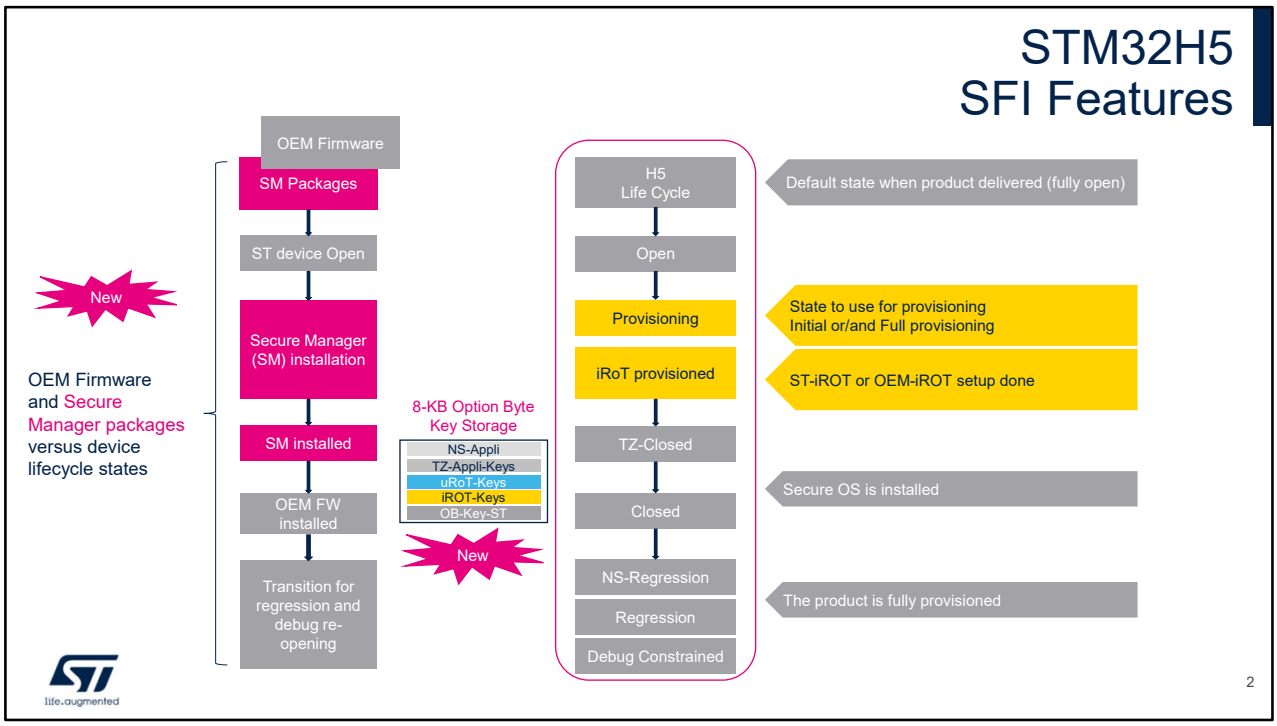
life.augmented

STM32H5 ROOT SECURITY SERVICES

Secure Firmware Install (SFI)

Hello, and welcome to this presentation describing the Secure Firmware Install (SFI) feature offered by the Root Security Services.



The RSS/SFI solution supported by STM32H5 allows secure firmware install and key provisioning during the provisioning state of the product life cycle.
SFI can be launched in the Provisioning state of the product life cycle, not in the next states.
Two SFI solutions are offered:

- SFI without Secure Manager: legacy SFI with Hardware Secure Module (HSM) in untrusted Manufacturing
- SFI with Secure Manager (with or without HSM).

