STM32WBA6xxx SESIP level 3 certification

Introduction

This document describes the preparative procedures and operational user guidance of STM32 general-purpose microcontrollers to make a secure system solution according to SESIP level 3 certification scheme.

This security guidance applies to any boards based on the devices in the table below.

Table 1. Applicable products

Reference	Products
STM32WBA6xxx	STM32WBA62xx, STM32WBA63xx, STM32WBA64xx, STM32WBA65xx

1 General information

This document applies to STM32WBA6xxx Arm®-based MCUs.

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

arm

Table 2. Specific acronyms

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

2 Reference documents

Table 3. List of reference documents

Reference	Document title and revision
[ST]	Technical note *STM32WBA6xxx SESIP Security Target* (ST0047), Rev 1
[RM]	Reference manual *STM32WBA6xxx advanced Arm®-based 32-bit MCUs* (RM0515), Rev 2
[UM2237]	User manual *STM32CubeProgrammer software description* (UM2237), Rev 24
[UM2609]	User manual *STM32CubeIDE user guide* (UM2609), Rev 9
[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)

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 memory is virgin (reading `0xFF` at any address location 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 STM32WBA6xx microcontroller by reading, with STM32CubeProgrammer, the DBGMCU_IDCODE register value as defined in [RM] and below. Supported part numbers are listed in Table 1.

Field	Address	Halfword value	Comments
Device identifier (DEV_ID)	0xE004 4000	0x4B0	STM32WBA6xxx version 1.1 (rev Z)
Revision identifier (REV_ID)	0xE004 4002	0x1001	

As part of the acceptance process, the Integrator must also read the following product configuration information.

Field	Address	Bitfields	Comments
Product configuration for STM32WBA6xxx	0x0BFA 0501	0, 2, and 7 set	SAES, AES, and PKA enabled

3.2 Secure installation and preparation (AGD_PRE.1.2C)

This section describes all the steps necessary for the 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:

- Configure the nonvolatile flash memory of the platform,
- Debug when the protections are disabled,
- And optionally interact with the board via a UART console (see following Terminal emulator)

The ST-LINK firmware programmed on the development board must be the V3J8M3 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 host, to run the sample scenario, and to customize applications delivered in the corresponding STM32Cube MCU Package.

STM32Cube firmware package

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

Download and install the latest STM32CubeWBA MCU package release for the STM32WBA6xxx devices from:

- https://github.com/STMicroelectronics/STM32CubeWBA
- Or https://www.st.com/en/embedded-software/stm32cubewba.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 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) and 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 the STM32CubeProgrammer software package.

Terminal emulator

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

3.2.3 Secure installation

The product preparation for STM32WBA6xxx is done in three steps, to ensure 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: Software build

- The required secure boot firmware and root parameters are linked to the HDP securable memory area, starting at the address 0x0C00 X000 (see **Step 3**).
- The parts of the application code not belonging to the secure boot must be linked outside the HDP securable memory area.
- A detailed building procedure is provided in the Annex, based on the secure boot code example available in the STM32Cube firmware package defined in Section 3.2.2.

Step 2: Software programming

- Copy the images generated at the previous step into the internal flash memory of the microcontroller.
  - The nonplatform required secure boot firmware and root parameters are loaded in the HDP securable memory at address 0x0C00 X000 (see **Step 3**).
  - All the other parts of the application code not belonging to the secure boot firmware are loaded outside the HDP securable memory area at the destination address according to the build parameters.

Step 3: STM32WBA6xxx static security protection programming

- Program TZEN=1 (TrustZone®), SWAP_BANK=0 (no swap)
- Set the unique boot entry:
  - Program the secure boot memory address modulo 128 bytes: SECBOOTADD0[24:0] = 0x018 X000 (secure flash memory 0x0C00 X000).
    - This address must be inside the secure HDP area, with WRP activated. X is defined by the value written in WRP1A_PSTRT or WRPA_PSTRT (see next).
  - Set BOOT_LOCK = 1.

