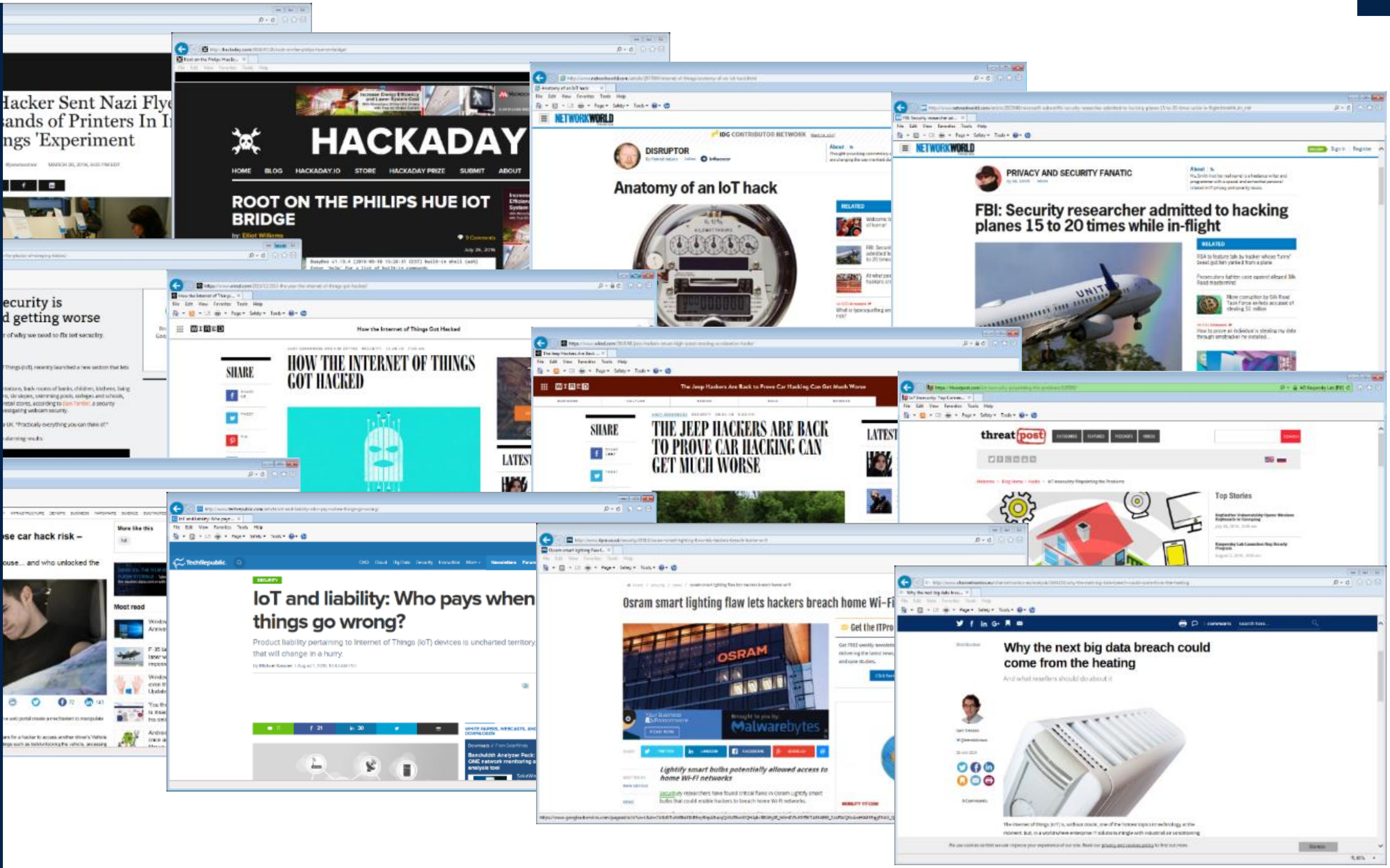
life.augmented

STM32Trust SBSFU

Secure Boot
Secure Firmware Update

STM32

In the news


Hacker Sent Nazi Fly
ands of Printers In I
ngs 'Experiment

HACKADAY

ROOT ON THE PHILIPS HUE IOT BRIDGE

Anatomy of an IoT hack

FBI: Security researcher admitted to hacking planes 15 to 20 times while in-flight

ecurity is
d getting worse

HOW THE INTERNET OF THINGS GOT HACKED

THE JEEP HACKERS ARE BACK TO PROVE CAR HACKING CAN GET MUCH WORSE

se car hack risk –

IoT and liability: Who pays when things go wrong?

Product liability pertaining to Internet of Things (IoT) devices is uncharted territory that will change in a hurry

Osram smart lighting flaw lets hackers breach home Wi-Fi

Lightify smart bulbs potentially allowed access to home Wi-Fi networks

Why the next big data breach could come from the heating

And what resellers should do about it

Connected objects
Our concern for tomorrow

2020

Operating system -based solutions

65%

Connected objects

20 Billion

2025

Embedded solutions

65%

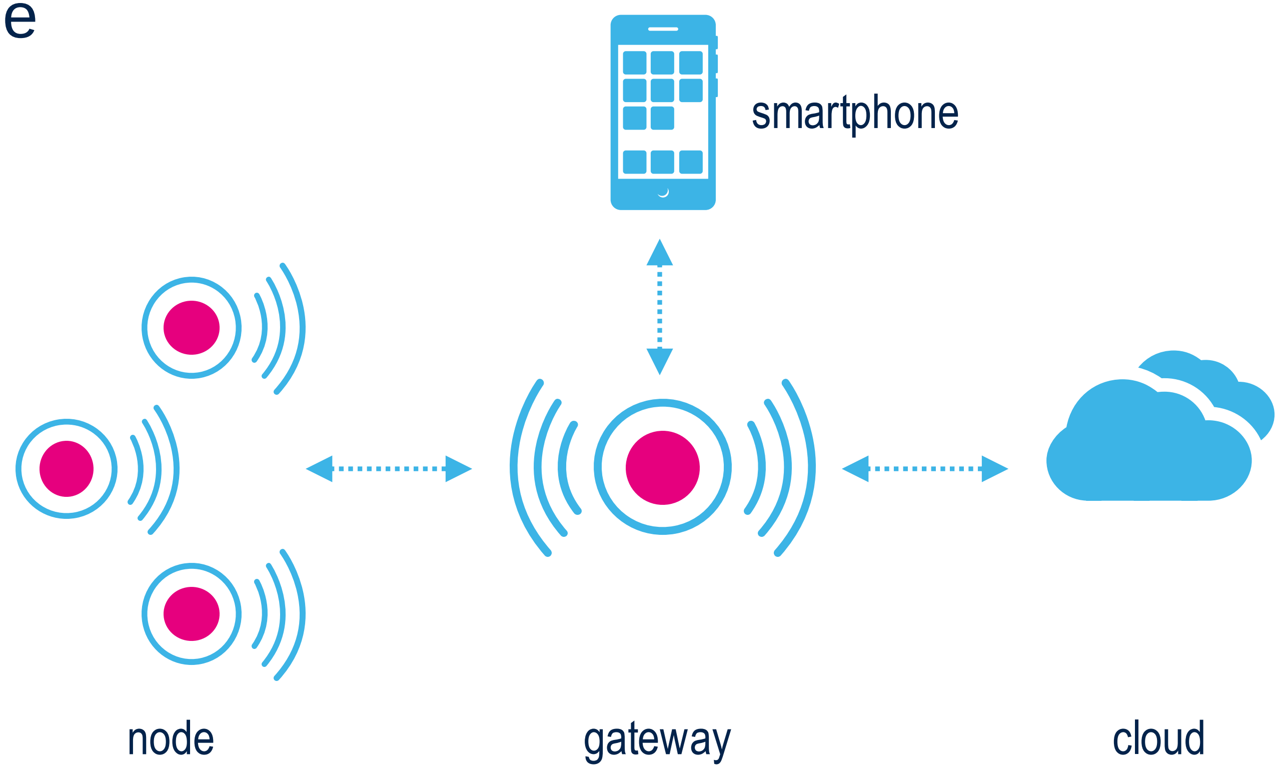
Connected objects

48 Billion

Security in embedded devices is crucial

Service providers need to protect the quality and reputation of:

- Their **Services**
  What the end customer pays for
- Their **Networks**
  Avoid Denial of Service
  Provide quality/reliability
- Their **Brand**
  Ensuring trust



Categories of attacks

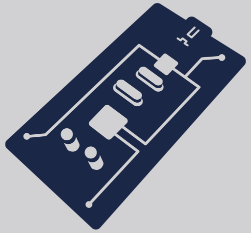
95% of IoT attacks today

Logical

- Local or remote
- Open ports
- SW Bugs
- Debug I/Fs

and more…

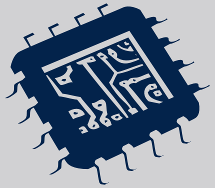
Cloning attacks



Board-level

- Memory probing
- « Mod-chips »
- Fault injection
- Side-channels

and more…



Chip-level

- Probing
- Laser
- FIB
- Reverse Eng.

and more…

Cost and expertise of attack materials

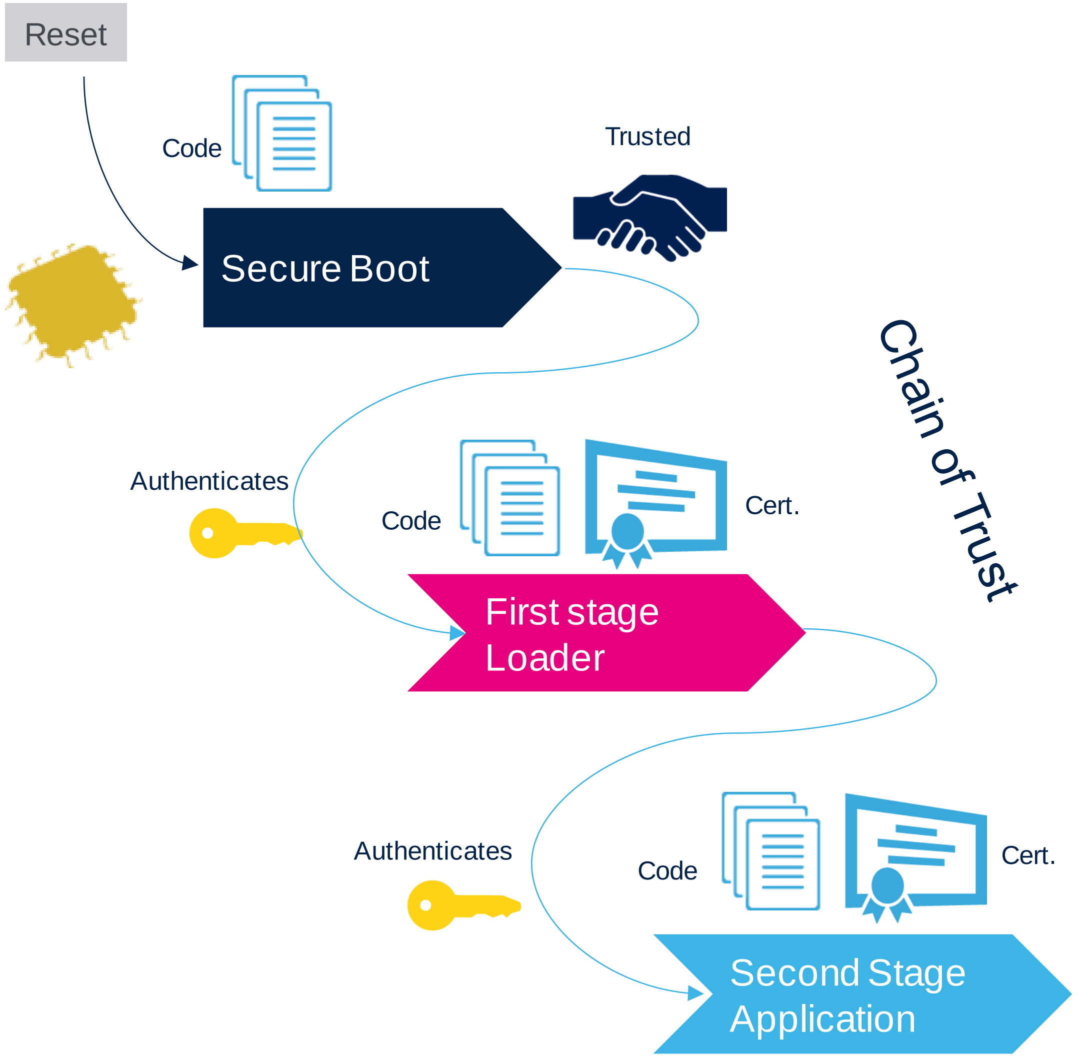
- **Logical attack**
  From outside the box
- **Board-level attack**
  From Inside the box
- **Chip-level attack**
  From Inside the chip

(now covered by STM32U5)

