

引言

X-CUBE-SBSFU安全启动和安全固件更新解决方案允许使用新固件版本更新STM32微控制器内置程序、添加新功能并更正潜在问题。更新过程以安全方式执行，以防止未经授权的更新以及访问设备上的机密数据。

安全启动（可信根服务）是一种不可变代码，总是在系统复位后执行。它检查STM32静态保护、激活STM32运行时保护措施，然后在每次执行前验证应用程序代码的真实性和完整性，以确保无效或恶意代码不能运行。

安全固件更新应用程序通过使用Ymodem协议的UART接口接收固件镜像。它会在安装代码之前检查其真实性和完整性。固件更新可更新整个固件镜像，或者部分镜像。示例代码可灵活配置，可配置使用对你或非对称加密方案，采用明文或密文方案。工程以两种方案提供：

- 单插槽配置，以使固件镜像尺寸最大化
- 双插槽配置，以确保安全镜像安装，并启用物联网器件中常用的OTA固件更新功能。

对于具有多个固件（如协议栈、中间件和用户应用程序）的复杂系统，固件镜像配置最多可扩展到三个固件镜像。

安全密钥管理服务通过PKCS #11 API（基于KEY ID的API）向用户应用程序提供加密服务，这些API在受保护和隔离的环境中执行。用户应用程序密钥存储在受保护和隔离的环境中，以便进行安全更新：真实性检查、数据解密和数据完整性检查。

STSAFE-A110是一种防篡改安全元件（通过硬件通用标准EAL5+认证），用于托管X509证书和密钥，并在安全启动和安全固件更新过程中执行用于固件镜像身份确认的验证。

X-CUBE-SBSFU用户手册（UM2262）讲解X-CUBE-SBSFU入门知识并详细介绍SBSFU功能。该应用笔记描述了如何调整X-CUBE-SBSFU并将其与用户应用程序集成；它回答了诸如以下问题：

- 如何将X-CUBE-SBSFU移植到另一块板？
- 如何根据用户的需求微调X-CUBE-SBSFU配置？
- 如何生成新的固件加密密钥？
- 如何调试X-CUBE-SBSFU？
- 如何调整SBSFU？
- 如何调整用户的应用程序？

注：
在该应用笔记中，使用IAR嵌入式Workbench[®] IDE作为示例，为项目配置提供指南。安全启动和安全固件更新应用程序被称为SBSFU。

注：
在名称为1_Image 的示例中对单插槽配置进行了演示。
在名为2_Images 的示例中对双插槽配置进行了演示。



目录

1	概述	6
2	相关文档	8
3	移植X-CUBE-SBSFU到另一块板上	9
3.1	硬件适配	9
3.2	内存映射定义	10
3.2.1	SBSFU区域定义参数	13
3.2.2	固件镜像插槽定义参数	14
3.2.3	特定项目的链接文件	16
3.2.4	多镜像配置	17
3.3	双核适配	18
4	SBSFU配置	20
4.1	要配置的特性	20
4.2	加密方案选择	21
4.3	安全配置	22
4.4	开发或生产模式配置	25
5	生成加密密钥	27
5.1	生成新的固件AES加密密钥	27
5.2	生成用于固件验证的新的公钥/私钥对 (ECDSA) 密钥对	27
5.3	STM32WB系列特殊性	28
5.4	KMS特殊性	29
5.5	STSAFE-A110特殊性	30
6	调试技巧	31
6.1	编译器优化级别	31
6.2	内存映射调整	31
6.3	调试SECoreBin	32
7	调整SBSFU	34
7.1	实现一种新的SBSFU加密方案	34

7.2	优化内存映射	36
7.3	如何激活隔离防火墙内部的中断管理环境	38
7.4	如何改善启动时间	39
8	调整用户应用程序	40
8.1	如何使应用程序兼容SBSFU	40
8.2	使用闪存存储用户数据	43
8.3	更改用户应用程序中的固件下载功能	44
8.4	如何变更固件版本	45
8.5	如何验证固件镜像	45
9	版本历史	47

表格索引

表1.	缩略语列表	6
表2.	术语列表	6
表3.	裁减SBSFU代码量	36
表4.	文档版本历史	47
表5.	中文文档版本历史	48

图片索引

图1.	SBSFU项目结构	9
图2.	闪存映射示例 (NUCLEO-L476RG)	10
图3.	链接器文件架构	11
图4.	具有MPU隔离的映射约束 (NUCLEO-G071RB示例)	12
图5.	映射约束, 面向用户应用程序执行	12
图6.	SBSFU区域 (NUCLEO-L476RG mapping_sbsfu.icf)	13
图7.	固件镜像插槽定义参数 (NUCLEO-L476RG mapping_fwimg.icf)	14
图8.	双存储区产品的防火墙配置约束	15
图9.	交换存储区之后的防火墙配置	15
图10.	特定于SECoreBin的链接文件	16
图11.	特定于SBSFU的链接文件	16
图12.	特定于UserApp的链接文件 (NUCLEO-L476RG示例)	17
图13.	多镜像配置	18
图14.	STM32H7系列双核适配	19
图15.	SBSFU配置	21
图16.	切换加密方案	22
图17.	STM32L4系列和STM32L0系列 安全配置 (app_sfuh.h)	23
图18.	STM32F4系列、STM32F7系列和STM32L1系列安全配置 (app_sfuh.h)	23
图19.	STM32G0系列、STM32G4系列和STM32H7系列安全配置 (app_sfuh.h)	24
图20.	STM32WB系列安全配置 (app_sfuh.h)	24
图21.	选项字节管理	26
图22.	新固件加密密钥	27
图23.	新私钥/公钥	28
图24.	密钥配置	29
图25.	KMS特殊性	30
图26.	STSAFE-A110密钥对	30
图27.	编译器优化	31
图28.	内存映射调整	32
图29.	检查WRP保护	32
图30.	在SECoreBin内进行调试	33
图31.	用户的加密方案实现	34
图32.	NUCLEO-G071RB上的内存映射优化示例 – 2个镜像	37
图33.	IDE调整	38
图34.	启动时间	39
图35.	向量表位置更新 (NUCLEO-L476RG示例)	40
图36.	用户应用程序二进制文件长度	41
图37.	IDE调整	41
图38.	空白Flash页面 (NUCLEO-L476RG示例)	43
图39.	UserApp固件下载概述	44
图40.	固件版本变更	45
图41.	验证菜单	46

1 概述

表 1 和 表 2 给出了相关缩略语和术语的定义，帮助您更好地理解本文档。

表1. 缩略语列表

缩略语	说明
AES	高级加密标准
DAP	调试访问端口
ECDSA	椭圆曲线数字签名算法
GCM	AES Galois/计数器模式
HAL	硬件抽象层
IDE	集成开发环境
FWALL	防火墙
MPU	存储器保护单元
OTFDEC	动态解密
PEM	隐私增强邮件
PCROP	专有代码读保护
RDP	读出器件保护
SB	安全启动
SE	安全引擎
SFU	安全固件更新
SBSFU	安全启动和安全固件更新流程
UART	通用异步收发器
WRP	写保护

表2. 术语列表

术语	说明
固件映像	可执行二进制镜像，作为用户应用程序由设备运行。
固件头文件	元数据包，描述要安装的固件映像。它包含固件信息和密码信息。
mbedTLS	TLS和SSL协议的mbed实现，以及各自的加密算法。
sfb文件	包含固件头文件和固件映像的二进制文件。

X-CUBE-SBSFU安全启动和安全固件更新扩展包可在基于Arm^{®(a)} Cortex[®]-M处理器的STM32 32位微控制器上运行。

arm

a. Arm是Arm Limited（或其子公司）在美国和其他地区的注册商标。

2 相关文档

1. 用户手册 适用于STM32H7系列的STM32CubeH7入门 (UM2204)
2. 用户手册 适用于STM32G4系列的STM32CubeG4入门 (UM2492)
3. 用户手册 适用于STM32L0系列的STM32CubeL0入门 (UM1754)
4. 用户手册 面向STM32L1系列的STM32CubeL1 MCU软件包入门 (UM1802)
5. 用户手册 适用于STM32WB系列的STM32CubeWB入门 (UM2550)
6. 用户手册 适用于STM32L4系列和STM32L4+系列的STM32CubeL4入门 (UM1860)
7. 用户手册 面向STM32F4系列的STM32CubeF4 MCU软件包入门 (UM1730)
8. 用户手册 面向STM32F7 系列的STM32CubeF7 MCU软件包入门 (UM1891)
9. 用户手册 适用于STM32G0系列的STM32CubeG0入门 (UM2303)
10. 用户手册 X-CUBE-SBSFU STM32Cube扩展包入门 (UM2262)
11. STM32Cube扩展包开发指南用户手册(UM2285)
12. STM32Cube扩展包开发清单用户手册(UM2312)
13. STM32CubeProgrammer软件描述用户手册(UM2237)
14. STM32 Cortex®-M4 MCU和MPU编程手册(PM0214)
15. STM32F7系列和STM32H7系列Cortex®-M7处理器编程手册 (PM0253)
16. STM32L0、STM32G0、STM32WL和STM32WB系列Cortex®-M0+编程手册(PM0223)
17. STSAFE-A110认证、最先进的外设安全措施以及IoT设备数据手册(DS12911)

3 移植X-CUBE-SBSFU到另一块板上

X-CUBE-SBSFU补充了STM32Cube™软件技术，使不同STM32微控制器之间更易移植。它附带一组在给定的STM32板上实现的示例，可作为有用的起点将X-CUBE-SBSFU移植到另一块STM32板。本文档中将NUCLEO-L476RG和NUCLEO-L432KC板作为示例。

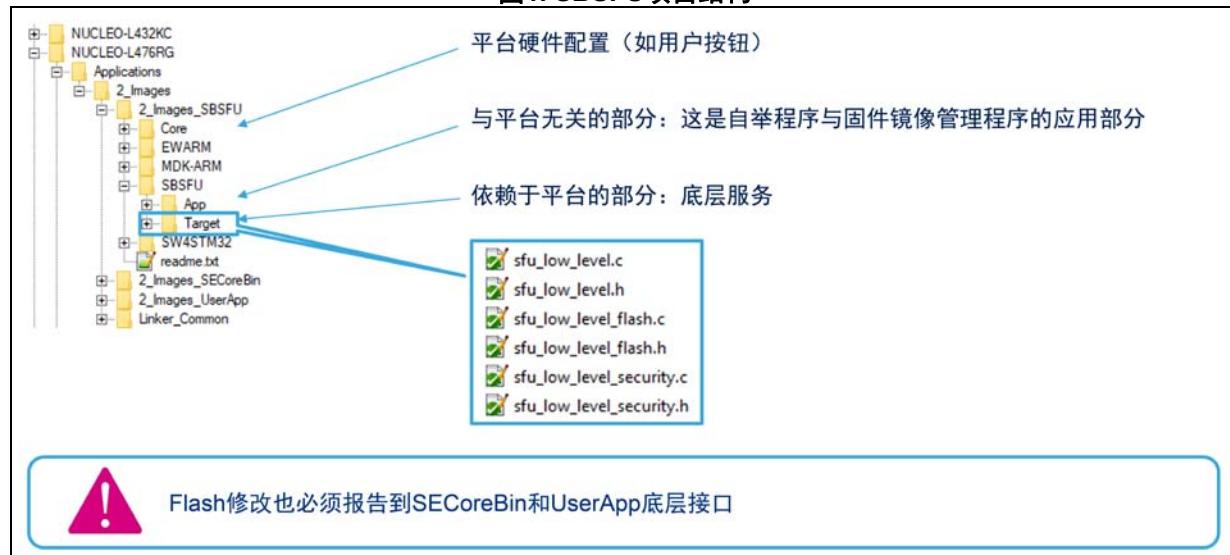
3.1 硬件适配

需要进行一些更改，以使X-CUBE-SBSFU适应另一块板：

1. GPIO配置，用于与主机PC进行UART通信（在`sfu_low_level.h`文件中）
2. Flash配置：NUCLEO-L432KC给出了单存储区Flash接口的例子，而NUCLEO-L476RG则基于双存储区（在`sfu_low_level.c`文件中）
3. 按钮配置：NUCLEO-L476RG给出了基于按钮的例子，而NUCLEO-L432KC使用GPIO模拟虚拟按钮（在`app_hw.h`文件中）
4. Tamper GPIO引脚配置（在`sfu_low_level_security.h`文件中）
5. DAP - 调试端口配置（在`sfu_low_level_security.h`文件中）
6. I²C总线配置，用于与STSAFE-A110进行通信（在`B-L4S5I-IOT01A\Applications\2_Images_STSAFE\2_Images_SECoreBin\stsafea_service_interface.c`文件中）。

