UM3370

User manual

STM32MP25xx security guidance for SESIP 3 Certification

Introduction

This document describes the preparative procedures and operational user guidance of the STM32MP25x microprocessor to make a secure system solution according to the SESIP level 3 certification scheme.

The security guidance described in this document applies to any boards based on the devices listed in the table below.

Table 1. **Applicable products**

Reference	Products
STM32MP25x	STM32MP251C, STM32MP251F, STM32MP253F, STM32MP255C, STM32MP255F, STM32MP257C, STM32MP257F

1 General information

This document applies to STM32MP25xx Arm®-based MPUs.

Note: Arm and TrustZone are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Table 2. **Specific acronyms**

Acronym	Description
HUK	Hardware unique key
HW	Hardware
IoT	Internet of Things
MPU	Microprocessing unit
RIF	Resource isolation framework
RMA	Return material for analysis
SCA	Side-channel attack
SESIP	Security evaluation standard for IoT platforms
SFR	Security-functional requirement
TD	Trusted domain
TOE	Target of evaluation

2 Reference documents

Table 3. List of reference documents

Reference	Document title and revision
[RM]	Reference manual *STM32MP25xx advanced Arm®-based 32/64-bit MPUs* (RM0457), release 5
[ST]	Technical note *STM32MP25xx SESIP Level 3 Security Target* (TN1533), revision 1
[DS]	Datasheet *STM32MP25xC/F datasheet* (DS14284), revision 1
[UM2237]	User manual *STM32CubeProgrammer software description* (UM2237), version 25
[UM2238]	User manual *STM32 Trusted Package Creator tool software description* (UM2238), revision 15
[UM2542]	User manual *STM32 key generator software description* (UM2542), revision 3
[UM2543]	User manual *STM32MP25x series signing tool software description* (UM2543), revision 4
[Readme_PRGFW]	STM32PRGFW-UTIL, revision 1.0.4 *https://github.com/STMicroelectronics/STM32PRGFW-UTIL/blob/main/README.md*
[IEEE1149]	IEEE 1149.1–2013
[WIK1]	Wiki page *Populate the target and boot the image*, revision 5.1.0
[WIK2]	Wiki page *Getting started*, revision 5.1.0

3 Preparative procedures

This section provides useful information for ensuring that the target of evaluation (TOE) has been received and installed in a secure manner as intended by the developer.

- Secure acceptance: procedures to check the device to be tested.
- Secure preparation of the operational environment: procedures to set up the environment needed to manage and prepare the final product.
- Secure installation: procedure to program and configure the product to be prepared.

3.1 Secure acceptance

Secure acceptance is the process in which the user securely receives the TOE and verifies its genuineness.

The TOE is distributed as an MPU device, with corresponding firmware packages that can be obtained from *www.st.com*. Refer to the cover page for the applicable devices.

How to accept an STM32MP25x MPU device

When the device is in the *Secured_Unlocked* default state, TOE genuineness can be verified using a debugger, reading data `0x2000 6505` at address `0x5423 6400`. More specifically:

- Bits [11:0] returns the die ID (`0x505` for STM32MP25xx MPU)
- Bits [31:16] returns the major revision ID (`0x2000` is silicon revision 2)

Additional values for minor revision must be read from OTP 102 (ID) with the value `0x11`:

- Bits [2:0] returns the minor revision ID (1 for the minor version)
- Bits [5:3] returns the major revision ID (2 for the major version).

Alternatively, the following methods can be used with any board mounted with the TOE:

- Set the boot switches to the off position (BOOT pins at all zeroes or all ones).
- Connect the USB Type-C® host port of the board to the user's computer.
- Power up the board and press the reset button.
- Use the STM32CubeProgrammer command-line interface (CLI):

Enter:

```
STM32_Programmer_CLI -c port=usb1 -p
```

Expected below answer from the device, prepared by ROM code:

```
-------------------------------------------------------------------
                 STM32CubeProgrammer v2.17.0
-------------------------------------------------------------------

USB speed      : High Speed (480MBit/s)
Manuf. ID      : STMicroelectronics
Product ID     : DFU in HS Mode @Device ID /0x505, @Revision ID /0x2000
SN             : 0022003B4136500400373653
DFU protocol   : 1.1
Board          : --
Device ID      : 0x0505
Device name    : STM32MP25xx
Device type    : MPU
Revision ID    : --
Device CPU     : Cortex-A35
PhaseID        : 0x01
```

We find in the product ID that it includes the die ID (`0x505` for the STM32MP25xx MPU) and the major revision ID (`0x2000` is the silicon major revision 2).

The STM32PRGFW-UTIL (see [Readme_PRGFW]) can be used to retrieve the OTP 102 value using the following commands:

-
  ```
  STM32_Programmer_CLI.exe -c port=usb1 -w <Your Directory Path>\Projects\<STM32 device>\
  Binary\FlashLayout_STM32PRGFW_UTIL.tsv
  ```

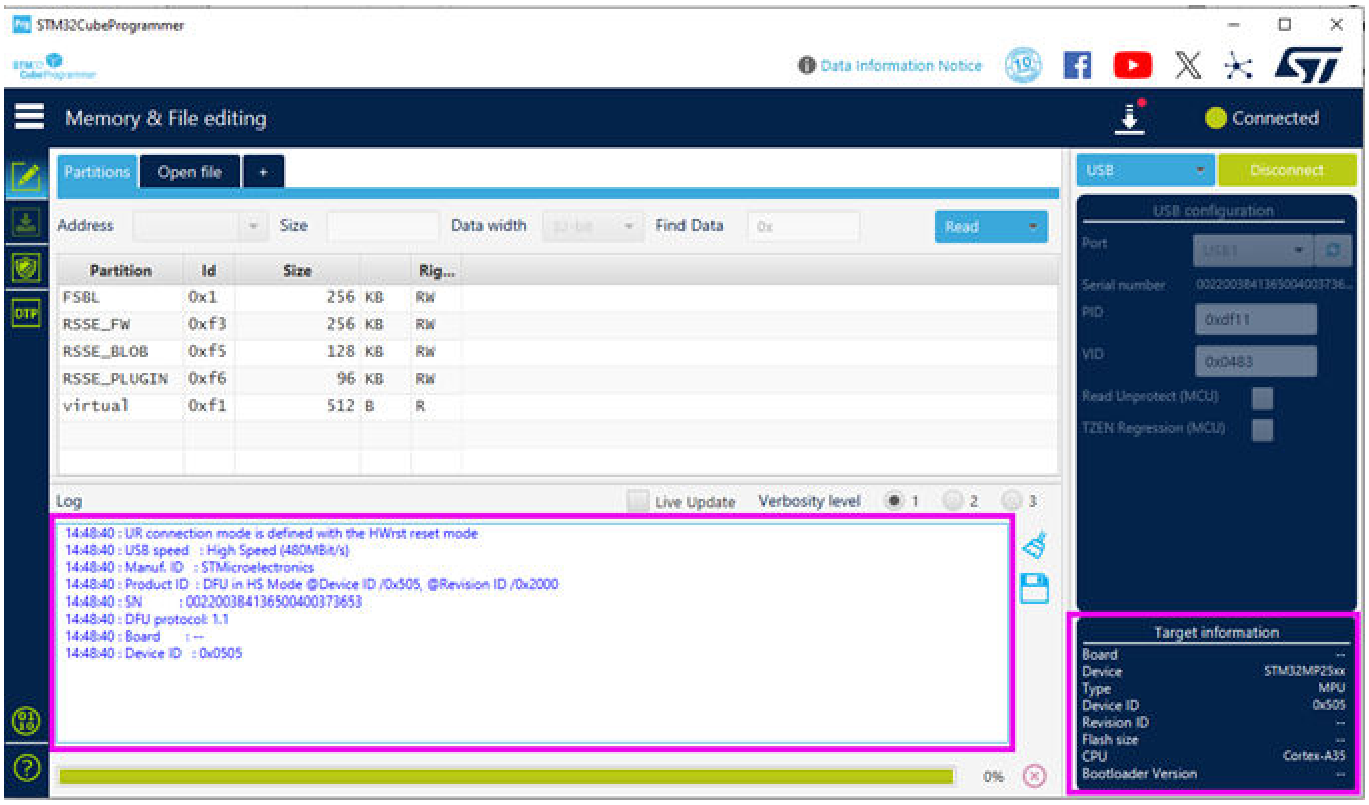
-
  ```
  STM32_Programmer_CLI.exe -c port=usb1 -otp displ word=102
  ```

```
-------------------------------------------------------------------
                 STM32CubeProgrammer v2.17.0
-------------------------------------------------------------------
USB speed      : High Speed (480MBit/s)
Manuf. ID      : STMicroelectronics
Product ID     : USB download gadget@Device ID /0x505, @Revision ID /0x, @Name /STM32MP257FAL
Rev.Y,
SN             : 0022003B4136500400373653
DFU protocol   : 1.1
Board          : --
Device ID      : 0x0505
Device name    : STM32MP257FAL Rev.Y
Device type    : MPU
Revision ID    : --
Device CPU     : Cortex-A35

UPLOADING OTP STRUCTURE ...
  Partition : 0xF2
  Size : 4096 Bytes

Uploading OTP data:
██████████████████████████████████████████████████

OTP Partition read successfully

OTP DATA WORDS :

Available to customer :
                      | Data102 : 0x00000011 ------------> STM32MP25 Rev2.1 (Rev Y)
                      | Status102 : 0x00000000
```

Alternatively, use the STM32Cube graphical user interface (GUI) as follows:

- Set the boot switches to the off position (BOOT pins[3:0]=0000 or 1111).
- Connect the USB Type-C® host port of the board to the user's computer.
- Power up the board and press the reset button.
- Open STM32CubeProgrammer GUI to choose the **USB link** (not **STLink**, set by default) in the connection picklist and click on the refresh button. The serial number is displayed if the USB link (DFU) is detected.
- Click on the **Connect** green button.
- The log panel displays the product ID as shown in Figure 1.

Figure 1. STM32MP25x acceptance using STM32CubeProgrammer GUI



Partition	Id	Size	Rig...
FSBL	0x1	256 KB	RW
RSSE_FW	0xf3	256 KB	RW
RSSE_BLOB	0xf5	128 KB	RW
RSSE_PLUGIN	0xf6	96 KB	RW
virtual	0xf1	512 B	R

```
14:48:40 : UR connection mode is defined with the HWrst reset mode
14:48:40 : USB speed   : High Speed (480MBit/s)
14:48:40 : Manuf. ID   : STMicroelectronics
14:48:40 : Product ID  : DFU in HS Mode @Device ID /0x505, @Revision ID /0x2000
14:48:40 : SN          : 002200384136500400373653
14:48:40 : DFU protocol: 1.1
14:48:40 : Board       : --
14:48:40 : Device ID   : 0x0505
```

Target information

Board	--
Device	STM32MP25xx
Type	MPU
Device ID	0x505
Revision ID	--
Flash size	--
CPU	Cortex-A35
Bootloader Version	--

How to select software packages?

