Overview of Secure Boot and Secure Firmware Update solution on Arm® TrustZone® STM32L5 Series microcontrollers

Introduction

This application note describes how to get a Secure Boot and Secure Firmware Update solution on Arm® TrustZone® STM32 microcontrollers based on the Arm® Cortex®-M33 processor. It also provides a top-level comparison of this solution versus the X-CUBE-SBSFU solution, which applies to non-TrustZone® STM32 microcontrollers based on the Arm® Cortex®-M0, Cortex®-M3, Cortex®-M4, or Cortex®-M7 processors. It provides as well top-level integration guidelines for the Secure Boot and Secure Firmware Update solution.

For Arm® TrustZone® STM32 microcontrollers, a Secure Boot and Secure Firmware Update solution is provided in the corresponding STM32Cube MCU Package. Contrary to the solution proposed in the X-CUBE-SBSFU STM32Cube Expansion Package, it is based on open-source TF-M (Trusted Firmware for Arm® Cortex®-M) reference implementation.

This application note applies to all TrustZone® STM32 microcontrollers. However, in this document, only the STM32L5 Series is considered as an example.

To get more information about the open-source TF-M reference implementation, refer to [TF-M].

1 General information

Throughout this application note, the terminology *X-CUBE-SBSFU* refers to the Secure Boot and Secure Firmware Update solution available in the X-CUBE-SBSFU STM32Cube Expansion Package, whereas the terminology *SBSFU* refers to the Secure Boot and Secure Firmware Update solution available in the STM32Cube MCU Package (STM32CubeL5 is used as an example).

Table 1 presents the definition of acronyms that are relevant for a better understanding of this document.

Table 1. List of acronyms

Acronym	Definition
AEAD	Authenticated encryption with associated data.
AES	Advanced encryption standard.
CBC	AES cipher block chaining.
CTR	AES counter mode.
EAT	Entity attestation token.
ECDSA	Elliptic curve digital signature algorithm.
GCM	AES Galois/counter mode.
HDP	Hide protection.
HUK	Hardware unique key.
ITS	Internal trusted storage.
KMS	Key management services.
MAC	Message authentication code.
MPU	Memory protection unit.
OEM	Original equipment manufacturer.
OTFDEC	On-the-fly decryption.
PKCS	Public-key cryptography standard.
PSA	Platform security architecture. Framework for securing devices.
RDP	Read protection.
RoT	Root of Trust.
RSA	Rivest–Shamir–Adleman algorithm.
SBSFU	Secure Boot and Secure Firmware Update.
SESIP	Security evaluation standard for IoT platforms.
SST	Secure storage.
TBSA-M	Trusted base system architecture for Arm® Cortex®-M.
TF-M	Trusted Firmware for M-class Arm® processors. TF-M provides a reference implementation of secure world software for Armv8-M.
TZ	TrustZone®.
WRP	Write protection.

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

2 References

The resources presented in Table 2 and Table 3 below are public and available either on STMicroelectronics web site at *www.st.com* or on third-parties websites.

Table 2. Document references

Reference	Document
[AN5156]	Application note(1): *Introduction to STM32 microcontrollers security*.
[UM2262]	User manual(1): *Getting started with the X-CUBE-SBSFU STM32Cube Expansion Package*.
[UM2671]	User manual(1): *Getting started with STM32CubeL5 TFM application*.
[TFM_USER_GUIDE]	TF-M user guide for v1.0-RC2: ci.trustedfirmware.org/job/tf-m-build-test-nightly/lastSuccessfulBuild/artifact/build-docs/tf-m_documents/install/doc/user_guide/html/index.html(2)
[PSA_API]	PSA developer APIs: developer.arm.com/architectures/security-architectures/platform-security-architecture#implement(2)

1. *Available on www.st.com. Contact STMicroelectronics when more information is needed.*
2. *This URL belongs to a third party. It is active at document publication, however STMicroelectronics shall not be liable for any change, move or inactivation of the URL or the referenced material.*

Table 3. Open-source software resources

Reference	Open-source software resource
[TF-M]	TF-M (Trusted Firmware-M) Arm Limited driven open-source software framework: www.trustedfirmware.org/(1)
[MCUboot]	MCUboot open-source software: juullabs-oss.github.io/mcuboot/(1)
[mbed-crypto]	mbed-crypto open-source software: github.com/ARMmbed/mbed-crypto(1)
[PSA]	PSA certification website: www.psacertified.org(1)

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

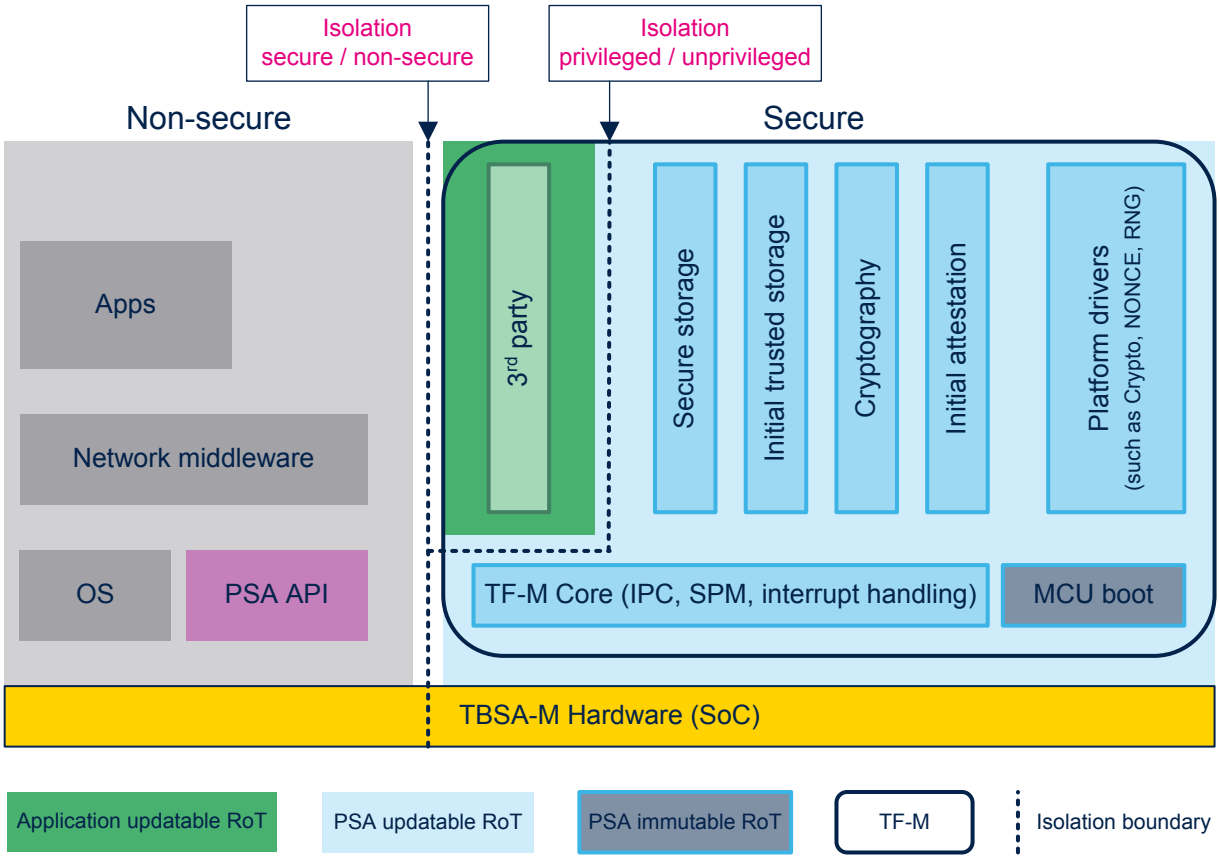
Note: Mbed is a trademark of Arm Limited (or its subsidiaries) in the US and or elsewhere.