图 1 显示SBSFU项目结构，以及期望移植更改的文件位置。

图1. SBSFU项目结构



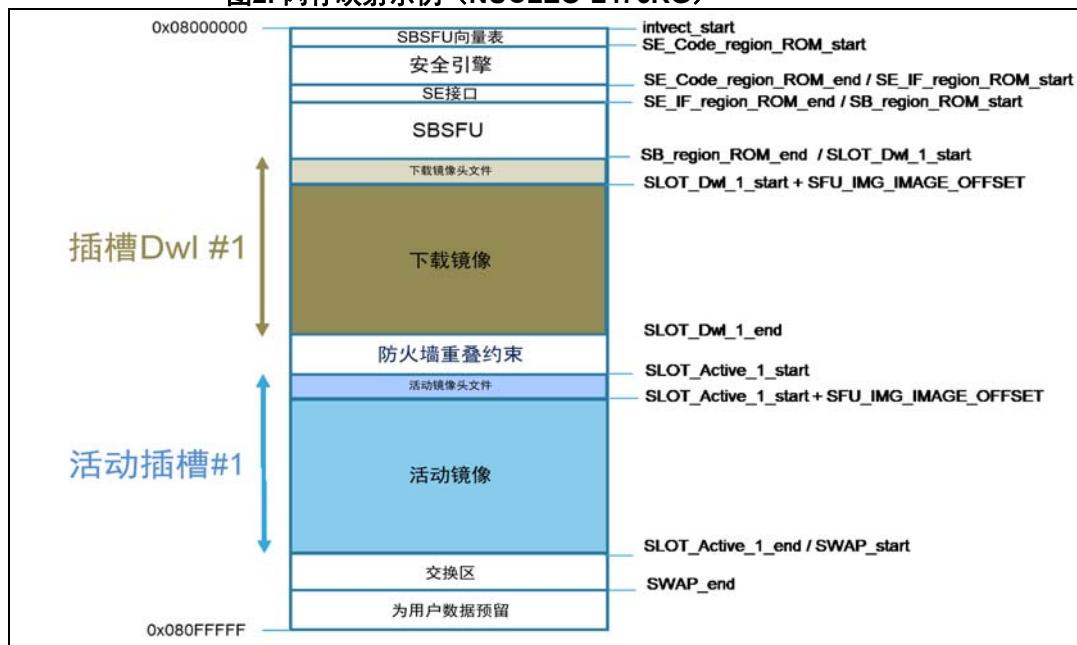
3.2 内存映射定义

正如X-CUBE-SBSFU用户手册（参见[\[10\]](#)）中所强调，一个关键因素是器件闪存内所有元素的布局：

- 安全引擎：受保护的环境，以管理所有关键数据和操作。
- SBSFU：安全启动和安全固件更新流程
- 活动插槽：该插槽包含了活动固件（带固件的固件头文件）
- 下载插槽：此插槽用于存储下载的固件（带加密固件的固件头文件），以便在下次重启时安装
- 交换区：用来在安装过程中交换活动插槽和下载插槽的内容的Flash存储区

[图 2](#)通过NUCLEO-L476RG示例讲解闪存映射。

图2. 闪存映射示例（NUCLEO-L476RG）



三个项目（SECoreBin、SBSFU、UserApp）共享的链接器文件定义在*Linker_Common*文件夹中分组，如图3中所示：

- *mapping_fwimg.icf* 包含固件镜像定义，比如活动插槽、下载插槽以及交换区
- *Mapping_sbsfu.icf*: *contains*: 包含SBSFU定义，例如SE_Code_region、SE_Key_region、以及SE_IF_region
- *Mapping_export.h*: 从 *mapping_sbsfu.icf* 和 *mapping_fwimg.icf*: 从 *mapping_sbsfu.icf* 和 *mapping_fwimg.icf* 导出符号到SBSFU应用程序

当添加更多代码后，每个区域可能要做相应的扩展，或者转移到另一个位置，这些都是允许的，只要最终的设置满足应用的安全需求。

图3. 链接器文件架构

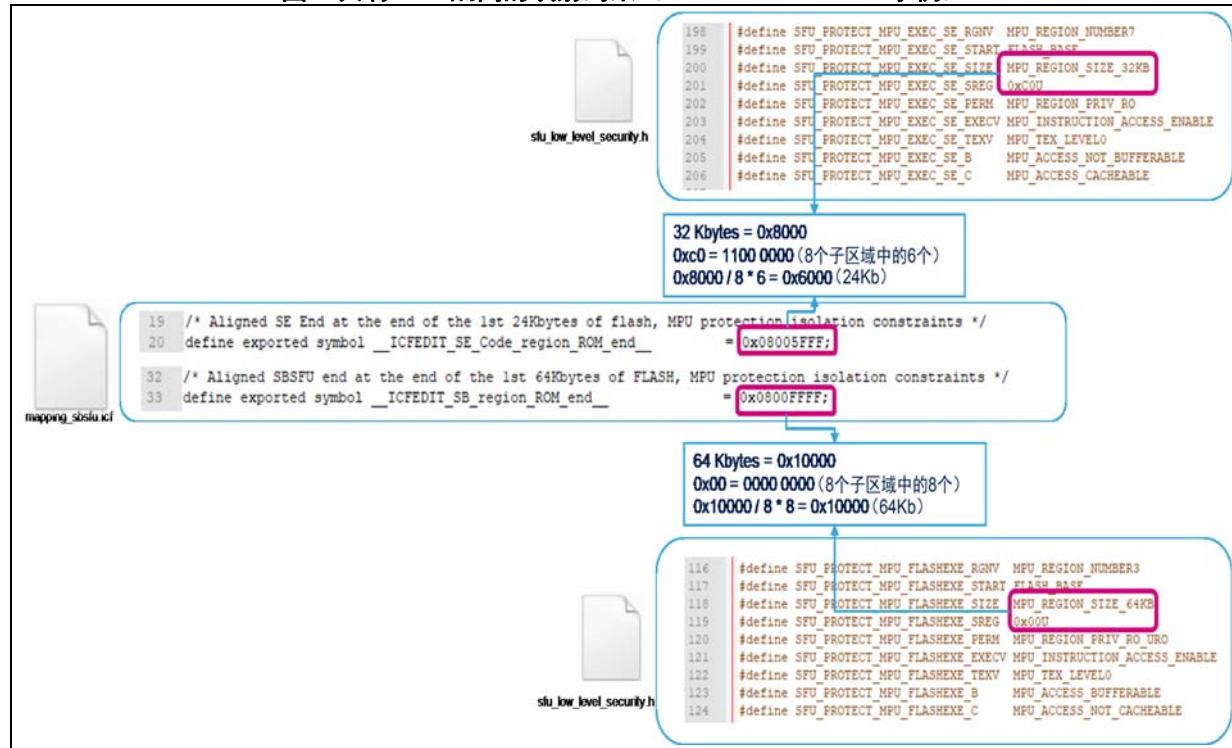


安全外设配置（RDP、WRP、PCROP、FWALL、安全用户存储区（如果当前型号拥有））是基于SBSFU链接符号自动计算的；但是MPU配置不一样，因为存在以下约束：

- 每个MPU区域的基址必须是MPU区域大小的倍数。
- 每个MPU区域可以划分为8个子区域，以调整大小。

图4中说明了具有MPU隔离的映射约束。

图4. 具有MPU隔离的映射约束 (NUCLEO-G071RB示例)



另一个典型用例是激活插槽区域的MPU配置，以授权执行用户应用程序。图5演示如何遵守NUCLEO-L073RZ上的MPU约束。

图5. 映射约束，面向用户应用程序执行



3.2.1 SBSFU区域定义参数

图 6在mapping_sbsfu.icf文件中给出这些参数，用于SBSFU区域的配置。

图6. SBSFU区域(NUCLEO-L476RG mapping_sbsfu.icf)

```

/* SE Code region protected by firewall */
define exported symbol __ICFEDIT_SE_Code_region_ROM_start__ = 0x08000200;
define exported symbol __ICFEDIT_SE_CallGate_region_ROM_start__ = __ICFEDIT_SE_Code_region_ROM_start__ + 4;
define exported symbol __ICFEDIT_SE_CallGate_Region_ROM_End__ = __ICFEDIT_SE_Code_region_ROM_start__ + 0xFF;

/* SE key region protected by firewall */
define exported symbol __ICFEDIT_SE_Key_region_ROM_start__ = __ICFEDIT_SE_CallGate_Region_ROM_End__ + 1;
define exported symbol __ICFEDIT_SE_Key_region_ROM_end__ = __ICFEDIT_SE_Key_region_ROM_start__ + 0xFF;

/* SE Startup: call before enabling firewall */
define exported symbol __ICFEDIT_SE_Startup_region_ROM_start__ = __ICFEDIT_SE_Key_region_ROM_end__ + 1;
define exported symbol __ICFEDIT_SE_Code_nokey_region_ROM_start__ = __ICFEDIT_SE_Startup_region_ROM_start__ + 0x100;
define exported symbol __ICFEDIT_SE_Code_region_ROM_end__ = __ICFEDIT_SE_Startup_region_ROM_start__ + 0x4BFF;

```

→ 更新大小时偏移允许自动调节：设置保护的SBSFU代码将其考虑在内。
在产品验证期间，用户负责验证保护。

→ 在约束情况下使用的绝对值（针对STM32F4、STM32F7、STM32G0、STM32G4、STM32L1和STM32H7的MPU配置）。

→ 区域起始地址必须256字节对齐（SE_CallGate除外）。

```

/* SE IF ROM: used to locate Secure Engine interface code out of firewall */
define exported symbol __ICFEDIT_SE_IF_region_ROM_start__ = __ICFEDIT_SE_Code_region_ROM_end__ + 1;
define exported symbol __ICFEDIT_SE_IF_region_ROM_end__ = __ICFEDIT_SE_IF_region_ROM_start__ + 0x4FF;

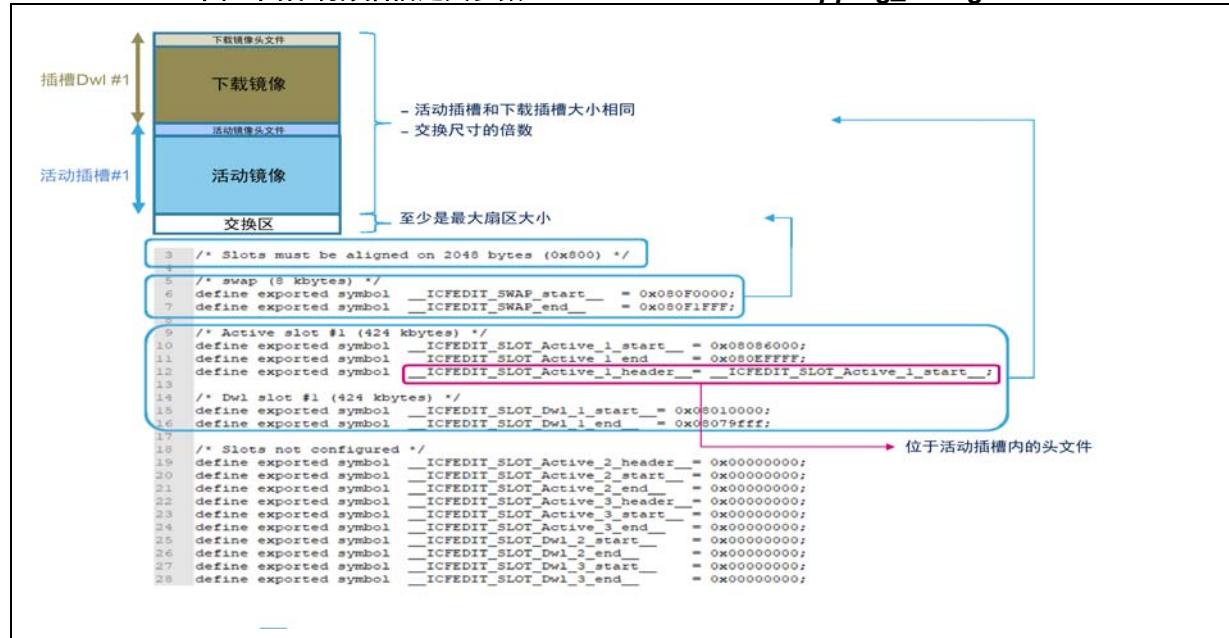
/* SBSFU Code region */
define exported symbol __ICFEDIT_SE_region_ROM_start__ = __ICFEDIT_SE_IF_region_ROM_end__ + 1;
define exported symbol __ICFEDIT_SE_region_ROM_end__ = 0x0800FFFF;