Even though software packages are not part of the TOE, they can be useful to the procedure defined in Secure installation and secure preparation (AGD_PRE.1.2C). Refer to that section for details.

3.2 Secure installation and secure preparation (AGD_PRE.1.2C)

This preparative procedure section describes all the steps necessary for the secure installation of the TOE and for the secure preparation of the operational environment by the security objectives for the operational environment as described in [ST].

3.2.1 Setup procedures

Hardware setup procedure

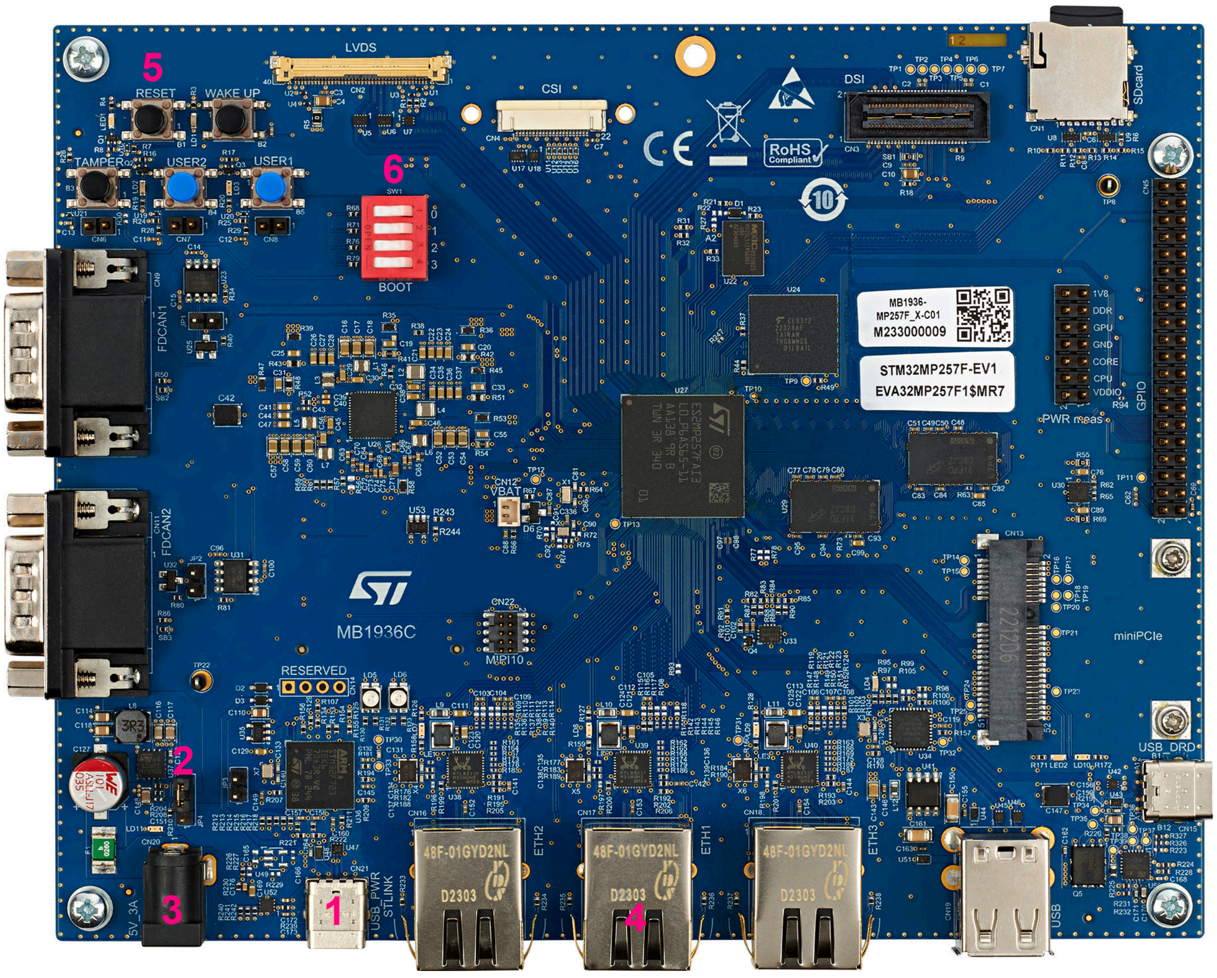
To set up the hardware environment, the development board must be connected to a personal computer via a USB cable. This connection with the PC allows the user to:

- Configure the platform nonvolatile memory (OTP + flash)
- Debug when the protections are disabled

The ST-LINK firmware programmed on the development board must be the V2J42M27 version or newer.

If the STM32MP257F-EV1 evaluation board shown in Figure 2 is used, the hardware setup procedure is described hereafter.

Figure 2. STM32MP257F-EV1 evaluation board



- Connect the micro USB Type-C® **(1)** between the laptop and the STLINK-V3 port of the board.
- Depending on the jumper JP4 **(2)**, connect the additional power supply on the CN20 jack **(3)**:
  1. JP4[1-2]: Select the USB Type-C® power CN21 as the main board supply
  2. JP4[2-3]: Select the 5 V/3 A power supply unit (not provided) on jack CN20 as main board supply
- Optionally connect the Ethernet cable between your Ethernet network and the Ethernet port **(4)** of the board.
- Use the reset button **(5)** when needed.
- The SW1 switch configuration **(6)** is used to select boot mode (some OTP could be used to force boot mode in case BOOT pins are not used on the board). Refer to Table 12 for details.

Software setup procedure

STMicroelectronics provides OpenSTLinux binary packages (*starter* packages) that can run directly on ST boards mounted with the TOE. Each starter package contains an exhaustive set of configured images to boot a nonsecure platform. OpenSTLinux is based on the trusted firmware-A (TF-A) reference implementation, which can be found at *https://www.trustedfirmware.org/projects/tf-a*.

All the steps to install and use starter packages are described on the Getting started wiki page. Those packages, with the associated tools (STM32 key generator, fiptool, and STM32 signing tool) can be used on both Linux® and Windows®.

More specifically, the following tools are required for the preparation and installation of the TOE, as described in Secure installation and secure preparation (AGD_PRE.1.2C):

- STM32CubeProgrammer version 2.17.0, described in [UM2237]
  - Includes STM32 signing tool and STM32 key generator
- Arm® TF-A fiptool version 2.8

For more details on the signing tool and key generator, refer to [UM2242] and [UM2243] respectively.

3.2.2 Secure installation

The STM32MP25x product preparation is done in six steps, to get a complete installation with a fully activated security configuration. All the steps must be performed successfully to have the product in its certified configuration.

Step 1: Generating secrets

In a trusted location, the user generates multiple ECC key pairs (up to 8) using the *STM32MP Key generator* tool (refer to [UM2542] for details). Only one private key is used to sign the boot images (*active key*). The revocation mechanism introduced required generating all the keys at the first step.

With the same tool, the user generates the hash of the table composed of the hash of those eight public keys. This hash must then be stored in the 144 to 151 OTP words of the TOE. Refer to Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C) for details on the TOE boot image authentication scheme.

Depending on the boot mode, A35 or M33 as the main processor, it is possible to use a different Root of Trust, one for A35, and another for M33. In that case, a second set of key pairs must be generated. The OEM_KEY1 (OTP words 144 to 151) is used for M33 firmware authentication. The OEM_KEY2 (OTP words 152 to 159) is used for A35 firmware authentication. The OTP 17 (Bit 8: oem_keys2_enable) must be set to use the OEM_KEY2.

Note: *Activating a root key in the TOE revokes all the previous ones. This means that if key#3 is activated and used for boot, then keys#1 and #2 are permanently automatically. Revocation is managed independently for both OEM_KEY1 and OEM_KEY2.*

If the user wants to manage firmware encryption, they also need to create a 128-bit secret to store in OTP the words 360 to 363 of the TOE. They also create a 32-bit derivation constant to be stored in the encrypted FSBL extension header. With this information, the STM32 key generator tool can compute a 128-bit encryption key that is used to encrypt the FSBL image using AES CBC chaining mode. The plain hash stored in the encrypted FSBL extension header is used as an IV input.

Following the authentication scheme, it is also possible to generate a dedicated 128-bit secret for a second firmware. This secret is stored in the OTP words 364 to 367. It is used when the user enables the OTP 17 (Bit 8: oem_keys2_enable). Refer to Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C) for details on image encryption.

Step 2: Provisioning of secrets

If the provisioning environment is nontrusted, STM32 tools can be used, as detailed in [UM2238]. This nontrusted provisioning environment scenario is out of the scope of the SESIP evaluation.

In a trusted environment, the user can use:

- The STM32PRGFW-UTIL that is connected to the STM32CubeProgrammer to update the OTP (see [Readme_PRGFW] for details), as described below:

Connect the board with a programming tool.

Load the FlashLayout_STM32PRGFW_UTIL.tsv with STM32CubeProgrammer.

Once loaded, open the OTP panel, choose the PKH file (publicKeysHashHashes.bin), and choose set the 144 as *Start wordID*. Choose the program button to start provisioning.

Repeat the same for the secret key with the Start word ID as 360.

In case of using a second key management, repeat the same for both publicKeysHashHashes.bin and secret key but do not forget to also provision the OTP 17 bit 8.

- The stm32key tool is included in the U-Boot firmware that is included in the OpenSTLinux image provided with the TOE. For example, the provision of the public key hash table in OTP words 144 to 151 is done as follows:

Copy the PKH file(s) (publicKeyhash.bin) in a file system partition like bootfs on a storage device like an SD card.