3 Arm® Trusted Firmware-M (TF-M) introduction

TF-M (refer to [TF-M]) is an Arm Limited driven open-source software framework providing a reference implementation of the PSA standard on the Arm® Cortex®-M33 (TrustZone®) processor:

- PSA immutable RoT (Root of Trust): immutable *"Secure Boot and Secure Firmware Update"* application executed after any reset. This application is based on MCUboot open source software (refer to [MCUboot]).
- PSA updatable RoT: *"secure"* application implementing a set of secure services isolated in the secure/privileged environment that can be called by the non-secure application at non-secure application run-time via the PSA APIs (refer to [PSA_API]):
  - Secure storage service: TF-M secure storage (SST) service implements PSA protected storage APIs allowing data encryption and writing the result in a possibly untrusted storage. The SST service implements an AES-GCM based AEAD encryption policy, as a reference, to protect data integrity and authenticity.
  - Internal trusted storage service: TF-M internal trusted storage (ITS) service implements PSA internal trusted storage APIs allowing the writing of data in a microcontroller built-in Flash memory region that will be isolated from non-secure or from unprivileged applications by means of the hardware security protection mechanisms.
  - Cryptography service: the TF-M crypto service implements the PSA Crypto APIs that allow an application to use cryptography primitives such as symmetric and asymmetric ciphers, hash, message authentication codes (MACs), and authenticated encryption with associated data (AEAD). It is based on MbedCrypto open-source software (refer to [mbed-crypto]).
  - Initial attestation service: the TF-M initial attestation service allows the application to prove the device identity during an authentication process to a verification entity. The initial attestation service can create a token on request, which contains a fix set of device specific data.
- Application updatable RoT: third-party secure services that are isolated in the secure/unprivileged environment and that can be called by the non-secure application at non-secure application run-time.

Figure 1. TF-M overview



4 X-CUBE-SBSFU vs. TF-M comparison

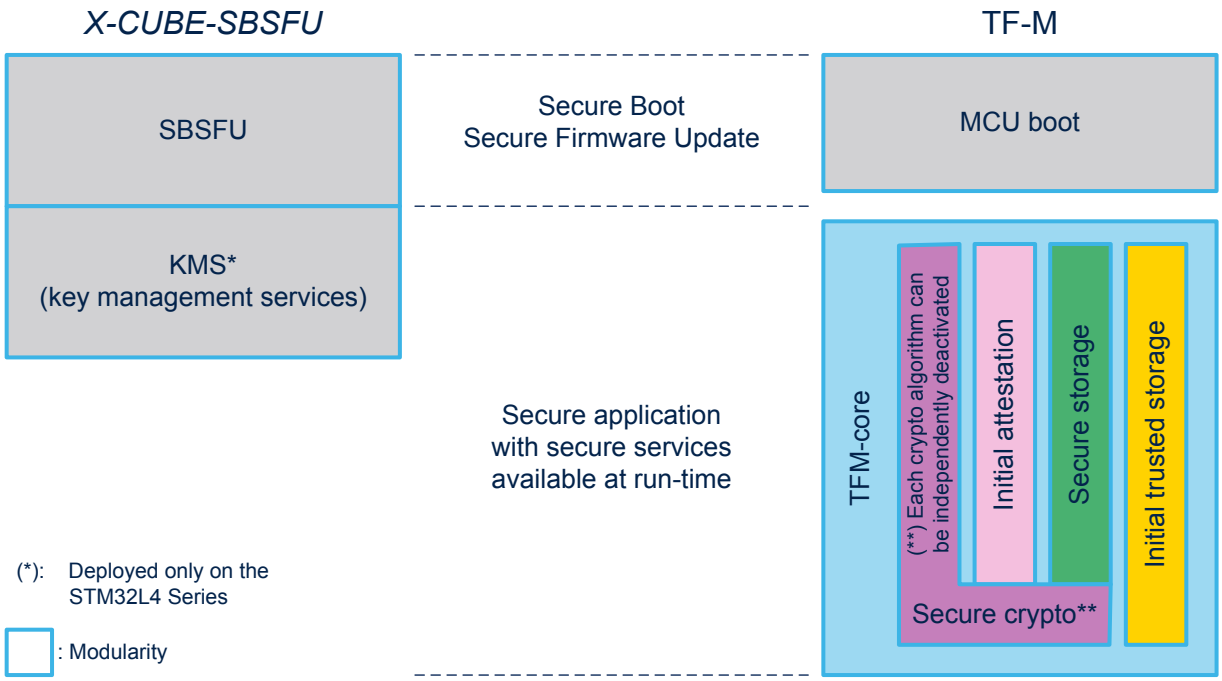
4.1 Overview

X-CUBE-SBSFU provides an STMicroelectronics implementation of Secure Boot and Secure Firmware Update, and optionally for some STM32 series only, secure KMS (key management services) service available at run-time for the user application.

The TF-M reference implementation provides Secure Boot and Secure Firmware Update services based on open-source MCU boot, and a set of secure services available at run-time for the user application.

The high-level comparison between *X-CUBE-SBSFU* and TF-M is shown in Figure 2.

Figure 2. *X-CUBE-SBSFU* vs. TF-M overview



The MCU boot part of the TF-M can be compared to *X-CUBE-SBSFU* (without KMS): it offers similar services.

X-CUBE-SBSFU KMS supports similar services as TF-M secure crypto services but the lists of cryptographic algorithms or features are not the same and APIs are different even if both are based on an opaque key API concept. Refer to the X-CUBE-SBSFU and TF-M APIs documents referenced in the related user manuals ([UM2262] and [UM2671]) to get more details about the supported features.

4.2 Top-level features

Even if *X-CUBE-SBSFU* and TF-M propose similar services, the features of both solutions are not exactly same. Table 4 summarizes the differences between *X-CUBE-SBSFU* in X-CUBE-SBSFU V2.3.1 and TF-M-based applications in STM32CubeL5 V1.3.0.

