

---

## STM32 Trusted Package Creator tool software description

---

### Introduction

STM32 Trusted Package Creator is part of the STM32CubeProgrammer tool set (STM32CUBEPROG), and allows the generation of secure firmware and modules to be used for STM32 secure programming solutions which are:

- Secure firmware install (SFI): SFI is a secure mechanism that allows secure installation of OEM firmware in untrusted production environments by having the whole firmware encrypted with an AES-GCM key.
- Secure module install (SMI): SMI is intended to protect a part of the firmware (a section of an ELF file) by also encrypting this section using an AES-GCM key.  
A combined SFI-SMI image is an SFI image that contains one or more module areas.
- Secure firmware upgrade (SFU): SFU is a solution to allow the upgrade of the STM32 microcontroller built-in program in a secure way. For more information about SFU, please refer to the X-CUBE-SBSFU package on <http://www.st.com> for further information.

This user manual details the software environment prerequisites, as well as the available features of the STM32 Trusted Package Creator tool software.



# Contents

<b>1</b>	<b>System requirements</b>	<b>5</b>
<b>2</b>	<b>Preparation processes</b>	<b>6</b>
2.1	SFI preparation process	6
2.2	SMI preparation process	8
2.3	SFU preparation process	10
<b>3</b>	<b>STM32 Trusted Package Creator tool commands</b>	<b>12</b>
3.1	Command line interface (CLI)	12
3.2	SFI generation command	13
3.3	SMI generation command	15
3.4	SFU generation command	16
<b>4</b>	<b>STM32 Trusted Package Creator tool graphical user interface (GUI)</b>	<b>18</b>
4.1	SFI generation	20
4.2	SMI generation	23
4.3	SFU generation	25
<b>5</b>	<b>Option bytes file</b>	<b>28</b>
<b>6</b>	<b>Log dialog</b>	<b>29</b>
<b>7</b>	<b>Settings</b>	<b>30</b>
<b>8</b>	<b>SFI/SMI checking</b>	<b>31</b>
<b>9</b>	<b>Revision history</b>	<b>32</b>

List of tables

Table 1. Document revision history ..... 32

## List of figures

Figure 1.	SFI preparation process .....	6
Figure 2.	SFI file structure .....	7
Figure 3.	SMI preparation process .....	8
Figure 4.	SMI file structure .....	9
Figure 5.	SFU preparation process .....	10
Figure 6.	SFU file structure .....	11
Figure 7.	STM32 Trusted Package Creator tool's available commands .....	12
Figure 8.	SFI generation with an ELF file .....	14
Figure 9.	SFI generation with a binary file .....	14
Figure 10.	Combined SFI-SMI generation .....	14
Figure 11.	SMI generation .....	16
Figure 12.	SFU generation .....	17
Figure 13.	STM32 Trusted Package Creator tool GUI SFI tab .....	18
Figure 14.	STM32 Trusted Package Creator tool GUI SMI tab .....	19
Figure 15.	STM32 Trusted Package Creator tool GUI SFU tab .....	20
Figure 16.	Firmware file addition .....	21
Figure 17.	Successful SFI generation .....	22
Figure 18.	ELF file selection .....	23
Figure 19.	Successful SMI generation .....	25
Figure 20.	Firmware file selection .....	26
Figure 21.	Successful SFU generation .....	27
Figure 22.	Example of an option bytes file .....	28
Figure 23.	Example of a log dialog .....	29
Figure 24.	Settings dialog .....	30
Figure 25.	SFI checking .....	31

# 1      **System requirements**

Supported operating systems and architectures:

- Linux® 32-bit and 64-bit (tested on Ubuntu 14.04)
- Windows® 7/8/10 32-bit and 64-bit
- macOS® (minimum version OS X® Yosemite)

STM32CubeProgrammer and STM32 Trusted Package Creator support STM32 32-bit devices based on Arm® Cortex®-M processors.

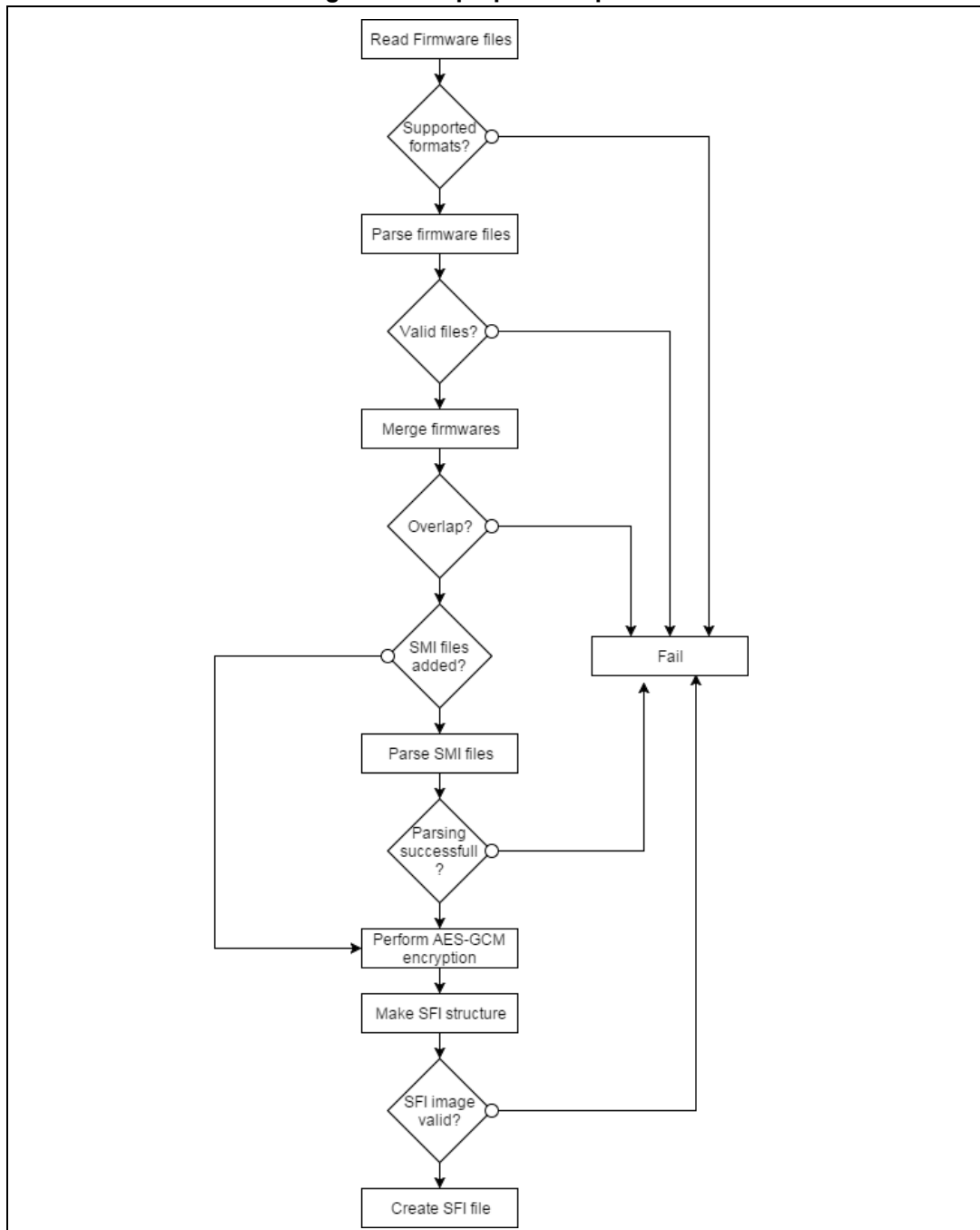
**arm**

## 2 Preparation processes

### 2.1 SFI preparation process

An SFI (Secure firmware install) image is a format created by STMicroelectronics that contains an encrypted and authenticated piece of firmware using an AES-GCM algorithm. The SFI preparation process is described in [Figure 1](#).

Figure 1. SFI preparation process



Before performing AES-GCM to encrypt an area, the tool calculates the Initialization Vector (IV) as:

$$IV = \text{nonce} + \text{Area Index}.$$

Where nonce is a number used only once as a start of an iterated process in the AES-GCM algorithm to give different cipher texts to same blocks of data.

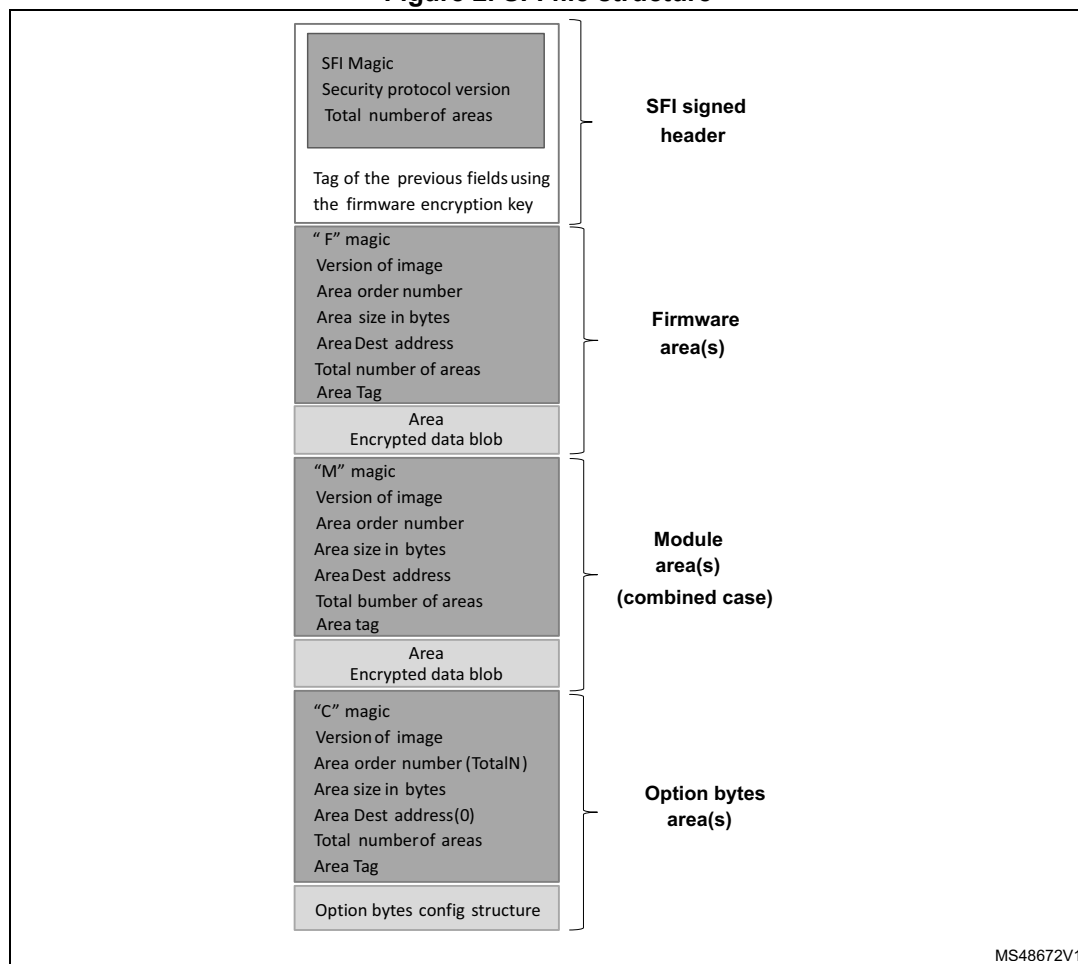
It then, it passes the area descriptor (starting from the magic to the total number of areas) as additional authenticated data (AAD).

