STM32C5xxxx security guidance for SESIP level 3 certification

Introduction

This document describes the preparative procedures and operational user guidance of STM32C5xxxx to make a secure system solution according to SESIP level 3 certification.

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

Table 1. Applicable products

Reference	Products
STM32C5xxxx	STM32C542xx, STM32C562xx, STM32C5A3xx

1 General information

The STM32C5xxxx devices are based on Arm® cores. For more information on Arm® Cortex®-33, refer to the technical reference manuals, available from www.arm.com website.

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

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

Table 2. Specific acronyms

Acronym	Description
HUK	Hardware unique key
CLI	Command line interface
HDP	Secure hide protection
HW	Hardware
IoT	Internet of Things
MCU	Microcontroller
TOE	Target of evaluation
RDP	Readout protection
RMA	Return material for analysis
SESIP	Security evaluation standard for IoT platforms
SFR	Security-functional requirement
WRP	Write protection

Certification applies to several STM32 products. Main differences between these products are that only STM32C5A3xx devices embed SAES, PKA, and CCB peripherals. This is described in the table below.

In the next sections, the reader must take care that SAES, PKA, CCB references only apply to STM32C5A3xx products.

Table 3. Peripherals in TOE per product

Component/interface	Description	STM32C542xx	STM32C562xx	STM32C5A3xx
System				
CPU	Arm® Cortex®-M33	Y	Y	Y
Debug ports	JTAG/SWD, DBGMCU	Y	Y	Y
Memories	FLASH	256 KB	512 KB	1 Mbyte
Peripherals	TAMP, RCC, SBS	Y	Y	Y
Cryptography				
TRNG	-	Y	Y	Y
SAES	-	-	-	Y
PKA	-	-	-	Y
CCB	-	-	-	Y

2 Reference documents

Table 4. List of reference documents

Reference	Document title and revision
[ST]	Security target *STM32C5xxxx SESIP security target for PSA Certified HW RoT* (ST0063), Rev 1
[RM]	Reference manual *STM32C5 Series Arm®-based 32-bit MCUs* (RM0522), Rev 1
[UM2237]	User manual *STM32CubeProgrammer software description* (UM2237), Rev 30
[IEEE1149]	EEE 1149.1–2013
[IHI0031]	Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2
[SM]	PSA Certified *Platform Security Model 1.1* (JSADEN014)
[PM0264]	*STM32 Cortex®-M33 MCUs and MPUs programming manual* (PM0264), Rev 8

3 TOE 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 product to be prepared.
- 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.
- Tera Term connection preparation procedure: procedure to configure the Tera Term tool before starting to prepare the product.

3.1 Secure acceptance

Secure acceptance is the process in which the user securely receives the TOE and verifies its genuineness (the integrity and authenticity of all its components). The TOE is distributed as an MCU device.

To ensure that the MCU is not manipulated during TOE delivery, the integrator must verify that the user flash is virgin (reading 0xFF everywhere with STM32CubeProgrammer). For more details on STM32CubeProgrammer, refer to [UM2237].

During the secure acceptance process it is the responsibility of the integrator to obtain the correct software package as described in Section 3.2.2.

The integrator accepts a microcontroller part of the STM32C5xxxx devices by reading, with STM32CubeProgrammer, the DBGMCU_IDCODE register value as defined in [RM]. Supported part numbers are listed in the tables below.

Table 5. Platform identifiers for STM32C542xx

Field	Address	Halfword value	Comments
Device identifier (DEV_ID)	0x4402 4000	0x44F	-
Revision identifier (REV_ID)	0x4402 4002	0x1001	Revision Z

Table 6. Platform identifiers for STM32C562xx

Field	Address	Halfword value	Comments
Device identifier (DEV_ID)	0x4402 4000	0x44E	-
Revision identifier (REV_ID)	0x4402 4002	0x1003	Revision Y

Table 7. Platform identifiers for STM32C5A3xx

Field	Address	Halfword value	Comments
Device identifier (DEV_ID)	0x4402 4000	0x45A	-
Revision identifier (REV_ID)	0x4402 4002	0x1001	Revision Z

As part of the acceptance process the integrator also reads the product configuration information as defined in the tables below. The integrator should not accept a TOE without the correct product configuration for STM32C5xxxx devices.

Table 8. Product configuration for STM32C542xx

Field	Address	Bitfields	Comments
Product configuration	0x4002 2428	4, 7 reset	AES, HASH enable

Table 9. Product configuration for STM32C562xx

Field	Address	Bitfields	Comments
Product configuration	0x4002 2428	4, 7 reset	AES, HASH enable

Table 10. Product configuration for STM32C5A3xx

Field	Address	Bitfields	Comments
Product configuration	0x4002 2428	1, 4, 5, 6, 7 reset	SAES, AES, PKA, CCB, HASH enable

3.2 Secure installation and preparation (AGD_PRE.1.2C)

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

3.2.1 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:

- Configuring the platform non-volatile memory (flash)
- Debugging when the protections are disabled.
- Optionally interacting with the board via a UART console (see Terminal emulator)

The ST-LINK firmware programmed on the development board must be the V3J15M6 version or newer.