Table 4. *X-CUBE-SBSFU* vs. TF-M top-level features

Security topic	*X-CUBE-SBSFU* in X-CUBE-SBSFU V2.3.1[1]	TF-M in STM32CubeL5 V1.3.0[1]
SBSFU	1 or 2 slots per image. New image via local loader or USER APP. Encrypted image execution in external Flash memory.	1 or 2 slots per image. New image via local loader or USER APP. Encrypted image execution in external Flash memory.
	Single firmware image. Full or **partial** update.	Single firmware image **or multiple (2) firmware images (secure and non-secure)**. Full update only.
	Symmetric crypto scheme. Asymmetric crypto scheme (ECDSA) with or without firmware encryption.	Asymmetric crypto scheme (**RSA** or ECDSA) with or without firmware encryption.
Run-time secure services	Secure services • 1 level of isolation • Interruption not managed • Main crypto services (STM32L4 Series only)	Secure services • **2 levels of isolation** • Non-secure interruption managed • **Complete crypto services (full SW or mixed SW&HW)** • **Initial attestation** • **Secure Storage (data encryption/integrity)** • **Internal trusted storage (data integrity)** • **Architecture ready to integrate unprivileged application services**

1. *Differences are highlighted in bold.*

To get an up-to-date view of the feature differences, refer to latest versions of the X-CUBE-SBSFU Expansion Package and STM32CubeL5 MCU Package available from their product pages on *www.st.com*.

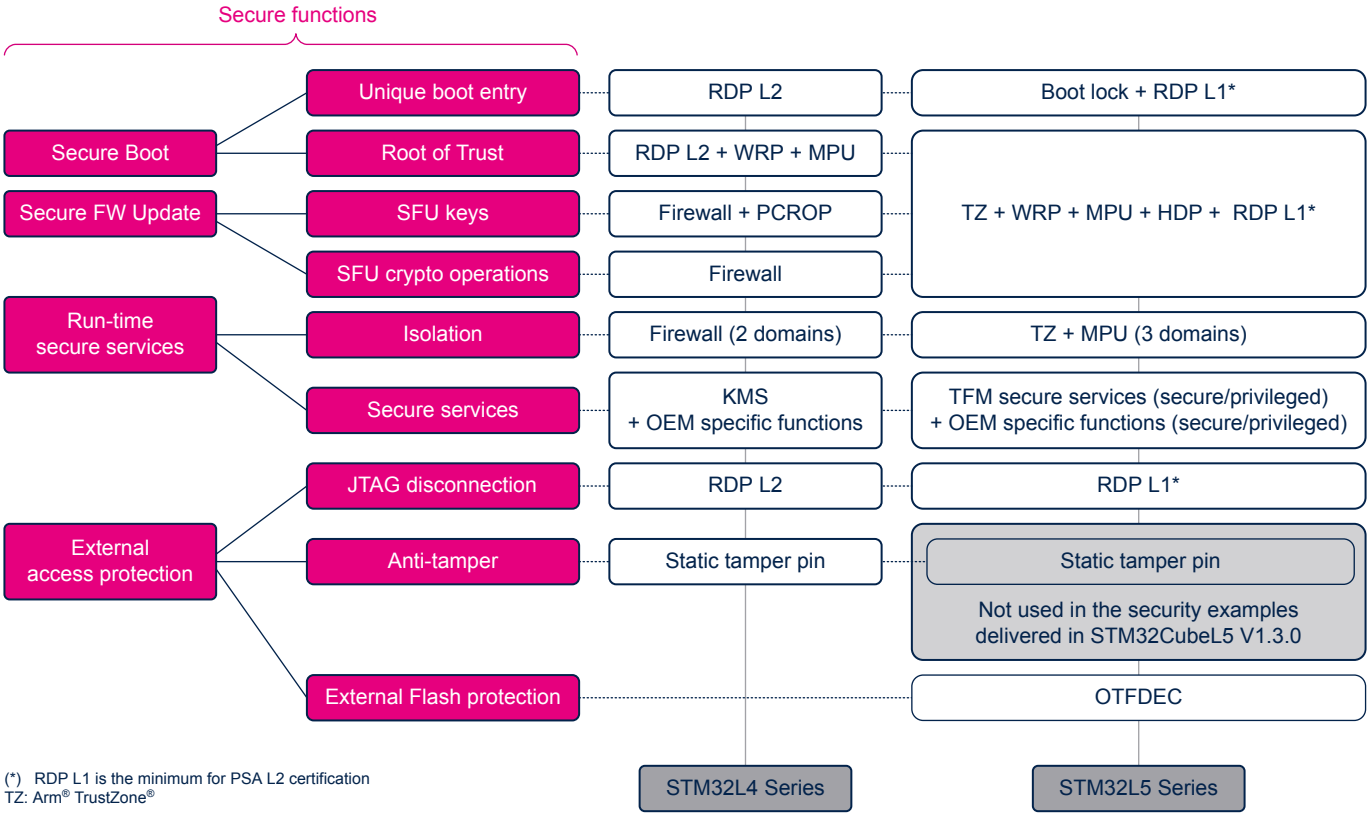
For more information on TF-M features, refer to [UM2671].

4.3 Hardware security

The security strategy of the TF-M-based applications in STM32CubeL5 is relying on TrustZone® and STM32L5 Series hardware security features.

Figure 3 shows the comparison of this security strategy with the *X-CUBE-SBSFU* security strategy in X-CUBE-SBSFU (for the STM32L4 Series as example).

Figure 3. *X-CUBE-SBSFU* (STM32L4 Series) and TF-M (STM32L5 Series) security strategy overview



(*) RDP L1 is the minimum for PSA L2 certification
TZ: Arm® TrustZone®

For more details on security strategy with TF-M, refer to [UM2671].