- Each segment in the input firmwares constitutes a firmware (F) area in the SFI file.
- Each SMI file (combined case) constitutes a module (M) area.
- The option bytes configuration constitute the configuration (C) area.

To generate a header tag, the tool performs an authenticated only AES-GCM encryption (without a plain text nor a cipher text) using the SFI header as an AAD and the nonce as the IV.

The structure of an SFI file is shown in [Figure 2](#).

**Figure 2. SFI file structure**



To prepare an SFI image from multiple firmware files you have to make sure that there is no overlap between their segments, otherwise you will get the error message: *“Overlap between segments, unable to merge firmware files”*.

Furthermore, in the case of a combined SFI-SMI image, there is also an overlap check between the areas (in case there is an overlap between firmware and module areas). If the check fails, an error message is shown: *“Overlap between SFI areas”*.

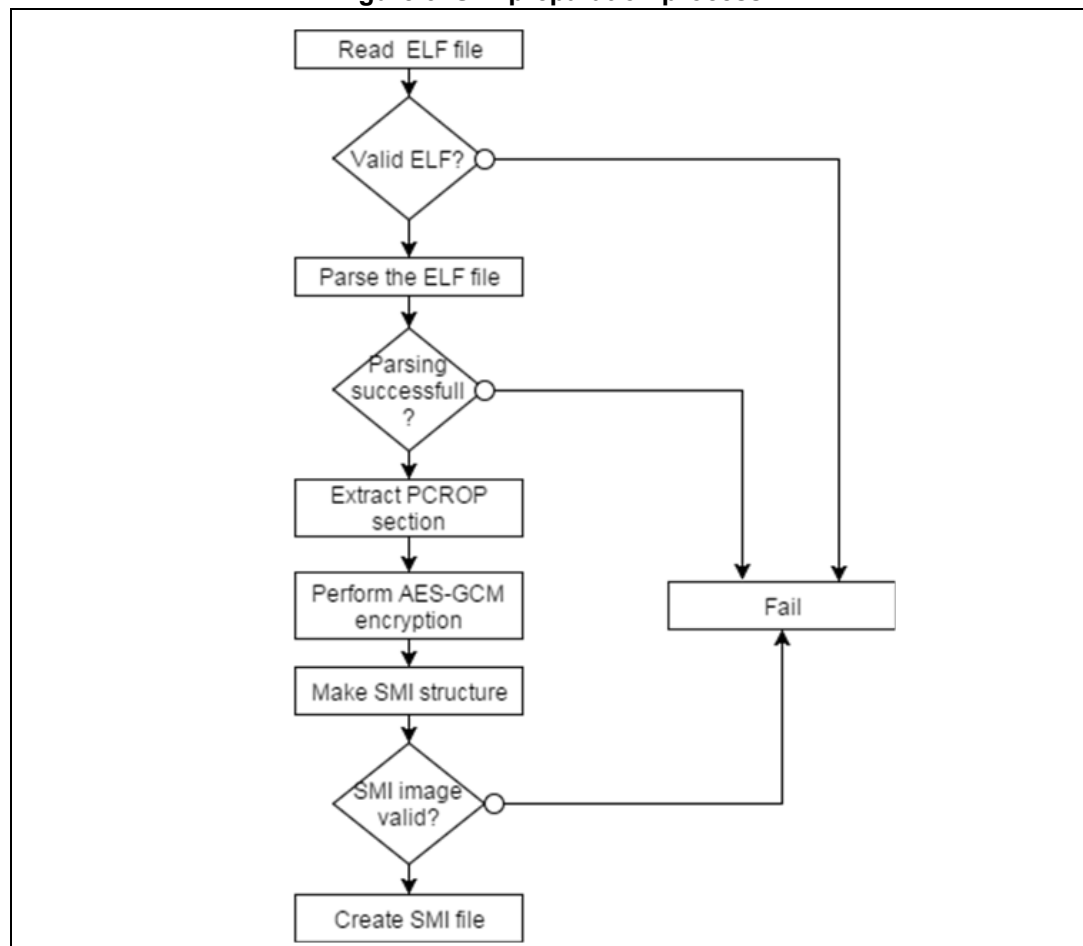
Also, all SFI areas must be located in Flash memory, otherwise the generation fails giving the error message: *“One or more SFI areas are not located in Flash memory”*.

## 2.2 SMI preparation process

An SMI image (Secure Module Install) protects only a module of the firmware.

The SMI preparation process is shown in [Figure 3](#).

**Figure 3. SMI preparation process**



The AES-GCM encryption is performed using the following inputs:

- Nonce as the initialization vector (IV)
- The security version as additional authenticated Data (AAD)



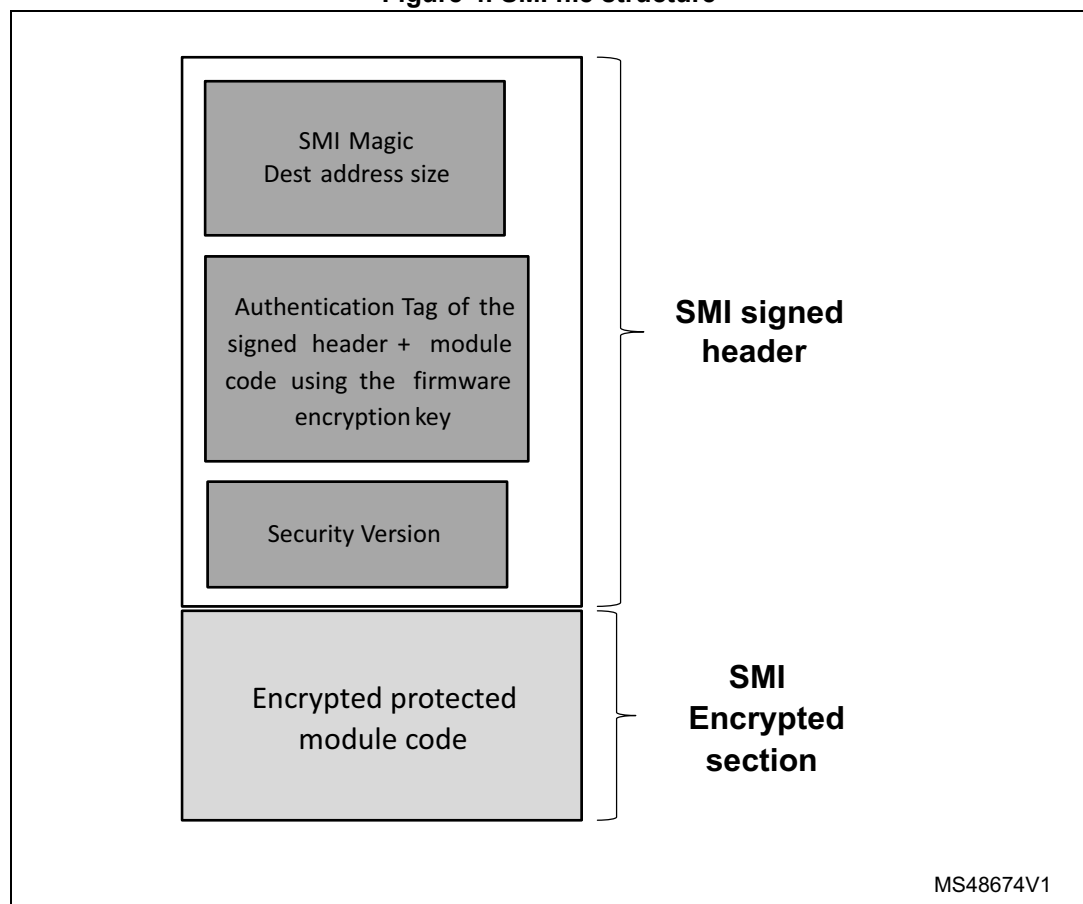
Before SMI preparation, the following checks are performed:

- A proprietary code read out protection (PCROP) section must be aligned on a Flash word (256 bits) otherwise a warning is shown
- The section's size must be at least 2 Flash words (512 bits) otherwise a warning is shown
- The section must end on a Flash word boundary (a 256 bit word) or a warning is shown
- If the section immediately following the PCROP area starts in the last Flash word of the PCROP section, the generation fails and an error message is shown.

After the SMI preparation, a clear (that is, not encrypted) ELF file is also generated containing program data and only clear sections of the code.

The structure of an SMI file is shown in [Figure 4](#).