The STM32Trust Secure Manager relieves the developers of writing and validating their own code while providing security services developed according to the best practices.
The STM32H5 provides a secure data storage in Flash Memory: the OBKey area

- Capacity of 2x 8 KB, that supports swapping
- Split into 5 domains, for which access control is

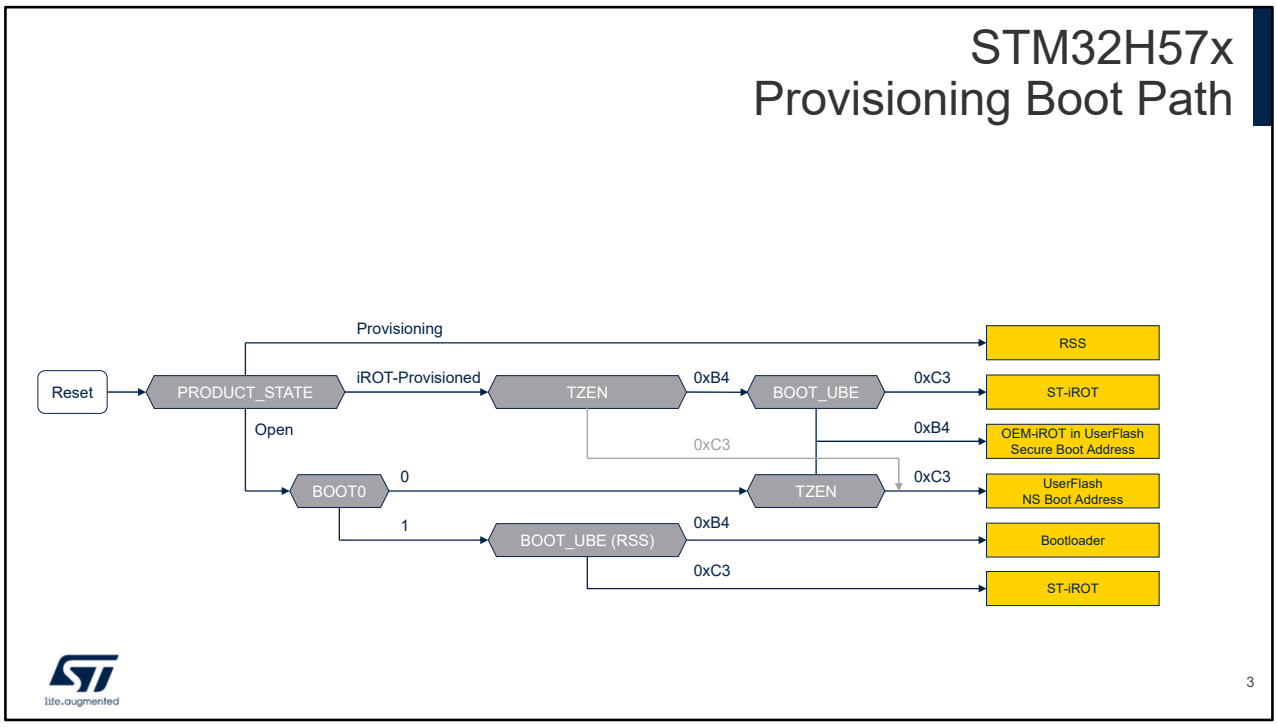
determined by current Hide Protect Level (HDPL) and EPOCH value, sent to the SAES module: HDPLs 0, 1, 2, 3 Secure, and 3 Non-Secure.

- The OBK keys can be stored encrypted based on SAES.

The following protections are implemented:

- Per domain encryption thanks to cryptography isolation
- Anti-rollback thanks to EPOCH counter of regressions
- Anti-cloning thanks to RHUK (unique per device).

SFI without HSM and OBK storage are new features in the STM32H57x Series. The flow in the left side shows the Secure Manager packages and the OEM Firmware installed with respect to the states of the STM32H5 life cycle.