Load in BSEC hash file, for example, from mmc0 partition 8 (ext4) in DDR-SDRAM

```
STM32MP> mmc dev 0
STM32MP> ext4load mmc 0:8 0xc0000000 publicKeysHashHashes.bin
32 bytes read in 50 ms (0 Bytes/s)
```

Select the key to be provisioned thanks to stm32key (example for OEM-KEY1 OTP word 144-151):

```
STM32MP> stm32key list
OEM-KEY1 : Hash of the 8 ECC Public Keys Hashes Table (ECDSA is the authentication algorithm)
 for FSBLA or M
OTP144..151
OEM-KEY2 : Hash of the 8 ECC Public Keys Hashes Table (ECDSA is the authentication algorithm)
 for FSBLM
OTP152..159
FIP-EDMK : Encryption/Decryption Master Key for FIP
OTP260..267
EDMK1 : Encryption/Decryption Master Key for FSBLA or M
OTP364..367
EDMK2 : Encryption/Decryption Master Key for FSBLM
OTP360..363
STM32MP> stm32key select OEM-KEY1
OEM-KEY1 selected
```

Verify that the OTP words are still valid:

```
STM32MP> stm32key read
OEM-KEY1 OTP 144: 00000000 lock : 00000000
OEM-KEY1 OTP 145: 00000000 lock : 00000000
OEM-KEY1 OTP 146: 00000000 lock : 00000000
OEM-KEY1 OTP 147: 00000000 lock : 00000000
OEM-KEY1 OTP 148: 00000000 lock : 00000000
OEM-KEY1 OTP 149: 00000000 lock : 00000000
OEM-KEY1 OTP 150: 00000000 lock : 00000000
OEM-KEY1 OTP 151: 00000000 lock : 00000000
OEM-KEY1 is not locked!
OEM-KEY1 is free!
```

Check that the hash is properly loaded in memory:

```
STM32MP> stm32key read 0xc0000000
OEM-KEY1 OTP 144: [c0000000] f96e3aeb
OEM-KEY1 OTP 145: [c0000004] 5c92b5ac
OEM-KEY1 OTP 146: [c0000008] 0abdde9e
OEM-KEY1 OTP 147: [c000000c] bb50fdb4
OEM-KEY1 OTP 148: [c0000010] 18ea200a
OEM-KEY1 OTP 149: [c0000014] 85658efa
OEM-KEY1 OTP 150: [c0000018] 4101b7b7
OEM-KEY1 OTP 151: [c000001c] ececbc99
```

If the hash key is right, the key in OTP can be written in OTP.

```
STM32MP> stm32key fuse -y 0xc0000000
```

Warning: *Fusing is a permanent operation and thus cannot be undone. OTPs are written and permanently locked to avoid further programming access.*

The read command confirms that the OTPs are written and permanently locked (bit 31 in lock word):

```
STM32MP> stm32key read
OEM-KEY1 OTP 144: f96e3aeb lock : 40000000
OEM-KEY1 OTP 145: 5c92b5ac lock : 40000000
OEM-KEY1 OTP 146: 0abdde9e lock : 40000000
OEM-KEY1 OTP 147: bb50fdb4 lock : 40000000
OEM-KEY1 OTP 148: 18ea200a lock : 40000000
OEM-KEY1 OTP 149: 85658efa lock : 40000000
OEM-KEY1 OTP 150: 4101b7b7 lock : 40000000
OEM-KEY1 OTP 151: ececbc99 lock : 40000000
```

Warning: *In the case of EDMK keys, the OTP content of the OTP is not displayed as fuses remain sticky write and reload locked. The lock bit is not displayed because the sticky reload avoids getting OTP status too.*

The device now contains the hash of the table of hashes, to authenticate images.

A similar method can be used to write in OTP the second authentication key by selecting the OEM-KEY2 in the stm32key list. This directly manages the control of the associated OTP 17 that selects the second authentication key management.

The secret key(s) for firmware encryption/decryption must be provisioned. The user must follow the same process listed before by choosing the EDMK1 in words 364 to 367 (or EDMK2 in words 360 to 363).

Warning: *The second secret key is only managed if the associated authentication key has been provisioned.*

Step 3: Provisioning and locking the RMA password

If the provisioning environment is nontrusted, STM32 tools can be used, as detailed in [UM2238]. This nontrusted provisioning environment scenario is out of the scope of the SESIP evaluation.

In a trusted environment, the user can use:

- The STM32PRGFW-UTIL through the STM32CubeProgrammer tool (see [Readme_PRGFW] for details), as described below:

Connect the board with a programming tool.

Load the FlashLayout_STM32PRGFW_UTIL.tsv with STM32CubeProgrammer.

Once loaded, open the OTP panel and scroll to the words 256 to 259. The RMA password (128-bit) must be set and **locked**. Click apply to load the OTP in the board.

- The fuse command is available in the U-Boot context delivered with the OpenSTLinux ecosystem.

1) Write as OTP the password value, for example, `0x78563412`:

```
STM32MP> fuse prog -y 0 256 0x78563412
```

Repeat this for the other part of the RMA passwords (OTP words 257-259).

Note: *This prog command cannot be executed twice because the values of this upper OTP word are ECC-protected. The RMA passwords cannot be read back.*

2) Lock the OTP words:

```
STM32MP> fuse prog -y 0 0x10000100 0x1
```

3) Check that the lock is properly set:

```
STM32MP> fuse read 0 0x10000100
Reading bank 0:
Word 0x10000100: 00000001
```

Step 4: FSBL image signing/encrypting

The FSBL image, loaded by the ROM, must be signed to be used on the certified TOE. STM32 signing tool can be used for that purpose, with the following command:

```
$ STM32MP_SigningTool_CLI.exe -bin fsbl-dk-sdcard.stm32 -pubk publicKey00.pem
publicKey01.pem publicKey02.pem publicKey03.pem publicKey04.pem publicKey05.pem
publicKey06.pem publicKey07.pem - prvk privateKey00.pem -pwd passwd -of 0x00000001 -hv 2.2 -o
 fsbl-sdcardsigned.stm32
```

Same with decryption and authentication:

```
$ STM32MP_SigningTool_CLI.exe -bin fsbl-dk-sdcard.stm32 -pubk publicKey00.pem publicKey01.pem
 publicKey02.pem publicKey03.pem publicKey04.pem publicKey05.pem publicKey06.pem publicKey07.
pem -prvk privateKey00.pem -iv 0x00000000 -pwd passwd -of 0x00000003 -encdc 0x25205f0e -enck
OEM_SECRET.bin -hv 2.2 -o fsbl-sdcardsigned.stm32
```

Using these commands, the FSBL is signed with the key index 0 (private key).

Note: Using an index upper than 0 default revokes the key indexes that are lower than the one used to sign.

Note: When dual authentication is enabled, the signing tool uses the correct authentication (and associated encryption keys): the FSBL for C-M33 uses the OEM_ROT_KEY1, and the FSBL for C-A35 uses the OEM_ROT_KEY2.

The same command must be used to sign the other FSBL firmware if the device selects the boot mode with C-M33 as TDCID.

Step 5: Image programming

Once the image(s) is(are) signed, it can be programmed into the flash memory on the target board with the STM32CubeProgrammer tool. Supported flash memory and its associated flash memory mapping are described in Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C).

Step 6: Lock the device

Without any other modification, the device can perform image authentication, but unauthenticated images can still be used and executed: the device is still *Secured_Unlocked*, as a kind of test mode to check that the PKH is properly set.

Before locking the device, it is important to ensure that all OTPs provisioned are programming locked to prevent any further OTP corruption. It is highly recommended to use different keys when C-M33 is selected as the main processor to ensure that the entire OTPs are written and locked.

As soon as the authentication process is confirmed, the device can be closed, and the user is forced to use signed images. The recommended process follows.

Burn the following OTP to change the product state from default *Secured_Unlocked* to *Secured_Locked*:

- Write `0xFF` in the word 8
- Write `0xF0` in the word 18.
- Write `0x100000` in the word 124 to the disabled scan state.

It is important not to reset the device during this fuse programming sequence as it can damage the device due to an undefined state.

A dedicated command is available using stm32key to directly lock the device when the provisioning is done:

- `STM32MP> stm32key close -y`

An additional OTP can also be set to forbid the debug management:

- Write `0xF00` to lock the debug functionality.

In that state, nonsigned first-stage bootloaders are no longer supported on the target device.

3.2.3 Certified configuration

After installation, the certified device configuration is:

- Life cycle: *Secured_Locked*
- Boot image verification: activated
- Boot image encryption: optional
- Internal and external tamper sources in the TAMP peripheral: deactivated (default), or not.

After installation, the certified device fuse configuration is:

- BSEC status: *BSEC-closed* (see [RM] Section 6.3.7)
- Root key hash 1 (OEM_KEY1_ROT): provisioned in OTP words 144 to 151.
  - The platform uses it to implement authentication and integrity protection.
  - Two algorithms are supported for ECDSA signature verification: P-256 NIST or Brainpool 256T1.
- OEM secret key 1 (OEM_KEY1_EDMK): provisioned in OTP words 364 to 367.
  - The platform is optionally using it for AES-128 GCM FSBL encryption.
- RMA password (OTP_RMA_LOCK_PSWD): provisioned and locked in OTP words 256 to 259.
- Bit 8 *fsbl_decrypt_prio*: set in OTP word 16.
  - To protect the FSBL decryption key against side-channel attacks.
- Bit 0 *Boot_traces_disabled*: set in OTP word 16.