Secure Boot
Root of Trust

Goal of Secure Boot / Root of Trust

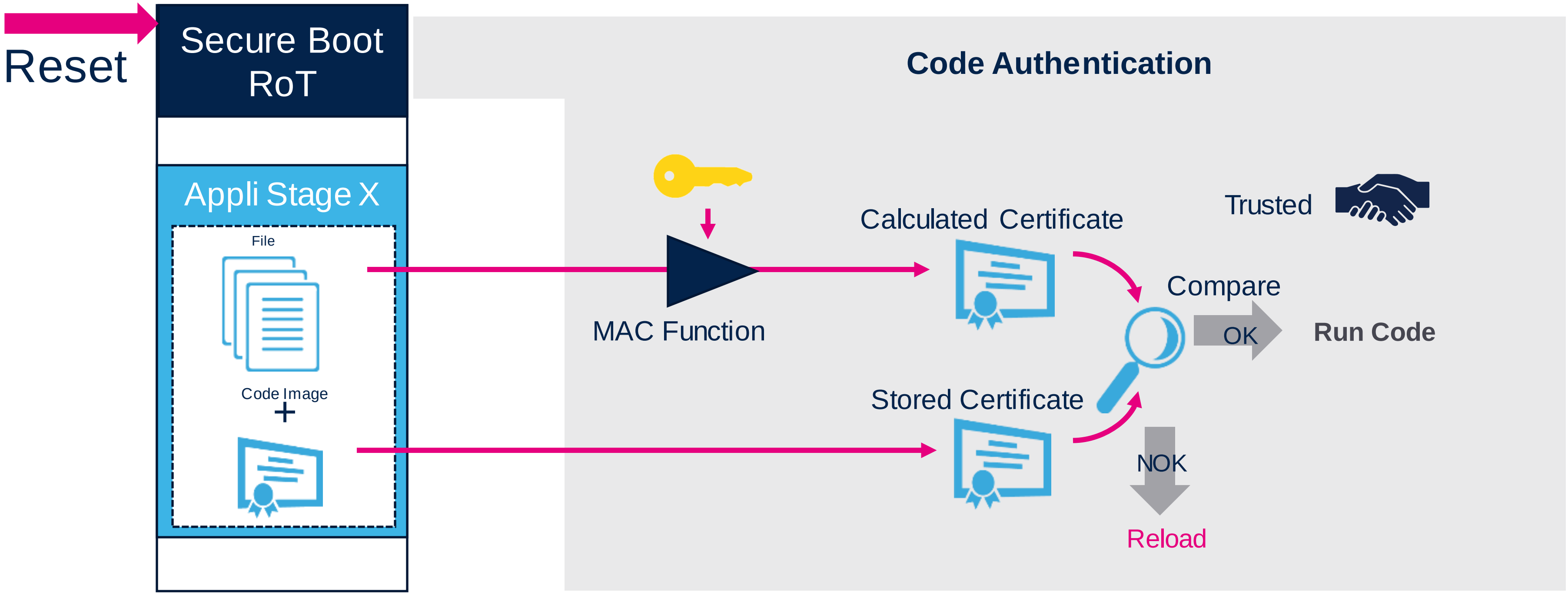
- Immutable Secure Boot code
- Executed first at reset
- Verify platform integrity
  - Clock settings
  - Register configurations
  - Memory protection
- Launch Root-of-Trust services
  - Code authentication
  - Uses cryptographic keys and encryption functions



life.augmented

Root of Trust general process

- Performed at each RESET, using a key 🔑 stored in the device
- It is a predictable process



Market trend

- Few OEMs are using Secure Boot / Secure Firmware Update
- No single standardized Secure Boot / Root of Trust model
- Key IoT players are spreading good security practices
- IoT standardization bodies are growing with clear security requirements

Secure Boot and **Secure Firmware Update**
help build the Root of Trust
that most potential **vendors will require** to access their networks

life.augmented

How to support this approach

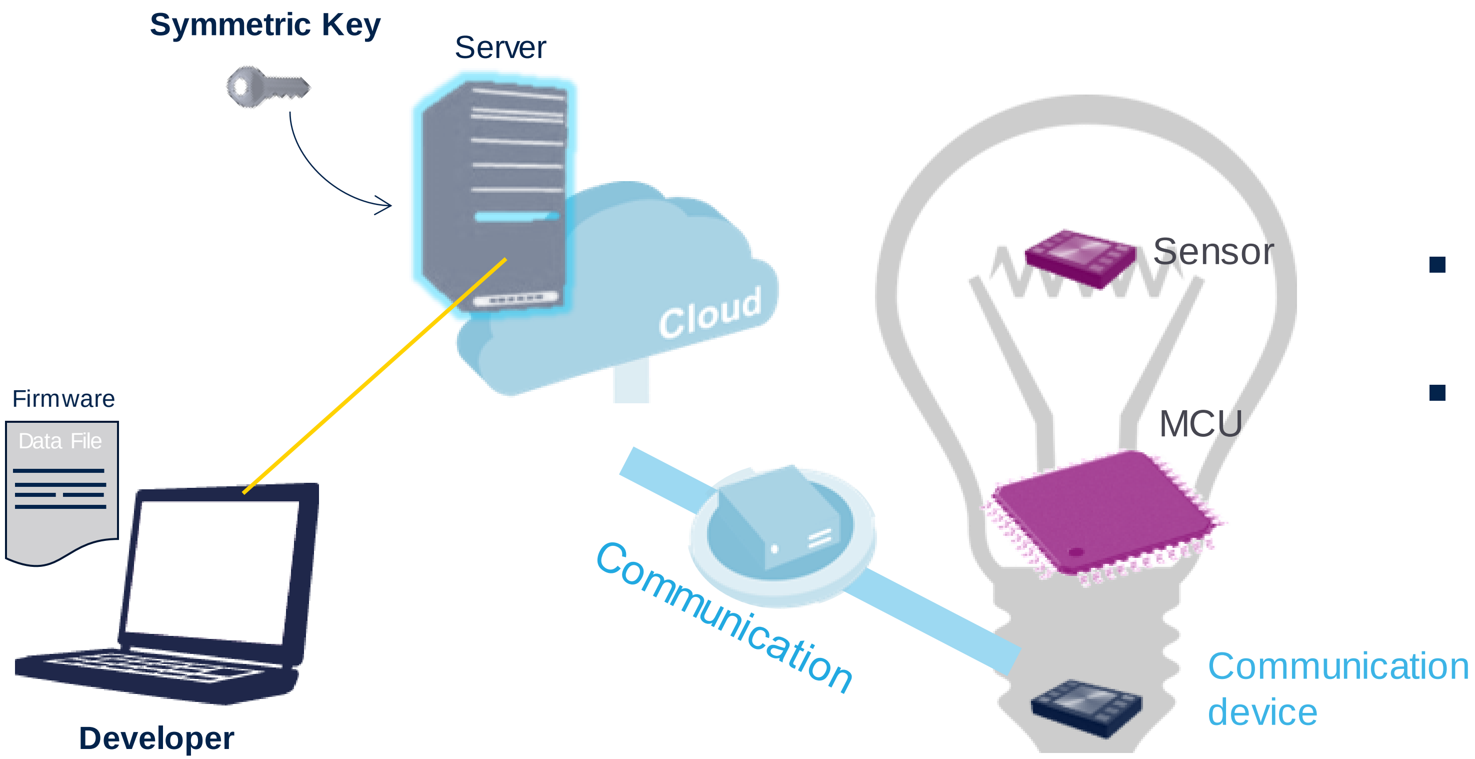
- Embedded ROMed code

SB / RoT approach	feasibility	remarks
One code on all STM32	☺	May not be market acceptable
Multiple code on STM32	☹	Diversify products Increase development, qualification, certification, cost

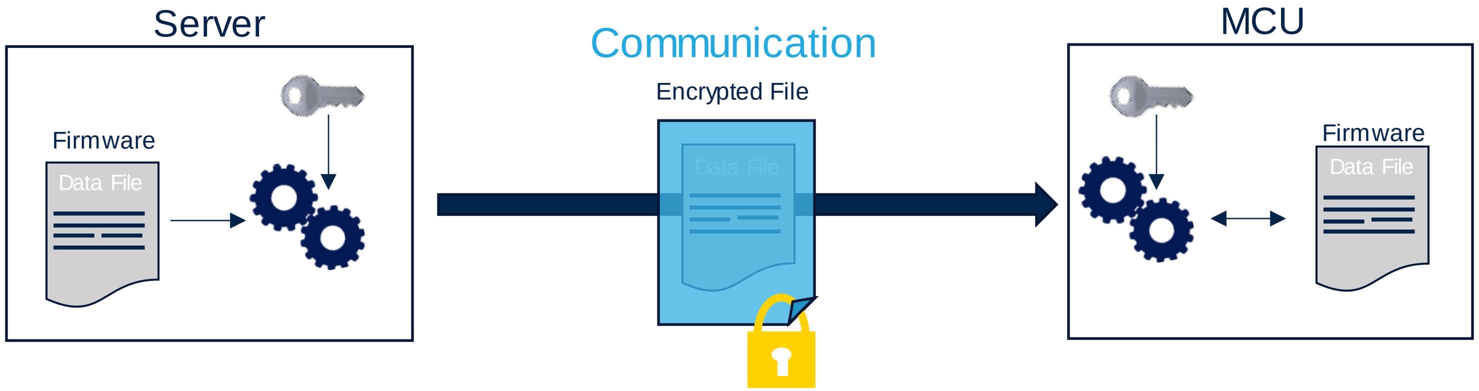
- ST's approach
  - Allow industries to develop their own Secure Boot / Root of Trust approach
  - Propose a way to securely load it into STM32
  - Propose a way to isolate and securely execute it within STM32

life.augmented

Secure Firmware Update



- Server sends Firmware (FW) Package
- Device verifies the new FW package, unwraps it and executes it



Secure Firmware Update

FIRMWARE UPDATE

Do not switch off the ignition!
Do not remove the USB stick!

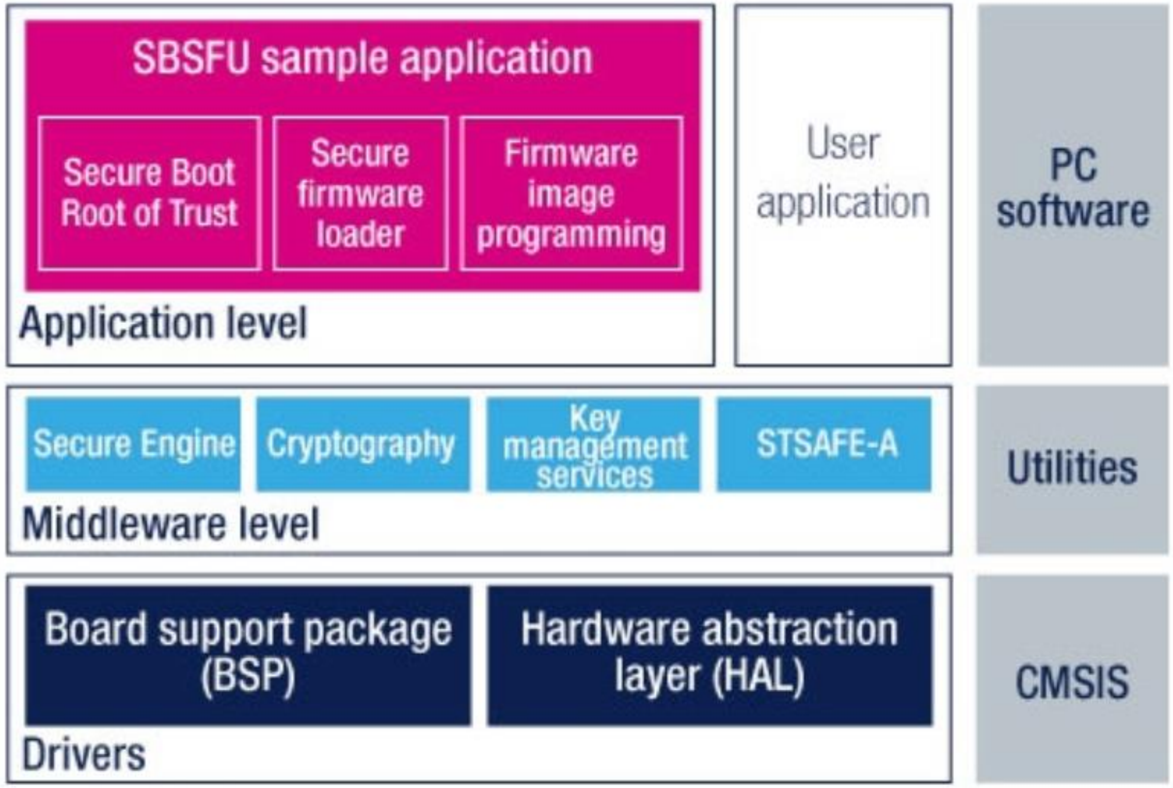
SWL Ongoing: System has limited functionality

- Complete process performed in a secure way
- Prevent unauthorized updates
- Access to secret code and key
- Access to confidential on-device data
- Developed in several software modules

Introducing X-CUBE-SBSFU

- **X-CUBE-SBSFU** is an STM32Cube expansion package which enables the secure update of the built-in STM32 program with new firmware versions and prevents:
  - unauthorized updates
  - access to confidential on-device data

- **X-CUBE-SBSFU** (on **STM32L4**)
  - Certified SESIP Level 3



SBSFU sample application

Secure Boot Root of Trust

Secure firmware loader

Firmware image programming

Application level

User application

PC software

Secure Engine

Cryptography

Key management services

STSAFE-A

Middleware level

Utilities

Board support package (BSP)

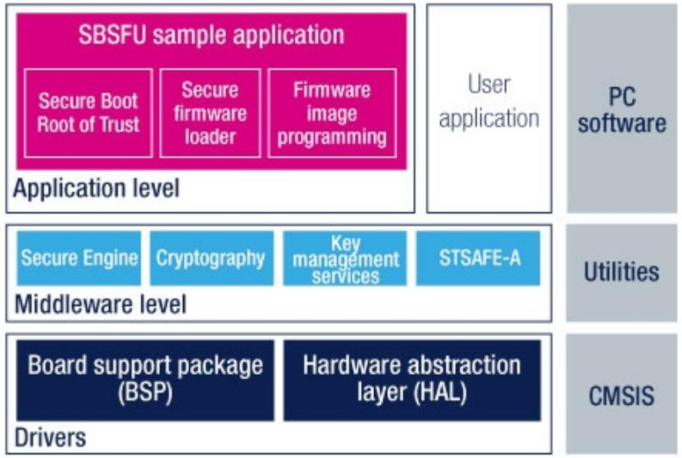
Hardware abstraction layer (HAL)

