BENINGO EMBEDDED GROUP

Secure Bootloader Design Techniques for MCU's

life.augmented

Session Overview

Objective:

- Explore and demonstrate secure bootloader design techniques

Topics:

- Embedded System Security Attacks
- Secure Bootloader Features
- STM32 X-CUBE-SBSFU
- Secure Bootloader Implementation
- SBSFU Demonstration
- Best practices for secure bootloader design

The Lecturer

Jacob Beningo

Principal Consultant

Social Media / Contact

- E : jacob@beningo.com
- T : 810-844-1522
- : Jacob_Beningo
- : Beningo Engineering
- : JacobBeningo
- EDN : Embedded Basics
- ARM Connected Community

Newsletters

- Embedded Bytes

http://bit.ly/1BAHYXm

Consulting

- Secure Bootloaders
- Code Reviews
- Architecture Design
- Real-time Software
- Expert Firmware Analysis
- Microcontroller Systems

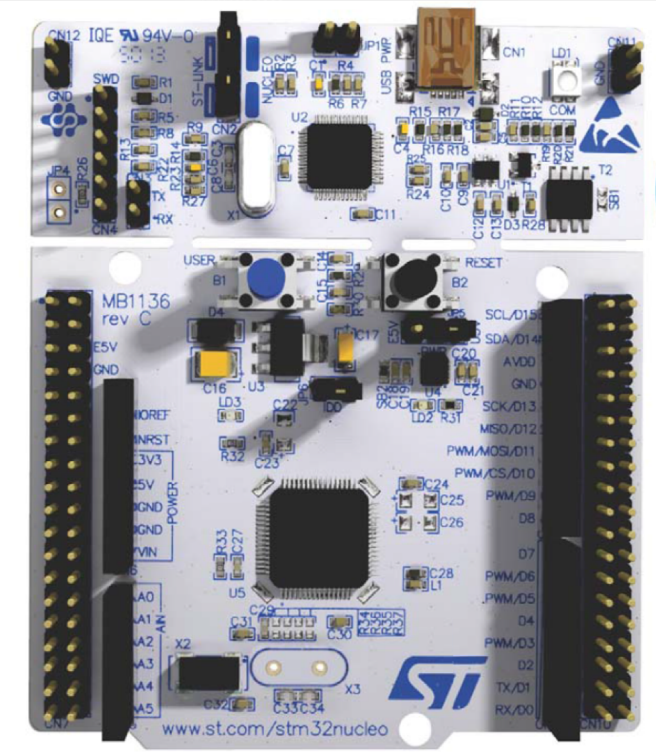
Embedded Training

- RTOS Workshop
- Bootloader Design
- Debugging Techniques
- Security Fundamentals
- Micro Python