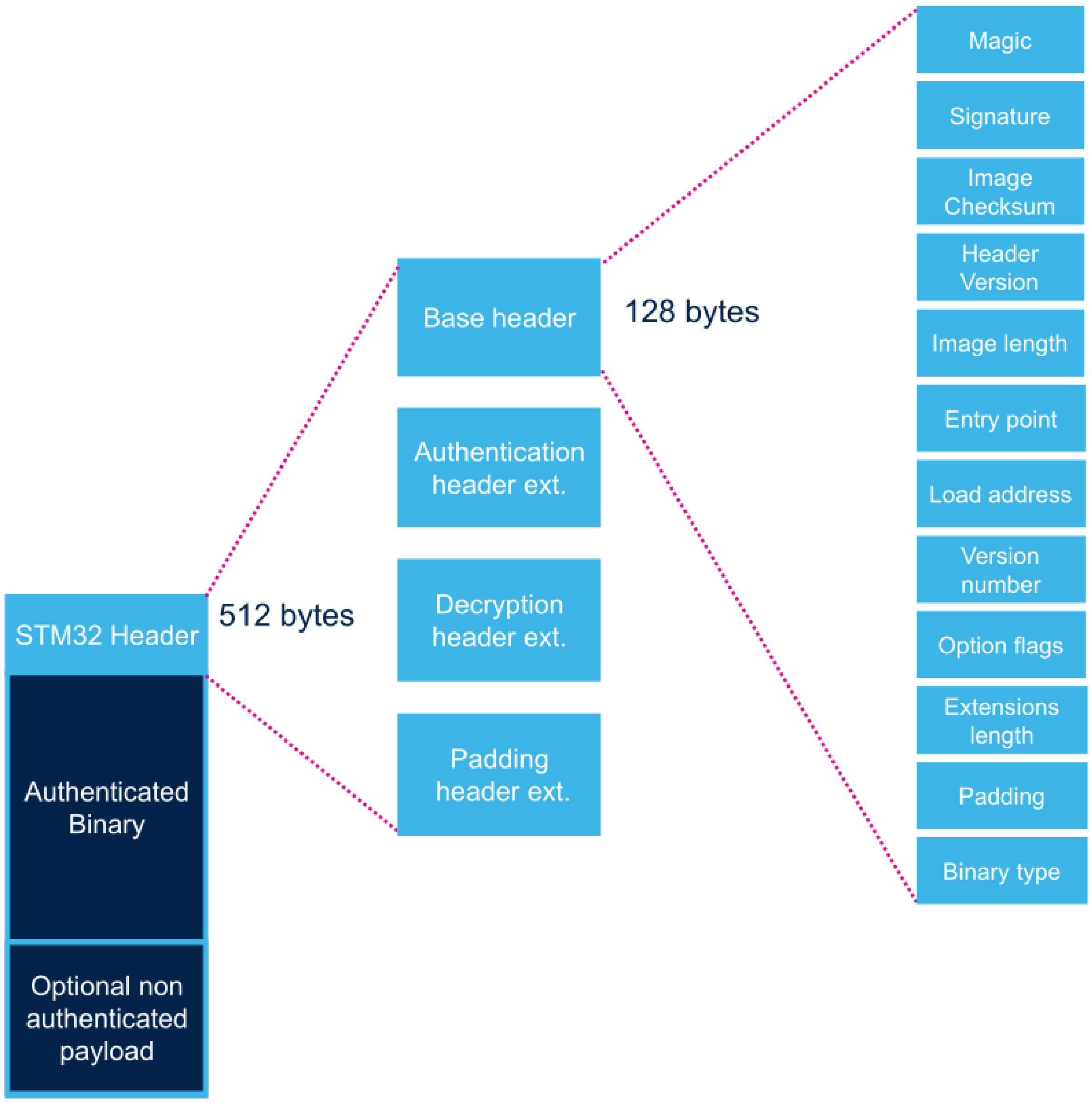
In the case of dual authentication, an additional fuse configuration is expected in the certified device:

- Root key hash 2 (OEM_KEY2_ROT): provisioned in OTP words 152 to 159.
  - The platform uses it to implement authentication and integrity protection of the second image.
  - Two algorithms are supported for ECDSA signature verification: P-256 NIST or Brainpool 256T1.
- OEM secret key 2 (OEM_KEY2_EDMK): provisioned in OTP words 360 to 363.
  - The platform is optionally using it for AES-128 GCM FSBL encryption of the second image.
- Bit 8 *oem_keys2_enable*: set in OTP word 17.

Refer to Section 7 *OTP mapping* in [RM] for details on the fuse configuration in the devices (including endianness).

Each binary image (signed or not) loaded by the platform needs to include a specific STM32 header (version 2.2), added on top of the binary data. This header includes two extension headers: one for FSBL authentication, and one for FSBL decryption. It is shown in Section 3.2.3.

Figure 3. Authenticated STM32 header (with extensions) with binary files



4 Operational user guidance

4.1 User roles

The user role Integrator is the only role involved in the preparative TOE procedure. The Integrator is responsible to:

- Receive the TOE,
- Perform the preparative procedures as described in TOE preparative procedures,
- And integrate the TOE into a final product.

Once the TOE is in certified configuration, the user-operational guidance is described in Operational guidance for the Integrator role.

The Integrator has full access to the TOE security features, as the STM32MP25x MPU devices are delivered in *Secured_Unlocked* state with the secure boot feature deactivated. The Integrator also has full access to the tools needed to program the TOE.

4.2 Operational guidance for the Integrator role

4.2.1 User-accessible functions and privileges (AGD_OPE.1.1C)

The main task of the Integrator is to integrate the TOE into a final product. To this end, the system Integrator has access to interfaces that are unavailable to other users, as described in Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C). The Integrator cannot change any parts inside the TOE but must configure the TOE to make it functional in the final secure state. The Integrator can change parts of the TOE without compromising the security of the TOE.

Follow the procedures described in Secure acceptance to check if the TOE is acceptable for the secure configuration. The secure configuration of the TOE might be impacted when changing some parts of the TOE but may also be impacted when changing some parts located outside the TOE scope. This section describes the changes that the Integrator can make and highlights what is covered in the evaluation scope and what might impact the secure configuration of the TOE.

Boot image authentication scheme

The secure boot on the TOE is certified in the ECDSA P-256 asymmetric crypto scheme configuration, as described in Secure installation and secure preparation (AGD_PRE.1.2C). Such configuration, stored in a dedicated authenticated part in the STM32 header, is summarized in Table 4.

Table 4. **STM32 header information for authentication**

Name		Length (in bits)	Byte offset	Description
Version number		32	96	Image version (monotonic number)

Extension flags		32	104	Bit[0]=1: Authentication is enabled.(1)
				Bit[1]=1: Decryption is enabled.(2)
				Bit[31]=1: Padding extension is enabled.(3)

Authentication extension header	Extension type	32	0	4 bytes in big endian: 'S', 'T', `0x00`, `0x02` = `0x53540002`
	Extension length	32	4	Number of bytes of header extension = 340
	Public key index	32	8	Index of the public key to be used
	Public key number	32	12	Number of public keys in table = 8
	ECDSA algorithm	32	16	1: P-256 NIST; 2: brainpool 256
	ECDSA public key	512	20	ECDSA public key to be used to verify the signature
	Algo and public key1 hash	256	84	Hash of (ECDSA algorithm and public key1)
	...	...	...	Hash of (ECDSA algorithm and public keyX), (X=2 to 7)
	Algo and public key8 hash	256	308	Hash of (ECDSA algorithm and public key8)

1. *Clear the bit to disable it. Authentication can only be disabled on Secured_Unlocked devices.*
2. *Clear the bit to disable it. Authentication can only be disabled on Secured_Unlocked devices.*
3. *Used to have a fixed header size of 512 bytes.*

During certified TOE boot image authentication, if boot image authenticity and integrity are not confirmed, the device either goes to the reboot state (watchdog reset) or switches to a different boot interface state, as defined in BSEC interface in Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C).

Version number and boot image revocation

The platform always verifies the image version and key revocation, as defined below:

- If the platform finds that the version listed in the STM32 header (Table 4) is greater than the one stored in OTP, it updates the version in OTP forbidding future usage of an older version.
  – The word OTP12 stores the counter for the FSBL-A image, while it is the word OTP19 that stores the counter for the FSBL-M image.
  – It is possible to update up to 32 times each FSBL image.
- The platform manages a revocation mask for signature authentication. If it detects a change between the key used in the header and the revocation mask in OTP, then the ECDSA key counter is updated according to the rule pictured in Figure 4. All keys with an ID inferior to the active one are revoked.
  – The word OTP17 stores the mask for authentication key1, while it is the word OTP22 that stores the mask for authentication key2.

Figure 4. **Usage of revocation mask stored in OTP**

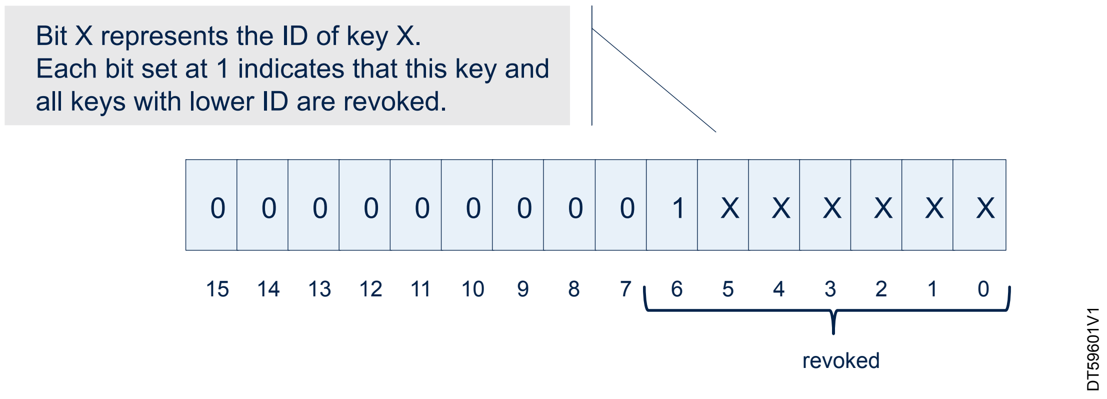

Bit X represents the ID of key X.
Each bit set at 1 indicates that this key and all keys with lower ID are revoked.
0 0 0 0 0 0 0 0 0 1 X X X X X X
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
revoked
DT59601V1

Image encryption

The TOE is certified with image encryption capability enabled or not. Such configuration, stored authentically in the STM32 header, is summarized in Table 5.

Enabling FSBL encryption ensures the confidentiality of the boot image. To ensure protection against side-channel attacks, bit 8 of OTP word 16 must be blown (use SAES).

Table 5. **STM32 header information for encryption**

Name		Length (in bits)	Byte offset	Description

Extension flags		32	104	Bit[0]=1: Authentication is enabled.(1)
				Bit[1]=1: Decryption is enabled.(2)
				Bit[31]=1: Padding extension is enabled.(3)

Decryption header extension	Extension type	32	0	4 bytes in big endian: 'S', 'T', `0x00`, `0x01` = `0x53540001`
	Extension length	32	4	Number of bytes of header extension = 32
	Key size	32	8	Size of extension key (128 bits)
	Derivation constant	32	12	Constant used to derive the decryption key from the controller key stored in OTP
	Plain hash	128	16	128 MSB bits of plain payload SHA256

1. *Clear the bit to disable it. Authentication can only be disabled on Secured_Unlocked devices.*
2. *Clear the bit to disable it. Authentication can only be disabled on Secured_Unlocked devices.*
3. *Used to have a fixed header size of 512 bytes.*

Jump to cold boot image

Out from cold boot, the default protections are properly set by the hardware to comply with the mandatory requirements to allow the platform to securely authenticate (and decrypt) the firmware. The secure ROM code starts as the TDCID and is allowed to control the entire system (RIF configuration).

On a cold boot, the ROM code authenticates (and decrypts) the firmware located in:

- SYSRAM at address `0x0E00 2400`, when C-A35 boot mode is selected.
- RETRAM at address `0x0E08 0000`, when C-M33 boot mode is selected (default boot address).

