Overview of the secure secret provisioning (SSP) on STM32MPx series

Introduction

The 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 address new market security requirements and protect customers against any leakage of their intellectual property (IP), STMicroelectronics introduces a new security concept: secure secret provisioning (SSP). This concept enables the programming of original equipment manufacturer (OEM) secrets into the STM32MPx series one-time programming (OTP) area and backup memory areas securely, ensuring confidentiality, authentication, and integrity checks.

The STM32MPx series supports protection mechanisms allowing 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 STM32MPx series SSP solution with its associated tools ecosystem, and explains how to use it to protect OEM secrets during the CM product manufacturing stage.

In the remainder of the document:

- STM32MPx refers to the STM32MP1 series and STM32MP2 series.
- STM32MP1 refers to the STM32MP1 series.
- STM32MP2 refers to the STM32MP2 series.
- STM32MP15 refers to the STM32MP151, STM32MP153 and STM32MP157 lines of products.
- STM32MP13 refers to the STM32MP131, STM32MP133 and STM32MP135 lines of products.
- STM32MP25 refers to the STM32MP251, STM32MP253 and STM32MP257 lines of products.
- STM32MP23 refers to the STM32MP231, STM32MP233 and STM32MP237 lines of products.
- STM32MP21 refers to the STM32MP211, STM32MP213 and STM32MP215 lines of products.

1 Reference documents

Table 1. Reference documents

Reference number	Document title
[AN4992]	Introduction to secure firmware install (SFI) for STM32 MCUs
[AN5827]	Guidelines for entering RMA state on STM32MP1 series MPUs
[AN5275]	Introduction to USB DFU/USART protocols used in STM32MP1 and STM32MP2 MPU bootloaders
[DB5423]	Secure secret provisioning (SSP) solution for STM32 microprocessors
[ROM_CODE]	https://wiki.st.com/stm32mpu/wiki/STM32_MPU_ROM_code_overview
[SECURE_BOOT]	https://wiki.st.com/stm32mpu/wiki/How_to_enable_secure_boot_on_STM32_MPU
[SSP]	https://wiki.st.com/stm32mpu/wiki/How_to_deploy_SSP_using_a_step-by-step_approach
[STM32HSM-V2]	https://www.st.com/en/development-tools/stm32hsm-v2.html
[STM32MPUSSP-UTIL]	https://www.st.com/en/development-tools/stm32mpussp-util.html
[STM32Trust]	https://www.st.com/stm32trust
[TrustedFirmware-A]	https://github.com/STMicroelectronics/arm-trusted-firmware(1).
[UM2237]	STM32CubeProgrammer software description
[UM2238]	STM32 Trusted Package Creator tool software description
[UM2542]	STM32 key generator software description
[UM2543]	STM32 signing tool software description

1. *This URL belongs to a third party. It is active at document publication, however STMicroelectronics shall not be liable for any change, move or inactivation of the URL or the referenced material.*

2 General information

This document applies to STM32MPx series Arm®-based devices.

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

arm

The following table defines the acronyms needed for a better understanding of this document.

Table 2. List of acronyms

Acronym	Description
AES	Advanced encryption standard (symmetric cryptographic method)
CM	Contract manufacturer
ECC	Elliptic curve cryptography
ECDSA	Elliptic curve digital signature algorithm
ECIES	Elliptic curve integrated encryption scheme
ST HSM	ST hardware security module
IV	Initialization vector
OEM	Original equipment manufacturer
OTP	One time programming
PKH	Public key hash
PKHTH	Public key hash table hash
SHA-256	Secure hash algorithm on 256 bits. SHA-256 is one variant of SHA-2 family
SSP	Secure secret provisioning
TF-A	TrustedFirmware-A, with A meaning Arm® Cortex®-A

3 STM32MPx secure secret provisioning

3.1 SSP principle overview

The SSP is a secure mechanism implemented in STM32 microprocessors that allows a secure and counted installation of OEM secrets in untrusted production environment (such the OEM contract manufacturer).

The STM32MPx series device may contain several secure data in OTP fuses:

- OEM PKH or PKHTH: the root of trust for the secure boot authentication.
- RMA passwords: used to change the device life state for hardware investigation purpose.
- Other OEM secrets that may be used by secure software.

The SSP is derived from the STM32 microcontrollers secure firmware install [AN4992], so it reuses the same kind of authentication and encryption mechanism.

- ST hardware security module (ST HSM).
- Secure device communication.

The SSP embedded code is split in two parts:

- One part in the [ROM_CODE] accessing the device private information.
- Another part in the SSP secure firmware.

The SSP process prevents the OEM secrets from:

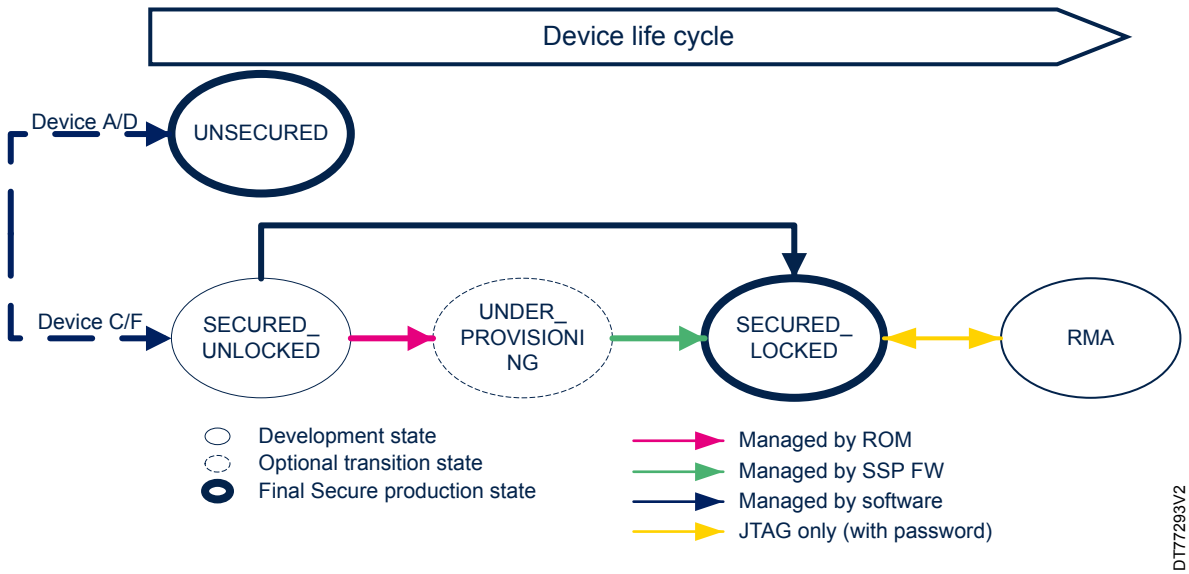
- Being accessed by the contract manufacturer.
- Being extracted or disclosed.

The SSP process is limited to a serial boot process during the initial provisioning. It can only be run once (except if an issue occurs during the process) and automatically sets the device as a secured_locked device once provisioning is done. It ensures that the device enables the secure boot feature, protects the secrets and only let OEM signed firmware to be executed.

3.2 Device life cycle

The device life cycle ensures that the device is used within the appropriated state. The SSP process ensures that the life cycle automatically changes from the *secured_unlock* device state to the *secured_locked* state in a secure way, using the intermediate *under_provisioning* state. It guarantees that all the required secrets necessary to provision the device and set it in a secure state are programmed. The SSP process is only executed during the *under_provisioning* state.

Figure 1. Device life cycle



3.3 OEM data and key exchange

Thanks to STM32MPx security features and cryptographic algorithm, STM32MPx supports secure OEM secrets programming in one time programming area (OTP). It ensures OEM firmware protection (confidentiality, authenticity, and integrity) during the STM32MPx [SECURE_BOOT] process. The OTP area provides secrets obfuscation thanks to robust hardware mechanisms.

STM32MP2x extends the support of additional backup memories that are tamper protected and may contain further OEM secrets.

3.3.1 Secrets encryption mechanism

This mechanism consists in having the OEM secrets encrypted using AES GCM with an OEM 128-bit transport key and 128-bit IV thanks to STM32 Trusted Package Creator tool [UM2238].

It generates an encrypted secret binary file that is protected and can be sent to the manufacturer for provisioning. This file is a mandatory input to initiate the SSP process.

The STM32MP1x uses a basic binary-encrypted format.

The STM32MP2x uses the same SFI format as the STM32MCU (see [AN4992]). It relies on multiple sections for each specific secret area.

3.3.2 OEM 128-bit transport key and IV protection

STM32HSMv2 [STM32HSM-V2] is a hardware security module that stores and provides the OEM transport key (and IV for AES GCM algorithm) in a secure way. Thanks to asymmetric encryption, a dedicated per-device license is generated by the ST HSM to embed the OEM transport key and IV.

The ST HSM manages device authentication and counts the number of generated licenses to avoid over production.

The ST HSM must be provisioned by the OEM using STM32 Trusted Package Creator tool [UM2238]. The OEM is able to:

- Define the OEM 128-bit transport key and IV used to encrypt secrets.
- Choose the personalization data file to identify the device to be provisioned.
- Choose the maximum number of devices to be provisioned (depends on the STM32HSM-V2 version used).

3.3.3 Device authentication

To control the provisioning, the device certificate is requested to generate the device dedicated license. The STM32MPx are default provisioned with an ST certificate used by the ST HSM to verify the device authenticity. It avoids using the ST HSM (and the secret files) on untrusted devices.

3.4 SSP process flow

3.4.1 OEM secrets preparation

Secret preparation is the initial step to be made by the OEM. It first lists the secrets that are used in the platform and must be provisioned during the production. Thanks to the STM32 Trusted Package Creator, dedicated panels are designed to construct the OEM secret files used during the manufacturing process.

3.4.1.1 OTP

The SecretGen panel includes, by default, a minimal set of secrets that need to be embedded in the device to ensures the secure platform configuration:

- STM32MP1x:
  - STM32MP15x: Only RMA password
  - STM32MP13x: RMA password and optional EDMK (FSBL encryption key)
- STM32MP2x:
  - OTP secure boot configuration words and Root of trust configuration:
    - FSBL monotonic counter (One for FSBL-A and one for FSBL-M)
    - Key revocation mask (One per FSBL-A or M)
    - Second root of trust activation (BOOTROM_CONFIG_8) when CM33-TD is selected
    - Debug lock configuration (BOOTROM_CONFIG_9)
    - OEM1/2 key Root of trust
    - OEM1/2 Encryption key
      These default selected secrets are not all mandatory (depending on the boot profile A35-TD or M33-TD).
  - STM32MP2x SSP supports additional features:
    - Automatic OTP programming lock feature.
    - Key wrapping to store the secrets encrypted in OTP. It is only available on additional secrets, not on the preselected ones (used by ROM code). Wrapping options are used to select the targeted CID that can unwrap the key and the key used to wrap the secret. It can be derivative hardware unique key (DHUK) or boot hardware key (BHK) or an XOR using both. As the wrapping uses AES algorithms, it limits the size of the wrapped secrets to 128 secrets aligned size.
    - HDPL level (hide protection level) wrapping (only on STM32MP21x): it can protect the unwrapping of the secret at different boot stage levels.

It also provides a method to add additional secrets properly. The SecretGen panel generates the first secret file in a format compatible with the targeted device and saves it to a dedicated plaintext binary file.

3.4.1.2 Backup secrets

STM32MP2x devices support the provisioning of backup memories such as backup RAM or backup registers.

It is useful for devices that includes a backup battery, and it is mandatory to support wrapping using BHK (stored in backup registers).

There is no specific format to include data; it just requires the offset in the memory to write the given data. When BHK must be provisioned, a 256-bit key file must be provided and set at the backup register start address. It must be locked but not wrapped.

An additional file is generated thanks to this panel, which stores additional secrets.