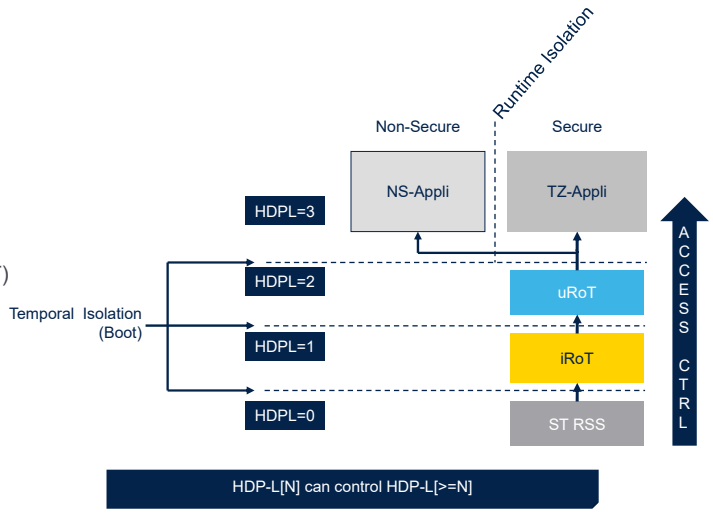


This figure details the selection of the first program executed when reset is negated.
First the current product state is read from option bytes.
If it is Provisioning, the Root Secure Services (RSS) is entered. It is programmed by ST in system memory during production and offers runtime services to user firmware.
If it is iROT-Provisioned, next step is reading the TZEN option byte.
If it is Open, the next step is getting the BOOT0 pin state.
Using boot pin set to high level and BOOT_UBE allows to launch either user flash code through ST-iROT or ST bootloader.
Thanks to unique boot entry (BOOT_UBE) it is possible to select the boot entry point between security services in system flash (ST-iROT) and proprietary boot entry (OEM-iROT) at the address defined by SECBOOTADD option

bytes.
This selection is possible only for products embedding cryptographic acceleration (STM32H573), when TrustZone is enabled (TZEN=0xB4).
When TrustZone is disabled (TZEN = 0xC3), the application selects a boot entry point located in the non-secure user flash memory, at the address defined by NSBOOTADD option bytes.

RSS & Temporal isolation

- Temporal Isolation Levels:
  - To manage access control on Code & Data
  - Code protected with Hide protection (HDP)
  - Data:
    - Flash OB-Keys (Physical Access Control)
      - 5 secure storage areas
        - HDPL0 ➔ST (never erased)
        - HDPL1 ➔iROT (ST-iROT or OEM-iROT)
        - HDPL2 ➔uROT
        - HDPL3 + Secure ➔Trust Zone
        - HDPL3 + NS ➔Non-Secure Appli
    - Data can be wrapped with DHUK
      - Based on HUK + Version counter
      - Different for each HDPLx



life.augmented

4

This figure explains the hierarchy of access.
The various programs involved in boot-time and run-time are presented on the right:

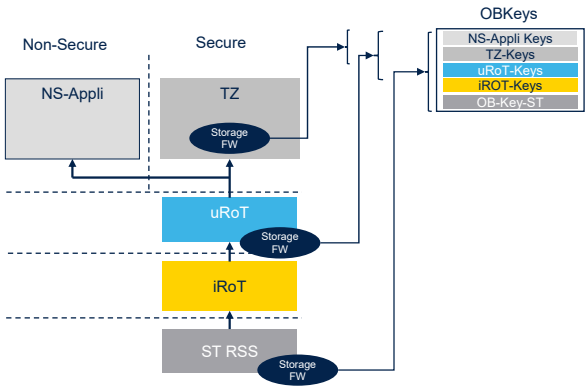
- The ST Root Secure Service (RSS)
- The ST Immutable Root Of Trust (iROT)
- The ST Updatable Root Of Trust (ST-uROT) or proprietary one (uROT)
- The Secure and non-secure applications.