**Figure 4. SMI file structure**

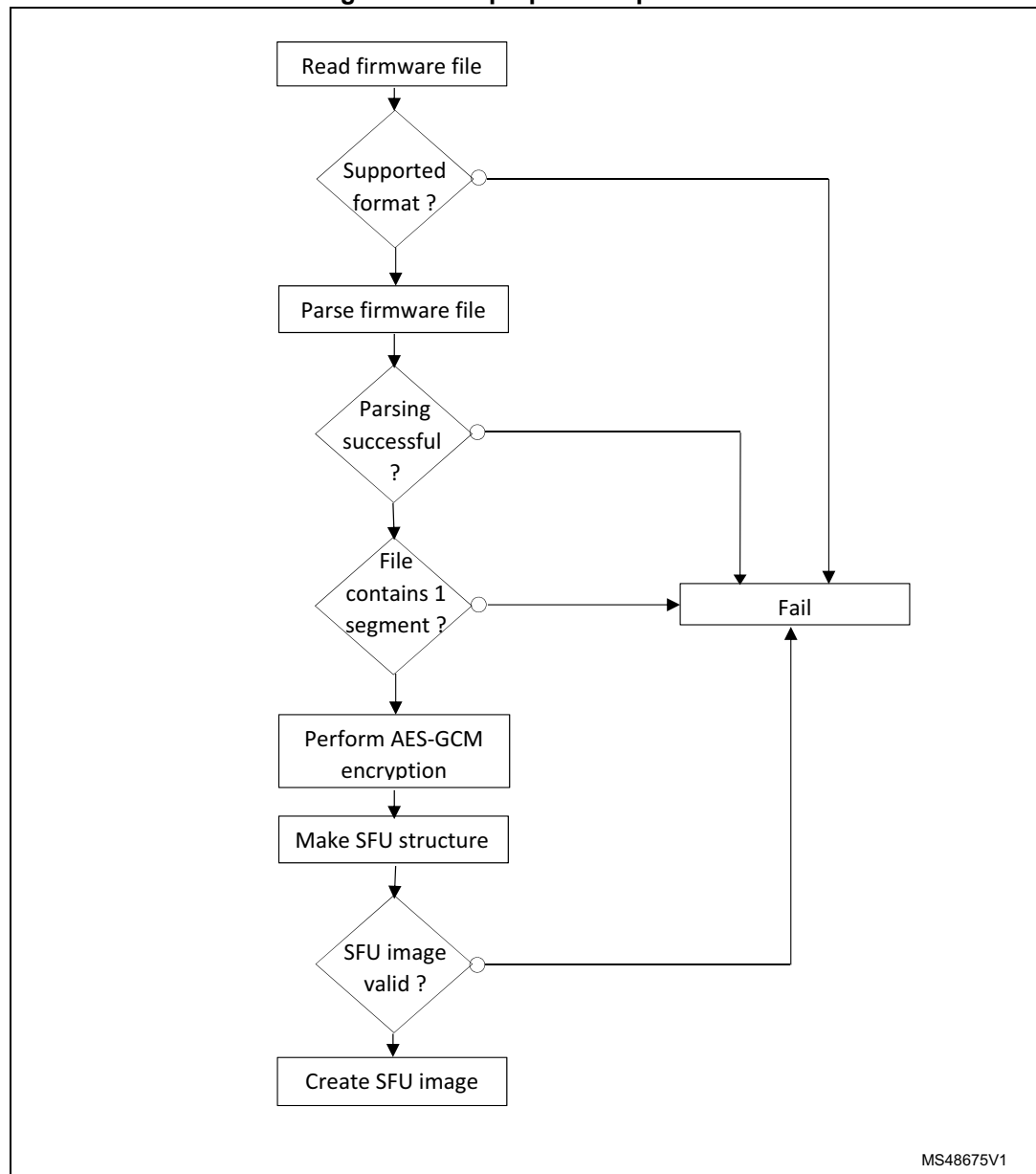


## 2.3 SFU preparation process

An SFU image (Secure Firmware Upgrade) allows the upgrade of the STM32 microcontroller built-in program in a secure way to prevent unauthorized updates.

The SFU preparation process is shown in [Figure 5](#).

**Figure 5. SFU preparation process**



2 files are generated, the SFU image header and the SFU encrypted firmware image.

For the header generation, the AES-GCM encryption is performed using the following inputs:

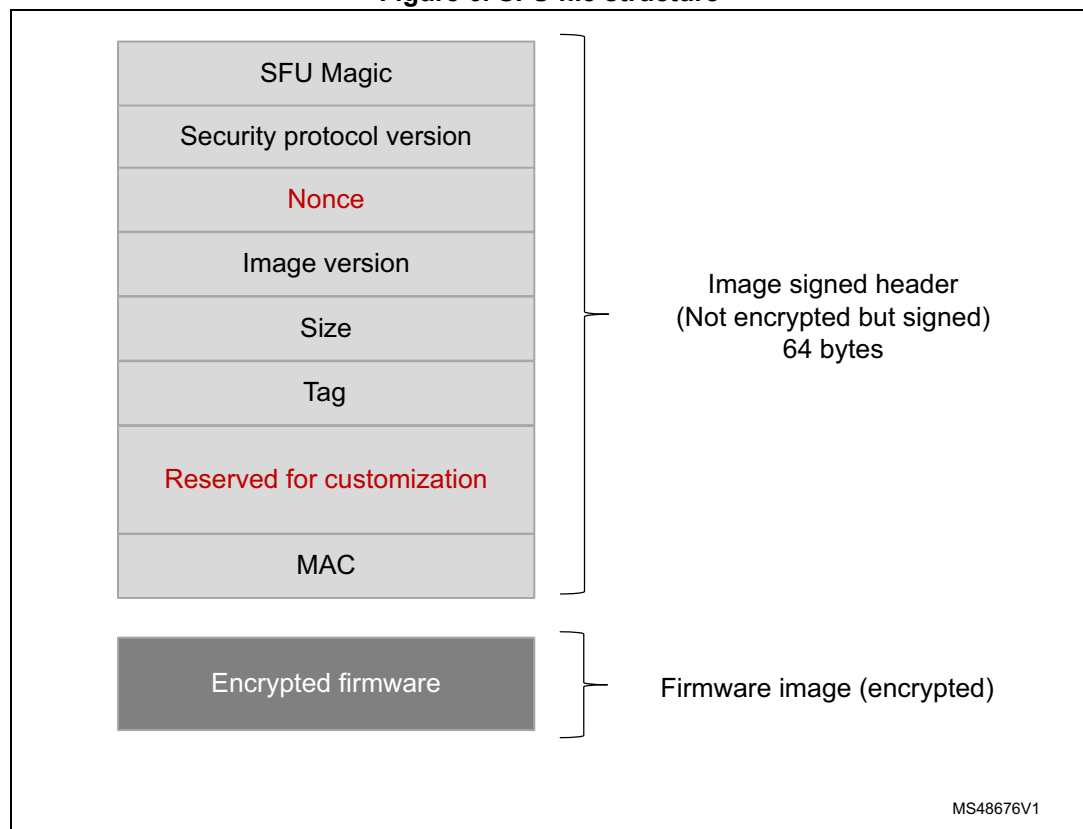
- Nonce as the Initialization Vector (IV)
- No Additional Authenticated Data (AAD)

And for the encrypted firmware image generation:

- Nonce as the Initialization Vector (IV)
- The header content as Additional Authenticated Data (AAD)

The structure of an SFU file is shown in [Figure 6](#).

**Figure 6. SFU file structure**



## 3 STM32 Trusted Package Creator tool commands

### 3.1 Command line interface (CLI)

The following sections describe how to use the STM32 Trusted Package Creator tool from the command line interface.

The available commands are shown in [Figure 7](#)

Figure 7. STM32 Trusted Package Creator tool's available commands

```

-----
STM32TrustedPackageCreator v1.0
-----

Usage :
$FMIPreparationTool_CLI.exe [option1] [value1] [option2] [value2]...

Information options
-?, -h, --help      : Display this help
-l, --log           : Generate a log file
                    [File_Name] : Log file's path and name, default file is ./trace.log

SFI preparation options
-sfi, --sfi         : Generate SFI image.
                    You also need to provide the information listed below
-fir, --firmware    : Add an input firmware file
                    <Firm_File> : Supported firmware files are ELF HEX SREC BIN
                    [Address]   : Only in case of BIN input file (in any base)
-k, --key           : AES-GCM encryption key
                    <Key_File>  : Bin file, its size must be 16 bytes
-n, --nonce         : AES-GCM nonce
                    <Nonce_File>: Bin file, its size must be 12 bytes
-v, --ver           : Image version
                    <Image_Version>: Its value must be in <0..255> (in any base)
-ob, --obfile       : Option bytes configuration file
                    <CSU_File>   : CSU file with 9 values
-m, --module        : Add an SMI file (optional for combined case)
                    <SMI_File>   : SMI file
                    [Address]   : Only in case of a relocatable SMI (with Address = 0)
-o, --outfile       : Generated SFI file
                    <Output_File>: SFI file to be created

SMI preparation options
-smi, --smi         : Generate SMI image
                    You also need to provide the information listed below
-elf, --elffile     : Input ELF file
                    <ELF_File>  : ELF file
-s, --sec           : Section to be encrypted
                    <Section>   : Section name in the ELF file
-k, --key           : AES-GCM encryption key
                    <Key_File>  : Bin file, its size must be 16 bytes
-n, --nonce         : AES-GCM nonce
                    <Nonce_File>: Bin file, its size must be 12 bytes
-sv, --sver         : Security version
                    <SV_File>   : Its size must be 16 bytes
-o, --outfile       : Generated SMI file
                    <Output_File>: SMI file to be created
-c, --clear         : Clear ELF file
                    <Clear_File>: Clear ELF file to be generated

SFU preparation options
-sfu, --sfu         : Generate SFU image.
                    You also need to provide the information listed below
-fir, --firmware    : Add an input firmware file (must have only 1 segment)
                    <Firm_File>  : Supported firmware files are ELF HEX SREC BIN
                    [Address]   : Only in case of BIN input file (in any base)
-k, --key           : AES-GCM encryption key
                    <Key_File>  : Bin file, its size must be 16 bytes
-n, --nonce         : AES-GCM nonce
                    <Nonce_File>: Bin file, its size must be 12 bytes
-v, --ver           : Image version
                    <Image_Version>: Its value must be in <0..255> (in any base)
-oh, --outhead       : Generated SFU header file
                    <Output_File>: SFU header file to be created
-os, --outsfu       : Generated SFU encrypted image file
                    <Output_File>: SFU encrypted image file to be created

```

## 3.2 SFI generation command

### **-sfi, --sfi**

**Description:** This command generates an SFI image file.

In order to generate an SFI image, the user must provide the mandatory inputs by using the options listed below.

### **-fir, --firmware**

**Description:** Add an input firmware file. Supported formats are Bin, Hex, Srec and ELF. This option can be used more than once in order to add multiple firmware files.

**Syntax:** -fir <Firmware\_file> [<Address>]

<Firmware\_file> : Firmware file.

[<Address>] : Address only for binary firmware.

### **-k, --key**

**Description:** Set the AES-GCM encryption key.

**Syntax:** -k <Key\_file>

<Key\_file> : A 16-byte binary file.

### **-n, --nonce**

**Description:** Set the AES-GCM nonce.

**Syntax:** -n <Nonce\_file>

<Nonce\_file> : A 12-byte binary file.

### **-v, --ver**

**Description:** Set the image version.

**Syntax:** -v <Image\_version>

<Image\_version> : A value between 0 and 255 in any base.

### **-ob, --obfile**

**Description:** Provide an option bytes file.

**Syntax:** -ob <CSV\_file>

<CSV\_file>: A csv file with 9 values.

### **-m, --module**

**Description:** Add an input SMI file. This option can be used more than once in order to add multiple SMI files. This is optional (for combined SFI-SMI).

**Syntax:** -m <SMI\_file>

<SMI\_file> : SMI file.

[<Address>] : Address only for relocatable SMI.

### **-o, --outfile**

**Description:** Set the SFI file to be created

**Syntax:**     -o        <out\_file>

<out\_file>:    SFI file to be generated, must have the .sfi extension.

