课程目录

life.augmented

life.augmented

STM32WL 安全特性介绍

David Liu

STM32

课程目录

life.augmented

life.augmented

STM32WL 安全特性概览

STM32

STM32WL – 安全需求与保护

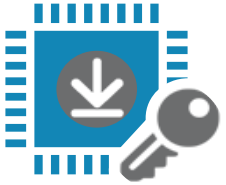
安全需求与保护

Backup clock circuit
Supply monitoring
Watchdog

设备完整性

RDP / CM0 / Debug CM4
Boot lock
Tamper

防入侵



Secure firmware Install(SFI)
Secure Boot(SB)
/ Upgrade(SFU)

安全安装/启动/更新

FLASH ECC
SRAM Parity bit
CRC
Hash 256

数据完整性

HDP
PCROP / WRP

固件IP防护

AES256
TRUE RNG
Key Management Services(KMS)

数据加密

96-bit Unique ID
64-bit IEEE Unique ID

可追溯性

MPU
GTZC

权限许可管理

Crypto lib
PKA（RSA/ECDSA）

认证

life.augmented

*软件实现
*硬件加解密模块

STM32WL- 信任链

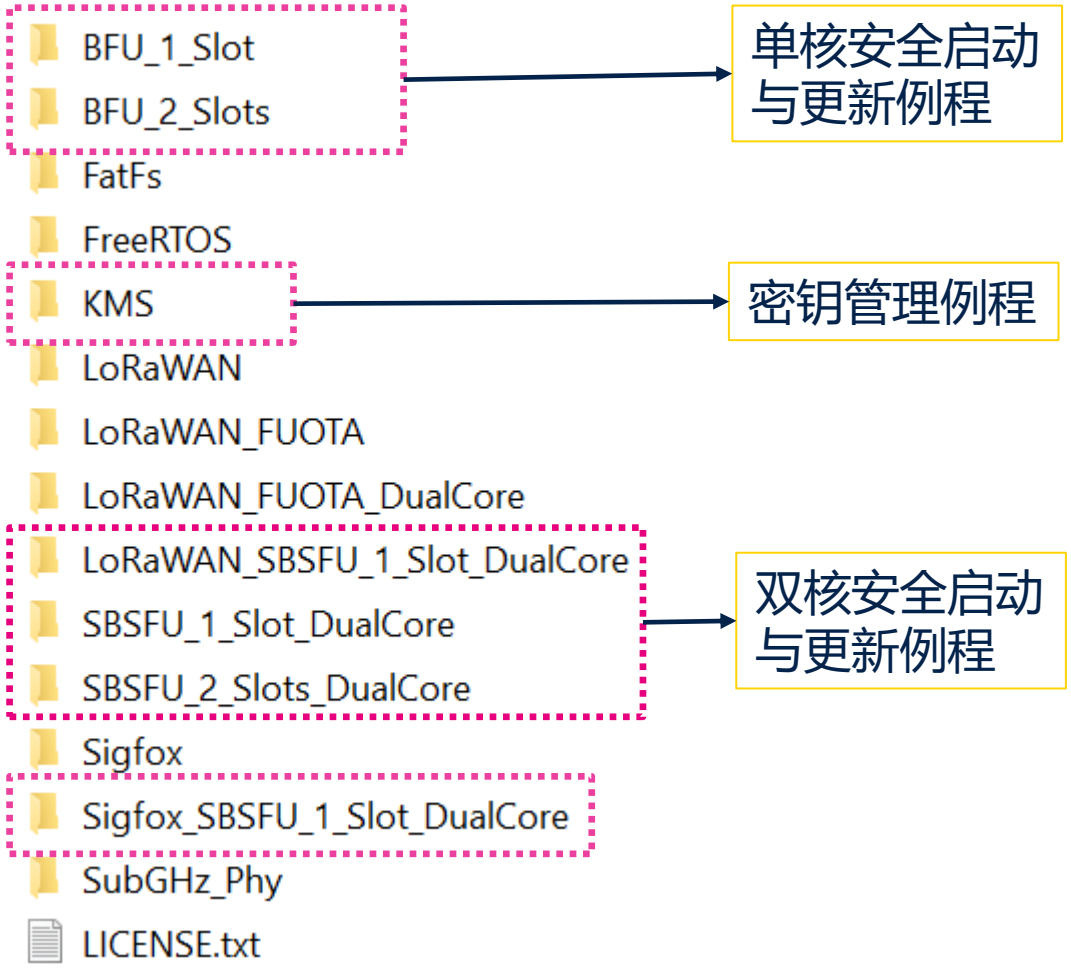
安全功能构建的信任链

- 复位时首先执行安全启动（唯一启动入口，启动代码不被修改）
- 接下来是身份验证和认证（RF协议栈 & 用户应用程序）
- 通过验证的应用程序才允许正式执行(以可信的方式)

life.augmented

STM32Cube_FW_WL安全例程

STM32Cube_FW_WL_V1.1.0\Projects\NUCLEO-WL55JC\Applications



STM32WL SBSFU例程对照表

	USART传输	OTA
单核	BFU_1_Slot BFU_2_Slots	LoRaWAN_FUOTA
双核	LoRaWAN_SBSFU_1_Slot_DualCore SBSFU_1_Slot_DualCore SBSFU_2_Slots_DualCore Sigfox_SBSFU_1_Slot_DualCore	LoRaWAN_FUOTA_DualCore
以上所有升级例程都集成了“安全”功能：即固件加解密和验证		

life.augmented

life.augmented

STM32WL 通用安全功能特性

STM32

通用安全功能特性

- Boot lock
  - 强制 CPU 从用户flash启动
- Readout Protection (RDP)
  - 三级保护 RDP0, RDP1, RDP2
  - 在不同访问条件下（从用户Flash启动，从SRAM启动，从系统Flash启动，调试端口连接），对不同区域（用户Flash，备份域Flash，部分SRAM）的访问权限进行控制
- Proprietary code Read Out Protection (PCROP)
  - 片上flash的私有代码保护
- Write protection (WRP)
  - 保护片上flash内容不被修改、擦除
- Memory protection unit(MPU)
  - 应用程序能够利用多个权限级别，进行任务隔离, 保护代码,数据和堆栈.

通用安全详细介绍

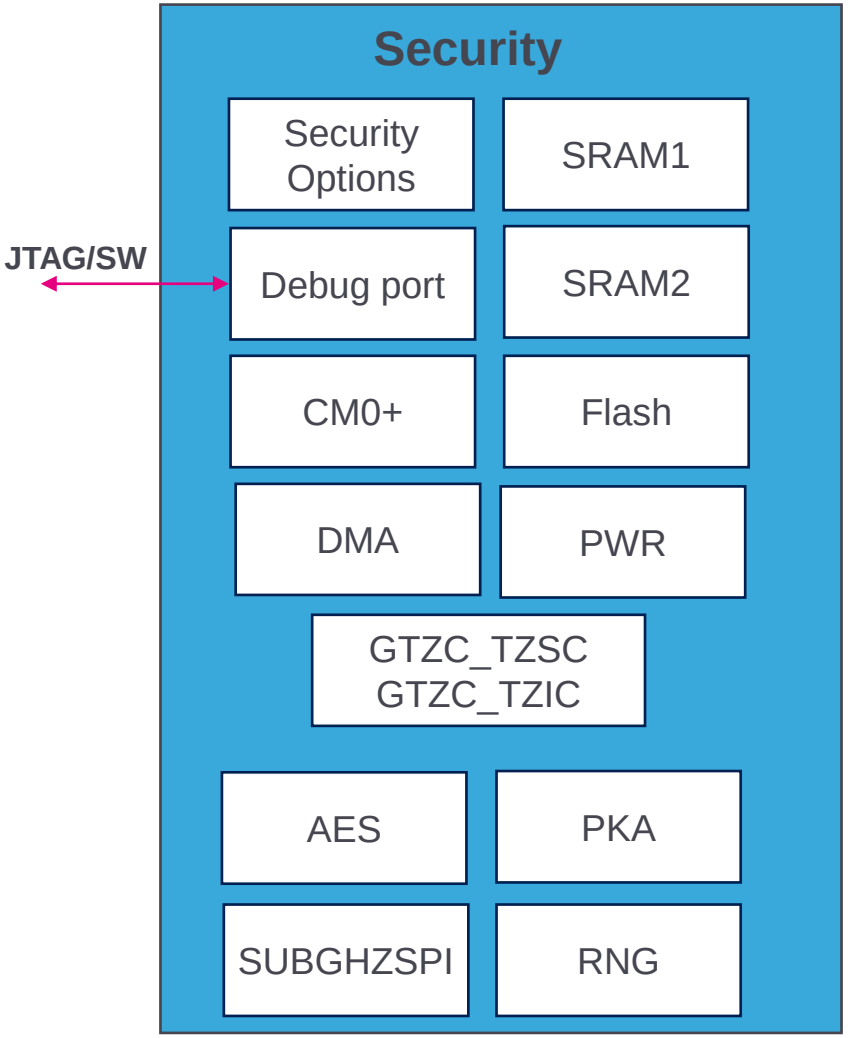
STM32信息安全课程线上学习视频链接：

本专栏为大家系统性解读STM32的安全技术，包括STM32的安全特性和软硬件资源。

- 生态系统 | 意法半导体STM | STM32/STM8微控制器 | MCU单片机 (stmcu.com.cn)
- 电堂科技 (51diantang.com)

STM32WL5x双核安全特性

概览



- 只能被Cortex-M0+ 访问
  - 安全存储区：
    - Flash memory, SRAM1, and SRAM2
  - 安全外设：
    - DMA, PWR, AES, PKA, TRNG, and SUBGHZSPI
- 通过Flash 选项字节 和GTZC控制安全和特权访问
  - 复位后CM4和CM0+以上都可访问，当选项字节和GTZC开启后只有CM0+能访问
- 通过非法访问中断进行安全入侵监控。

优点

- 在CM0+安全存储区上可以安全存储密钥
- 可以安全实现密码学和射频通信
- 实现认证和安全固件安装、更新

life.augmented

STM32WL5双核安全特性

存储器安全：Flash 和 SRAM

- 通过选项字节配置Flash和SRAM的安全属性.
- 安全属性开启后只能被Cortex-M0+ 访问

特权保护

- 保护 Cortex-M0+ 安全特权资源免受安全非特权访问

安全外设

- 通过选项字节配置.
  - SUBGHZSPI, 确保 sub-GHz 射频通信安全.开启后只能被Cortex-M0+ 访问
- 通过在 Cortex-M0+ 上安全固件运行时启用
  - 启用后AES、PKA、TRNG 只能被Cortex-M0+ 访问。
  - 启用后DMA通道只能被Cortex-M0+ 访问。

STM32WL5双核安全特性

非法访问保护

向 Cortex-M0+ 发出对安全或特权资源的非法访问的信号。

安全启动

设置CM4和CM0+唯一启动入口。

安全调试

关闭调试端口，安全存储区和安全外设无法通过调试端口访问。

通过选项字节配置Cortex-M0+ 安全特性

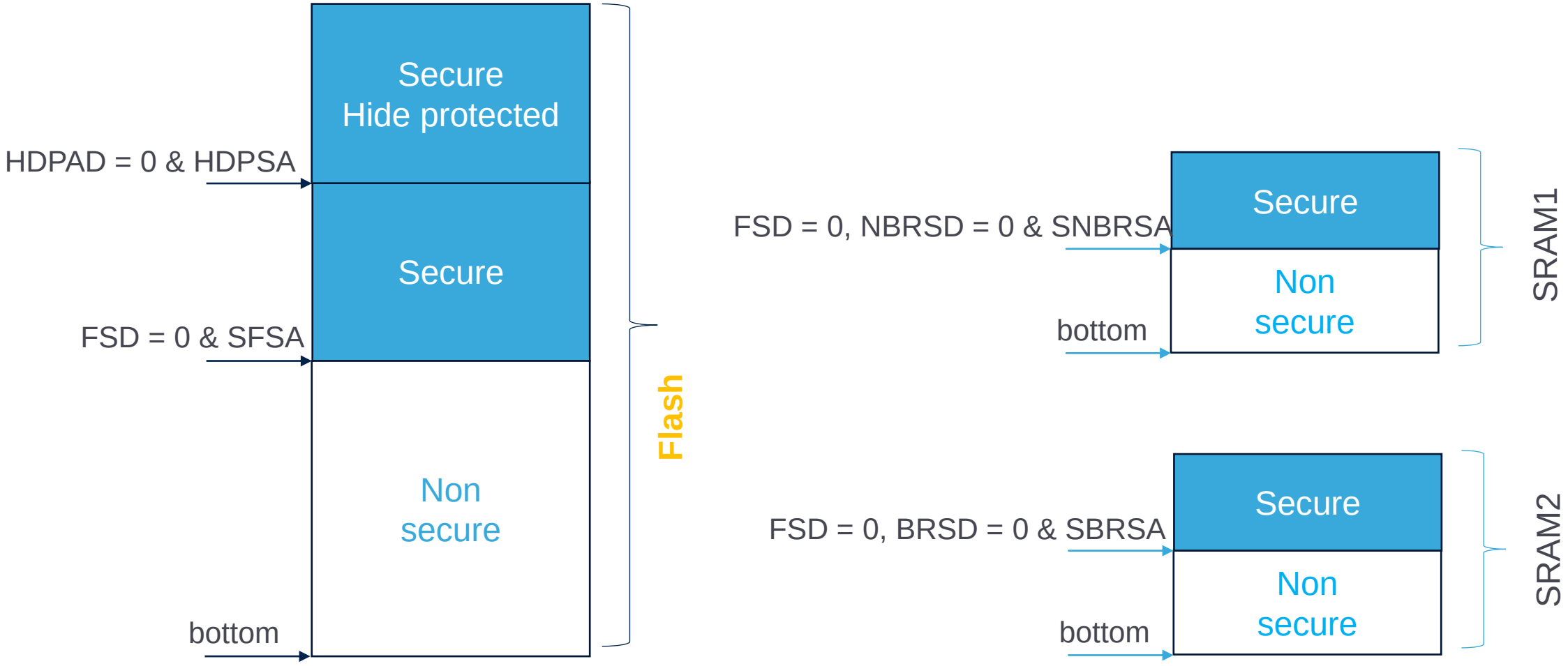
通过选项字节配置Cortex-M0+ 安全特性

Register	Fields									
OPTR (*)	C2BOOT_LOCK	User Options							ESE	RDP
SFR (*)	SUBGHZSPISD	res.	HDPAD		HDPSA	res.	DDS	res.	FSD	SFSA
SRRVR (*)	C2OPT	NBRSD	SNBRSA	res.	BRSD	SBRSA	SBRV			

*OPTR: Options Register
*SFR: Secure Flash Register
*SRRVR: Secure Ram and Reset Vector Register

- 当ESE=1 启用 Cortex-M0+ 安全性后，安全用户选项只能由 Cortex-M0+ 写入。
  - 非安全 Cortex-M4 可以读取安全用户选项，从而确定安全设置。

存储器安全：Flash 和 SRAM



只能被 Cortex-M0+访问

Cortex-M4 和Cortex-M0+都可以访问

life.augmented

通过GTZC配置安全属性

GTZC(Global security controller) 包含两个模块:

- **TZSC(TrustZone security controller) :**

TrustZone 安全控制器， 此子模块定义从属外围设备的安全/特权状态。 它还控制MPCWM 的非特权区域大小。

- **TZIC(TrustZone illegal access controller）：**

TrustZone 非法访问控制器， 该子模块收集系统中的所有非法访问事件，并向 CM0+ NVIC 生成安全中断。

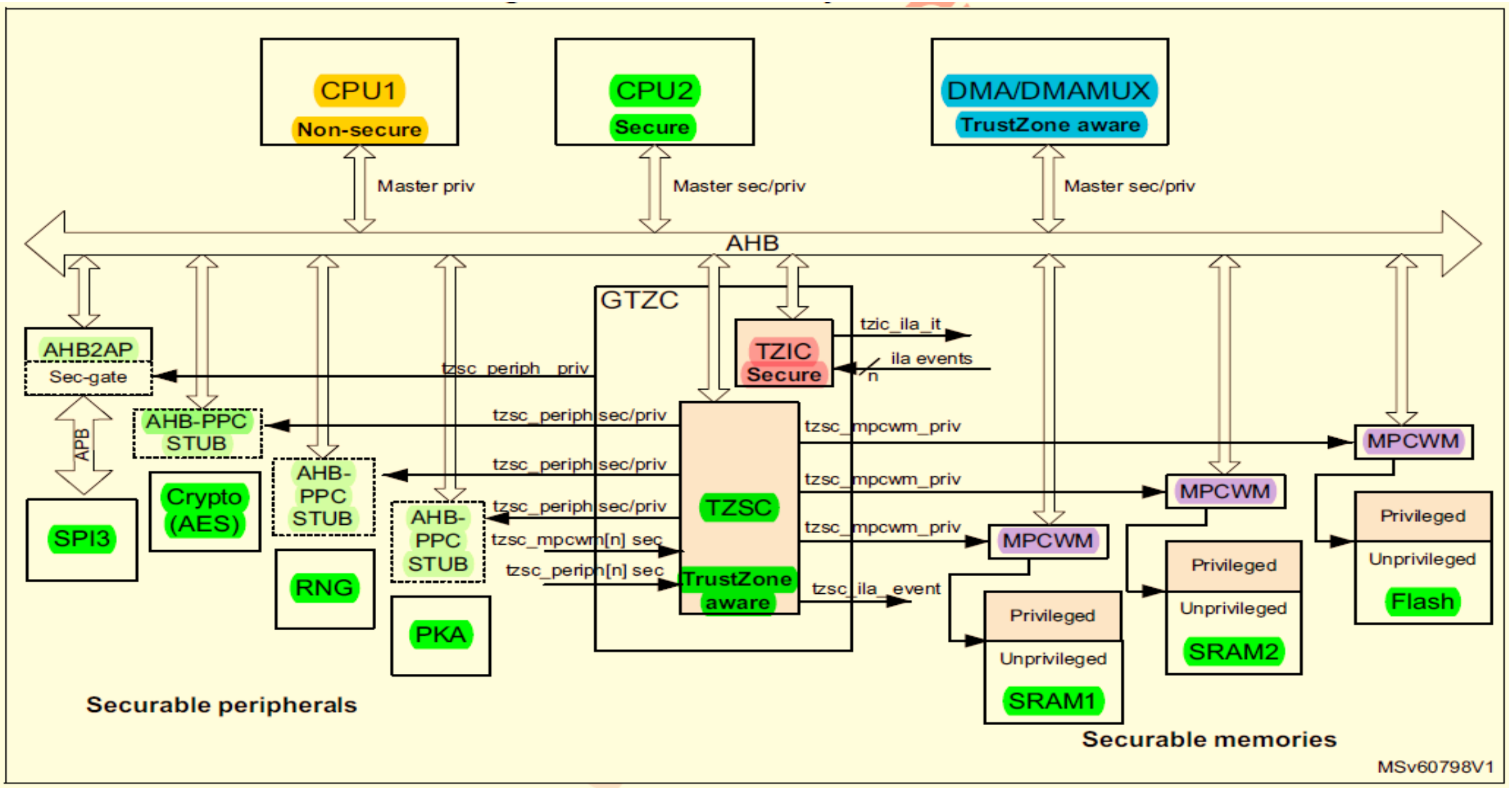
这两个模块用于配置系统安全和权限，例如:

- 片上flash和 RAM的特权访问
- AHB 和 APB 外设的安全/特权状态

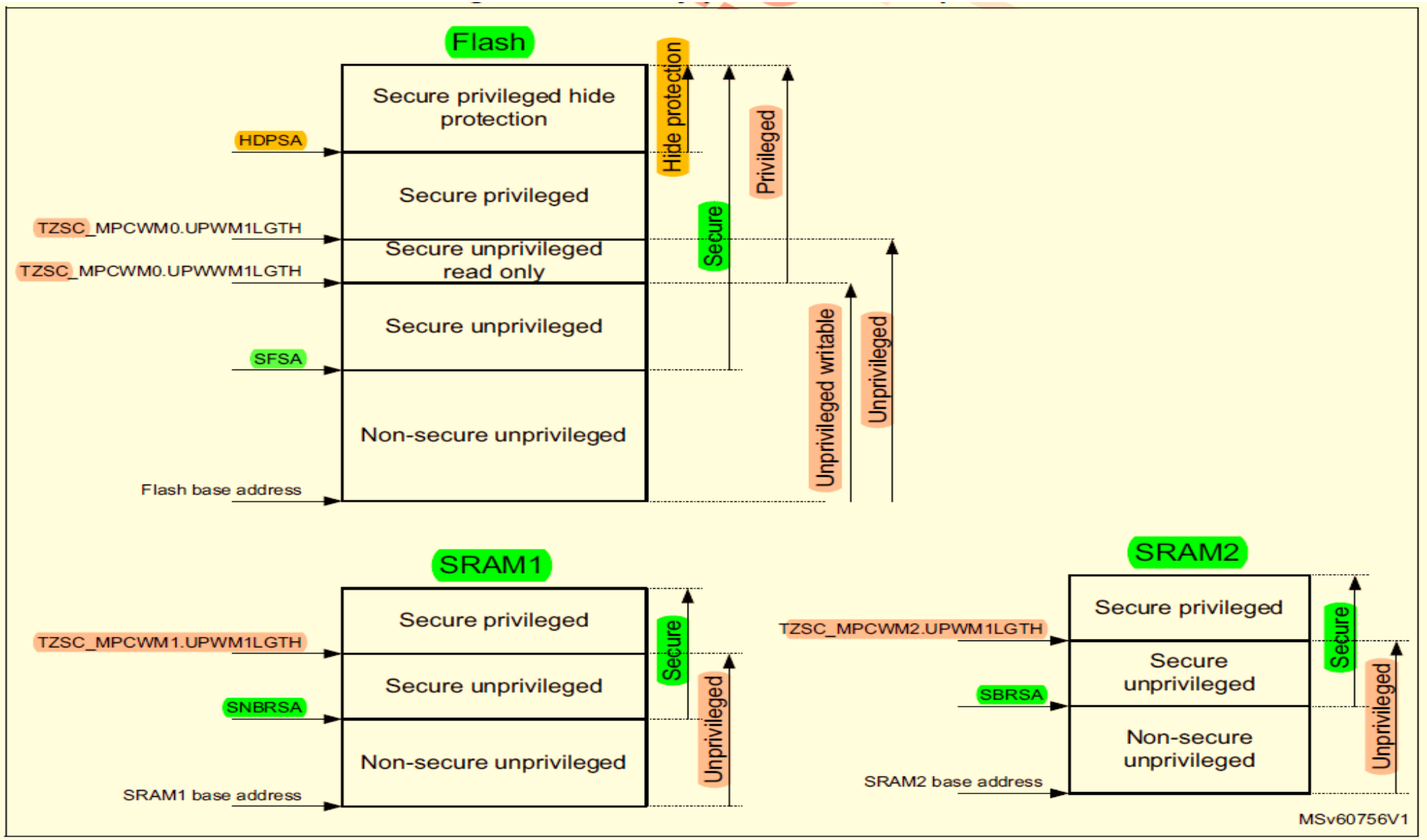
特权保护

- 权限保护由 GTZC_TZSC 中的寄存器位设置。
  - 允许保护特权资源免受非特权访问。
- 每个存储器（Flash/SRAM1/SRAM2）都可以使用一个特权水印。
- 权限保护适用于以下资源
  - 存储器、sub-GHZ 射频模块、AES、PKA、TRNG、DMA 通道。

GTZC 架构


CPU1
Non-secure
CPU2
Secure
DMA/DMAMUX
TrustZone aware
Master priv
Master sec/priv
Master sec/priv
AHB
GTZC
AHB2AP
Sec-gate
APB
AHB-PPC STUB
AHB-PPC STUB
AHB-PPC STUB
TZIC
Secure
tzic_ila_it
ila events
n
tzsc_periph_priv
tzsc_periph sec/priv
tzsc_periph sec/priv
tzsc_periph sec/priv
tzsc_mpcwm[n] sec
tzsc_periph[n] sec
TZSC
TrustZone aware
tzsc_mpcwm_priv
tzsc_mpcwm_priv
tzsc_mpcwm_priv
tzsc_ila_event
MPCWM
MPCWM
MPCWM
Privileged
Unprivileged
SRAM1
Privileged
Unprivileged
SRAM2
Privileged
Unprivileged
Flash
SPI3
Crypto (AES)
RNG
PKA
Securable peripherals
Securable memories
MSv60798V1

存储器安全和特权保护


Flash
Secure privileged hide protection
Secure privileged
Secure unprivileged read only
Secure unprivileged
Non-secure unprivileged
HDPSA
TZSC_MPCWM0.UPWM1LGTH
TZSC_MPCWM0.UPWWM1LGTH
SFSA
Flash base address
Hide protection
Secure
Privileged
Unprivileged writable
Unprivileged
SRAM1
Secure privileged
Secure unprivileged
Non-secure unprivileged
TZSC_MPCWM1.UPWM1LGTH
SNBRSA
SRAM1 base address
Secure
Unprivileged
SRAM2
Secure privileged
Secure unprivileged
Non-secure unprivileged
TZSC_MPCWM2.UPWM1LGTH
SBRSA
SRAM2 base address
Secure
Unprivileged
MSv60756V1

安全外设

- GTZC_TZSC配置 的安全外设。
  - 安全外设：AES、PKA 和TRNG。
  - 允许在运行时设置外设安全属性。
  - 允许在安全 CM0+ 和非安全 CM4 之间根据需要共享外设。

- DMA 寄存器配置DMA 通道安全性。
  - 允许在运行时设置DMA 通道安全属性。
  - 允许在安全 CM0+ 和非安全 CM4 之间根据需要共享 DMA 通道。

- 安全用户选项字节设置Sub-GHz 射频模块安全属性
  - 由 SUBGHZSPISD 选项字节控制，上电生效。
  - 只能由安全的 Cortex-M0+ 访问sub-GHz 射频模块， 。

非法访问保护

- 启用GTZC_TZIC后，向Cortex-M0+ 发出对安全资源的非法访问的信号。非法访问会将 Cortex-M0+ 从任何操作模式唤醒。
- 非法访问信息可能来自于对以下存储器和外设的访问：
  - 安全/特权存储区 Flash、SRAM1 和 SRAM2。
  - 安全/特权外设 DMA、DMAMUX、SUBGHZSPI、AES、PKA 和TRNG。
  - GTZC 和 PWR 中的安全和权限控制。

Boot Lock

- CM4 boot lock 信任链
  - 设置BOOT_LOCK 强制 CM4 从用户flash启动。

- CM0+ boot lock 信任链
  - 设置C2BOOT_LOCK 强制 CM0+ 从 SBRV 和 C2OPT 启动。

安全调试

- 安全用户选项字节(DDS)设置调试访问
  - DDS=1,禁用对 Cortex-M0+ 的调试端口访问。

- 调试访问控制独立于安全可以在安全和非安全模式下启用和禁用调试。
  - 在安全模式下(ESE=1)，调试访问只能由安全 Cortex-M0+ 更改
  - 在非安全模式下(ESE=0)， Cortex-M4和Cortex-M0+ 都可以通过DDS使能或关闭Cortex-M0+ 调试

life.augmented

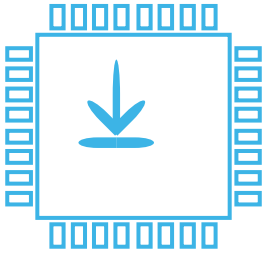
STM32WL 的安全功能

STM32

STM32WL – LoRa节点安全三大需求

LoRa节点安全三大需求

- 确保节点以安全的方式安装/更新软件
  - 安全安装（SFI）
  - 安全更新（Secure Firmware Update）
- 确保节点运行的是安全的软件
  - 安全启动（Secure Boot）
- 确保节点软件以及关键信息被保护，不受攻击
  - 执行与保护模块，安全密钥管理服务(KMS)



安全下载

固件IP保护

数据加密

life.augmented

STM32WL的安全功能

STM32WL三大安全功能

- **KMS**（密钥管理服务）
  - 通过标准 PKCS#11 API 提供加密服务

- **SBSFU**（安全启动和安全固件更新）
  - 以安全的方式启动 STM32
  - 每个启动阶段(SB)在执行之前,都是不可变的或经过身份验证的
  - 安全固件更新(SFU)允许以安全的方式更新新的固件版本
  - 新固件在更新之前将首先进行身份验证和解密

- **SFI**（安全固件安装）
  - 解决首次安装固件的安全问题
  - 客户使用自己的固件加密密钥 和 ST 工具(PackageCreator)加密其固件
  - 产线使用支持SFI的工具对用户的加密密钥进行烧录

life.augmented

STM32WL安全功能 KMS

KMS-概览

KMS概览

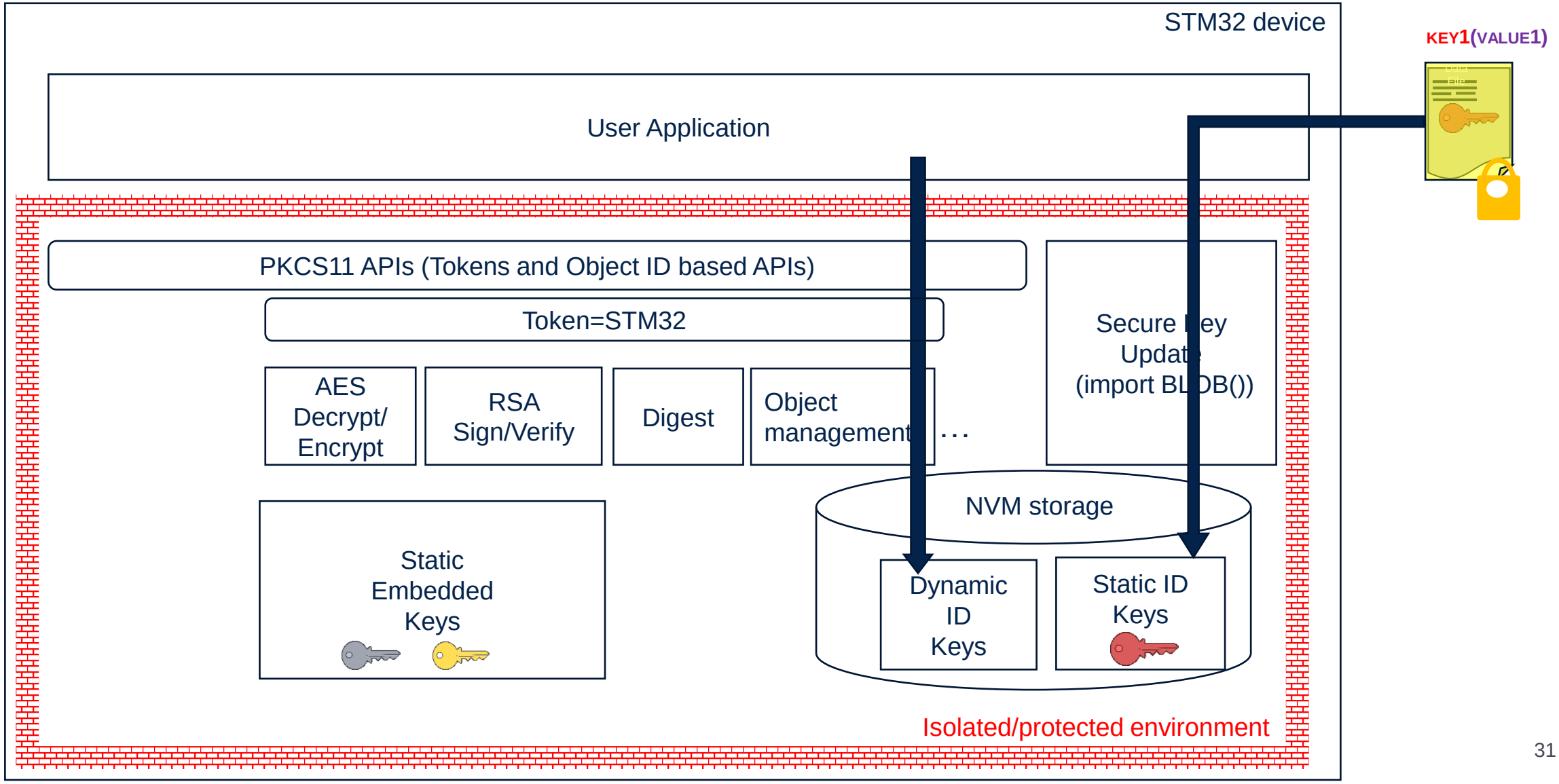
- KMS
  - 统一管理和使用密钥的方式，提供密钥管理服务
  - 基于标准 PKCS#11 API 提供加密服务
  - 在受保护的环境中执行，以确保密钥值不会被未经授权的代码访问