Each of these programs is assigned an HDPL.
Access to the hide protection area can be denied by progressing the HDPL level in the System configuration, boot and security (SBS).
For example, the part of the flash containing the RSS becomes hidden when the RSS transitions the HDPL from 0 to 1.
The HDPL level can be only cleared by a system reset, there

is no means to deactivate the area protected by HDP.
The protected HDP area is defined by setting its size using start and end sectors in a similar way as the secure watermark.
The current HDPL also determines the access control rules and keys used for encryption / decryption.
Non-volatile data and code of a particular HDPL can be encrypted with the Derived Hardware Unique Key (DHUK), that includes the monotonic counter, making sensitive resources undecipherable from higher HDPLs.

STM32H5
Hardware Secure Data Storage

- Hardware Secure Data Storage / Use cases
  - For application (runtime isolation)
    - Data Storage firmware
      - Isolated in Secure Privilege
      - Provide services to Secure Non-Privilege
      - Provide services to non-secure
  - Key Provisioning
    - During OEM product manufacturing (Provisioning state)
      - SFI when manufacturing not trusted
      - Using RSS-Lib when manufacturing is trusted
    - Key Provisioning using application services
      - Based on SKP Secure Key Provisioning services embedded in FW.
        - Embedded in uROT, SecureOS



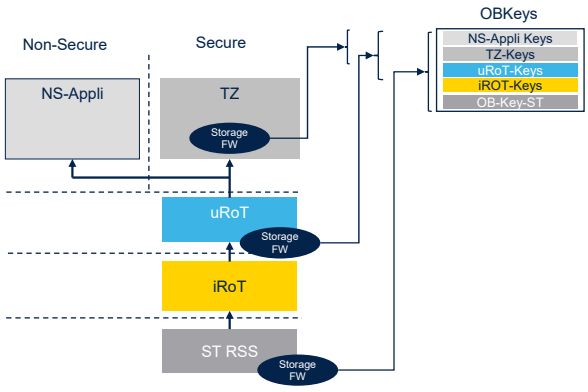
life.augmented

5

Hardware secure storage control improves the security, by isolating programs running at different privilege and security levels.
It also protects OBKeys against accesses and utilization from lower level HDPLs.
Finally, hardware secure storage enables secure provisioning of firmware, by different OEMs.
Note that even if we are talking about Key storage, any kind of data can be stored.

STM32H5
Hardware Secure Data Storage

- HW Secure Data Storage / Features
  - 3 levels of protection
    - TrustZone, Access Ctrl, Crypto isolation
  - In Flash Memory: OBKey area
    - Capacity 8 KB (swapped)
    - Splitted in 5 domains:
      - HDPL0, 1,2,3S, 3NS
      - HW Access Control
  - Stored Encrypted
    - The OBK keys can be stored encrypted based on SAES
      - Per domain encryption ➔Cryptography isolation
      - Anti-rollback ➔thanks to EPOCH counter of regressions
      - Anti-cloning ➔Thanks to RHUK (unique per device)



life.augmented

6

At the receipt of an OBK read, write or execute access, secure and privilege attributes are first checked, followed by OBK-HDPL.
The OBK-HDPL value must exactly match the HDPL assigned to the key location, otherwise an error is raised.
OBK storage is split into 5 domains, one per HDPL from 0 to 2 and two for HDPL3, one for secure keys, one for non secure keys.
The OBK keys can be stored encrypted based on SAES, with per-domain encryption.
The encryption mechanism also ensures anti-rollback and anti-cloning.

Secure firmware install (SFI)