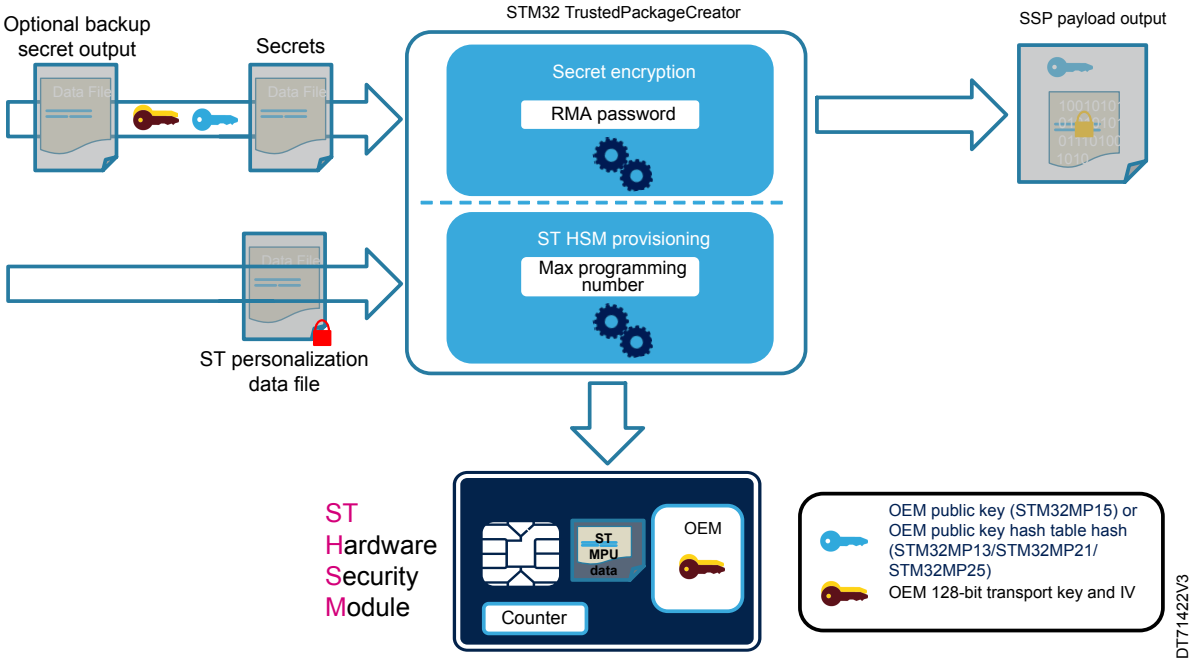
3.4.2 OEM secrets encryption and ST HSM provisioning

This part is generated by the OEM. It prepares the materials to be sent to the manufacturer. After the ECC key pair(s) generation and the secrets generation, it is sent to the STM32 TrustedPackageCreator tool [UM2238] to:

o Generate the final encrypted SSP secret file

o Provision the [STM32HSM-V2] with the associated device personalization data and the OEM 128-bit transport key (and IV).

Figure 2. **STM32 secret generation**



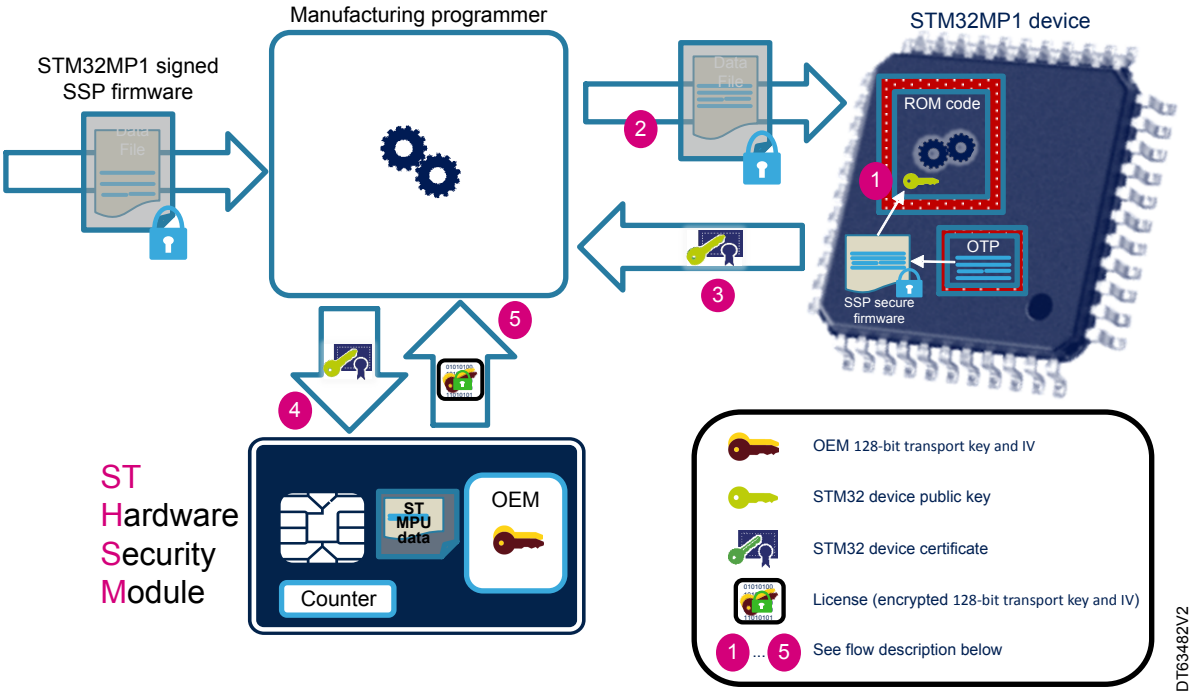
3.4.3 SSP initialization and device authentication

3.4.3.1 STM32MP1

On STM32MP1 series, the SSP firmware is required to start the SSP initialization and device authentication. It is delivered as source code and needs to be customized by the OEM.

It must have been signed with the ECC key linked to the OEM PKH for STM32MP15 or PKHTH for STM32MP13. See secure boot page in [SECURE_BOOT].

Figure 3. **STM32MP1 device authentication**



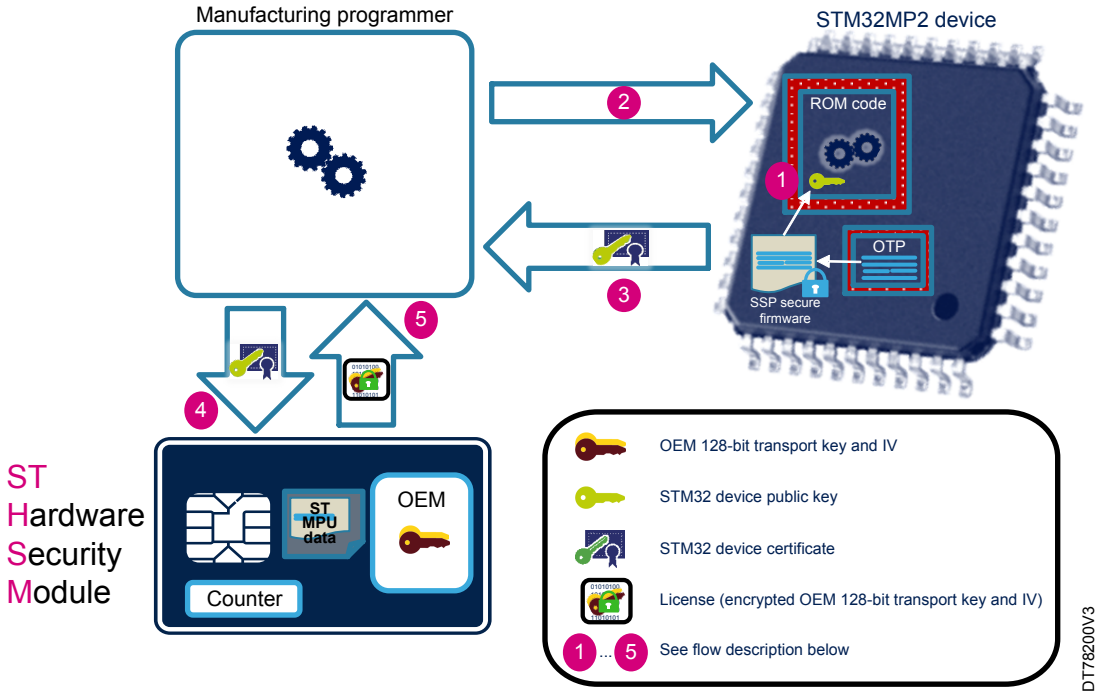
Flow description:

1. The device starts ROM code in serial boot and wait for programmer connection
2. The programmer sends the SSP firmware
   This is done twice:
   a. Once to initialize the SSP request and reset
   b. Once to reconnect to the programmer for certificate exchange
3. The SSP firmware sends the certificate to programmer
4. The programmer sends the certificate to ST HSM for device authentication
5. The ST HSM gives to the programmer the device dedicated license with the transport OEM key/IV

3.4.3.2 *STM32MP2*

On STM32MP2 series, the SSP initialization and authentication is managed directly with the ROM code.

Figure 4. STM32MP2 device authentication



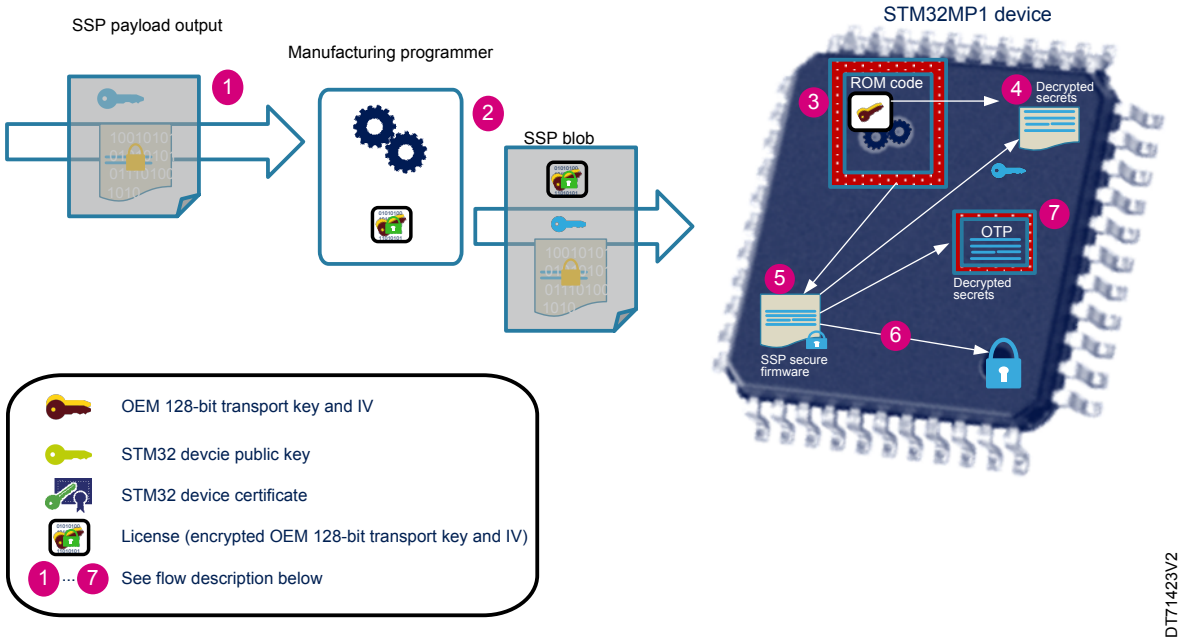
Flow description:

1. The device starts ROM code in serial boot and wait for programmer connection
2. The programmer requests the device certificate
3. The ROM code sends the certificate to programmer
4. The programmer sends the certificate to ST HSM for device authentication
5. The ST HSM gives to the programmer the device dedicated license with the encrypted OEM 128-bit transport key/IV

3.4.4 Secrets download and provisioning

3.4.4.1 *STM32MP1*

Figure 5. STM32MP1 secrets download and provisioning



Flow description:

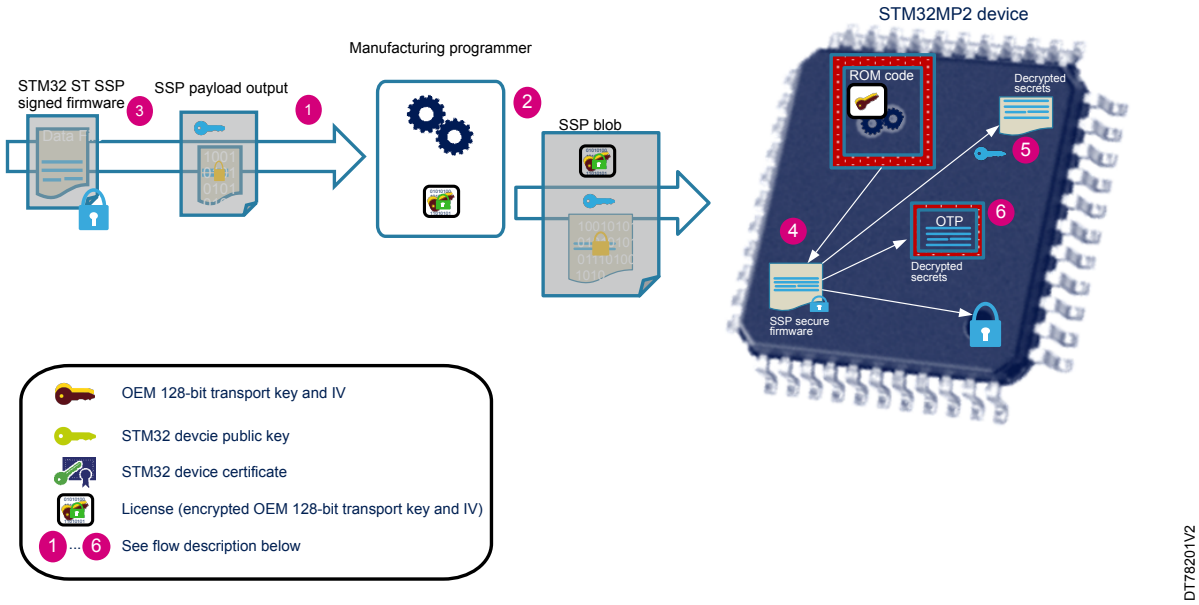
1. SSP payload output is sent to the programmer.
2. Programmer concatenates the license with the payload and sends it to the SSP firmware and resets the device.
3. ROM code decrypts the license to retrieve the OEM AES key.
4. ROM code decrypts and checks the secrets integrity.
5. ROM code authenticates the SSP firmware thanks to the OEM key (or PKHTH in STM32MP13) and jumps in SSP firmware once verified.
6. SSP firmware writes the OEM public key hash or public key hash table hash and changes the device state to secure close.
7. SSP firmware provisions the OTP fuses with the decrypted secrets. Contexts are cleaned and device is reset.