Example:

With an ELF file:

```
STM32TrustedPackageCreator_CLI -sfi -fir ELF_firmware.axf
-k
test_firmware_key.bin -n nonce.bin -ob FIR_ob.csv -v
23 -o test.sfi
```

**Figure 8. SFI generation with an ELF file**

```
C:\STM32TrustedPackageCreator\bin>STM32TrustedPackageCreator_CLI.exe -sfi -fir E
LF_firmware.axf -k test_firmware_key.bin -n nonce.bin -ob FIR_ob.csv -v 23 -o te
st.sfi
SUCCES
```

With a binary file:

```
STM32TrustedPackageCreator_CLI -sfi -fir bin_firmware.bin
0x8000000 -k test_firmware_key.bin -n nonce.bin -ob
FIR_ob.csv -v 23 -o test.sfi
```

**Figure 9. SFI generation with a binary file**

```
C:\STM32TrustedPackageCreator\bin>STM32TrustedPackageCreator_CLI.exe -sfi -fir b
in_firmware.bin 0x08000000 -k test_firmware_key.bin -n nonce.bin -ob FIR_ob.csv
-v 23 -o test.sfi
SUCCES
```

Combined SFI-SMI:

```
STM32TrustedPackageCreator_CLI -sfi -fir ELF_firmwrae.axf -fir
bin_firmware.bin 0x8000000 -m FIR_pcrop.smi -k test_firm-
ware_key.bin -n nonce.bin -ob FIR_ob.csv -v 23 -o
test.sfi
```

**Figure 10. Combined SFI-SMI generation**

```
C:\STM32TrustedPackageCreator\bin>STM32TrustedPackageCreator_CLI.exe -sfi -fir b
in_firmware.bin 0x08000000 -m FIR_pcrop.smi -k test_firmware_key.bin -n nonce.bi
n -ob FIR_ob.csv -v 23 -o test.sfi
SUCCES
```

### 3.3 SMI generation command

#### **-smi, --smi**

**Description:** This command generates an SMI image file.

In order to generate an SMI image, the user must provide the mandatory inputs by using the options listed below.

#### **-elf, --elffile**

**Description:** Set the input ELF file.

**Syntax:** `-elf <ELF_file>`

`<ELF_file>` : ELF file. An ELF file can have any of the extensions: .elf, axf, .o, so, .out

#### **-s, --sec**

**Description:** Set the name of the section to be encrypted.

**Syntax:** `-s <section_name>`

`<section_name>`: Section name.

#### **-k, --key**

**Description:** Set the AES-GCM encryption key.

**Syntax:** `-k <Key_file>`

`<Key_file>` : A 16-byte binary file.

#### **-n, --nonce**

**Description:** Set the AES-GCM nonce.

**Syntax:** `-n <Nonce_file>`

`<Nonce_file>`: A 12-byte binary file.

#### **-sv, --sver**

**Description:** Set the security version file.

**Syntax:** `-sv <SV_file>`

`<SV_file>`: A 16 byte file.

#### **-o, --outfile**

**Description:** Set the SMI file to be created

**Syntax:** `-o <out_file>`

`<out_file>`: SMI file to be generated, must have the .smi extension.

#### **-c, --clear**

**Description:** Set the clear ELF file to be created.

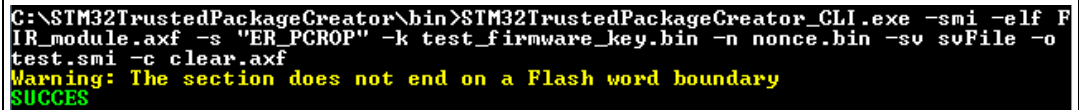
**Syntax:** `-c <ELF_file>`

`<ELF_file>`: Clear ELF file to be generated.

## Example

```
STM32TrustedPackageCreator_CLI -smi -elf FIR_module.axf -s
"ER_PCR0P" -k test_firmware_key.bin -n nonce.bin -sv
svFile -o test.smi -c clear.smi
```

Figure 11. SMI generation



```
C:\STM32TrustedPackageCreator\bin>STM32TrustedPackageCreator_CLI.exe -smi -elf F
IR_module.axf -s "ER_PCR0P" -k test_firmware_key.bin -n nonce.bin -sv svFile -o
test.smi -c clear.smi
Warning: The section does not end on a Flash word boundary
SUCCESS
```

### 3.4 SFU generation command

#### -sfu, --sfu

**Description:** This command generates an SFU image file.

In order to generate an SFU image, the user must provide the mandatory inputs by using the options listed below.

#### -fir, --firmware

**Description:** Set an input firmware file. Supported formats are Bin, Hex, Srec and ELF. The firmware file must contain only one segment.

**Syntax:** -fir <Firmware\_file> [<Address>]

< Firmware \_file>: Firmware file.

[<Address>]: Address only for binary firmwares.

#### -k, --key

**Description:** Set the AES-GCM encryption key.

**Syntax:** -k <Key\_file>

<Key \_file>: A 16-byte binary file

#### -n, --nonce

**Description:** Set the AES-GCM nonce.

**Syntax:** -n <Nonce\_file>

<Nonce \_file>: A 12-byte binary file.

#### -v, --ver

**Description:** Set the image version.

**Syntax:** -v <Image\_version>

<Image\_version>: A value between 0 and 255 in any base.

#### -oh, --outheader

**Description:** Set the SFU header file to be created.

**Syntax:** -oh <out\_file>



<out\_file>: SFU header file to be generated, must have the .sfuh extension.

**-os, --outsfu**

**Description:** Set the SFU file to be created.

**Syntax:**        -os        <out\_file>

<out\_file>:        SFU file to be generated, must have the .sfu extension.

**Example:**

```
SFMIPreparationTool_CLI    -sfu    -fir    bin_firmware.bin    -k  
test_firmware_key.bin    -n    nonce.bin    -v    23  
-oh    out.sfuh    -os    out.sfu
```

**Figure 12. SFU generation**

```
C:\STM32TrustedPackageCreator\bin>STM32TrustedPackageCreator_CLI.exe -fir bin_fi  
rmware.bin -k test_firmware_key.bin -n nonce.bin -v 23 -oh out.sfuh -os out.sfu  
-sfu  
SUCCESS
```

# 4 STM32 Trusted Package Creator tool graphical user interface (GUI)

This section describes how to use the STM32 Trusted Package Creator tool with its graphical user interface.

The STM32 Trusted Package Creator tool GUI presents three tabs: one for SFI generation (Figure 13), one for SMI generation (Figure 14) and one for SFU generation (Figure 15).