- Secure firmware install (SFI):
  - Allows secure and counted installation of OEM firmware in untrusted production environments (such as OEM contract manufacturer)
  - SFI is implemented using the secure RSS and the non-secure immutable bootloader
  - OEM firmware protected by SFI can be stored in embedded flash or encrypted in external flash
  - The number of STM32 devices on which the firmware has been installed can be counted by the HSM
- When external flash memory is targeted by SFI:
  - The OTFDEC peripheral can be used to accelerate encryption
  - SFI can re-encrypt OEM external firmware using the AES key(s) dedicated to the OTFDEC peripheral
  - Keys can be globally managed (by the tools), or they can be device-specific (e.g., locally computed using the true RNG peripheral)

life.augmented

7

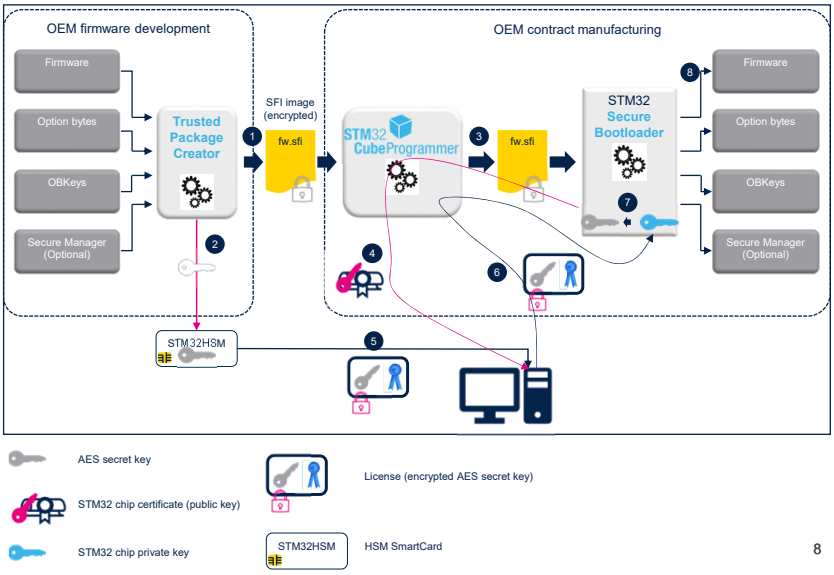
Secure firmware install (SFI) is a global solution for STM32H5 Series of microcontrollers, allowing secure and counted installation of OEM firmware in untrusted production environments, such as OEM contract manufacturer.
SFI is implemented using the secure RSS and the non-secure immutable bootloader.
OEM firmware protected by SFI can be stored in the device's embedded flash or encrypted in external flash connected via OCTOSPI and OTFDEC units.
Authenticity, integrity and confidentiality of the OEM internal firmware are checked before embedded flash is programmed with decrypted firmware.
The STM32H5 SFI solution consists in having the entire OEM firmware, OBKeys and option bytes encrypted with an AES secret key, thanks to STM32 Trusted Package Creator tool.

This is done during the development of the OEM firmware. Confidentiality of this AES secret key is ensured using a unique key pair dedicated to the STM32 device, with the private key readable only by RSS.
OTFDEC uses AES-128 in counter mode to ensure On-the-fly 128-bit decryption during the OCTOSPI memory-mapped read operations.
SFI can be used to encrypt OEM external firmware using the AES key(s) dedicated to the OTFDEC peripheral.
This key can be:

- Common to all devices (in this case tools could perform the encryption), or
- Unique per device (in this case firmware is encrypted inside the device).

SFI to internal flash

1. SFI image (encrypted) available from *STM32 Trusted Package Creator*
2. The OEM programs the HSM with the AES secret key
3. Start of the SFI process
4. Device certificate retrieval through the STM32 CubeProgrammer
5. The HSM generates the license based on the certificate.
6. The HSM provides the license to the STM32 through the STM32 CubeProgrammer
7. The RSS retrieves the OEM AES secret key encrypted in the license
8. The encrypted firmware and option bytes are decrypted then programmed



life.augmented

8

The installation of secure firmware in internal Flash memory is achieved through the following steps:

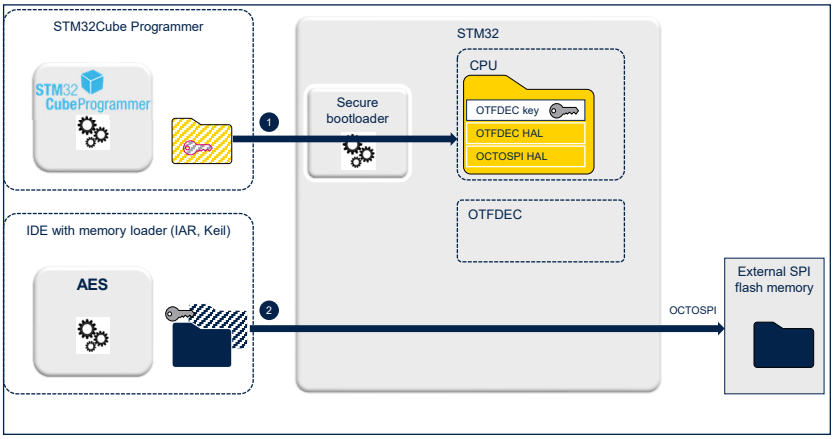
1. The (encrypted) SFI image is available from STM32 Trusted Package Creator
2. The OEM programs the HSM with the AES secret key
3. Start of the SFI process
4. Device certificate retrieval
5. STM32 device authentication in the HSM
6. The HSM provides the license to the STM32 device
7. The RSS retrieves the OEM AES secret key encrypted in the license
8. Encrypted firmware and option bytes are transferred, decrypted then programmed.

Note that three keys are used in this sequence:

- The AES symmetric key
- The STM32chip private and public asymmetric keys.

SFI to internal & external flash (1)

1. Secure programming of internal Flash memory, using SFI
2. Encryption then programming of external firmware and data
   - ✓ See next slide for more details



life.augmented

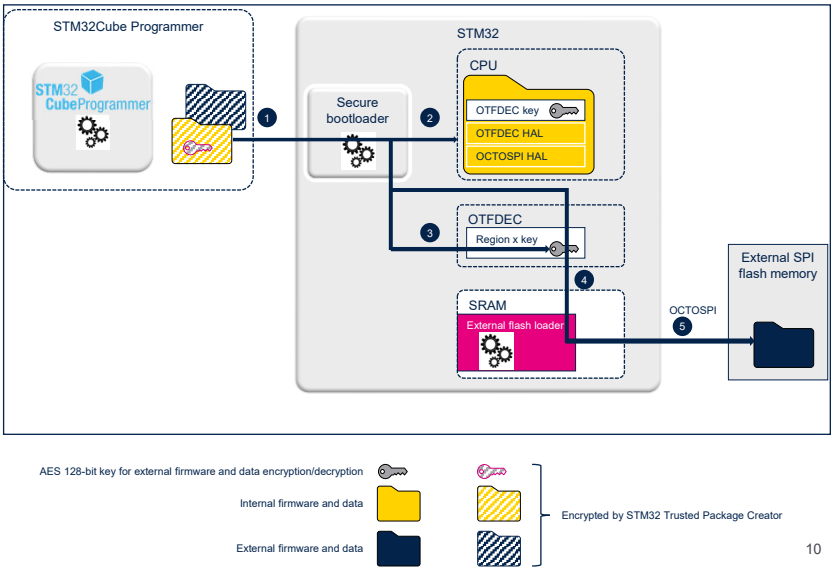
9

The cryptographic engine responsible for the on-the-fly external Flash memory decryption (OTFDEC) supports the AES standard cryptographic algorithm.
Thanks to this standard algorithm, the OEM can encrypt the external firmware and data on the host before programming it into the external Flash memory, without using the STM32 secure bootloader.
This slide shows that the secure programming of internal Flash memory (1) and the encryption plus programming of the external firmware and data (2) can be done in two separate flows.
The first flow uses the secure bootloader, while the second one uses the OEM host to program the external Flash memory.
Then, during each secure boot, the secure internal firmware first copies the AES firmware and data key(s) into write-only

OTFDEC key registers, then activates the OTFDEC region tied to those keys.
At this point, the CPU can seamlessly read data and fetch code from external Flash memory once the OCTOSPI driver has been initialized.