3.4.4.2 *STM32MP2*

On STM32MP2 series, the SSP firmware is a dedicated signed firmware delivered by ST. It is loaded by the programmer during the download process. It is authenticated by the ROM code, started to verify the license and provision the secrets.

The SSP firmware is delivered with the STM32MPU device certificate. The ST SSP signed firmware is specific according to the device family used.

Figure 6. STM32MP2 secrets download and provisioning



Flow description:

1. Encrypted SSP payload is sent to the programmer.
2. Programmer concatenates the generated device license with the payload and sends it to the ROM code.
3. Programmer sends the ST SSP signed Firmware to the ROM code.
4. ROM code authenticate the ST SSP signed Firmware, set the device in under provisioning state and execute the firmware.
5. SSP Firmware authenticates the license, decrypts and checks the secrets integrity.
6. SSP firmware writes the secrets in OTP, clears the context and changes the device state to secured_locked if the process completes successfully. In case of an error, the device remains in the under_provisioning state.

3.5 SSP process steps

The SSP processing is split in different phases:

- SSP initialization (only on STM32MP1 series)
- SSP device authentication
- SSP secret decryption and secret programming

The SSP process can only be run in serial mode.

On STM32MP1 series, the process is managed between the secure firmware and the ROM code, exchanging information using secure SYSRAM area which requires a device reset between each step. The SYSRAM must remain supplied during all transitions to mantain operational integrity and ensure the context exchange remains secure.

On STM32MP2 series, this is a one-way processing. The authentication is owned by the ROM code which exchanges the certificate with the programmer tools. The ST SSP signed firmware is downloaded with the device secrets. The ROM code detects the ST SSP firmware during authentication request and change the device's life cycle. The ST SSP loaded firmware verify the license, decodes the secrets and programs the secrets.

A maximum number of SSP attempts is authorized on the device to avoid potential hack (STM32MPU limited to 4 SSP attempts).

3.5.1 SSP initialization

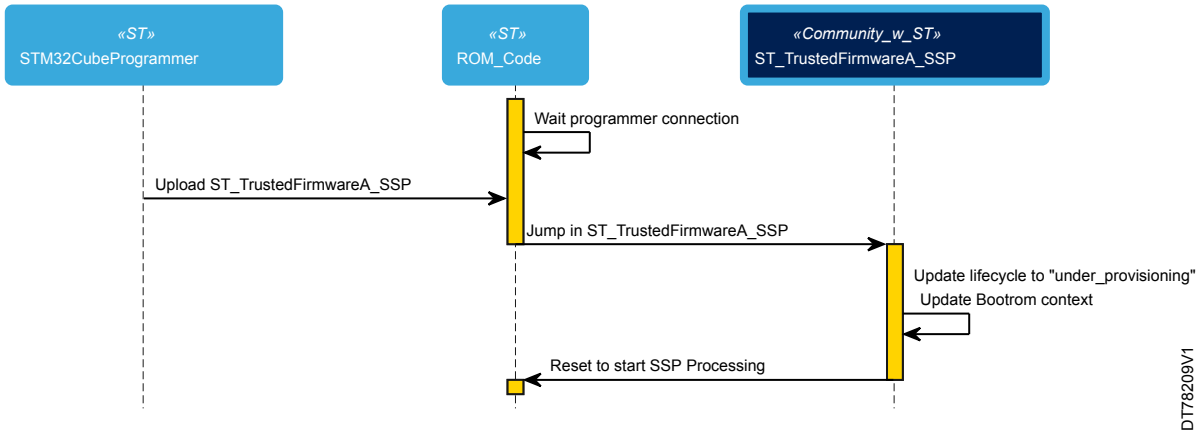
This step is only required on STM32MP1 series.

This step is the entry point to start the SSP process. It is directly started using the SSP secure firmware. It consists of changing the device life cycle to *under_provisioning* state and send the SSP start command to the ROM code using the SYSRAM exchange context.

By changing the device life cycle in *under_provisioning* state, the ROM code and the SSP firmware can access specific information.

On STM32MP1 series, the SSP secure firmware is used to start the SSP sequence. It is downloaded by the programmer and loaded by the ROM code (configured in serial boot mode). It automatically programs the SSP OTP request bit (if no SSP sequence previously started) and fills the ROM code context with the start command. It requires a software reset to switch to the next sequence. Because the exchange is done using a SYSRAM command exchange, the SYSRAM must be kept supplied during reset. At this stage, the SSP firmware is not yet authenticated because it does not manage any critical information.

Figure 7. STM32MP1 SSP initialization



On STM32MP2 series, this step is skipped as the SSP process can be directly started without specific initialization.

3.5.2 SSP device identification and encrypted secret file upload

The SSP identification can be started when the device life cycle is in *secured_unlocked* or *under_provisioning* state. It is no more available when the device is in *secured_locked* state.

3.5.2.1 STM32MP1

This step requires a software reset. The device is still in serial boot mode and the SSP request has been set.

The programmer must download one more time the SSP signed firmware. To allow the device identification, a certificate is required. The SSP firmware prepares the device certificate that is sent to the programmer for authentication.

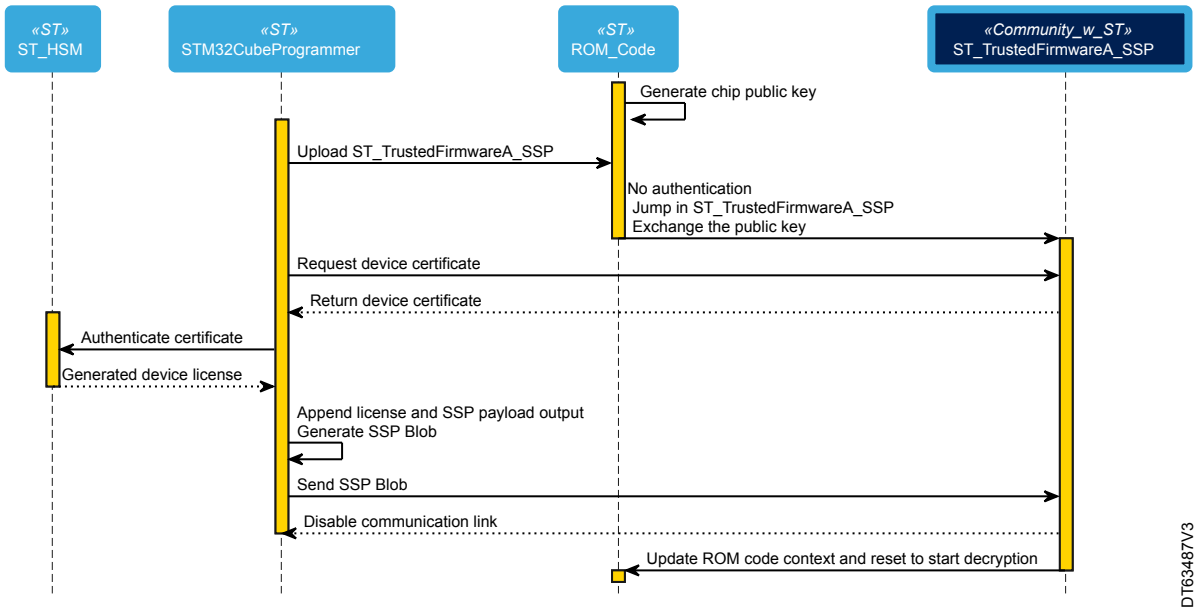
The programming tool downloads the SSP firmware again and it is executed without authentication.

The SSP secure firmware creates the 136-byte certificate using OTP values and generated ECC public key. The certificate is downloaded by the host tool using a dedicated command.

The device certificate is sent to the ST HSM connected to the programming tool (ex: STM32CubeProgrammer) to authenticate the STM32MPx device. A specific device license is generated if the certificate has been properly identified. Each generated license decreases the programming counter number in the ST HSM.

When receiving the device license, the programming tool can upload it, appended with the OEM encrypted secret file previously generated by OEM. Both are uploaded to the device in a single command request, and stored in internal secure memory. When the package is completely received, the SSP secure firmware stops the serial connection and sets a new magic value in the ROM code context to complete the current state. A software reset is automatically initiated in the last step.

Figure 8. **STM32MP1 device certificate and secret upload**



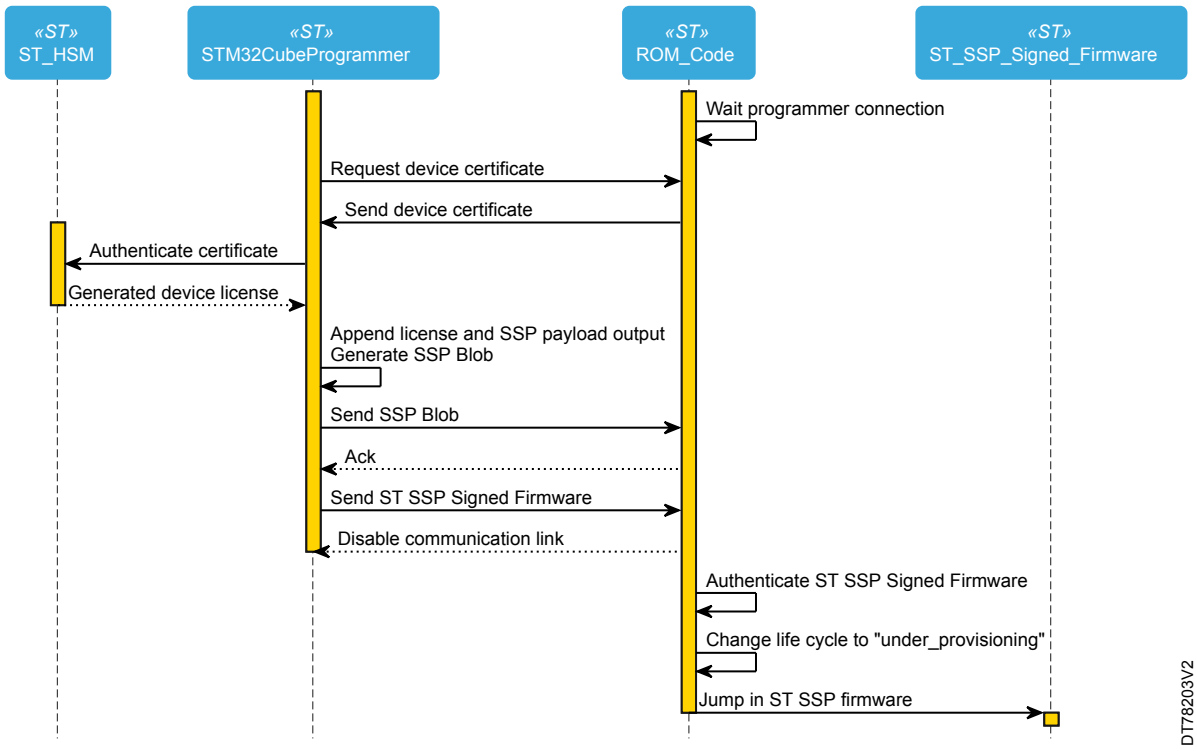
3.5.2.2 *STM32MP2*

The device is booted in serial boot mode; the programmer can require the device certificate (136-byte) to identify the STM32MP device. This request can be done without any limitation.

Once received, as for the STM32MP1 series, the certificate is sent to the ST HSM to generate the associated device license. If authentication is confirmed, the generated device license can be appended to the SSP secret file and sent to the device.

To finalize the upload process, the programmer must also download the ST SSP signed firmware. Downloading the firmware completes the secret authentication step, disconnects the programmer link, authenticates the ST SSP firmware and switches the life cycle to *under_provisioning* state. When this state is reached, the device remains locked in this specific life cycle, which does not allow any futher programming, with the exception of uploading again the secret and the ST SSP signed firmware (in case of potential error during the process).

Figure 9. **STM32MP2 device certificate and secret upload**



At this stage, all the mandatory binaries are uploaded in internal memories and the next step is automatically started.

3.5.3 Secret decryption and OTP programming

This is the final step of the SSP process. This critical sequence is made in a fully closed secure environment. All external accesses are closed during this step including loaded interface and debugger.