Drivers

CMSIS

X-CUBE-SBSFU package overview

- **Secure Boot** (SB) module
  - Execution with Root of Trust service
  - Application authentication and Integrity check before execution
- **Secure Firmware Update** (SFU) module
  - Detect new FW version to install
    - From local download service
    - Pre-downloaded OTA via User application from previous execution…
  - Manage FW version (check unauthorized updates or unauthorized installation)
  - Secure FW upgrade:
    - FW Authentication and Integrity check / decryption / installation
  - In case of any error occurring during new image installation rollback to the previous valid version
- **Secure element** support
  - STSAFE-A middleware provides a complete set of APIs to access all the features of STSAFE-A110 secure element



SBSFU sample application

Secure Boot Root of Trust

Secure firmware loader

Firmware image programming

User application

PC software

Application level

Secure Engine

Cryptography

Key management services

STSAFE-A

Utilities

Middleware level

Board support package (BSP)

Hardware abstraction layer (HAL)

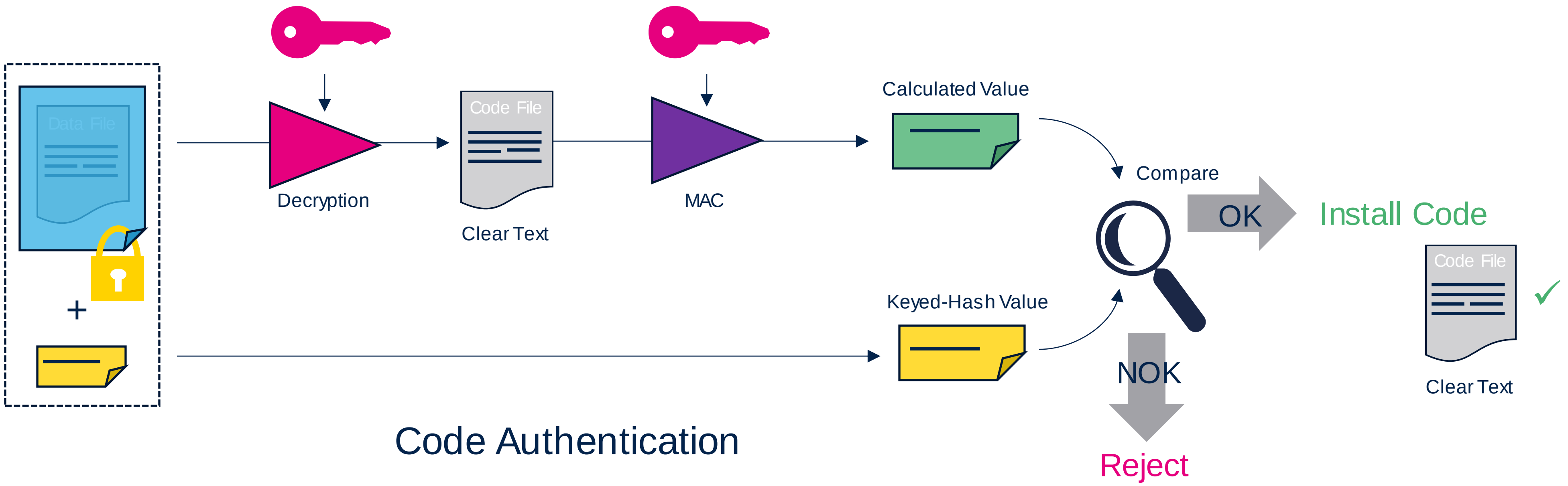
CMSIS

Drivers

- **Secure Engine** (SE) module
  - Code isolated from main firmware → secure execution
  - Dedicated to crypto algorithms execution
  - Manage secure key storage
- **Key Management Services** (KMS)
  - The KMS services provide cryptographic services to the user application through the PKCS #11 APIs (KEY ID-based APIs) that are executed inside the secure enclave

Secure Firmware Update process

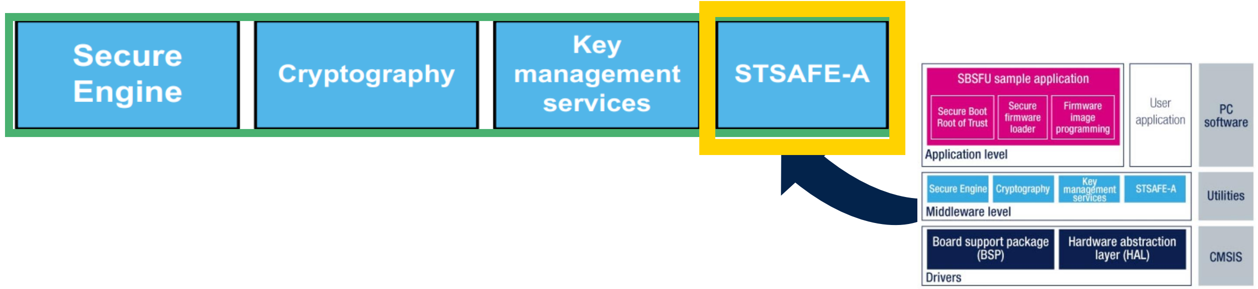
- Performed when a new image is available by using a shared key 🔑 stored in the device
- Each new image is authenticated before being installed



Code Authentication

Middleware

- Secure Engine
  - provides a protected environment to manage all critical data and operations
- Cryptography:
  - X-CUBE-CRYPTOLIB
  - mbedTLS
  - mbed-crypto
- Key Management Services
  - The secure key management services provide cryptographic services to the user application through the PKCS #11 APIs (KEY ID-based APIs) that are executed inside the secure enclave
- STSAFE-A
  - STSAFE-A middleware provides a complete set of APIs to access all the features of STSAFE-A110 secure element



Secure Engine

Cryptography

Key management services

STSAFE-A

SBSFU sample application

Secure Boot Root of Trust

Secure firmware loader

Firmware image programming

Application level

User application

PC software

Secure Engine

Cryptography

Key management services

STSAFE-A

Middleware level

Utilities

Board support package (BSP)

Hardware abstraction layer (HAL)

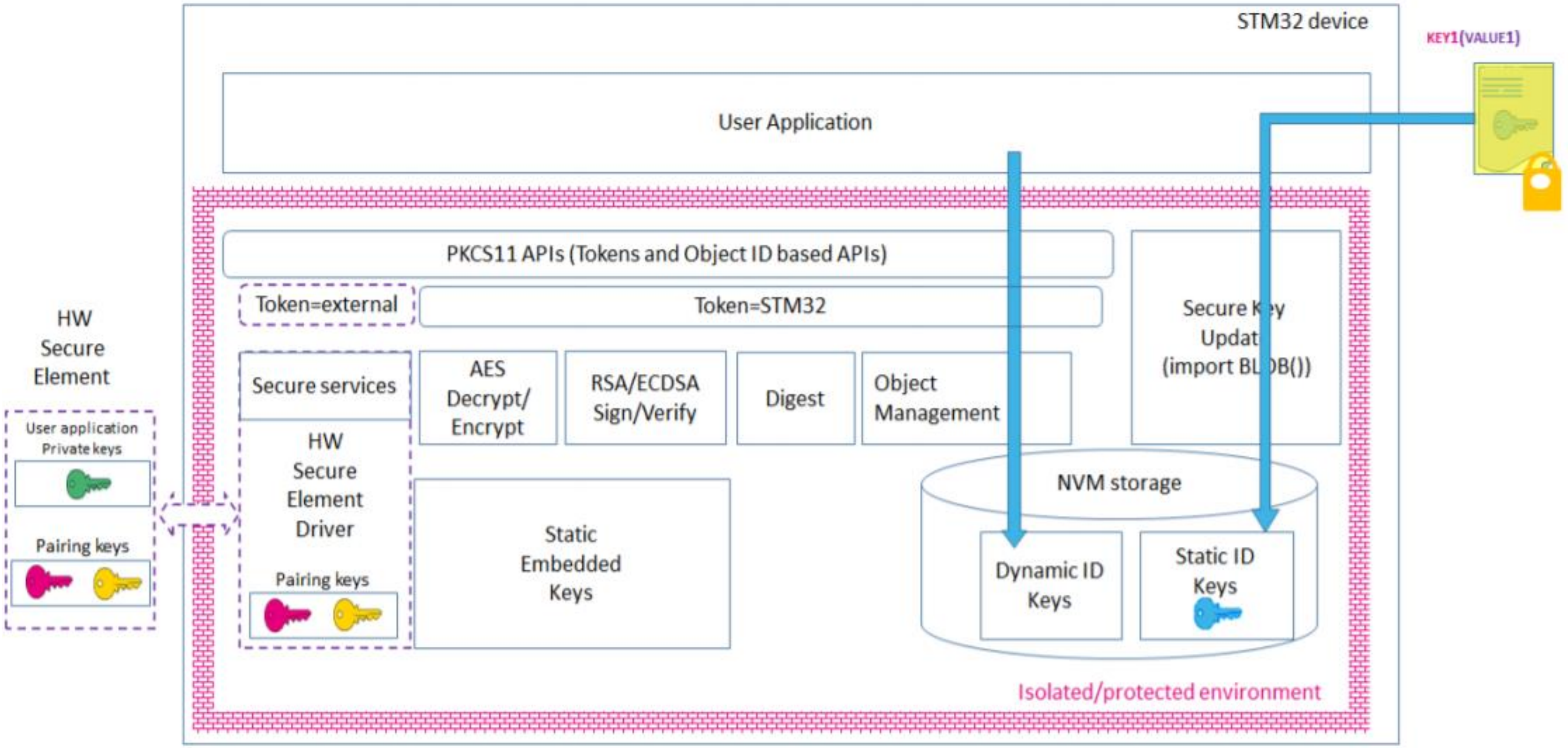
Drivers

CMSIS

life.augmented

Key management services - KMS

- Provide partial PKCS11 support
- Opaque key management
- Access to secure element



STM32 device

KEY1(VALUE1)

User Application

PKCS11 APIs (Tokens and Object ID based APIs)

Token=external

Token=STM32

Secure Key Update (import BLOB())

HW Secure Element

User application Private keys

Pairing keys

Secure services

HW Secure Element Driver

Pairing keys

AES Decrypt/ Encrypt

RSA/ECDSA Sign/Verify

Digest

Object Management

Static Embedded Keys

NVM storage

Dynamic ID Keys

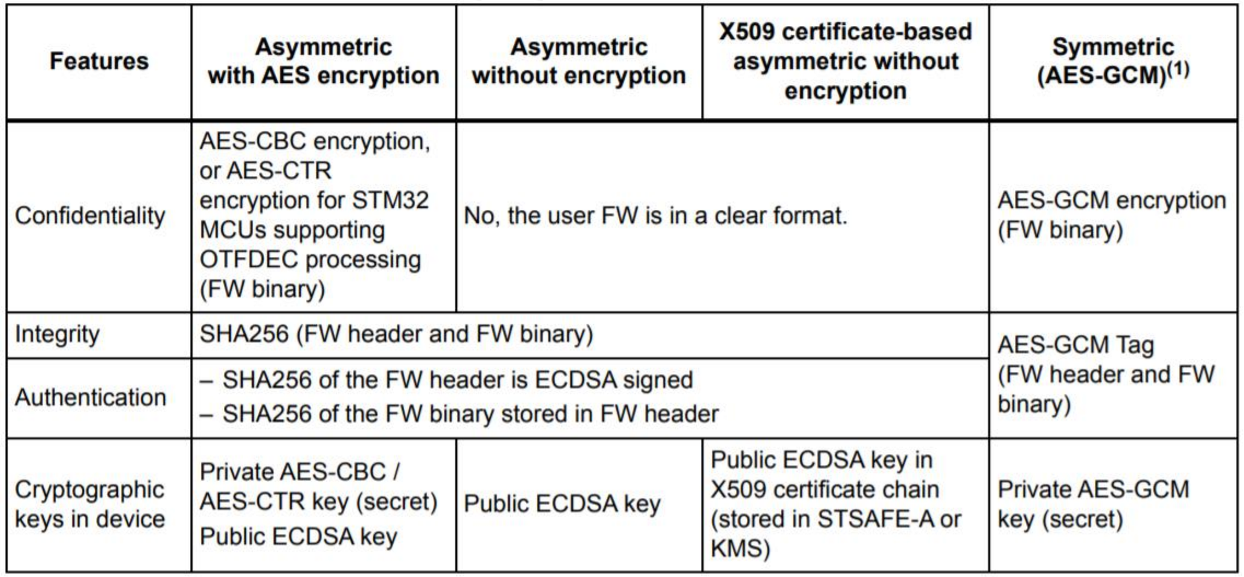
Static ID Keys

Isolated/protected environment

Cryptography

Four cryptographic schemes using both asymmetric and symmetric cryptography

- **ECDSA asymmetric** cryptography for firmware verification with AES-CBC or AES-CTR symmetric cryptography for firmware encryption
- **ECDSA asymmetric** cryptography for firmware verification without firmware encryption
- **X509 certificate-based ECDSA asymmetric** cryptography for firmware verification without firmware encryption
- **AES-GCM symmetric** cryptography for both firmware verification and encryption



Features	Asymmetric with AES encryption	Asymmetric without encryption	X509 certificate-based asymmetric without encryption	Symmetric (AES-GCM)[1]
Confidentiality	AES-CBC encryption, or AES-CTR encryption for STM32 MCUs supporting OTFDEC processing (FW binary)	No, the user FW is in a clear format.		AES-GCM encryption (FW binary)
Integrity	SHA256 (FW header and FW binary)			AES-GCM Tag (FW header and FW binary)
Authentication	– SHA256 of the FW header is ECDSA signed – SHA256 of the FW binary stored in FW header			
Cryptographic keys in device	Private AES-CBC / AES-CTR key (secret) Public ECDSA key	Public ECDSA key	Public ECDSA key in X509 certificate chain (stored in STSAFE-A or KMS)	Private AES-GCM key (secret)