www.beningo.com

Hands-on Example Materials

NUCLEO-L476RG



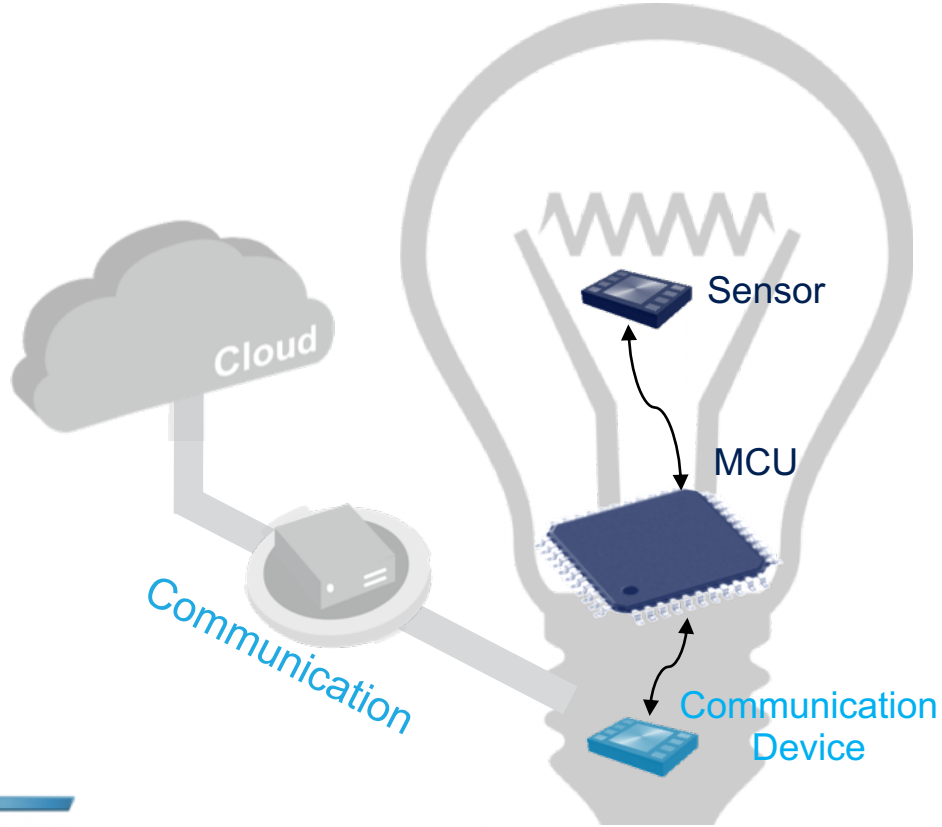
arm KEIL

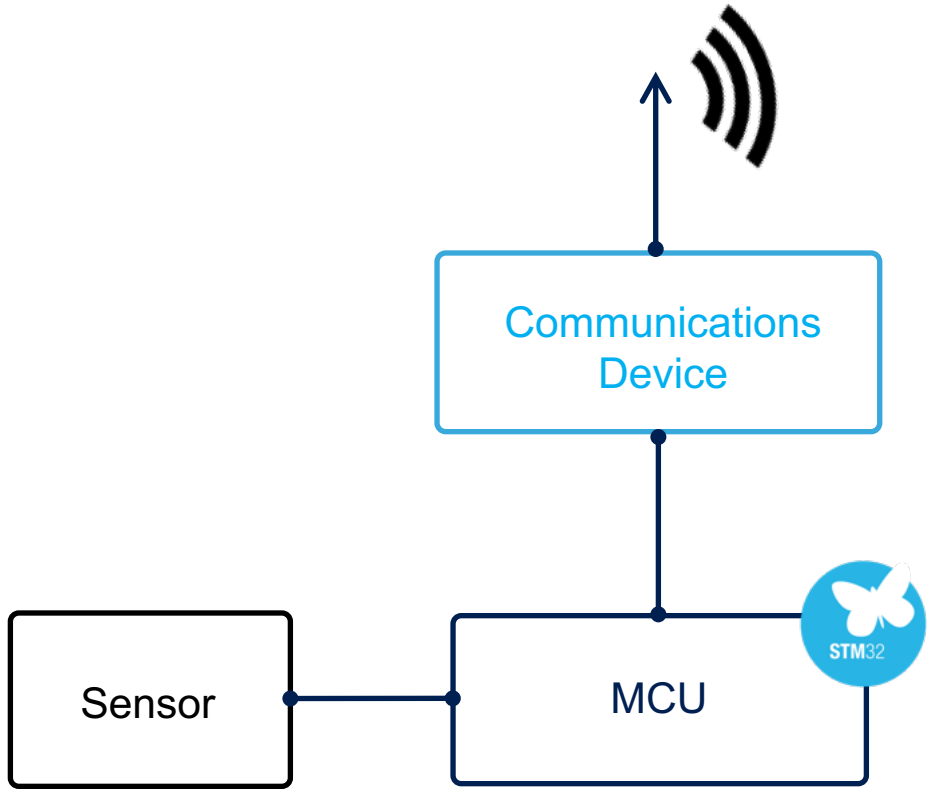
µVision® 5

Integrated Development Environment

ARMKEIL
Microcontroller tools

A Simple IoT Device





life.augmented

Attack Categories

Logical attack

- Remote Exploit of software bugs and open ports

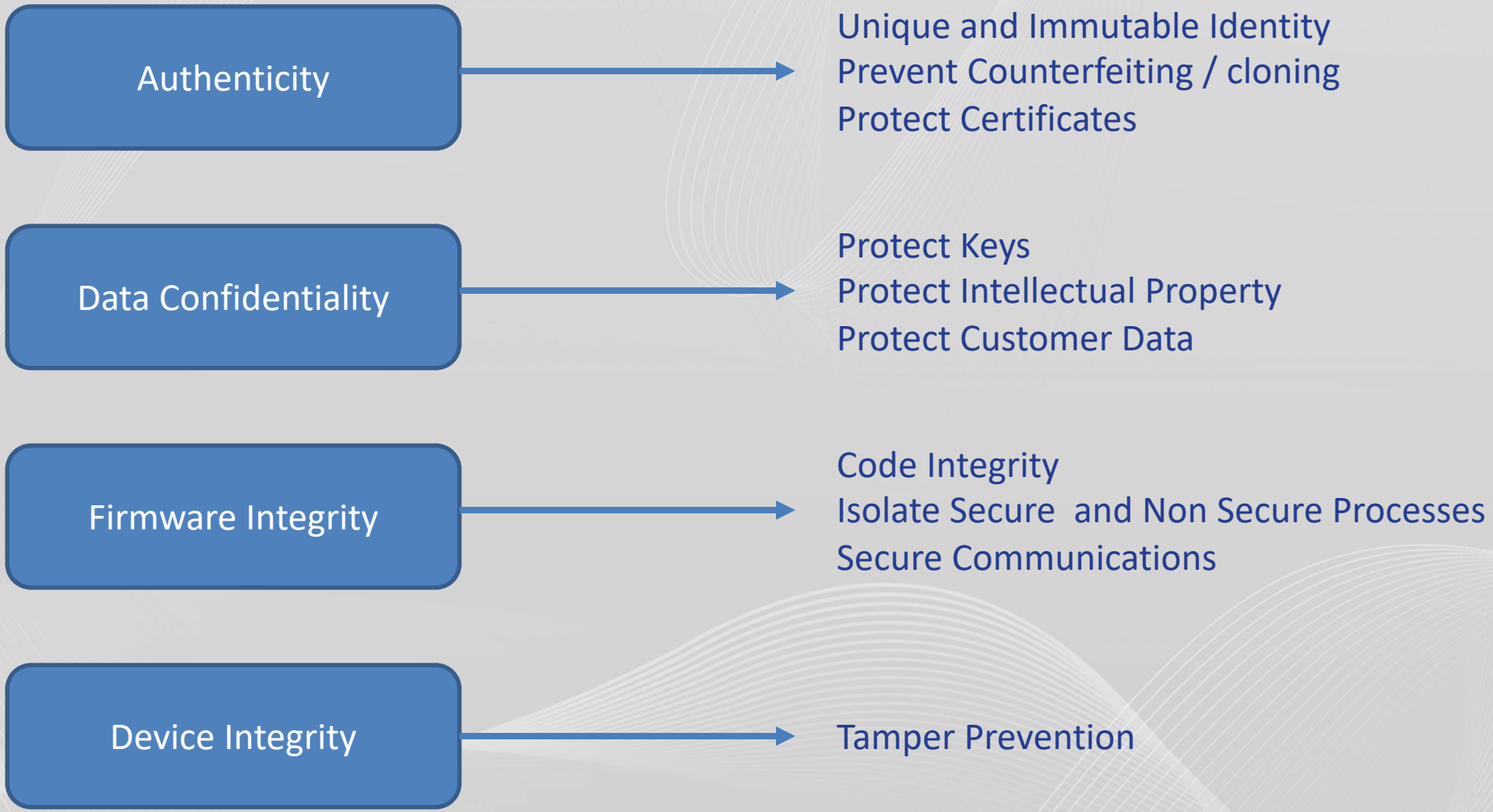
Board level attack

- Debug ports, physical access to I/F
- Side Channel attack (SPA/DPA, Profiling)
- Timing attack, faults injection

Chip level attack

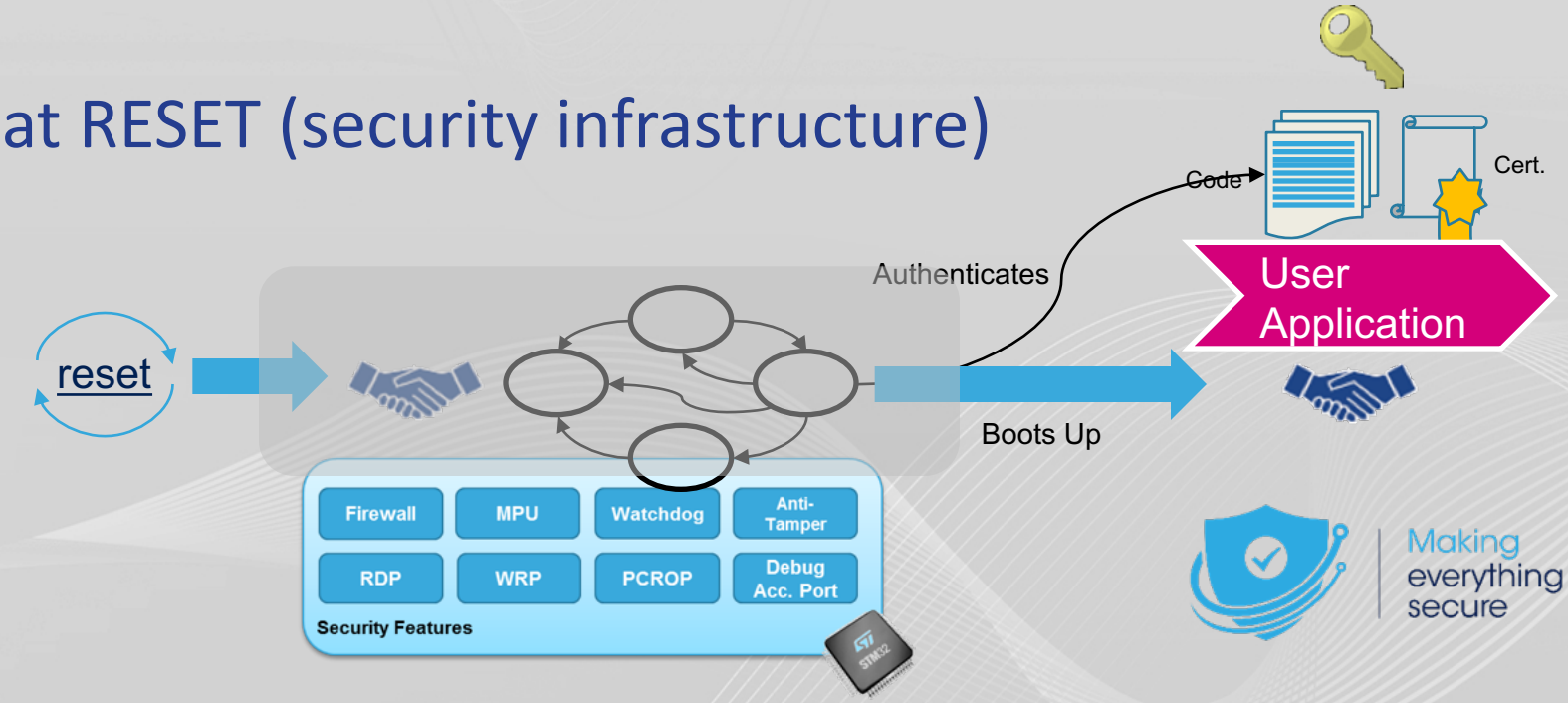
- FIB (cutting and rewiring signals)
- Physical delayering, reverse-engineering

What features does a secure bootloader have?