3.5.3.1 STM32MP1

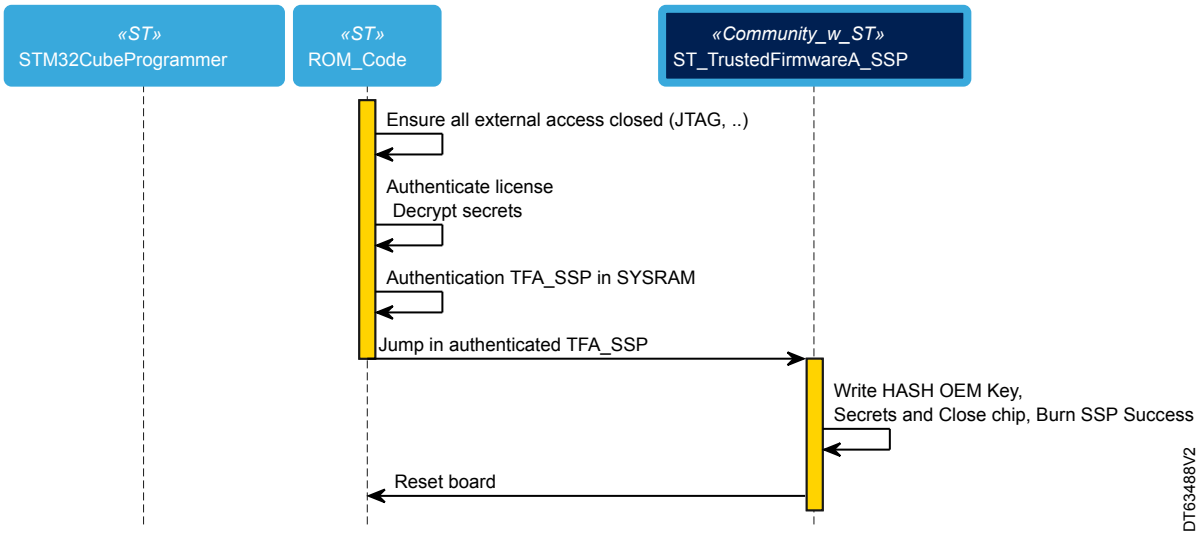
On STM32MP1 series, it starts after an automatic system reset. The ROM code:

- Controls the license (stored in SYSRAM during the previous step) and decrypts the transport key and IV from ST HSM generated license.
- Decrypts the secrets (stored in SYSRAM during the previous step) using the transport key and checks the encryption tag thanks to the AES-GCM algorithm.
- If the check is correct, the ROM code is able to authenticate the SSP secure firmware still resident in SYSRAM secure with the OEM public key from the encrypted secret file.
- If authentication is confirmed, the ROM code gives the decrypted secrets to the SSP secure firmware and jumps in it.

The SSP secure firmware:

- Checks the OTPs are free for programming,
- Programs the OEM public key hash or the OEM public key hash of table hash,
- Programs the OTP secure close device to force firmware authentication,
- Programs the secrets in the OTP,
- Programs the SSP success bit to complete the OTP processing,
- Cleans all secrets in the secure SYSRAM memory,
- Resets the target.

Figure 10. **STM32MP1 secrets decryption and programming**



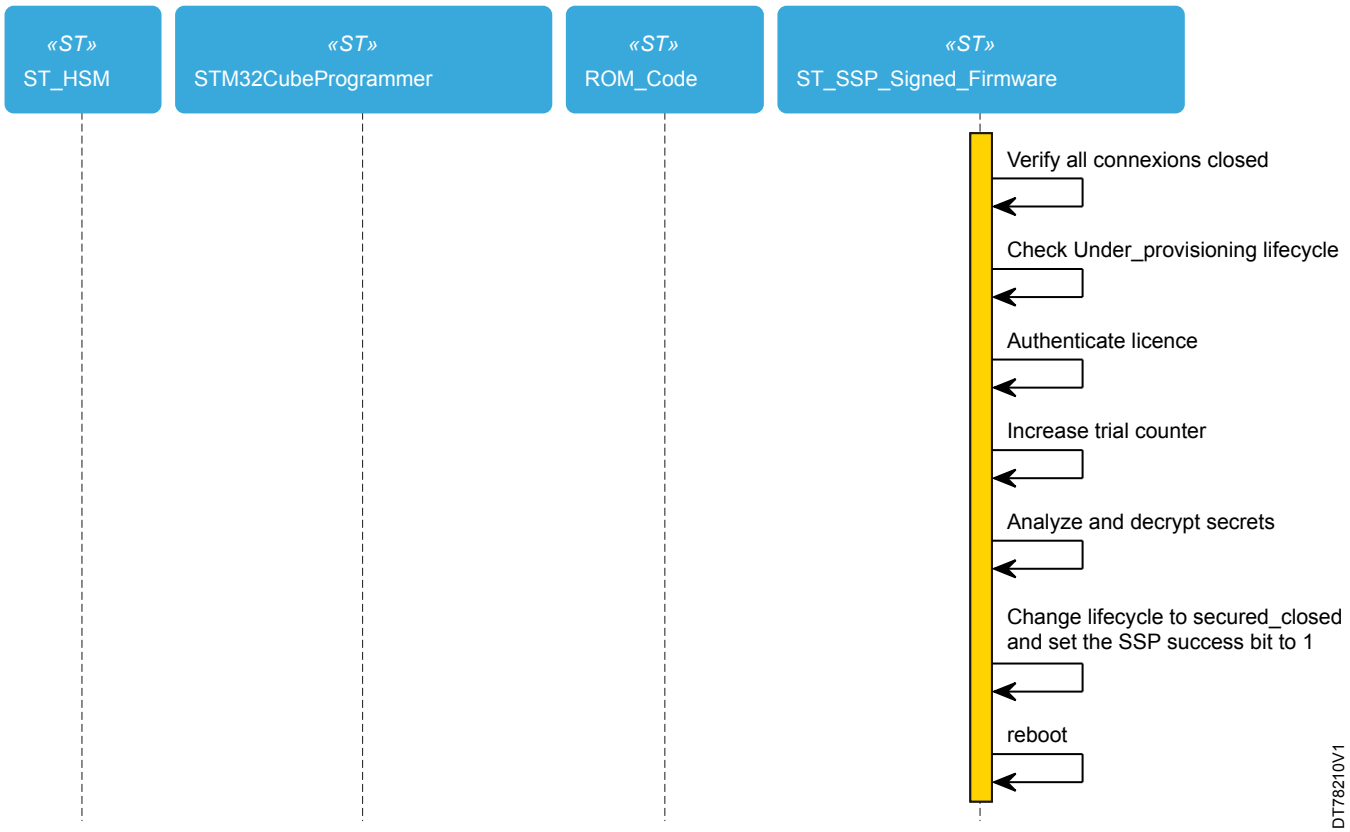
Upon completion of this sequence, the device is fully secure, secure boot authentication is mandatory based on the OEM public key used, the debugger is closed by default and secrets are available for the secure domain.

3.5.3.2 *STM32MP2*

On STM32MP2 series, it starts by jumping in the ST SSP firmware which:

- Controls that all external connections are closed,
- Checks the device life cycle which must be in "under_provisioning" state,
- Controls the license and decrypts the transport key and IV from ST HSM generated license,
- Increases the trial counter and controls that it is less than four trial,
- Analyzes the blob sections and decrypts the secrets (stored in RETRAM during the previous step) using the transport key and checks the encryption tag thanks to the AES-GCM algorithm,
- Programs the additional backup memories if any,
- Programs the OTP at the end, after all backup memory areas. Checks they are free for programming,
- Changes the life cycle to closed_locked,
- Cleans all secrets in internal memory,
- Resets the target.

Figure 11. **STM32MP2 secret decryption and programming**



3.5.4 Error cases

In case of reset or error during the two first steps, the SSP processing can be relaunched without any attempts control.

In case of error during the last step (SSP firmware authentication error, corrupted license, corrupted secrets file), the attempt counter is incremented till it reaches the maximum tries. The STM32MPx allows four tries, once achieved, the SSP processing cannot be fully ended.

4 Security concern

Once ST information (certificate) has been provisioned to each device, the OEM can rely on secure secrets provisioning and let the OEM-CM populating the devices with their secrets.

The secrets prepared by the OEM are of three different types:

- RMA password: unlock/relock on STM32MP15 and STM32MP2 series, unlock on STM32MP13.
- Root of trust: OEM public key on STM32MP15 or hash of the table hash of the public keys on STM32MP13 and STM32MP2 series.
- OTP secrets

Before the SSP execution, the device is in a secured_unlocked state (no secure boot). The SSP is in charge of switching automatically the devices to the secured_locked state that forces the secure boot authentication.

4.1 Return material authorization (RMA)

Refer to [AN5827].

In case of device failure, STMicroelectronics must provide a way to switch the device into secure open mode to be able to run tests without seeing any secret.

This mechanism reopens the device (RMA unlock) from a secure close state and hides the upper OTP (secrets). On STM32MP15 and STM32MP2 it is possible to lock it again thanks to RMA relock password. The RMA process is protected by a password (or 2 passwords for STM32MP15). Password(s) must be set by the OEM, as part of his secrets during the provisioning step.

The OTP RMA is read protected by the ROM code and cannot be accessed later. It is mandatory to define a password with a valid (non-zero) value.

4.2 OEM authentication key(s)

OEM authentication key(s) is a major part of the secure boot sequence. Secure boot is ECDSA verification based, the authentication key is used to validate the signature of the loaded binary.

The key hash on STM32MP15 or key hash table hash on STM32MP13 and STM32MP2 series (to manage multiple keys) is stored into the device to be used as a reference during the secure boot authentication process. On STM32MP15, the ROM code must check the key against the hash that has been safely programmed in OTP. On STM32MP13 and STM32MP2 series, the key is verified by checking the hash in the STM32 header and verifying the hash table from the header against the hash table hash stored in OTP.

4.3 OTP secrets

Other secrets are chosen by OEM and can be used in secure environment.

It can be passwords, keys, all kind of sensitive information that must be only managed by the secure software and protected in a hardware secure area.

5 Secret preparation

A secret file must be created prior to SSP processing. This secret file must fit into the OTP area reserved for customers. The file can be created using the STM32 Trusted Package Creator tool [UM2238].

The OTP memory is organized as 32-bit words.

On STM32MP1 series, secrets are limited to OTP memory.

On the STM32MP1 series:

- One OTP word is reserved for RMA password(s): OTP 56.
- 37 free words are reserved for customer usage; the secret size can be up to 148 bytes: OTP 59 to 95 (OTP 57 and 58 are defined as reserved for MAC address and cannot be provisioned using SSP).

The 37 free words include specific areas such as OEM secret for encrypted boot which must be set in OTP 92 to 95 on STM32MP13.

A 148 bytes binary OTP secret file must be used as the reference to construct the secret file.

On the STM32MP2 series, the secrets can be made of OTP areas and/or backup areas. The secret generated file extends the SFI initial structure and adds specific areas for OTP and backup memories.

The secret area in OTP contains mandatory assets such as RMA password, PKHTH, encryption keys.

In OTP sections:

- 4 OTP words are reserved for RMA password(s): OTP 256, 257, 258, 259 (required).
- 2 sets of 8 OTP words are reserved for PKHTH for 2 OEM keys management (first set is required).
- 2 sets of 4 OTP words are reserved for EDMK for the 2 OEM keys management (not mandatory).
- 100 OTP secret words on STM32MP25/23x are free to the OEM.
- 88 OTP secret words on STM32MP21 are free to the OEM.

Additional OTP words located in lower and middle section can be provisioned directly through the OTP area.

Thanks to SFI format, some additional areas to manage provisioning of backup memories (such as backup registers and backup RAM) are available. It requires a battery to maintain the device V_{BAT} and keep this part available during power down.

5.1 RMA password(s)

The RMA password(s) are chosen by OEM. There are part of the secret file and placed as the first 4-byte word.

On the STM32MP15xx, the RMA passwords are provided using 15 bits (maximum value is 0x7FFF).

The two passwords are contained in the same 32-bit word:

Table 3. STM32MP15 RMA passwords

Bit [31-30]	Bit [29-15]	Bit [0-14]
Not used	RMA relock	RMA unlock

Examples:

RMA unlock: 0x3210

RMA relock: 0x7654

Final OTP: 0x3B2A3210 (8-bit coding style: 10 32 2A 3B)

On the STM32MP13xx, the RMA password is provided using 32 bits.

Table 4. STM32MP13 RMA password

Bit [31-0]
32-bit password

Examples:

RMA unlock: 0x87654321

Final OTP: 0x87654321 (8-bit coding style: 21 43 65 87)

Table 5. **STM32MP2x RMA password**

Bit [31-0]
OTP_RMA_LOCK_PSWD0
OTP_RMA_LOCK_PSWD1
OTP_RMA_LOCK_PSWD2
OTP_RMA_LOCK_PSWD3

Examples:

RMA 128-bit password: 0xAABBCCDDEEFF55667788990011223344

Final OTP: 0xAABBCCDD 0xEEFF5566 0x77889900 0x11223344 (8-bit coding style: DD CC BB AA)

To simplify the creation of RMA passwords and prevent formatting issues, use the STM32TrustedPackageCreator tool.

5.2 OTP secret file

On STM32MP1 series, the secret file must be a 148-byte binary file format. It is created from key files, certificates or other data required by OEM to run its secure software.

The file represents the OTP area referenced as “free for user”. The OTP fuses are accessible through 32-bit BSEC registers, so the padding (free words) is allowed by adding 0 in 32-bit word aligned.

Table 6. **STM32MP1 OTP area examples**

OTP [59:66]	OTP 67	OTP 68	OTP [69:95]
256-bit key	4-bytes padding (0)	16-bit password1 / 16-bit password2	….

This file can be easily generated thanks to the STM32 Trusted Package Creator tool [UM2238] that exposed the OTP area available for secrets. It automatically generates the correct binary format for the encryption step.

On STM32MP2 series, the secret file contains OTP and the associated parameters. STM32MP2 series offers some new functions that allow the user to:

- Lock the OTP after programing
- OTP key wrapping
- OTP encryption

The wrapping and encryption implement AES-ECB, AES-CBC or AES-CTR (only for encryption) modes and allow the user to encrypt data for a specific processor thanks to CID (compartment ID).