3.2.2 Software setup procedure

This section lists the minimum requirements for the developer to set up the SDK on a Windows® 11 host, run the sample scenario and customize applications delivered in the corresponding STM32Cube Software Package for STM32C5 series.

STM32Cube firmware package

The STM32Cube firmware package includes necessary drivers, middleware, template projects, example applications and compiler toolchain for software development on the target STM32 microcontroller.

Download and install the latest STM32Cube Software Package for STM32C5 series from:

- https://www.st.com/en/embedded-software/stm32cubec5.html

Software tools for programming STM32 microcontrollers

STM32CubeProgrammer (STM32CubeProg) is an all-in-one multi-OS software tool for programming STM32 microcontrollers. It provides an easy-to-use and efficient environment for reading, writing, and verifying device memory through both the debug interface (JTAG and SWD) and the bootloader interface (UART and USB).

STM32CubeProgrammer offers a wide range of features to program STM32 microcontroller internal memories (such as flash memory, RAM, and option bytes) as well as external memories. STM32CubeProgrammer also allows option programming and upload, programming content verification, and microcontroller programming automation through scripting.

STM32CubeProgrammer is delivered in GUI (graphical user interface) and CLI (command-line interface) versions.

For more details about STM32CubeProgrammer, refer to [UM2237].

Note: Always use the latest release of STM32CubeProgrammer software package.

Terminal emulator

A terminal emulator software is needed to control the correctness of the installation if the example described in Appendix A is used. The example in this document is based on Tera Term, an open-source free software terminal emulator that can be downloaded from the https://osdn.net/projects/ttssh2/ webpage. Any other similar tool can be used instead.

3.3 Secure installation

The product preparation for microcontrollers that are part of the STM32C5xxxx device is done in two steps, to get a complete installation with fully activated security configuration. All the steps must be performed successfully to have the product in its certified configuration.

Step 1: Software programming

- Program the non-platform required secure boot firmware and root parameters in the HDP securable memory at address according to the integrator mapping.
- All the other parts of the application code not belonging to the secure boot firmware are programmed outside of the HDP securable memory area at the destination address according to the integrator mapping.

Step 2: static security protections programming

- Set the unique boot entry:
  - FLASH_OPTSR.SWAP_BANK=0 (no swap)
  - Program FLASH_BOOTR.BOOTADD = the boot memory address modulo 256 bytes
  - Set FLASH_BOOTR.BOOT_LOCK = 0xB4 (locked).
- Program the platform's hide protection area (HDP) located in the secure area, so that the non-platform required secure firmware and root parameters matches the hide protection area:
  - Program FLASH_HDPxR.HDPx_STRT[6:0] = first page of the hide protection area.
  - Program FLASH_HDPxR.HDPx_END[6:0] = last page of the hide protection area.
    - With x value can be 1 or/and 2
- Program a write-protected memory area (WRP) on the immutable area. This area related to the immutable RoT must be included in the HDP area.
  - Program FLASH_WRPxR.WRPSGx[31:0] = block based for WRP area in bank x.
    - With x value set to 1 or 2, depending on integrator mapping
- Optionally, program 128 bit OEM Key
  - Program FLASH_OEMKEYRx.OEMKEY[31:0] registers with integrator's choice value.
    - With x value set from 1 to 4 (the 4 registers must be programmed)

Note: For SFR "field return of platform", OEM Key must be programmed

- Program the Read Protection (RDP)
  - Program FLASH_OPTSR.RDP_LEVEL = 0x72 (RDP level2)