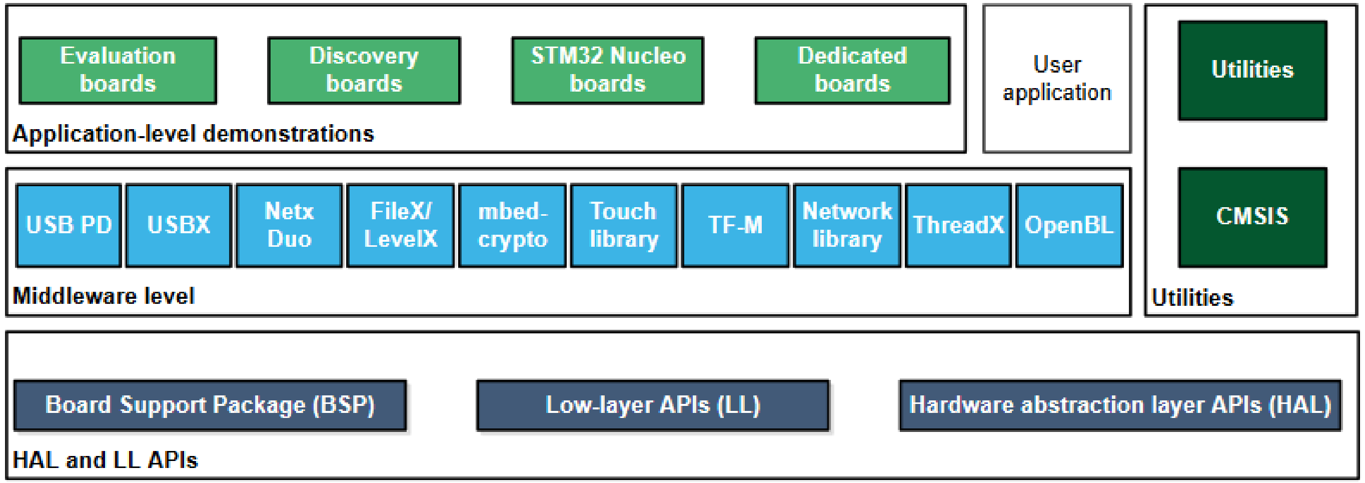
Secure engine

- The **Secure Engine (SE)** concept defines a protected enclave exporting a set of secure functions executed in a trusted environment
- It allows the partitioning between privileged & un-privileged application segments
- It uses firewall and/or MPU with a call-gate mechanism
  - Preventing un-privileged functions to execute in privileged mode

Introducing STM32Cube

- **Applies to STM32L5 and STM32U5**

- **Embedded SW for STM32U5 Series**
  - Production-ready HAL and LL API drivers
  - CMSIS CORE, DSP and RTOS SW components
  - Comprehensive middleware around Azure RTOS & ARM TF-M
  - Certified SESIP Level 3



Evaluation boards
Discovery boards
STM32 Nucleo boards
Dedicated boards
Application-level demonstrations

User application

Utilities
CMSIS
Utilities

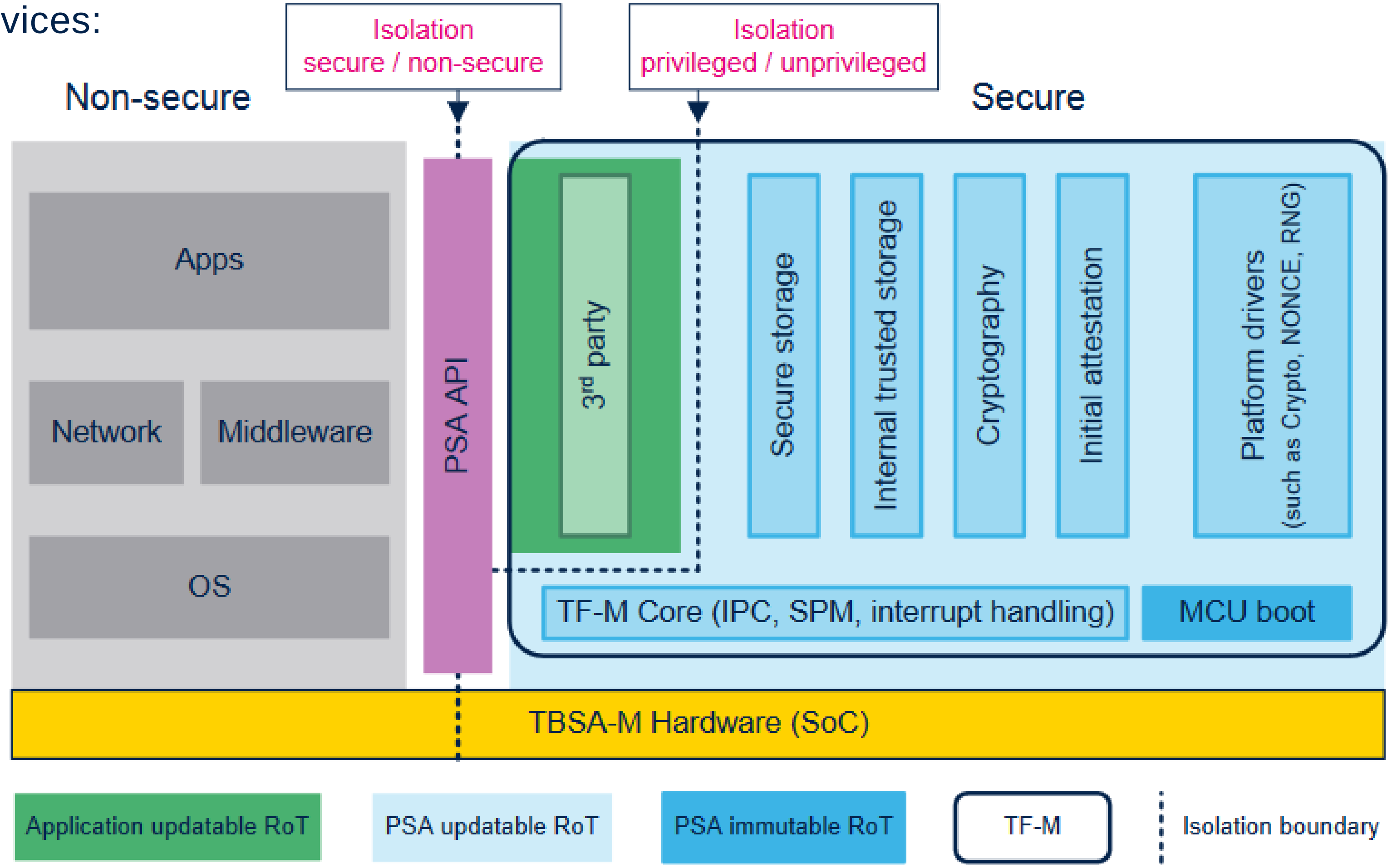
USB PD
USBX
Netx Duo
FileX/ LevelX
mbed-crypto
Touch library
TF-M
Network library
ThreadX
OpenBL
Middleware level

Board Support Package (BSP)
Low-layer APIs (LL)
Hardware abstraction layer APIs (HAL)
HAL and LL APIs

STM32Cube with ARM PSA & TF-M

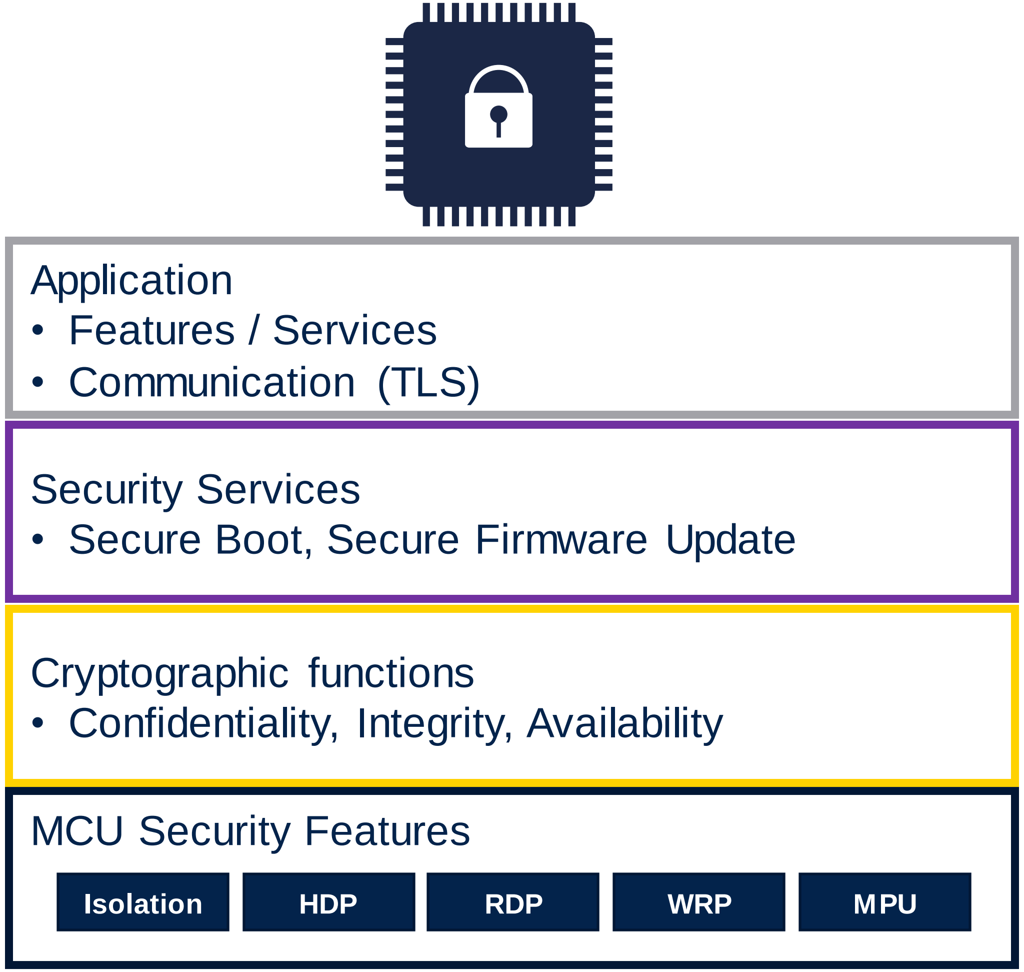
- **TF-M a reference implementation of ARM PSA standard**
  - ARM Cortex-M33 processor with TrustZone
  - PSA immutable RoT as Secure Boot and Secure Firmware Update
  - PSA updateable RoT featuring secure services:
    - Secure Storage (SST)
    - Internal Trusted Storage (ITS)
    - Crypto
    - Initial attestation
  - Application updatable RoT
    - 3rd-party secure services

    in secure/unprivileged environment



Security layering

- MCU Security Features
  - Used to establish a robust platform on which trusted processes and associated cryptography can be performed
- Cryptographic Functions
  - Preserve confidentiality, verify integrity, authenticity
- Secure Boot and Secure Firmware Update
  - Establishing a Root of Trust
  - Building a system that can evolve to counter new threats, add new functionality, fix bugs in a controlled and secure way once device is in the field



Application
- Features / Services
- Communication (TLS)

Security Services
- Secure Boot, Secure Firmware Update

Cryptographic functions
- Confidentiality, Integrity, Availability

MCU Security Features

STM32 Static Memory Protections

Readout Protection (RDP)

- Level 0: no readout protection
- Level 0.5: secure memory readout protection (L5/U5 with TrustZone)
- Level 1: memory readout protection
- Level 2: chip readout protection

Flash code, register and secure SRAM

Can’t be dumped through debug I/F or by the CPU itself booted from external memory

Write protection (WRP)

- 1 each per Flash / SRAM sector

Flash code is protected from unwanted write/erase operations

Hide protection (HDP)

- Applies to U5

Watermark-based secure area

Execute once then access denied

STM32 Dynamic Protections

Isolation

- Code or data protection in Flash or SRAM

Trusted execution region
Ideal to protect sensitive function and IP from the rest of the application
Firewall on L4, ARM TrustZone on L5/U5

MPU

- Memory isolation
- Hard-fault or core lock-up in case of violation

Read, Write, execute attribute per region
Prevent Stack Overflow
System protection against unintended modification

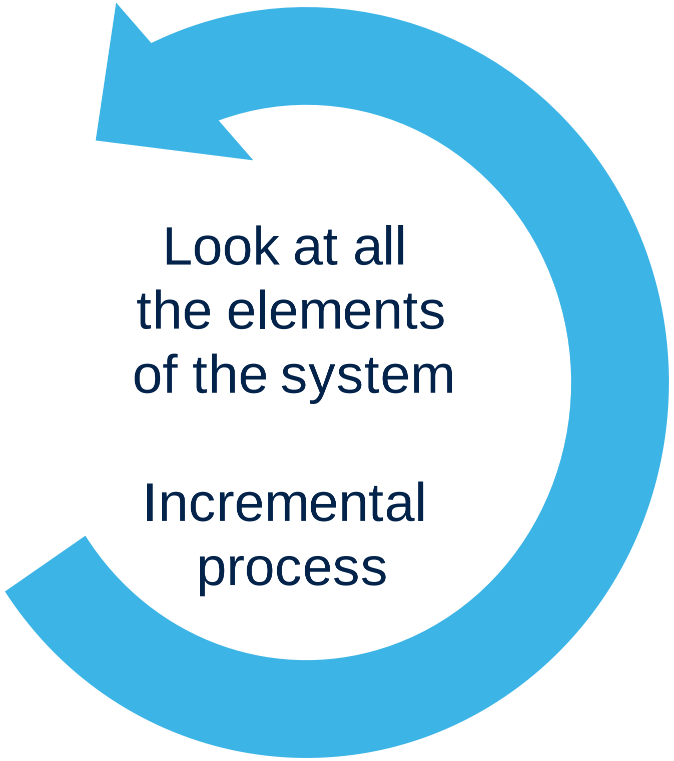
Backup domain and Anti-Tamper

- Independent voltage
- RTC, Backup SRAM
- Tamper detection pin

Detection of tamper event
Reset of all backup register
Time stamp event

Security mindset

- Start by defining your security needs
  - What do you want to protect?
  - What do you want to protect your asset against?
- Look into how to protect your asset
- Evaluate the level of protection
  - Does it fully protect your application?
  - Does it bring additional weaknesses?
  - Does it require additional elements to be optimal?