```

3.2.2 固件镜像插槽定义参数

图 7 在 *mapping_sbsfu.icf* 文件中给出这些参数，用于镜像区域的配置。

图7. 固件镜像插槽定义参数 (NUCLEO-L476RG *mapping_fwimg.icf*)



遵守SBSFU约束要求满足以下条件：

- 插槽区域必须与闪存扇区大小对齐，对于STM32L4系列，闪存扇区大小为2048字节 (0x800)。
- 交换区的最小尺寸是4 Kb，至少与最大扇区的尺寸相当。
- 活动插槽和下载插槽的尺寸必须是交换区尺寸的倍数。
- 活动插槽和下载插槽的尺寸必须相等，除非使用局部更新功能。

在某些配置中（带OTFDEC的外部Flash，多镜像配置），头文件必须位于活动插槽外部的自身闪存扇区中，以在隔离的环境中受到保护。

对于STM32L4双存储区闪存器件，特定于防火墙的约束有：

- 防火墙代码段必须在bank1中，防火墙非易失性数据（包括激活插槽的头文件）段必须在bank2中。
- 非易失性数据段必须与防火墙代码段重叠，以确保即使交换了存储区，机密也始终受到保护。

[图 8：双存储区产品的防火墙配置约束](#)和[图 9：交换存储区之后的防火墙配置](#)举例说明 NUCLEO-L476RG 上的防火墙配置，以及交换存储区的后果。

图8. 双存储区产品的防火墙配置约束

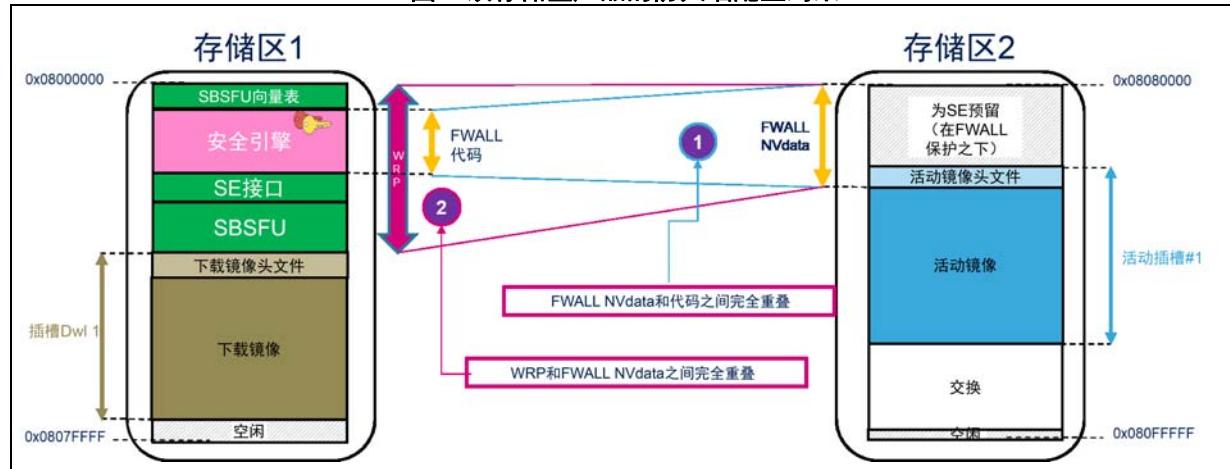
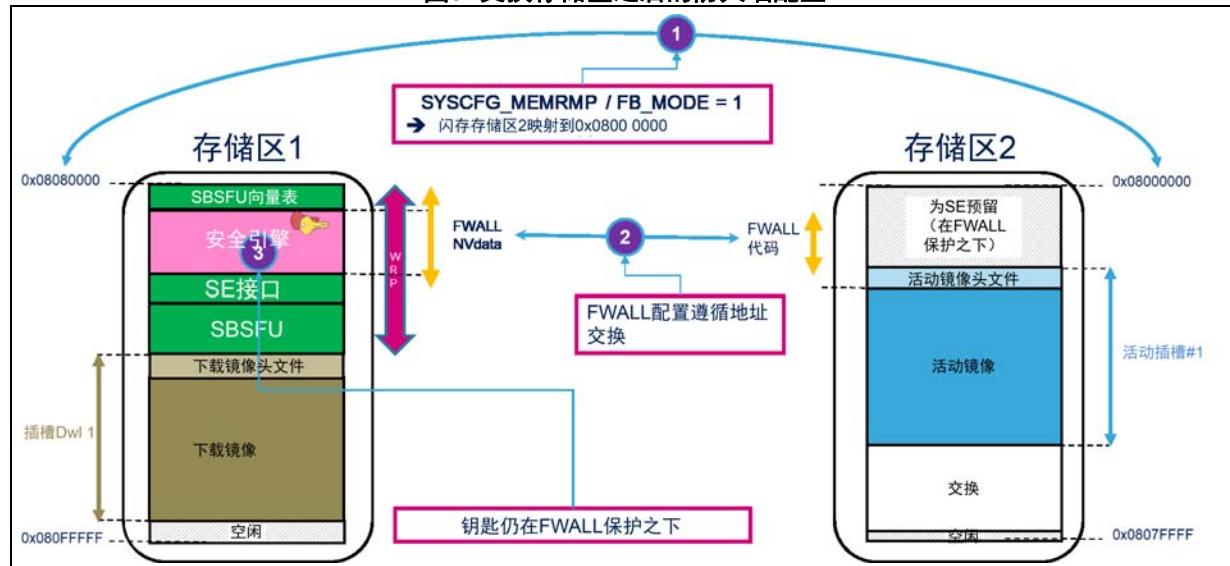


图9. 交换存储区之后的防火墙配置



对于STM32G0系列、STM32G4系列，以及STM32H7系列，存在一种约束：激活插槽的头文件必须恰好映射在SBSFU代码之后，这样才能受到安全内存的保护。

SFU_IMAGE_OFFSET值取决于STM32微控制器系列：

- 对于STM32L4系列、STM32L0系列、STM32L1系列、STM32WB系列、以及STM32F4系列，使用默认值：512字节。
- 对于STM32F7系列和STM32H7系列：1024字节。
(因为采用Cortex®-M7，向量表必须与1024字节对应)。
- 对于STM32G0系列：2048字节。
安全用户内存结束地址与闪存扇区大小一致。
- 对于STM32G4系列：4096字节。
安全用户内存结束地址与闪存扇区大小一致。
- 对于STSAFE-A变体：2048字节。
为了包含X509证书，镜像头部的长度为2048字节。

注：如果系列器件带有基于MPU的隔离或基于防火墙的隔离，激活插槽配置的MPU约束必须按照图5所示加以验证。

3.2.3 特定项目的链接文件

SECoreBin放置关键代码和数据（如机密），如图10中所示。

图10. 特定于SECoreBin的链接文件

```

14 do not initialize { section .noinit, section BOOTINFO_DATA};
15 define block SE_VECTOR with alignment = 512 {readonly section .intvec};
16
17 /*****
18 /* placement instructions */
19 *****/
20 place at address mem: _ICFEDIT_SE_CallGate_region_ROM_start_ { readonly section .SE_CallGate_Code };
21 place at address mem: _ICFEDIT_SE_Key_region_ROM_start_ { readonly section .SE_Key_Data };
22 place at address mem: _ICFEDIT_SE_Startup_region_ROM_start_ { readonly section .SE_Startup_Code };
23 place in SE_ROM_region {readonly, block SE_VECTOR};
24 place in SE_RAM_region {readwrite, section BOOTINFO_DATA};

```



SBSFU链接文件负责SBSFU应用程序内存布局，包括SECoreBin二进制文件，如图11中所示。

图11. 特定于SBSFU的链接文件

```

/*****
/* placement instructions */
****/
place at address mem: _ICFEDIT_intvec_start_ { readonly section .intvec };
place at address mem: _ICFEDIT_SE_CallGate_region_ROM_start_ { readonly section SE_CORE_Bin }; • SECoreBin项目生成
place in SE_IF_ROM_region {section .SE_IF_Code};
place in SB_ROM_region {readonly};
place in SB_SRAM1_region {readwrite, block CSTACK, block HEAP };

```

UserApp必须配置为在活动插槽（活动插槽起始地址以及SFU_IMG_IMAGE_OFFSET）中运行，如图 12 中所示，其中，SFU_IMG_IMAGE_OFFSET是面向STM32L4系列的512字节。

图12. 特定于UserApp的链接文件（NUCLEO-L476RG示例）

```

13 /*-Specials-*/
14 define exported symbol __ICFEDIT_intvec_start__ = __ICFEDIT_SLOT_Active_1_start__ + 512;
15
16 /*-Memory Regions-*/
17 define symbol __ICFEDIT_region_ROM_start__ = __ICFEDIT_intvec_start__;
18 define symbol __ICFEDIT_region_ROM_end__ = __ICFEDIT_SLOT_Active_1_end__;
19
20 define symbol __ICFEDIT_region_RAM_start__ = __ICFEDIT_SE_region_RAM_end__ + 1;
21 define symbol __ICFEDIT_region_RAM_end__ = 0x20017FFF;
22
23 /* to make sure the binary size is a multiple of the AES block size (16 bytes) and L4 flash
24   writing unit (8 bytes) */
25 define root section aes_block_padding with alignment=16
26 {
27   udata8 "Force Alignment";
28   pad_to 16;
29 };
30
31 place in ROM_region { readonly, last section aes_block_padding };
32

```

- UserApp必须配置为从活动插槽起始地址 + SFU_IMG_OFFSET (STM32L4系列为512) 运行
- SE使用的受保护RAM (FWALL, MPU⁽¹⁾) 不能重复使用
- 固件的大小应该是AES块大小和FLASH写入的数据单元的倍数