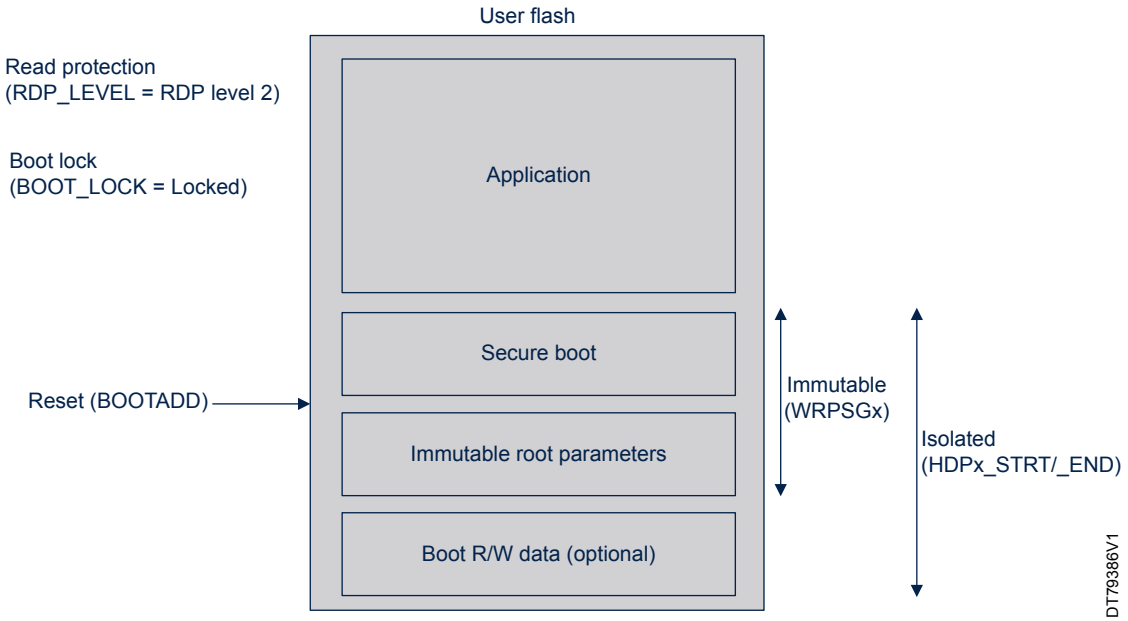
A detailed programming procedure example is provided in the Appendix A, based on secure boot code example available in the STM32Cube Software Package for STM32C5 series defined in Section 3.2.2.

Certified configuration

After achieving the two secure installation steps, the platform is in its secured configuration.

Figure 1 illustrates the configuration of the secure boot firmware and the root parameters in bank1.

Figure 1. TOE certified configuration



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 for:

- Receiving the TOE
- Performing the preparative procedures as described in TOE preparative procedures
- Integrating the TOE into a final product

The integrator has full access to the TOE security features, as the STM32C5xxxx MCU devices are delivered in RDP0 state without any features activated for NVM protection. The integrator also has full access to the tools needed to program the TOE.

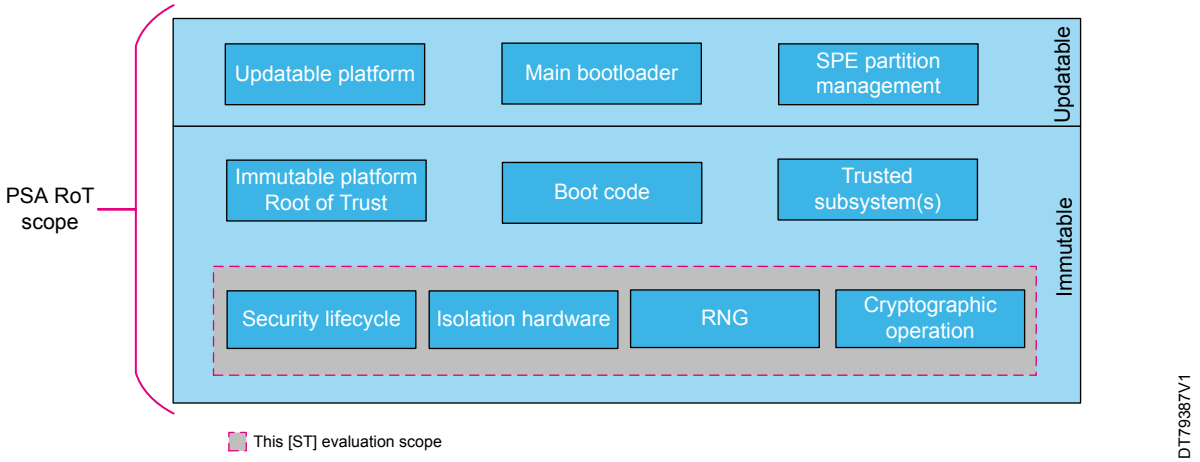
4.2 Operational guidance for the integrator roles

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 Section 4.2.2.

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 outside the TOE without compromising the security of the TOE as shown in Figure 2.

Figure 2. TOE perimeter



Follow the procedures described in Section 3.1 to check if the TOE is acceptable for the secure configuration. The secure configuration of the TOE may 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.

Product firmware

The integrator must first load the product firmware prior to setting up the security configuration of the final product.

The integrator must split the product firmware into two separated areas:

- Securable memory area that includes an HDP area in which the integrator can locate its protected assets. This HDP section contains an immutable area safeguarded by write protection (WRP). This area might contain, for example, a first-stage secure boot code with root parameters composed of private and immutable personalization data or cryptographic elements depending on the integrator's product security requirements.
- A non-HDP memory area where the integrator can locate the next boot stages plus the application firmware.