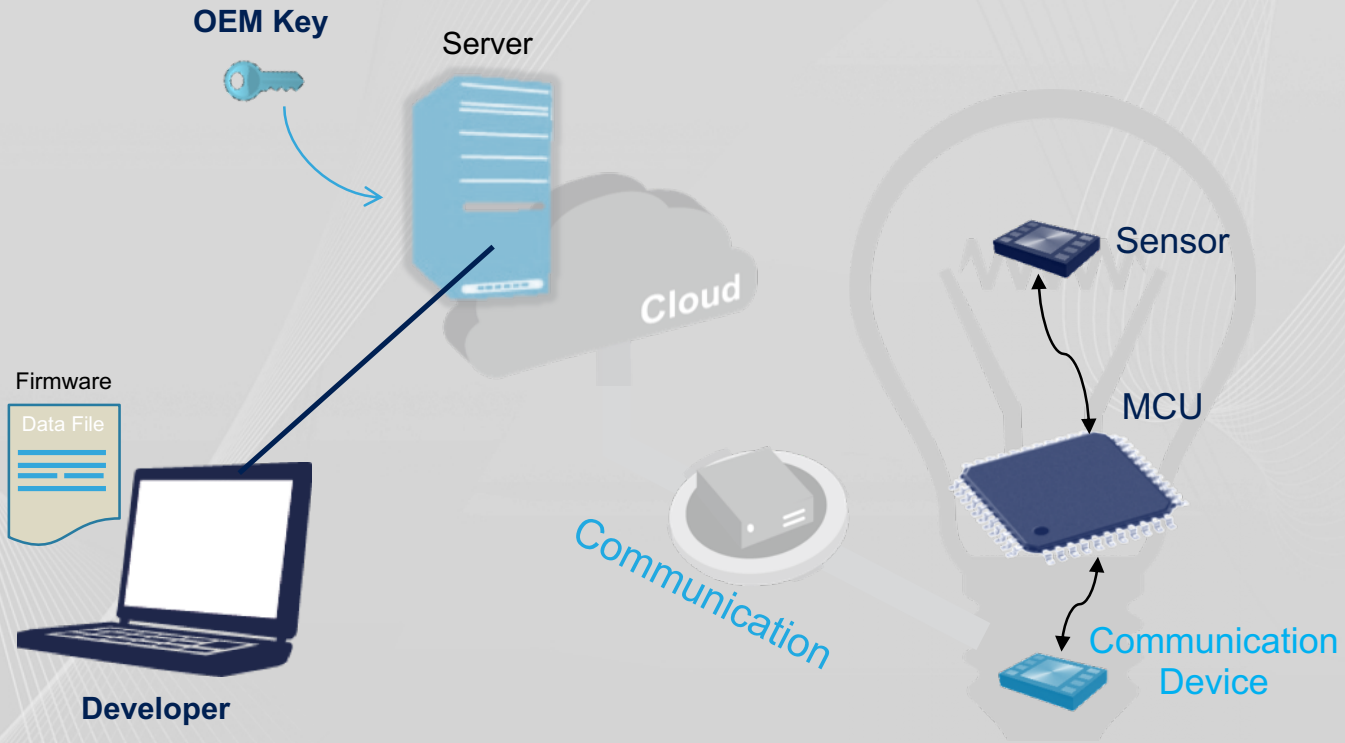
Secure Boot

- Secure Boot is about **ensuring a Chain of Trust** established and maintained throughout the runtime execution of code on the product.
- Secure Boot is used as a **Root of Trust** using cryptographic functions to confirm the **authenticity and integrity of the user firmware** before allowing it to run
  - Unique Entry point at RESET (security infrastructure)
  - Immutable code
  - Authentication
  - Integrity

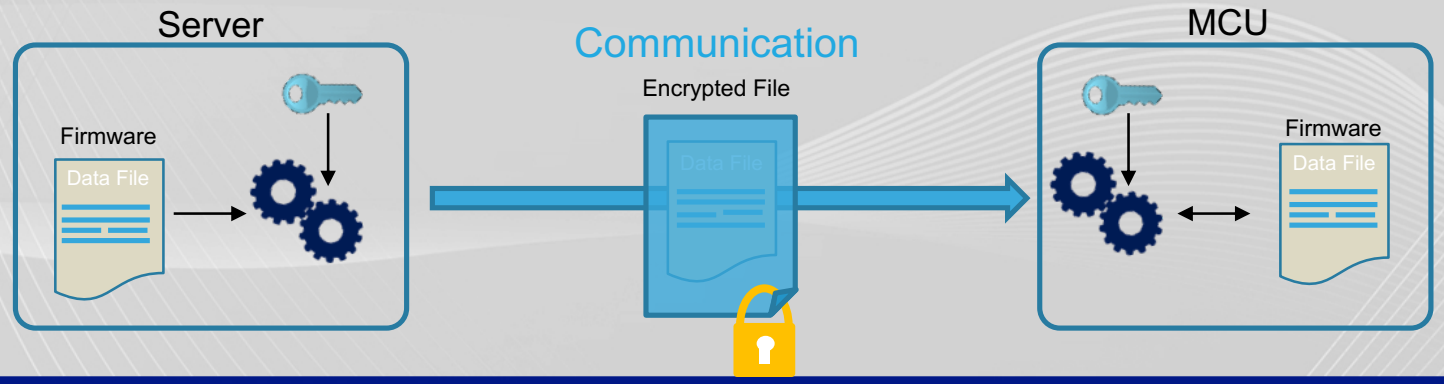


Making everything secure

Secure Firmware Update



- Server sends FW Package
- Device receives, checks/stores/installs new FW package and executes it



Making everything secure

X-CUBE-SBSFU in a Nutshell

- X-CUBE-SBSFU is provided as **reference code** **to demonstrate state-of-the-art usage of the STM32 security protection mechanisms**. It is a **starting point** for OEMs to develop their own Secure Boot and Secure Firmware Update applications as a function of their product security requirement levels.
  - The SBSFU application is **an example** illustrating how this can be achieved
    - This is only one item of the package
  - **Security never comes for free**
    - SBSFU is **NOT** an off-the-shelf secure bootloader with secure firmware update capability
    - Customers to use it as an example to understand how to leverage the STM32 assets
    - Customers must analyze their system and build their own solution **under their own responsibility**

X-CUBE-SBSFU Features Overview

- Secure Boot (Root of Trust):
  - Activate and Check right secure mechanisms of STM32 platform to protect critical operation and secret from attacks
  - Check Authentication and Integrity of User Application before execution
- New (Encrypted) Firmware download via USART Virtual com
- FW installation management:
  - Detect new (Encrypted) Firmware version to install
    - From local download service
    - Pre-downloaded OTA via User Application from previous execution
  - Manage Firmware version (check unauthorized updates or unauthorized installation)
  - Secure Firmware Update:
    - Firmware Authentication and Integrity check
    - Firmware Decryption
    - Firmware Installation
  - In case of any error occurring during new image installation rollback to the previous valid Firmware version
  - Execute new installed Firmware (once Authenticated and Integrity checked)

X-CUBE-SBSFU Features Overview

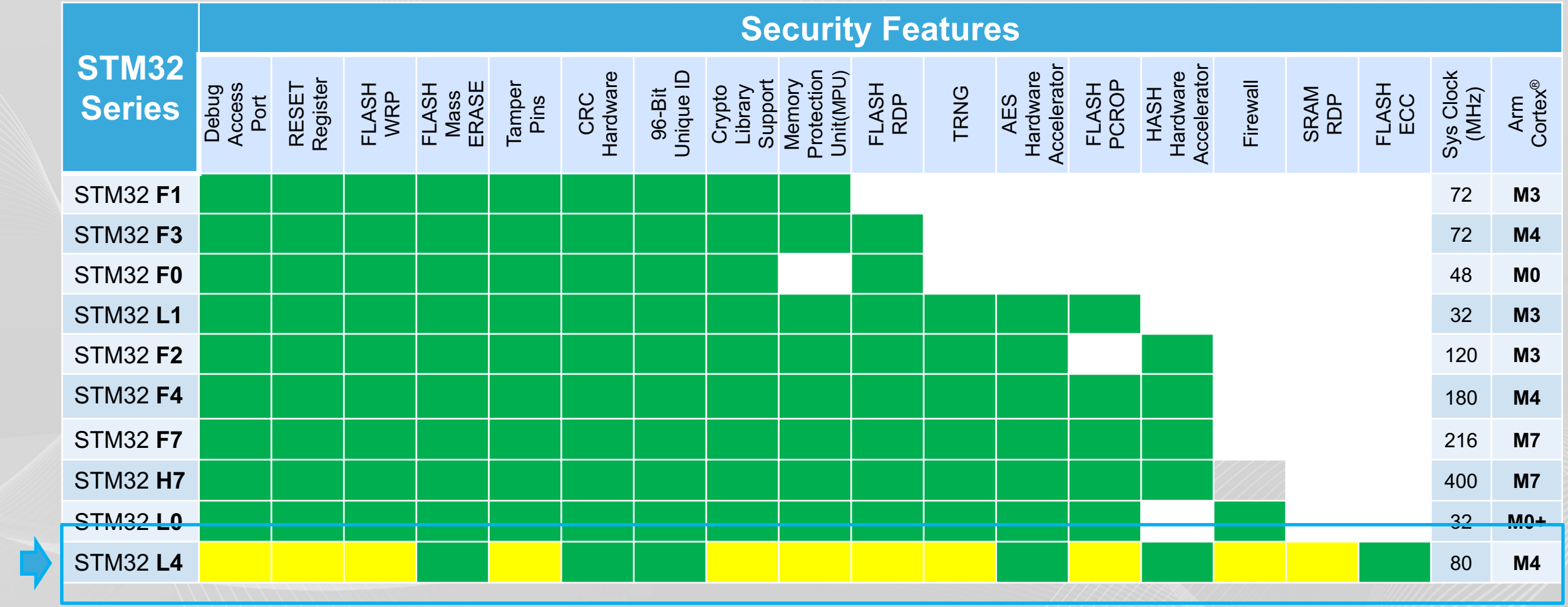
- 3 cryptographic schemes are provided

Table 3. Cryptographic scheme comparison

Features	Asymmetric with AES encryption	Asymmetric without encryption	Symmetric (AES GCM)
Confidentiality	AES CBC encryption (FW binary)	None: the user FW is in 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 key (secret) Public ECDSA key	Public ECDSA key	Private AES GCM key (secret)