Figure 13. STM32 Trusted Package Creator tool GUI SFI tab

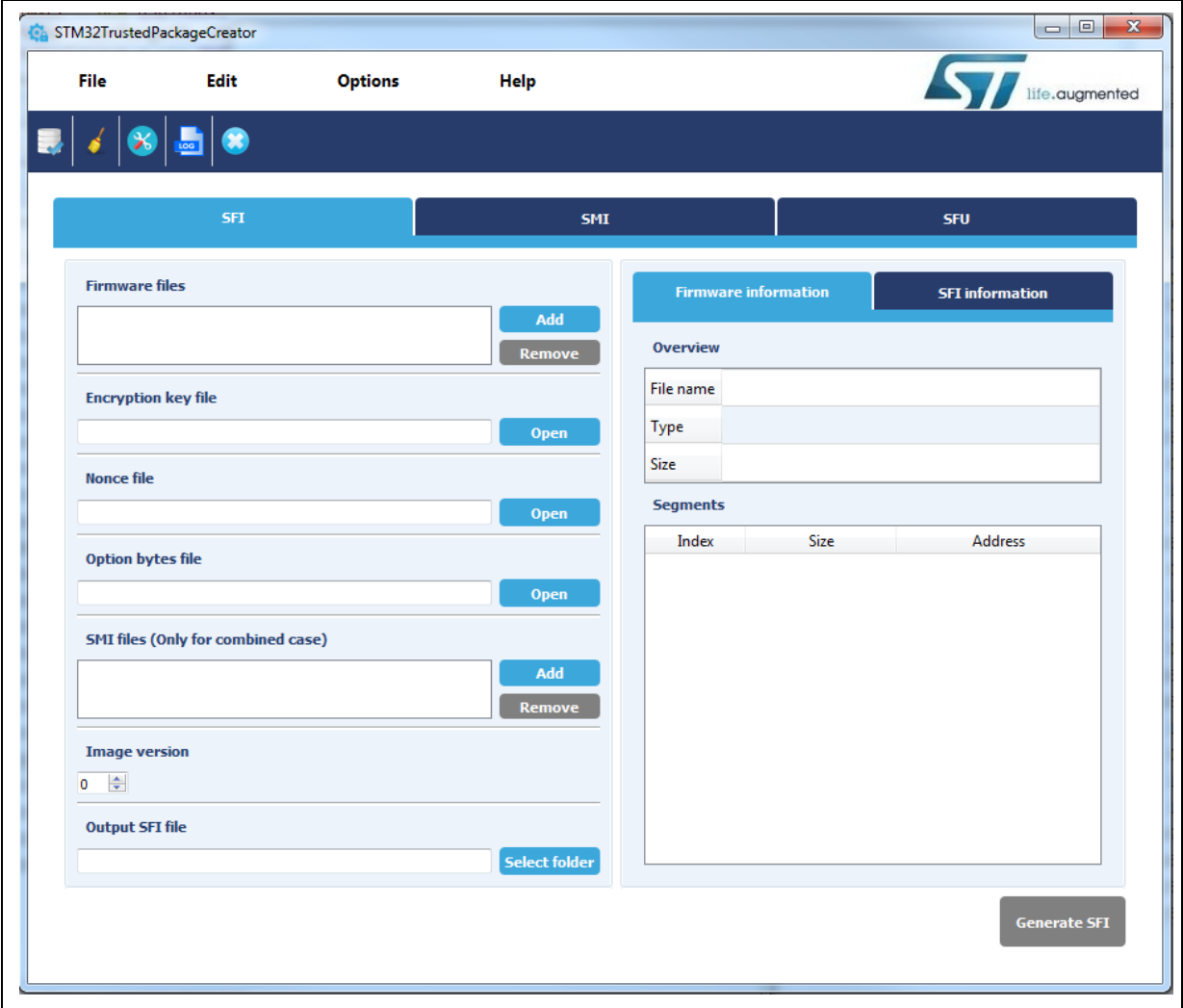


Figure 14. STM32 Trusted Package Creator tool GUI SMI tab

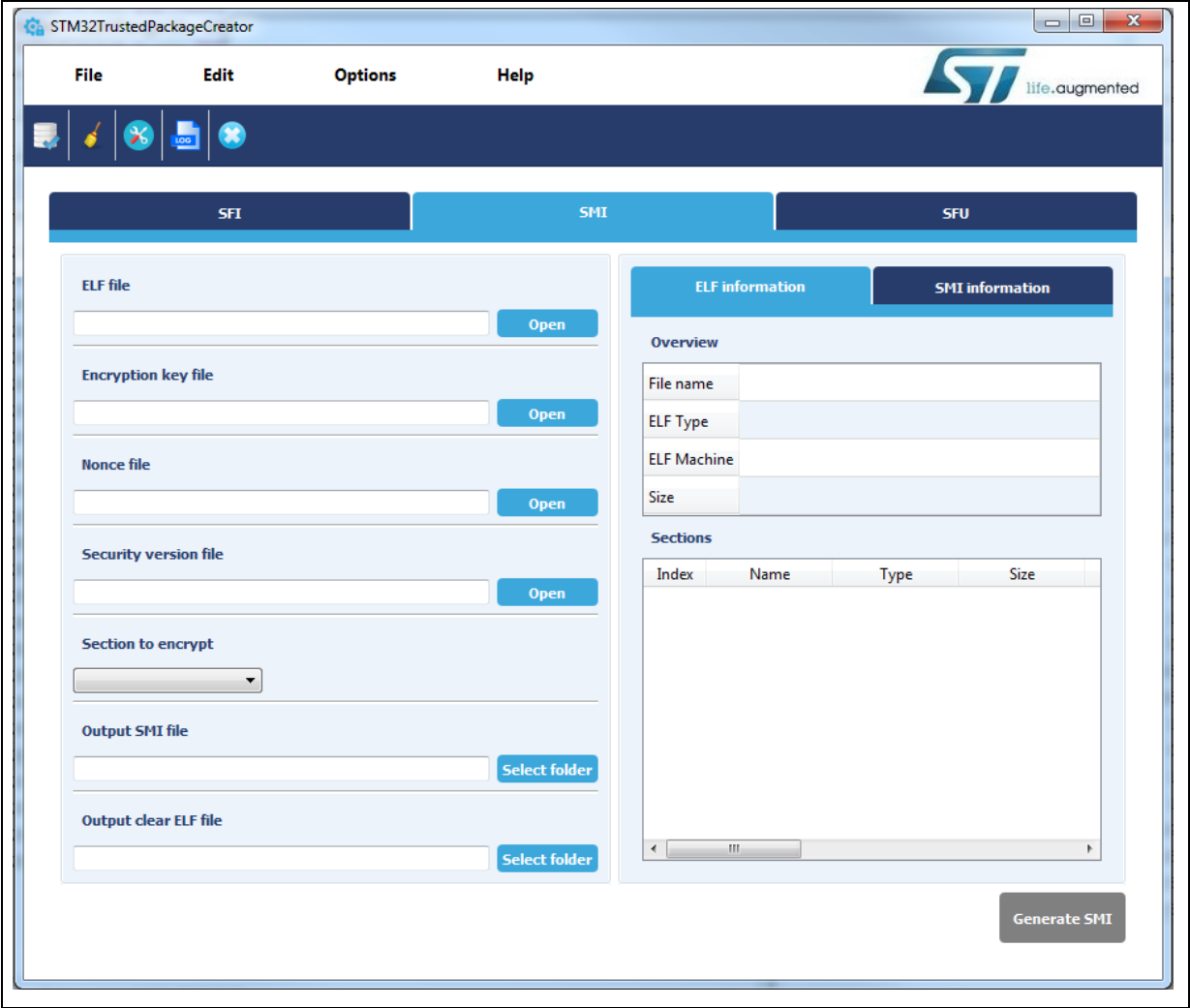
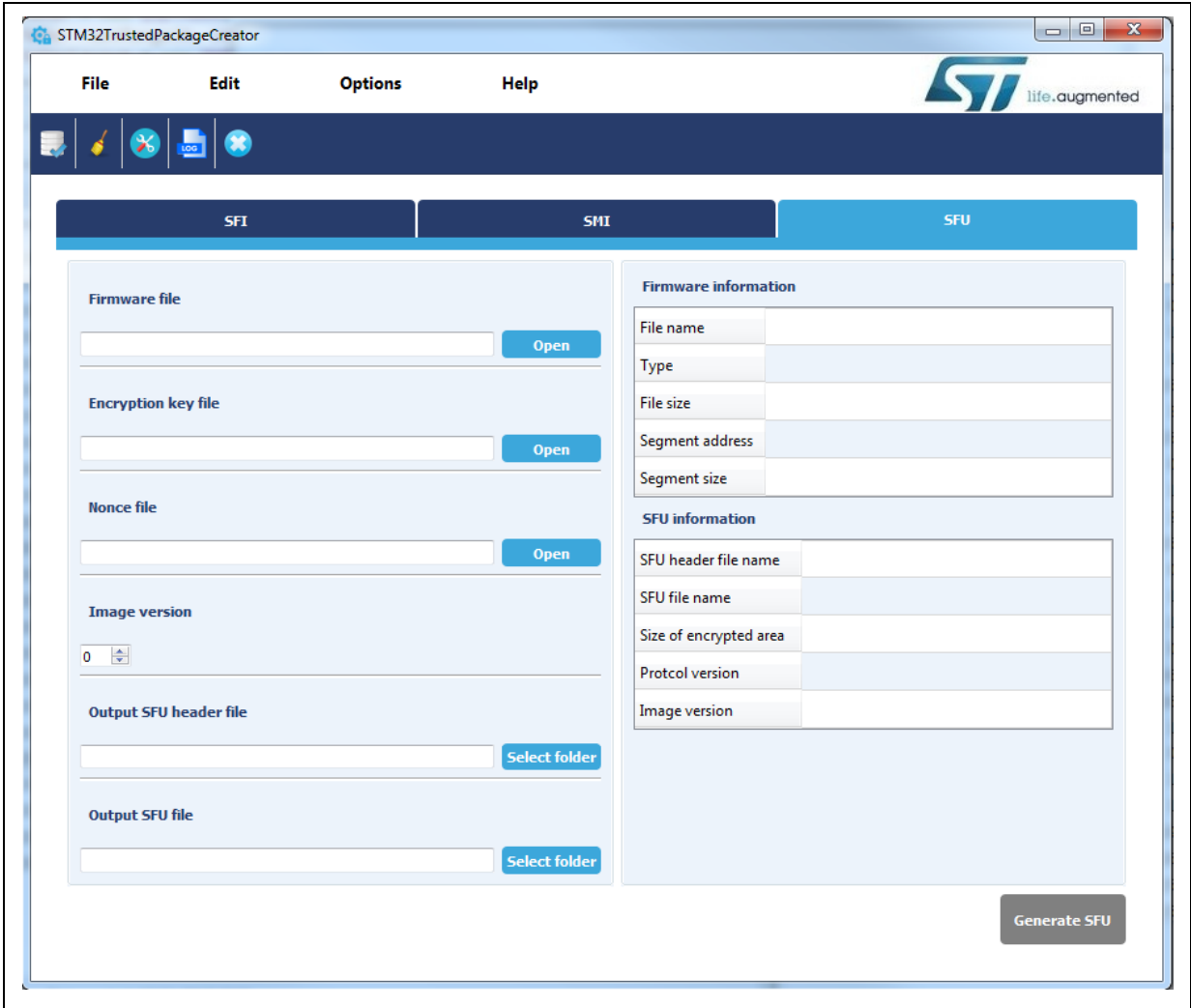


Figure 15. STM32 Trusted Package Creator tool GUI SFU tab



## 4.1 SFI generation

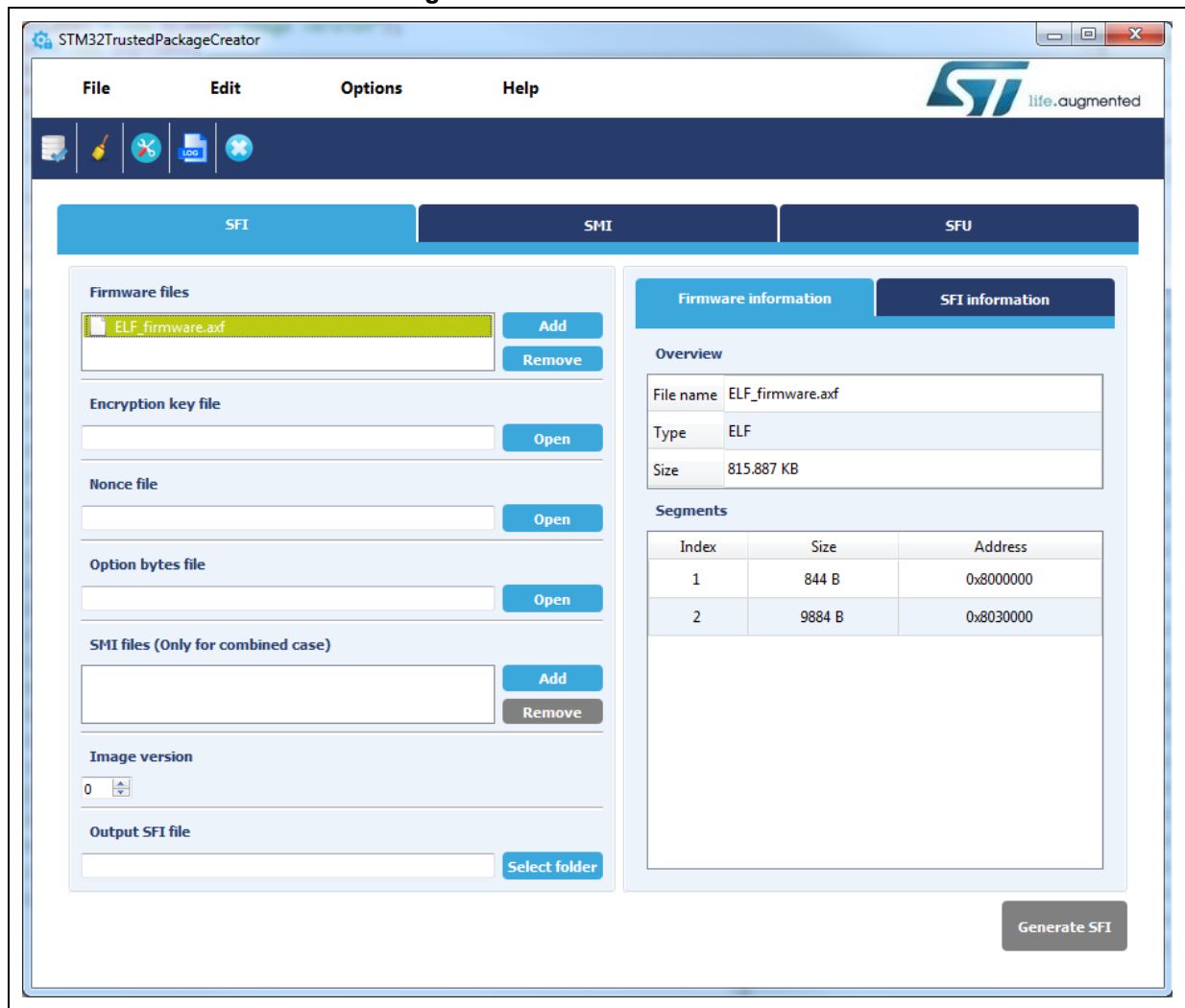
To validate the SFI generation request, the user has to fill in the input fields with valid values:

### Firmware files:

The user must add the input firmware files with the **add** button.

*Note: If the file is valid, it is added in the firmware files list. Selecting it makes several pieces of related information appear in the Firmware information section (Figure 16), otherwise an error message box is shown saying either the file could not be opened or the file is not valid. If the file is in a binary format, a dialog box appears requesting that an address be provided. A file can be removed with the **remove** button.*

Figure 16. Firmware file addition

**Encryption key and nonce file:**

The encryption key and nonce file can be selected by entering their paths (absolute or relative), or by selection with the **open** button. Notice that sizes must be respected (16 bytes for the key and 12 bytes for nonce).

**Option bytes file:**

The option bytes file can be selected the same way. Only csv files are supported.

**SMI files:**

SMI files can be added the same way as the firmware files. Selecting a file makes several pieces of related information appear in the Firmware information section.

**Image version:**

Image version value in [0..255].