1. 取决于STM32微控制器系列。

3.2.4 多镜像配置

最多可配置三个活动插槽(SFU_NB_MAX_ACTIVE_IMAGE)和三个下载插槽(SFU_NB_MAX_DLW_AREA)。

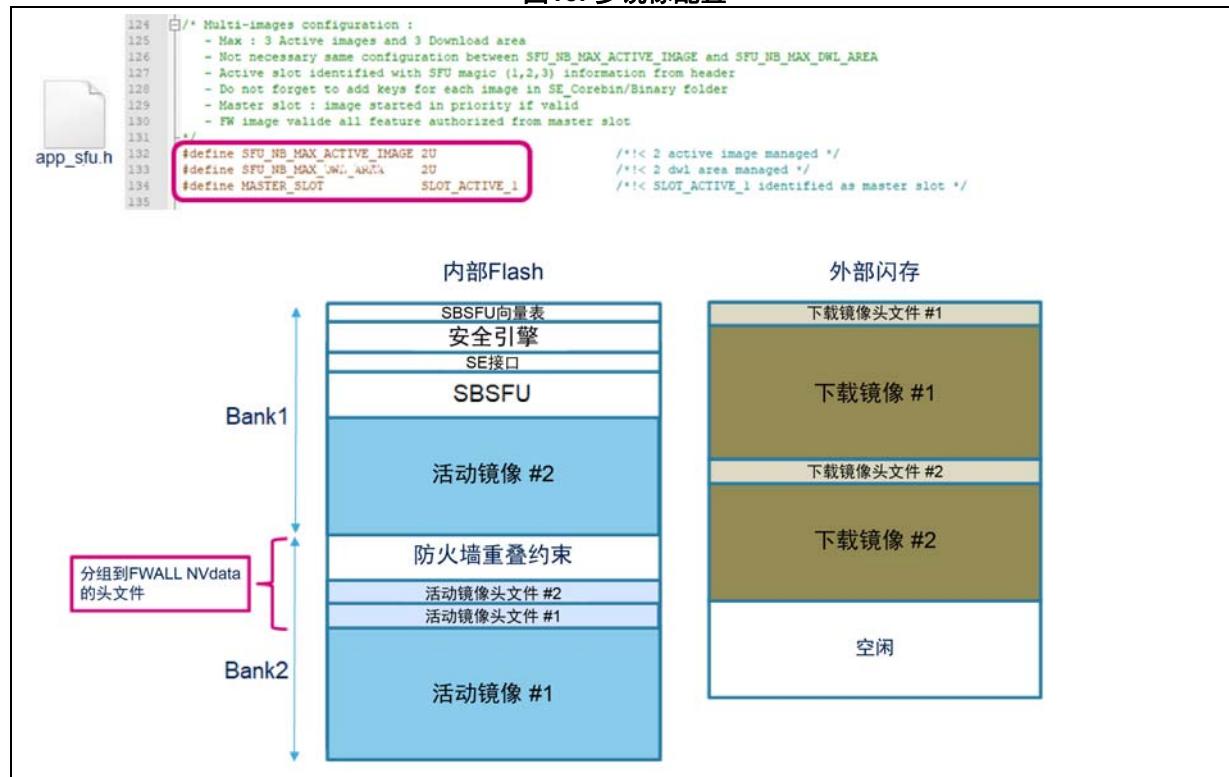
在安装过程中，通过固件镜像头文件中的SFU魔术标签 (SFU1、SFU2或SFU3) 来识别活动插槽。根据固件兼容性约束，如不需要同时安装固件，则可为三个活动插槽配置单个下载插槽，以优化内存占用。

启动时，在验证所有固件镜像的真实性和完整性之后，SBSFU按照优先级跳入位于MASTER_SLOT内部的活动固件镜像。

作为约束，所有头文件均必须分组到单个区域中，以便在隔离环境中受到保护。每个头文件均必须位于其自己的闪存扇区中。

图 13 显示了在B-L475E-IOT01A板的2_Images_ExtFlash中提供的多镜像配置示例。

图13. 多镜像配置



3.3 双核适配

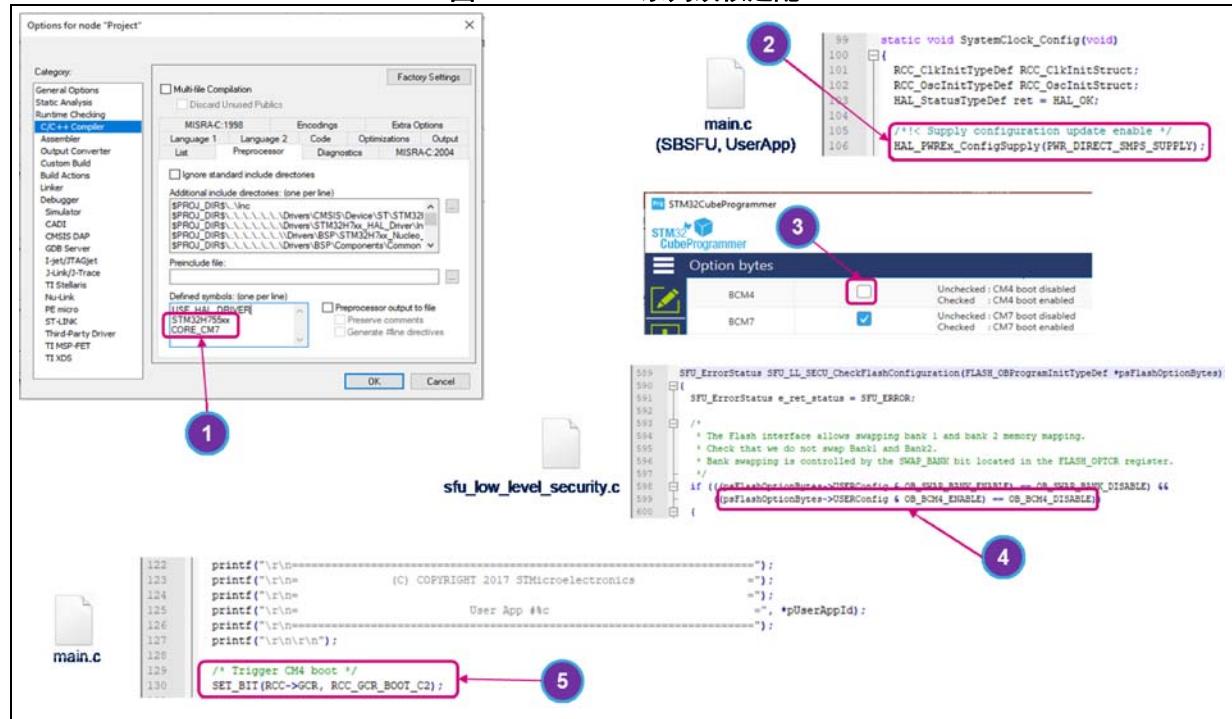
对于STM32H7系列双核产品，当SBSFU运行于CM7上时，必须禁用CM4启动。

因此，在通过SBSFU验证所有固件镜像的真实性与完整性之后，在CM7上启动的用户应用程序可以触发CM4启动。

例如，要将为NUCLEO-H753ZI提供的应用程序移植到NUCLEO-H755ZI-Q上，则需要以下修改，如图 14中所示：

1. 通过添加STM32H755xx和CORE_CM7定义符号来修改IDE配置。
2. 在SystemClock_Config()函数中将供电配置从LDO更改为SMPMS。
3. 禁用Cortex M4启动：必须在选项字节中取消对BCM4位的选中。
4. 新增了SFU_LL_SECU_CheckFlashConfiguration()函数的BCM4位状态控制。
5. 新增了UserApplication项目，CM4启动触发器。

图14. STM32H7系列双核适配



可调整插槽配置，以管理两个固件镜像，一个专用于CM7，另一个专用于CM4。更多详细信息，请参见3.2.4 多镜像配置。

4 SBSFU配置

4.1 要配置的特性

X-CUBE-SBSFU支持：

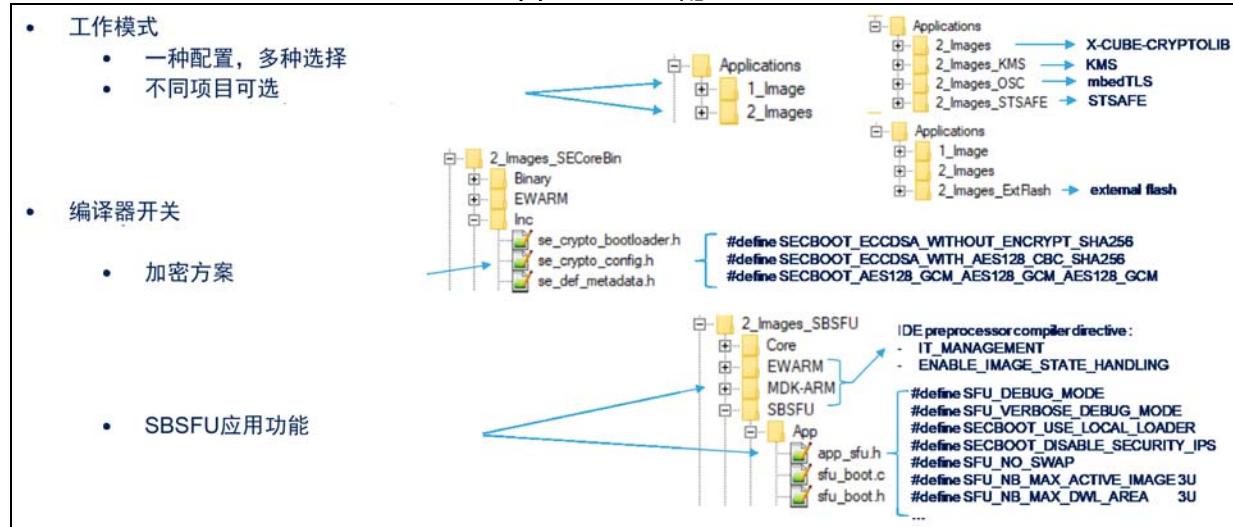
- 2种操作模式：双插槽和单插槽配置
- 3种加密方案，使用对称和非对称加密操作
- 两种加密中间件：
 - ST的中间件：X-CUBE-CRYPTOLIB库，集成在1_Image和2_Images变体中。
 - 第三方中间件：mbedtls（开源代码）加密服务。为2_Images_OSC变体中的32L496GDISCOVERY、B-L475E-IOT01A、32F413HDISCOVERY、32F769IDISCOVERY、P-NUCLEO-WB55和NUCLEO-H753ZI Nucleo板提供了示例。
- STSAFE-A110安全元件用于托管X509证书和密钥。为2_Images_STSAFE变体中的B-L4S5I-IOT01A板提供了示例。
- KMS中间件。为2_Images_KMS变体中的B-L475E-IOT01A和B-L4S5I-IOT01A板提供了示例。
- 外部闪存，带OTFDEC。为2_Images_ExtFlash变体中的STM32H7B3I-DK板提供一个示例，该示例使用了一种采用AES-CTR固件加密的特定方案。
- 外部闪存，不带OTFDEC。为2_Images_ExtFlash变体中的STM32H750B-DK板提供了示例。活动插槽以及下载插槽映射到外部闪存，因此无法确保固件机密性。
- 外部闪存，不带OTFDEC。为2_Images_ExtFlash变体中的B-L475E-IOT01A板提供了示例。选择无交换的特定安装程序SFU_NO_SWAP，使下载插槽始终保持加密，以此来确保机密性。
- 外部闪存，不带OTFDEC。为2_Images_ExtFlash变体中的STM32WB5MM-DK板提供了示例。下载插槽以及备份插槽映射到外部闪存中。选择无交换的特定安装程序SFU_NO_SWAP，使这两种插槽始终保持加密，以此来确保机密性。更多详细信息，可参见X-CUBE-SBSFU STM32Cube扩展包入门用户手册（UM2262）的附录H。

以下配置选项可通过编程开关来设置：

- 可以移除本地加载器，以减少内存占用（仅限双插槽）。
- 可以激活Verbose开关，使调试更容易。
- 可以禁用调试模式（在SBSFU执行期间，终端上不再有printf）以减少内存占用。
- 可以关闭安全IP，使调试更容易。
- 带固件镜像验证的安装过程。如果固件镜像未通过用户应用程序验证，则将触发向上一固件镜像的回滚。
- 面向具有多个固件的复杂系统（如协议栈、中间件和用户应用程序）的多镜像配置。
- 隔离防火墙环境内部的中断管理，适用于需要低延迟中断处理的应用。