Recommended security protection programming is described in Section 3.2.2.

RDP regression

In its certified configuration the platform is deployed in RDP level 2.

The RDP value in FLASH_OPTSR indicates the protection level of the product, as explained in section 3.7.1 of [RM].

The OEMKEY must be provisioned before setting RDP level 2 to allow regression to RDPlevel 0.

For SFR field return of platform, the user should not store any data in OTP as this data is not erased during RDP regression.

HDP securable memory area

The platform is certified with the HDP securable memory enabled. The integrator shall link the immutable Root of Trust of the product firmware inside the securable memory area and configure the associated options bytes as follows:

- FLASH_HDPxR.HDPx_STRT: index of the first page of the HDPx securable memory area.
- FLASH_HDPxR.HDPx_END: index of the last page of the HDPx securable memory area.
  - With x value can be 1 or/and 2, depending on integrator mapping

Additionally, for STM32C5xxxx devices, the product firmware shall activate the hide protection just before jumping out of the securable memory area by setting the INCR_HDPL to a value different than 0xB4 in SBS_HDPLCR register. Refer to section 3.2.1 in [RM] for more details on HDP.

HDP securable memory area

The platform is certified with the HDP securable memory enabled. The integrator shall link the immutable Root of Trust of the product firmware inside the securable memory area and configure the associated options bytes as follows:

- FLASH_HDPxR.HDPx_STRT: index of the first page of the HDPx securable memory area.
- FLASH_HDPxR.HDPx_END: index of the last page of the HDPx securable memory area.
  - With x value can be 1 or/and 2, depending on integrator mapping

Additionally, for STM32C5xxxx devices, the product firmware shall activate the hide protection just before jumping out of the securable memory area by setting the INCR_HDPL to a value different than 0xB4 in SBS_HDPLCR register. Refer to section 3.2.1 in [RM] for more details on HDP.

Boot configuration

In its certified configuration the platform starts by executing code from the address defined by option byte FLASH_BOOTR.BOOTADD, because the option byte BOOT_LOCK is set. This address must be located in a write protected and HDP protected area, as described in Section 3.2.2.

Fault injection attacks countermeasures

The platform claims resistance against physical attacker, to fully leverage it, the integrator must enable the following protections.

The integrator must implement the following software countermeasures within its application when performing sensitive operations (including cryptographic ones):

- Redundancy:
  - For example:
    - Program sensitive registers twice and read back the result to verify that it is the expected one.
    - Perform sensitive operation twice and verify that results are equal.
    - Implement the inverse of the cryptographic operation: encrypt then decrypt or decrypt then encrypt.
  - In case of verification error:
    - Implement security response, for example platform reset.
- Random timing jitter:
  - For example, apply random loop (using RNG peripheral) before 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 security response, for example platform reset.

In addition to the above protection, the integrator should implement anti-tampering protection:

- Enable cryptographic internal tampers detection (TAMP peripheral):
  - When a tamper flag raises (tampering detected), the integrator must implement security response, for example platform reset. Refer to the section 3.3.3 “Tamper detection and response” in [RM].

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] for details.

Hardware-accelerated cryptography

In its configuration with additional cryptography (see Section 3.1) the platform 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 cryptographic 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, and signature using protected modular exponentiation (MODE=0x3)
  - ECC, protected scalar multiplication (MODE=0x20) and signature (MODE=0x24)

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

As any software dealing with sensitive data, the software driving hardware cryptography accelerators should follow the guidelines described in section Fault injection attacks countermeasures (for example, timing randomization, control flow…). The software handling crypto operations should comply with the recommendations provided in the corresponding specification, as indicated in table 15 of the [ST].

Cryptographic key storage

The integrator must use the SAES peripheral to protect the confidentiality of AES 128 or 256-bit keys in its KeyStore, using the key wrapping/ unwrapping method described in the subsection “SAES operation with wrapped keys” of the SAES section in [RM].

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

The integrator must use the CCB to protect the confidentiality of the RSA and ECC private Keys in its keystore using the special coupling and chaining operations method as described in the subsection “CCB coupling and chaining operations” of the CCB in [RM].

Note: The wrapping/unwrapping functions are not usable while an active tamper event is not cleared (see next section).

Anti tamper peripheral

The platform resets with internal and external tamper sources deactivated in the TAMP peripheral. The integrator must configure TAMP peripheral with anti-tamper methods available (internal tampers) in the device when managing following functions:

- Non-volatile derived hardware unique key (DHUK) usage.
- Crypto functions in hardware engines (SAES, AES, HASH, PKA, and CCB).