SFI to internal & external flash (2)

1. Create an SFI image, with:
   - Internal firmware and data (including external Flash memory drivers)
   - External firmware and the data *AES key*
   - External firmware and data
2. Internal Flash memory programming
3. External firmware and data *AES key* programming in the OTFDEC peripheral
   - Alternatively, such key(s) can be managed locally in the device, not globally in the flashing tools
4. External Flash memory chunk encryption
   - Not required if the image is already encrypted with the AES key
5. External Flash memory programming by the user's firmware



This slide represents the sequence where the STM32 secure bootloader handles both internal firmware installation and external firmware installation with a global external Flash memory AES key and the help of an external Flash memory loader. The numerical steps are shown in the diagram.

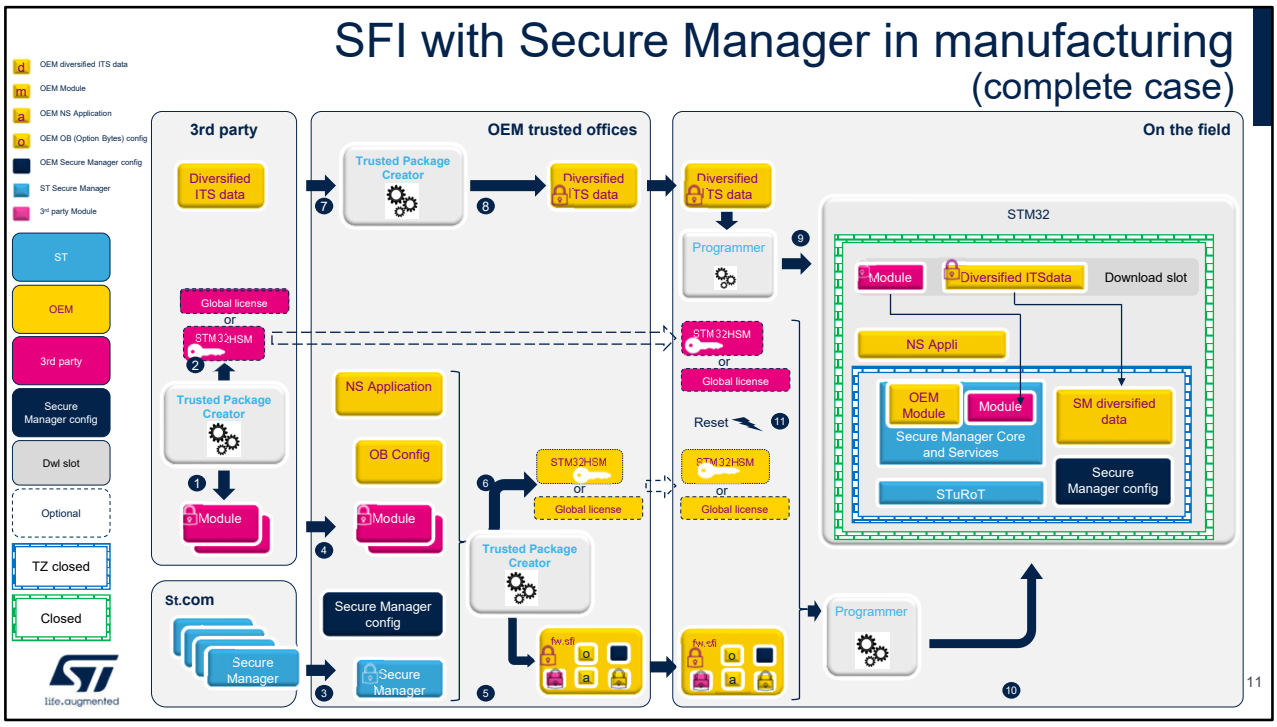
1. Create an SFI image, with a) internal firmware and data (including external Flash memory drivers), b) external firmware and the data AES key, and c) external firmware and data
2. Internal Flash memory programming, as described on the previous slide.
3. External firmware and data AES key programming in the OTFDEC peripheral. Alternately to what is drawn on the slide, this key can be managed locally in the device, not globally in the flashing tools.
4. External Flash memory chunk encryption, required if the