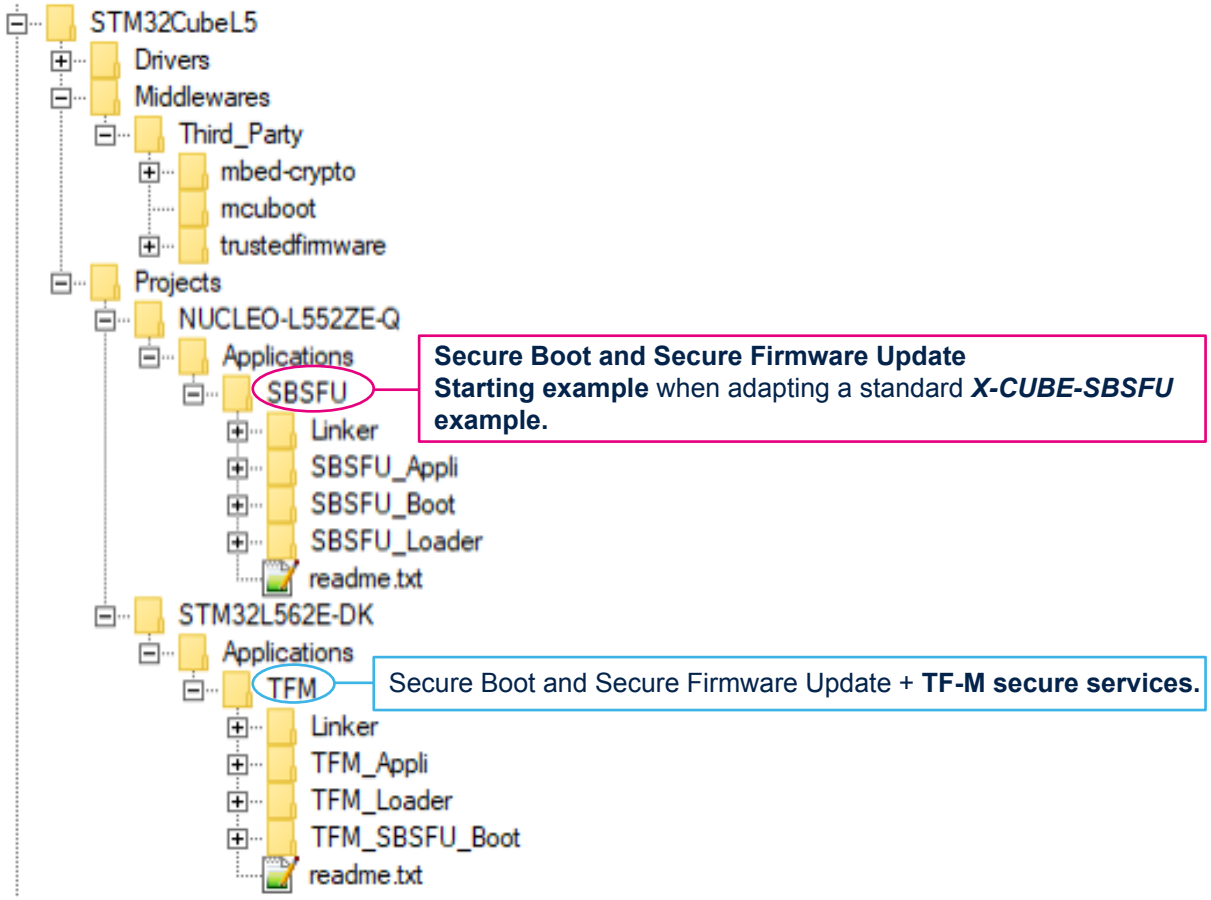
5 STM32CubeL5 TF-M-based applications

The STM32CubeL5 MCU Package proposes two different applications based on the TF-M reference implementation, ported onto the STM32L5 Series to take benefit of the hardware security features.

- *SBSFU*: it consists of the *"Secure Boot and Secure Firmware Update"* application (named `SBSFU_Boot`) and simple user application example (named `SBSFU_Appli`). A local loader application example (named `SBSFU_Loader`) is also included.
- TFM: it consists of the *"Secure Boot and Secure Firmware Update"* application (named `TFM_SBSFU_Boot`) and user application with TFM secure services at run-time (named `TFM_Appli`). A local loader application example (named `TFM_Loader`) is also included.

Users of *X-CUBE-SBSFU* without KMS are advised to consider the migration to the STM32L5 Series and use of STM32CubeL5 *SBSFU* application. Users of *X-CUBE-SBSFU* with KMS are advised to consider the migration to the STM32L5 Series and use of STM32CubeL5 TFM application (possibly removing some secure services or cryptographic algorithms to fit the application needs).

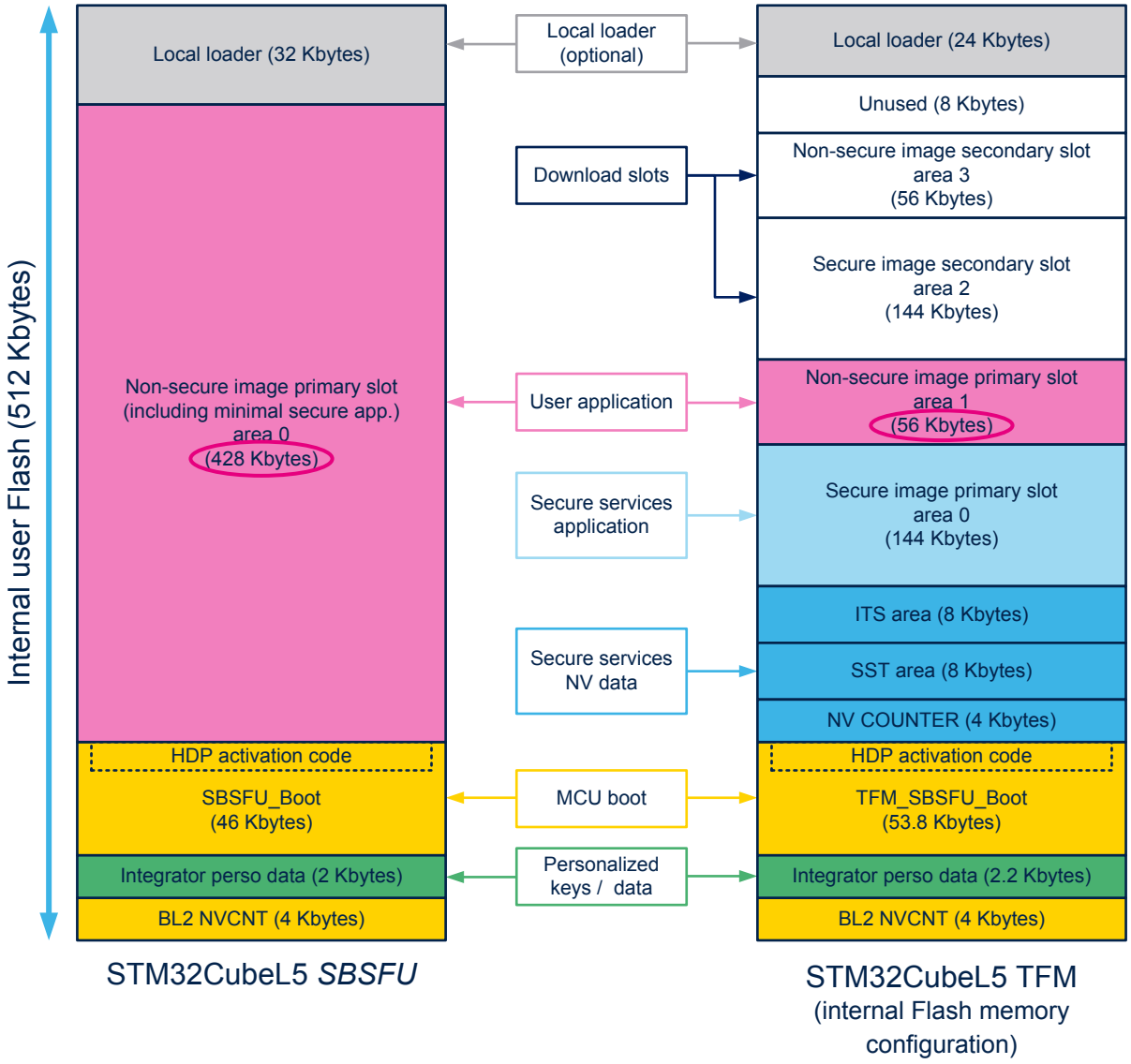
Figure 4. STM32CubeL5 applications based on TF-M



For each application, the memory footprint depends on the configuration (refer to the *Memory layout* section in [UM2671]).

By removing the TF-M secure services at run-time and by proposing 1 firmware image configuration combined with primary slot only configuration, the STM32CubeL5 *SBSFU* application maximizes the amount of internal Flash memory available for the user application as shown in Figure 5.

Figure 5. Memory footprint example of STM32CubeL5 applications based on TF-M



For more details on memory mapping, refer to the *Memory layout* section in [UM2671].