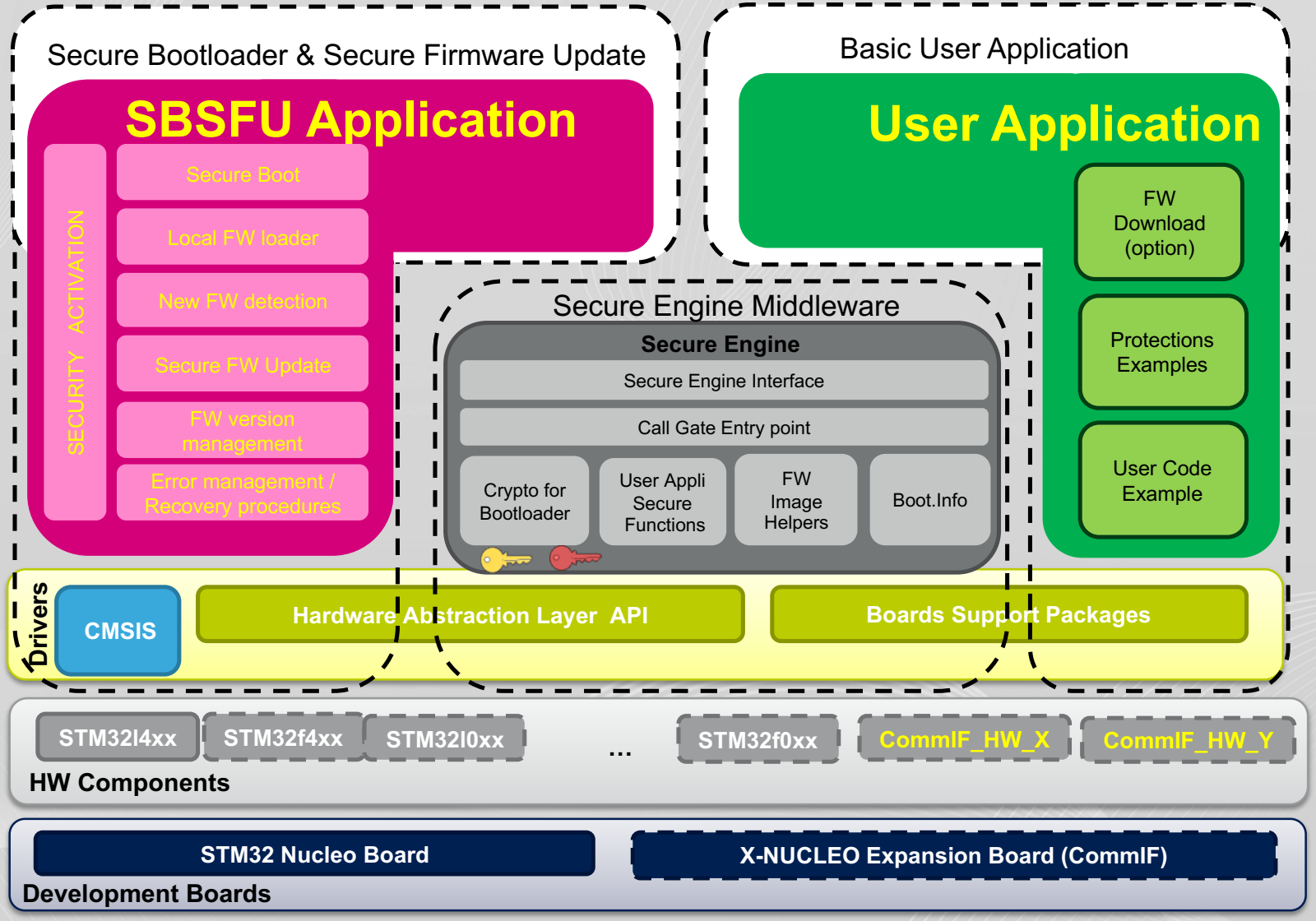
- The X-CUBE-SBSFU Architecture allows switching from one scheme to another via compiler switch.

STM32 Security Features

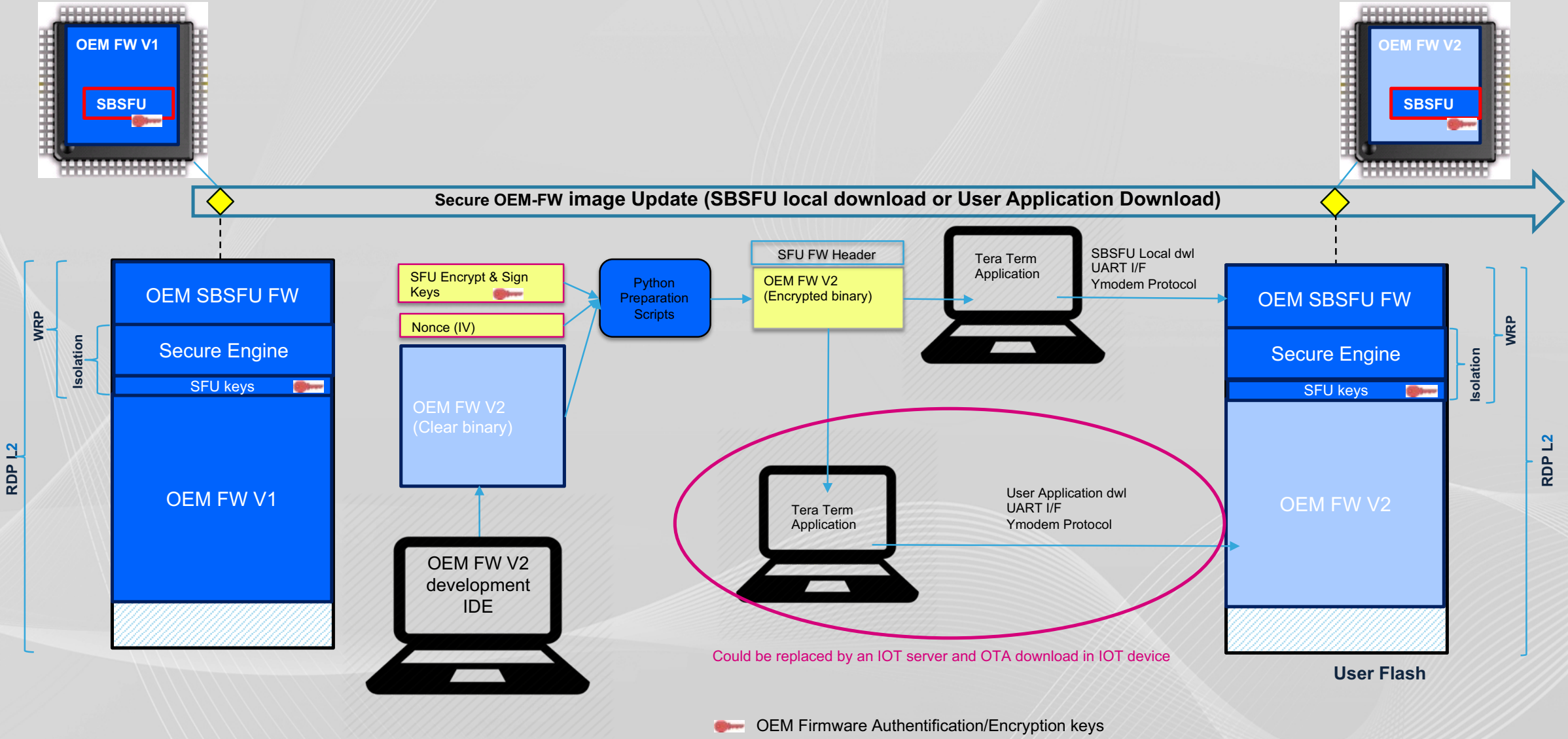


STM32 Series	Security Features																		
	Debug Access Port	RESET Register	FLASH WRP	FLASH Mass ERASE	Tamper Pins	CRC Hardware	96-Bit Unique ID	Crypto Library Support	Memory Protection Unit(MPU)	FLASH RDP	TRNG	AES Hardware Accelerator	FLASH PCROP	HASH Hardware Accelerator	Firewall	SRAM RDP	FLASH ECC	Sys Clock (MHz)	Arm Cortex®
STM32 **F1**	■	■	■	■	■	■	■	■	■									72	**M3**
STM32 **F3**	■	■	■	■	■	■	■	■	■	■								72	**M4**
STM32 **F0**	■	■	■	■	■	■	■	■		■								48	**M0**
STM32 **L1**	■	■	■	■	■	■	■	■	■	■	■	■	■					32	**M3**
STM32 **F2**	■	■	■	■	■	■	■	■	■	■	■	■		■				120	**M3**
STM32 **F4**	■	■	■	■	■	■	■	■	■	■	■	■	■	■				180	**M4**
STM32 **F7**	■	■	■	■	■	■	■	■	■	■	■	■	■	■				216	**M7**
STM32 **H7**	■	■	■	■	■	■	■	■	■	■	■	■	■	■				400	**M7**
STM32 **L0**	■	■	■	■	■	■	■	■	■	■	■	■	■		■			32	**M0+**
STM32 **L4**	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	80	**M4**