On STM32MP21x, it is possible to add a specific control of the temporal level of usage (HDPL) that manage a specific key wrapping that might protect the secret to be used at a specific boot stage.

The STM32 Trusted Package Creator tool [UM2238] implements the secret file generator based on OTP mapping. It allows the user to select the OTP address, the encryption mode, and lock.

Note: In the secret files, bytes set to 0 and aligned on 32-bit word are ignored during the OTP programming.

5.3 Backup secret file

Backup secret files are only available for STM32MP2 series. It allows the OEM to provision the backup memories such as backup RAM or backup registers during the production.

It requires to have a V_{BAT} power supply connected on the board to ensure that the backup domain is maintained during low power.

The backup secret files can be multiple as they contain pure data sections defined/located at OEM chosen offsets inside the memory.

For backup areas, the data must consist of 32-bit aligned words to match the backup register topology.

Example: To provision the boot hardware key (BHK) located in the first backup registers, the start offset must be 0 at the address 0x46010100 (base address of the backup register section).

The STM32 Trusted Package Creator tool [UM2238] implements the backup file generator. It allows the user to select the binary file and the offset address, the encryption mode, and to lock it (if available). The tool detects the memory overlap and generates the final output file.

5.4 STM32MP1 encrypted secret file

The encrypted secret file follows a specific layout that guarantees a secure transaction during transport and decryption.

The STM32TrustedPackage tool creates an .ssp file as described below:

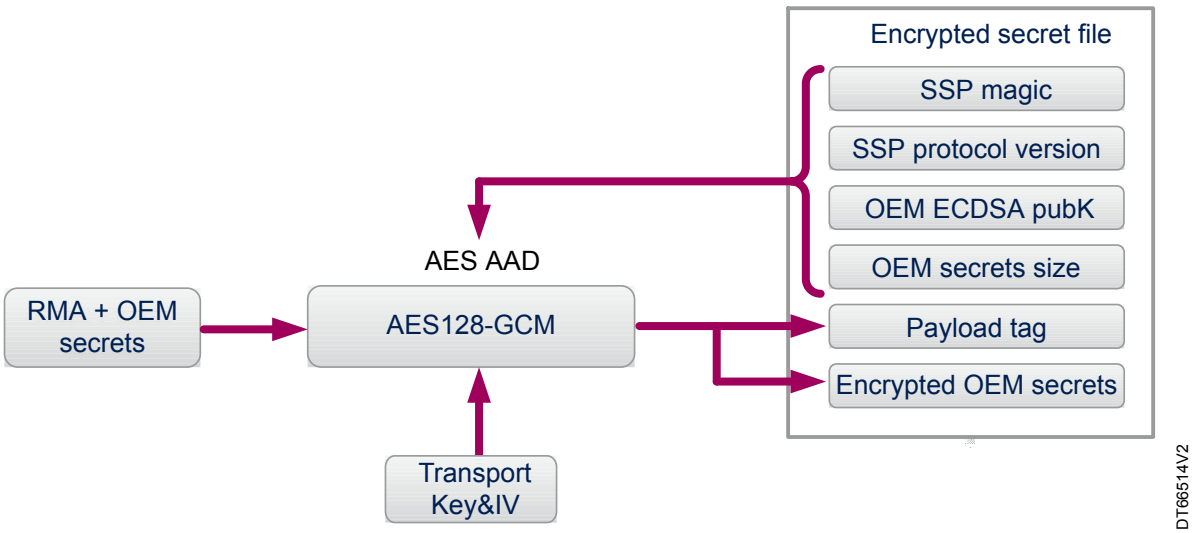
Table 7. Encrypted secret file layout

-	Size	Content
SSP magic field	4 bytes	'SSPP': magic identifier for SSP payload,
SSP protocol version	4 bytes	SSP version used,
OEM ECC public key or OEM public keys HASH table HASH	64 bytes (STM32MP15) 32 bytes (STM32MP13)	OEM ECC public key (STM32MP15) or OEM ECC public keys hash table hash (STM32MP13)
OEM secret size	4 bytes	Size of OEM secrets, in bytes,
Payload tag	16 bytes	Cryptographic "signature" of all fields above, to ensure their integrity (AES GCM tag),
Encrypted OEM secrets	152 bytes	Encrypted OEM secrets. Including RMA password (4-bytes), excluding 57 and 58 OTP defined as MAC ADDR.

The first layout part (SSP identification fields, ECC public key or public key hash table hash, and secret size) is used as additional authenticated data (AAD) to generate the payload tag. This is checked by the ROM code during the decryption. It ensures the integrity of the downloaded data. The generated encrypted secret file usage is limited to a specific ST HSM as the transport key and IV used for encryption are stored in the HSM.

This encrypted file is generated by STM32TrustedPackageCreator tool.

Figure 12. Encryption file scheme



5.5 STM32MP2 encrypted secret file

The STM32MP2 encrypted secret file is using the 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 binary format 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 OTP area containing the OTP secrets to be programmed when the SSP is complete.

The STM32TrustedPackage tool creates an .ssp file including the OTP area and optional backup areas.

6 SSP secure firmware

The SSP process requires specific firmware to manage the provisioning. This firmware is loaded by the ROM code and executed in the under_provisioning life cycle to program the secrets.

6.1 STM32MP1

The SSP secure firmware is derived from the standard secure firmware (see [TrustedFirmware-A]) used as the first stage bootloader from the STM32MPx ecosystem release. It includes additional features for the SSP processing.

It reuses the serial boot protocol (USB and UART) from the generic code. The major change is the SSP library that manages the different SSP phases. Other updates are linked to size optimization to reduce the binary to only the required functions.

6.1.1 Build

The build instructions are described in the TF-A wiki page, see [SSP].

6.1.2 Signed binary

The SSP firmware must be STM32 signed, thanks to STM32 signing tool [UM2543], with the OEM private key to ensure the OEM firmware authenticity. It is authenticated by the ROM at the final SSP stage based on the HASH calculated from the OEM public key (STM32MP15) or provided with the OEM secrets (STM32MP13).

Without this authentication, the SSP processing fails.

6.2 STM32MP2

On STM32MP2, the SSP secure firmware is a dedicated ST firmware which is signed and encrypted with specific ST keys. It is ready to use and does not need any customer customization. It is delivered with [STM32MPUSSP-UTIL] that contains the firmware and device personalization package.

7 SSP tool management

On each SSP install step, the user must use the STM32 ecosystem tools to manage the secure programming and the SSP flow.

Overall, there are three main steps to perform using SSP tools:

- The encrypted secrets file generation
- The ST HSM provisioning
- The SSP procedure with STM32TrustedPackageCreator tool

7.1 Secret files generation

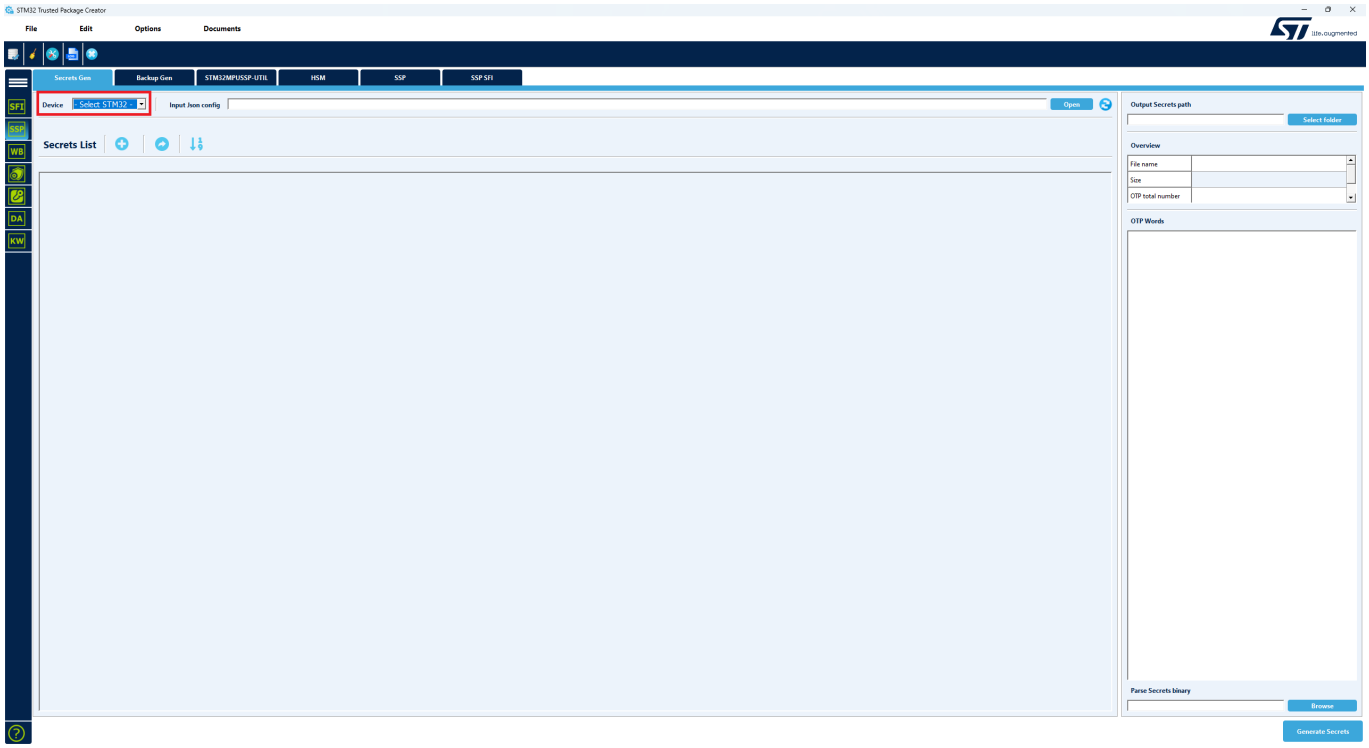
The secret files generation is provided by the STM32 Trusted Package Creator tool with its ready for use secret files graphical user interface. The tool provides two different panels:

- One for the OTP secret file
- One for backup file generation

7.1.1 Secrets Gen panel

This panel is the one dedicated to OTP secrets management. It allows OEM to easily create the well-formatted file that is injected in the next encryption panel.

Figure 13. Secret Gen panel

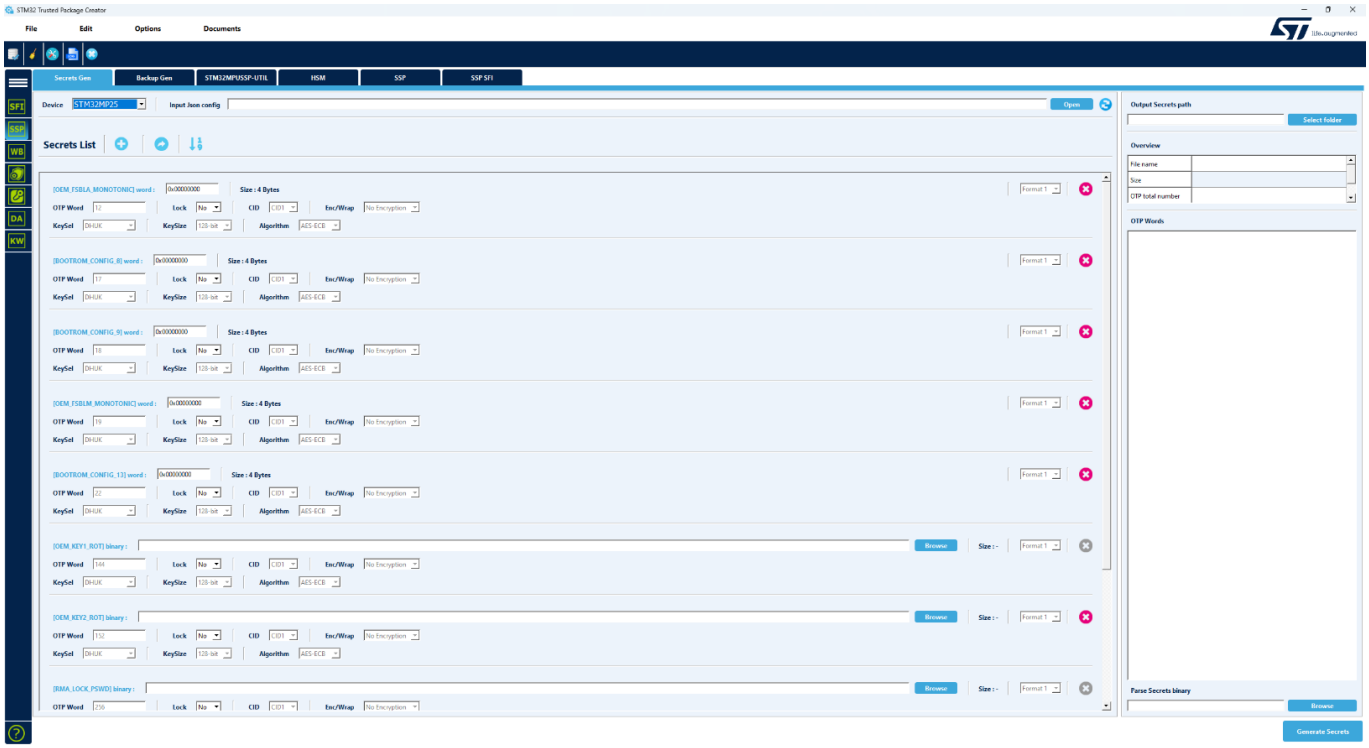


The first step is to select the device to be provisioned. By selecting the device, the associated mandatory or recommended fields are listed in the panel.