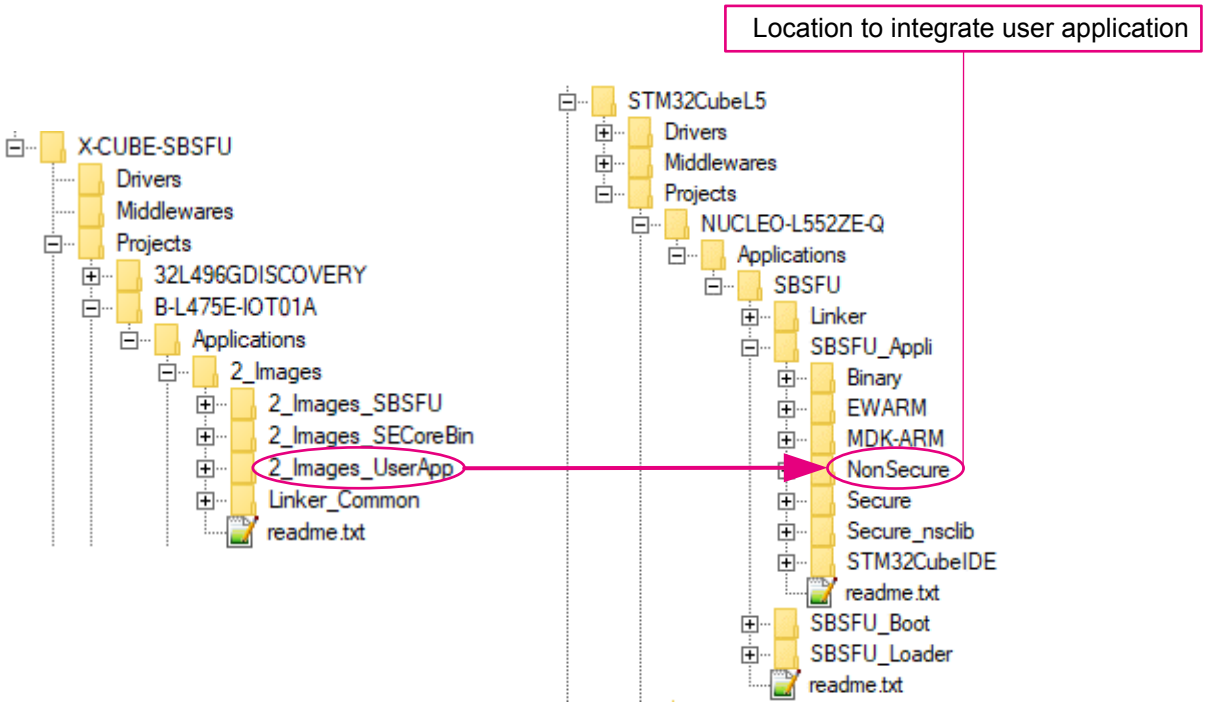
6 STM32CubeL5 *SBSFU* application

6.1 User application integration

When migrating from the *X-CUBE-SBSFU* application to STM32CubeL5 *SBSFU*, the user application must be integrated into the `SBSFU/SBSFU_Appli/NonSecure` folder as shown in Figure 6.

This folder contains a simple user application example.

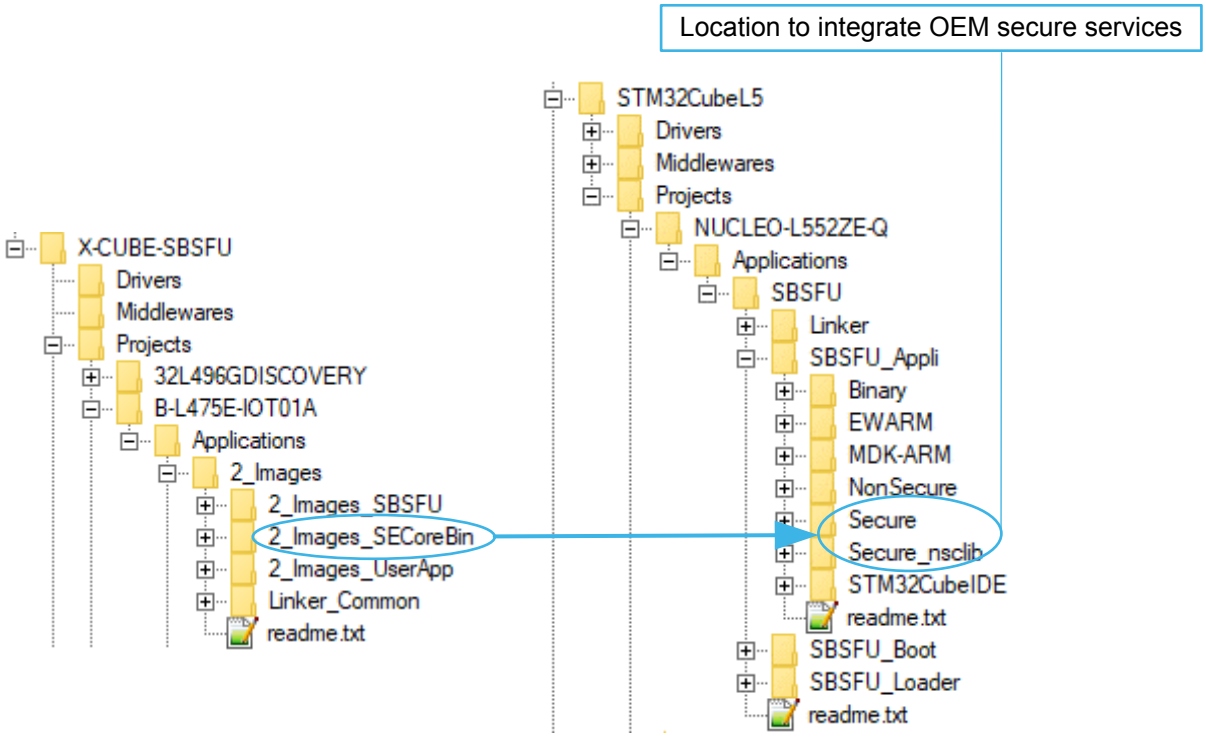
Figure 6. User application integration



6.2 OEM secure services integration

If OEM own secure services are also implemented in X-CUBE-SBSFU, these OEM secure services must be integrated into the `SBSFU/SBSFU_Appli/Secure` and `SBSFU/SBSFU_Appli/Secure_ncslib` folders, following STM32CubeL5 TrustZone® HAL examples, as shown in Figure 7.
These folders contain a simple example of OEM secure service: *"Secure GPIO Toggle"*.