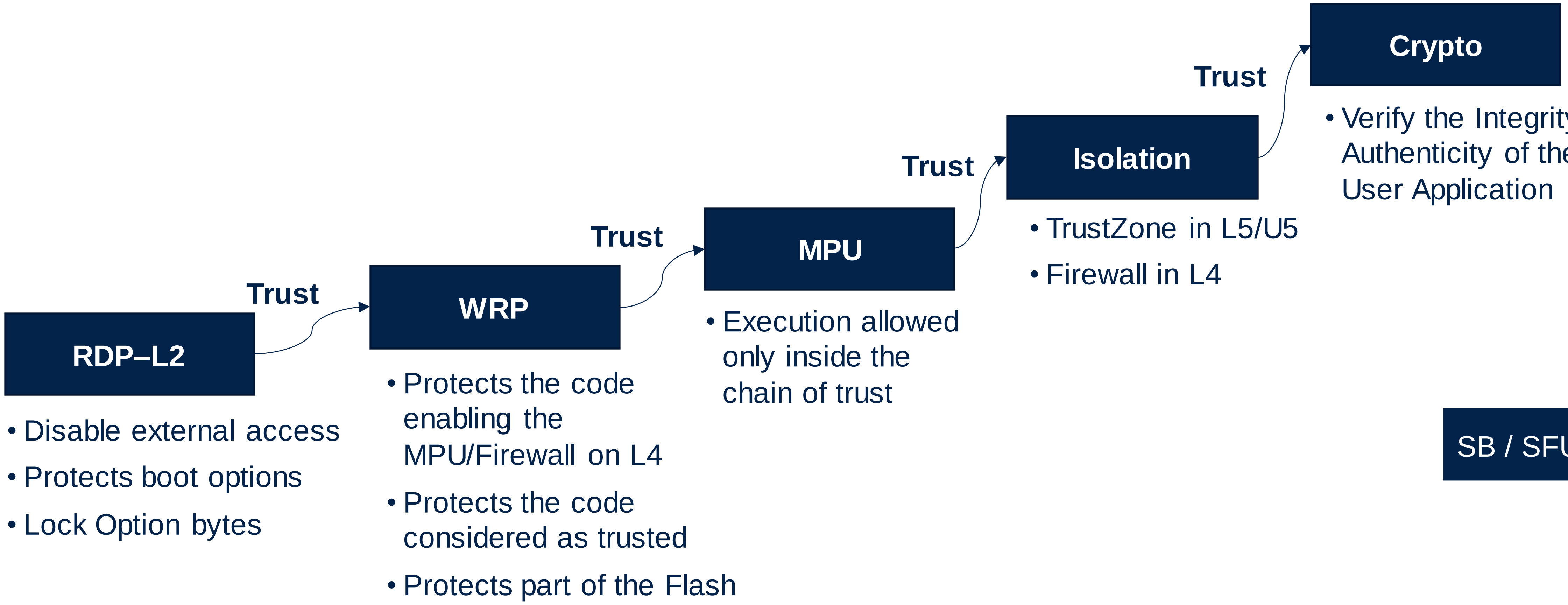


Look at all the elements of the system

Incremental process

life.augmented

Protecting the chain of trust using memory protection assets



RDP–L2

- Disable external access
- Protects boot options
- Lock Option bytes

Trust

WRP

- Protects the code enabling the MPU/Firewall on L4
- Protects the code considered as trusted
- Protects part of the Flash

Trust

MPU

- Execution allowed only inside the chain of trust

Trust

Isolation

- TrustZone in L5/U5
- Firewall in L4

Trust

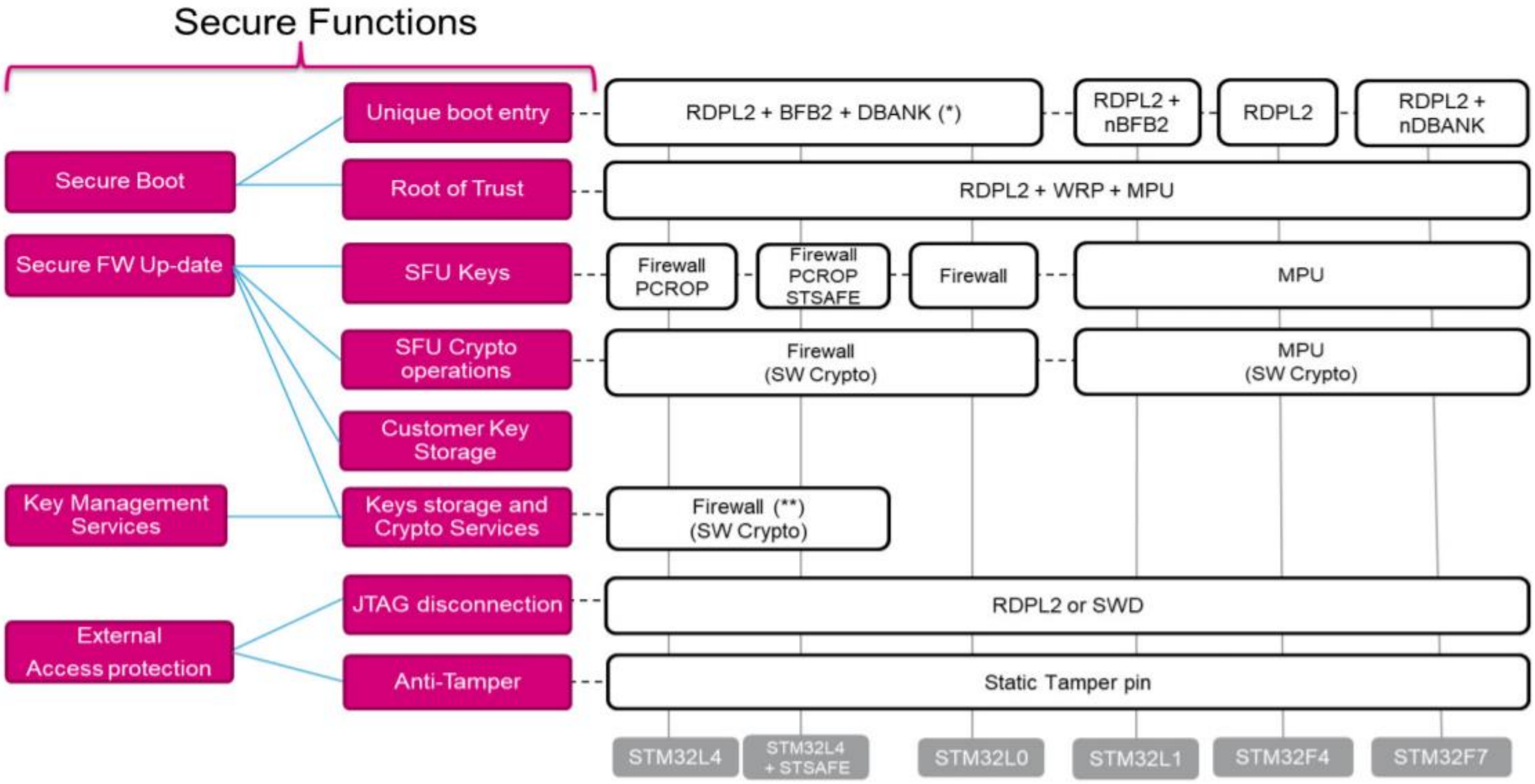
Crypto

- Verify the Integrity, Authenticity of the User Application

SB / SFU

life.augmented

Security implementations 1/2



Secure Functions

Secure Boot

Secure FW Up-date

Key Management Services

External Access protection

Unique boot entry

Root of Trust

SFU Keys

SFU Crypto operations

Customer Key Storage

Keys storage and Crypto Services

JTAG disconnection

Anti-Tamper

RDPL2 + BFB2 + DBANK (*)

RDPL2 + nBFB2

RDPL2

RDPL2 + nDBANK

RDPL2 + WRP + MPU

Firewall PCROP

Firewall PCROP STSAFE

Firewall

MPU

Firewall (SW Crypto)

MPU (SW Crypto)

Firewall (**) (SW Crypto)

RDPL2 or SWD

Static Tamper pin

STM32L4

STM32L4 + STSAFE

STM32L0

STM32L1

STM32F4

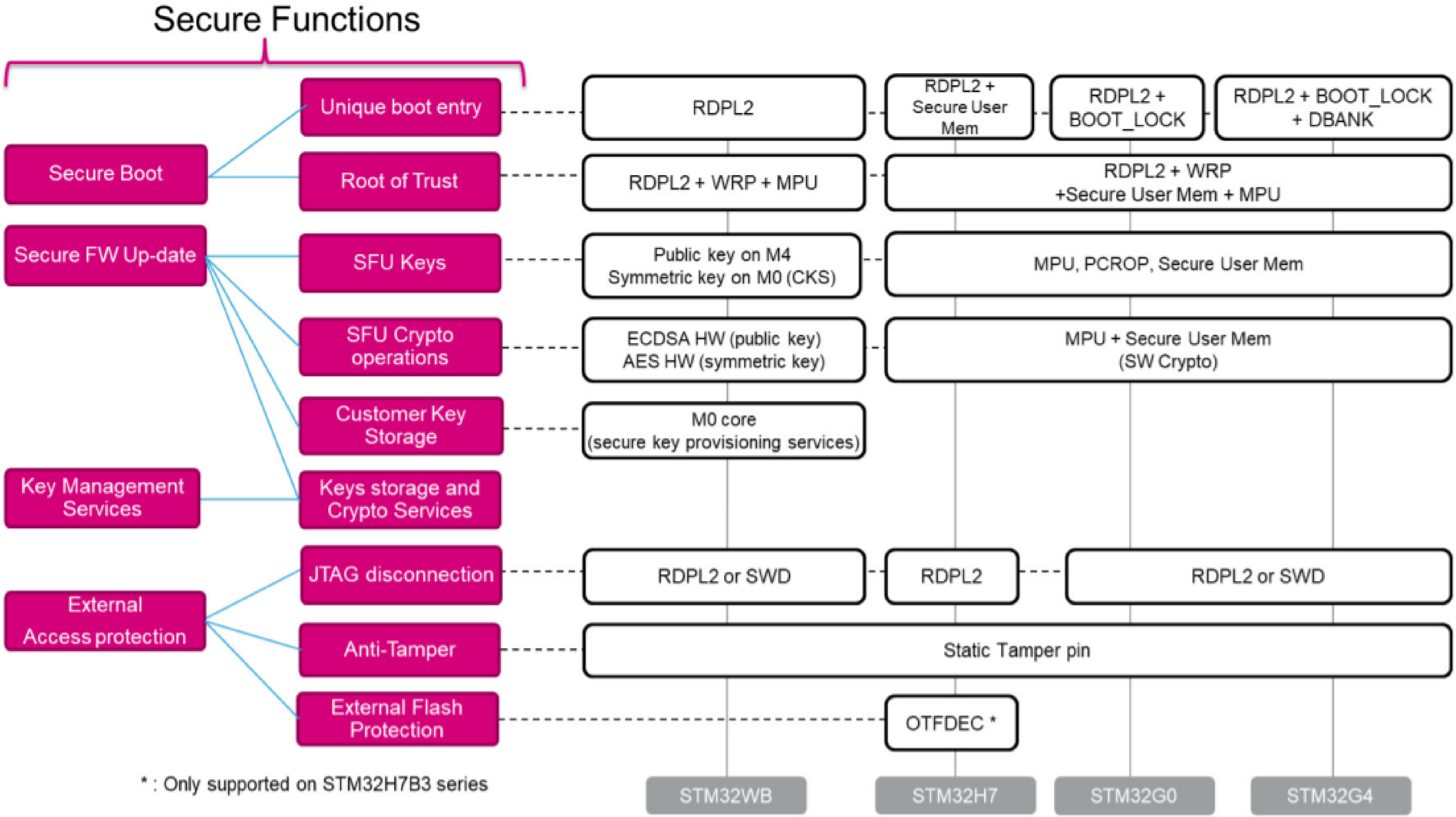
STM32F7

(*) available only on STM32L4S5 series

(**) example provided on B-L475E-IOT01A and B-L4S5I-IOT01A boards

From UM2262 technical document

Security implementations 2/2



Secure Functions

Secure Functions		STM32WB	STM32H7	STM32G0	STM32G4
Secure Boot	Unique boot entry	RDPL2	RDPL2 + Secure User Mem	RDPL2 + BOOT_LOCK	RDPL2 + BOOT_LOCK + DBANK
	Root of Trust	RDPL2 + WRP + MPU	RDPL2 + WRP +Secure User Mem + MPU		
Secure FW Up-date	SFU Keys	Public key on M4 Symmetric key on M0 (CKS)	MPU, PCROP, Secure User Mem		
	SFU Crypto operations	ECDSA HW (public key) AES HW (symmetric key)	MPU + Secure User Mem (SW Crypto)		
	Customer Key Storage	M0 core (secure key provisioning services)			
Key Management Services	Keys storage and Crypto Services				
External Access protection	JTAG disconnection	RDPL2 or SWD	RDPL2	RDPL2 or SWD	
	Anti-Tamper	Static Tamper pin			
	External Flash Protection		OTFDEC *		