KMS-特征

- KMS主要特征
  - 对象管理（创建、更新、删除）
  - AES 加密/解密
  - 摘要功能
  - RSA 和ECDSA 的签名/验证
  - 密钥管理功能：密钥生成/派生

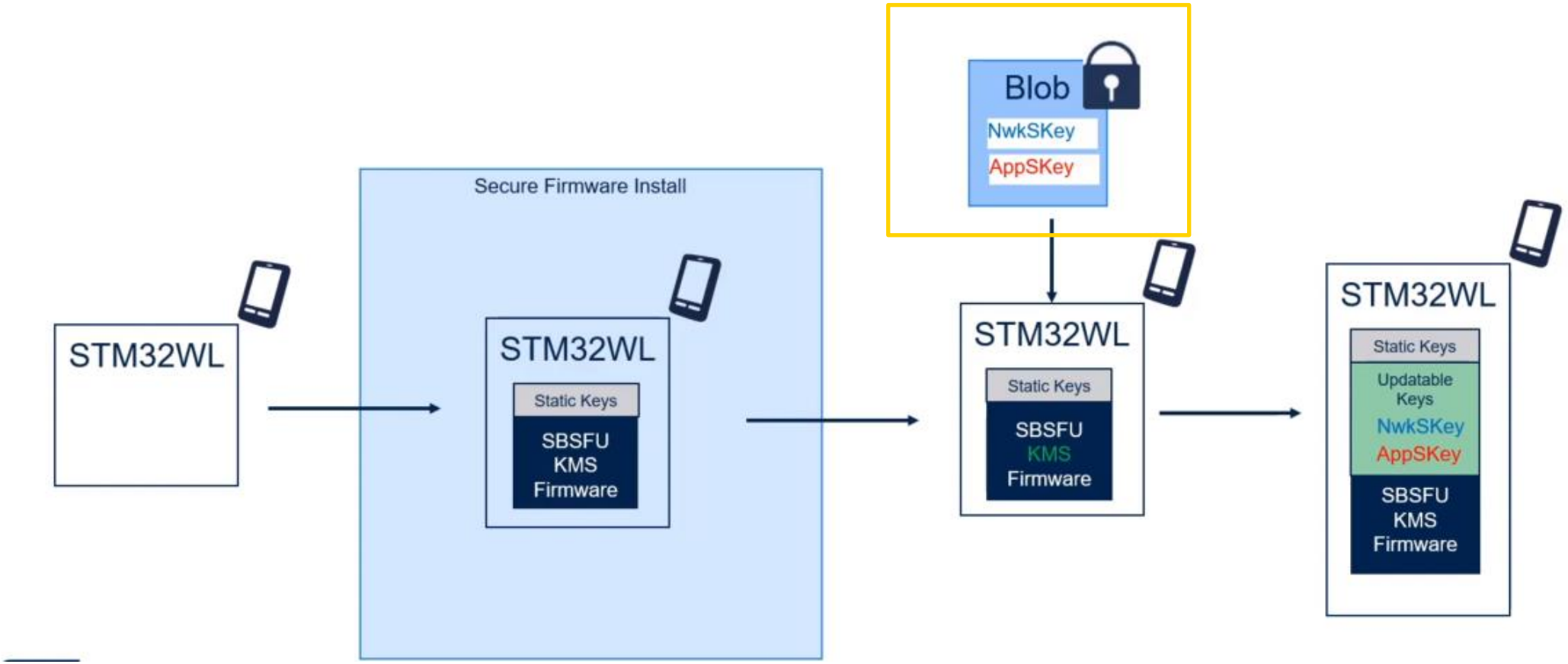
- KMS 管理 3 种类型的密钥
  - **静态嵌入密钥**
    - 嵌入代码中的预定义密钥
    - 此类密钥不能修改
  - **具有静态 ID 的可更新密钥**
    - 密钥 ID 在系统中预定义
    - 可以使用静态嵌入式根密钥通过安全程序在 NVM 存储中更新密钥值
    - 此类密钥无法删除
  - **具有动态 ID 的可更新密钥**
    - 创建密钥时定义密钥 ID
    - 密钥值是使用内部函数创建的。 通常，DeriveKey() 函数创建动态对象
    - 这样的密钥可以删除

KMS – 架构


STM32 device
KEY1(VALUE1)
User Application
PKCS11 APIs (Tokens and Object ID based APIs)
Token=STM32
AES
Decrypt/
Encrypt
RSA
Sign/Verify
Digest
Object
management
...
Secure Key
Update
(import BLOB())
NVM storage
Static
Embedded
Keys
Dynamic
ID
Keys
Static ID
Keys
Isolated/protected environment

KMS – 应用例程

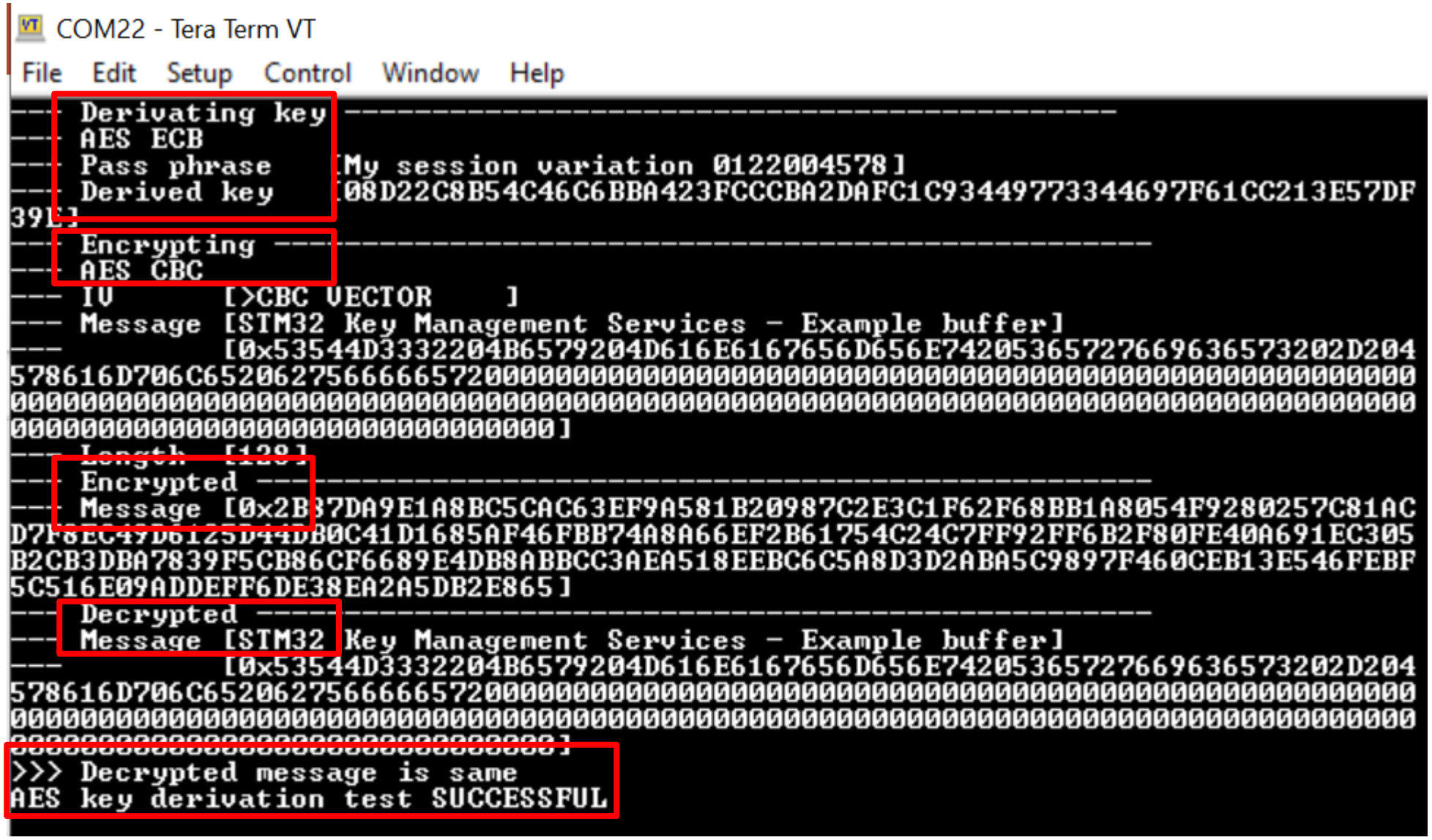
LoRaWAN 安全密钥（APPSKey, NwKSKey/Appkey）



KMS – 应用例程

STM32Cube_FW_WL_V1.1.0\Projects\NUCLEO-WL55JC\Applications\KMS

KMS – 应用例程



life.augmented

STM32WL 安全功能SBSFU

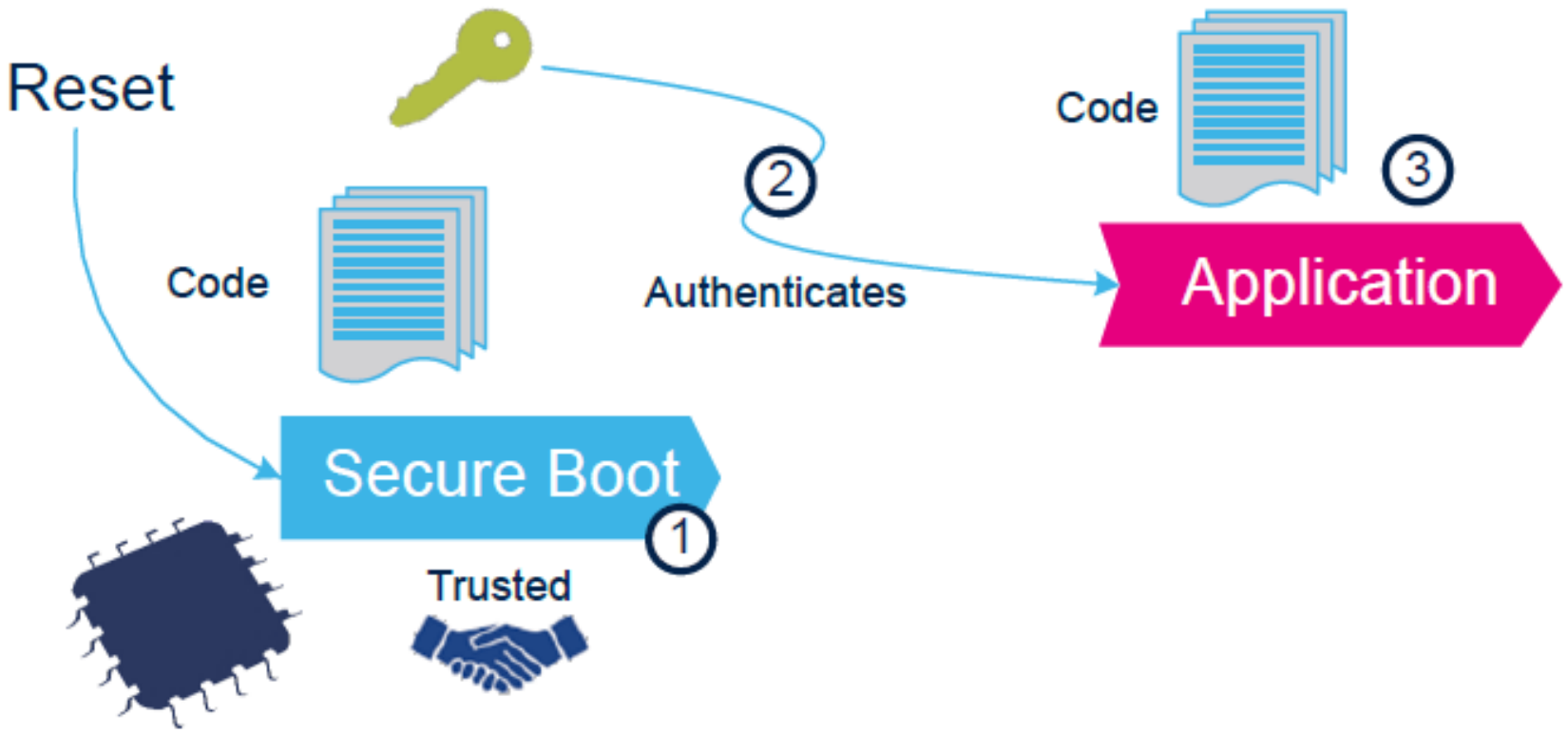
SBSFU概览

SBSFU=安全启动（Secure Boot）+ 安全更新（Secure Firmware Update）

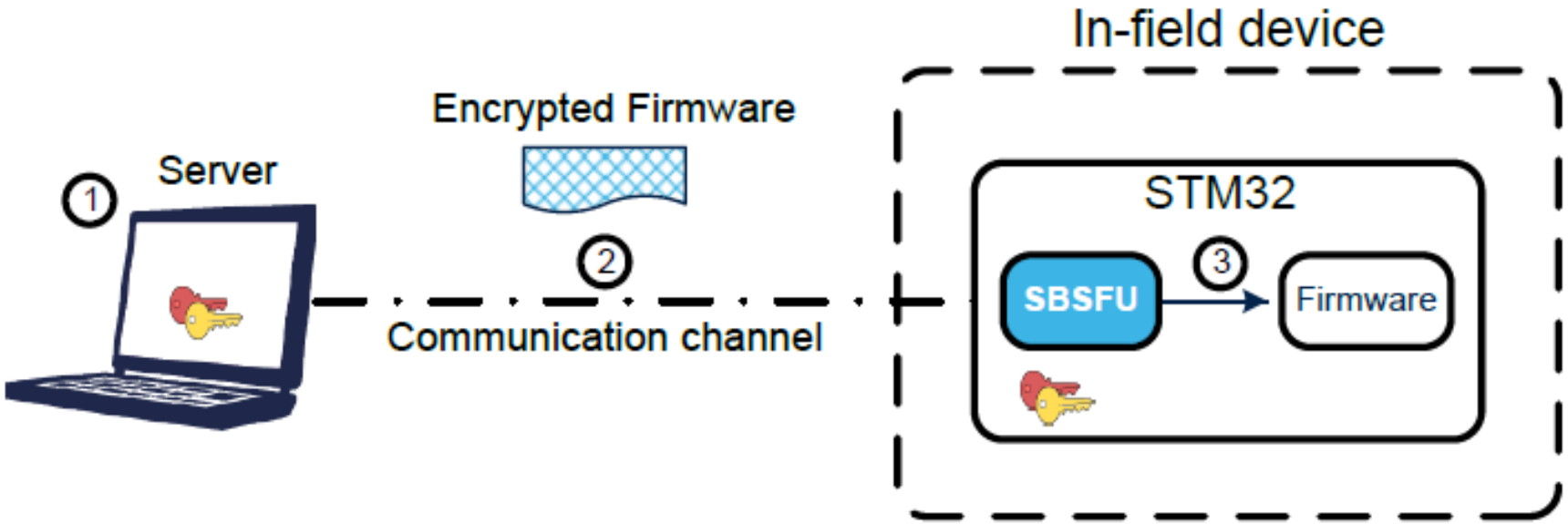
- 安全启动
  - 从正确的安全内存位置启动
  - 每个应用固件在执行之前都经过身份验证