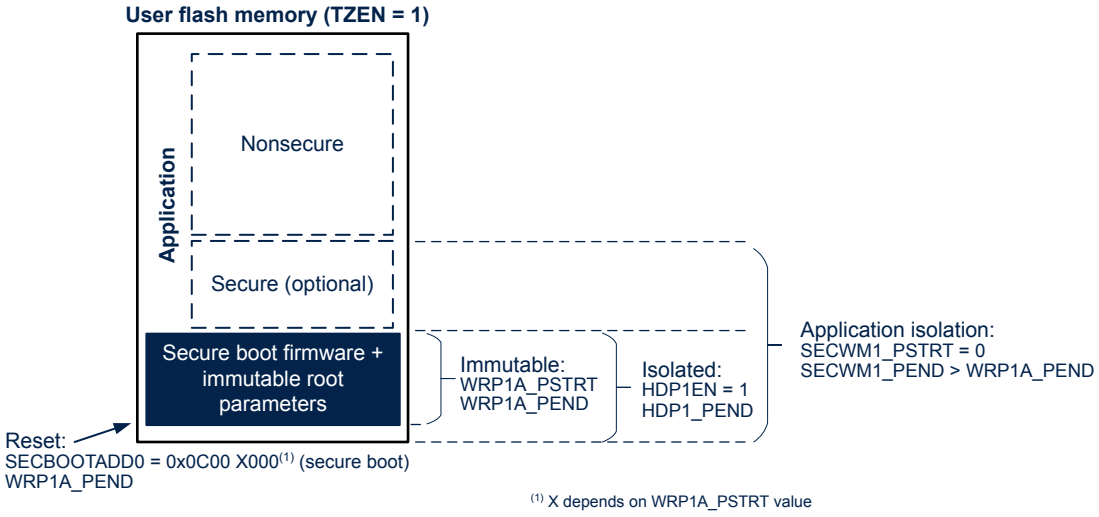
- Program the platform's hide protection area (HDP) located in the secure area:
  - Program FLASH_SECWM1R1.SECWM1_PSTRT[6:0] = 0 (internal flash memory base address)
  - Program FLASH_SECWM1R1.SECWM1_PEND[6:0] = last page of the secure area
  - Program FLASH_SECWM1R2.HDP1EN = 1
  - Program FLASH_SECWM1R2.HDP1_PEND[6:0] = last page including the secure boot firmware and root parameters programmed in **Step 2**. It must be lower than SECWM1_PEND.
- In FLASH_WRPAR, program a write-protected memory area (WRP) on the immutable area. This area must include the HDP area.
  - WRP1A_PSTRT = first page of the WRP area
  - WRP1A_PEND = the last page of the WRP area. It must be equal to HDP1_PEND.
  - UNLOCK = 0 (locked)
- Program read protection in level 2: RDP = 0xCC

A detailed programming procedure is provided in the Annex, based on the secure boot code example available in the STM32Cube firmware package defined in Section 3.2.2.

Certified configuration

After achieving the three secure installation steps, the platform is in its secured configuration, summarized in Figure 1.

Figure 1. TOE certified configuration



4 Operational user guidance

4.1 User roles

The *Integrator* user role 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, and
- Integrating the TOE into a final product.

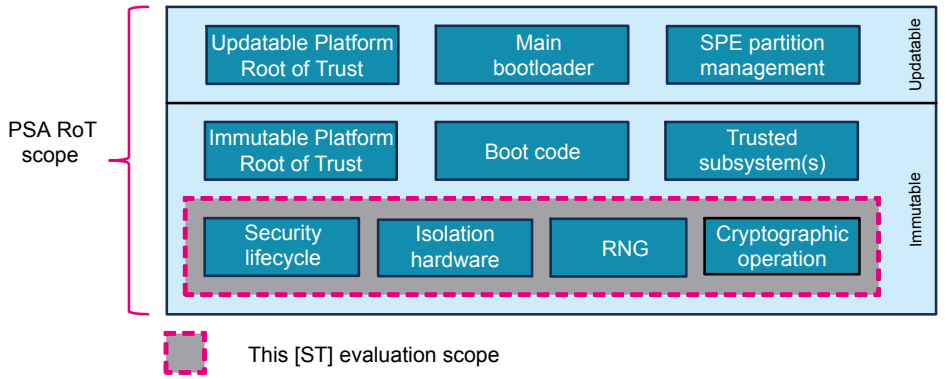
The Integrator has full access to the TOE security features, as the STM32WBA6xxx MCU devices are delivered in RDP0 state without any features activated for NVM protection. The Integrator also has full access to the tools to program the TOE.

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 can access interfaces 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: 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 also 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 before setting up the security configuration of the final product. The Integrator must split the product firmware into two separate areas:

- Securable memory area that includes an HDP area in which the Integrator can locate its protected assets. 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 3.2.3.

RDP Level

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

Note that the RDP value in FLASH_OPTR indicates the protection level of the product, as explained in Section 3.11.1 of [RM].

- RDP Level 0 if RDP[7:0] = 0xAA
- RDP Level 2 if RDP[7:0] = 0xCC
- RDP Level 1 if RDP[7:0] is not 0xAA or 0xCC

Note: The existing feature of regression to RDP Level 0 based on OEM1KEY is out of the scope of the TOE.