* : Only supported on STM32H7B3 series

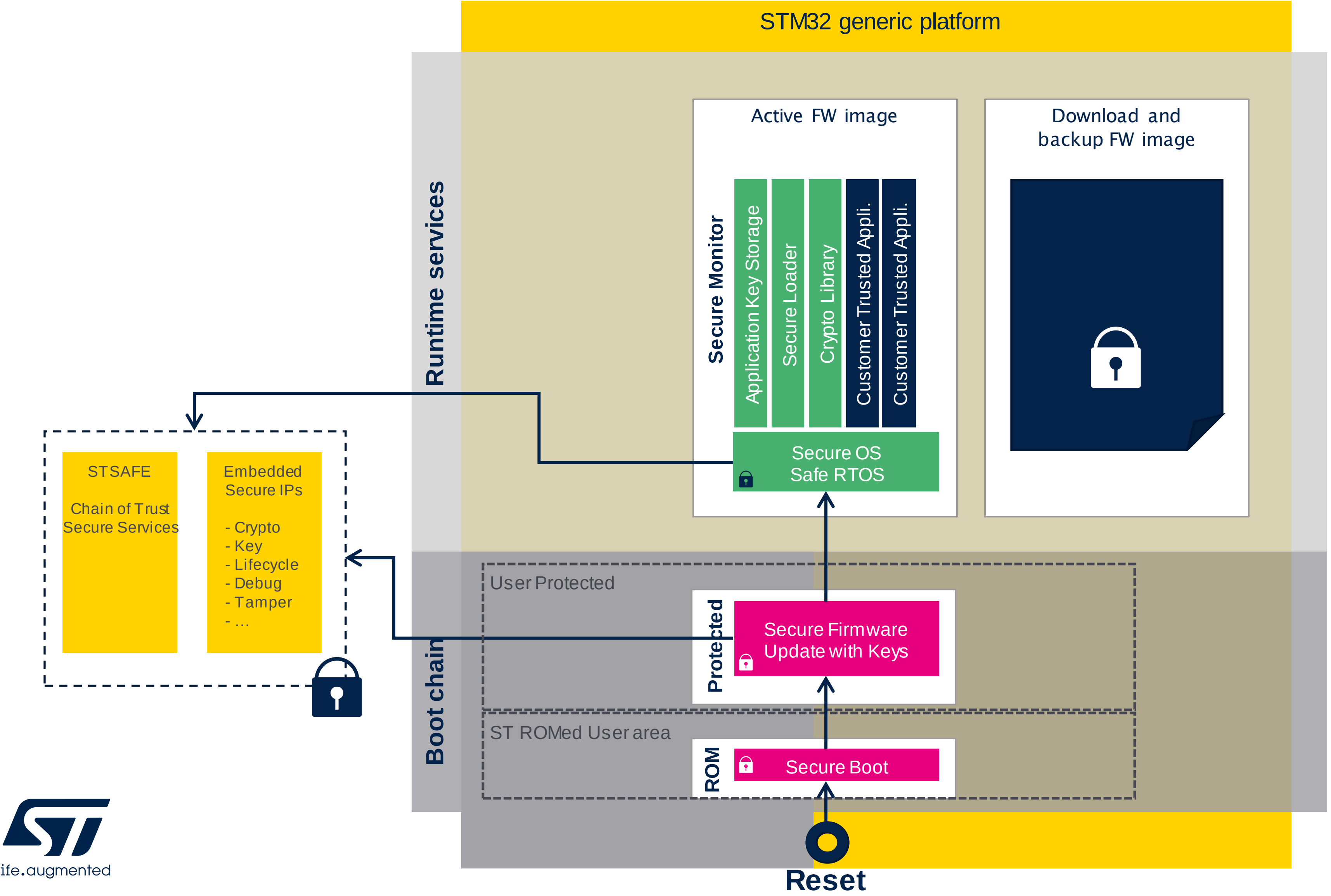
From UM2262 technical document

SBSFU
use cases covered

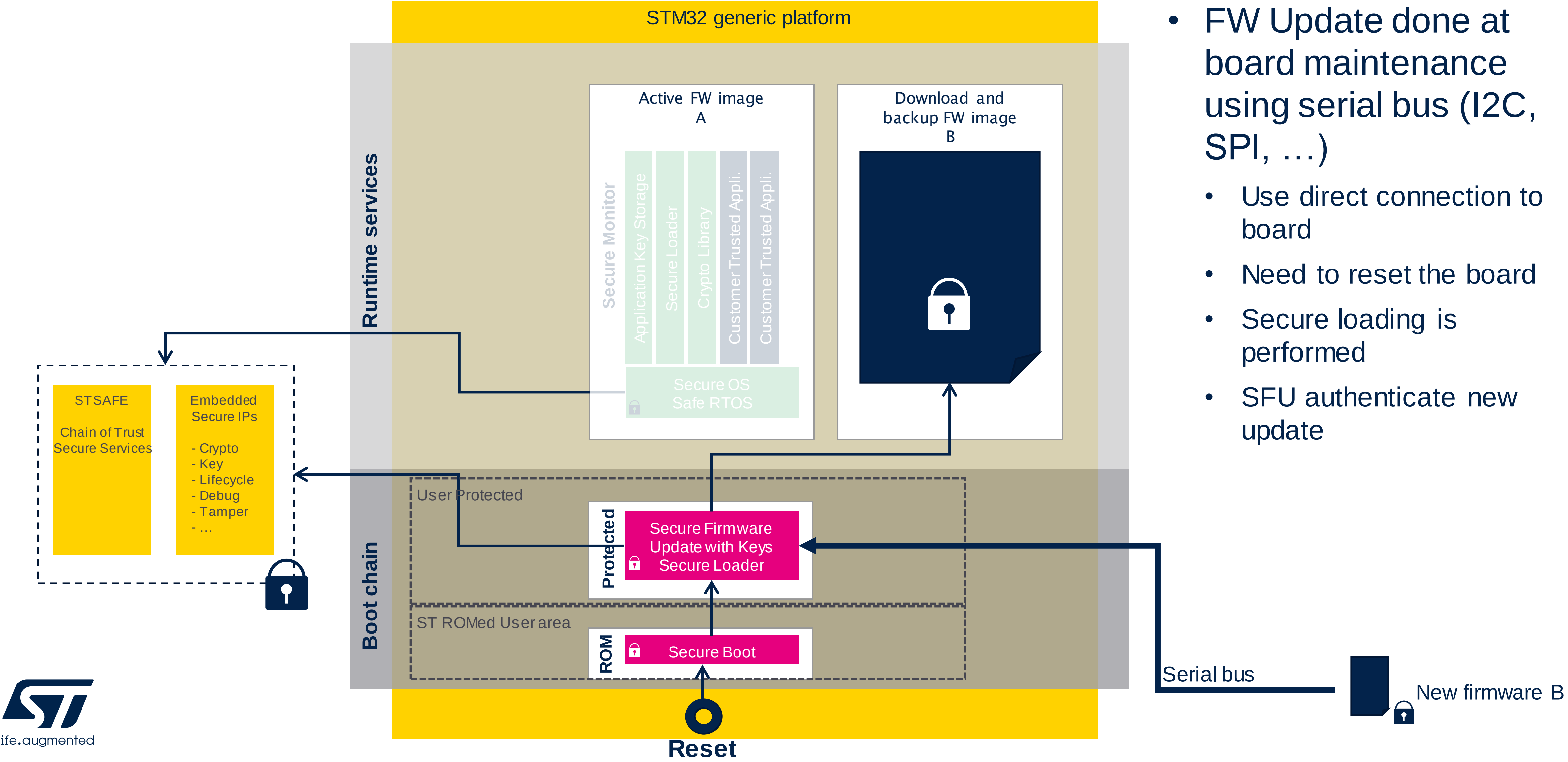
- Manual Firmware Update
  - Usually operated by a human action
  - Use a physical connection between the updater tool and the MCU like
    - UART, SPI, USB… Wired connection
  - Allow to stop the running application during the update
  - In case of update error, retry is manually managed

- Over-The-Air Firmware Update (FOTA)
  - Stand alone update operation
  - Use device connectivity to receive and manage the update
    - Wi-Fi, LPWAN, BT/BLE…
  - Running application shall manage its own firmware update
  - Retry may be difficult to support

SBSFU Flow

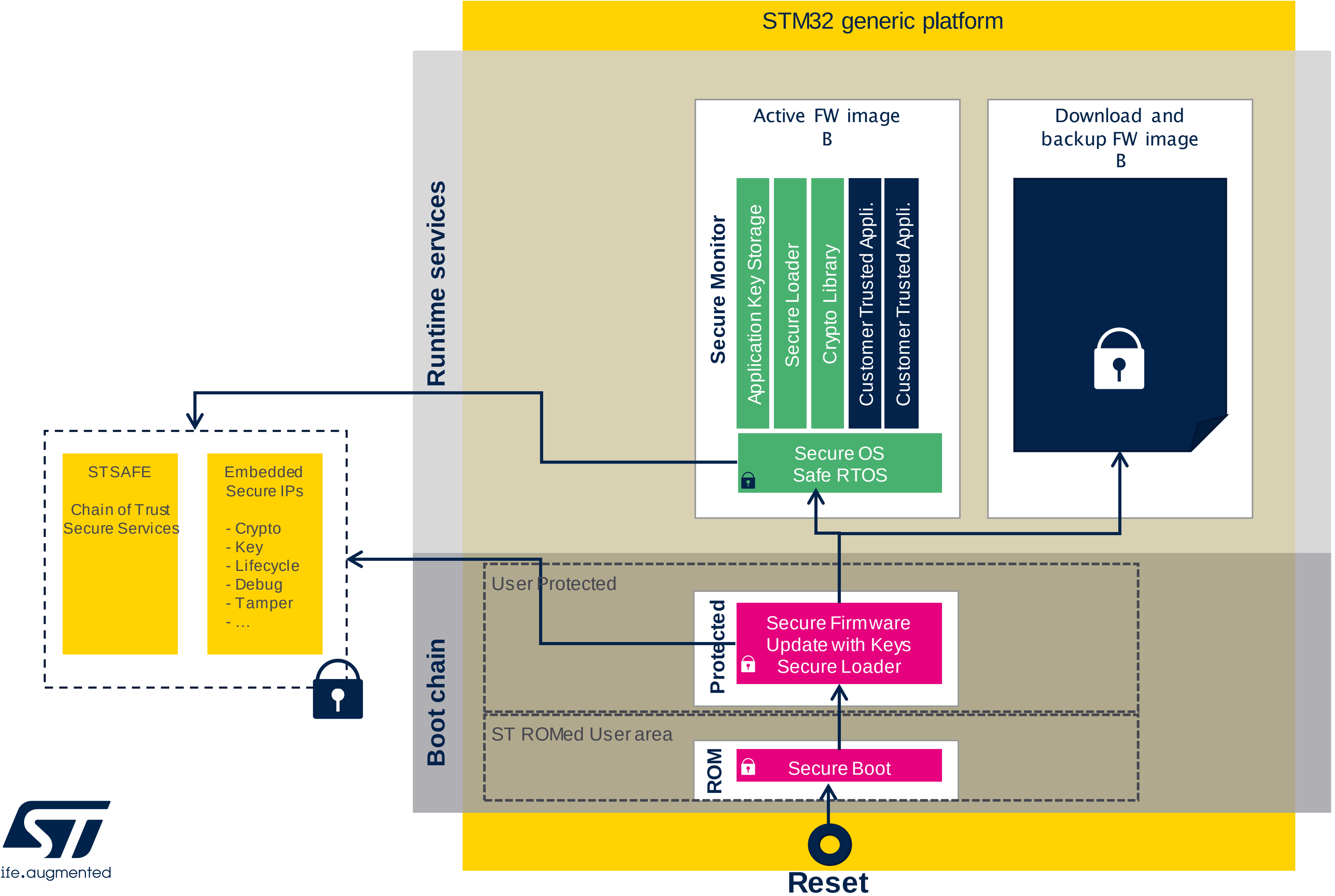

STM32 generic platform
Runtime services
Boot chain
Active FW image
Secure Monitor
Application Key Storage
Secure Loader
Crypto Library
Customer Trusted Appli.
Customer Trusted Appli.
Secure OS
Safe RTOS
Download and
backup FW image
STSAFE
Chain of Trust
Secure Services
Embedded
Secure IPs
- Crypto
- Key
- Lifecycle
- Debug
- Tamper
- …
User Protected
Protected
Secure Firmware
Update with Keys
ST ROMed User area
ROM
Secure Boot
Reset

