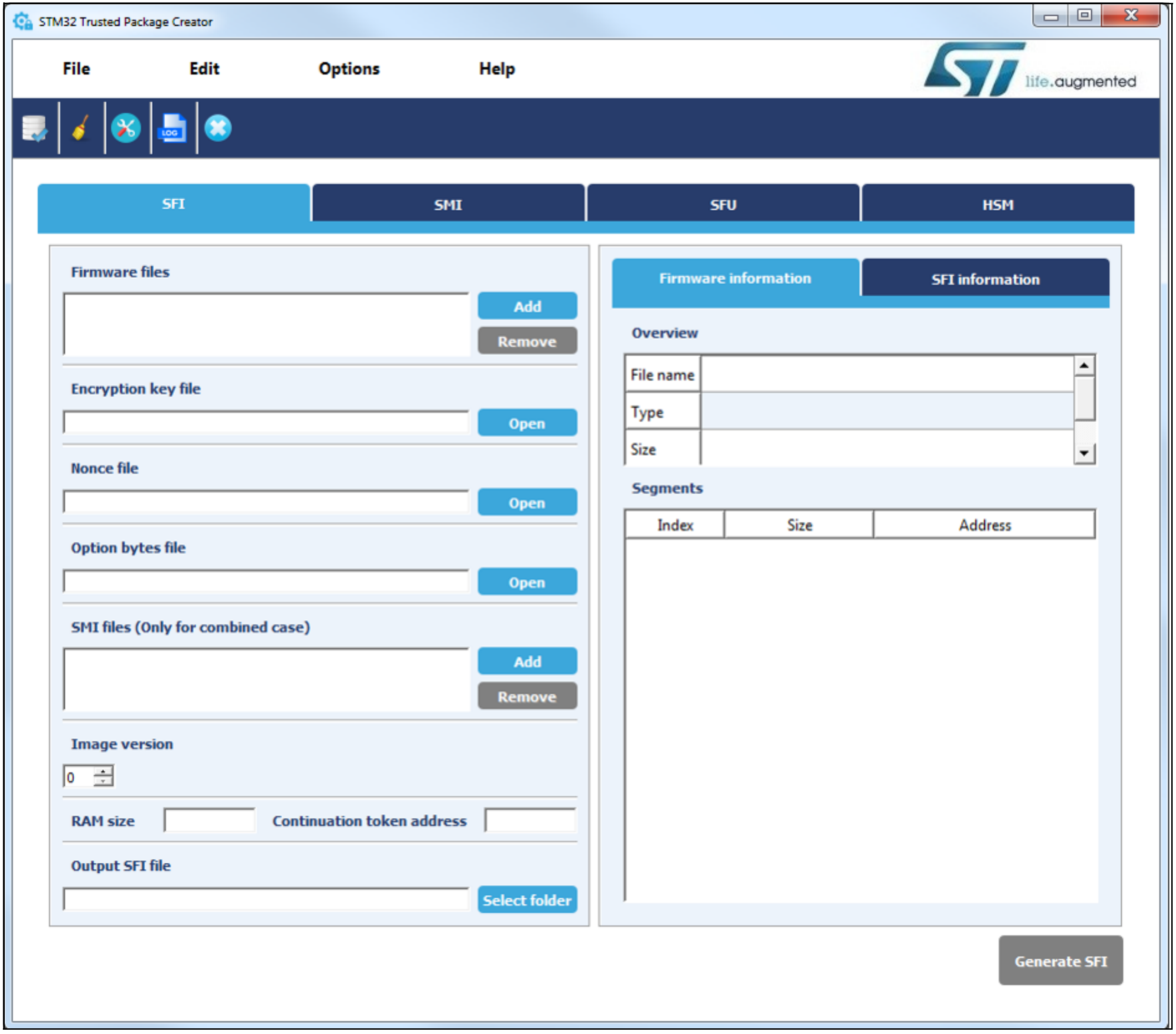
More information is available in *[RM0453]*.

4 SFI image preparation

The SFI image can be prepared by using STMicroelectronics STM32 Trusted Package Creator[a] 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.

It allows the generation of SFI images for STM32 microcontrollers.