image was not encrypted by the STM32 Trusted Package Creator

5. External Flash memory programming by the user's firmware.

Then, during each secure boot, the secure internal firmware first copies the AES firmware and data key(s) into write-only OTFDEC key registers, then activates the OTFDEC region tied to those keys.

At this point the CPU can seamlessly read data and fetch code from external Flash memory once the OCTOSPI driver has been initialized.



This slide presents for Secure Firmware Install procedure in the complete case where OEM wants to install:

- Secure Manager
- Secure Manager configuration
- Non-secure application
- Option Bytes configuration
- Secure Modules
- Data diversified per device.

In the complete use case, two SmartCards are required:

- One associated with SFI file (fw.sfi), containing OEM key to protect this SFI file
- The other one associated with 3rd party modules, containing 3rd party key.

The SFI file (fw.sfi) includes Secure Modules and the Secure Manager, in addition to non-secure application, Secure Manager configuration and Option Bytes configuration.

Note that there is 2 other SFI cases with STM32H5 Series :

- SFI without Secure Manager+ HSM required (legacy configuration)
- SFI with Secure Manager without HSM.

The steps to install the Secure Manager into the STM32H5's flash memory is as follows:

1. The third party/OEM uses Trusted Package Creator (TPC) to encrypt the modules
2. The third party/OEM can use the TPC to optionally set a license for each module and program the HSM (which is a smartcard, with its own OEM AES secret key, its own nonce, and a maximum installation counter) or generate a global license file
3. The Secure Manager is downloaded from st.com website
4. The Secure Manager offers the possibility to install the third-party secure modules. In this case, the Secure Manager requires these modules to have specific format developed via the SMDK (see the presentation on the Secure Manager for more details)
5. The Trusted Package Creator generates the fw.sfi file and generates a global license file from Option Byte configuration, Secure Manager configuration, Secure Manager, third party modules and NS Application.
6. The OEM uses Trusted Package Creator to program the HSM
7. The third-party/OEM can change the factory ITS (Internal Trusted Storage) blob default content to include its own keys and data
8. Diversified ITS data are encrypted using TPC
9. The STM32Cubeprogrammer can be used to install the Diversified ITS data. ITS data are programmed into a download slot to be updated by the Secure Manager.
10. The STM32CubeProgrammer is used to program the Secure Manager, the Secure Manager Configuration and Option Bytes into a part of the flash memory, that will be

closed. It also installs the NS application in the non-secure part of the memory.

11. Reset the chip to finish the procedure.

Rishi

References

- For more details and additional information, refer to the following
  - RM0481: STM32H563/H573 and STM32H562 Reference Manual
  - AN2606: STM32 microcontroller system memory boot mode
  - AN4992: Overview of secure firmware install (SFI)
  - UM2237: STM32CubeProgrammer software description
  - UM2238: STM32 Trusted Package Creator software description
  - Secure Manager Getting Started
  - Secure Manager User Manual
  - SFI WIKI Pages for H5: https://wiki.st.com/stm32mcu/wiki/Security:SFI_for_STM32H5

life.augmented

12

For more details, please refer to:

- Application note AN2606 about the STM32 microcontroller system memory boot mode
- Application note AN4992 about the Overview of secure firmware install (SFI)
- User manuals for the STM32CubeProgrammer and STM32 Trusted Package Creator are also available on the ST website.

Thank you

life.augmented

Thank you for attending this presentation!
You can also refer to the following presentations on STM32H5 security features:

- Security overview
- Enhanced key storage
- Life cycle
- Debug authentication
- Secure Manager.