图 15 呈现的SBSFU配置解决方案带相关文件和编译开关。

图15. SBSFU配置



4.2 加密方案选择

X-CUBE-SBSFU提供了同时使用非对称和对称加密的三种加密方案。

- 用于固件验证的ECDSA非对称加密和用于固件解密的AES-CBC对称加密
- 用于固件验证（明文）的ECDSA非对称加密技术
- 用于固件验证和解密的AES-GCM对称加密技术

如图 16中所描述，使用SECBOOT_CRYPTO_SCHEME编译开关在这些方案中进行选择。

图16. 切换加密方案



注：对于B-L4S5I-IOT01A STSAFE和KMS变体，选择了SECBOOT_X509_ECDSA_WITHOUT_ENCRYPT_SHA256加密方案。

对于带动态解密(OTFDEC)的外部闪存，选择SECBOOT_ECCDSA_WITH_AES128_CTR_SHA256加密方案。

4.3 安全配置

SBSFU示例采用了STM32安全保护配置，允许保护机密免受外部和内部攻击。

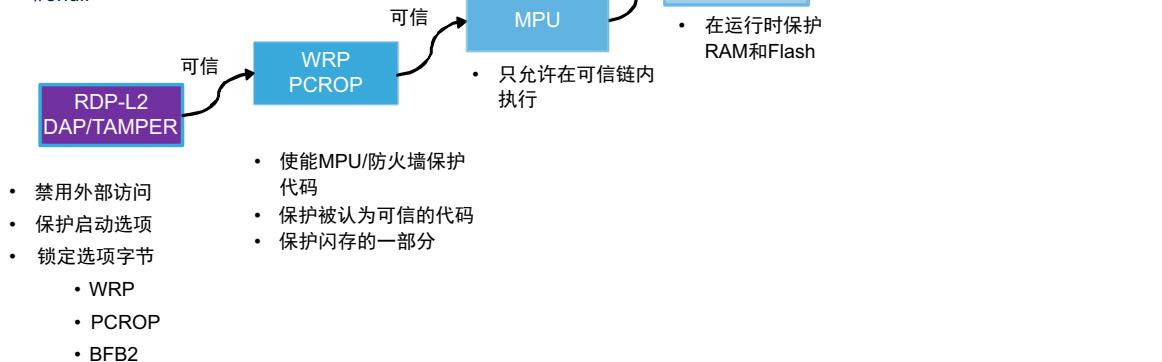
STM32的各种安全外设可由用户决定是否单独停用，以实现不同的保护级别（例如，对于STM32L4系列器件，防火墙和PCROP允许激活防止内部攻击的保护措施）。任何STM32安全配置修改都需要在系统产品级别进行安全保护评估，以确保根据产品约束和规范适当地设置保护。

在开发阶段，为了简化调试，可能需要禁用所有IP。

图 17 显示文件app_sfu.h中针对STM32L4系列和STM32L0系列提供的各种安全配置解决方案。

图17. STM32L4系列和STM32L0系列 安全配置 (app_sfu.h)

```
// #define SECBOOT_DISABLE_SECURITY_IPS /*!< Disable all security IPs at once when activated */
#ifndef SECBOOT_DISABLE_SECURITY_IPS
#define SFU_WRP_PROTECT_ENABLE
#define SFU_RDP_PROTECT_ENABLE
#define SFU_PCROP_PROTECT_ENABLE
#define SFU_FWALL_PROTECT_ENABLE
#define SFU_TAMPER_PROTECT_ENABLE
#define SFU_DAP_PROTECT_ENABLE
#define SFU_DMA_PROTECT_ENABLE
#define SFU_IWDG_PROTECT_ENABLE
#define SFU_MPUPROTECT_ENABLE
#define SFU_MPUSERAPP_ACTIVATION
#endif
```



MSv51250V3

图 18 显示文件 app_sfu.h 中针对 STM32F4 系列、STM32F7 系列和 STM32L1 系列提供的各种安全配置解决方案。

图18. STM32F4系列、STM32F7系列和STM32L1 系列安全配置 (app_sfu.h)

```
/*!< Disable all security IPs at once when activated */
#ifndef SECBOOT_DISABLE_SECURITY_IPS
#define SFU_WRP_PROTECT_ENABLE
#define SFU_RDP_PROTECT_ENABLE
#define SFU_TAMPER_PROTECT_ENABLE
#define SFU_DAP_PROTECT_ENABLE
#define SFU_DMA_PROTECT_ENABLE
#define SFU_IWDG_PROTECT_ENABLE
#define SFU_MPUPROTECT_ENABLE
#endif
```

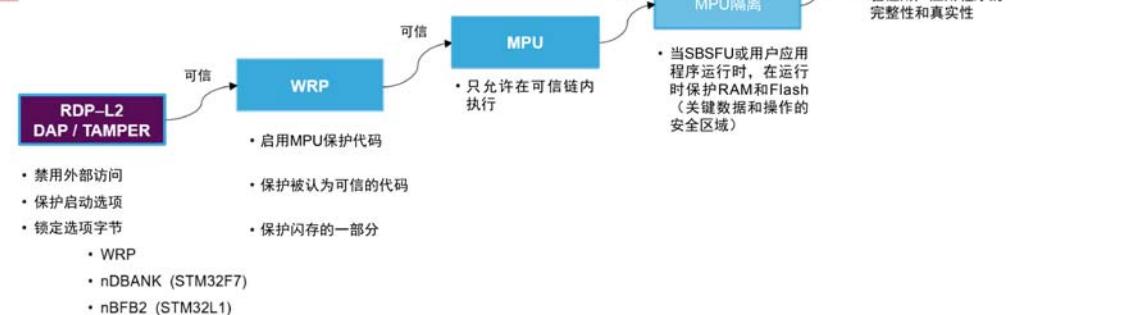


图 19 显示文件 *app_sfu.h* 中针对 STM32WB 系列提供的各种安全配置解决方案。

图19. STM32G0系列、STM32G4系列和STM32H7系列
安全配置 (*app_sfu.h*)

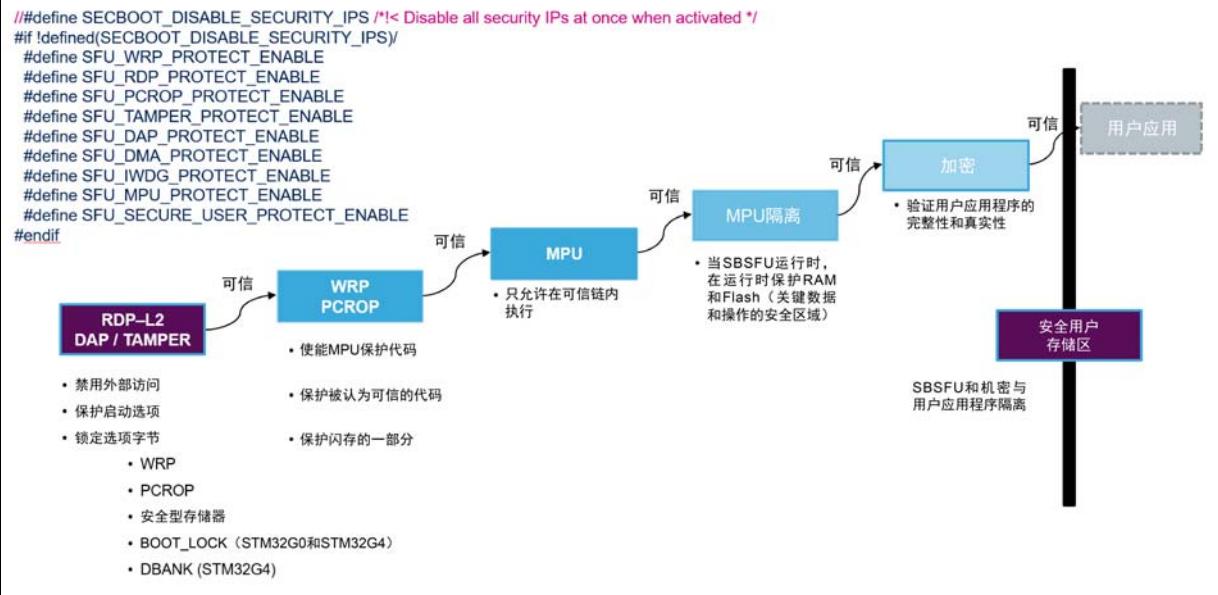
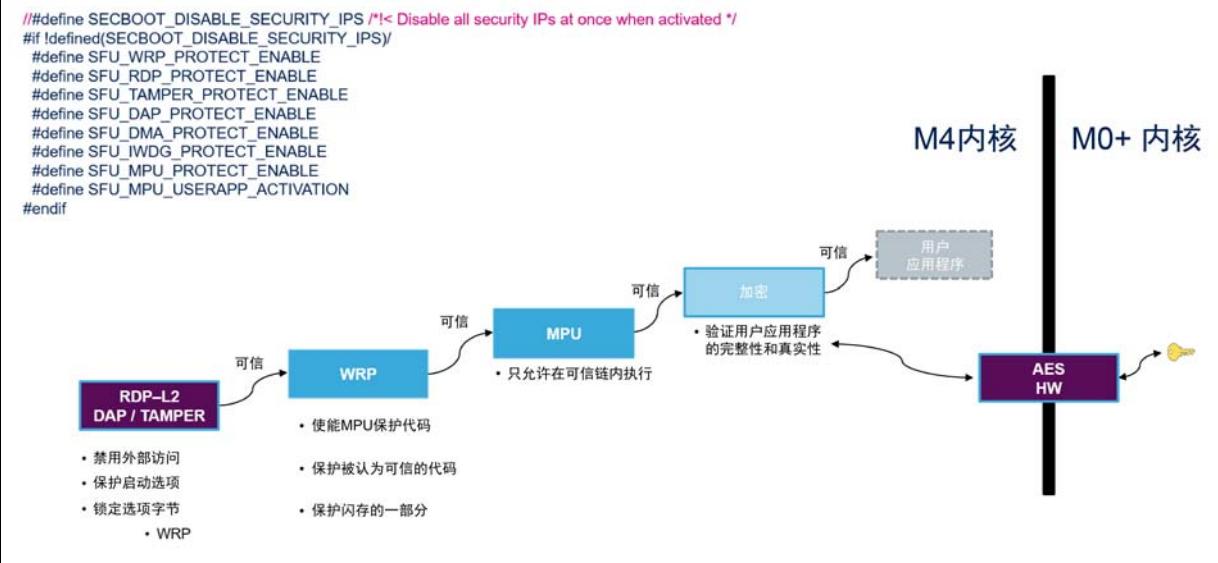


图 20 显示文件 *app_sfu.h* 中针对 STM32WB 系列提供的各种安全配置解决方案。

图20. STM32WB系列安全配置 (*app_sfu.h*)



4.4 开发或生产模式配置

进行任何代码修改之前的一步通常是在开发模式下配置SBSFU项目，以启用IDE调试设施和添加SBSFU调试跟踪：

1. 禁用所有安全保护: SFU_xxx_PROTECT_ENABLE
2. 禁用SFU_FINAL_SECURE_LOCK_ENABLE
3. 激活SFU_FWIMG_BLOCK_ON_ABNORMAL_ERRORS_MODE
4. 激活SECBOOT_OB_DEV_MODE
5. 可以选择激活verbose模式: SFU_VERBOSE_DEBUG_MODE (如需详细了解对映射的影响, 请参阅[第 6.2 节：内存映射调整](#))。

在开发阶段的最后, SBSFU项目必须配置为生产模式, 以便最终版本:

1. 激活所有需要的安全保护: SFU_xxx_PROTECT_ENABLE
2. 禁用verbose模式: SFU_VERBOSE_DEBUG_MODE
3. 禁用SFU_FWIMG_BLOCK_ON_ABNORMAL_ERRORS_MODE
4. 禁用SECBOOT_OB_DEV_MODE
5. 激活SFU_FINAL_SECURE_LOCK_ENABLE以配置RDP级别2。在STM32H7系列器件上, 如果启用了SFU_FINAL_SECURE_LOCK_ENABLE, 也会配置安全用户内存。
6. 禁用SFU_DEBUG_MODE, 以删除可能被攻击者利用的所有SBSFU印记。

读保护级别2是强制性的, 以实现最高级别的保护并实现可信根。用户负责在产品制造阶段被编程的最终软件中将其激活。

在生产模式下, 安全启动检查选项字节值 (RDP、WRP、PCROP、安全用户内存) 和块的执行, 以防检测到错误配置。少数其他选项字节必须配置 (具体取决于平台), 例如:

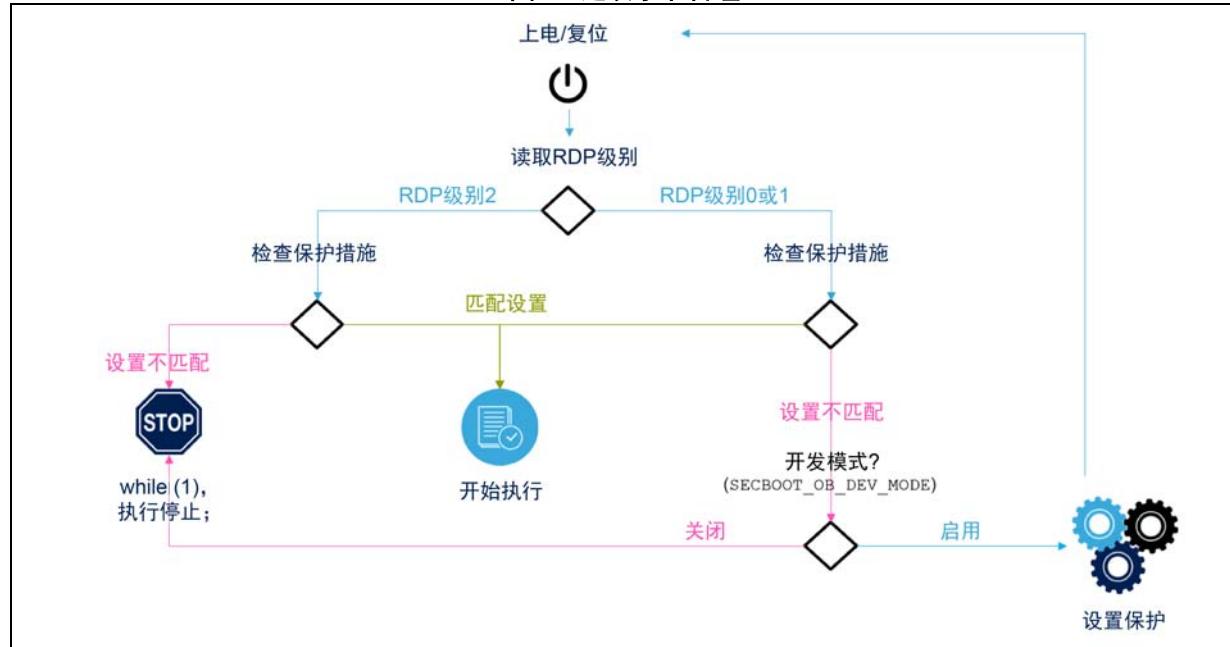
- 为采用双存储区闪存的STM32L4系列和STM32L0系列禁用BFB2
- 为STM32F7系列启用nDBANK
- 为STM32L1系列启用nBFB2
- 为STM32G0系列和STM32G4系列启用BOOT_LOCK
- 在STM32G4系列和B-L4S5I-IOT01A板上禁用DBANK

注意:

在生产阶段对软件进行编程之后, 选项字节必须通过STM32CubeProgrammer (STM32CubeProg) 配置为生产模式值。如果不这样做, 器件仍然是非安全。请参阅[\[13\]](#), 获取STM32CubeProgrammer的使用方法。

图 21 显示如何在SBSFU启动时管理选项字节：

图21. 选项字节管理



5 生成加密密钥

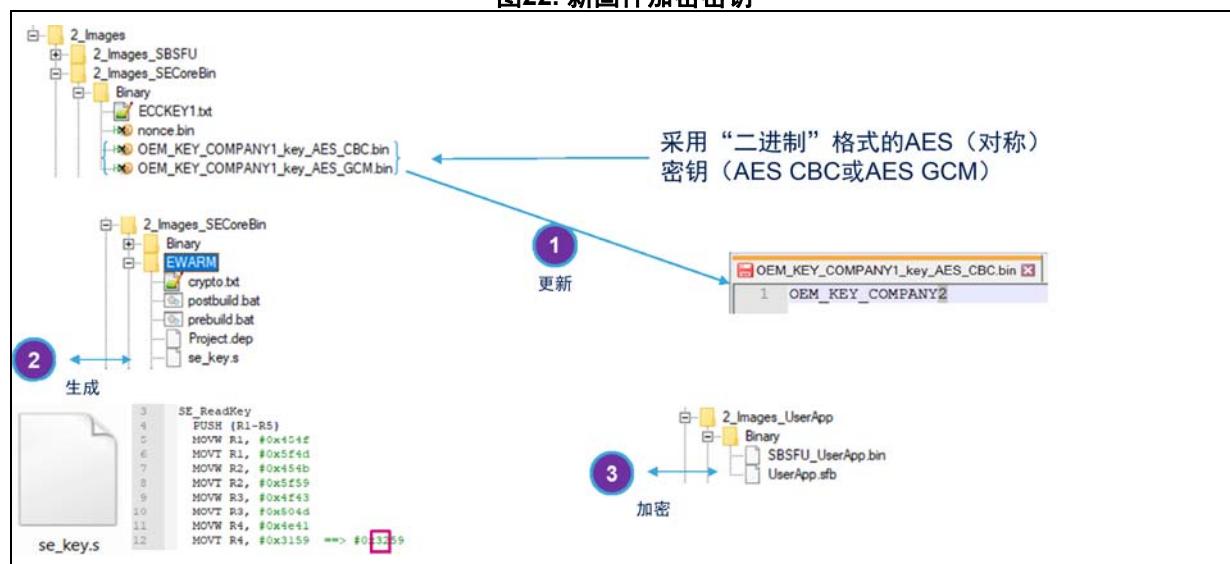
5.1 生成新的固件AES加密密钥

密钥生成和固件加密是在编译过程中通过 *prebuild.bat* 和 *postbuild.bat* 脚本自动进行的（请参阅 [10]，获取有关构建过程的详细描述）。

图 22 显示用于修改活动插槽#1的固件加密密钥的几个步骤。这同样适用于活动插槽#2或#3：

1. 在文件 *OEM_KEY_COMPANY1_keys_AES_xxx.bin* 中更改密钥值
2. 编译 *SECoreBin*：执行 *prebuild.bat*，生成 *se_key.s*
3. 编译 *UserApp*：执行 *postbuild.bat*，加密 *UserApp*

图22. 新固件加密密钥



5.2 生成用于固件验证的新的公钥/私钥对（ECDSA）密钥对

至于AES加密密钥，如果私有密钥（*ECCKEY1.txt*）发生更改，公共密钥（*SE_ReadKey_Pub()*）将自动修改。

图 23 显示修改用于激活插槽1的ECDSA非对称加密固件验证的私有密钥和公共密钥的几个步骤。这同样适用于活动插槽#2或#3：

1. 在文件 *ECCKEY1.txt* 中更改密钥值
2. 编译 *SECoreBin*：执行 *prebuild.bat*，生成 *se_key.s*
3. 编译 *UserApp*：执行 *postbuild.bat*，加密 *UserApp*

图23. 新私钥/公钥



5.3 STM32WB系列特殊性

对于STM32WB系列，AES加密密钥不是通过`prebuild.bat`脚本进行处理，而是配置在M0+内核中。`SECoreBin/readme.txt`中描述了STSAFE-A100配置过程。

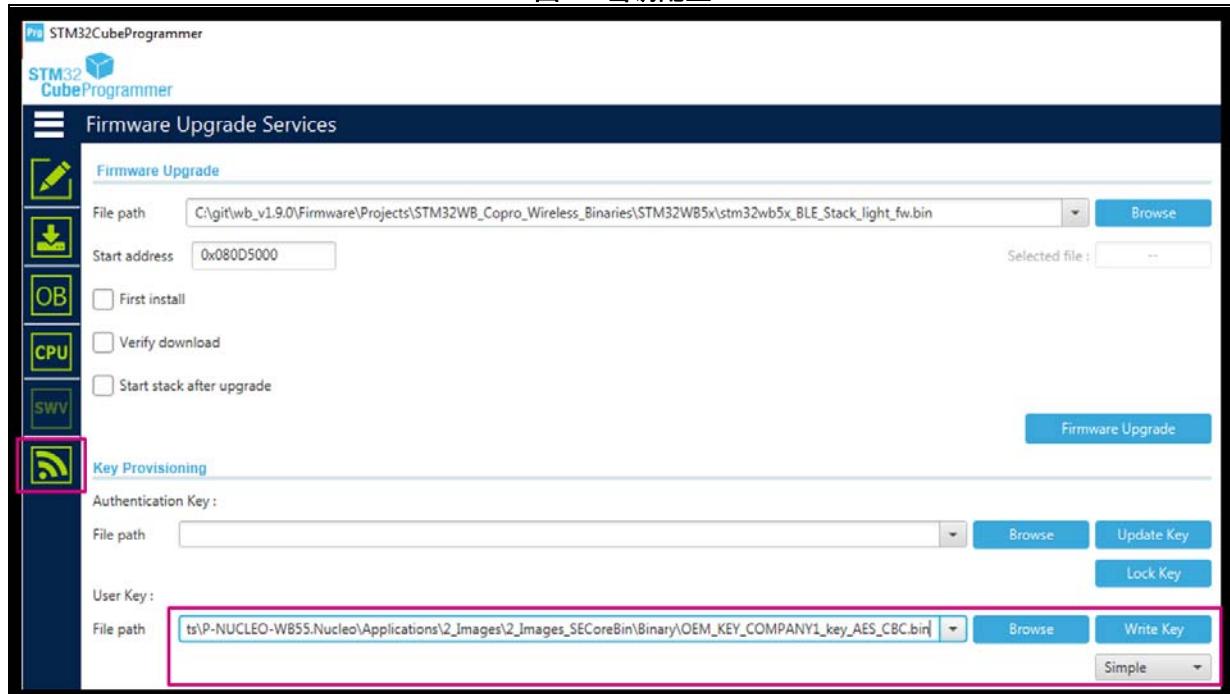
配置AES密钥的另一种方法为使用最新的`STM32CubeProgrammer`版本。自V2.5.0起，M0+密钥配置作为固件升级服务(FUS)提供。

首先，连接到自举程序USB接口：

1. 选中nBOOT1和nSWBOOT0。
2. 通过将Boot0引脚连接为VDD来选择正确的启动模式：
 - a) 对于P-NUCLEO-WB55Nucleo板：CN7.5(VDD)与CN7.7(Boot0)之间的跳线接通。
 - b) 对于STM32WB5MM-DK探索板：在焊接引脚接头和另一个跳线选择JP2上的“USB MCU”后，CN13(VDD-Boot0)上的跳线接通。
3. USB电缆连接至USB_USER接口。
4. 电源接通（拔下/插入USB电缆连接至ST-LINK）。