Figure 13. STM32 Trusted Package Creator



[a] *STM32 Trusted Package Creator tool is included inside STM32CubeProgrammer tool package (STM32CubeProg) available on www.st.com..*

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

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 plan to apply on the STM32 he programs thanks to SFI. Among option bytes, the user must define the intended RDP value 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.

Table 3. Minimum RDP requirements

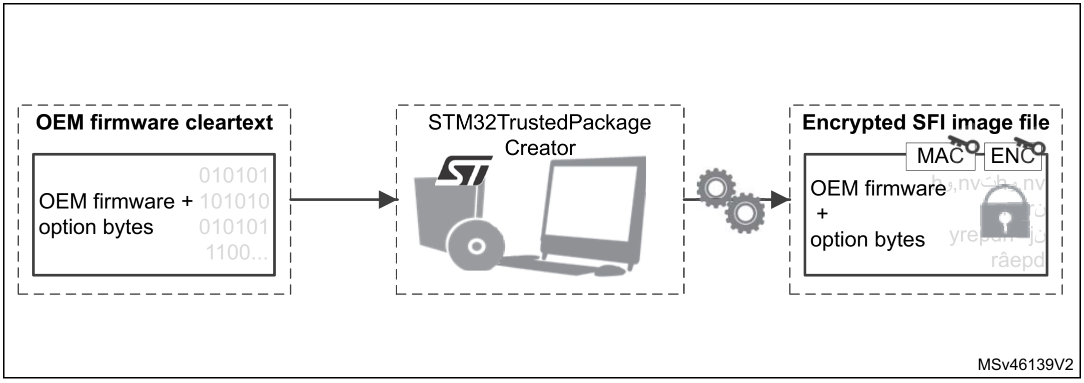
Microcontroller	Use Case	Min RDP level
STM32H75xxI	SFI on internal Flash memory	1
STM32H7B3xI	SFI on internal Flash memory and external Flash memories	1
STM32H733xx/ STM32H735xx	SFI on internal Flash memory and external Flash memories	1
STM32L462CE	SFI on internal Flash memory	1
STM32L5	SFI on internal Flash memory Protection of secure internal Flash memory only	0.5
	SFI on internal Flash memory Protection of secure and non-secure internal Flash memory only	1
	SFI on internal and external[1] Flash memories Protection on both memories	1
STM32WL5	SFI on internal Flash memory Protection of secure internal Flash memory only	0[2]
	SFI on internal Flash memory Protection of secure and non-secure internal Flash memory only	1

1. Applicable to STM32L562xx only (with OTFDEC).
2. With DDS = 1 and FSD = 0.

3. Generate the binary image(s). A binary image does not need to be contiguous (i.e. it can cover multiple disjoint areas).
4. On STM32L462CE, binary image(s) must start with an offset of 4 Kbytes (starting address 0x08001000) from the beginning of the internal Flash memory and must not use the last 42 Kbytes of the internal user Flash memory.