- 安全更新
  - 安全固件更新允许以安全的方式更新新的固件版本
  - 新固件在更新之前将首先进行身份验证和解密

安全启动


Reset
Code
Secure Boot
1
Trusted
2
Authenticates
Code
3
Application

安全更新

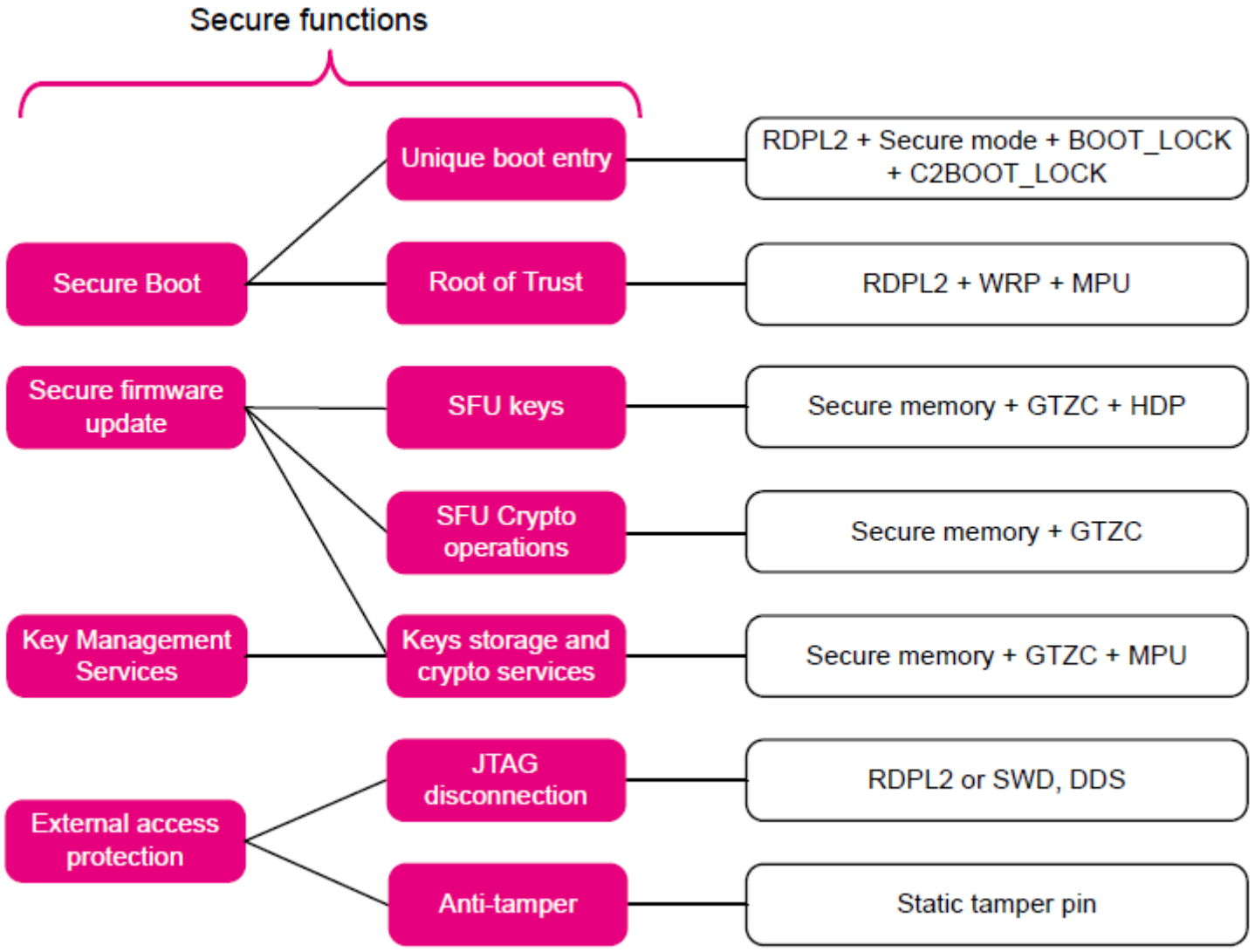

In-field device
Encrypted Firmware
Server
1
2
Communication channel
STM32
SBSFU
3
Firmware

SBSFU安全策略

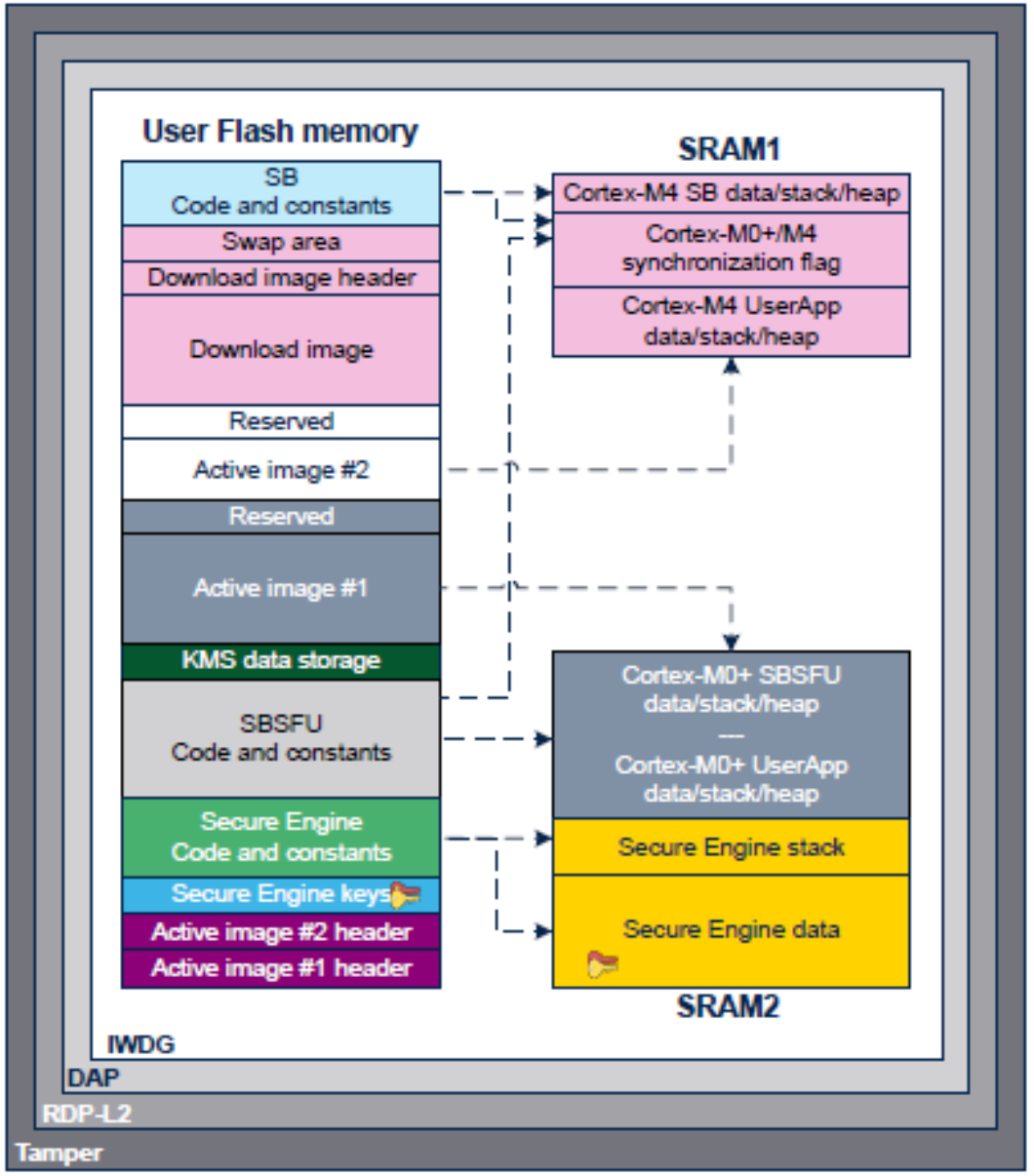
SBSFU安全策略基于以下概念：

- 通过BootLock实现唯一启动入口：执行Cortex®-M4 / Cortex®-M0+安全启动代码
- 两个核运行的安全代码以及加解密操作所需要的密钥不可更改
- 创建一个与 SBSFU 程序和用户应用程序隔离的执行环境，来存储机密信息(例如密钥)，以及运行关键操作(如运行加密算法)
- 在应用程序执行期间只能调用 SBSFU 提供的有限接口
- 禁止对设备的 JTAG 访问
- 监控系统：入侵检测和 看门狗

SBSFU安全保护


Secure functions
Secure Boot
Unique boot entry
Root of Trust
RDPL2 + Secure mode + BOOT_LOCK + C2BOOT_LOCK
RDPL2 + WRP + MPU
Secure firmware update
SFU keys
SFU Crypto operations
Key Management Services
Keys storage and crypto services
Secure memory + GTZC + HDP
Secure memory + GTZC
Secure memory + GTZC + MPU
External access protection
JTAG disconnection
Anti-tamper
RDPL2 or SWD, DDS
Static tamper pin

SBSFU执行保护



Legend

- Cortex-M4 + WRP + MPU-Privileged + MPU-RX
- Cortex-M4 + MPU-RW (code/data)
- Cortex-M4 + MPU-NA during boot

R: Read
W: Write
X: eXecute
NA: No Access

Note:

- *Cortex-M4 = Non-secure unprivileged (not executable by Cortex-M0+)*
- *All the Cortex-M4 area is MPU-RW for the Cortex-M0+ (but limited by WRP)*

- Cortex-M0+ + MPU-RW (code/data)
- Cortex-M0+ + MPU-Privileged + MPU-RW
- Cortex-M0+ + WRP + MPU-RX
- Cortex-M0+ + WRP + TZSC-Privileged + MPU-RX
- Cortex-M0+ + WRP + TZSC-Privileged + MPU-RX + HDP
- Cortex-M0+ + TZSC-Privileged + MPU-Privileged + MPU-RW + HDP
- Cortex-M0+ + TZSC-Privileged + MPU-RW (data)

Note:

- *Cortex-M0+ = Secure (not accessible by Cortex-M4)*

SBSFU防御外部攻击

防御外部攻击

- **RDP（读保护）**：设置RDP 2，以实现最高级别的保护并实现信任根：
  - 禁止通过 JTAG 硬件接口对 RAM 和flash进行外部访问。
  - 防止SBSFU 代码更改。
  - 选项字节不能更改。 这意味着不能再更改其他保护，例如 WRP。

- **Tamper（防篡改保护）**：防篡改保护用于检测设备上的物理篡改行为并采取相关对策。 在篡改检测的情况下，SBSFU 应用示例会强制重启。

- **DAP（ 调试访问端口 ）**： DAP 保护包括停用 DAP（调试访问端口）。停用后，JTAG 引脚不再连接到 STM32 设备内部总线。 RDP 2 会 自动禁用 DAP。

- **IWDG（看门狗）**：IWDG 是一个自由运行的递减计数器。 一旦运行，就无法停止。它必须复位之前定期刷新。该机制控制 SBSFU 执行持续时间。

SBSFU防御内部攻击

防御内部攻击

- **安全模式 (ESE = 1)：**设置后，安全flash和 RAM 只能被Cortex®-M0+ 内核， Cortex®-M4 内核无法访问。这可以保护限制对某些处于安全和/或特权模式外设的访问，如 AES、RNG、PKA、PWR、FLASHIF 和 DMA（DMA1、DMA2 DMAMUX）。默认情况下，SUBGHZSPI 可由安全的非特权应用程序访问；但可以限制成只允许安全特权应用程序的访问。

- **WRP（写保护）**：写保护用于保护可信代码免受外部攻击和内部修改，比如通过WRP保护SBSFU公钥不被修改。

- **TZSC（安全控制器）**：管理所有关键数据和操作（Secure Engine）的受保护环境, 通过利用 TZSC 与其他软件组件实现隔离。 只有通过安全特权级别的软件执行才能访问安全引擎（Secure Engine）代码和数据。这种对安全引擎（Secure Engine）服务和资源的严格访问控制，是通过GTZC 和 MPU实现的，如下表所述。

Region content	Secure Privileged permission	Non-Secure and/or unprivileged permission
Secure Engine code and constants	Read only (execution allowed)	No access
Secure Engine stack and data	Read write (not executable)	No access

SBSFU防御内部攻击

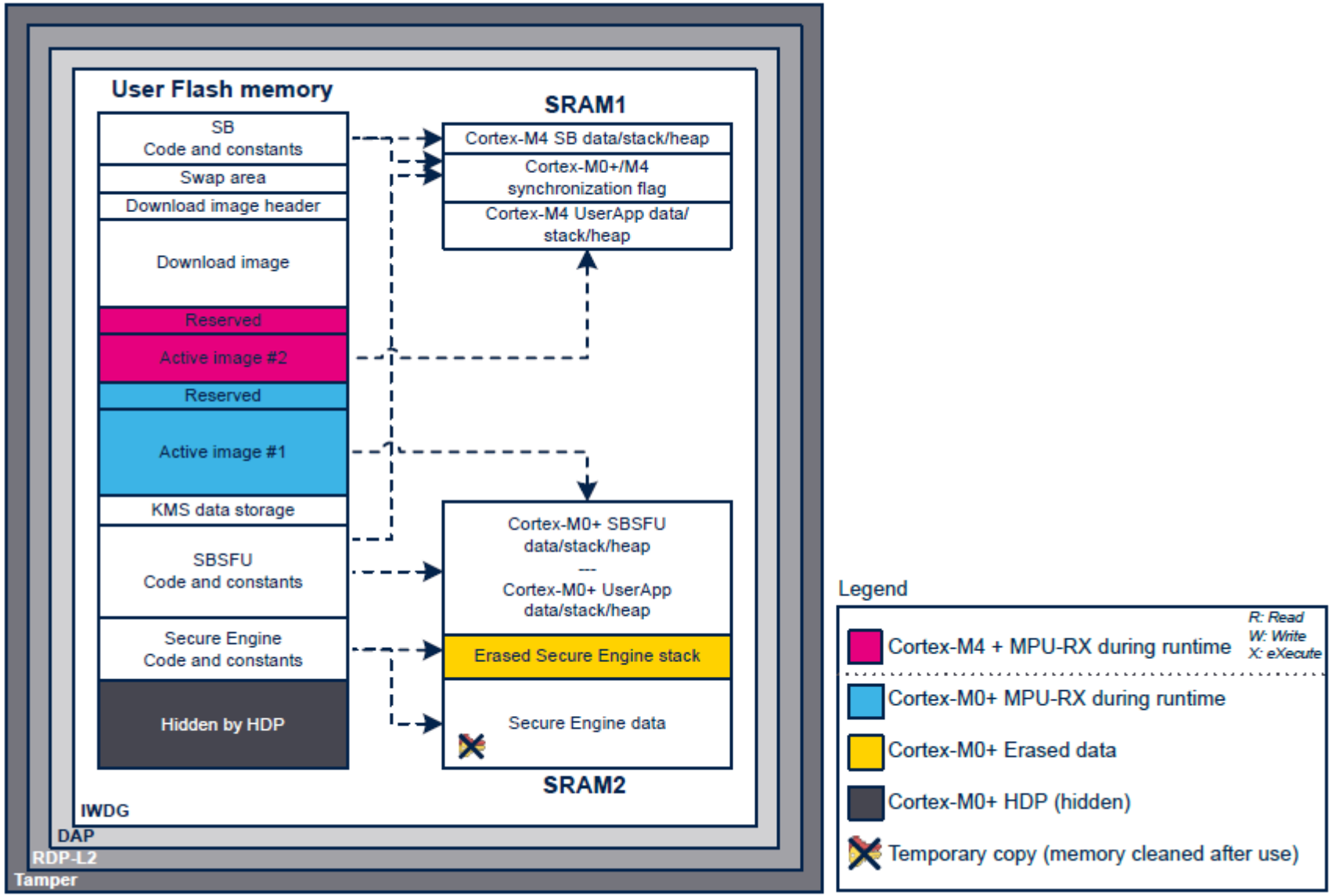
防御内部攻击

- **TZIC（安全非法访问控制器）**为每个检测到的非法访问配置中断，从而可以根据不同访问进行响应。

- **MPU（内存保护单元）：** MPU 用于设置flash和 SRAM 的访问权限，使嵌入式系统更加健壮。
  - 在 SBSFU 应用示例中，MPU 可以确保在 SBSFU 代码执行期间不会从任何内存执行其他代码。
  - 当离开 SBSFU 应用程序时，MPU 配置会更新，以授权用户应用程序代码后续的执行。
  - 在运行用户应用程序时（不仅在运行 SBSFU 代码时），安全引擎（Secure Engine）隔离设置和主管调用机制（TZSC 相关）仍然适用。