Depending on the boot mode selection, the platform manages the jump to cold boot image differently.

When C-A35 boot mode is selected:

- In 32-bit mode, the ROM code stores the address of boot context in the Cortex®-A35 r0 register, then jumps directly into the entry point listed in the STM32 header (see below).
- In 64-bit mode (default), the ROM code indicates the entry point in the C-A35 CA35SS_SYSCFG_VBAR_CR register, sets the CA35SS_SYSCFG_AARCH_MODE_CR to `0x1` and reset the Cortex®-A35. The C-A35 then restarts in 64-bit mode, with all the cores running.

When C-M33 boot mode is selected:

- The platform calculates the CRC of the FSBL code loaded in RETRAM and writes it in the RAMCFG CRC register.
- The platform sets the C-A35 in hold boot to avoid any wake-up not authorized by C-M33.
- The platform configures the security of the C-M33 core, configures the C-A35 for DStandby, and then unmaps the ROM memory.
- Finally, the platform configures the C-M33 as TDCID, releases the C-M33 from reset, and goes in wfi() to allow the C-A35 to enter in low-power mode (DStandby).

Related boot information is given in the STM32 header in the table below:

Name	Length (in bits)	Byte offset	Description
Image length	32	76	Length of image in bytes
Image entry point	32	80	Entry point of the image
Binary type	32	108	Used to check binary type. It must be set to `0x30` for FSBLM.

Jump to DStandby wake-up image

When the platform exits DStandby mode, the Integrator has two options:

- SYSRAM retention is selected before DStandby.
  - In that case, the platform checks that the backup register 11 is secure and contains a valid jump address. This address must be a pointer to the retention code in the SYSRAM area.
  - The required nonplatform trusted firmware must not modify the value the C-A35 secure firmware has written in backup register 11.
  - The SYSRAM, which is only accessible by the C-A35, contains the remaining secure code and can be executed directly without authentication.
- SYSRAM retention was not selected before DStandby.
  - In that case, the platform follows the same boot sequence as a standard cold boot, after doing mandatory checks to prevent any incorrect RIF configuration. The Integrator must therefore correctly configure the RIF, as described in Section Resource isolation.
  - In this scenario, the loaded image is limited to C-A35.

Fault injection attacks countermeasures

To meet the target of platform resistance against physical attackers, the Integrator must implement the following software countermeasures within its application when performing sensitive operations (including cryptographic ones):

- Redundancy:
  - For example:
    - Perform the sensitive operation twice and verify that the results are equal.
    - Implement the inverse of the cryptographic operation, as described in the following Section Hardware-accelerated cryptography.
  - In case of verification error:
    - Make sure that the results are erased or cannot be accessed by the user
    - Implement a security response, for example, platform reset.
- Random timing jitter:
  - For example, apply a random loop (using the RNG peripheral) before executing a sensitive operation.

- Execution control flow:
  - For example:
    - Use a finite state machine in which transitions are verified to be legit.
    - Use a scattered known computation with a verified result at the end.
  - In case of control flow error:
    - Implement a security response, for example, a platform reset.

True random number generator

For the device to generate random numbers as specified in NIST SP800-90B, the Integrator must use the TRNG peripheral with the configuration A. Refer to the subsection *validation conditions* of the RNG section in the [RM].

Hardware-accelerated cryptography

The TOE is certified with hardware-accelerated cryptography that is protected against SCA, fault injection, and timing analysis attacks.

To achieve the required resistance against hardware attacks, the Integrator must use the following hardware-accelerated cryptography function, detailed in the corresponding sections of the [RM].

- SAES peripheral:
  - AES-128 and AES-256:
    - Encryption/decryption
    - Authenticated encryption or decryption
    - Cipher-based message authentication code computation
- PKA peripheral:
  - RSA decryption using protected modular exponentiation (MODE=`0x3`).
  - ECC scalar multiplication (MODE=`0x20`) and ECDSA signature (MODE=`0x24`).
- RNG peripheral for cryptographic key generation (ECDH, ECIES, and ECDSA algorithms)
  - In FIPS PUB 186-4 specification, Section B.4 NIST proposes two methods for the generation of the ECC private key (*extra random bits* or *testing candidates*). The Integrator must select one of those two methods when using the RNG peripheral to compute the ECC private keys.

Note: *SAES and PKA cannot operate if the RNG peripheral is not properly initialized, with its AHB clock running.*

When implementing an RSA or ECC function with the PKA peripheral the Integrator must check that, when writing sensitive data such as nonces or private keys into the PKA RAM, the written data is correct and has not been altered before starting the PKA operation.

When implementing an AES function with the SAES peripheral, the Integrator must follow the below guidelines to meet SESIP security assurance level 3 (or equivalent).

- Implement systematically the inverse of the cryptographic operation (encrypt then decrypt or decrypt then encrypt) and compare the result with the initial cryptographic input. The Integrator must implement a random timing jitter between the cryptographic operation and its inverse.
- Implement redundancy when verifying results. The Integrator must implement a random timing jitter between each result comparison.
- Implement a control flow that verifies that each step mentioned above is completed.

Additionally, the Integrator should follow the below guidelines when implementing the AES function with the SAES peripheral:

- Activate the internal tamper 9 dedicated to cryptographic peripherals fault.

Finally, when an error related to the aforementioned countermeasures is detected, the Integrator must take appropriate action according to its security policy (as an example, the application might reset the system).

Cryptographic hash operations, available in the HASH peripheral, must not be used to manipulate sensitive information.

Cryptographic key storage

Using the SAES peripheral the Integrator can protect the integrity and confidentiality of AES 128 or 256-bit keys in its KeyStore, using encryption or authenticated encryption with DHUK. The resulting decrypted keys are automatically stored in SCA-protected, write-only SAES key registers, without disclosing any clear-text key data to the application. Additionally, if the application tries to overwrite part of the key (integrity attack) the whole key is automatically erased.

If the Integrator needs to check that an unwrapped key is not corrupted, he can run a known AES calculation with it and compare the result with an expected value.

Refer to [RM] Section 58.4.14 *SAES operation with wrapped keys* for details.

As with any software dealing with sensitive data, the software driving the SAES peripheral should follow the guidelines described in Section Fault injection attacks countermeasures (for example timing randomization, control flow…).

Note: Keys stored in SAES (in registers or as derived hardware- unique key) are not usable in case of a tamper event, or when the HKLOCK sticky bit is set in the BSEC_LOCKR register.

Nonvolatile storage of secrets in OTP

BSEC implements hardware countermeasures against physical attacks that TOE uses to go to a locked state before the attacker compromises any of the other functional requirements.

The platform uses the BSEC reading, writing, or programming mode lock capability of BSEC to isolate itself from the application. Integrator can do the same for its assets.

After any reading or programming request in BSEC, the peripheral reports a specific status structure when the BUSY bit is cleared. When this structure is different than 0, the read or program operation might have failed. Retry the operation if it is recommended in the table describing the structure.

To benefit from the automatic protection of OTP secrets, when the device is decommissioned for return material for analysis (field return), secure applications must store their secret keys in OTP words 256 to 367.

For more information on BSEC, refer to [RM] Section 6.

Secure execution

Through a mix of special software and hardware features, the devices ensure the correct operation of their functions against abnormal situations caused by programmer errors, software attacks through network access, or a local attempt to tamper code execution. Some of those features are listed below:

- The memory protection unit (MPU) of Cortex®-M33 can restrict the read and write accesses to memory regions (including regions mapped to peripherals), based on operating mode (privileged, unprivileged) or based on data/instruction fetch criteria.
- Independent watchdog 3 (for Cortex®-M33) and independent watchdog 1 (for Cortex®-A35) can be used for secure software monitoring.

Resource isolation

Like the platform ROM code, the Integrator can use resource isolation hardware features such as the Cortex® privilege mode and Arm® TrustZone® security extension, extended to allocated I/Os, DMA channels, memories, and peripherals.

For other leaders than the Cortex®-M33 and Cortex®-A35, the Integrator can use the STM32 resource isolation framework (RIF) to allocate peripherals, fixed-sized embedded memory blocks, and variable-sized address space regions, under the control of the security firmware. Refer to [RM] Section 5.2 for details.

Note: The RETRAM used for restoring RIF configuration following a low-power mode must be reserved to a trusted domain CPU only (using RISAB5).

Before setting the D1 domain in DStandby mode, the Integrator must set the RIF configuration as listed hereafter. This configuration must be set in the nonplatform trusted firmware boot before the DStandby exit that triggers the platform ROM code execution. For cold boots, the Integrator has nothing to do, as the platform correctly manages the default parameters.

a) The platform ensures exclusive access to the peripherals listed in Table 7, using two methods:

- RIF block configuration is properly configured and **locked** (see Table 7)

- If the RIF configuration of a controller listed in Table 6 corresponds to CID=`0x1` secure, the platform verifies if the corresponding peripheral is running or not, using associated RCC registers (clock enabled). In such a case, the platform goes to the failure state, as at least one controller is accessing the same resources as the platform ROM code.
- All HPDMA channels assigned to CID1 secure must not be running.

Table 6. **RIF controller peripheral checks**

Bus controller peripheral	RIFSC RIMU index	RIFSC RISUP index	DStandby exit RISUP lock check
SDMMC1	1	76	Yes
SDMMC2	2	77	Yes
SDMMC3	3	78	Yes
USB3DR	4	66	Yes
USBH	5	63	Yes
ETH1	6	60	Yes
ETH2	7	61	Yes
PCIE	8	68	Yes
GPU	9	79	Yes
DCMIPP	10	87	Yes
LTDC	11, 12, 13	NA	No
VDEC	14	89	Yes
VENC	15	90	Yes

Table 7. **Secure peripherals RIF configurations used by ROM code**

Peripheral	RIF resource number	RIF block	Comment	Secure level checked	Check at cold boot	DStandby exit
BSEC clock	R103	RCC	-	Secure	Yes	Yes
CPU1 boot, reset, IWDG management	R70	RCC	-	Secure	Yes	Yes
CPU PWR1	R2	PWR	Power control	Secure	Yes	Yes
STGEN	R33	RCC	Flexgen	Secure	Yes	Yes
RNG	R92	RIFSC	-	Secure	Yes	Yes
PKA	R93	RIFSC	-	Secure	Yes	Yes
SAES	R94	RIFSC	Depends on OTP 16 fsbl_decrypt_prio	Secure	Yes	Yes
HASH	R95	RIFSC	-	Secure	Yes	Yes
CRYP1	R96	RIFSC	Depends on OTP 16 fsbl_decrypt_prio	Secure	Yes	Yes
SYSRAM/RISAB1 and RISAB2	R74	RCC	-	Secure	Yes	Yes
CA35SS	R106	RCC	-	Secure	Yes	Yes

b) SDMMC1 expected RIF configuration is defined in Table 8.