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

Figure 14. SFI image preparation procedure

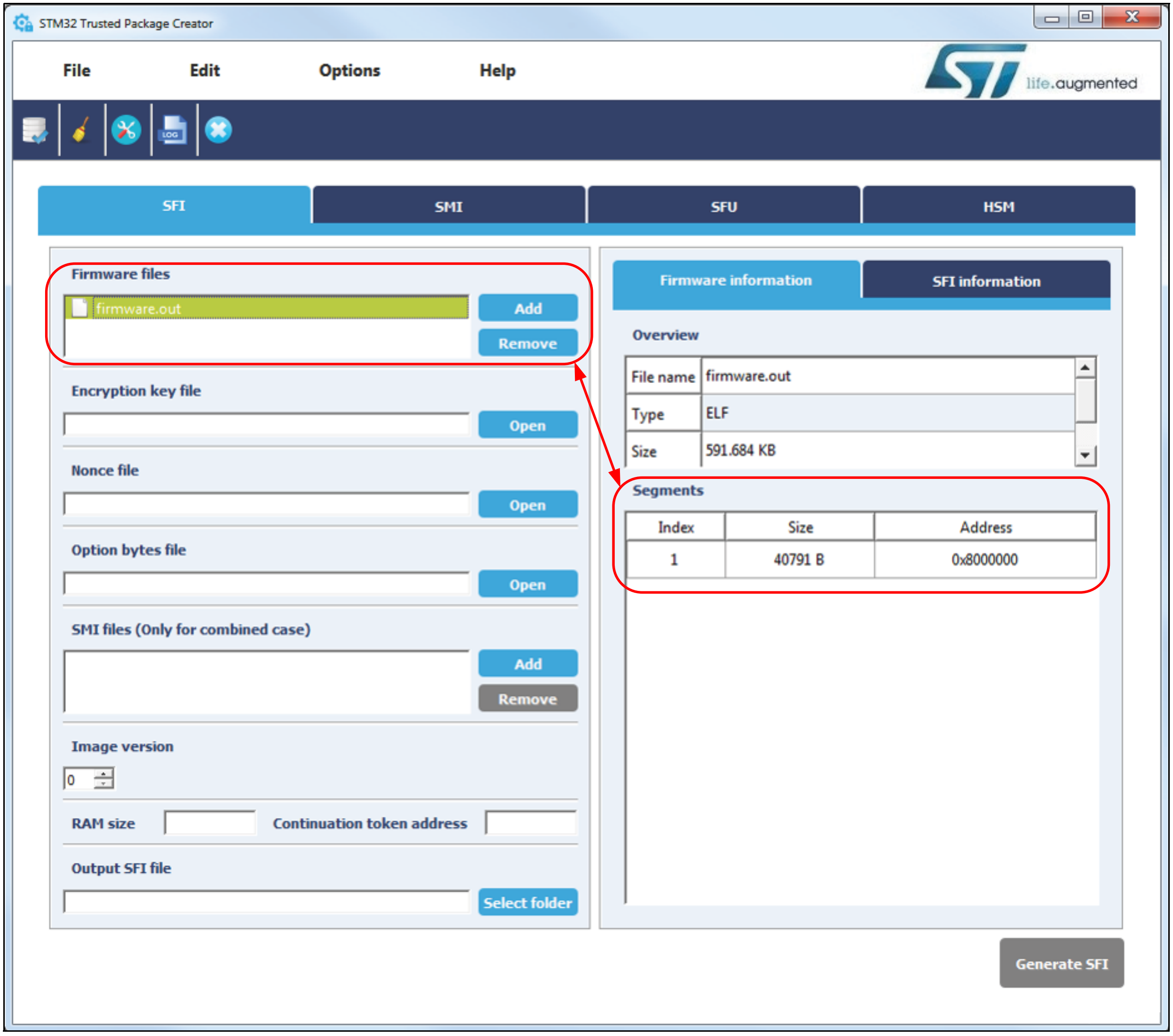


[a] *STM32 Trusted Package Creator tool is included inside STM32CubeProgrammer tool package (STM32CubeProg) available on www.st.com.*

4.2.1 Internal Flash memory only

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

Figure 15. SFI image generation tab example



Firmware files: firmware.out

Encryption key file:

Nonce file:

Option bytes file:

SMI files (Only for combined case):

Image version: 0

RAM size: Continuation token address:

Output SFI file:

Overview

File name	firmware.out
Type	ELF
Size	591.684 KB

Segments

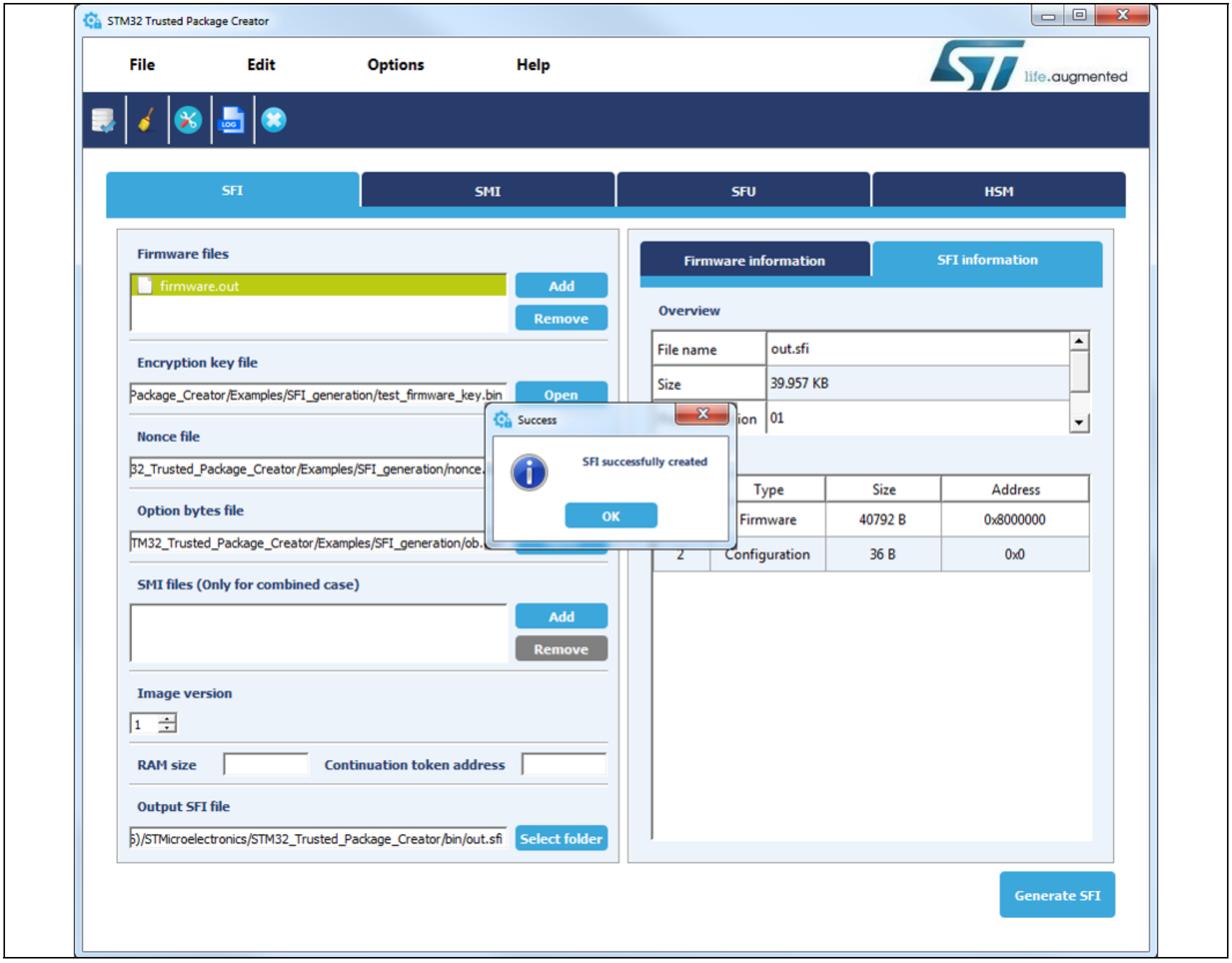
Index	Size	Address
1	40791 B	0x8000000

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.
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 the available RAM size for the SFI operation. This field is not relevant for STM32L462CE, other products can use 0x19000 (100 KB) as a nominal value.
8. Select the address where the continuation token is going to be stored. This field is only relevant for STM32H75xxI/STM32H7B3xI products, 0x081FF000 can be used as a nominal value.
9. For STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx only, select SMI files, if any (optional field).
10. Select the **Output** file folder path where the SFI image, *out.sfi*, is going to be generated.
11. 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 STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx products.

Figure 16. SFI image successful generation



Firmware files: firmware.out

Encryption key file: Package_Creator/Examples/SFI_generation/test_firmware_key.bin

Nonce file: 32_Trusted_Package_Creator/Examples/SFI_generation/nonce.

Option bytes file: TM32_Trusted_Package_Creator/Examples/SFI_generation/ob.