Figure 7. OEM secure services integration (*SBSFU*)



6.3 Keys personalization

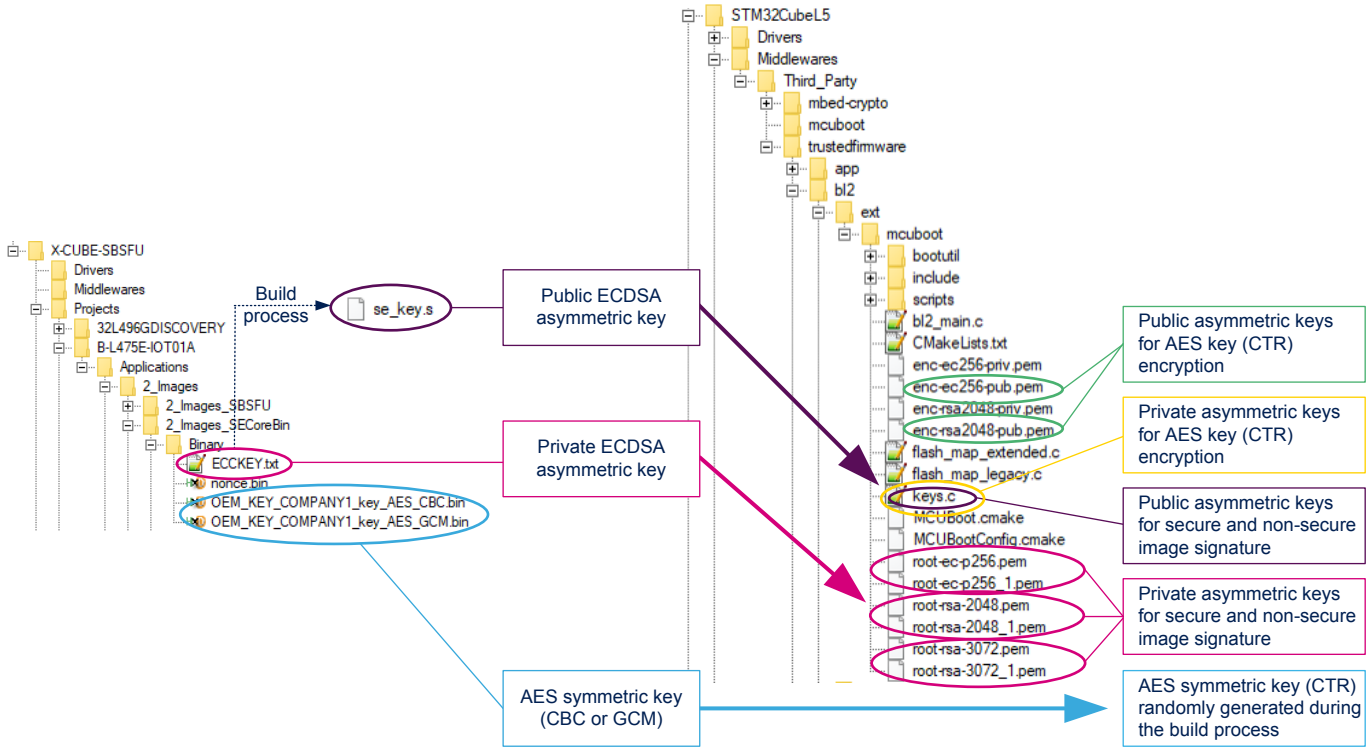
In *X-CUBE-SBSFU*, the personalization data are cryptographic keys:

- ECDSA asymmetric key: for firmware image signature
- AES symmetric key (CBC or GCM): for firmware image encryption

In *SBSFU* in STM32CubeL5 V1.3.0, for firmware image signature, there are two RSA or ECDSA asymmetric keys (one for secure image, and one for non-secure image) to personalize, compared to one ECDSA asymmetric key in *X-CUBE-SBSFU*. It must be noticed that contrary to *X-CUBE-SBSFU*, the public asymmetric keys are not automatically generated during the build process of STM32CubeL5 *SBSFU* but need to be provided by the user together with the private asymmetric keys (refer to Figure 8).

SBSFU in STM32CubeL5 V1.3.0 supports firmware encryption with AES-CTR cryptography. Compared to *X-CUBE-SBSFU*, the AES-CTR key is not present in the personalized data, but is randomly generated during each build process, is encrypted (RSA-OAEP or ECIES-P256) and provided in the firmware image itself. The asymmetric key (RSA or ECDSA) used to encrypt the AES-CTR key is distinct from the asymmetric signature keys. Both public and private asymmetric keys for AES-CTR key encryption must be provided by the user (refer to Figure 8).

Figure 8. Firmware image keys personalization



The two private RSA or ECDSA asymmetric keys used to sign the secure and non-secure firmware images are not embedded in the Flash memory, whereas the two associated public RSA or ECDSA asymmetric keys are present in the build output of the `SBSFU_Boot` project. They are embedded in a dedicated immutable Flash region (personalization data area) as shown in Figure 9.

The public RSA or ECDSA asymmetric key used to encrypt the AES-CTR key is not embedded in the Flash memory, whereas the associated private RSA or ECDSA asymmetric key is present in the build output of `SBSFU_Boot` project, in the personalization data area as well, as shown in Figure 9.

Figure 9. Integrator personalized data area in STM32CubeL5 *SBSFU*

7 STM32CubeL5 TFM application

The top-level integration guidelines provided in Section 6 STM32CubeL5 SBSFU application are applicable to the STM32CubeL5 TFM application. In this section, additional top-level integration guidelines specific to the STM32CubeL5 TFM application are provided.

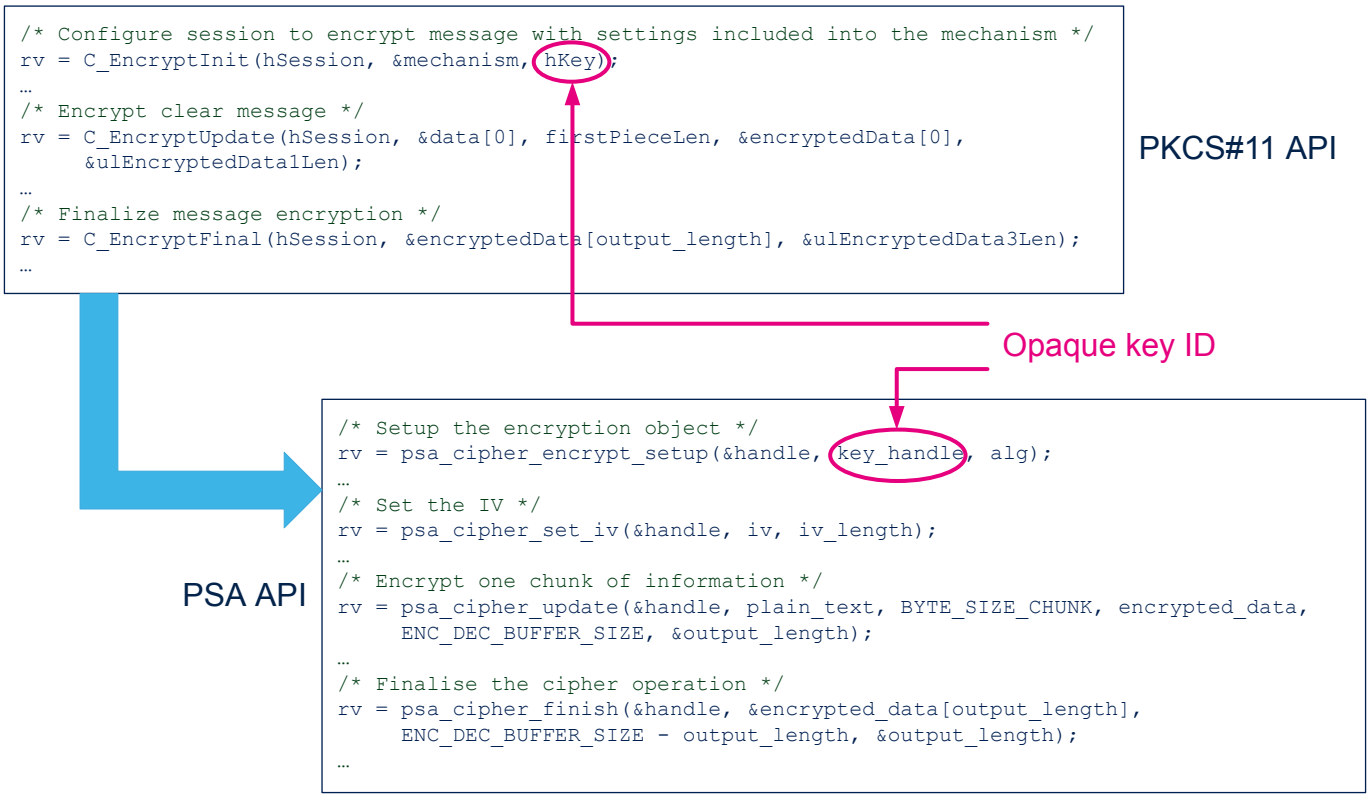
To get more information on STM32CubeL5 TFM application, refer to [UM2671].