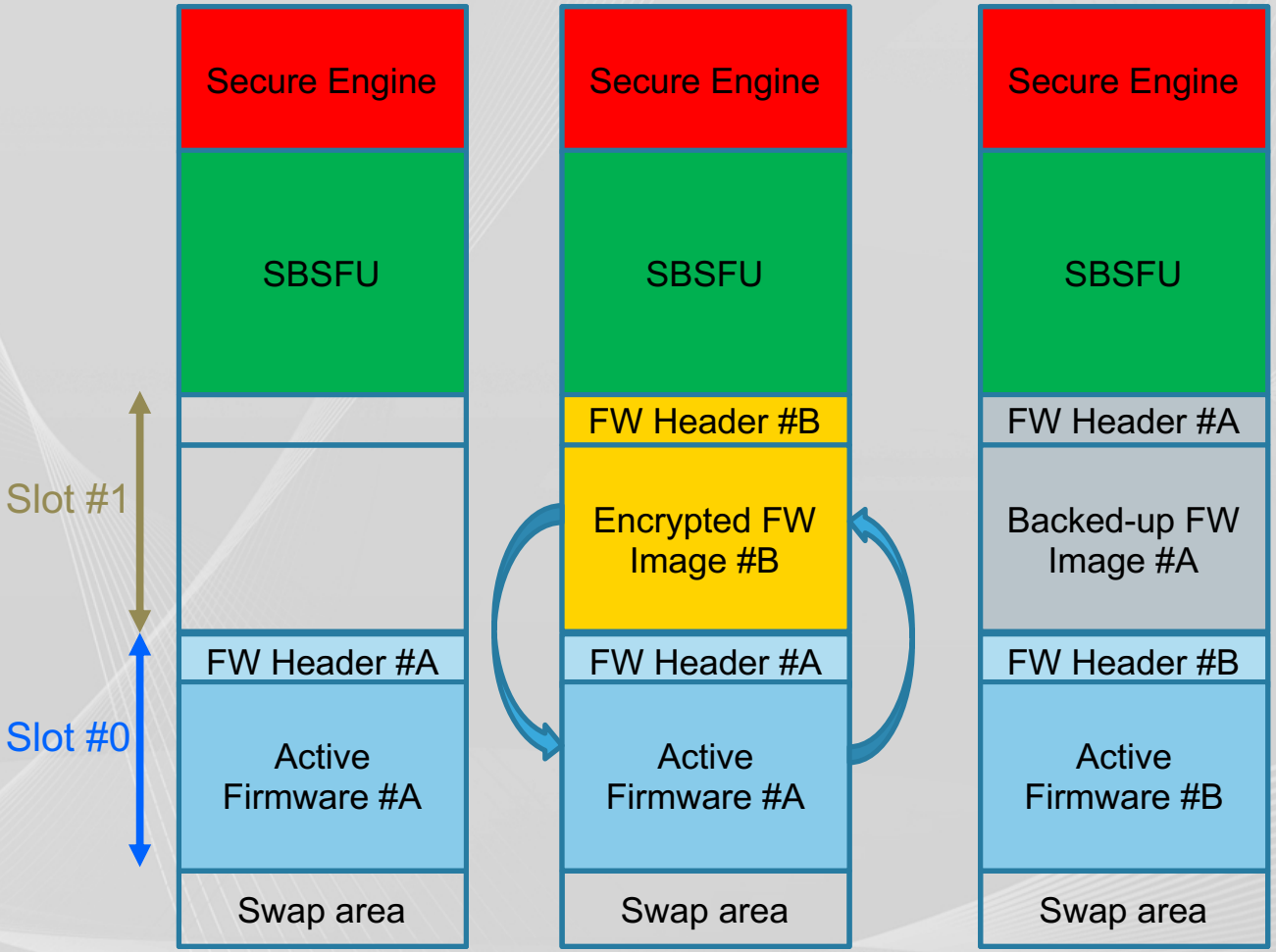
Package Architecture Overview



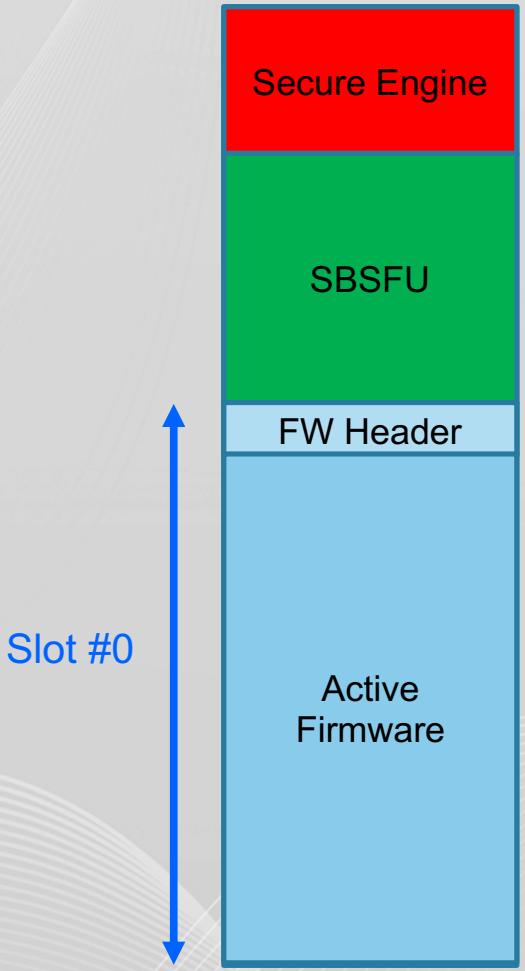
X-CUBE-SBSFU Ecosystem Overview



Firmware Image Programming



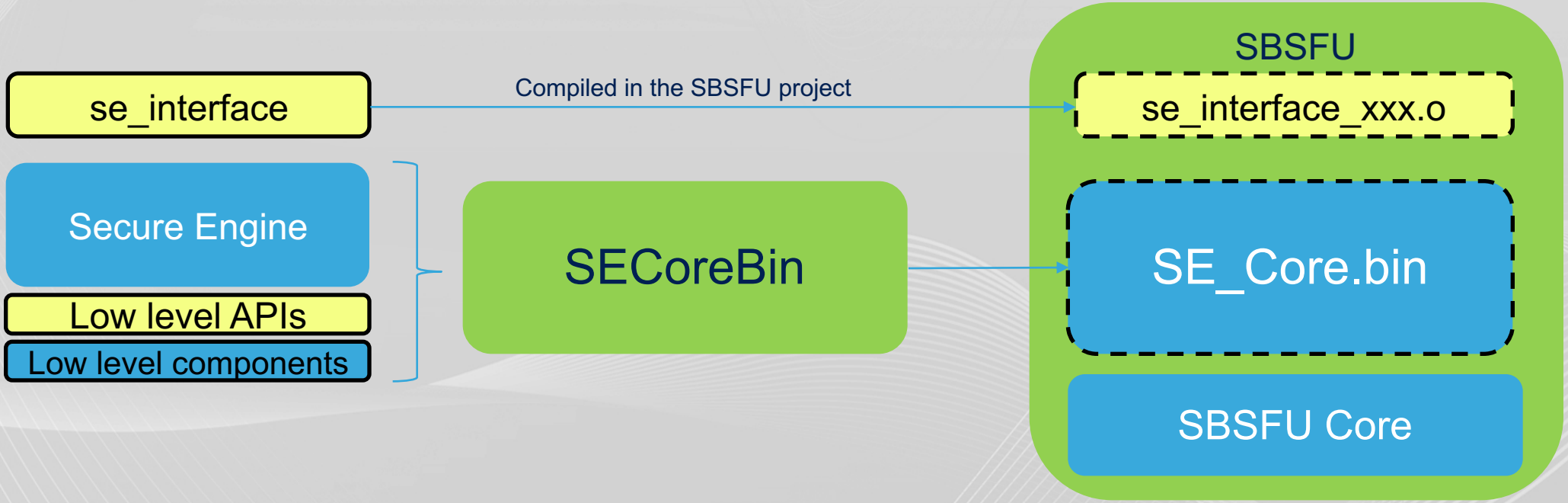
Dual Image Mode Of Operation



Single Image Mode Of Operation

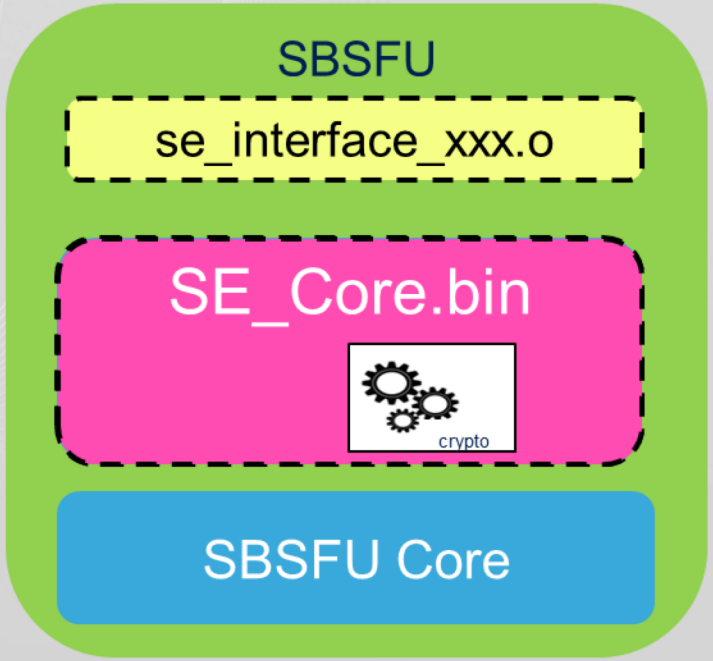
Component Deployment Model

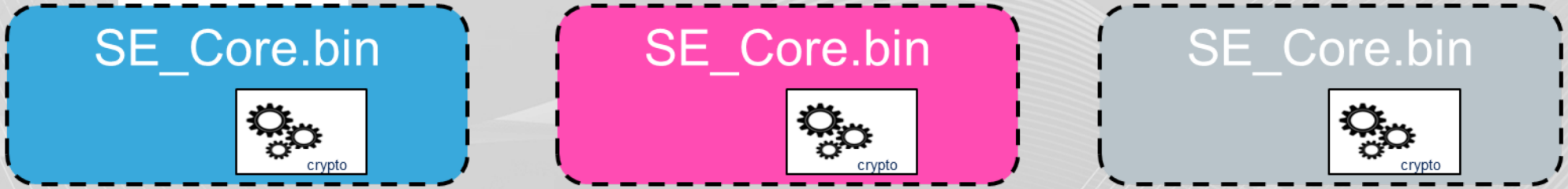
- UserApp: sample user application (mutable firmware: this is the updated FW)
- SECoreBin: the binary running in the secure enclave (isolated execution environment)
- SBSFU: the Secure Bootloader with Secure Firmware Update capability
- Secure Engine, SECoreBin and SBSFU are tightly coupled



Component Deployment Model

- Choosing a cryptographic scheme means choosing your SECoreBin flavor (*)