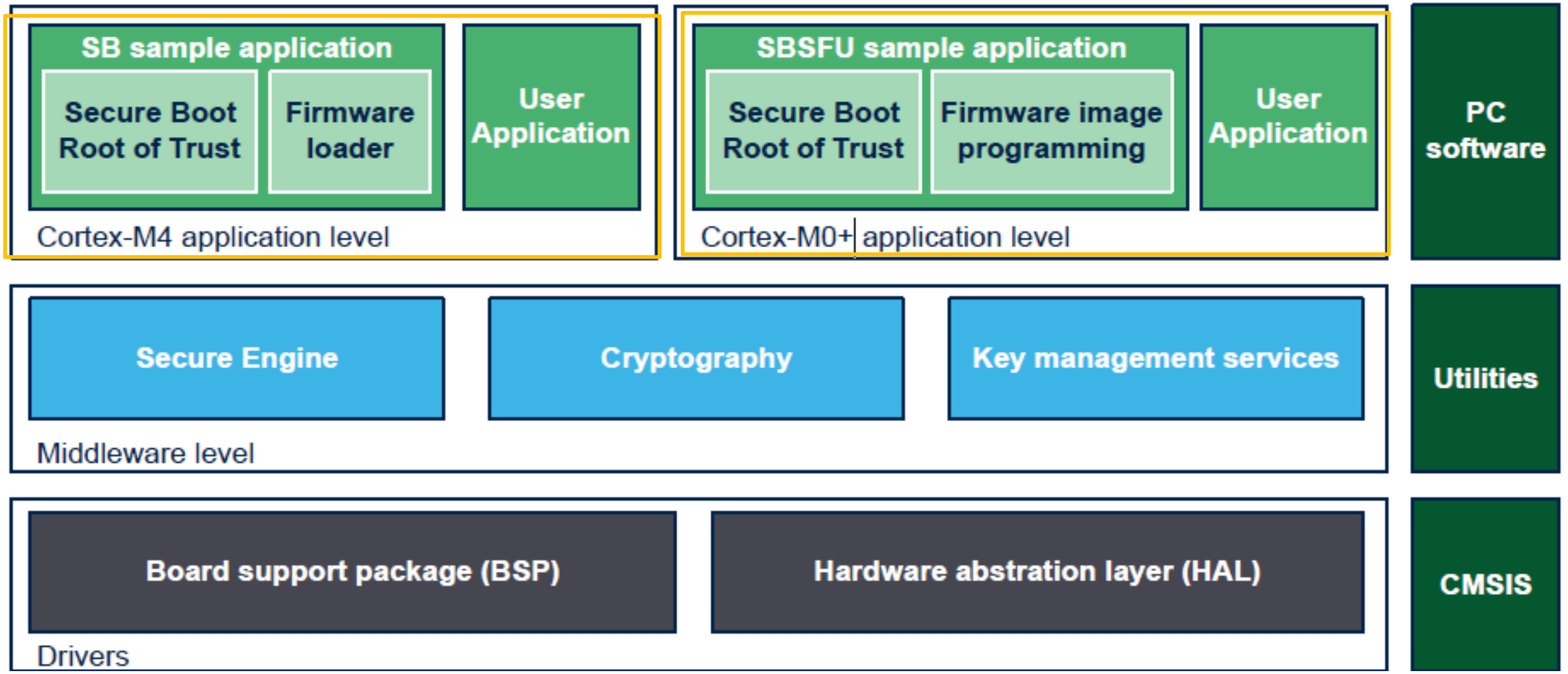
- **HDP（隐藏保护）：** 用户应用程序运时SBSFU密钥和应用程序的标头将被隐藏并且无法再被访问。

应用程序运行时保护


User Flash memory
SB
Code and constants
Swap area
Download image header
Download image
Reserved
Active image #2
Reserved
Active image #1
KMS data storage
SBSFU
Code and constants
Secure Engine
Code and constants
Hidden by HDP
SRAM1
Cortex-M4 SB data/stack/heap
Cortex-M0+/M4
synchronization flag
Cortex-M4 UserApp data/
stack/heap
Cortex-M0+ SBSFU
data/stack/heap

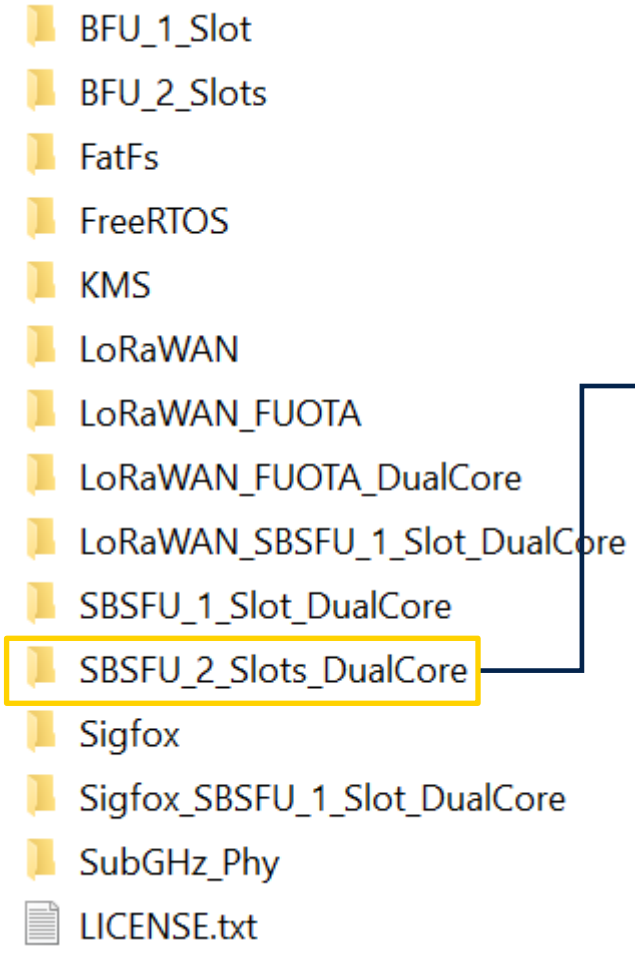
Cortex-M0+ UserApp
data/stack/heap
Erased Secure Engine stack
Secure Engine data
SRAM2
IWDG
DAP
RDP-L2
Tamper
Legend
R: Read
W: Write
X: eXecute
Cortex-M4 + MPU-RX during runtime
Cortex-M0+ MPU-RX during runtime
Cortex-M0+ Erased data
Cortex-M0+ HDP (hidden)
Temporary copy (memory cleaned after use)

SBSFU软件架构


SB sample application
Secure Boot Root of Trust
Firmware loader
User Application
Cortex-M4 application level
SBSFU sample application
Secure Boot Root of Trust
Firmware image programming
User Application
Cortex-M0+ application level
PC software
Secure Engine
Cryptography
Key management services
Middleware level
Utilities
Board support package (BSP)
Hardware abstration layer (HAL)
Drivers
CMSIS

STM32Cube_FW_WL例程

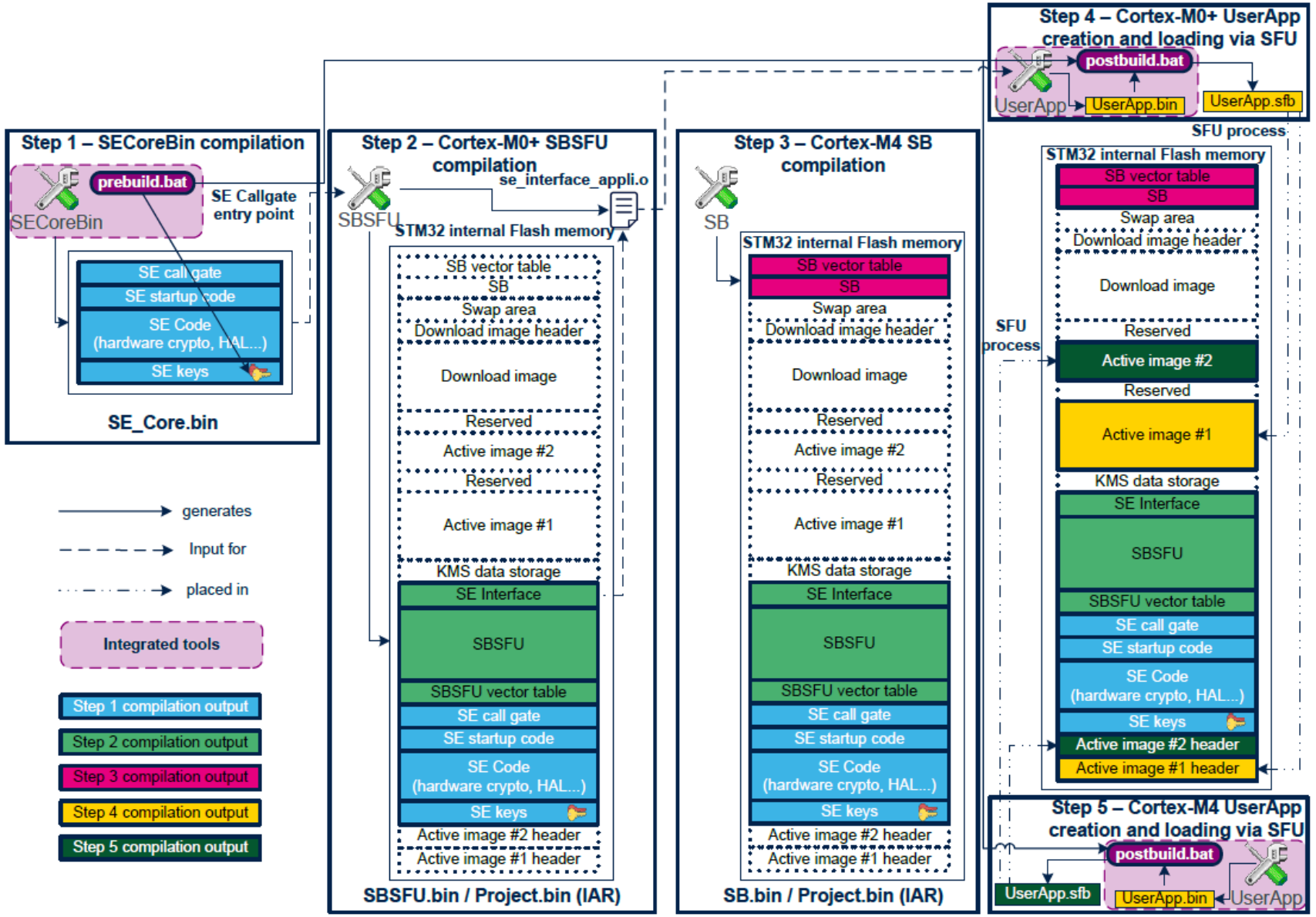
STM32Cube_FW_WL_V1.1.0\Projects\NUCLEO-WL55JC\Applications



- 2_Images_KMS_Blob：生成要下载的密钥管理服务 Blob 二进制文件，通过 ImportBlob() API 使用 KMS
- 2_Images_SBSFU：主要负责安全启动和安全升级的流程管理
- 2_Images_SECoreBin：定义加解密方案，为保护密钥和加解密操作提供安全的环境
- 2_Images_UserApp_M0Plus
- 2_Images_UserApp_M4：双核应用程序, 通过postbuild脚本，生成UserApp.sfb文件，和SBSFU_UserApp.bin
- Linker_Common

life.augmented

SBSFU 项目间的耦合



SBSFU 安全配置过程

M0+/M4 app_sfu_common.h

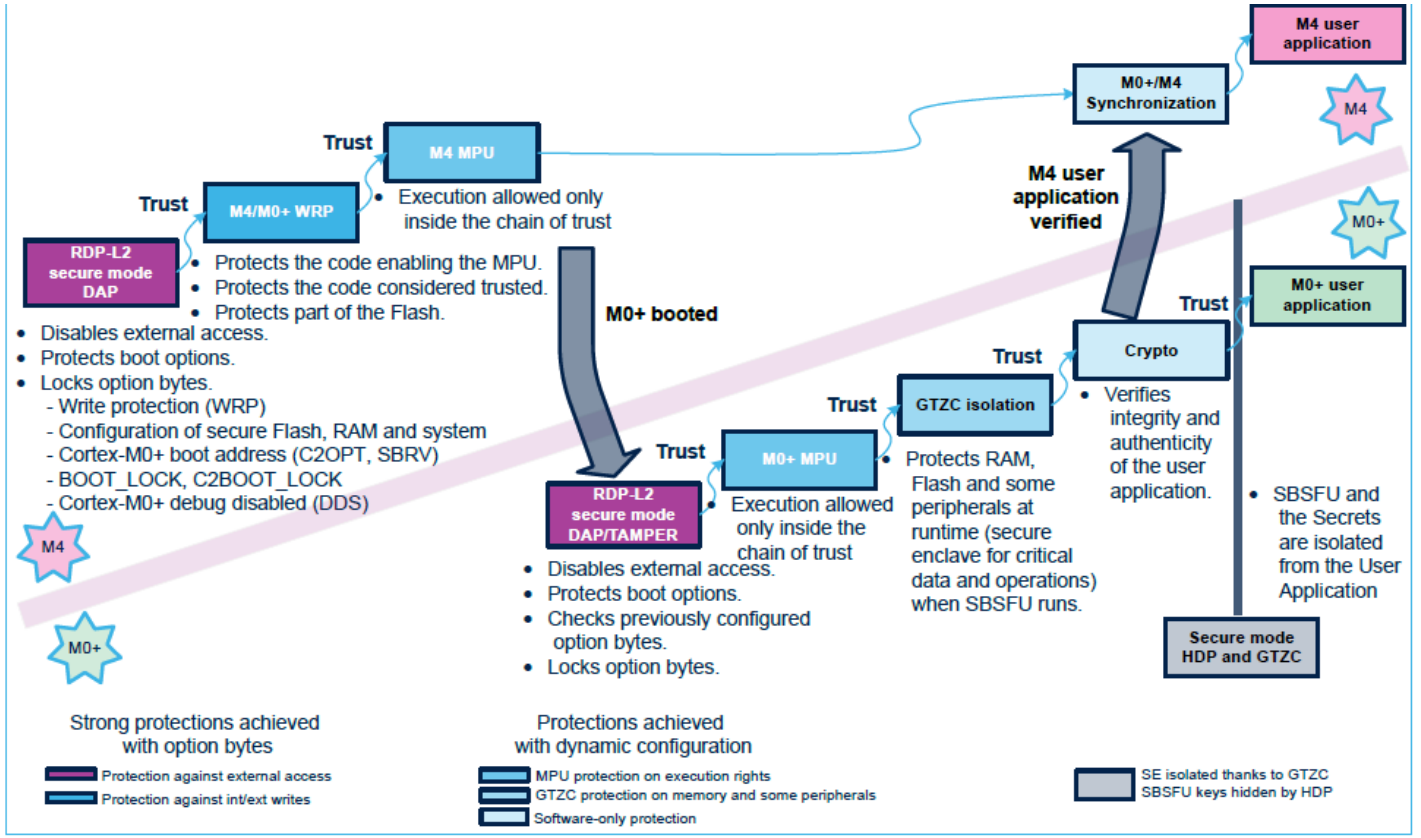
```
/*#define SECBOOT_DISABLE_SECURITY_IPS*/  /*!< Disable all security IPs at once when activated */
#if !defined(SECBOOT_DISABLE_SECURITY_IPS)
#define SFU_WRP_PROTECT_ENABLE
#define SFU_DAP_PROTECT_ENABLE
#define SFU_DMA_PROTECT_ENABLE
#define SFU_IWDG_PROTECT_ENABLE
#define SFU_C2_DDS_PROTECT_ENABLE
#define SFU_SECURE_USER_PROTECT_ENABLE
#endif /* !SECBOOT_DISABLE_SECURITY_IPS */
```