HDP securable memory area

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

- HDP1EN = 1
- HDP1_PEND: index of the last page of the HDP securable memory area

Additionally, the product firmware must activate the hidden protection just before jumping out of the securable memory area by setting the HDP1_ACCDIS bit in FLASH_SECHDPCR register. Refer to Section 3.6.1 of [RM] for more details on HDP.

Boot configuration

In its certified configuration, the platform starts by executing code from the secure address defined by the SECBOOTADD0 option byte, because the BOOT_LOCK option is set. This address must be in a secure, write-protected, and HDP-protected area, as described in Section 3.2.3.

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 TOE hardware cryptographic accelerators.
  - 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, a 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 *validation conditions* subsection of the RNG section of [RM] for details.

TOE hardware cryptographic accelerators

The TOE is certified with hardware-accelerated cryptography that is protected against side-channel, 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 [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.
- Increase the AES key size to 256-bit, or limit the number of AES computations using the same key to 8 million.

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 key storage

The Integrator can 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 *SAES operation with wrapped keys* subsection of the SAES section of [RM].

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

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

Antitamper 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 derived hardware unique key (DHUK) usage.
- Volatile key storage in backup register (TAMP_BKPxR) or in embedded SRAM2.
- Battery-powered storage in 2-Kbyte backup SRAM. Size depends on the configuration bit.
- Cryptographic functions in hardware engines (SAES, AES, HASH, PKA, OTFDEC).

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

When a tamper flag raises (tampering detected), the Integrator must implement a security response, for example, a platform reset. Refer to Section 3.7.3 *Tamper detection and response* of [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: User-accessible functions and privileges (AGD_OPE.1.1C), the Integrator interacts with the TOE interfaces described in this section:

- Physical chip interface
- Flash option bytes and flash registers
- JTAG/SWD debug interface
- True random number generation
- Cryptographic functions interface

Physical chip interface

After providing the power supply and clocks and deasserting the reset signal (Refer to RCC and PWR sections of [RM] for details on power-on and reset procedures), the platform firmware starts at a unique boot hardcoded address.

Method of use:

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

Parameters:

- Not applicable

Actions:

- The Cortex®-M33 processor first executes in secure mode the code located in an internal flash memory area which is HDP and write-protected.

Errors:

- The platform firmware does not start properly if the expected vector table of the Integrator firmware is not located at the boot address in the user flash memory. The Cortex®-M33 goes to error states if its vector table points toward a nonsecure area (see Section 2.3.2 of [RM] for details).

Flash option bytes

The Integrator in the secure boot must read the flash option bytes to confirm that the platform is set in the certified configuration defined in Section 3.2.3.

Method of use:

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

Parameters:

- FLASH_OPTR.TZEN = 1
- FLASH_OPTR.RDP[7:0] = 0xCC (RDP2)
- FLASH_OPTR.SWAP_BANK = 0 (no swap)
- In FLASH_SECBOOTADD0R:
  - BOOT_LOCK = 1
  - SECBOOTADD0[24:0] as defined in Section 3.2.3
- In FLASH_SECWM1R1:
  - SECWM1_PSTRT[6:0] = 0
  - SECWM1_PEND[6:0] as defined in Section 3.2.3
- In FLASH_ SECWM1R2:
  - HDP1EN = 1 (HDP1 enabled)
  - HDP1_PEND[7:0] as defined in Section 3.2.3
- In FLASH_WRP1AR:
  - WRP1A_PSTRT[6:0] and WRP1A_PEND[6:0] as defined in Section 3.2.3

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 abort the boot operation, by resetting the platform or looping infinitely.

Flash 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 HDP1_ACCDIS bit in the FLASH_SECHDPCR register (refer to [RM] Section 7.9)

Parameters:

- In FLASH_SECHDPCR, the HDP1_ACCDIS bit

Actions:

- Set the HDP1_ACCDIS bit.
- Verify TOE HDP activation by reading anywhere in the TOE HDP area a 32-bit word that is programmed with a nonzero value. The returned 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.

JTAG/SWD debug interface

The standard JTAG/SWD interface allows debugging of the TOE and the Integrator application. It is used according to [IEEE1149] and [IHI0031]. When RDP is Level 2 (certified configuration), all debug features are disabled. If the OEM2KEY is provisioned, the JTAG/SWD interface remains enabled on reset to inject the OEM2KEY as part of the regression request to RDP level 1.

Method of use:

- JTAG/SWD physical link to transport OEM2KEY value

Parameters:

- OEM2KEY stored in FLASH_OEM2KEYRx registers. Refer to *OEM2 RDP lock mechanism* subsection in Section 7.6.2 of [RM].