然后，允许固件升级服务面板的功能键配置，如图 24中所示。

图24. 密钥配置



5.4 KMS特殊性

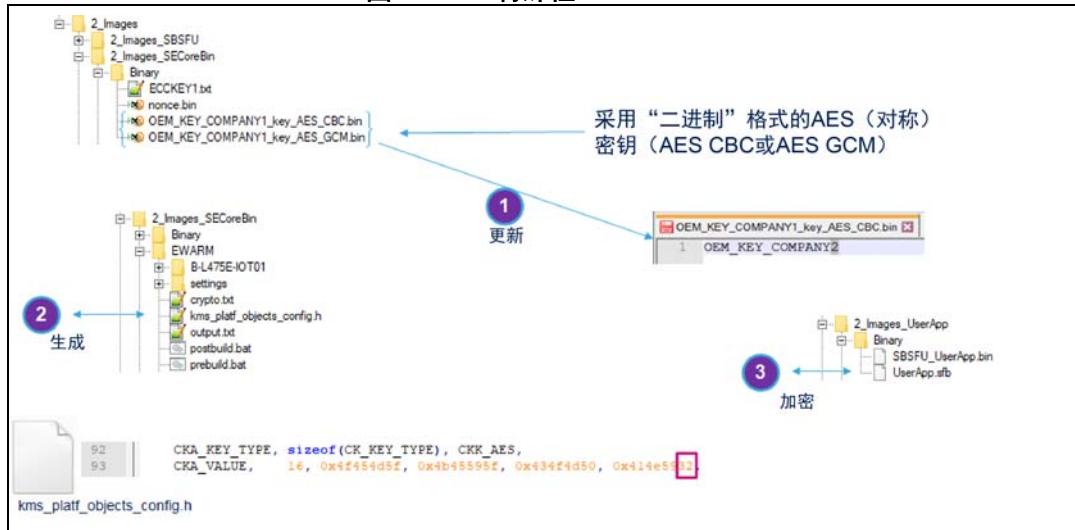
有了KMS中间件集成，SBSFU密钥不再存储在受到PCROP保护的扇区中。它们作为静态嵌入式密钥存储在KMS代码中。

[图 25](#)显示了活动插槽#1的固件加密密钥修改示例。这同样适用于活动插槽#2或#3：

1. 在文件OEM_KEY_COMPANY1_keys_AES_xxx.bin中更改密钥值
2. 编译SECoreBin: 执行prebuild.bat, 生成kms_platf_objects_config.h
3. 编译UserApp: 执行postbuild.bat, 加密UserApp

相同的过程适用于固件ECDSA验证密钥、BLOB AES加密密钥和BLOB ECDSA验证密钥。

图25. KMS特殊性



5.5 STSAFE-A110特殊性

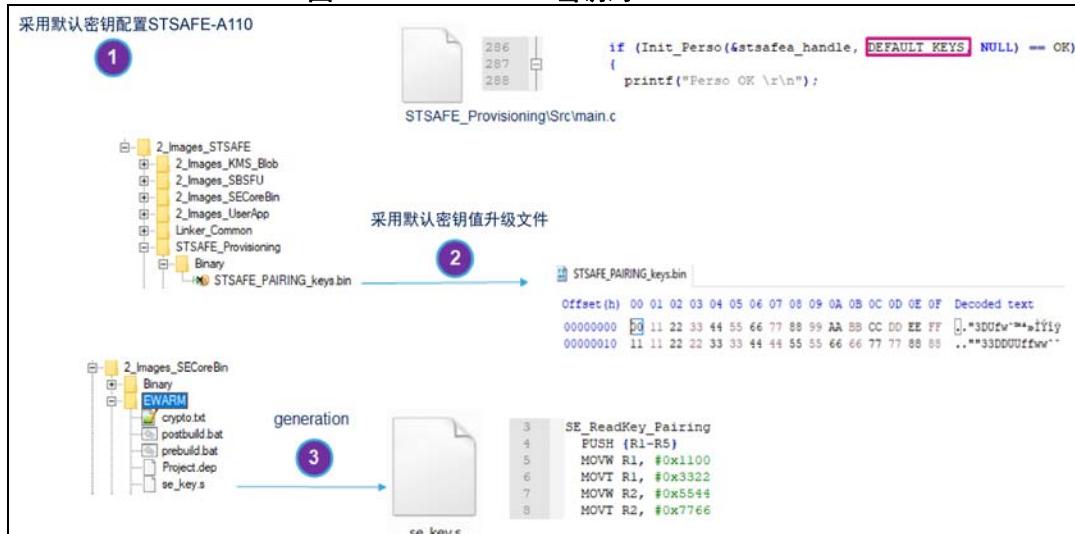
正如UM2262的附录G中所解释的那样，STM32和STSAFE-A110必须使用密钥对和X509证书进行配置。

*STSAFE_Provisioning/readme.txt*中描述了STSAFE-A110配置过程。

图26显示了一个密钥对配置示例：

1. 采用默认密钥对配置STSAFE-A110
2. 相应地更新STSAFE_PAIRING_keys.bin
3. 编译SECoreBin: 执行prebuild.bat, 生成se_key.s。

图26. STSAFE-A110密钥对

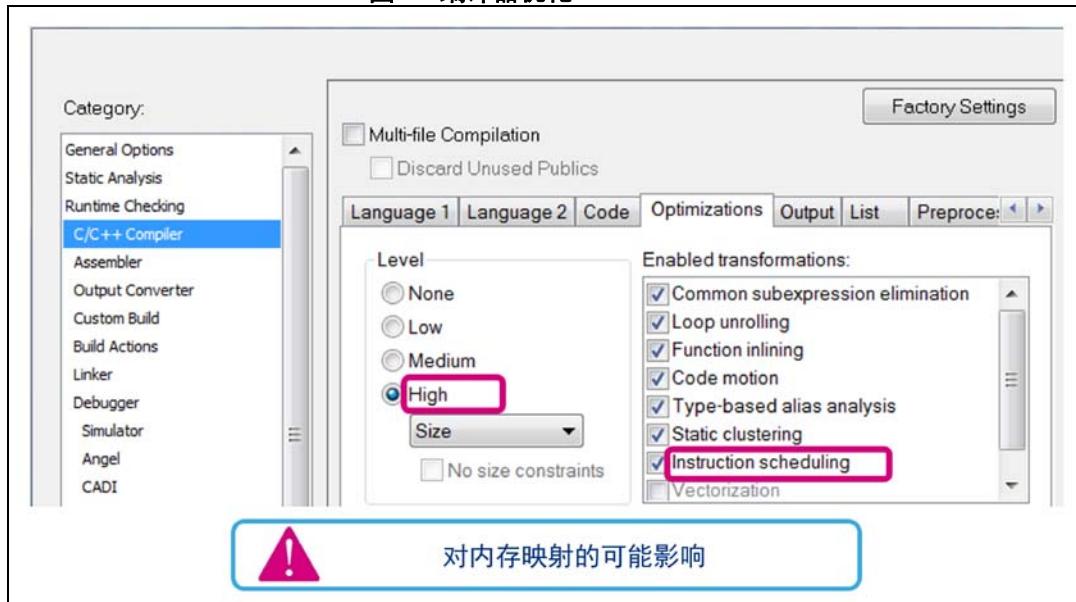


6 调试技巧

6.1 编译器优化级别

在交付项目时，针对生成文件大小方面进行了最高级别编译器优化。这样的优化会使调试变得复杂。更改编译器优化级别可能影响内存映射。

图27. 编译器优化



6.2 内存映射调整

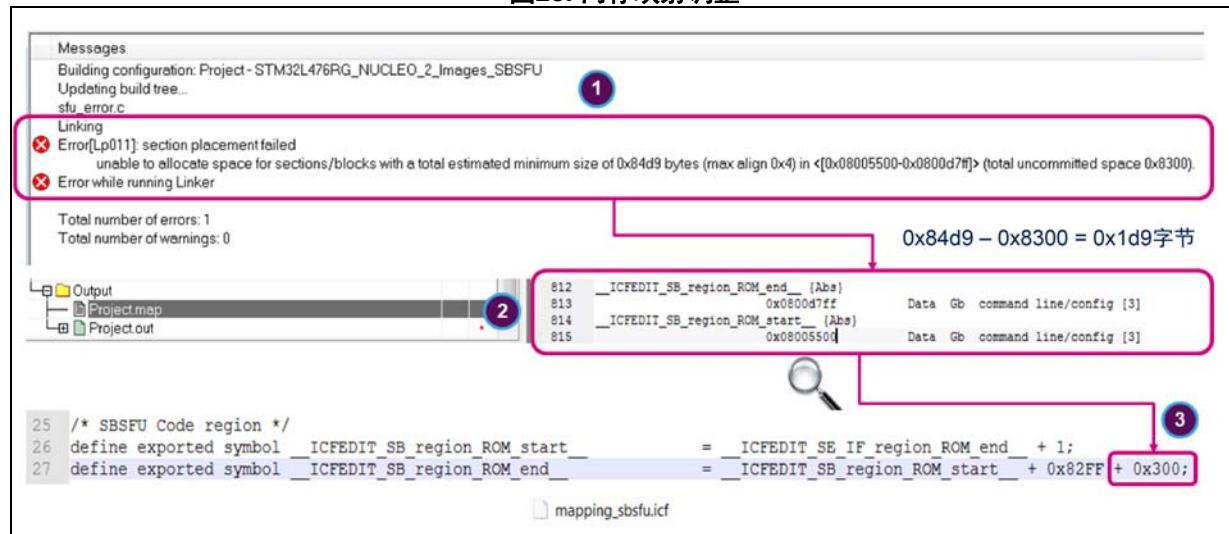
当更改编译器优化级别或使用verbose编译开关激活开发模式时，用户可能必须调整SBSFU内存映射，例如减少固件镜像插槽以避免重叠。

注意： 安全外设配置（RDP、WRP、PCROP、FWALL、安全用户内存（如果该系列支持））是基于SBSFU链接器符号自动计算的；但是MPU配置不一样，因为存在[第 3.2 节：内存映射定义](#)详细说明的约束。暂时禁用MPU保护对于调试来说是一种有效的变通方法。

[图 28](#)根据一个示例描述内存调整的3个步骤：

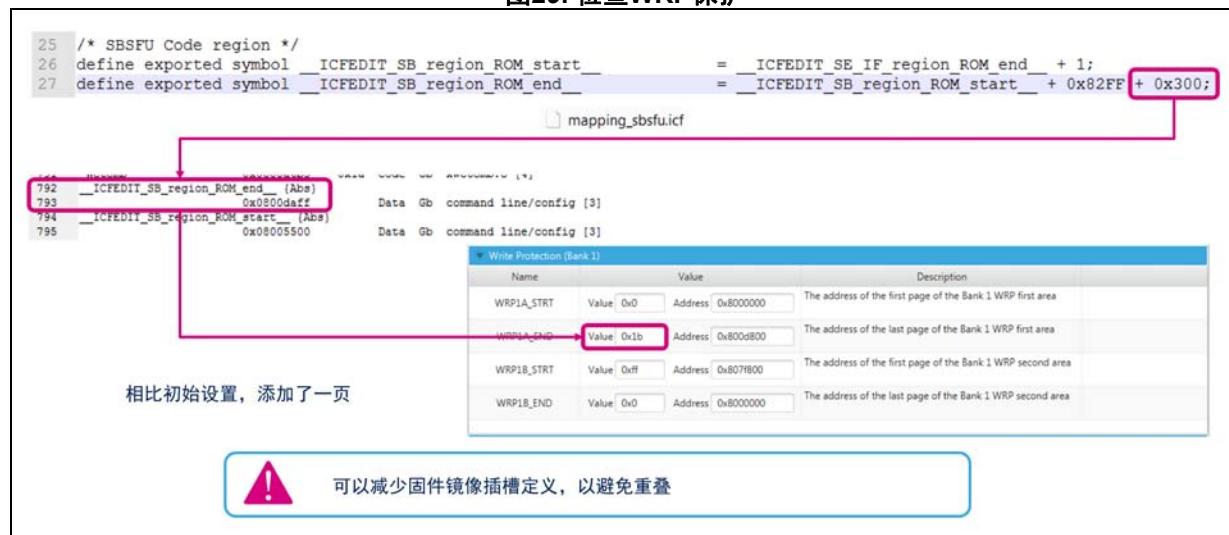
1. 通过分析链接器消息来确定GAP: 0x1d9字节
2. 通过查看*project.map*文件确定相关区域: __ICFEDIT_SB_region_ROM_start__
3. 在文件*mapping_sbsfu.icf*中实施修改: 0x300字节