Table 8. **SDMMC1 RIF configuration**

Function	RIF resource number	RIF block	Resource description	Secure level set	Check at cold boot	Check at DStandby exit
Root clock SDMMC1	R51	RCC	Flexgen 51	Nonsecure	Yes	Yes
Clock GPIOE	R94	RCC	GPIO E clock config	Nonsecure	Yes	Yes
I/O voltage protection	R6	PWR	VDDIO1 (PWR_C8)	Secure	Yes	Yes
GPIO PE3	R3	GPIOE	SDMMC1_CK	Nonsecure	Yes	Yes
GPIO PE2	R2	GPIOE	SDMMC1_CMD	Nonsecure	Yes	Yes
GPIO PE4	R4	GPIOE	SDMMC1_D0	Nonsecure	Yes	Yes
SDMMC1	R76	RIFSC	SDMMC1 controller	Nonsecure	Yes	Yes

c) SDMMC2 expected RIF configuration is defined in Table 9.

Table 9. **SDMMC2 RIF configuration**

Function	RIF resource number	RIF block	Resource description	Secure level set	Check at cold boot	Check at DStandby exit
Root clock SDMMC2	R52	RCC	Flexgen 52	Nonsecure	Yes	Yes
Clock GPIOE	R94	RCC	GPIO E clock config	Nonsecure	Yes	Yes
I/O voltage protection	R5	PWR	VDDIO2 (PWR_C7)	Secure	Yes	Yes
GPIO PE14	R14	GPIOE	SDMMC2_CK	Nonsecure	Yes	Yes
GPIO PE15	R15	GPIOE	SDMMC2_CMD	Nonsecure	Yes	Yes
GPIO PE13	R13	GPIOE	SDMMC2_D0	Nonsecure	Yes	Yes
SDMMC2	R77	RIFSC	SDMMC1 controller	Nonsecure	Yes	Yes

d) OCTOSPI1 expected RIF configuration is defined in Table 10.

Table 10. **OCTOSPI1 RIF configuration**

Function	RIF resource number	RIF block	Resource description	Secure level set	Check at cold boot	Check at DStandby exit
Root clock Octospi1	R48	RCC	Flexgen 48	Nonsecure	Yes	Yes
Octospi1 clock gating	R110	RCC	Octospi1 clock and reset	Nonsecure	Yes	Yes
Octospi1 interface	R74	RIFSC	Octospi1 interface	Nonsecure	Yes	Yes
IOM	R111	RIFSC	I/O manager	Secure	Yes	No
I/O voltage protection	R0	PWR	VDDIO3 & VDDIO4 (PWR_CR1)	Secure	Yes	No
Clock GPIOB	R91	RCC	GPIO B clock config	Nonsecure	Yes	No
Clock GPIOD	R93	RCC	GPIO B clock config	Nonsecure	Yes	No
GPIO PD 0, 3-5	R0, 3-5	GPIOD	Port 1 config single wire	Nonsecure	Yes	No
GPIO PD 1, 2, 6-11	R1, 2, 6-11	GPIOD	Port 1 HyperFlash™	Nonsecure	Yes	No
GPIO PE 0, 1, 8,10	R0, 1, 8, 10	GPIOE	Port 2 config single wire	Nonsecure	Yes	No
GPIO PE 2-7, 9, 11	R2-7, 9, 11	GPIOE	Port 2 HyperFlash™	Nonsecure	Yes	No

e) FMC expected RIF configuration is defined in Table 11.

Table 11. **FMC RIF configuration**

Function	RIF resource number	RIF block	Resource description	Secure level set	Check at cold boot	Check at DStandby exit
Root clock FMC	R50	RCC	Flexgen 50	Nonsecure	Yes	No
FMC clock gating	R112	RCC	FMC clock and reset	Nonsecure	Yes	No
Clock GPIOB	R91	RCC	GPIO B clock config	Nonsecure	Yes	No
Clock GPIOD	R93	RCC	GPIO D clock config	Nonsecure	Yes	No
Clock GPIOE	R94	RCC	GPIO E clock config	Nonsecure	Yes	No
I/O voltage protection	R5	PWR	VDDIO2 (PWR_C7)	Secure	Yes	Yes
GPIO PB 13-14	R13-14	GPIOB	8-bit config (12 I/Os)	Nonsecure	Yes	No
GPIO PD 12,14-15	R12, 14-15	GPIOD	8-bit config (12 I/Os)	Nonsecure	Yes	No
GPIO PE 6-9, 11-15	R6-9, 11-15	GPIOE	8-bit config (12 I/Os)	Nonsecure	Yes	No
GPIO PB 5-11	R5-11	GPIOB	16-bit config added	Nonsecure	Yes	No
GPIO PD13	R13	GPIOD	16-bit config added	Nonsecure	Yes	No
FMC R0	R0	FMC	FMC common resource	Nonsecure	Yes	No
FMC R5	R5	FMC	FMC NAND controller	Nonsecure	Yes	Yes

Return material for analysis (field return)

The final product manufacturer must invoke this functionality before returning the TOE device for failure analysis to STMicroelectronics. It is defined in [RM] Section 6.3.7 *Opening BSEC*.

The TOE is certified with a provisioned and locked 128-bit RMA password. The Integrator has the privilege and responsibility to secure it following the recommendations described in Secure installation and secure preparation (AGD_PRE.1.2C).

Anti-tamper peripheral

The platform resets with internal and external tamper sources deactivated in the TAMP peripheral.

To meet the platform resistance against physical attackers, the Integrator should configure the TAMP peripheral with anti-tamper methods available in the device when using the following functions:

- Nonvolatile fuse storage (including derived hardware unique key)
- Volatile secret key storage in backup domain registers
- Embedded SRAM1 storage
- Crypto functions in hardware engines (SAES, CRYP, HASH, PKA)

Those methods are described in [RM] Section 74 tamper and backup registers (TAMP).

When a tamper flag raises (tampering detected), the Integrator must implement a security response, for example, a platform reset.

Secure debug

The secure Cortex®-A35 debug feature while the platform execution is not available to the users. It is implemented but not accessible.

For the other debug features of the product, the platform is certified with all debug features disabled out of cold boot reset. This debug protection can be frozen anytime until the next reset by setting the DENREG bit in the BSEC peripheral. The platform systematically sets this bit when *debug_lock* fuses are burnt in OTP word 18. Refer to the JTAG interface in Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C) for details.

When the Integrator activates any debug capability on the device following a cold boot, the debugging of the platform ROM code remains not accessible.

4.2.2 Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C)

To exercise the functions and privileges described in User-accessible functions and privileges (AGD_OPE.1.1C), the Integrator interacts with the TOE interfaces described in this section:

- Physical chip interface
- BOOT pins
- BSEC interface
- Boot image interface
- True random number generation interface
- Cryptographic functions interface
- JTAG interface
- RIF interface

This section highlights all security parameters under the control of the Integrator, indicating the secure values as appropriate.

Physical chip interface

After providing the power supply and clocks and deasserting the reset signal, the platform starts on secure ROM code. During runtime, the Integrator code can put the device in DStandby low-power mode or can assert the reset of the C-A35 processor. The platform manages both cases, ensuring that C-A35 secure only executes authentic code.

Optionally, the Integrator can activate the tamper-detection and response hardware present on the physical chip interface.

Method of use:

- [Cold boot]
  - Activate the power supplies and clocks of the platform.
  - Reset the device (power-on-reset, global system reset).
  - C-A35 securely executes platform ROM code, triggering image download via boot interface(s).

- [DStandby exit without SYSRAM retention]
  - During runtime, the application triggers an entry to DStandby without SYSRAM retention.
  - The device exits DStandby, restoring the power supply to the C-A35 domain.
  - C-A35 securely executes platform ROM code, triggering image download via an external flash interface.
- [DStandby exit with SYSRAM retention]
  - During runtime, the application triggers an entry to DStandby with SYSRAM retention.
  - The device exits DStandby, restoring the power supply to the C-A35 domain.
  - C-A35 securely executes platform ROM code, then branches to the DStandby wake-up image in SYSRAM secure.
- [C-A35 processor reset]
  - During runtime, the application resets the C-A35.
  - C-A35 securely executes platform ROM code, triggering image download via an external flash interface.

Parameters:

- Refer to [RM] Section 20.3 about reset pins, and Section 33 for CPU wake-up, interrupt, and event generation.
- Refer to [RM] Section 19.6.8 about D1 domain exit from DStandby mode.
- Refer to [RM] Section 19.5.1 about activation of SYSRAM retention.
- Refer to [RM] Section 74.6 about TAMP registers.

Actions:

- Define the reset behavior of the platform using BOOT pins and OTP words (see next two subsections)
- Activate SYSRAM retention before switching D1 main to DStandby mode.
- Set C1RST in RCC_C1RSTCSETR to reset C-A35 and its related watchdogs (IWDG1/2) if required.
- Activate the tamper detection method defined in [RM] Section 74.3.10 (optional)
- Activate backup registers and other device secrets erase (see [RM] Section 74.3.11) (optional)

Errors:

- The platform goes to a locked state if the platform's authenticity or integrity cannot be ensured. A power-on-reset or global system reset is necessary to exit this locked state.

BOOT pins

After providing the power supply and clocks but before deasserting the reset signals, the user can instruct the platform how to manage the boot process via dedicated boot pins.

Method of use:

- If serial boot is needed, connect the USB Type-A host port of the TOE to the user's computer.
- Modify the electrical values on the boot pins.
- Activate the power supplies and clocks of the platform or wake up the platform.
- C-A35 securely executes platform ROM code that verifies the state of the boot pins before starting the secure initialization process.

Parameters:

- BOOT0 to BOOT3 pins (for pinout information refer to [DS]). The effect of those pins depends on the value of bits 24 and 25 in OTP word 11 (*bootpins_layout_sel*), as summarized in Table 12.