M4 app_sfu.h

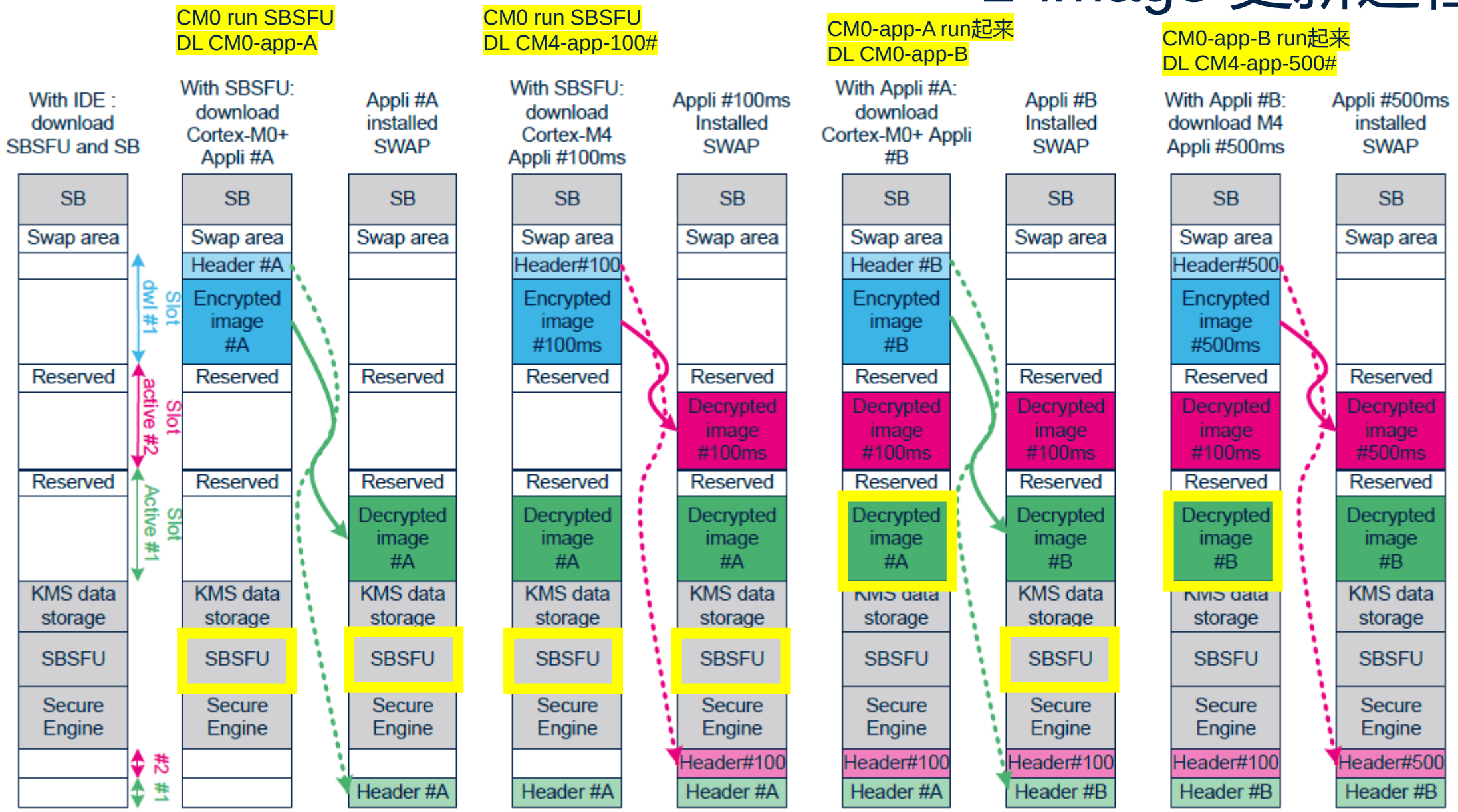
```
#if !defined(SECBOOT_DISABLE_SECURITY_IPS)
#define SFU_MPU_PROTECT_ENABLE
#define SFU_MPU_USERAPP_ACTIVATION
#endif /* !SECBOOT_DISABLE_SECURITY_IPS */
```

M0+ app_sfu.h

```
#if !defined(SECBOOT_DISABLE_SECURITY_IPS)
#define SFU_RDP_PROTECT_ENABLE
#define SFU_MPU_PROTECT_ENABLE
#define SFU_MPU_USERAPP_ACTIVATION
#define SFU_GTZC_PROTECT_ENABLE
#define SFU_C2SWDBG_PROTECT_ENABLE
#endif /* !SECBOOT_DISABLE_SECURITY_IPS */
```



2-image 更新过程


With IDE : download SBSFU and SB
CM0 run SBSFU
DL CM0-app-A
With SBSFU: download Cortex-M0+ Appli #A
Appli #A installed SWAP
CM0 run SBSFU
DL CM4-app-100#
With SBSFU: download Cortex-M4 Appli #100ms
Appli #100ms Installed SWAP
CM0-app-A run起来
DL CM0-app-B
With Appli #A: download Cortex-M0+ Appli #B
Appli #B Installed SWAP
CM0-app-B run起来
DL CM4-app-500#
With Appli #B: download M4 Appli #500ms
Appli #500ms installed SWAP
SB
Swap area
Header #A
Encrypted image #A
Reserved
Decrypted image #A
KMS data storage
SBSFU
Secure Engine
Header #A
Header#100
Encrypted image #100ms
Decrypted image #100ms
Header#100
Header #B
Encrypted image #B
Decrypted image #B
Header#500
Encrypted image #500ms
Decrypted image #500ms
Header#500
Slot dwl #1
active #2
Slot
Active #1
Slot
#2
#1

例程操作步骤一

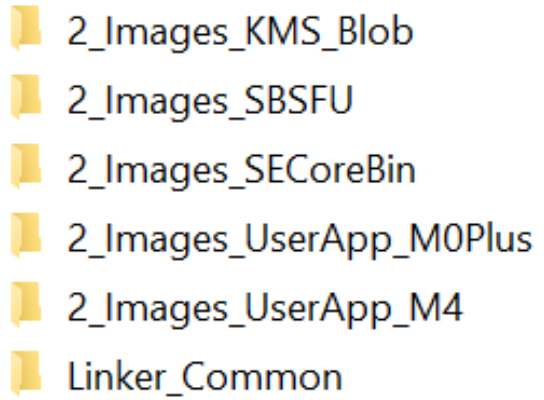
1按以下顺序编译项目。 这是强制性的，因为每个项目都需要一些由前一个项目编译生成的对象：

1, - 2_Images_SECoreBin (see also SECoreBin/readme.txt)

2,- 2_Images_SBSFU (using both CM4 and CM0+ workspaces)

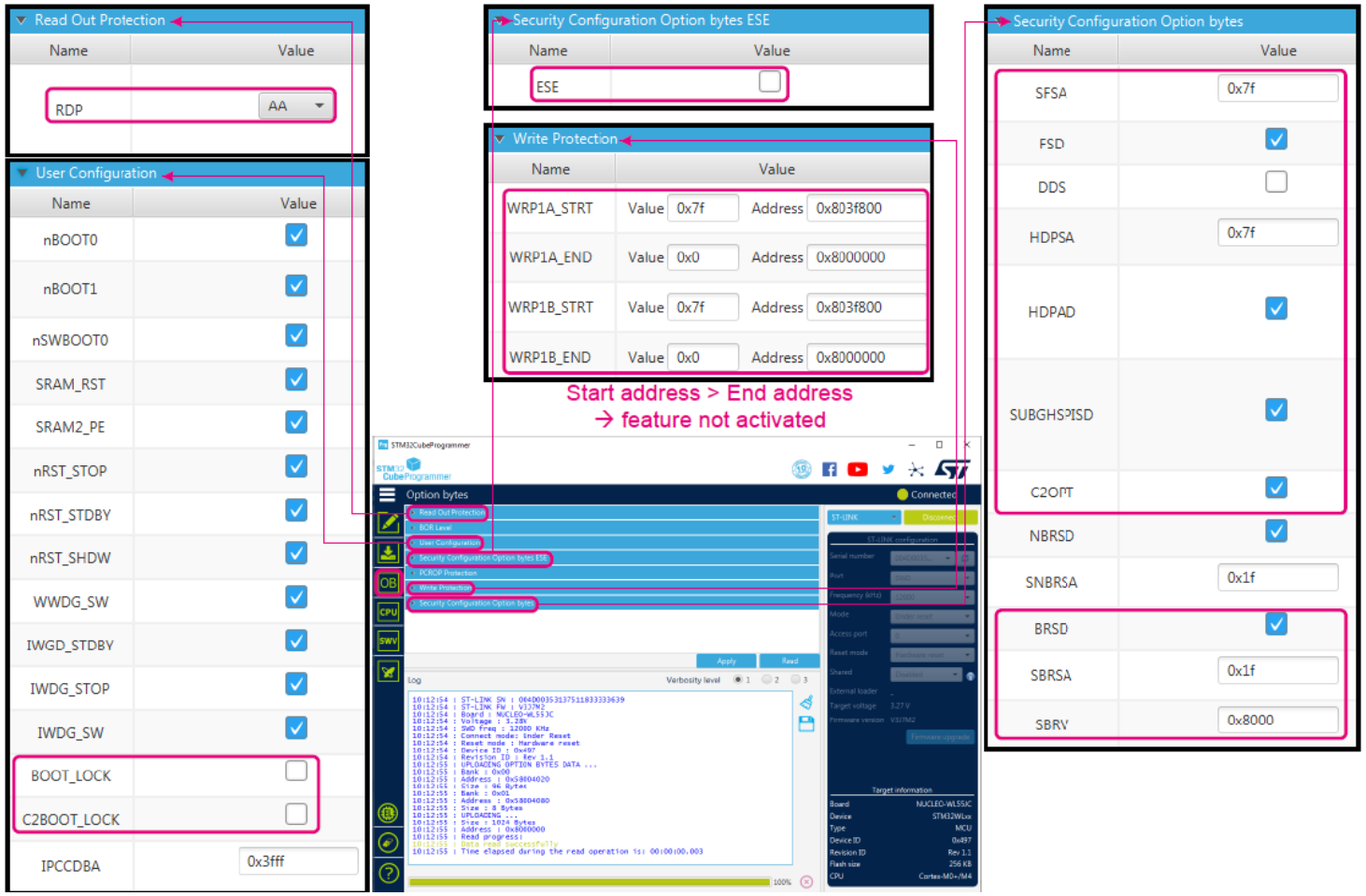
3, - 2_Images_UserApp_M0Plus (see also UserApp_M0Plus/readme.txt)

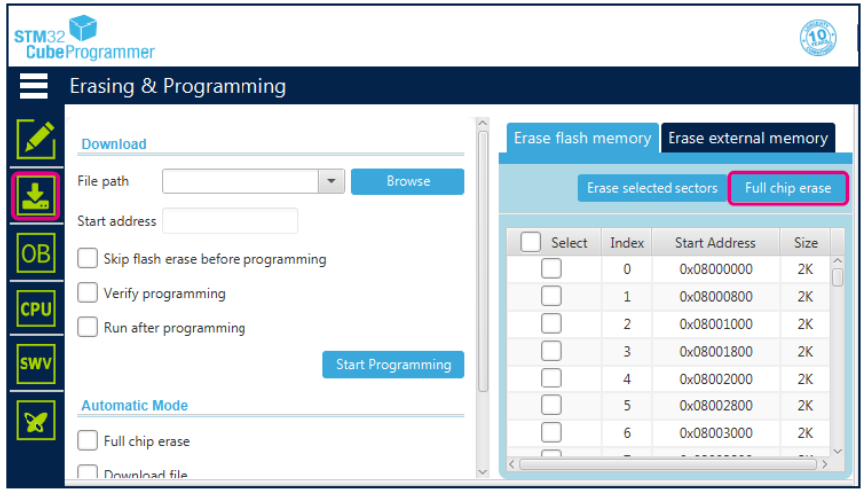
4, - 2_Images_UserApp_M4 (see also UserApp_M4/readme.txt)