7.1 Cryptographic secure services at run-time

In *X-CUBE-SBSFU*, the KMS services are provided to the user application through PKCS#11 APIs. In STM32CubeL5 TFM application, the secure cryptography services are provided to the user application through PSA cryptographic APIs. Both are based on an opaque key APIs concept.

Figure 10 shows an example of API usage difference for AES encryption.

Figure 10. PSA API migration example

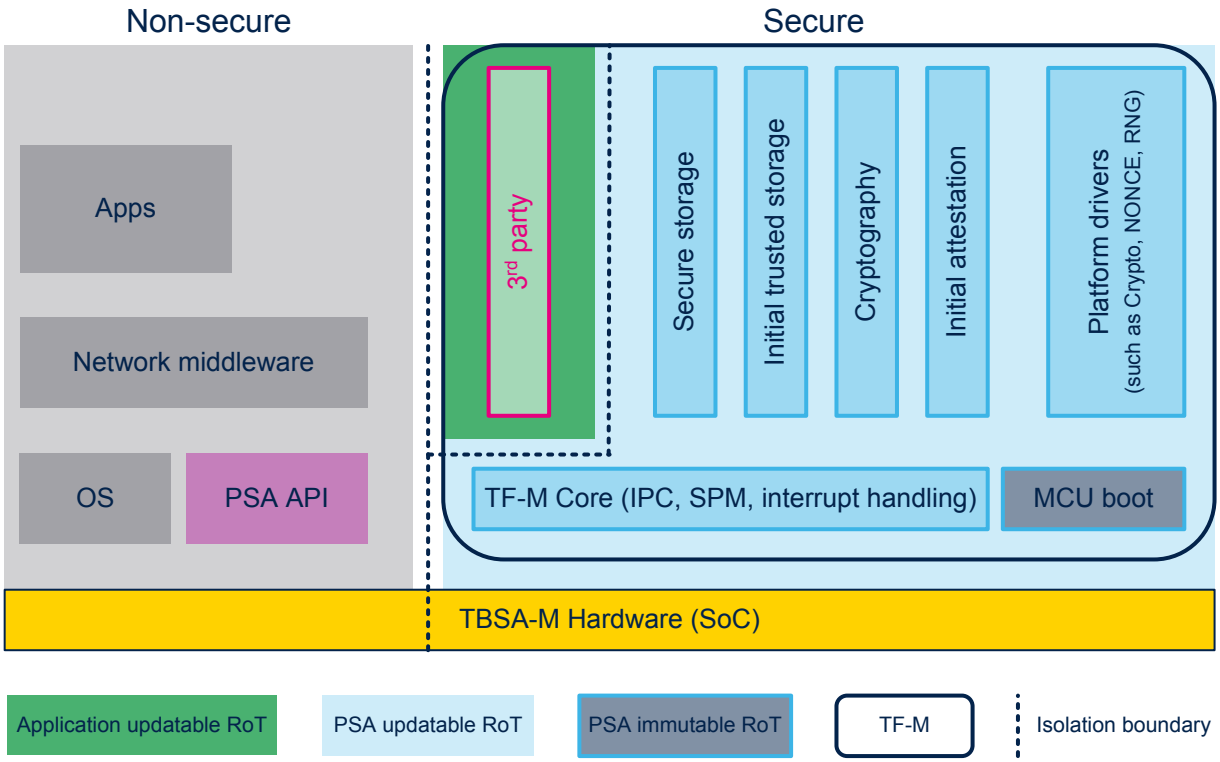


For more information on PSA APIs, refer to TFM user application example and [PSA_API].

7.2 OEM secure services integration

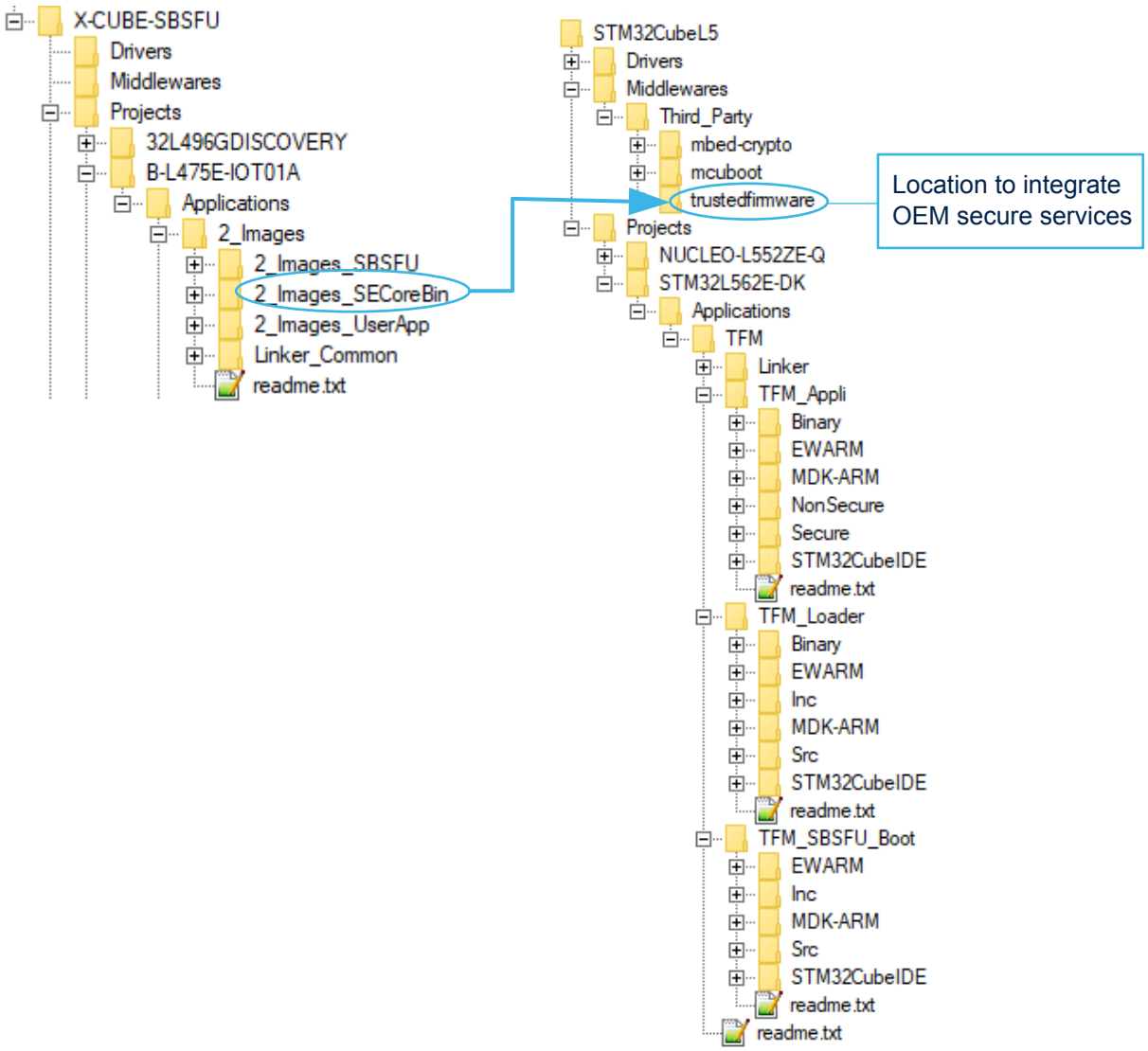
As shown in Figure 11, the OEM secure services must be integrated as third-party secure services in the secure/unprivileged part of the secure application (referred to as *"Application RoT from TFM framework"*). For more information on *"Application RoT from TFM framework"*, refer to [UM2671].