For TAMP peripheral configuration, refer to its dedicated section in [RM].

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 Section 4.2.1, the integrator interacts with the TOE interfaces described in this section:

- Physical chip interface
- Flash option bytes, SBS registers and flash memory interface
- JTAG/SWD debug interface
- True random number generation
- Cryptographic functions interface
- TAMP peripheral
- DBGMCU
- RCC peripheral
- CPU

Physical chip interface

After providing power supply and clocks and de-asserting the reset signal (refer to RCC and PWR sections in [RM] for details on power-on and reset procedures), the platform firmware starts at a unique boot address stored in FLASH_BOOTR.BOOTADD.

Method of use:

- Activate power supplies and clocks of the platform.
- Reset the device.

Parameters:

- Not applicable

Actions:

- The Cortex-M33 processor first executes the code located in an internal flash area HDP and write protected.

Errors:

- The platform firmware does not start properly if its vector table of the integrator firmware is not located at the boot address in the user flash. The Cortex-M33 goes to error states if its vector table points toward a non-secure area (see section 2.2.2 in [RM] for details).

Flash option bytes

The integrator in the secure boot shall read the flash option bytes to confirm that the platform is set in the certified configuration defined in Section 3.2.2.

Method of use:

- The Cortex-M33 processor performs read access in the flash registers.

Parameters:

- The list of flash option bytes and their configuration are presented in *Step 2: static security protection programming* of Section 3.3.

Actions:

- Verify the value of each enumerated configuration byte.

Errors:

- The actions fail if the content read from option bytes differs from the programmed values. In that case, the integrator firmware must cancel the boot operation, by resetting the platform or looping infinitely.

SBS registers

When the secure boot firmware ends its execution, it must jump to a code located in a non-HDP area. The jump procedure must activate the TOE HDP area protection, making it inaccessible by any read or write operation until the next reset.

Method of use:

- Set INCR_HDPL to a value different than 0xB4 in SBS_HDPLCR register (refer to [RM] in section 12.3)

Parameters:

- In SBS_HDPLCR the INCR_HDPL bits

Actions:

- Set INCR_HDPL to a value different than 0xB4
- Verify TOE HDP activation by reading anywhere in the TOE HDP area a 32-bit word that was programmed with a non-zero value. The read value must be zero.

Errors:

- If the verification value is not zero the TOE HDP area protection is not properly activated. In that case, the product firmware must be restarted.

Flash memory

To verify the platform identity, the integrator has to verify the platform configuration by reading at the address 0x4002 2428.

STM32C542xx and STM32C562xx

Method of use:

- The Cortex-M33 processor reads at address 0x4002 2428 and checks that the bits 4 and 7 are reset. These bits indicate whether the AES and HASH IP are enabled or not.

Parameters:

- Address of the platform configuration

Actions:

- The Cortex-M33 processor reads at address 0x4002 2428.
- Verify that the bits 4 and 7 are reset.

Errors:

- If one of the bits 4 and 7 are not reset, then the platform configuration is not correct, and the integrator must not accept the TOE.

STM32C5A3xx

Method of use:

- The Cortex-M33 processor reads at address 0x4002 2428 and checks that the bits 1, 4, 5, 6 and 7 are reset. These bits indicate whether the SAES, AES, PKA, CCB, and HASH IP are enabled or not.

Parameters:

- Address of the platform configuration

Actions:

- The Cortex-M33 processor reads at address 0x4002 2428.
- Verify that the bits 1, 4, 5, 6 and 7 are reset.

Errors:

- If one of the bits 1, 4, 5, 6 and 7 are not reset, then the platform configuration is not correct, and the integrator must not accept the TOE.

JTAG/SWD debug interface

Standard JTAG with SWD interface allows debugging of the TOE and integrator application. It is used according to [IEEE1149] and [IHI0031]. When RDP is level 2 (certified configuration), all debug features are disabled. If the OEMKEY was provisioned, the JTAG/SWD interface remains enabled under reset to interface with DBGMCU_DBG_AUTH_HOST and DBGMCU_DBG_AUTH_DEVICE registers to obtain device identification and provide the OEMKEY to request the RDP regression to level 0.

Method of use:

- JTAG/SWD physical link to transport OEM password key value

Parameters:

- 128-bit OEM password key stored in FLASH_OEMKEYRx registers. Refer to subsection *OEM lock activation* in section 6.5.9 of [RM].
- Device specific authentication ID stored in DBGMCU_DBG_AUTH_DEVICE. Refer to section 49.12 of [RM].
- DBGMCU_DBG_AUTH_HOST: The OEMKEY must be written to this register to permit RDP regression. Refer to section 49.12 of [RM].