Table 12. **BOOT pins effect**

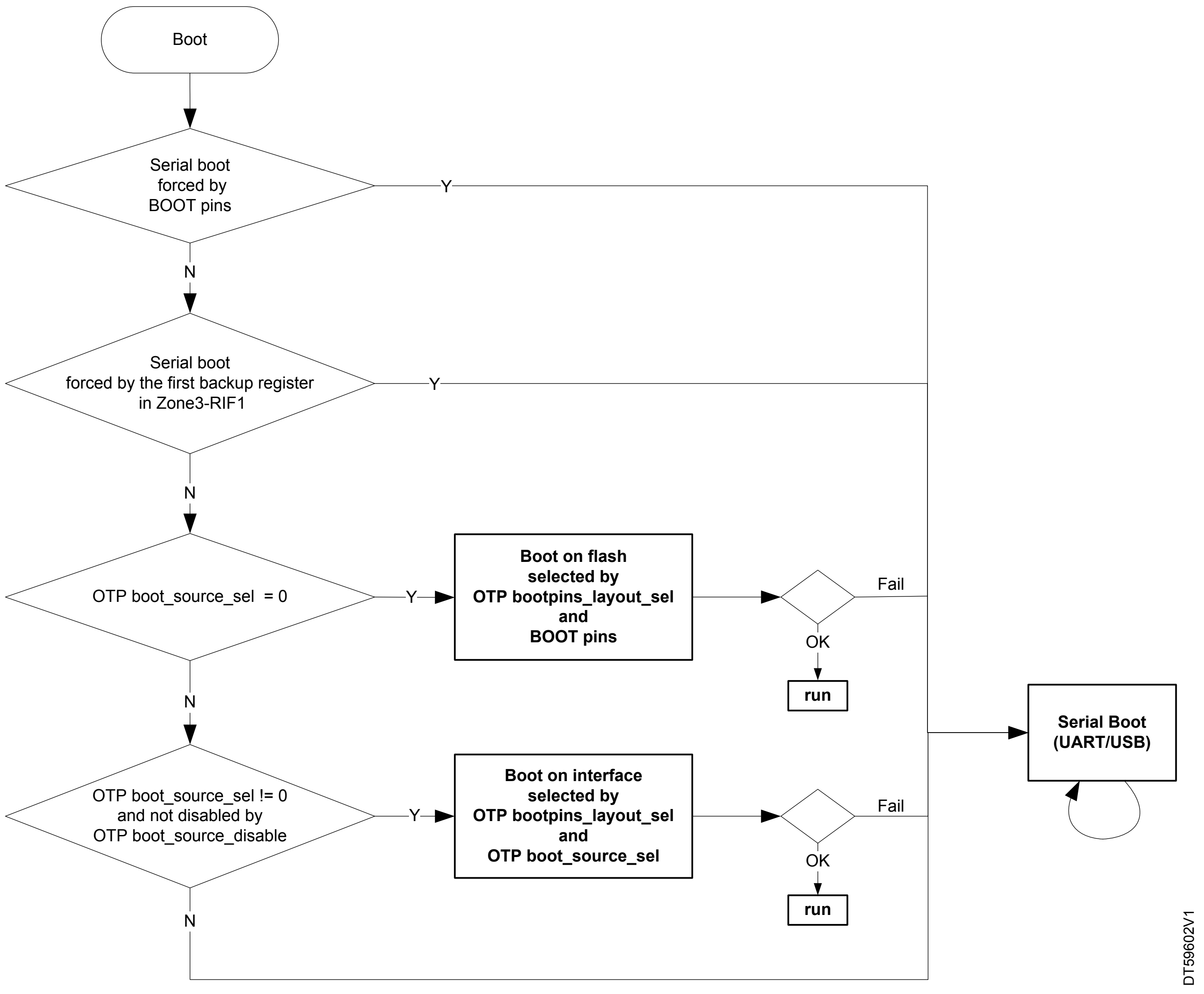
BOOT[3:0] pins value	BOOT pins layout selection (OTP 11, bits[25:24] bootpins_layout_sel)							
	`0b00` (default)				`0b01`	`0b10`	`0b11`	
	Cortex®-A35 TD	Cortex®-M33 TD	Cortex®-M33 TD		Cortex®-A35 TD	Cortex®-M33 TD	Cortex®-M33 TD	
			Cortex®-A35 boot	Cortex®-M33 boot			Cortex®-A35	Cortex®-M33
`0x0`	Serial boot (UART/USB)							
`0x1`	SD card	-	-	-	SD card	SD card	Serial NAND	Serial NOR
`0x2`	e-MMC	-	-	-	e-MMC	e-MMC	e-MMC	Serial NOR
`0x4`	Serial NOR	-	-	-	Serial NOR	Serial NOR	SLC NAND	Serial NOR
`0x5`	-						e-MMC	Serial NOR
`0x6`	-						e-MMC	HyperFlash™
`0x7`	-	SD card	-	-	HyperFlash™	HyperFlash™	Serial NAND	HyperFlash™
`0x8`	-	e-MMC	-	-	Serial NAND	Serial NAND	e-MMC	HyperFlash™
`0x9`	-						SD card	Serial NOR
`0xA`	-						SD card	HyperFlash™
`0xB`	-	Serial NOR	-	-	SLC NAND	SLC NAND	SLC NAND	HyperFlash™
`0xD`	-	-	e-MMC	Serial NOR	SD card	SD card	SD card	Serial NOR
`0xE`	-	-	SD card	Serial NOR	e-MMC	e-MMC	SD card	HyperFlash™
`0xF`	Serial boot (UART/USB)							

- The first backup register defined in the Zone3-RIF1 (read/write nonsecure). See [RM] Section 74.3.8 for details.
  – Set the value to `0xFF` and reset the device. It forces the mode to UART and USB mode.

Actions:

- Set electrical values to BOOT pins. Those are connected to user switches SW1 on the STM32MP257F-EV1 evaluation kit.
- Depending on the parameter values (BOOT pins parameters, BSEC interface BOOTROM_CONFIG_2) the overall boot source selection options available to the Integrator are summarized in Figure 5.

Figure 5. ROM boot source selection


Boot
Serial boot forced by BOOT pins
Y
N
Serial boot forced by the first backup register in Zone3-RIF1
Y
N
OTP boot_source_sel = 0
Y
Boot on flash selected by OTP bootpins_layout_sel and BOOT pins
Fail
OK
run
N
Serial Boot (UART/USB)
OTP boot_source_sel != 0 and not disabled by OTP boot_source_disable
Y
Boot on interface selected by OTP bootpins_layout_sel and OTP boot_source_sel
Fail
OK
run
N
DT59602V1

Errors:

- Not applicable.

BSEC interface

After providing the power supply and clocks but before deasserting the reset signal, the user can instruct the platform how to manage the boot process using dedicated OTP words.

Other usage of the BSEC peripheral is up to the user of the platform.

Method of use:

- [Cold boot]
  - Activate the power supplies and clocks of the platform.
  - Reset the device. The platform ROM code starts.
  - The platform branches to authenticated code in SRAM after locking some of its assets in OTP.
  - Trusted domain CPU with TDCID performs read and write access to BSEC registers.
- [Warm reset/DStandby exit or C-A35 reset]
  - Reset the device following a DStandby exit or reset only the C-A35 processor.
  - Depending on the case, the C-A35 processor executes secure code in ROM before branching to application code in SYSRAM. Some platform assets are locked out of reset.
  - Trusted domain CPU with TDCID performs read and write access to BSEC registers.
- Via BSEC registers trusted domain CPU can modify the platform secure boot behavior blowing fuses.

Parameters:

- Refer to [RM] Section 6.5 BSEC registers.
- BOOT pins layout selection in fuse word 11 (bit 24 and 25, BOOTROM_CONFIG_2), defined in Table 12. Those fuses also select the trusted domain CPU following a cold boot (see [RM] Section 5.2.6).
- Flash memory boot configuration in fuse word 11 (BOOTROM_CONFIG_2)

OTP field	Offset	Size (in bits)	Default	Description
boot_source_sel	[29:26]	4	0	[0-15]: Select one among the 16 possible boot sources. Refers to BOOT pins [0:3] value.
boot_source_disable	[23:16]	8	0	If different from zero, each bit disables a boot source. bit[0]: FMC NAND, bit[1]: QSPI NOR, bit[2]: eMMC, bit[3]: SD, bit[4]: UART. Default to UART if all are disabled.

- Serial boot configuration in fuse word 11 (BOOTROM_CONFIG_2)

OTP field	Offset	Size (in bits)	Default	Description
boot_source_sel	[29:26]	4	0	Keep to zero (virgin) to boot on the serial interface.
disable_uart	[14:10]	5	0	`0b00001`: disables USART2 `0b00010`: disables UART5 `0b00100`: disables UART6 `0b01000`: disables UART8 `0b10000`: disables UART9

Actions:

- If the external flash memory boot configuration must be changed, write to fuse word 11 (BOOTROM_CONFIG_2). If the serial boot configuration must be changed, write to fuse word 11 (BOOTROM_CONFIG_2).
- For other changes to on-chip nonvolatile memory, a trusted domain CPU with TDCID can read and write to BSEC registers (refer to [RM] Section 6).

Errors:

- Refer to [RM] Section 6.3.15 BSEC error management.

Boot image interface

Depending on the mode of operation, the platform uses the boot image stored in SYSRAM or RETRAM, after updating the RIF firewall configuration. Access to those memories is not possible before initiating a cold boot.

Method of use:

- [Cold boot]
  - Prepare the boot image in external flash (see Step 5: Image programming in Section 3.2.2), or ready the STM32 boot tool (see Software setup procedure in Section 3.2.1).
  - Follow the method in Section Physical chip interface, [Cold boot] mode of operation.
- [Warm reset, DStandby exit with SYSRAM retention]
  - Prepare the boot image in SYSRAM.
  - Follow the method in Section Physical chip interface, [DStandby exit with SYSRAM retention] mode of operation.
- [Warm reset, DStandby exit without SYSRAM retention]
  - Prepare the boot image in external flash (a method outside the scope of this document).
  - Follow the method in Section Physical chip interface, [DStandby exit without SYSRAM retention] mode of operation.

Parameters:

- Boot image, defined in Jump to cold boot image of Section 4.2.1.
- Wake-up image, defined in Jump to DStandby wake-up image of Section 4.2.1.

Actions:

- [Cold boot or DStandby exit without SYSRAM retention or C-A35 processor reset]: No direct action is possible to boot the image area in SYSRAM and RETRAM.
- [DStandby exit with SYSRAM retention] Write DStandby wake-up image in SYSRAM. After exit from DStandby with SYSRAM retention, the platform executes this image.