- Implementing an alternate cryptographic scheme means
  - Instantiating the templates with the required
    - Code implementing the APIs
    - Information in the Firmware Header structure
  - Updating the FW image preparation tool if needed

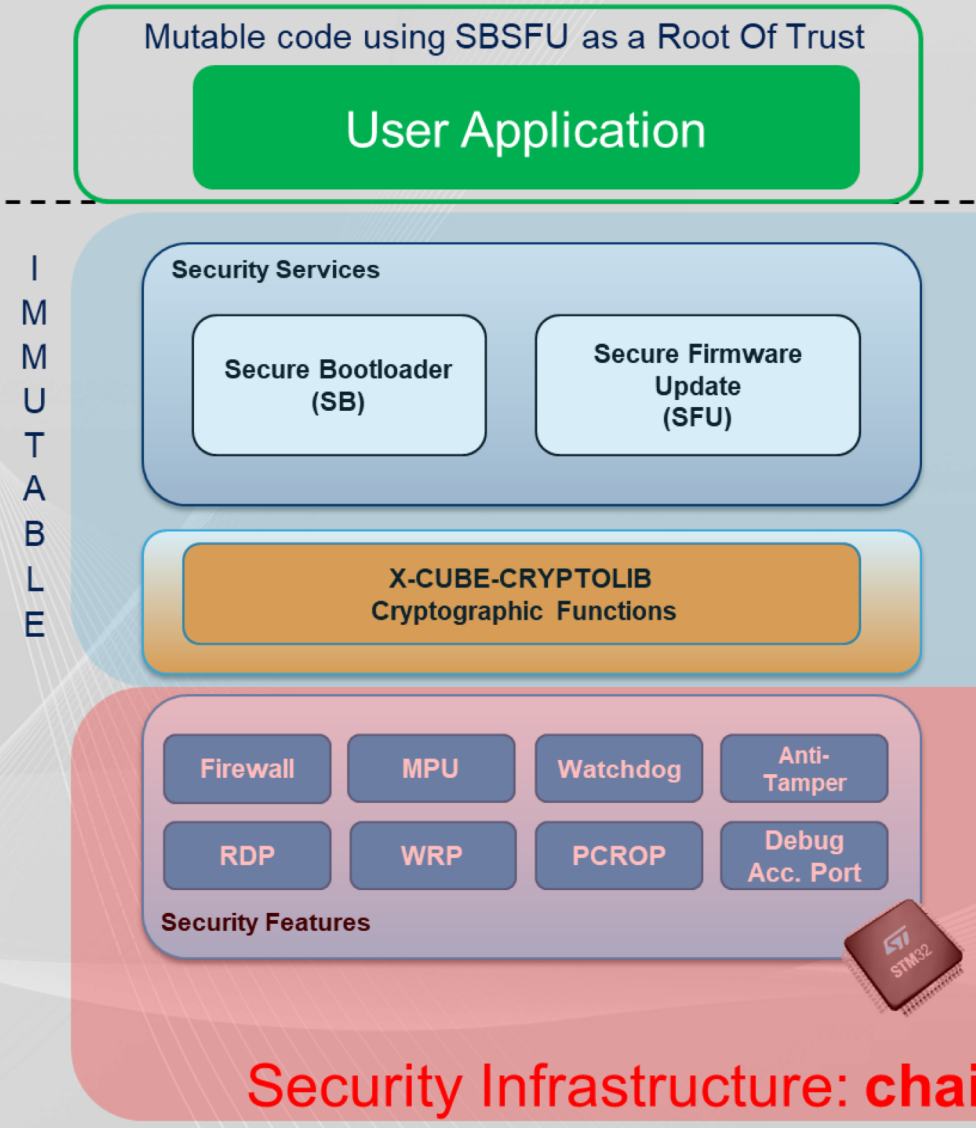




Default Crypto Scheme

Alternate Crypto Schemes

Security Layering

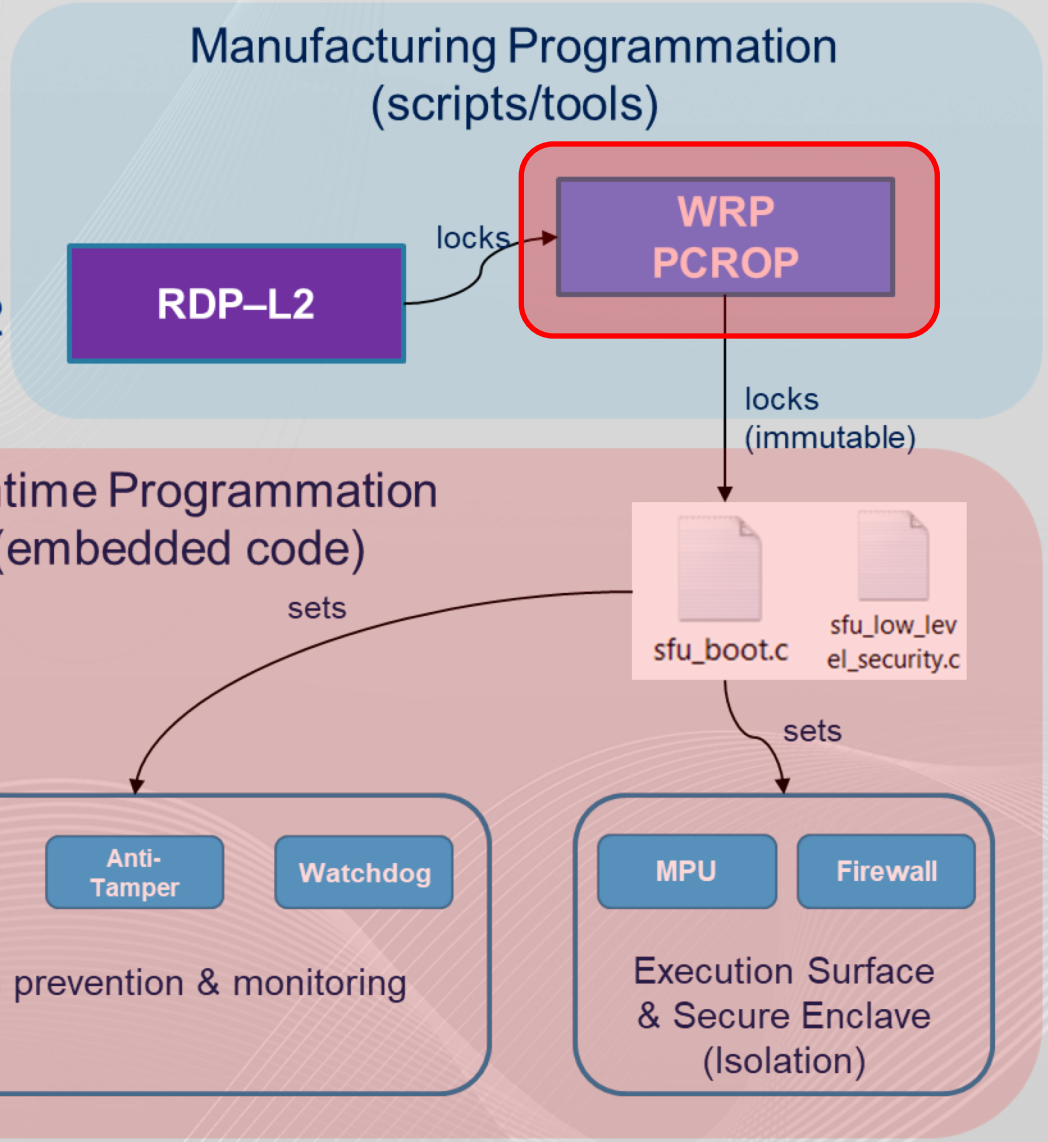


- Secure Boot and Secure Firmware Update
  - Use combination Security features working together to offer multiple layers of protection
- Cryptographic Functions
  - Preserve confidentiality, verify integrity, authenticity
- MCU Security Features
  - Used to establish a robust platform on which trusted processes and associated cryptography can be performed