Actions:

- Read the device specific authentication ID in DBGMCU_DBG_AUTH_DEVICE
- Inject the OEMKEY in DBGMCU_DBG_AUTH_HOST through JTAG/SWD under reset.
- If the received value is identical to the programmed OEMKEY value, the hardware starts a regression to RDP level 0.

Errors:

- RDP level remains set to level 2, when provided OEMKEY value was wrong.

True random number generation

When the platform is not executing the application can use the RNG 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] in section 26.7: RNG registers.

Actions:

- Refer to [RM] in section 26.3.4: RNG initialization.
- Refer to [RM] in section 26.3.5: RNG operation.
- Refer to [RM] in section 26.3.8: RNG low-power use.
- Refer to [RM] in section 26.4: RNG interrupts.

Errors:

- Refer to [RM] in section 26.3.7: Error management.

Cryptographic functions interface

The platform provides to the application 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, PKA and CCB through a bank of memory mapped registers, and the embedded RAM in the PKA peripheral.

Parameters:

- Refer to [RM] in section 28.8: SAES registers.
- Refer to [RM] in section 30.7: PKA registers.
- Refer to [RM] in section 30.7.4: PKA RAM parameters.
- Refer to [RM] in section 25.7: CCB registers.

Actions:

- Refer to SAES [RM] in section 28.4, 28.5 and 28.6
- Refer to PKA [RM] in section 30.3 and 30.6
- Refer to CCB [RM] in section 25.4 and 25.6

Errors:

- Refer to [RM] in section 28.4.19: SAES error management.
- Refer to [RM] in section 30.3.8: PKA error management.
- Refer to [RM] in section 25.4.5: CCB error.

TAMP peripheral

The TAMP peripheral is used to protect sensitive data from external attacks

Method of use:

- The integrator needs to activate and configure the cryptographic internal tampers detection in the TAMP peripheral when managing following functions:
  - Non-volatile derived hardware unique key (DHUK) usage.
  - Crypto functions in hardware engines (SAES, AES, HASH, PKA, and CCB).

Parameters:

- Refer to [RM] in section 40.6: TAMP registers

Actions:

- The tamper detection can be configured for the following purposes:
  - Erase the backup registers and other device secrets stored in SRAMs or peripherals listed in [RM] in *Table 400: TAMP interconnection*.
  - Block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in [RM] *Table 400: TAMP interconnection*.
  - Generate an interrupt, capable to wake up from low-power modes.
  - Generate a hardware trigger for the low-power timers, or an RTC timestamp event.
    - Refer to TAMP [RM] in section 40.3.9: Tamper detection configuration and initialization

Note: For sensitive registers, fault injection attacks countermeasures must be applied.

Errors:

- To avoid false detection, it is possible to configure each tamper source in potential mode, so that the erase is not launched by hardware on tamper detection. The erase can then be launched by software after software checks. Refer to section *Tamper detection* in section 40.3.7 to [RM].

DBGMCU

When RDP is level 2 (certified configuration), the integrator needs to interface with DBGMCU_DBG_AUTH_HOST, and DBGMCU_DBG_AUTH_DEVICE registers to obtain device identification and provide OEMKEY password to request RDP regression.

The device and revision identifier (REV_ID, DEV_ID) of the platform are stored in DBGMCU registers (DBGMCU_IDCODE).

Method of use:

- The integrator needs to interact with DBGMCU registers to inject the OEMKEY and to retrieve the device and revision identifier of the platform (REV_ID, DEV_ID).

Parameters:

- Device specific authentication ID stored in DBGMCU_DBG_AUTH_DEVICE. Refer to section 49.12 of [RM].
- DBGMCU_DBG_AUTH_HOST: The OEMKEY must be written to this register to permit RDP regression. Refer to subsection *OEM RDP regression mechanism* in section 6.5.9 of [RM].
- REV_ID and DEV_ID are stored in DBGMCU_IDCODE register.

Actions:

- Read the device specific authentication ID in DBGMCU_DBG_AUTH_DEVICE. Inject the OEMKEY in DBGMCU_DBG_AUTH_HOST through JTAG/SWD under reset. Release reset. If the received value is identical to the programmed OEMKEY value, the hardware starts the regression. Refer to subsection *OEM RDP regression mechanism* in section 6.5.9 of [RM].
- The integrator accepts a microcontroller part of the STM32C5xxxx devices by reading, with STM32CubeProgrammer, the DBGMCU_IDCODE register value as defined in [RM] below:

Errors:

- RDP level remains set to level 2, if the wrong password is provided or the sequence is not respected.
- If the DBGMCU_IDCODE value does not correspond to platform references defined in Section 3.1, the integrator should not accept this microcontroller.