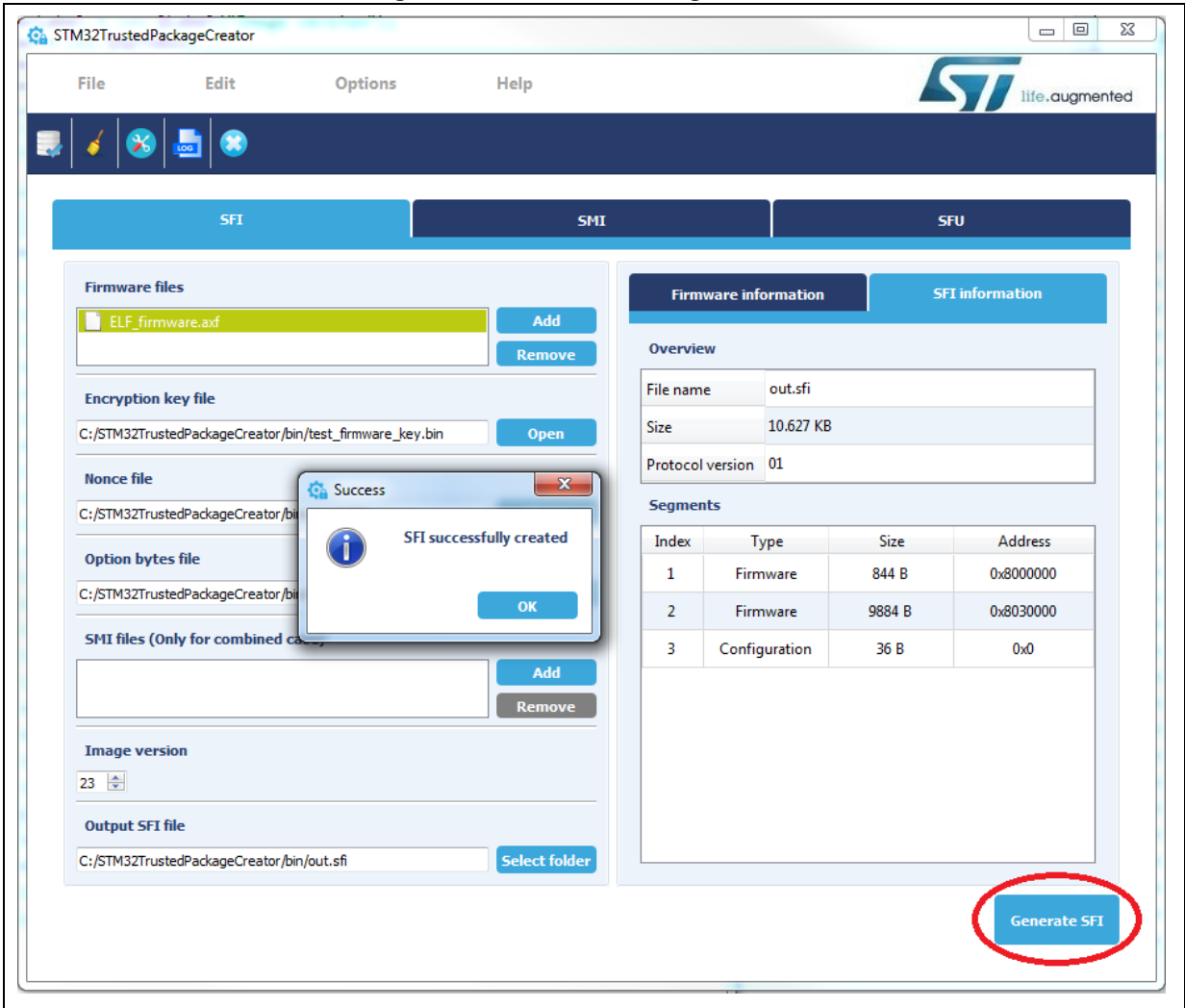
Output file:

An output file can be selected by entering its path (absolute or relative), or with the **select folder** button, note that with the latter way, a name *out.sfi* is suggested, you can keep or change it

When all fields are properly filled in, the **Generate SFI** button becomes active. The user may generate the SFI file by clicking on it.

If everything goes well, a message box indicating successful generation appears (Figure 17) and information about the generated SFI file is displayed in the SFI information section.

Figure 17. Successful SFI generation



## 4.2 SMI generation

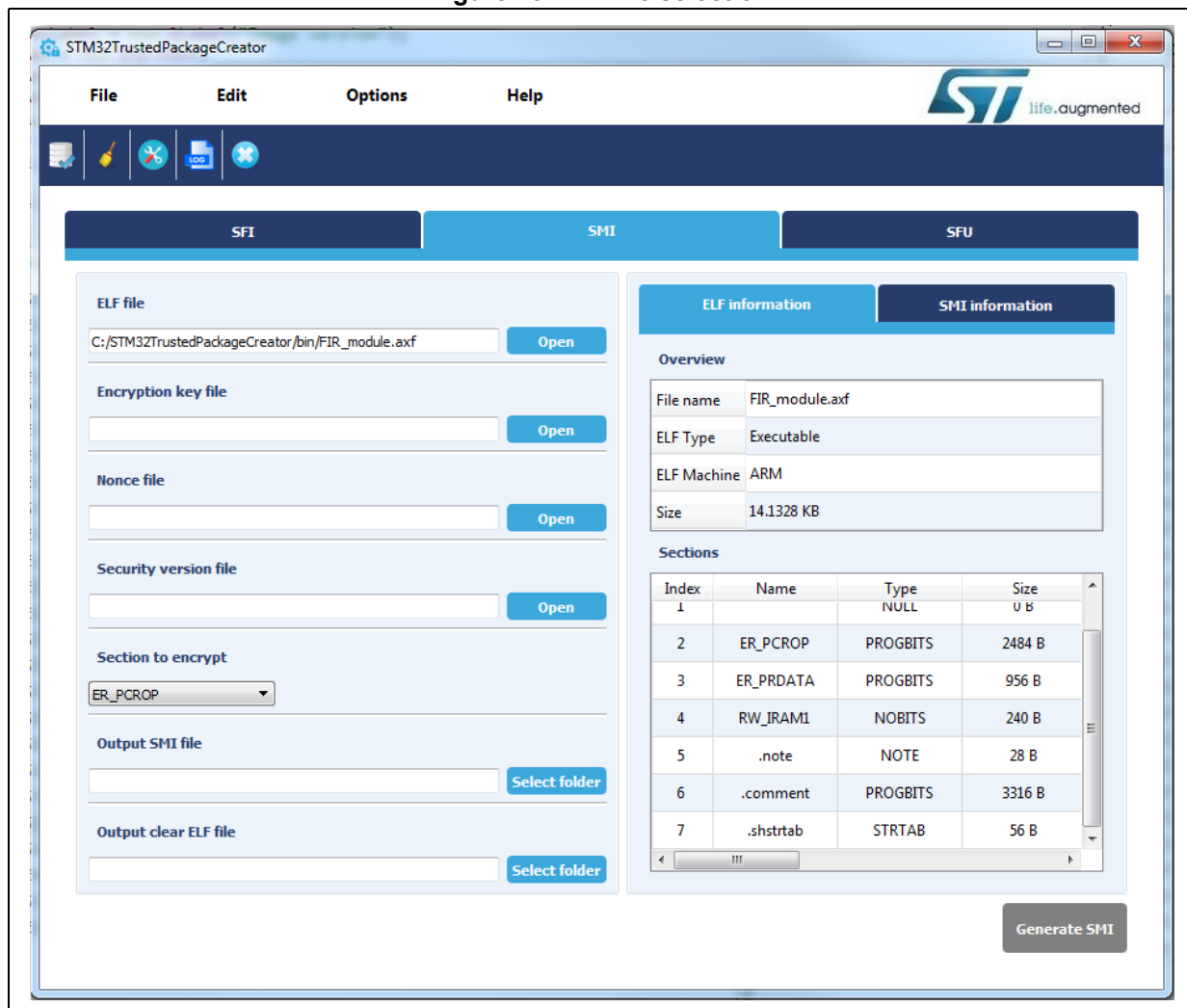
As for SFI generation, the user must provide the input information.

### Elf file:

In this case the input file can be only an elf file.

If the file is valid, information is displayed in the “*ELF information*” tab (Figure 18), otherwise an error message box is shown saying either the file could not be opened or that the file is not valid.

Figure 18. ELF file selection



**Encryption key and nonce file:**

As for SFI, the Encryption key and nonce file can be selected in the same way as the firmware file. Notice that sizes must be respected (16 bytes for the key and 12 bytes for nonce).

**Security version file:**

The security version file size must be 16 bytes.

A security version file is provided under the *Security\_Version* folder.

**Section:**

This is a section list that can be used to select the name of the section to be encrypted.

**output files:**

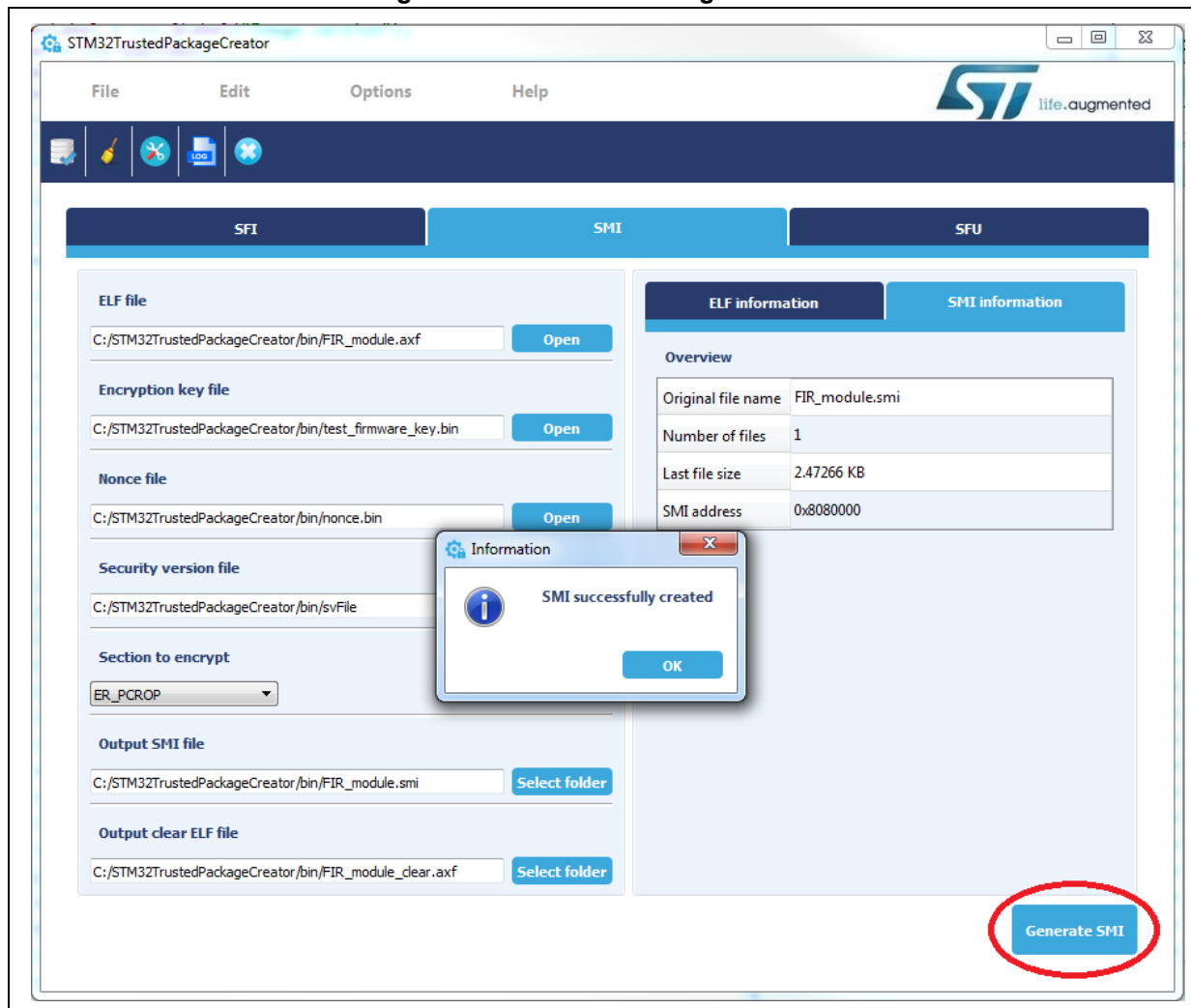
Output files can be selected by entering their paths (absolute or relative), or with the **Select folder** button, note that with the latter way a name is suggested, which can be kept or changed.