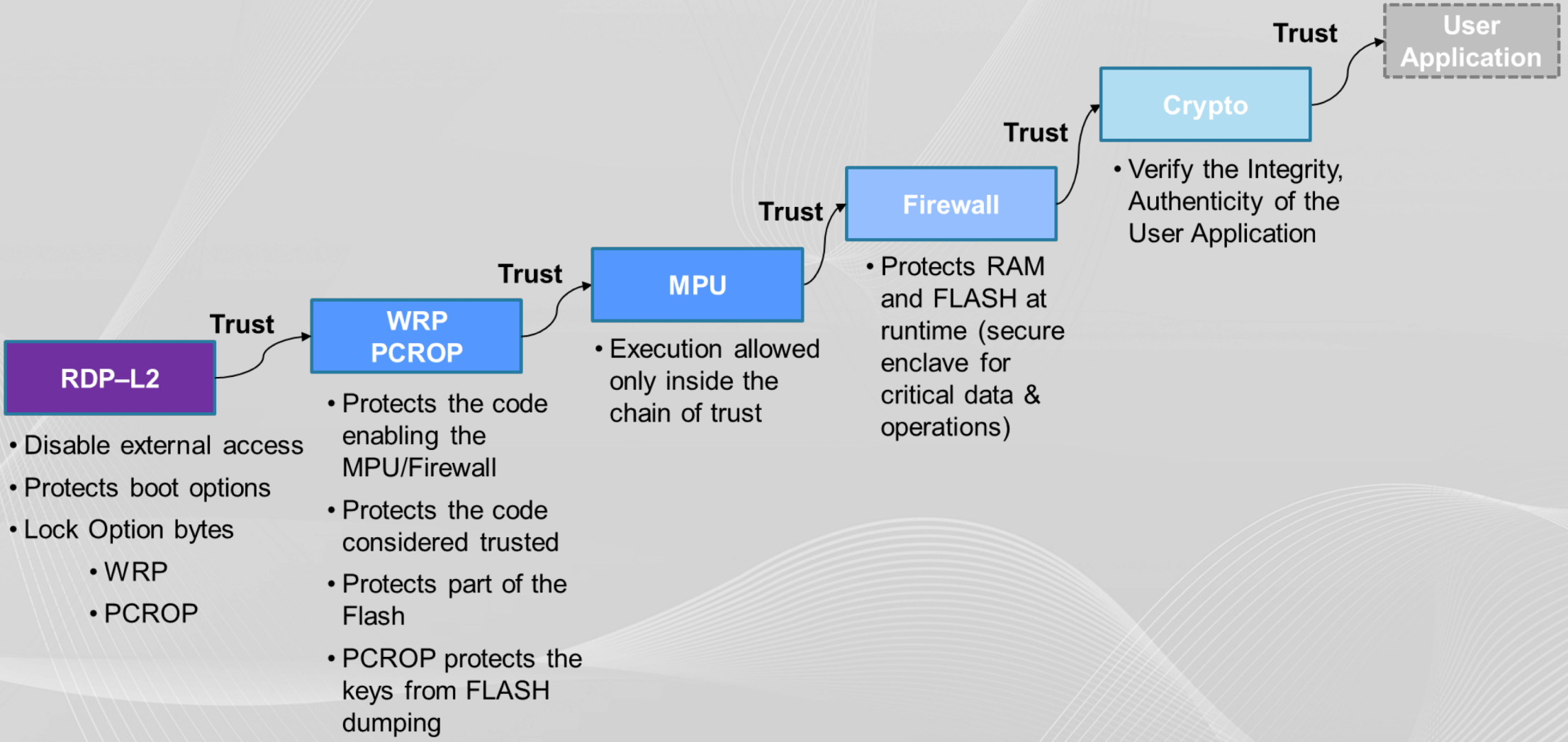
Security Infrastructure: **chain of protections** building a **Root Of Trust**

Types of Protections

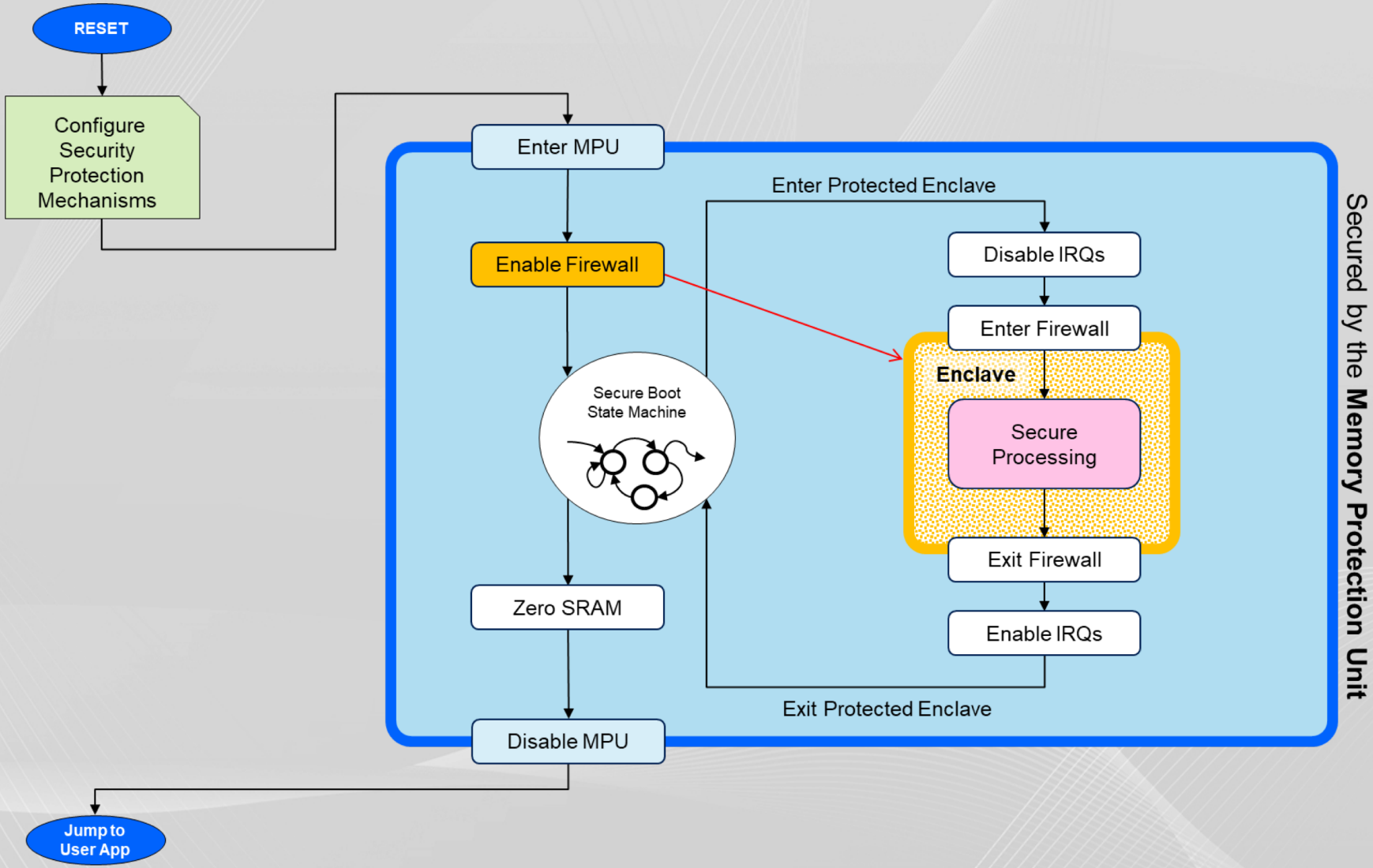
- Static Protections
  - Controlled by Option Bytes
  - Always active once enabled (not disabled at reset)
  - Option Bytes shall be locked thanks to RDP Level 2
  - Disables external access (FLASH/RAM)
  - Protects code & secrets from external access
- Runtime Protections
  - Must be programmed at each reset
  - Controls the Surface of Execution
  - Creates a Secure Enclave
    - Protection against inner attacks
  - Monitors the system