Use case 1: industrial firmware update



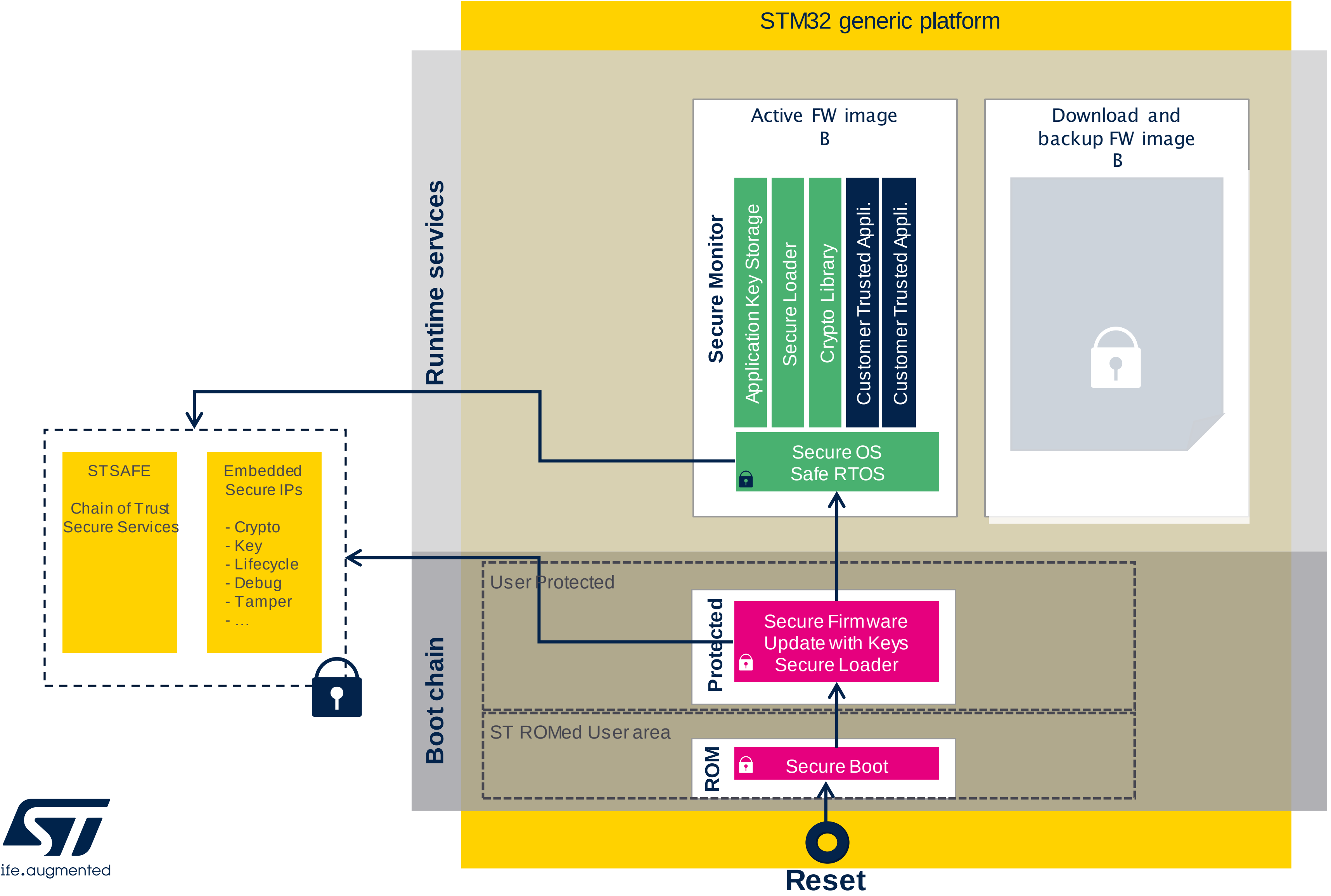
- FW Update done at board maintenance using serial bus (I2C, SPI, …)
  - Use direct connection to board
  - Need to reset the board
  - Secure loading is performed
  - SFU authenticate new update

Use case 1: industrial firmware update



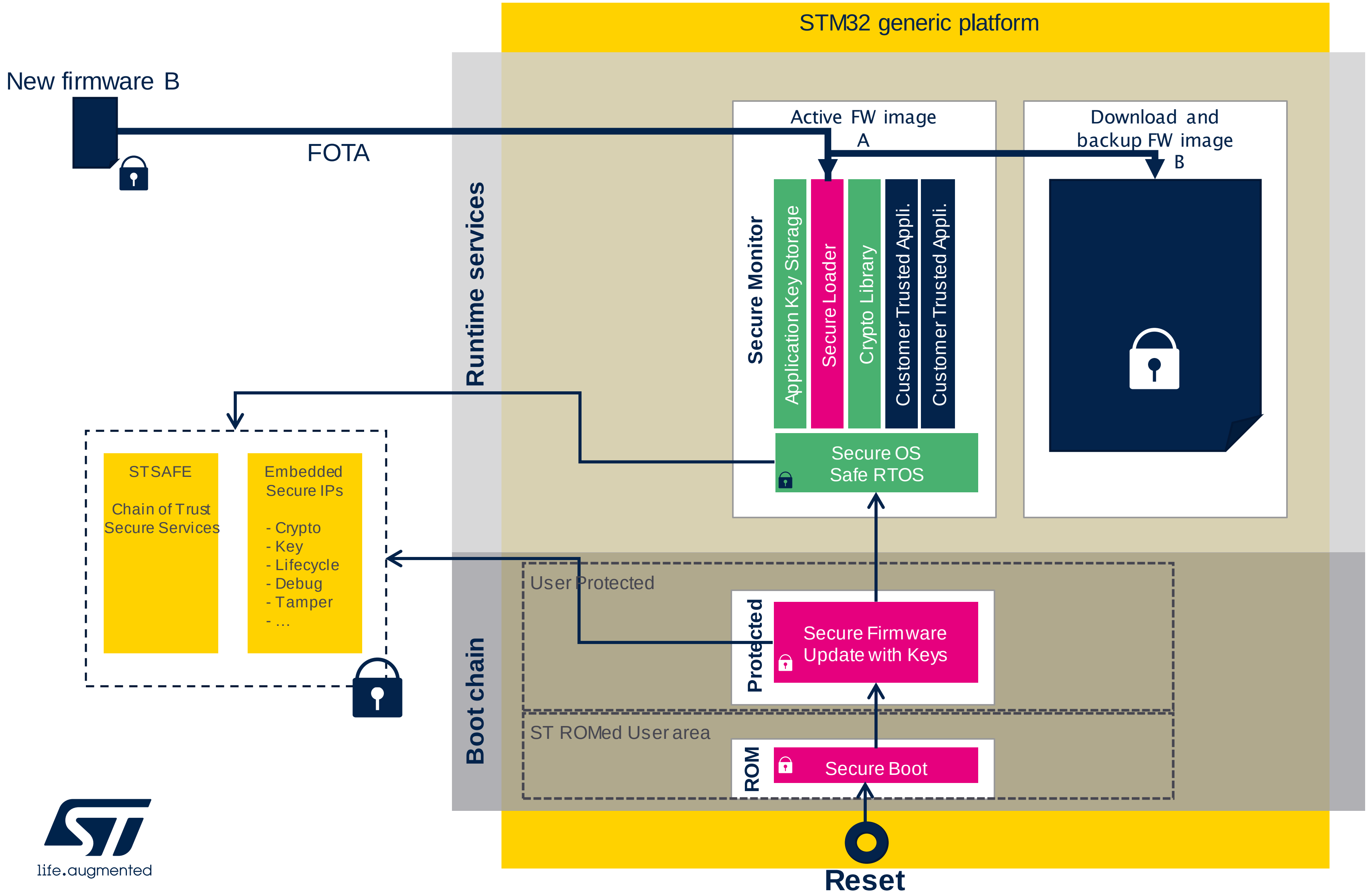
- FW Update done at board maintenance using serial bus (I2C, SPI, …)
  - Use direct connection to board
  - Need to reset the board
  - Secure loading is performed
  - SFU authenticate new update
  - SFU decrypt and flash the new code

Use case 1: industrial firmware update



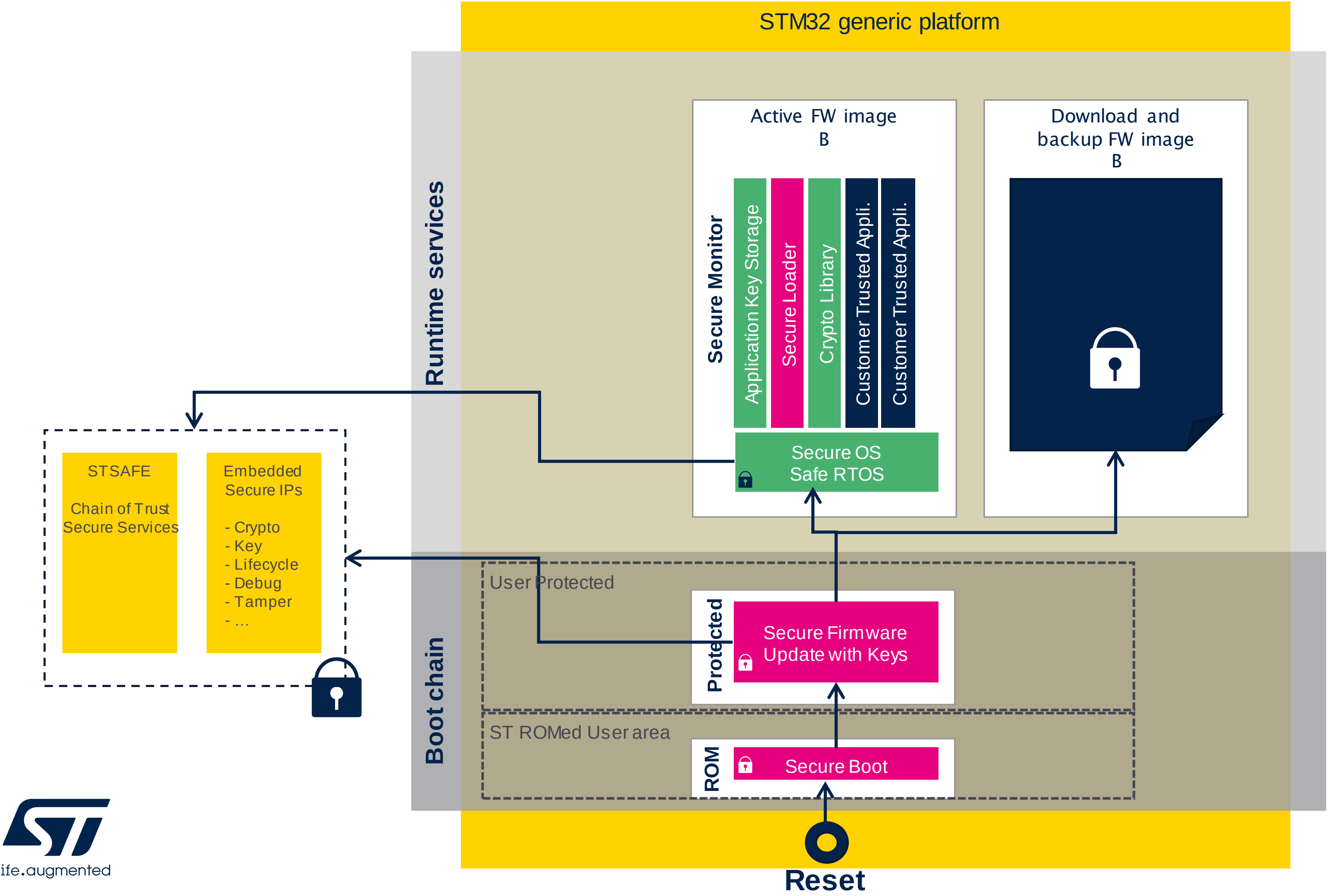
- FW Update done at board maintenance using serial bus (I2C, SPI, …)
  - Use direct connection to board
  - Need to reset the board
  - Secure loading is performed
  - SFU authenticate new update
  - SFU decrypt and flash the new code
  - Application restart after Secure Boot authentication

Use case 2: firmware update over-the-air



- FW update carried out using application connectivity channel
  - Use application connectivity channel
  - Secure loading is performed at runtime

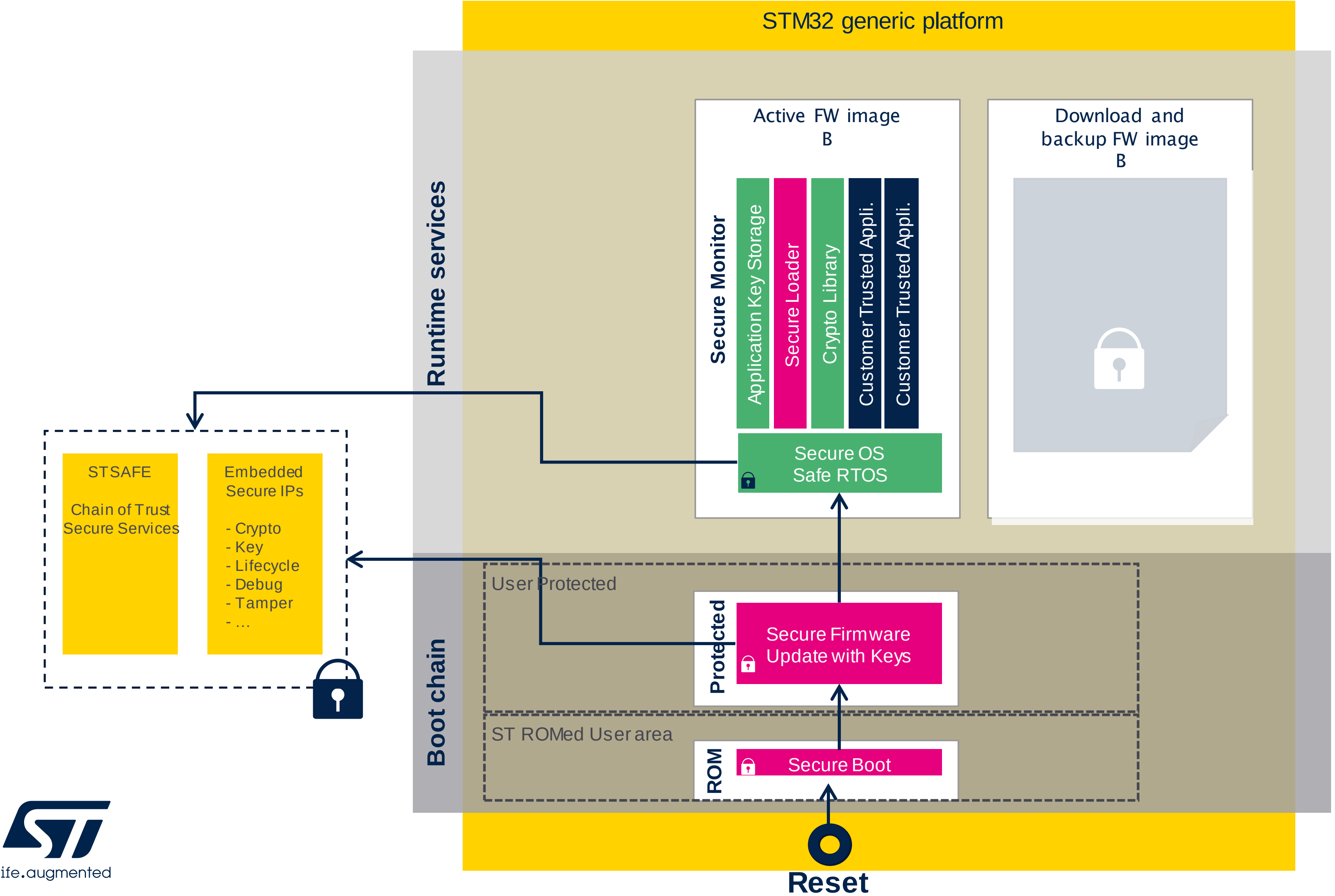
Use case 2: firmware update over-the-air