life.augmented

步骤二

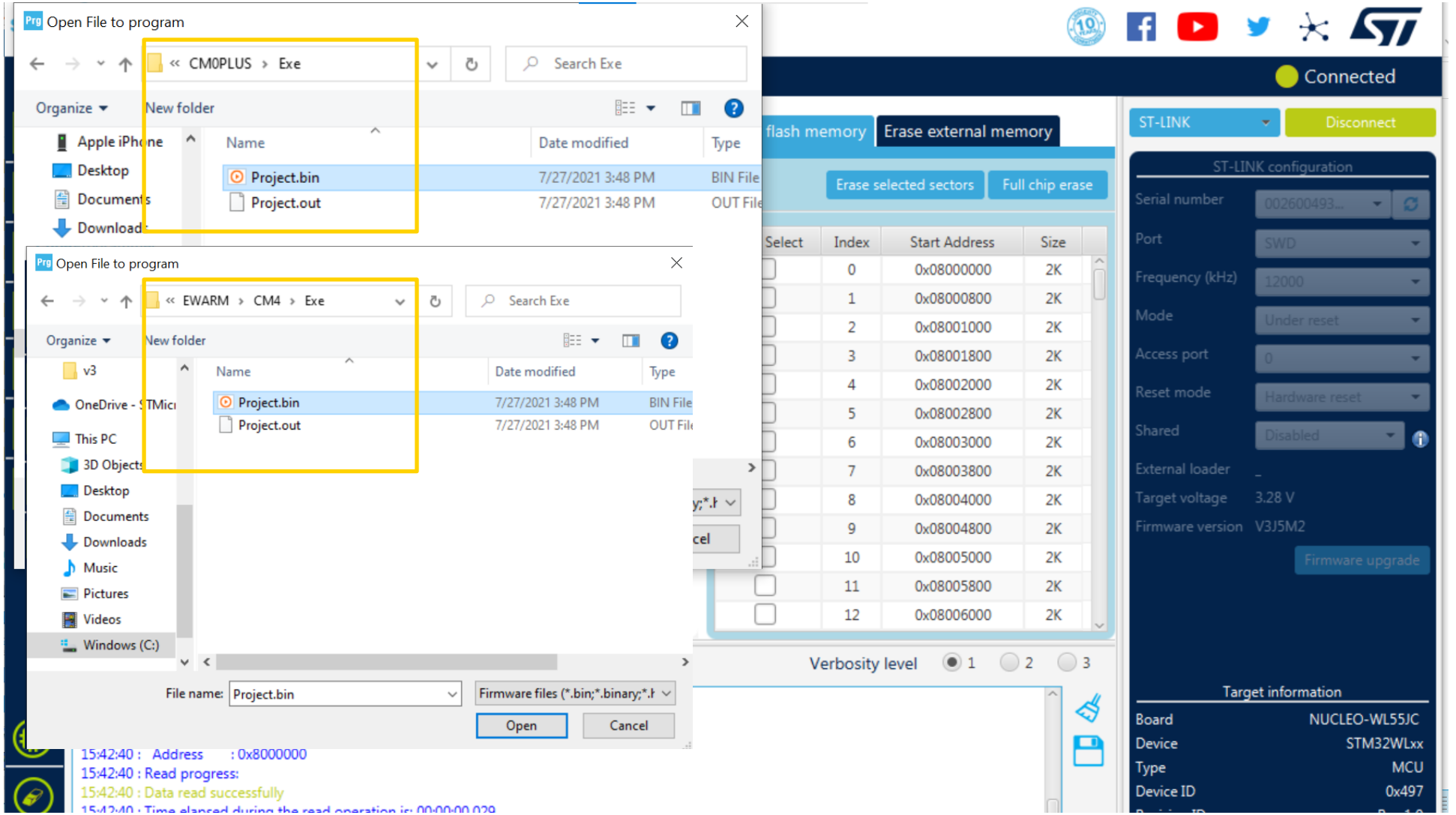




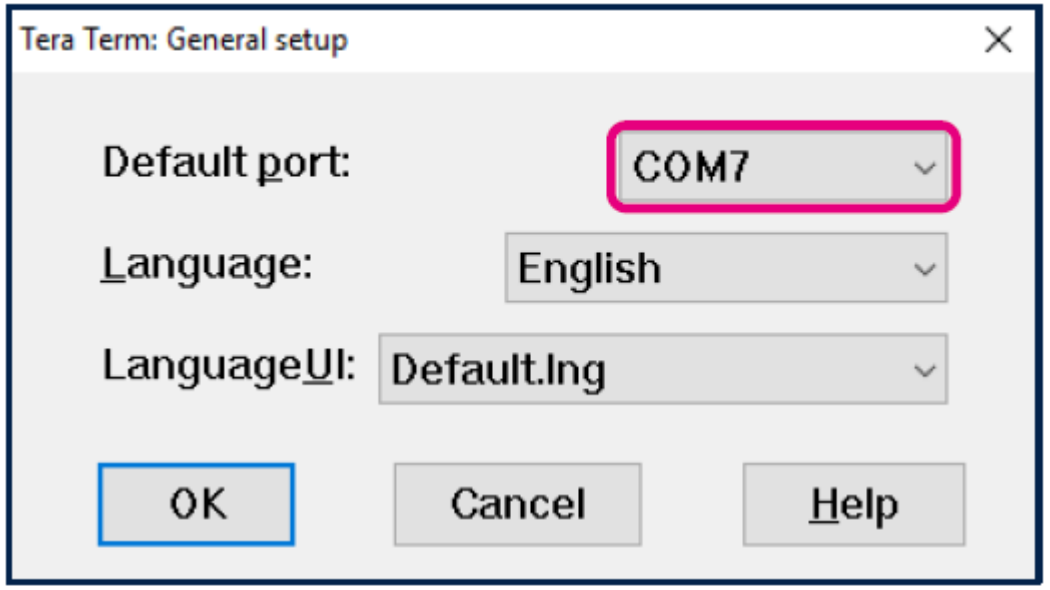
life.augmented

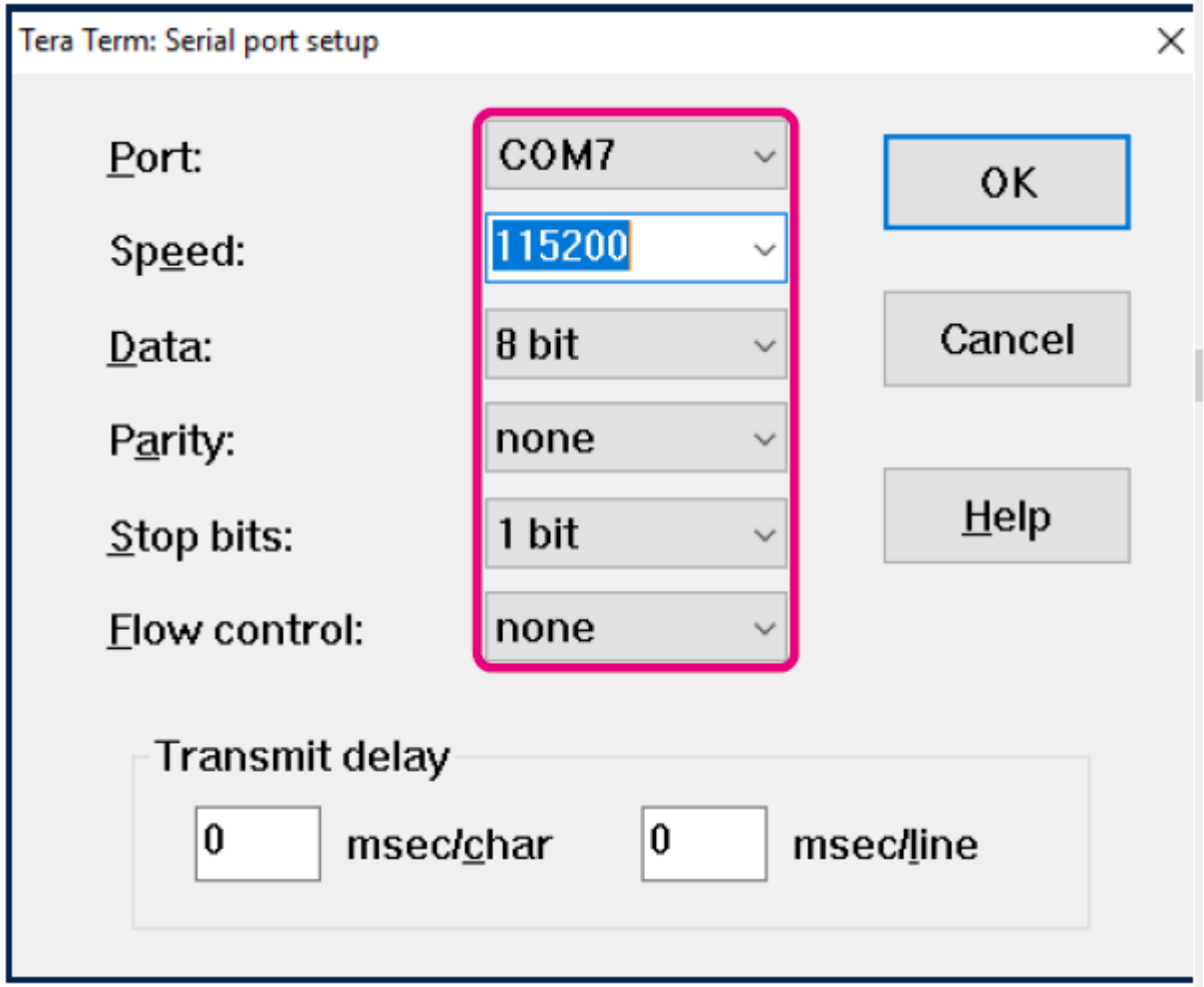
步骤三

下载相应的SBSFU文件到CM4和CM0+



步骤四


Tera Term: General setup
Default port:
COM7
Language:
English
LanguageUI:
Default.lng
OK
Cancel
Help


Tera Term: Serial port setup
Port:
COM7
Speed:
115200
Data:
8 bit
Parity:
none
Stop bits:
1 bit
Flow control:
none
OK
Cancel
Help
Transmit delay
0
msec/char
0
msec/line

步骤四

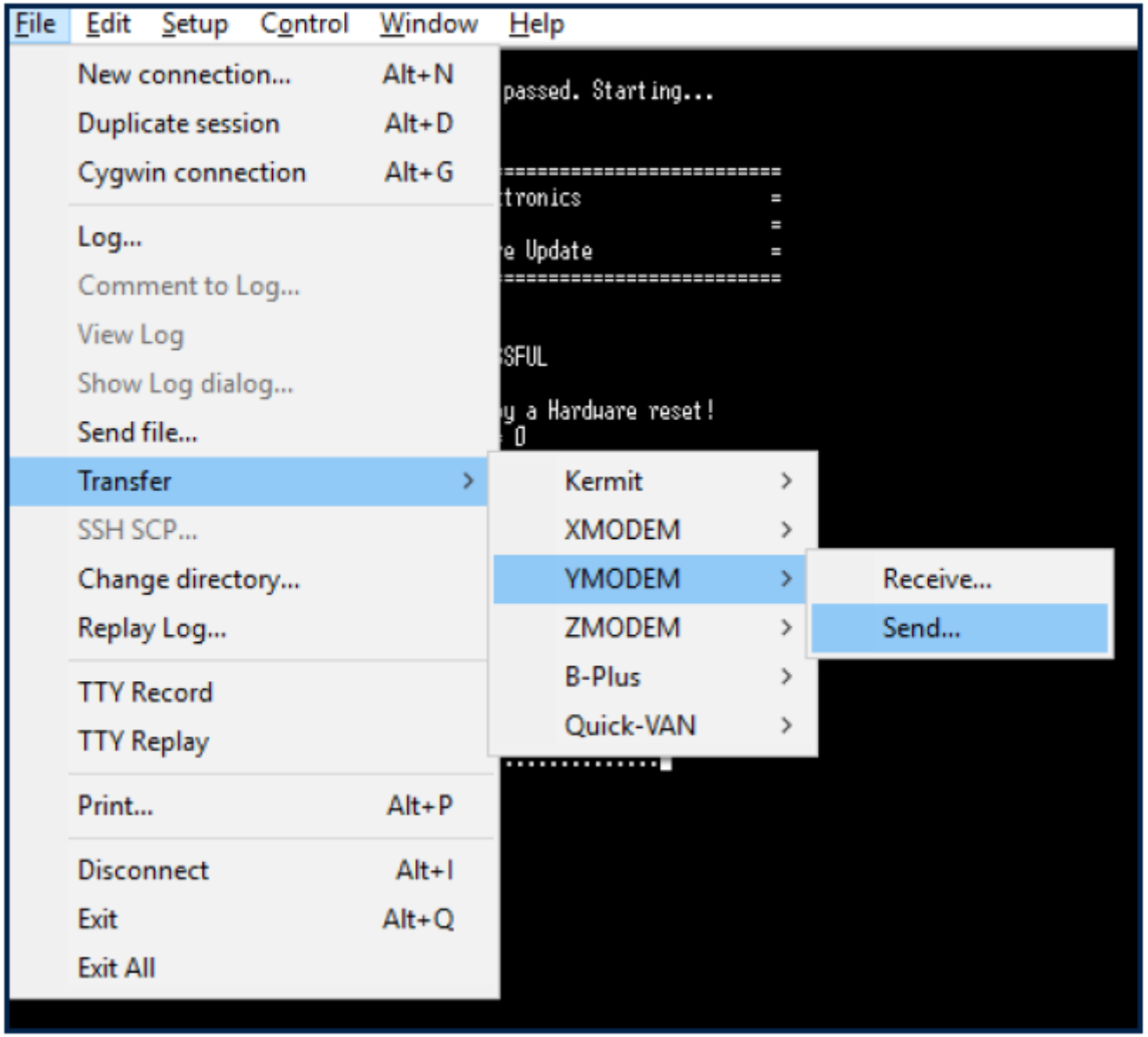
SBSFU log

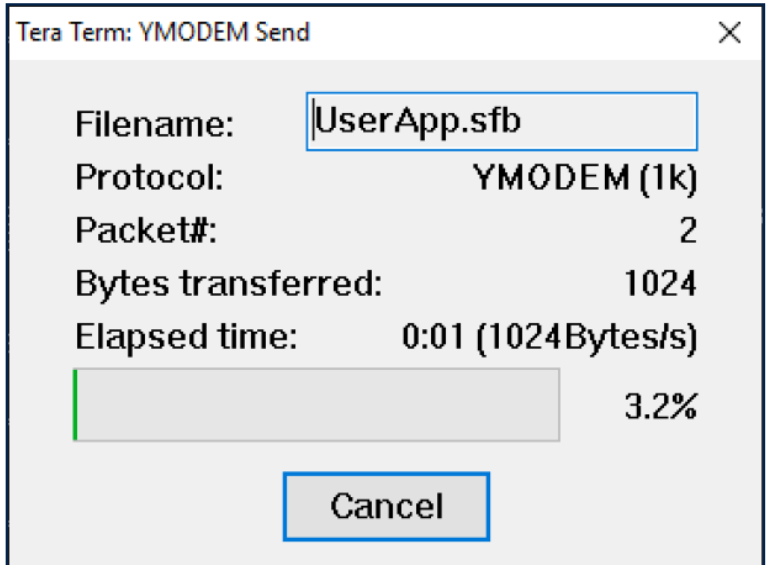
```
= [SBOOT] System Security Check successfully passed. Starting...


=================================================================
=              (C) COPYRIGHT 2017 STMicroelectronics            =
=                                                               =
=              Secure Boot and Secure Firmware Update           =
=================================================================


= [SBOOT] SECURE ENGINE INITIALIZATION SUCCESSFUL
= [SBOOT] STATE: CHECK STATUS ON RESET
          INFO: A Reboot has been triggered by a Hardware reset!
          Consecutive Boot on error counter = 0
          INFO: Last execution detected error was: No error. Success.
= [SBOOT] STATE: CHECK NEW FIRMWARE TO DOWNLOAD
= [SBOOT] STATE: CHECK KMS BLOB TO INSTALL
= [SBOOT] STATE: CHECK USER FW STATUS
          No valid FW found in the active slots nor new FW to be installed
          No valid FW and no local loader: execution stopped.

=================================================================
=                           Loader                              =
=================================================================

          File> Transfer> YMODEM> Send ......
```