Additional secrets can be manually added by the user by clicking the + button. It is added to the list of secret that are embedded in the generated file.

It is also possible to open a json file that described a previous exported secret topology that could be reused to just update the secret values.

Figure 14. **Secret Gen for STM32MP25**

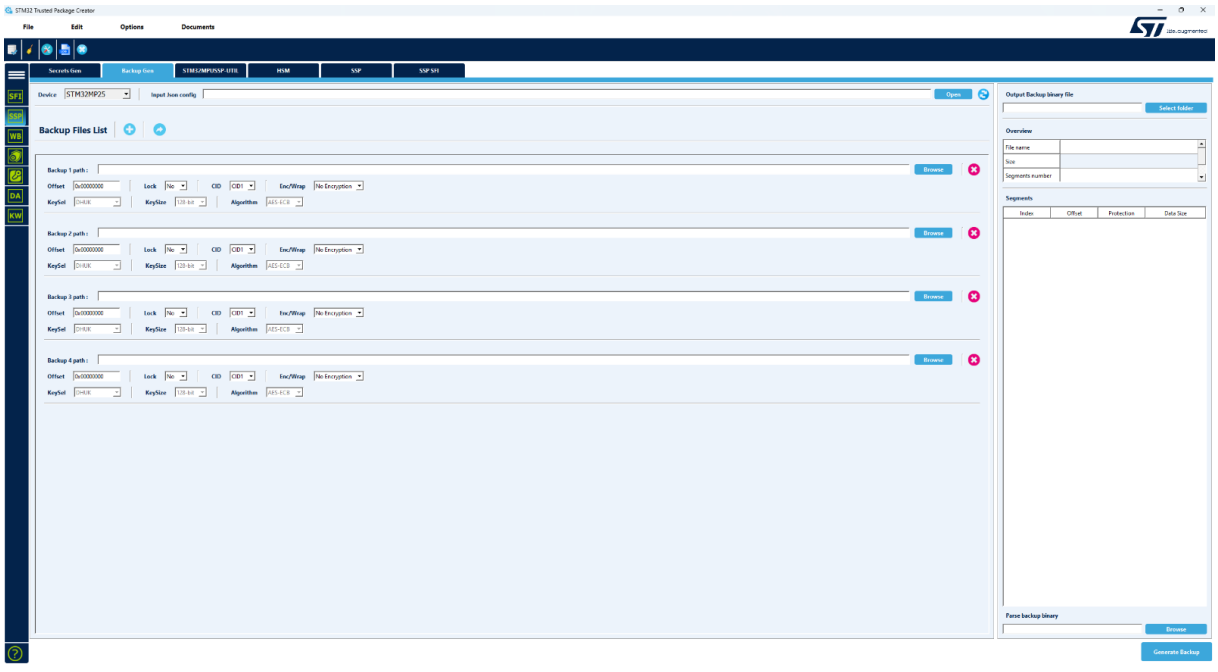


Once the panel has been completed, the user generates the file by clicking the *"Generate Secrets"* button. The tool generates a binary file with secret in plain text. The file is just format to embed the secret to be later imported in the SSP or SSP SFI panel.

7.1.2 Backup Gen panel

The Backup Gen panel gives the way to create backup memory files to provision memories at production time. The device must be selected to ensure that the correct options are displayed in the panel.

Figure 15. **Backup Gen Panel**



By adding new parts, the Backup Gen panel shows new entries in the list that need to be inserted at the correct offset address (absolute address) and manages to wrap the secrets for the corresponding CID.

Once the panel has been completed, the user generates the file by clicking the *"Generate Backup"* button. The tool generates a binary file with secret in plain text. The file is just format to embed the secret to be later imported in the SSP or SSP SFI panel.

7.2 Encrypted secrets file generation

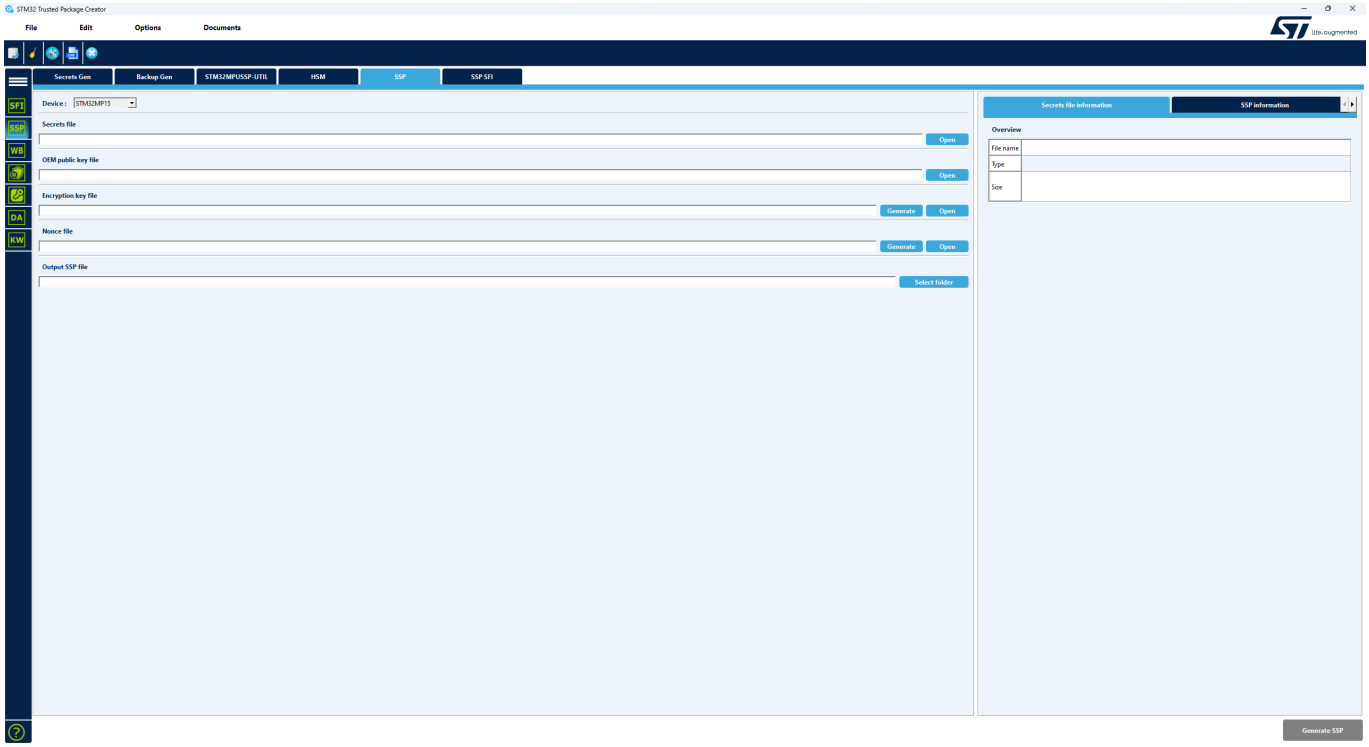
The encrypted secrets generation is provided by the STM32 Trusted Package Creator tool with its graphical user interface and command line interface to generate a SSP file ready for use.

The tool usage is described in the STM32 Trusted Package Creator User manual [UM2238].

7.2.1 STM32MP1 encrypted secrets file generation with graphical user interface

The STM32 TrustedPackageCreator tool GUI presents an SSP tab. In order to generate an encrypted secret file, the user must fill in the input fields with valid values.

Figure 16. STM32TrustedPackageCreator SSP GUI tab for STM32MP1



Device: choose STM32MP13 or STM32MP15

Secrets file: binary file of size 148 bytes to be encrypted. Can be selected by entering file path (absolute or relative), or by selection with the **Open** button.

(STM32MP13) OEM root public key table hash: *publicKeysHashHashes.bin* file (32-bytes).

(STM32MP15) OEM public key file: *public_key.pem file* (178 bytes).

Encryption key and nonce files: the encryption key (transport key) and nonce file can be selected by entering their paths (absolute or relative), or by selection with the **Open** button. The **Generate** button can be used to directly generate a random value (16-bytes for key, 16-bytes for nonce).

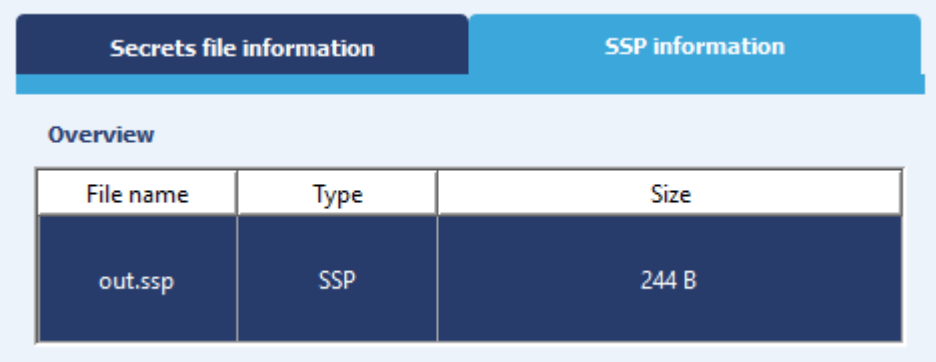
Output SSP file: selects the output directory by mentioning the SSP file name to be created with .ssp extension.

When all fields are properly filled in, the user can start the generation by clicking on the **Generate SSP** button (the button becomes active).

Once the generation is done, the user can get SSP information from the SSP overview section.

It generates a 212-bytes file on STM32MP13 and a 244-bytes file on STM32MP15.

Figure 17. **SSP output information**

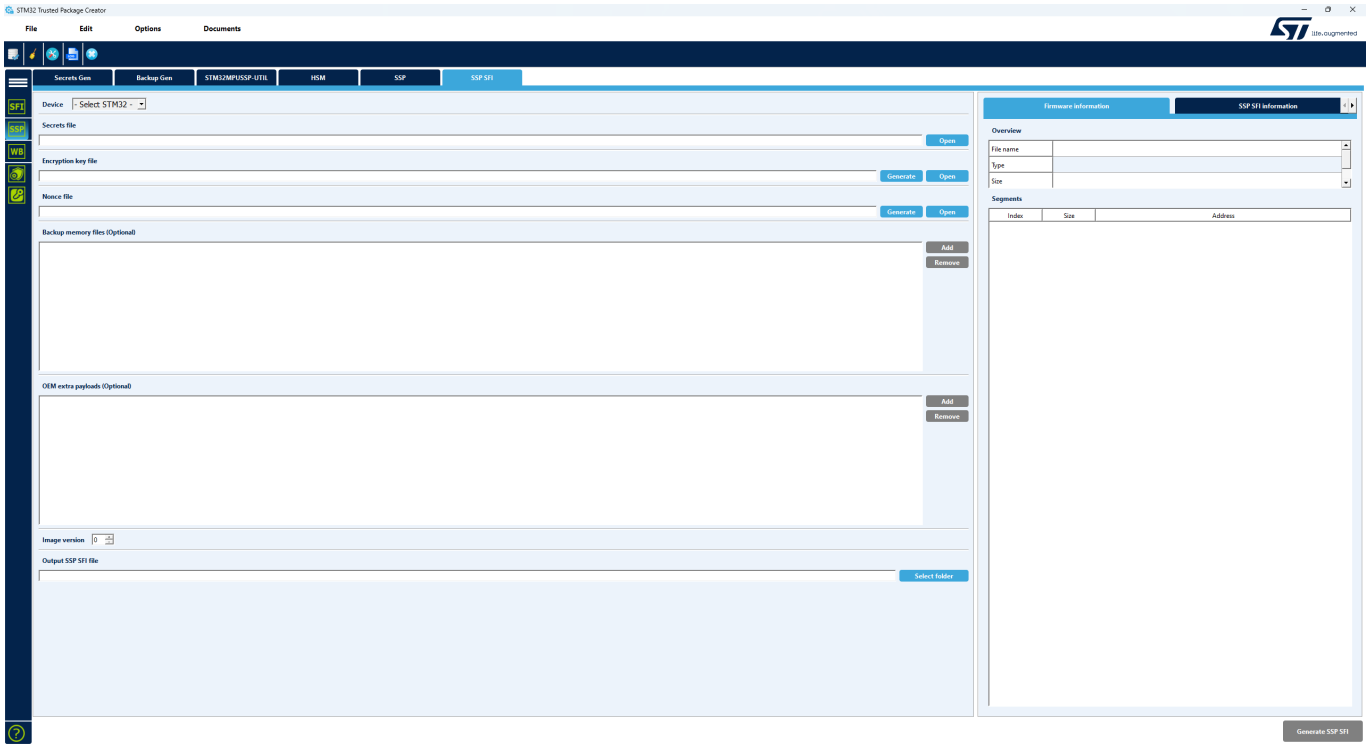


- **File name**: SSP output file name.
- **Type**: SSP format.
- **Size**: indicates the generated file size including all data fields.