- FW update carried out using application connectivity channel
  - Use application connectivity channel
  - Secure loading is performed at runtime
  - Need to reset the firmware
  - SFU authenticate new update, decrypt and Flash the new code

life.augmented

Use case 2: firmware update over-the-air



- FW update carried out using application connectivity channel
  - Use application connectivity channel
  - Secure loading is performed at runtime
  - Need to reset the firmware
  - SFU authenticate new update, decrypt and Flash the new code
  - Application restart after Secure Boot authentication

SBSFU: 2 implementations

1. **Modular approach**: SBSFU solution & TF-M
    - Secure Boot module is immutable code
    - Secure Engine is isolated from the rest of the codes
    - Secure Firmware Update includes Root of Trust verification runtime code
    - iROT with SBSFU, uROT with TF-M services on L5/U5
2. **Monolithic approach**: BFU solution
    - Secure Boot and Secure Firmware Updates form a single immutable code protected by a single method: we call it Boot – FW Update
    - It includes cryptographic and SFU key
    - Introduce Root-of-Trust protection mechanism into STM32

life.augmented

SBSFU support on STM32
Modular approach

	F4	F7	H7 dual	H7 single	L0	L1	L4 / L4+	U5 / L5*	G0	G4	WB
X-CUBE-SBSFU STM32Cube for L5/U5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ (M4)
FW Update											
FW Update key and crypto	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ (M4)
Engine for User Application	✓	✓			✓	✓	✓	✓			
Key Storage code							✓	✓			
Key Storage (volatile data storage)							✓	✓			
Secure Key management PKCS#11							✓				
STSAFE lib							✓				
Customer Key Storage											✓ (Sec-M0)

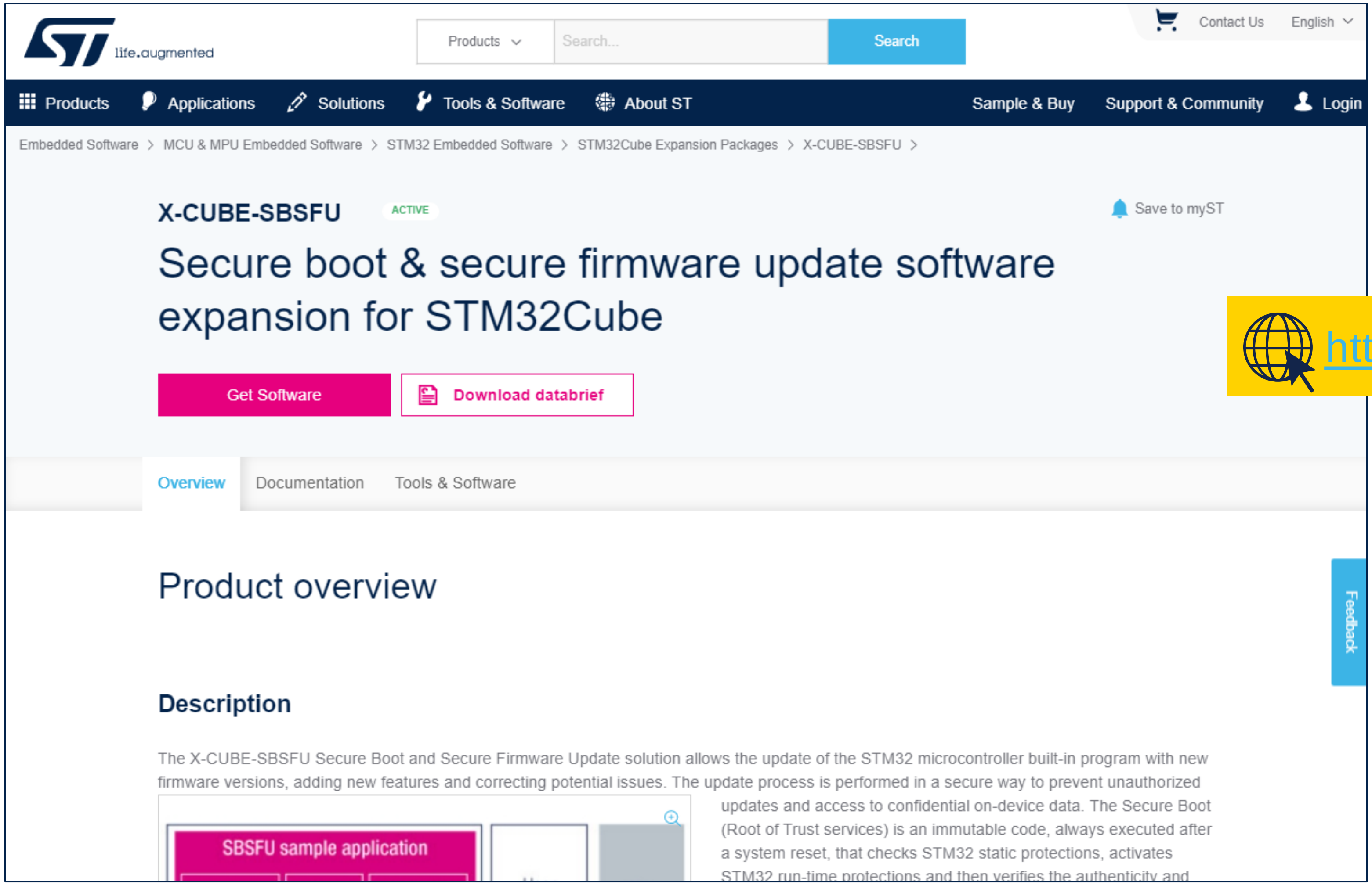
Note: *Using ARM TrustZone

life.augmented

BFU support on STM32
Monolithic approach

	F4	F7	H7 dual	H7 single	L0	L1	L4 / L4+	U5 / L5 With TZ	G0	G4	WB
BFU: SW Modules source code without HW memory protection											
Boot	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FW Update	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Crypto Engine	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Key Storage	✓	✓	✓	✓			✓	✓		✓	✓

X-CUBE-SBSFU package on the web



Embedded Software > MCU & MPU Embedded Software > STM32 Embedded Software > STM32Cube Expansion Packages > X-CUBE-SBSFU >

X-CUBE-SBSFU ACTIVE

Secure boot & secure firmware update software expansion for STM32Cube

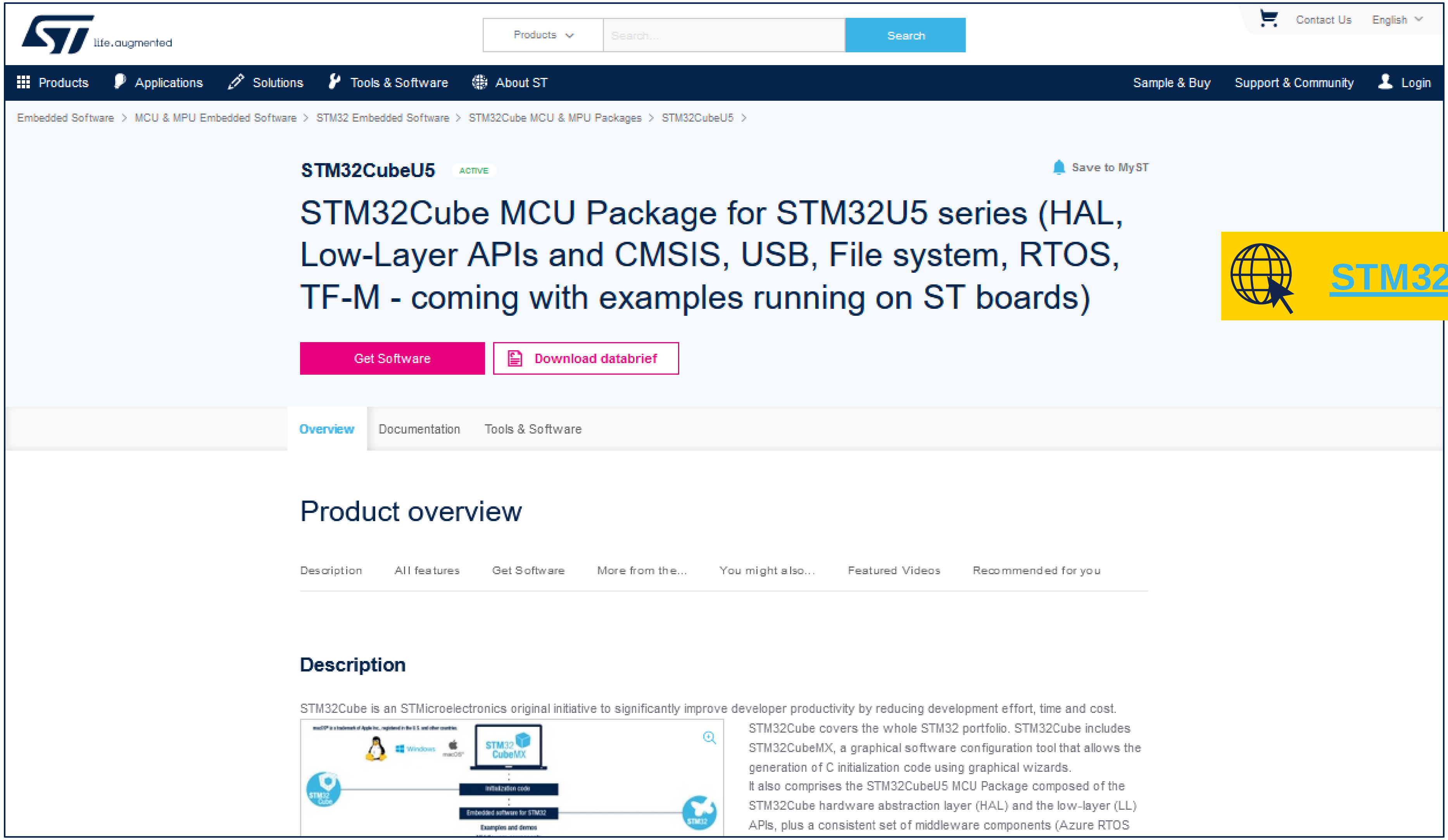
Product overview

Description

The X-CUBE-SBSFU Secure Boot and Secure Firmware Update solution allows the update of the STM32 microcontroller built-in program with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way to prevent unauthorized updates and access to confidential on-device data. The Secure Boot (Root of Trust services) is an immutable code, always executed after a system reset, that checks STM32 static protections, activates STM32 run-time protections and then verifies the authenticity and

http://www.st.com/x-cube-sbsfu

STM32Cube MCU package on the web



Products | Search... | Search

Contact Us English

Products | Applications | Solutions | Tools & Software | About ST | Sample & Buy | Support & Community | Login

Embedded Software > MCU & MPU Embedded Software > STM32 Embedded Software > STM32Cube MCU & MPU Packages > STM32CubeU5 >

STM32CubeU5 ACTIVE

Save to MyST

STM32Cube MCU Package for STM32U5 series (HAL, Low-Layer APIs and CMSIS, USB, File system, RTOS, TF-M - coming with examples running on ST boards)

Get Software | Download databrief

Overview | Documentation | Tools & Software

Product overview

Description | All features | Get Software | More from the... | You might also... | Featured Videos | Recommended for you

Description

STM32Cube is an STMicroelectronics original initiative to significantly improve developer productivity by reducing development effort, time and cost.

STM32Cube covers the whole STM32 portfolio. STM32Cube includes STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.

It also comprises the STM32CubeU5 MCU Package composed of the STM32Cube hardware abstraction layer (HAL) and the low-layer (LL) APIs, plus a consistent set of middleware components (Azure RTOS

STM32CubeU5

Recommendations

- Reduce risk
  - Design products protected against attacks within their whole life cycle
- Understand the value of your assets
  - Perform threat analysis
  - Confidentially, availability and integrity are key
- Apply best security practices to develop and maintain secure products
  - Use security features and tools to achieve robust products
  - Work with trusted and experienced partners
- Visit st.com for more information on X-CUBE-SBSFU and for updates on STM32CubeU5

Our technology starts with You

life.augmented