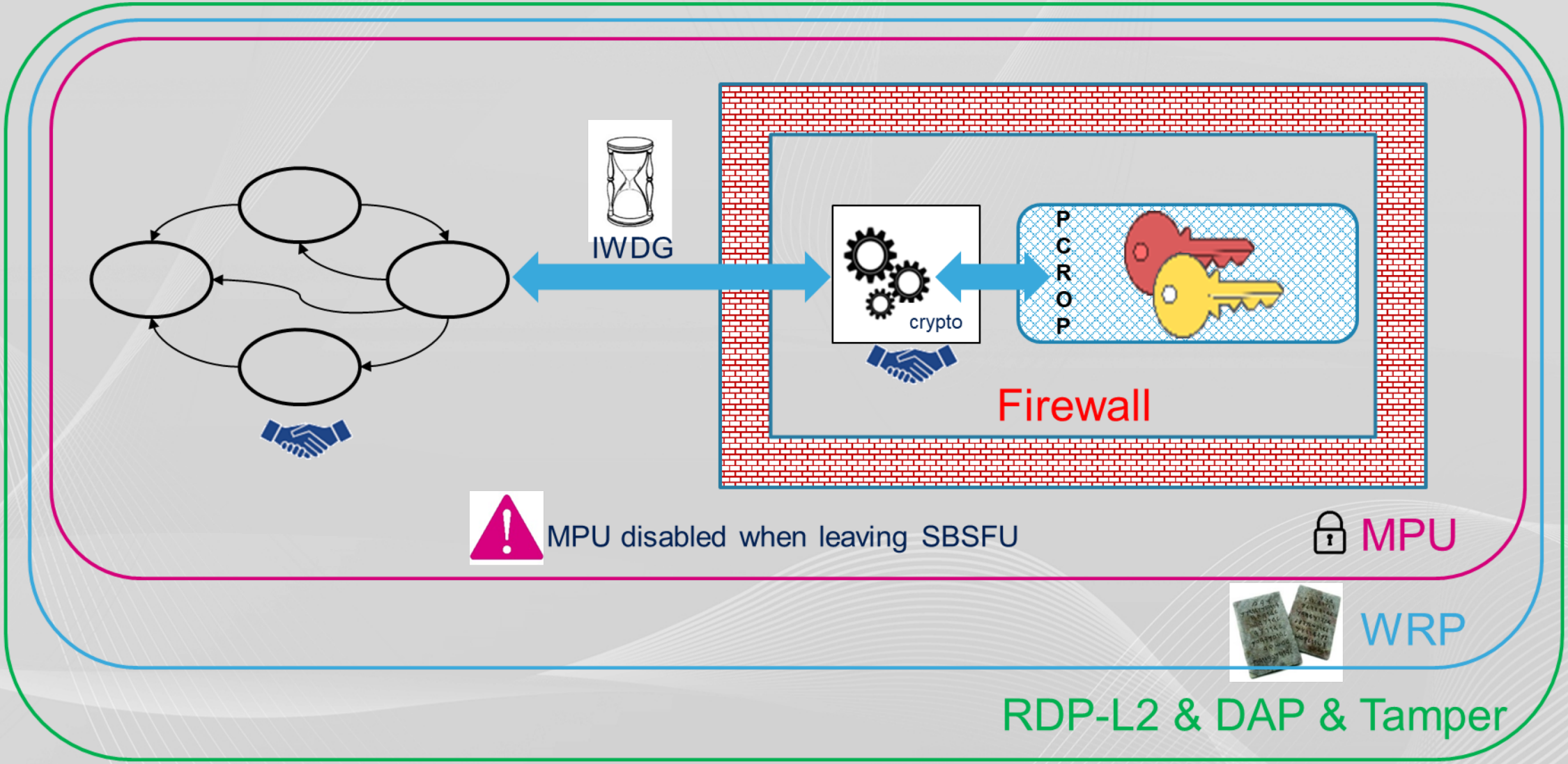
Chain of Trust



Protected Processing



Secure Enclave: Secrets Storage



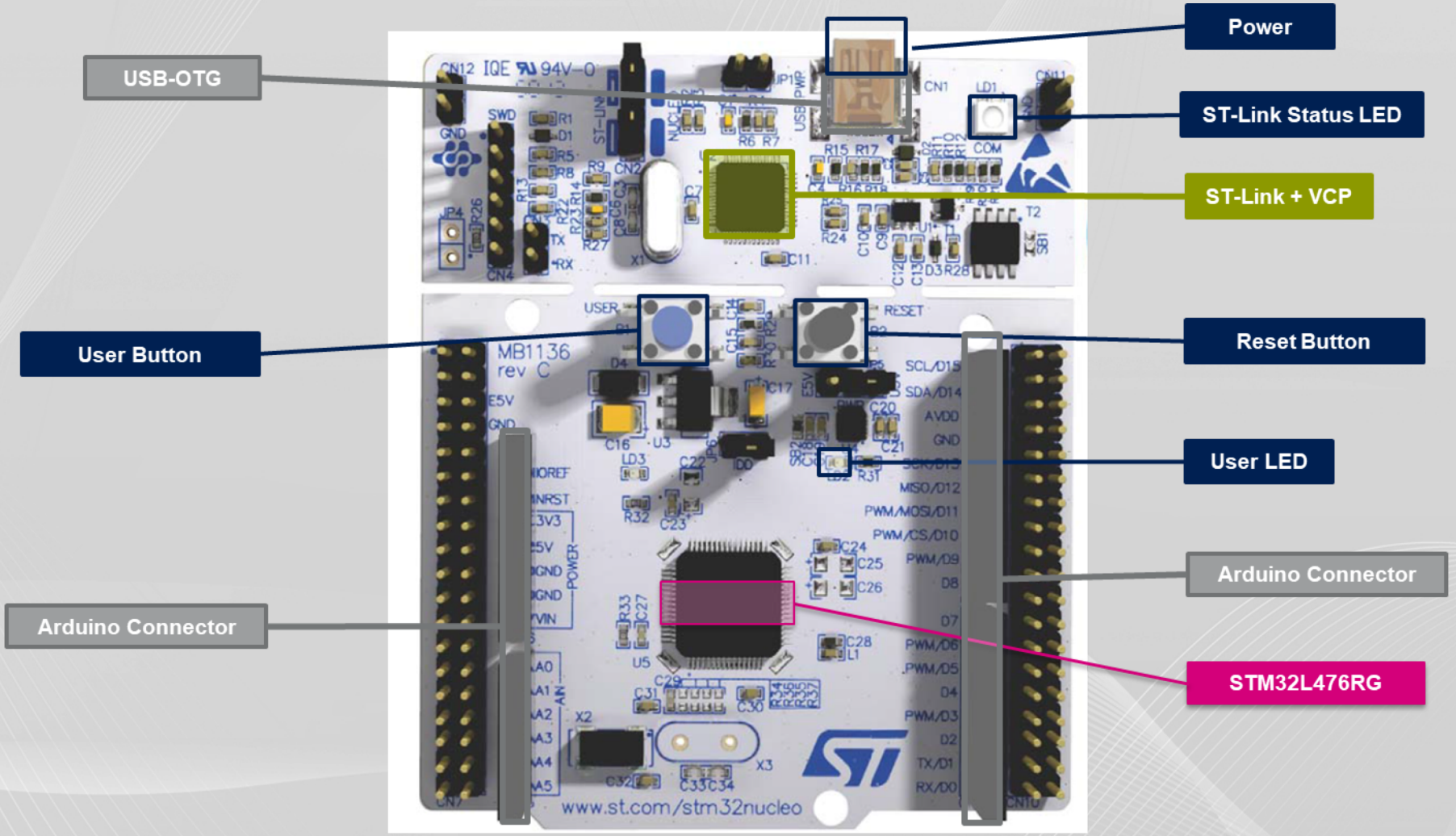
Non protected areas ….

- The FLASH slots storing the User Firmware Images are not protected
  - Only the Header of the Active Slot is protected by Firewall
- The Active Firmware Image can be altered: SBSFU does NOT prevent this
  - After altering the Active Firmware Image you can install any valid FW version
  - This is a way to re-install version N-1 despite the ‘anti-rollback’ check at installation stage
- The slot #1 can be altered too
  - Denial of Service is possible (no rollback possible)
- A lot more things can probably be done

SBSFU is **an example (the security grade is unknown)**

SBSFU is for free (available on st.com)…security is not…

The NUCLEO-L476RG



DEMO Project Setup

BENINGO EMBEDDED GROUP

Bootloader Demo

Secure Bootloader Best Practices

- Start your secure bootloader design early!
- Remember that security isn't free
- Select a microcontroller that supports security
- Lock the flash security bits to protect the bootloader and application
  - Secure boot should be immutable
- Securely store private keys
- Clearly identify up-front the level of security that is necessary for the bootloader
- Develop a chain of trust
- Use signatures to authenticate the firmware source

Going Further

- Download beningo.com resources
  - C Doxygen templates
  - RTOS Best Practice Guide
  - Bootloader White Paper
  - Bootloader Design Techniques Course
- STM Resources
  - X-CUBE-SBSFU

Questions