SMI files (Only for combined case)

Image version: 1

RAM size

Continuation token address

Output SFI file: 5)/STMicroelectronics/STM32_Trusted_Package_Creator/bin/out.sfi

Overview

File name	out.sfi
Size	39.957 KB
ion	01

	Type	Size	Address
	Firmware	40792 B	0x8000000
2	Configuration	36 B	0x0

Success

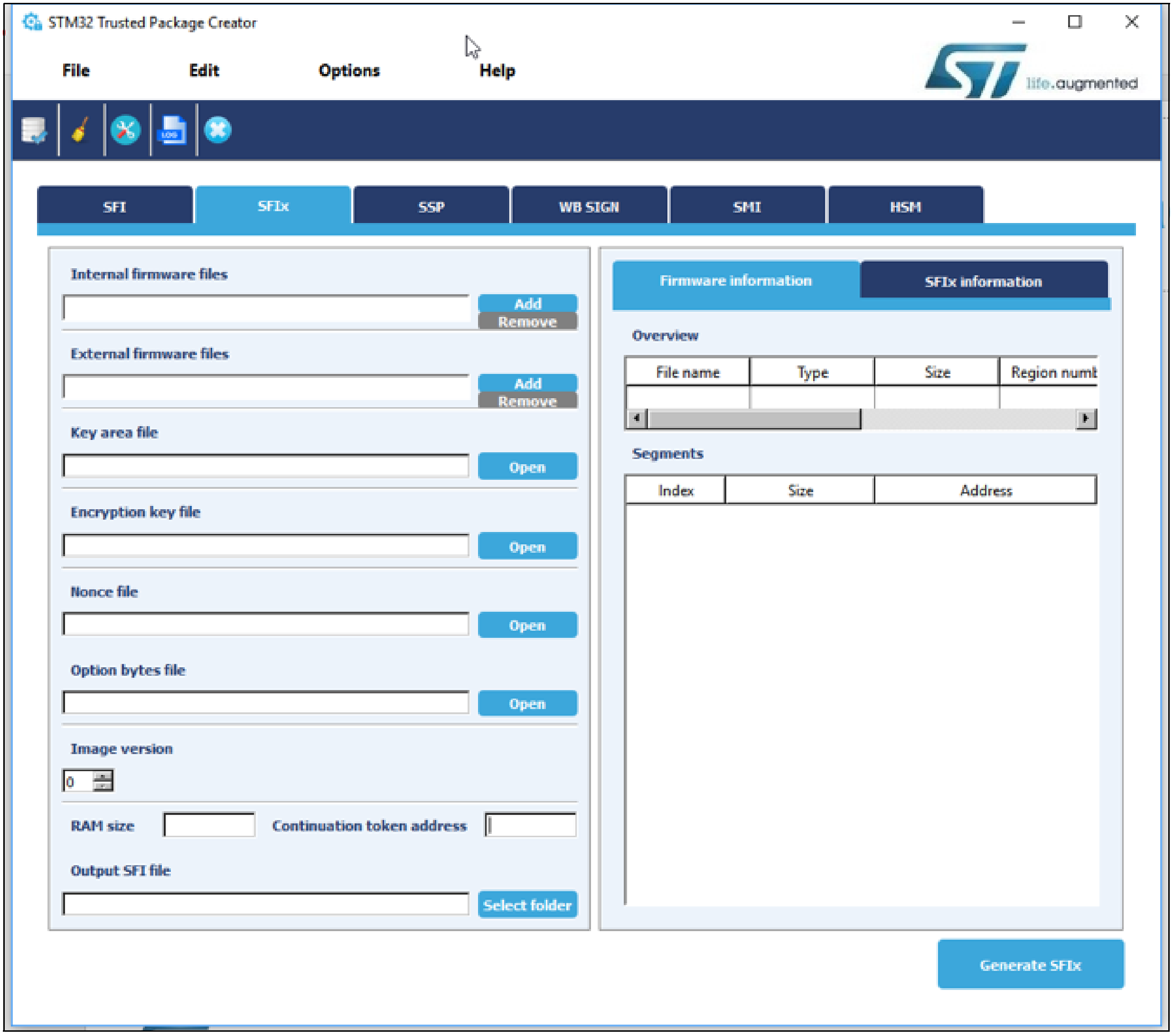
SFI successfully created

OK

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 and fill in the interface fields with valid values:

Figure 17. SFIx image generation tab example



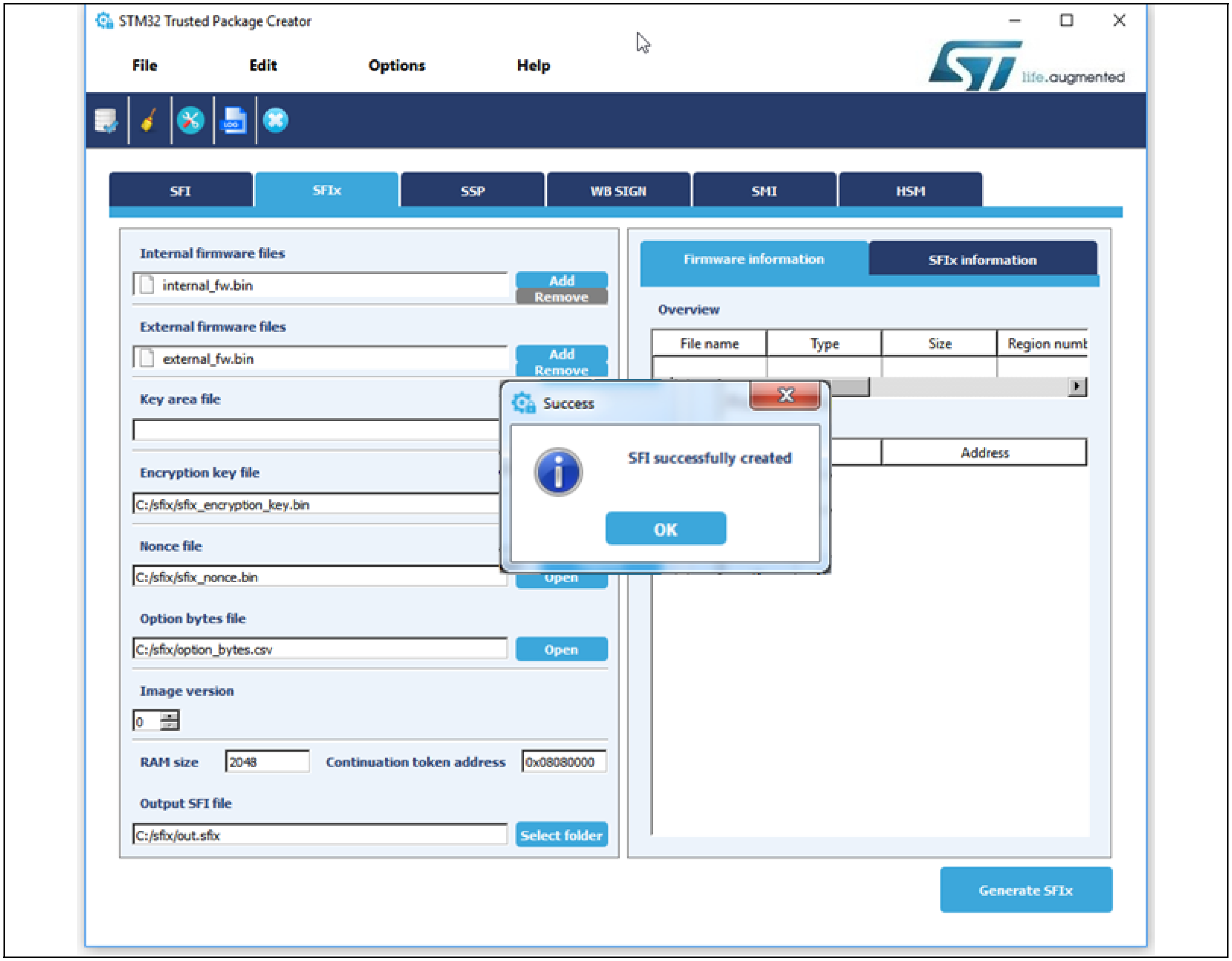
1. Add the internal firmware file(s) using the **Add** button located within the **Internal firmware files** area appended to the input firmware files list. If the file is in bin format, the destination address has to be filled. If OEM uses global AES external Flash memory keys, then internal firmware 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 **Add** button (this is optional). Key area requests STM32 to randomly generate unique AES key set at a specified address filled by user within key area file at *.csv* 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.
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. Enter the **RAM size** that can be used for SFIx process and the address in internal flash memory where **Continuation tokens** can be stored.
9. Select the **Output** file folder path where the SFI image, *out.sfi*, is going to be generated.
10. Generate the SFIx image file by clicking the “**Generate SFIx**” button. If all the fields are filled properly, the “SFI created successfully” message is displayed.