Reset and clock control (RCC) peripheral

The application can use the RCC to manage the different kinds of reset and generates all clocks for the bus and peripherals. At startup the HSIS is used as system clock source after startup from reset, configured at 48 MHz. The application can use different clock sources to drive the system clock and configure additional clock sources.

Method of use:

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

Parameters:

- Refer to [RM] in section 9.8: RCC registers.

Actions:

- Refer to [RM] in section 9.3: RCC reset functional description
- Refer to [RM] in section 9.4: RCC clock functional description
- Refer to [RM] in section 9.6: RCC in low-power modes
- Refer to [RM] in section 9.7: RCC interrupts.

Errors:

- If a clock source that is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *RCC clock control register (RCC_CR)* indicate which clocks are ready and which clock is currently used as a system clock. Refer to [RM] in section 9.4: RCC clock functional description.

CPU

The CPU executes application code by fetching instructions from memory, processing data, and storing results back into memory. It uses system registers to manage control and configuration tasks, ensuring proper operation and interaction with peripherals.

Method of use:

- The CPU fetches instructions from memory to execute.
- The CPU reads data from memory or peripheral registers.
- The CPU writes data to memory or peripheral registers.

Parameters:

- Program counter (PC), memory address, data size, data value
- Refer to [PM0264] in section 2.1.3: Core registers
- Refer to [PM0264] in section 3.1: Instruction set summary

Actions:

- Fetch instructions from the specified memory address
- Read and write data from the specified memory address or peripheral register
- Refer to [PM0264] in section 2.3: Memory model

Errors:

- Faults can occur on instruction fetches, instruction execution, and data accesses. Refer to [PM0264] in section 2.6: Fault handling

4.2.3 Security-relevant events (AGD_OPE.1.4C)

Once configured according to Section 3.2, the platform detects any unauthorized access and any unexpected configuration:

- Erroneous values found in the option bytes of the security configuration at boot time:
  - Cancel the boot sequence by resetting or blocking the platform (integrator-defined implementation)
- Illegal access in the HDP securable memory area:
  - Read returns zero.
  - Write causes a bus error. The integrator firmware can implement an error handler.
- Attempt to reprogram the security option bytes in RDP level 2:
  - Status FLASH_SR.OPTCHANGEERR raised with a possible interrupt toward an user implementation defined error handler.
- Closed JTAG/SWD access violation:
  - The connection request is not transmitted to the access port and debug port. The request is ignored.
  - generation of internal tamper (tamp_itamp6 input)
- Wrong OEMKEY password key injection for RDP level 2 to level 0 regression:
  - The detection ignores the regression request and restarts in RDP level 2 at the next boot.
- 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 interrupt toward a user implementation-defined error handler.
- Detection of attacks to RNG, SAES, CCB and PKA by an attacker with physical access
  - Peripheral goes to locked mode + generation of internal tamper (tamp_itamp9 input), disabled by default.

4.2.4 Security measures (AGD_OPE.1.6C)

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

To achieve TRUSTED_INTEGRATOR and LIFECYCLE, the following measures must be taken:

- Verify the genuineness of the TOE as described in Section 3.1.
  - Follow all guidelines described and referenced in Section 3.2.
  - Follow all guidelines described in Section 4.2.1 and Section 4.2.2 regarding the implementation of the non-platform required firmware described in [ST] in section 1.4.5.
  - In manufacturing phase, the integrator must securely provision the TOE immutable data specific to the integrator or specific to the product as stated in the product firmware HDP securable memory area of Section 4.2.1.
  - Once the integrator finishes the production of a final user application, they must set the MCU hardware static protections as stated in Section 3.3. To protect the complete product including the user application, the final product state must be RDP2.

To achieve UNIQUE_ID and KEY_MANAGEMENT, the following measures must be taken:

- The integrator must protect the integrity of the immutable Root of Trust of the non-platform required firmware ([ST] in section 1.4.5) until it is programmed and properly protected inside the securable memory area for each device.
- The persons responsible for the application of the procedures described in Section 3, and the persons involved in the delivery and protection of the product must have the required skills and must be aware of the security issues.
- In the case that any part of the preparative procedures of the platform or a party other than the integrator integrates any part of the preparative procedures of the integrated platform, the integrator must guarantee sufficient guidance is provided to this party.
- The integrator can define any type of immutable data unique per product that might allow product identification after its field deployment. It is recommended that the integrator puts in place a data management process, possibly based on database solution, ensuring new unique data generation.
- The integrator must protect the integrity of all the data protected in the HDP securable memory area until they are provisioned and properly protected inside the platform of each device. Moreover, the integrator must protect the confidentiality of the secret data that are stored in the HDP securable memory area.
- Once immutable data of the non-platform required firmware are generated for a new product, the integrator must program them in the defined format at the correct location corresponding to the securable memory area and must protect them as described in Section 3.3.
- The integrator must protect the integrity and confidentiality of the cryptographic secret keys encapsulated in the root parameters as defined in Section 4.2.1.