Figure 11. 3rd party secure services in TF-M



These services must be integrated in the `Middlewares/trustedfirmware` folder as shown in Figure 12. For more information, refer to [TFM_USER_GUIDE].

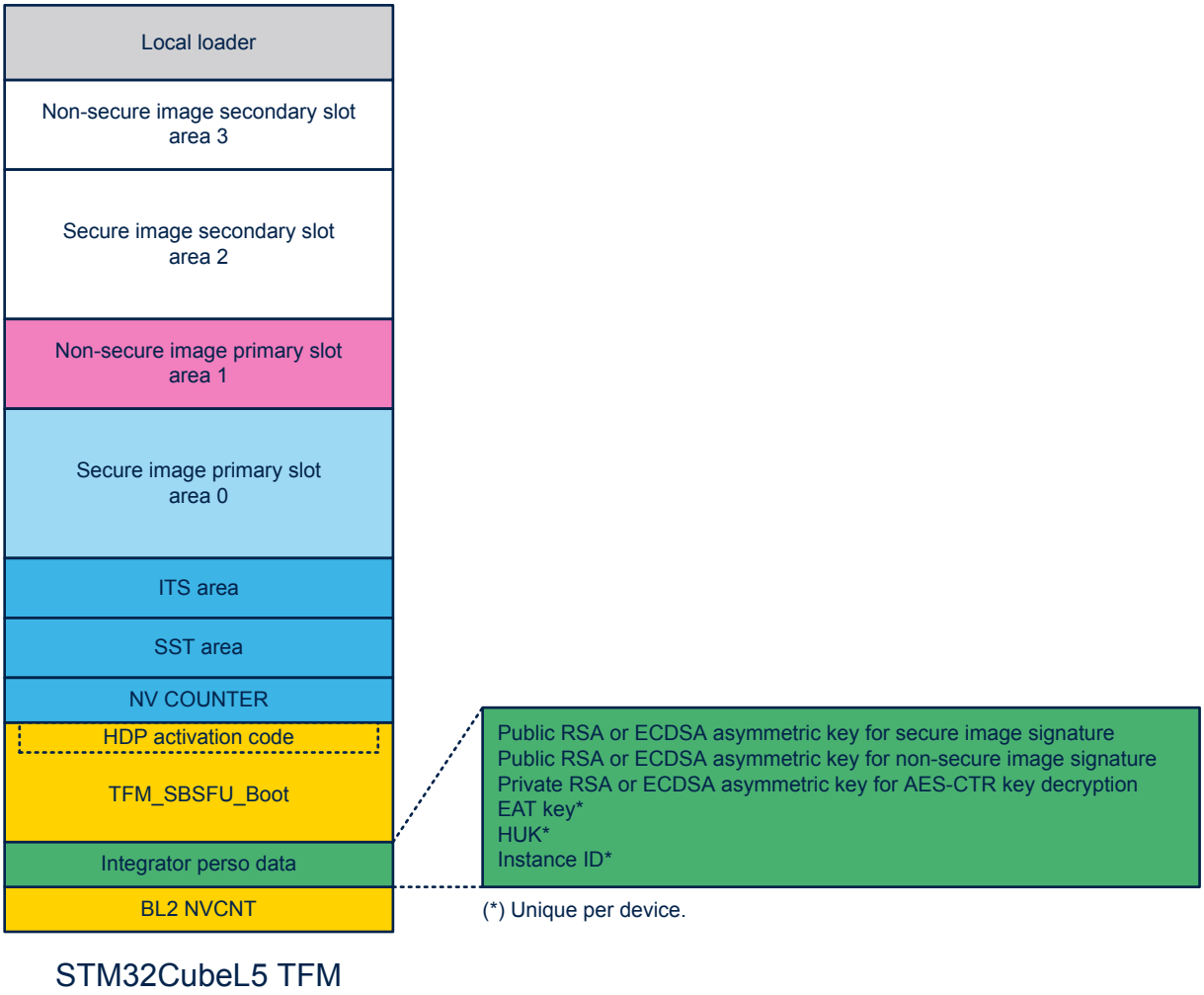
Figure 12. OEM secure services integration (TFM)



7.3 Data personalization

In addition to the firmware image authentication keys, additional data requires personalization for the TFM application: EAT key, HUK, and Instance ID. These data are required for TF-M initial attestation service. They are product-specific (unique per device). These data, together with the asymmetric keys for images signature and AES CTR key decryption (refer to Section 6.3 Keys personalization), are grouped in the dedicated immutable Flash region (personalization data area), which must be personalized for each device in production, before activating the final security configuration.

Figure 13. Personalization data region



For more details on personalization data, refer to section *Integrator role description* in [UM2671].

Revision history

Table 5. **Document revision history**

Date	Version	Changes
20-Feb-2020	1	Initial release.
24-Jul-2020	2	Updated the entire document for STM32CubeL5 firmware V1.3.0 release. New features in `SBSFU_Boot`: image encryption, external Flash memory support with OTFDEC, configurable crypto schemes, configurable image number mode, configurable slot mode, and RSA hardware accelerator. Local loader introduced.

Contents

List of tables

List of figures