7.2.2 STM32MP2 encrypted secrets file generation with graphical user interface

The STM32 Trusted Package Creator tool GUI presents an SSP SFI tab. In order to generate an encrypted secret file, the user must fill in the input fields with valid values.

Figure 18. **STM32TrustedPackageCreator SSP SFI GUI tab for STM32MP2**



Device: choose STM32MP2x device

Secrets file: binary file generated within the Secret Gen panel (contains OTP secrets). Can be selected by entering the file path (absolute or relative), or by selection with the **Open** button.

Encryption key and nonce files: the encryption key (transport key) and nonce file can be selected by entering their paths (absolute or relative), or by selection with the **Open** button. The **Generate** button can be used to directly generate a random value (16-byte for key, 16-byte for nonce).

Backup memory files (optional): Select additional backup memory files generated in the Backup Gen panel. Click Add to select files or click Remove to delete a file from the list. Multiple files can be added.

OEM extra payloads (optional): Reserved for future use.

Image version: Image version of the SFI file.

Output SSP SFI file: Select the output directory by mentioning the SSP file name to be created with .ssp extension.

When all fields are properly filled in, the user can start the generation by clicking on the **Generate SSP SFI** button (the button becomes active).

Once the generation is done, the user can get SSP SFI information from the SSP SFI information giving the overview and the different segments (area) created in the ssp-sfi file.

7.2.3 Encrypted secrets file generation with the command line interface

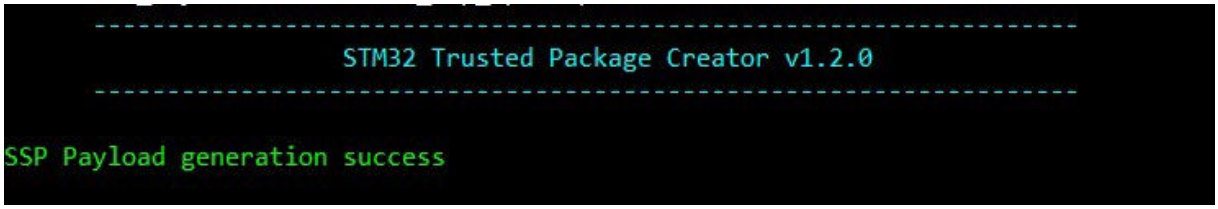
The STM32 Trusted Package Creator tool CLI exports an SSP command with various options to generate the encrypted secrets. Detailed commands can be found in the user manual [UM2238].

Command example (for STM32MP15xx):

```
STM32TrustedPackageCreator_CLI -ssp -ru 0x312 -rr 0xECA
-b "C:\SSP\secrets\secrets.bin"
-pk "C:\SSP\OEMPublicKey.pem" -k "C:\SSP\key.bin"
-n "C:\SSP\nonce.bin" -o "C:\out.ssp"
```

Once the operation is done, a green message is displayed to indicate that the generation is finished successfully; otherwise, an error occurred.

Figure 19. SSP generation success



7.3 ST HSM provisioning

This section describes the steps required to configure an ST Hardware Secure Module (ST HSM) to generate the firmware licenses for STM32 secure programming using the STM32 trusted package creator GUI and CLI tools.

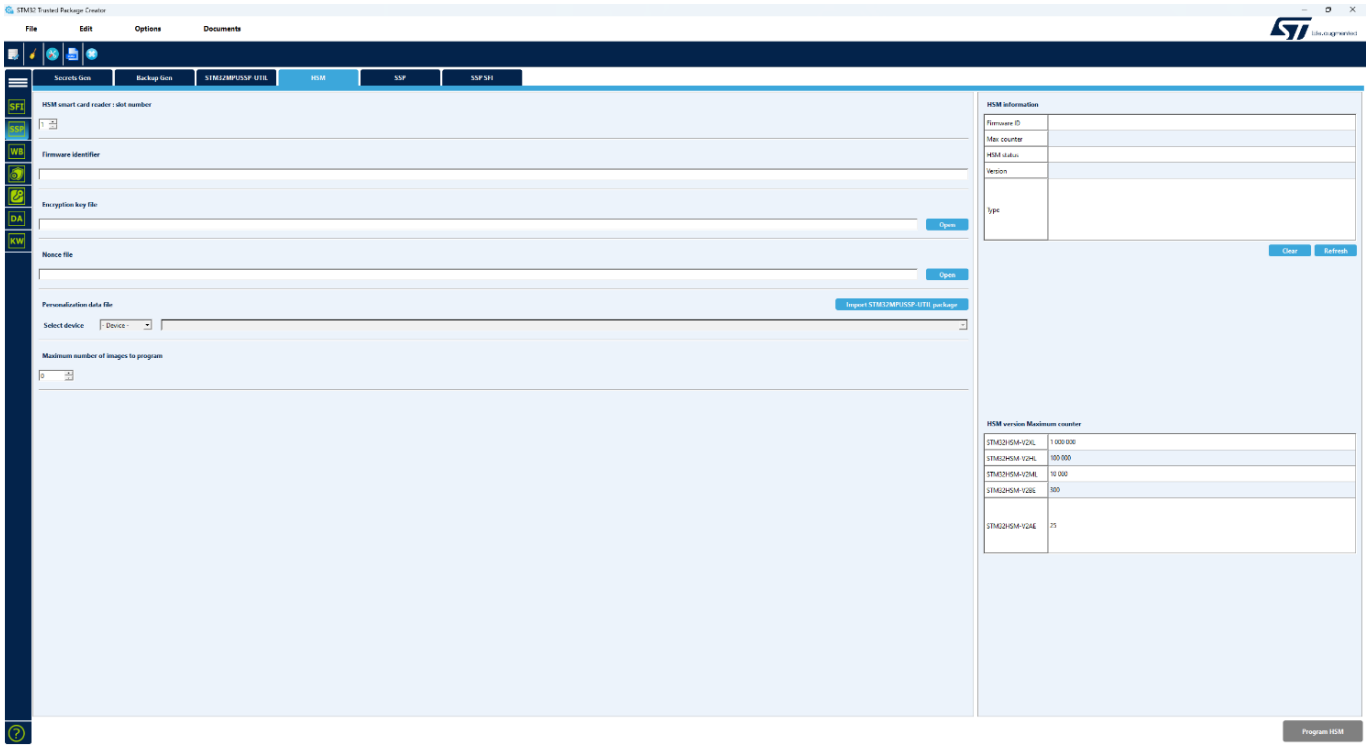
This ST HSM version 2 allows the generation of the firmware licenses targeting STM32MPx secure programming devices that are chosen via a personalization data at the OEM site.

7.3.1 ST HSM provisioning with the graphical user interface

When an OEM needs to deliver an ST HSM to a programming house to be deployed as a license generation tool for a relevant STM32 device programming, the OEM must perform some customization on his ST HSM first. He must program the ST HSM with all the data needed for the license scheme deployment in the production line. This OEM data is:

- **Counter**: the counter is set to a maximum value that corresponds to the maximum number of licenses that can be delivered by the ST HSM, it aims to prevent over programming. It is decremented with each license delivered by the ST HSM. No more licenses are delivered by the HSM once the counter is equal to zero. The maximum counter value must not exceed the predefined maximum value (1 million units).
- **Encryption key (transport key)**: this key is 32 bytes and is composed of two fields, the initialization vector (IV) or nonce (first field), and the key (last field) that are used to AES128-GCM encrypt the binary and generate the SSP file. Both fields are 16-byte long, but the last 4 bytes of the nonce must be zero (only 96 bits of the nonce are used in the AES128-GCM algorithm). Both fields must remain secret, so they are encrypted before being sent to the device. The key and the nonce remain the same for all licenses for a given piece of firmware. However, they must be different for different firmware or different versions of the same firmware. Consequently, ST HSM must be changed.
- **Firmware identifier**: identifies the correct ST HSM for a given firmware.
- **Personalization data**: 108 bytes that are encrypted before being sent to the device in cases where each device has its own configuration. It must be chosen according to the device family and revision present in the STM32Trustedpackage creator delivery.

Figure 20. **ST HSM programming tab**



The tab parameters are as follows:

- **ST HSM card index**: specifies the smart card reader slot number.
- **Firmware identifier**: identifies the correct ST HSM for a given piece of firmware.
- **Encryption key file**: a binary file containing the transport key used to encrypt the SSP output file. The key is 16-byte long. It can also be generated.
- **Nonce file**: a binary file containing the nonce used to encrypt the SSP output file. The nonce is 16-byte long. It can also be generated.
- **Select device**: Using the STM32MPUSSP-UTIL package, select the personalization data used to identify the device.
- **Maximum counter**: the maximum number of licenses that can be delivered by the ST HSM (it aims to prevent over programming).

When all fields are properly filled in, the user can program the ST HSM file by clicking on the program ST HSM button (the button becomes active).

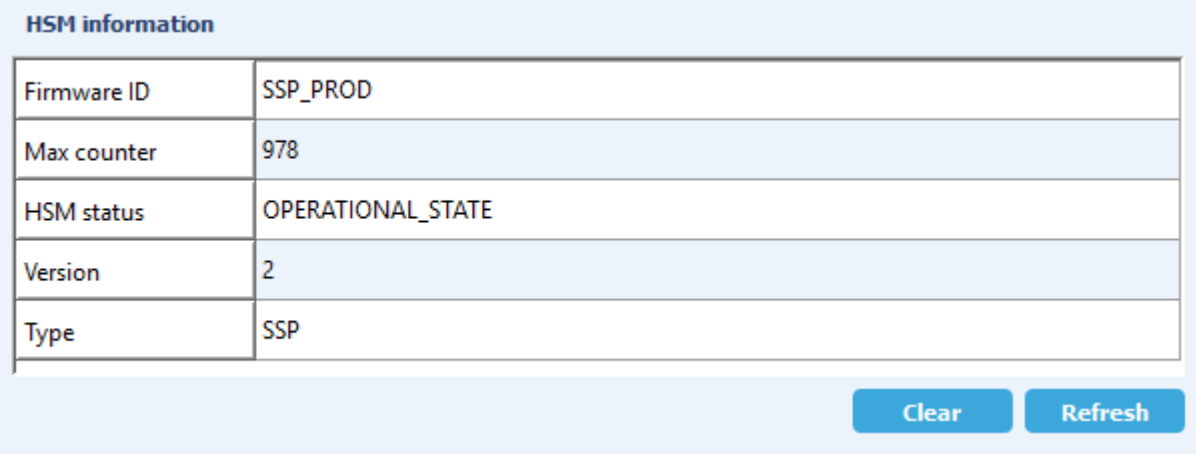
The [STM32MPUSSP-UTIL] package provides all personalization package files, ready to be used on the SSP flow. Each personalization package file name starts with a number, which is the product ID of the device. The user must select the correct one:

- *5010100D* file for STM32MP13xC/F lines rev.Y (1.2)
- *5000200A* file for STM32MP15xC/F lines rev.Z (2.0) and rev.Y (2.1)
- *5050200E* file for STM32MP23xC/F lines and STM32MP25xC/F lines rev.Y(2.1)
- *50301008* file for STM32MP21xC/F lines rev.Z (1.1)

7.3.2 ST HSM GUI reading information

Once this data is programed into the ST HSM, the ST HSM is automatically locked then the user can extract the associated information.

Figure 21. **Reading ST HSM information**



This panel displays the firmware identifier, the maximum counter and the ST HSM status.

- **ST HSM status** can be:
  - OEM_STATE: ST HSM not programmed.
  - OPERATIONAL_STATE: ST HSM programmed and locked. It can no longer be programmed.
- **Version**: indicates the hardware version of the used HSM, it can be
  - 1: ST HSM version 1
  - 2: ST HSM version 2.
- **Type**: indicates which security process is used:
  - SFI: static license for a complete application.
  - SMI: static license for a library
  - SSP: static license for secrets.
  - -: type not available

7.3.3 ST HSM provisioning with command line interface

It is also possible to use the STM32 Trusted Package Creator for ST HSM provisioning with command line. The commands are detailed in the user manual [UM2238].

The STM32TrustedPackageCreator tool provides all personalization package files, ready to be used on the SSP flow. To obtain all the supported packages, go to the *STM32MPUSSP-UTIL* directory residing in the tool installation path. Each file name starts with a number, which is the product ID of the device. The user must select the correct one.