When all fields are properly filled in, the user may generate the SFI file by a click on the **Generate SFI** button (the button becomes active).

A message box informing the user that generation was successful appears ([Figure 19](#)), in addition of information about the generated SMI file, otherwise an error is displayed.



Figure 19. Successful SMI generation



### 4.3 SFU generation

Input fields in this case are similar to the SFI generation use case.

#### Firmwares files:

The user needs to enter the input firmware files.

If the file is valid, several pieces of related information appear in the Firmware information section ([Figure 20](#)), otherwise an error message box is shown. Note that the firmware file must have only one segment.

Figure 20. Firmware file selection

The screenshot shows the STM32 Trusted Package Creator GUI. The window title is "STM32TrustedPackageCreator". The menu bar includes "File", "Edit", "Options", and "Help". The toolbar contains icons for file operations. The main interface has three tabs: "SFI", "SMI", and "SFU". The "SFU" tab is currently selected.

**Firmware file selection section:**

- Firmware file:** A text field containing "C:/STM32TrustedPackageCreator/bin/file.hex" and an "Open" button.
- Encryption key file:** A text field and an "Open" button.
- Nonce file:** A text field and an "Open" button.
- Image version:** A numeric input field with a value of "0" and a spinner.
- Output SFU header file:** A text field and a "Select folder" button.
- Output SFU file:** A text field and a "Select folder" button.

**Firmware information section:**

File name	file.hex
Type	Intel Hex
File size	27.2002 KB
Segment address	0x8030000
Segment size	9884 B

**SFU information section:**

SFU header file name	
SFU file name	
Size of encrypted area	
Protocol version	
Image version	

A "Generate SFU" button is located at the bottom right of the interface.

**Encryption key and nonce file:**

Same as for SFI and SMI use cases.

**Image version:**

Image version value in [0..255].

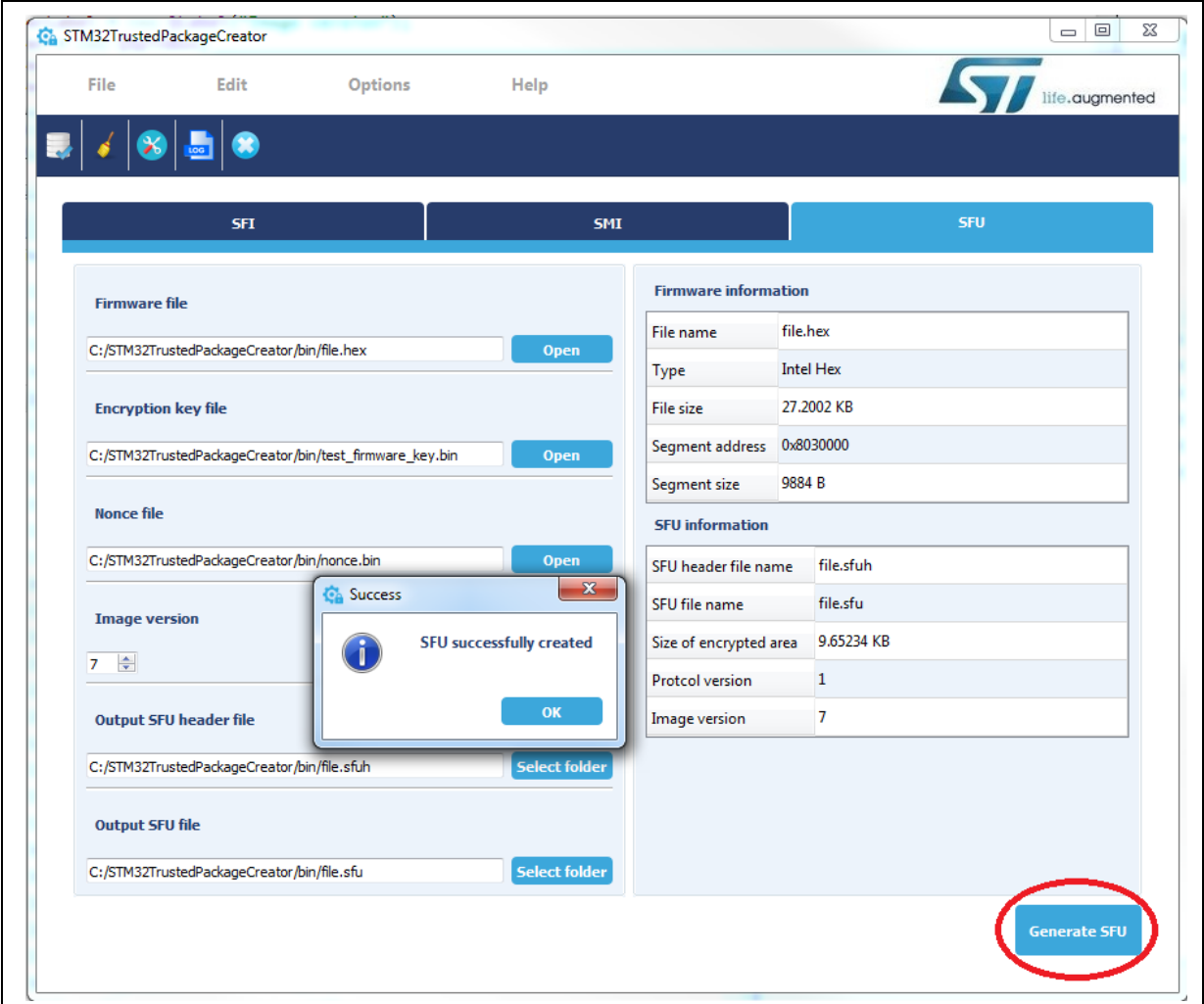
**output files:**

Output files can be selected by entering their paths (absolute or relative), or with the **select folder** button, note that in the latter way a name is suggested, which can be kept or changed.

When all fields are properly filled in, the **Generate SFU** button becomes active. The user may generate the SFU file by clicking on it.

If everything goes well, a message box informing that generation is successful appears ([Figure 21](#)) and information about the generated SFU file is displayed in the SFU information section.

Figure 21. Successful SFU generation



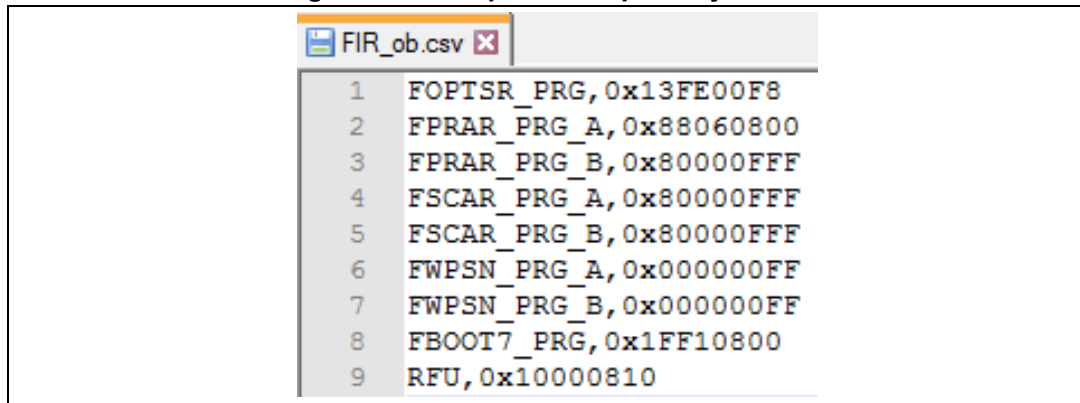
## 5 Option bytes file

The option bytes file field is mandatory for SFI applications only, it allows option bytes to be programmed during secure firmware install.

Only CSV (Comma Separated Value) format is supported for such files, it is composed of two vectors: a register name and its value.

All of the 9 option-byte registers must be configured (a total of 9 lines in a csv file).