Actions:

- Inject the OEM2 password through JTAG/SWD under reset.
- If the received value is identical to the programmed OEM2KEY value, the hardware starts a regression to RDP level 1.

Errors:

- RDP level remains set to level 2 when provided OEM2KEY value was wrong.

True random number generation

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

Method of use:

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

Parameters:

- Refer to [RM] Section 25.7 RNG registers.

Actions:

- Refer to [RM] Section 25.3.4 RNG initialization.
- Refer to [RM] Section 25.3.5 RNG operation.
- Refer to [RM] Section 25.3.8 RNG low-power use.
- Refer to [RM] Section 25.4 RNG interrupts.

Errors:

- Refer to [RM] Section 25.3.7 Error management.

Cryptographic functions interface

The platform provides the application with a set of hardware peripheral 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 PKA peripheral.

Parameters:

- Refer to [RM] Section 27.9 SAES registers.
- Refer to [RM] Section 29.8 PKA registers.
- Refer to [RM] Section 29.4 PKA RAM parameters.

Actions:

- Refer to SAES [RM] Sections 27.4, 27.5, and 27.6.
- Refer to PKA [RM] Sections 29.3 and 29.6.

Errors:

- Refer to [RM] Section 27.4.19 SAES error management.
- Refer to [RM] Section 29.3.7 PKA error management.

4.2.3 Security-relevant events (AGD_OPE.1.4C)

Once configured according to Section 3.2: Secure installation and preparation (AGD_PRE.1.2C), 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.WRPERR or FLASH_SR.OPTWERR raised with a possible interrupt to a 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.
- Wrong OEM2 password injection for RDP level 2 to level 1 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 to a user implementation-defined error handler.
- Detection of attacks to RNG, SAES, and PKA by an attacker with physical access:
  - The peripheral goes to locked mode and an internal tamper is generated (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] 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: Secure acceptance.
- Follow all guidelines described and referenced in Section 3.2: Secure installation and 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) regarding the implementation of the nonplatform required firmware described in [ST] Section 1.4.5.
- 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 the product firmware HDP securable memory area of Section 4.2.1: User-accessible functions and privileges (AGD_OPE.1.1C).
- Once the Integrator finishes the production of a final user application, they must set the MCU hardware static protections as stated in Section 3.2.3: Secure installation. 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 nonplatform required firmware ([ST] 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: TOE preparative procedures, 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 a 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 nonplatform 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 3.2.3: Secure installation.
- 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 nonplatform required secure firmware. The security configuration, dictated by Section 3.2: Secure installation and preparation (AGD_PRE.1.2C), unauthorizes all the other alternative boot modes.

Regression mode

The Integrator owning the OEM2KEY to unlock RDP2 injects this value over the JTAG interface while maintaining the platform under reset, as explained in the *OEM2 RDP lock mechanism* subsection of [RM] Section 7.6.2. If the OEM2KEY value is wrong, the regression request aborts and any access to the flash memory is blocked until the next power-on-reset.

5 Annex

The following steps describe the operation to build a secure boot example code with its demonstration application and its loader application based on the STM32CubeWBA MCU Package for STM32WBA6xxx under Windows®. For more details on STM32CubeIDE, refer to [UM2609].

- Install the package in a folder.
- Execute `.\Projects\STM32WBA65I-DK1\ROT_Provisioning\OEMiROT\regression.sh` to initialize the device.
- Execute `.\Projects\STM32WBA65I-DK1\ROT_Provisioning\OEMiROT\provisioning.sh` to generate and program the binaries into the user flash memory and to configure the static security protections. During the provisioning process:
  - Select RDP level 2.
  - Generate the binaries as follows:
    - Open the `.\Projects\STM32WBA65I-DK1\Applications\ROT\OEMiROT_Boot\STM32CubeIDE\` folder.
    - Double-click the `.project` file to open automatically STM32CubeIDE to import and setup of the secure boot example.
    - Open the `.\Projects\STM32WBA65I-DK\Applications\ROT\OEMiROT_Appli_TrustZone\STM32CubeIDE\` folder.
    - Double-click the `.project` file to import automatically and set up the demonstration application in STM32CubeIDE.
    - Select `Menu→Project→Build All`.
    - Wait for the expected message: `Build Finished. 0 errors, 0 warnings Verification of correct installation.`
- Connect Tera Term console on serial port `STMicroelectronics STLink Virtual COM Port` configured at 115200 bauds, 1 start bit, 1 stop bit, no parity.
- Reset the platform.
- Verify the application is alive with the message:

```
============================================================
=          (C) COPYRIGHT 2024 STMicroelectronics           =
=                                                          =
============================================================
```

Revision history

Table 4. **Document revision history**

Date	Revision	Changes
23-Jul-2025	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.

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.