Example:

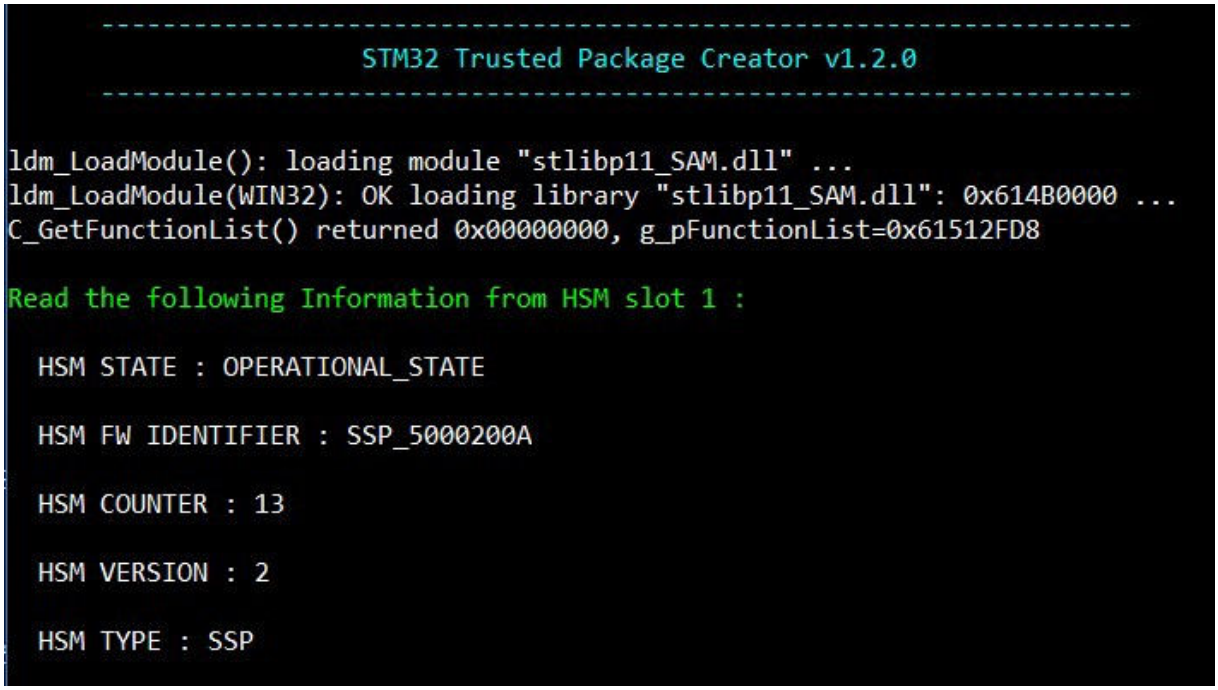
```
STM32TrustedPackageCreator_CLI -hsm -i 1 -k
          "C:\TrustedFiles\key.bin" -n "C:\TrustedFiles\nonce.bin" -id SSP_5000200A -mc 13 -pd
          "C:\SSP\enc_ST_Perso_MPU.bin"
```

7.3.4 ST HSM CLI reading information

Get all the associate ST HSM information using the following command:

```
STM32TrustedPackageCreator_CLI -hsm -i 1 -info
```

Figure 22. **Get ST HSM information in CLI mode**



7.4 SSP procedure with STM32CubeProgrammer (example)

In this part the STM32CubeProgrammer tool is used in the CLI mode (the only mode available so far for a secure programming) to program the SSP image already created with the STM32TrustedPackageCreator tool. The STM32CubeProgrammer supports communication with ST HSMs (hardware secure modules based on smart card) to generate a license for the connected STM32MPx device during the SSP install.

The SSP flow can be performed using either USB or UART interfaces.

The STM32CubeProgrammer exports a simple SSP command with some options to perform the SSP programming flow.

-ssp, --ssp

Description: programs an SSP file

Syntax: -ssp <ssp_file_path> <ssp-fw-path> <hsm=0|1> <license_path|slot=slotID>

<ssp_file_path>: SSP file path to be programmed, bin or ssp extensions

<ssp-fw-path>: SSP signed firmware path

<hsm=0|1>: sets the user option for ST HSM use (do not use ST HSM or use ST HSM), default value: hsm=0

<license_path|slot=slotID>: path to the license file (if hsm=0)

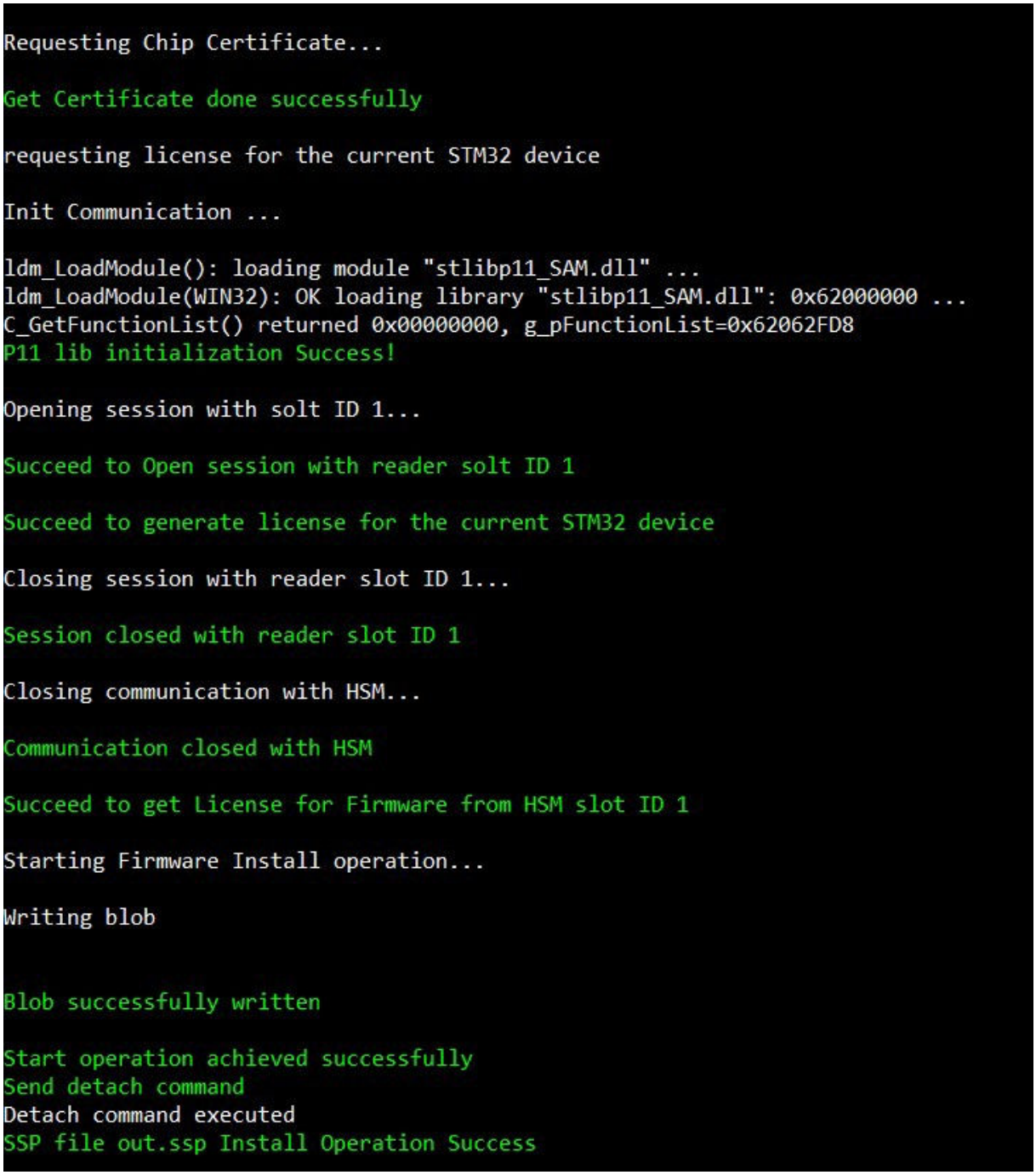
reader slot ID if HSM is used (if hsm=1)

Example using USB DFU bootloader interface:

```
STM32_Programmer_CLI.exe -c
        port=usb1 -ssp "out.ssp"
          "tf-a-ssp-stm32mp157f-dk2-trusted.stm32" hsm=1 slot=1
```

Note: All SSP traces are shown on the output console.

Figure 23. **SSP install success**



If there is any faulty input, the SSP process is aborted, then an error message is displayed to indicate the root cause of the issue.

STM32 CubeProgrammer is delivered as an example. See [STM32Trust] for partner tools.

8 Programming protocol extension

Because the SSP process is mainly based on the USB DFU or UART loading protocol, it requires to extend the already defined protocol for STM32MPx series, see [AN5275].

The protocol add-ons is part of the SSP secure firmware.

8.1 UART/USART command set

8.1.1 STM32MP1

In UART/USART protocol, the 0xF3 partition ID is used for SSP exchange. A 0xF3 read downloads the device certificate and a 0xF3 write uploads the encrypted secrets.

8.1.2 STM32MP2

In UART/USART protocol, the 0xF3, 0xF5 partition ID are used for SSP exchange. A 0xF3 read downloads the device certificate and a 0xF3 write uploads the RSSe SSP firmware. 0xF5 write uploads the encrypted secret file (blob).

8.1.3 Read partition command (0x12)

The read command is not implemented in the default TF-A. It is required in case of SSP so that the STM32CubeProgrammer can download the device certificate.

Table 8. Read partition command

Byte	Description
1	0x12 = read partition
2	0xED = XOR of byte 1
-	Wait for ACK or NACK
3	Partition Id = 0xF3 for SSP
4-7	Offset address
8	Checksum byte: XOR (byte 3 to byte 7)
-	Wait for ACK or NACK
9	Number of bytes to be received -1 (N = [0,255])
10	Checksum byte: XOR (byte 9)
-	Wait for ACK or NACK

8.2 USB

8.2.1 STM32MP1

Because the TF-A and ROM code share the same USB DFU stack, it is not possible to add more partition ID. The **0x0** partition (used for Flash layout) is reused for the SSP management.

- The certificate is retrieved using the DFU Upload command on the phase ID **0x0**.
- The encrypted binary file is sent to the device using the default download command using the partition **0x0**.

8.2.1.1 STM32MP13

TF-A SSP is doing a USB re-enumeration of the USB. It can refresh the partition table to add the specific SSP partition.

The **0xF3** partition is dedicated to the SSP communication to upload the device certificate and download data.

8.2.1.2 STM32MP15

Because TF-A and ROM code share the same USB DFU stack, it is not possible to add more partition ID.

The partition ID **0x0** is reused to download and upload the SSP data.

8.2.2 STM32MP2

The ROM code is the only component that share interface to the download manager.
The certificate is downloaded using the dedicated **0xF3** partition ID.
The encrypted secret file is downloaded using the **0xF5** partition ID.
The RSSE_SSP firmware is downloaded (as last partition) using the **0xF3** partition ID.

8.2.3 DFU download

Table 9. DFU SetOffset command

Byte	Description
1	Phase ID for binary download (0x0 on STM32MP15, 0xF3 on STM32MP13, 0xF3 and 0xF5 on STM32MP2)
2-5	Offset int the partition

Start command is still required to finalize the transfer and update to the next phase.

Table 10. DFU Start command

Byte	Description
1	0xFF
2-5	Address

8.2.4 DFU upload

DFU upload has been updated to allow retrieving the device certificate.

Table 11. DFU Upload command

Byte	Description
1	Phase ID for binary download (0x0 on STM32MP15, 0xF3 on STM32MP13, 0xF3 and 0xF5 on STM32MP2)
2-5	Download address (not used)
6-9	Offset (not used)

Example for SSP execution on STM32MP1x:

1. Upload certificate
   Byte 1 **0xF3** (or Byte 1 **0x00** on STM32MP15)
   Byte 2-5 <Address>
   Byte 6-9 0x00000000
2. License and encrypted secrets file download
   Byte 1 **0xF3** (or Byte 1 **0x00** on STM32MP15)
   Byte 2-5 <Address>
3. Send start command to end process
   Byte 1 0xFF
   Byte 2-5 0xFFFFFFFFF
4. Request DFU_DETACH after manifestation
   Byte 1 0x00
   Byte 2-5 0xFFFFFFFF
   Byte 6 0x01

Example for execution on STM32MP2:

1. Upload certificate
   Byte 1 **0xF3**
   Byte 2-5 <Address>
   Byte 6-9 0x00000000
2. Encrypted secrets file download
   Byte 1 **0xF5**
   Byte 2-5 <Address>
3. RSSE SSP firmware download
   Byte 1 **0xF3**
   Byte 2-5 <Address>
4. Send start command to end process
   Byte 1 0xFF
   Byte 2-5 0xFFFFFFFFF
5. Request DFU_DETACH after manifestation
   Byte 1 0x00
   Byte 2-5 0xFFFFFFFF
   Byte 6 0x01

Revision history

Table 12. **Document revision history**

Date	Version	Changes
03-Sep-2020	1	Initial release.
03-Mar-2023	2	Introduced STM32MP131, STM32MP133 and STM32MP135 lines of products. All sections have been updated. Addition of: Section 8.2.1.1: STM32MP13. Section 8.2.1.2: STM32MP15. Section 8.2.3: DFU download. Section 8.2.4: DFU upload.
13-Jun-2025	3	Introduced STM32MP2 series. All sections have been updated. Addition of: Section 3.2: Device life cycle Section 3.4.1: OEM secrets preparation Section 3.4.1.1: **OTP** Section 3.4.1.2: Backup secrets Section 3.4.1.2: Backup secrets Section 3.4.4.2: STM32MP2 Section 3.5.2.2: STM32MP2 Section 3.5.3.2: STM32MP2 Section 5.3: Backup secret file Section 5.5: STM32MP2 encrypted secret file Section 6.2: STM32MP2 Section 7.1: Secret files generation Section 7.1.1: Secrets Gen panel Section 7.1.2: Backup Gen panel Section 7.2.2: STM32MP2 encrypted secrets file generation with graphical user interface Section 8.1.2: STM32MP2 Section 8.2.2: STM32MP2

Contents

List of tables

List of figures