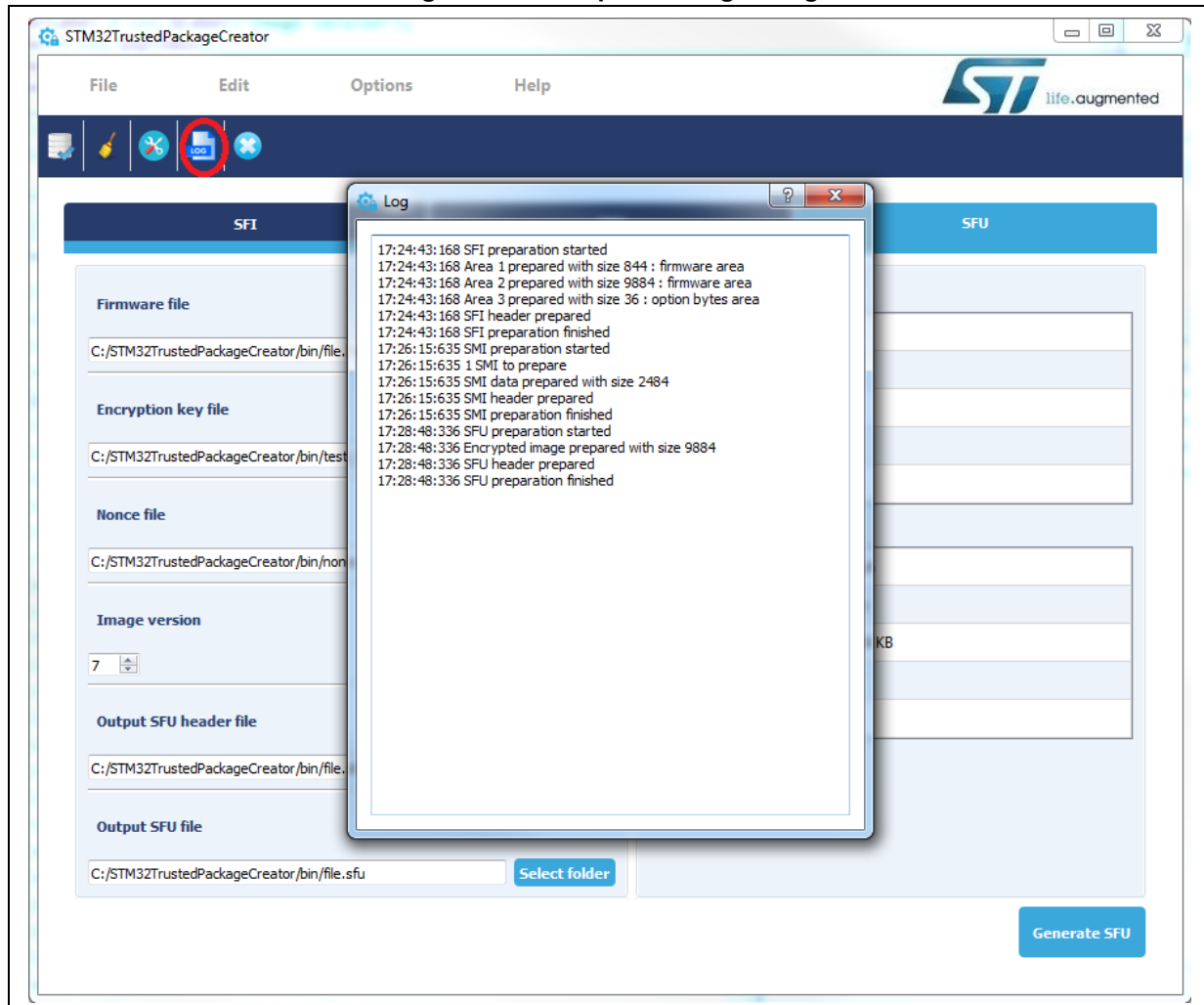
**Figure 22. Example of an option bytes file**



## 6 Log dialog

A log can be visualized by clicking the **log** button in the tools bar or in the menu bar: Options-> log.

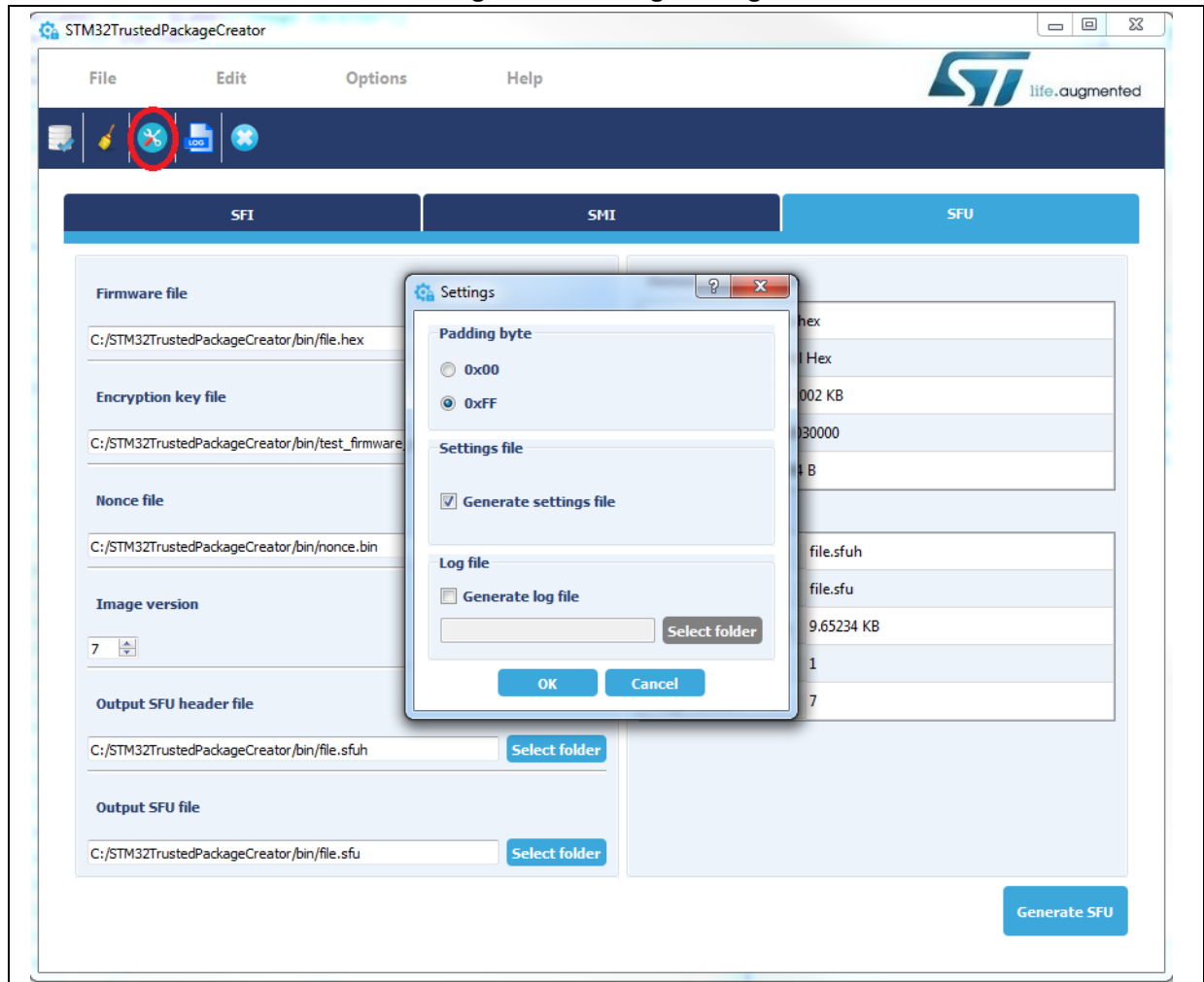
Figure 23. Example of a log dialog



## 7 Settings

The settings dialog can be accessed by clicking the **settings** button in the tools bar or in the menu bar: Options -> settings.

Figure 24. Settings dialog



### Padding byte:

When parsing files, padding may be added to fill the gap between segments separated by 16 bytes or less, in order to merge them and reduce the number of segments. The user may have the choice between 0xFF (default value) or 0x00.

### Settings file:

When checked, a *settings.ini* file is generated in the executable folder. It saves the application state: window size and field contents.

### Log file:

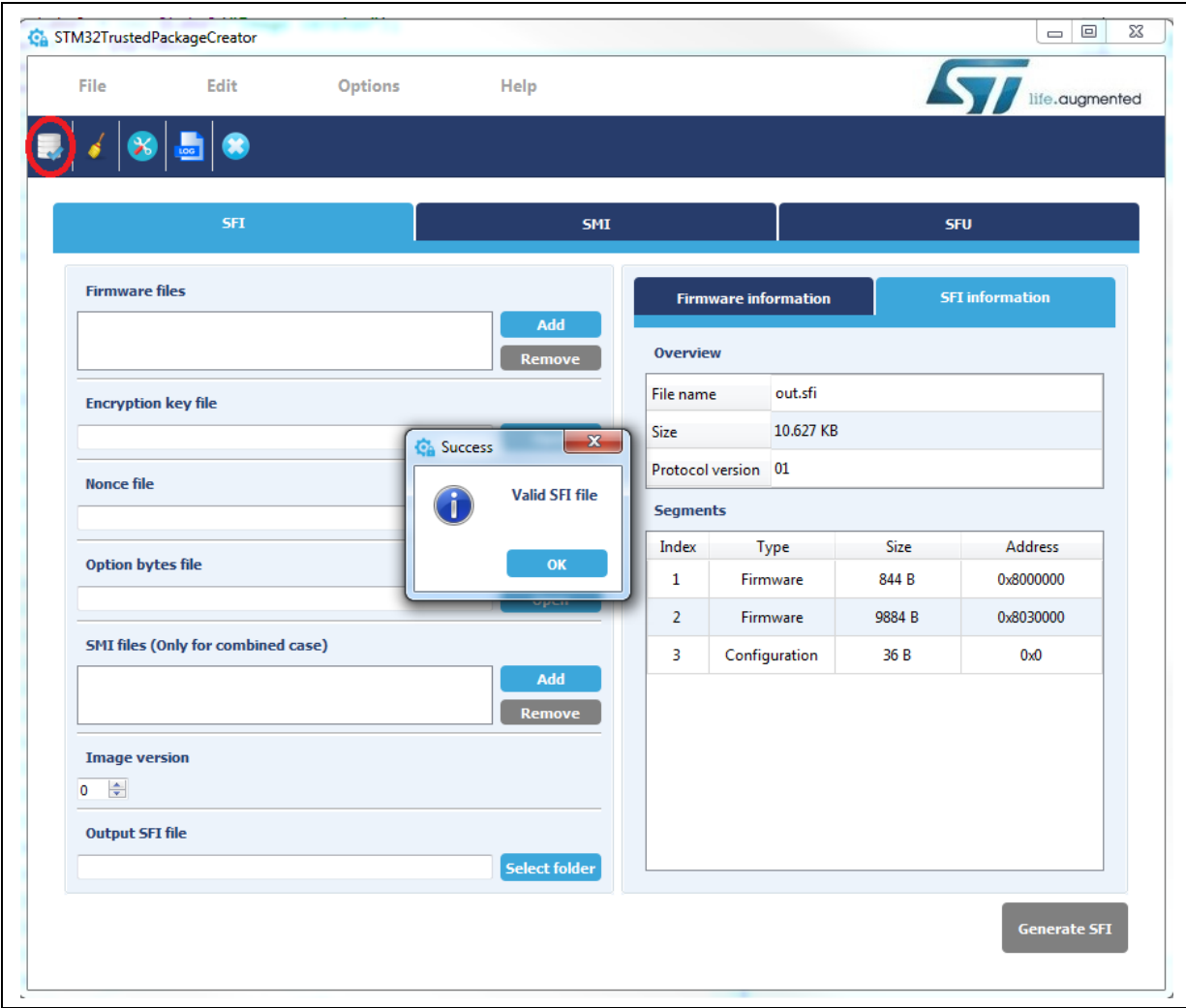
When checked, a log file is generated in the selected path.

# 8 SFI/SMI checking

SFI/SMI checking can be accessed by clicking the **Check SFI/SMI** button in the tools bar or in the menu bar: File -> Check SFI/SMI.

This allows SFI or SMI file validity to be checked, in addition to displaying information about it.

Figure 25. SFI checking



9      **Revision history**

**Table 1. Document revision history**

Date	Revision	Changes
20-Dec-2017	1	Initial release.





**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2017 STMicroelectronics – All rights reserved