life.augmented

步骤五

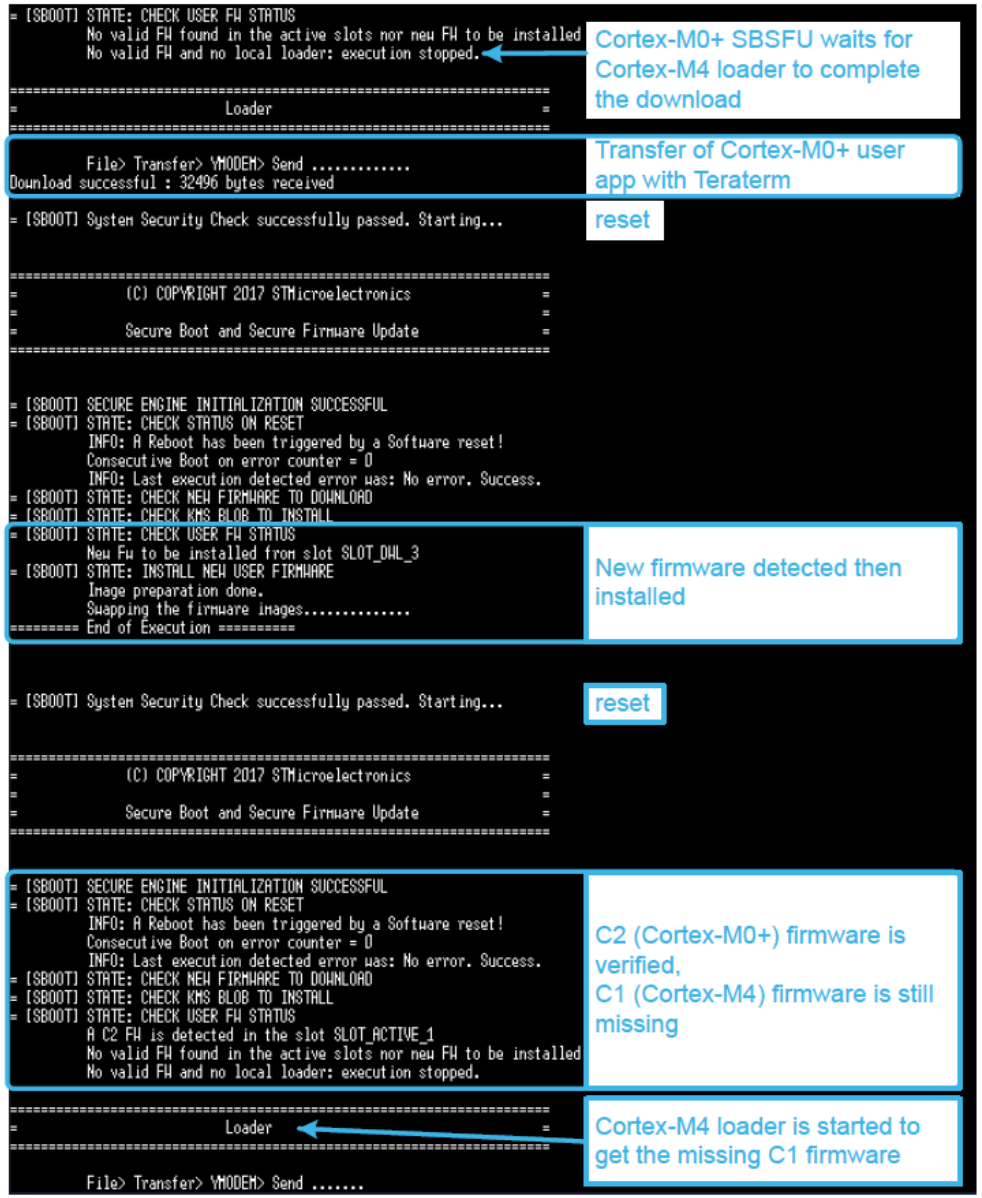
SBSFU 加密固件传输,先传输CM0+固件，再传输CM4固件





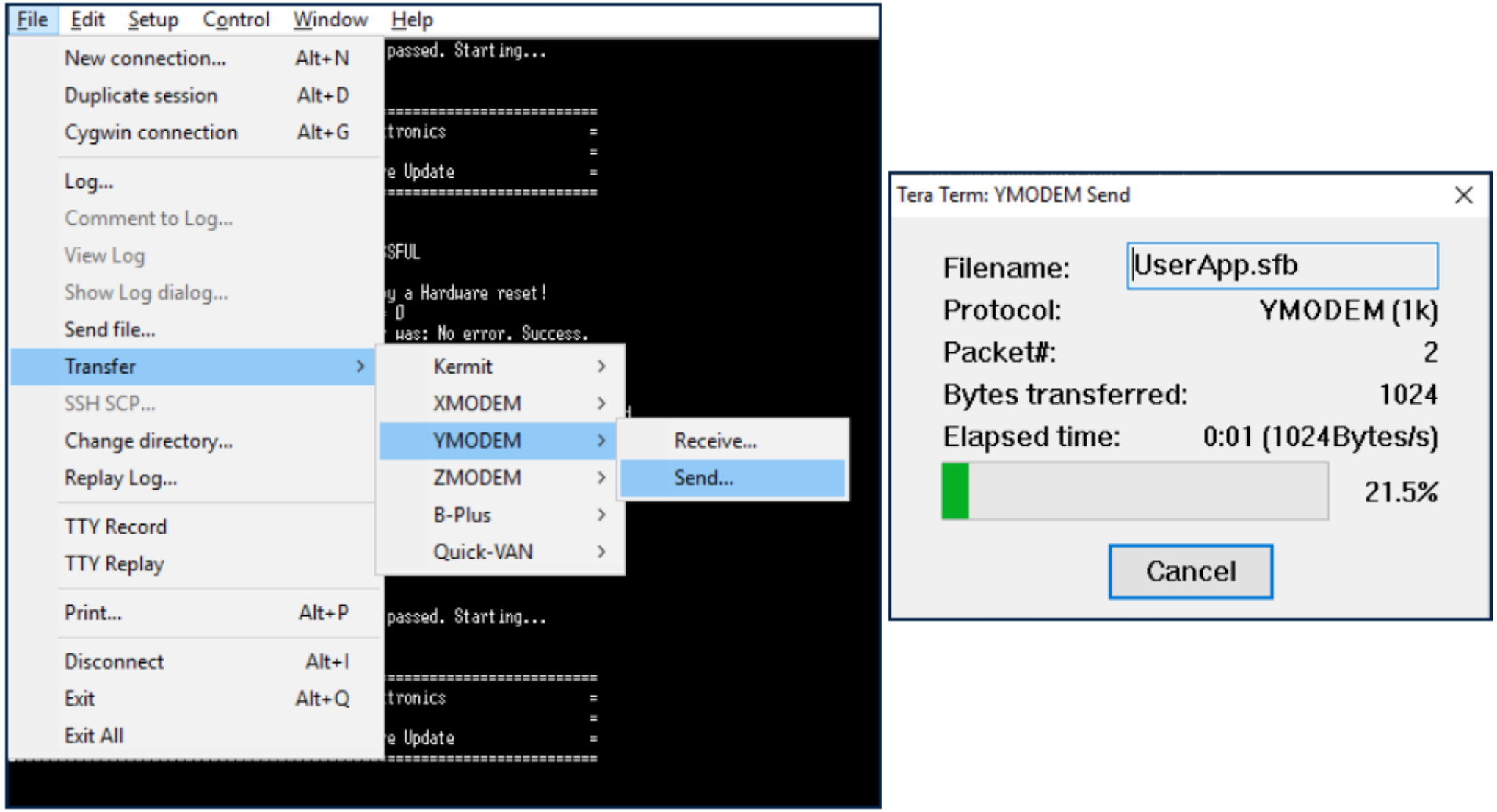
步骤六

SBSFU 第一个加密固件(CM0+)传输完成



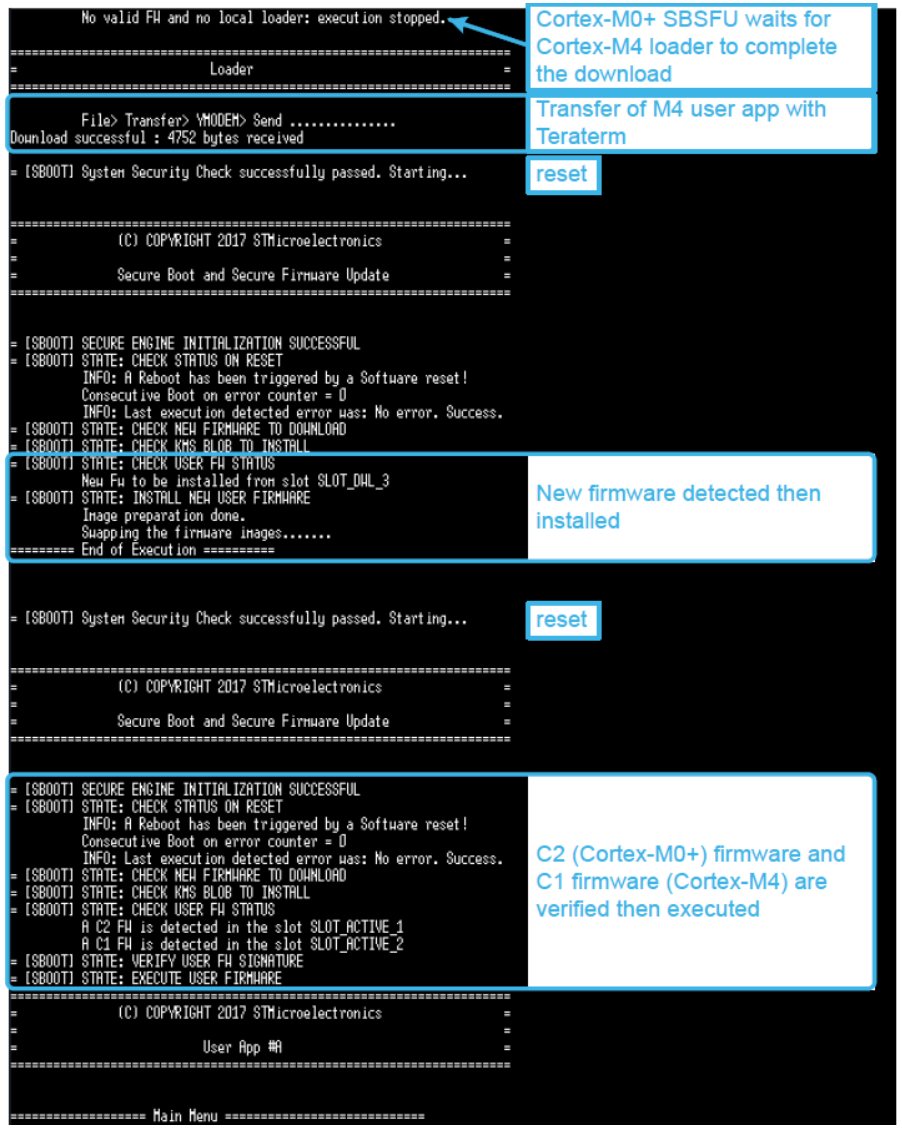
SBSFU 第二个加密固件(CM4)传输

步骤七



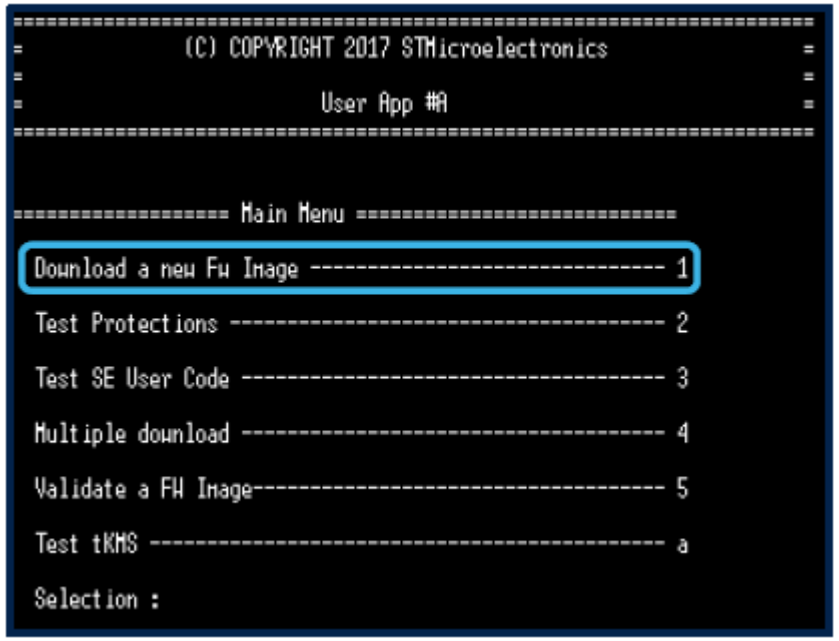
步骤八

SBSFU 第二个加密固件(cm4)传输完成



步骤九

SBSFU 更新CM0+固件



SBSFU 测试保护项目

```
================== Test Menu ============================
 Test : CORRUPT ACTIVE IMAGE ------------------------- 1
 Test SE isolation - CODE ---------------------------- 2
 Test SE isolation - VDATA --------------------------- 3
 Test Protection: HDP -------------------------------- 4
 Test Protection: HRP -------------------------------- 5
 Test Protection: IWDG ------------------------------- 6
 Test Protection: TAMPER ----------------------------- 7
 Test Protection: GTZSC ------------------------------ 8
 Test KMS_DataStorage isolation ---------------------- 9
 Test synchronization flag protection ---------------- a
 Previous Menu --------------------------------------- x
 Selection :
```

STM32WL5x安全功能SFI

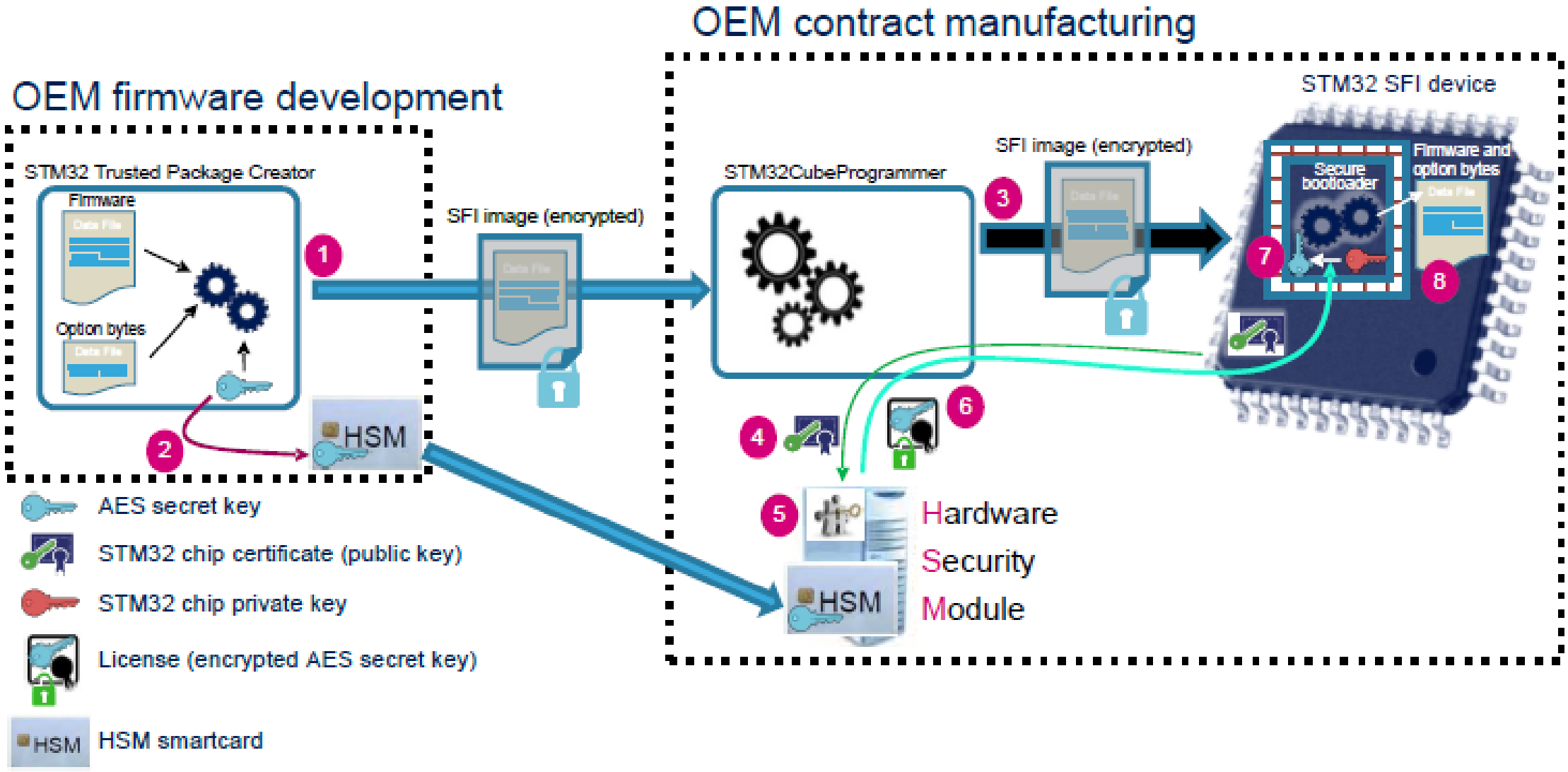
SFI概览

- SFI的目标
  - 确保客户研发的固件在第三方(不可靠)产线上不会泄露
  - 控制客户研发的固件被烧录的次数

- SFI的方法
  - 使用签名过的证书对目标芯片进行验证 → 确保是STM32
  - 烧录器外部总线上传输的是固件密文 → 保证OEM固件的保密性
  - 使用License文件 → 控制烧录的次数

- SFI适用于STM32WL5X双核系列

SFI 操作过程



life.augmented

总结

总结

STM32WL 安全特性概览

STM32WL 通用安全功能特性

- BL,WRP,RDP,PCROP,MPU

STM32WL STM32WL5双核安全特性

- 安全存储（CM0+），安全外设/特权访问(GTZC)

STM32WL三大安全功能

- KMS， SBSFU，SFI

Thank you

life.augmented