More information is available in *[UM2238]*.

Figure 18. SFIx image successful generation



STM32 Trusted Package Creator

Internal firmware files: internal_fw.bin

External firmware files: external_fw.bin

Key area file:

Encryption key file: C:/sfix/sfix_encryption_key.bin

Nonce file: C:/sfix/sfix_nonce.bin

Option bytes file: C:/sfix/option_bytes.csv

Image version: 0

RAM size: 2048

Continuation token address: 0x08080000

Output SFI file: C:/sfix/out.sfix

Success

SFI successfully created

5 SFI HSM key provisioning

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

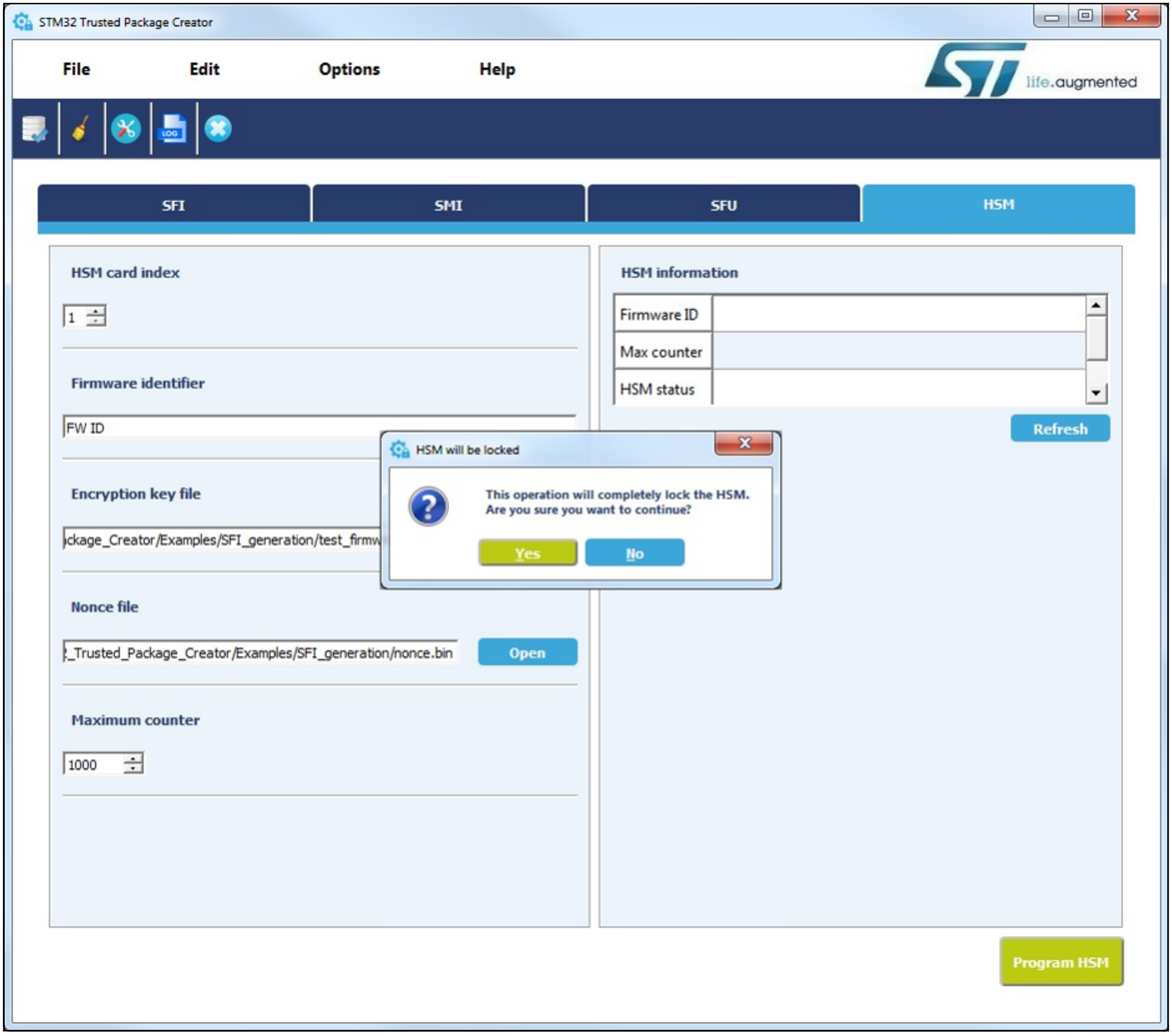
Note: HSM configuration can be done only once, the HSM is locked after successful programming.