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.

Normal boot mode

After reset, the platform is forced to boot into the non-platform required secure firmware. The security configuration, dictated by Section 3.2, unauthorizes all the other alternative boot modes.

Regression mode

The integrator owning the OEMKEY to unlock RDP2 injects this value over the JTAG interface while maintaining the platform under reset, as explained in subsection *OEM RDP regression mechanism* of [RM] in section 6.5.9. If OEMKEY value is wrong, the regression request aborts and any access to the flash is blocked until the next power-on-reset.

Appendix A STM32C5xxxx devices

A.1 Getting started with secure boot example

The following steps describe the operations to configure, build, and program static security protections and a secure boot example code with its demonstration application based on the STM32Cube Software Package for STM32C5 series.

A.1.1 Prerequisites

Select one of the following boards, depending on selected device: NUCLEO-C5A3ZG, NUCLEO-C542RC, or NUCLEO-C562RE. In the following sections, the selected board is referred as {Board-name}.

1. Install the STM32Cube Software Package for STM32C5 series
2. Install the STM32CubeProgrammer (including STM32TrustedPackageCreator)
3. Install the preferred IDE/toolchain (STM32CubeIDE for VS Code, IAR™ EWARM or Keil® µVision)
4. Install Python 3.10 or greater
5. Install required Python modules listed in STM32Cube Software Package for STM32C5 series from the folder *rot/rot_provisioning* and run: `pip install -r requirements.txt`

A.1.2 Software configuration, build, and device programming

1. Run the provisioning script in STM32Cube Software Package for STM32C5 series, from the folder *examples/advanced/rot/oemirot_overwrite/{Board-name}/provisioning: python provisioning.py*
2. Follow script guidance prompts, and ensure to select the final RDP level: RDP_2
3. Verify that there is no error during the different steps, and installation is successful, with the final prompt: `Successful installation`

After successful installation, the device is programmed with OEMiRoT and demonstration application binaries, and the static security hardware protections (BOOTADD, BOOT_LOCK, HDP, WRP, OEMKEY, and RDP option bytes).

A.1.3 Verification of correct installation

1. Connect Tera Term console on serial port *STMicroelectronics STLink Virtual COM Port* configured at 115200 bauds, 8-bit data, 1 stop bit, no parity, no flow control.
2. Reset the platform (press reset button on board).
3. Verify that the application is alive with the message displayed on console:

Figure 3. Verification of correct installation



DT79388V1

Revision history

Table 11. **Document revision history**

Date	Version	Changes
05-Mar-2026	1	Initial release.

Contents

1 General information 2
2 Reference documents 3
3 TOE preparative procedures 4
3.1 Secure acceptance 4
3.2 Secure installation and preparation (AGD_PRE.1.2C) 5
3.2.1 Hardware setup procedure 5
3.2.2 Software setup procedure 5
3.3 Secure installation 6
4 Operational user guidance 8
4.1 User roles 8
4.2 Operational guidance for the integrator roles 8
4.2.1 User-accessible functions and privileges (AGD_OPE.1.1C) 8
4.2.2 Available interfaces and methods of use (AGD_OPE.1.2C and AGD_OPE.1.3C) 11
4.2.3 Security-relevant events (AGD_OPE.1.4C) 15
4.2.4 Security measures (AGD_OPE.1.6C) 16
4.2.5 Modes of operation (AGD_OPE.1.5C) 17
Appendix A STM32C5xxxx devices 18
A.1 Getting started with secure boot example 18
A.1.1 Prerequisites 18
A.1.2 Software configuration, build, and device programming 18
A.1.3 Verification of correct installation 18
Revision history 19
List of tables 21
List of figures 22

List of tables

Table 1. Applicable products . . . 1
Table 2. Specific acronyms . . . 2
Table 3. Peripherals in TOE per product . . . 2
Table 4. List of reference documents . . . 3
Table 5. Platform identifiers for STM32C542xx . . . 4
Table 6. Platform identifiers for STM32C562xx . . . 4
Table 7. Platform identifiers for STM32C5A3xx . . . 4
Table 8. Product configuration for STM32C542xx . . . 4
Table 9. Product configuration for STM32C562xx . . . 5
Table 10. Product configuration for STM32C5A3xx . . . 5
Table 11. Document revision history . . . 19

List of figures

Figure 1. TOE certified configuration 7
Figure 2. TOE perimeter 8
Figure 3. Verification of correct installation 18

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.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The 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.

The 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 the 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.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

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.

© 2026 STMicroelectronics – All rights reserved