Errors:

- [Cold boot] If the boot image selected by the BOOT pins and OTP parameters fails the platform goes to serial boot mode, as pictured in Figure 5.
- [DStandby exit without SYSRAM retention, or C-A35 processor reset] If the boot image stored in the selected external flash device fails the platform goes to an error state (blocking failure).
- [DStandby exit with SYSRAM retention] If the DStandby wake-up image is not located in SYSRAM, in an area reserved for C-A35 secure, the platform goes to the external flash memory boot mode (like for cold boot).

True random number generation interface

The platform includes an RNG peripheral compliant with NIST SP800-90B recommendations. When the platform is not executing, the application must use this peripheral to generate true random numbers.

Method of use:

- The firmware interacts with the hardware TRNG through a bank of memory-mapped registers.

Parameters:

- Refer to [RM] Section 57.7 *RNG registers*.

Actions:

- Refer to [RM] Section 57.3.4 *RNG initialization*.
- Refer to [RM] Section 57.3.5 *RNG operation*.
- Refer to [RM] Section 57.3.8 *RNG low-power use*.
- Refer to [RM] Section 57.4 *RNG interrupts*.

Errors:

- Refer to [RM] Section 57.3.7 *Error management*.

Cryptographic functions interface

When the platform is not executing, the platform provides the application with a set of hardware IP registers to access cryptographic operations and cryptographic key store resources. Some of those cryptographic operations are protected against side-channel attacks (AES, modular exponentiation, signature, ECC scalar multiplication).

Method of use:

- The firmware interacts with the hardware SAES and PKA through a bank of memory-mapped registers, and the embedded RAM in the peripheral (for the PKA).

Parameters:

- Refer to [RM] Section 58.8 *SAES registers*.
- Refer to [RM] Section 62.7 *PKA registers*, and Section 62.4 (for PKA RAM parameters).

Actions:

- Refer to [RM] SAES sections 58.4, 58.5 and 58.6.
- Refer to [RM] PKA sections 62.3 and 62.6.

Errors:

- Refer to [RM] Section 58.4.19 *SAES error management*.
- Refer to [RM] Section 62.3.7 *PKA error management*.

JTAG interface

Standard JTAG with SWD interface allows debugging of the Integrator application. It is used according to [IEEE1149]. When the device is in the *Secured_Locked* state, all debug features are disabled after a cold reset. The JTAG/SWD interface remains enabled on reset only to inject the RMA password to request the transition to *RMA* state.

The Integrator can use the BSEC peripheral to enable debugging of its code or prevent such debugging until the next system reset.

The platform ROM code can never be debugged or traced using the JTAG interface.

Method of use:

- JTAG physical link to transport RMA password value. The method is defined in [RM] Section 6.3.7 *Opening BSEC*.
- Following platform secure boot use BSEC_DENR register to allow debugging of the Integrator application. JTAG/SWD can then be used with a debugger (C-A35 or C-M33).

Parameters:

- OTP_RMA_LOCK_PSWD, stored in OTP words 256 to 259 (see [RM] Section 7).
- BSEC_DENR register (refer to [RM] Section 6.3.10 for details).
- DENLOCK bit in BSEC_LOCKR register. Set to make writes to the BSEC_DENR register ignored until the next system reset.
- *debug_lock* fuses in OTP fuse word 18. Burn any bits 8 to 11 in OTP18 (BOOTROM_CONFIG_9) to make writes to the BSEC_DENR register ignored permanently.

Actions:

- Inject the RMA password through JTAG/SWD under reset. If the received value is identical to the programmed OTP_RMA_LOCK_PSWD value, the hardware starts a transition to the *RMA* state.
- Connect a debugger for a debug session (C-A35, C-M33, or both).

Errors:

- When injecting the RMA passwords:
  - Device state does not change following a reset when the provided RMA password is wrong.
  - If RMA trials are more than four RMA attempts, the device state never changes.
- When connecting a debugger:
  - The debug session fails if the BSEC_DENR configuration is not correct.

RIF interface

Using the nonplatform trusted firmware the Integrator shall configure the RIF as defined in tables 6 to 11 of Section 4.2.1.

Method of use:

- The secure processor performs write access to the RIF registers to store the configuration at cold boot and lock to ensure that no change occurs when restoring from standby.

Parameters;

- RIF parameters are described in the tables 6 to 11 of Section 4.2.1. They must be aligned with the chosen boot device.

Actions:

- Verify the value of each configuration and ensure that the locks are properly set.

Errors:

- The value is not properly set in registers.
- The locks are defined in the registers. A cold boot is required to clean the values.

4.2.3 Security-relevant events (AGD_OPE.1.4C)

Once configured according to Section 3.2: Secure installation and secure preparation (AGD_PRE.1.2C), the platform triggers the following security-relevant events when the Integrator uses the functions and privileges described in Section 4.2.1.

- Images authenticity or integrity violation:
  – An incorrect signature verification ends with a boot error (red LED). It can be due to an incorrect key pair usage or a corrupted memory. The second image (backup) is loaded to check if it can boot, otherwise the boot process is stalled.
- Images decryption error:
  – An incorrect decryption is detected when the hash of the plain text does not match the header one. It ends in a boot error (red LED). The second image (backup) is loaded to check if it can boot, otherwise the boot process is stalled.
- Unexpected boot image:
  – Wrong header, nonconsistent header (for example, image size not consistent with header), illegal parameter in header.
  – Image stored at the wrong address.
  – All these unrecoverable issues end in a backup image boot or a ROM boot failure.
- Revoked boot image:
  – The signed key in the boot image is revoked in fuses.
  – The image version number is lower than the one stored in fuses.
  – All these unrecoverable issues end in a backup image boot or a ROM boot failure.
- Erroneous values found in the OTP of the security configuration at boot time:
  – Abort of the boot sequence.
- Closed JTAG/SWD access violation:
  – The connection request is not transmitted to the access port and debug port. The request is ignored.
- Erroneous RMA password injection:
  – The platform ignores the RMA request and restarts in its current life cycle state at the next boot.
- Detection of attacks to RNG, SAES, and PKA by an attacker with physical access
  – Peripheral goes to locked mode + generation of internal tamper (tamp_itamp9 input), disabled by default.
- Error when reading a fuse word
  – Refer to [RM] Section 6.3.6 BSEC read and programming status reporting.
- Illegal access to a resource protected by RIF
  – Refer to [RM] Section 5.2.4 RIF illegal access definition.
- Attempts to compromise RNG entropy properties by disturbing physical RNG interfaces:
  – The RNG hardware raises alarms on clock and noise source defects, with a possible interruption to a user-defined implementation error handler.

4.2.4 Security measures (AGD_OPE.1.6C)

This section describes the Integrator user role and the security measures to be followed to fulfill the security objectives for the operational environment as described in the [ST] Section 2.1.

- The operating system or application code is expected to verify the correct version of all platform components that it depends on (hardware revision 2.1).
  – Refer to the method in How to accept an STM32MP25x MPU device.

- The operating system or application code must use the secure boot feature, by:
  - Copying twice the secure boot image, as described in Section Jump to cold boot image.
  - Follow the guidelines on RIF config setup to have a secure boot functional following DStandby exit. Refer to Resource isolation for details.
  - When needed, store the wake-up image in SYSRAM, as described in Jump to DStandby wake-up image.

Additionally, the Integrator user role must take the following measures:

- Verify the genuineness of the TOE as described in Section 3.1: Secure acceptance.
- Follow all guidelines described and referenced in Section 3.2: Secure installation and secure preparation (AGD_PRE.1.2C).
- Follow all guidelines described in Section 4.2.1: User-accessible functions and privileges (AGD_OPE.1.1C) and Section 4.2.2: Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C) to integrate the TOE into a full IoT solution.
- In the manufacturing phase, the Integrator must securely provision the TOE immutable data specific to the Integrator or specific to the product as stated in Step 2: Provisioning of secrets.
- Once the Integrator finishes the production of a final user application, he must set the STM32MP25x device hardware static protections as stated in Section 3.2.2: Secure installation and Section 3.2.3: Certified configuration.

4.2.5 Modes of operation (AGD_OPE.1.5C)

This section identifies all possible modes of operation of the platform (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

[Cold boot]

The Integrator selects the boot device from where the ROM code loads the image. After a cold boot reset, the platform requests the nonsecure ROM to load the boot image in embedded SRAM, then branches the boot CPU to execute this image if it is authentic. The Integrator can also select the boot CPU to be the trusted domain CPU.

In case of errors detected during the image authentication/decryption or validation (header parameters), the platform aborts the boot process and stays in a failure endless loop.

[DStandby exit without SYSRAM retention, or C-A35 processor reset]

The Integrator wakes up the C-A35 after a low-power mode or C-M33 boot. After C-A35 reset the platform requests the nonsecure ROM to load from external flash memory the boot image in SYSRAM, then branches the C-A35 to execute this image if it is authentic.

If the platform finds that the RIF configuration is not the one expected, it aborts the boot process and remains in an endless loop of failure.

In case of errors detected during the image authentication/decryption or validation (header parameters), the platform also aborts the boot process and stays in a failure endless loop.

[DStandby exit with SYSRAM retention]

The Integrator wakes up the C-A35 after a low-power mode or C-M33 boot. After the C-A35 reset, the platform checks the tamper 11 configuration, which must be secured, and reads the CPU jump address which must be inside the SYSRAM area. If both checks are OK the C-A35 secure jumps to the specified address to continue the warm boot process.

If platform checks fail or backup register security level, the platform switches to the mode of operation [DStandby exit without SYSRAM retention].

[RMA password injection]

The Integrator owning the RMA password value injects it over the JTAG/SWD interface while maintaining the platform under reset as explained in subsection *Opening BSEC* of [RM] Section 6.3.7. If the RMA password value is wrong, the platform ignores the RMA request and waits for a reboot.

5 Annexes

5.1 Nonsecure ROM bootloader interfaces

When allowed by secure boot ROM, the nonsecure ROM bootloader of the device supports the following flash memory interfaces:

- Serial NOR flash memory via Quad-SPI peripheral
- Serial NAND flash memory via Quad-SPI peripheral
- Parallel NAND flash memory via FMC peripheral
- SD card via SDMMC peripheral
- eMMC via SDMMC peripheral

Depending on the case, the storage of the boot image must follow the rules defined in the [WIK1] product wiki page.

Revision history

Table 13. **Document revision history**

Date	Revision	Changes
12-Dec-2024	1	Initial release.

Contents

List of tables

List of figures

IMPORTANT NOTICE – 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 acknowledgment.

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. For additional information about ST trademarks, refer to www.st.com/trademarks. 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.

© 2024 STMicroelectronics – All rights reserved