1. Insert a virgin HSM in smartcard reader.
2. Select **HSM** tab of STM32 Trusted Package Creator.
3. Add a firmware identifier (allows OEM to identify the correct 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.
5. Configure the maximum installation counter.
6. Configure HSM by clicking the **Program HSM** button. A feedback window indicating that 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, HSM can be used to secure install protected firmware on **counter** number of STM32 devices.

[a] Contact ST sales representative for HSM ordering information.

Figure 19. HSM key provisionning



HSM card index: 1

Firmware identifier: FW ID

Encryption key file: ckage_Creator/Examples/SFI_generation/test_firmw

Nonce file: :_Trusted_Package_Creator/Examples/SFI_generation/nonce.bin

Maximum counter: 1000

HSM information

Firmware ID	
Max counter	
HSM status	

HSM will be locked

This operation will completely lock the HSM.
Are you sure you want to continue?

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 and STM32WL5: load secure bootloader part II within SRAM2.
   b) Request the device certificate from the secure bootloader.
   c) Request a dedicated license from the HSM. The HSM uses the device certificate to produce a dedicated license for a given firmware to be stored in this particular STM32 device.
   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 can be used on OEM-CM production lines.

It supports the secure programming of SFI images for:

- STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx microcontrollers via USART, SPI, USB-DFU bootloader or JTAG interfaces.
- STM32L462CE microcontrollers via USART or SPI interfaces.
- STM32L5 microcontrollers via USART, SPI, I2C, FDCAN, USB or JTAG.
- STM32WL5 microcontrollers via USART, 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 STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx only, to start the SFI process, the bootloader activates the security via OB programming, which in turn activates the secure bootloader.

STM32CubeProgrammer example command using USART interface on STM32 devices supporting SFI and with 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 protocol=static
"C:\SFI\data.sfi" hsm=1 slot=1
```

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

7 Known limitations

7.1 STM32H75xxI 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 STM32L462CE known limitations

On STM32L462CE, the installed firmware size must be a multiple of 4 bytes and if greater than 64 Kbytes it must not be a multiple of 256 bytes to avoid any installation issue (padding bytes must be added in the binary to take into account these limitations).

8 Revision history

Table 4. 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: STM32H75xxI/STM32H7B3xI/STM32H733xx/STM32H735xx secure bootloader*.
07-Oct-2019	5	Updated *Table 2: Glossary terms*. Updated *Section 3.2.1: Secure bootloader overview*. Updated *Section 3.2.2: 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 applied to STM32L562xx and STM32H7B3xI/STM32H733xx/STM32H735xx (with OTFDEC)*. Updated *Section 2.2: SFI security features*. Added *Section 3.3: STM32L5*. Updated *Section 4.2: SFI firmware image creation procedure*. Added *Section 4.2.2: Both internal and external Flash memories*. Updated *Section 6.1: Secure firmware installation flow*.
14-Jan-2020	7	Introduced STM32H7B3xI. Updated *Table 1: Applicable products*. Updated title of *Section 2.1.2: SFI and external Flash memory applied to STM32L562xx and STM32H7B3xI/STM32H733xx/STM32H735xx (with OTFDEC)*. Updated *Section 3.1: STM32H75xxI/STM32H7B3xI/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*.

Table 4. Document revision history (continued)

Date	Revision	Changes
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 applied to STM32L562xx and STM32H7B3xI/STM32H733xx/STM32H735xx (with OTFDEC)*. Updated *Section 3.1: STM32H75xxI/STM32H7B3xI/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*.
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 3.1.1: Secure bootloader overview*. Updated *Section 3.2.1: 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 HSM key provisioning*. Updated *Section 6.1: Secure firmware installation flow*.