图28. 内存映射调整



必须检查内存映射调整对安全外设配置的影响，尽管它是自动计算的。例如，使用 STM32CubeProgrammer (STM32CubeProg) 检查WRP配置，如图 29 中所示。

图29. 检查WRP保护

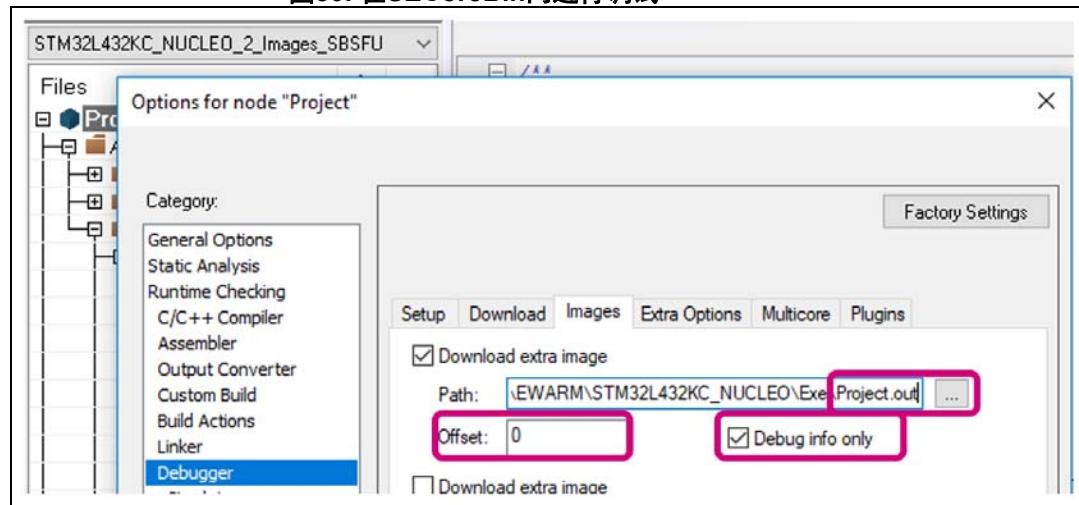


6.3 调试SECoreBin

为了在SECoreBin内部进行调试，必须更改SBSFU项目选项以加载SECoreBin符号。这是在调试器菜单中执行的，如图 30 中所示：

- 浏览并选择文件 *Project.out*
 - 设置 **偏移** 为 0
 - 勾选 **仅调试信息** 复选框

图30. 在SECoreBin内进行调试



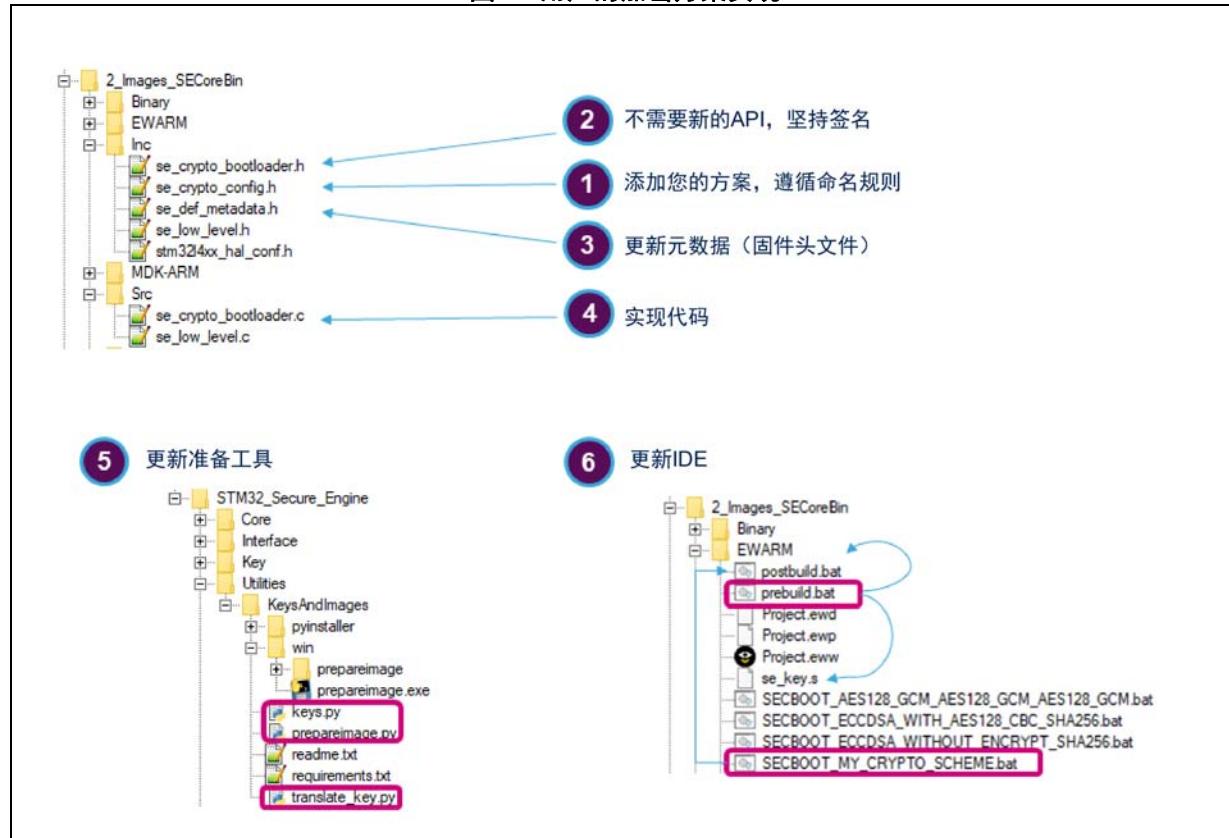
7 调整SBSFU

7.1 实现一种新的SBSFU加密方案

X-CUBE-SBSFU附带一些预定义的加密方案（请参见第 4.2 节：加密方案选择第 21 页）。还可以使用用户的加密方案来扩展软件包。

为了实现新的SBSFU加密方案，请遵循图 31 中已说明并在下面描述的步骤。

图31. 用户的加密方案实现



更新在器件端运行的代码：

1. 步骤1：为SECBOOT_CRYPTO_SCHEME定义新的值。
2. 步骤2：仔细查看引导程序所需的API签名。加密服务必须具有相同的签名，以避免更新SBSFU代码。
3. 步骤3：定义一个新的SE_FwRawHeaderTypeDef结构，并遵守约束，以保持兼容现有SBSFU代码。
4. 步骤4：在se_crypto_bootloader.c中实现加密服务代码。

更新在主机端运行的工具，以准备密钥和固件镜像：

5. **步骤5：**更新准备工具以支持新的加密方案*prepareimage.py*; *translate_key.py*; *keys.py*。
6. **步骤6：**更新IDE以生成适当的密钥和固件镜像。
 - 需要一个新的批处理文件，以使用适当的命令调用准备工具；*prebuild.bat*复制该批处理文件，用于创建*postbuild.bat*。
 - 必须更新*prebuild.bat*以考虑到新的加密方案并生成适当的密钥和*postbuild.bat*。

7.2 优化内存映射

有几个选项可以减少SBSFU代码量，以便最大化用户应用程序插槽的大小。[表 3](#)总结了其中一些选项。

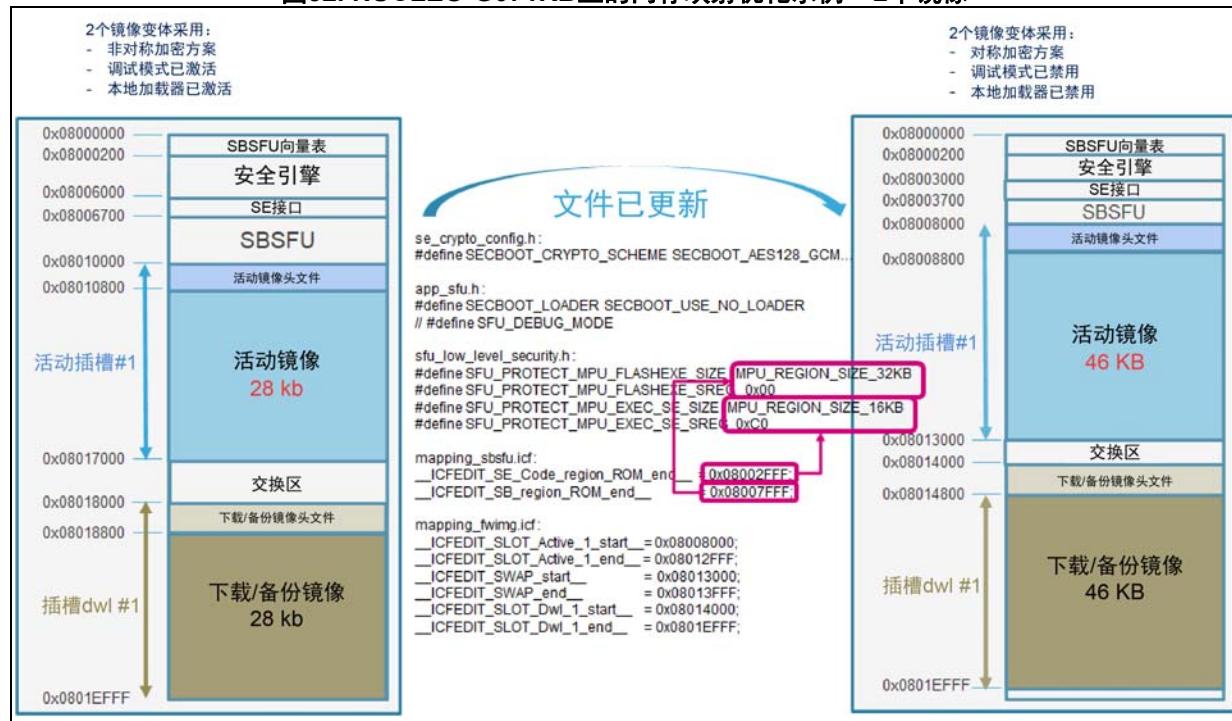
表3. 裁减SBSFU代码量

选项	描述 / 结果	内存开销
选择1-镜像变体	不再可能从用户应用程序下载新的固件镜像。	插槽的大小翻倍 vs. 2-镜像项目
选择AES-GCM对称加密方案	共享对称密钥机密存储在器件中。	~ 9 KB
禁用SFU_DEBUG_MODE	在SBSFU执行过程中，终端不再显示任何信息	~ 9 KB
禁用SECBOOT_USE_LOCAL_LOADER	SBSFU应用程序内不再有本地加载程序。它不兼容1-镜像变体。	~3 KB
实现硬件解密	选择集成加密硬件IP的STM32器件。	具体取决于用户实现
如果在STM32上运行的所有代码都是完全可信且稳定的，那么可以为STM32F4/F7/G0/G4/H7/L1移除基于MPU的安全引擎内部隔离。	移除MPU区域的对齐约束。	最高可达12 KB，具体取决于产品

总内存开销取决于[第 3.2 节：内存映射定义第 10 页](#)中描述的映射约束。

作为示例，[图 32](#)突出强调了要进行的映射修改。从使用对称加密方案的两个镜像开始，SFU_DEBUG_MODE_USE_LOCAL_LOADER开关被禁用，导致用户应用程序的大小增加了16 KB。

图32. NUCLEO-G071RB上的内存映射优化示例 – 2个镜像



在文件夹NUCLEO-G031K8\Applications\1_Image中，为NUCLEO-G031K8提供了另一个内存优化示例，在该板上可用的64 KB中，32 KB被分配给用户应用程序。

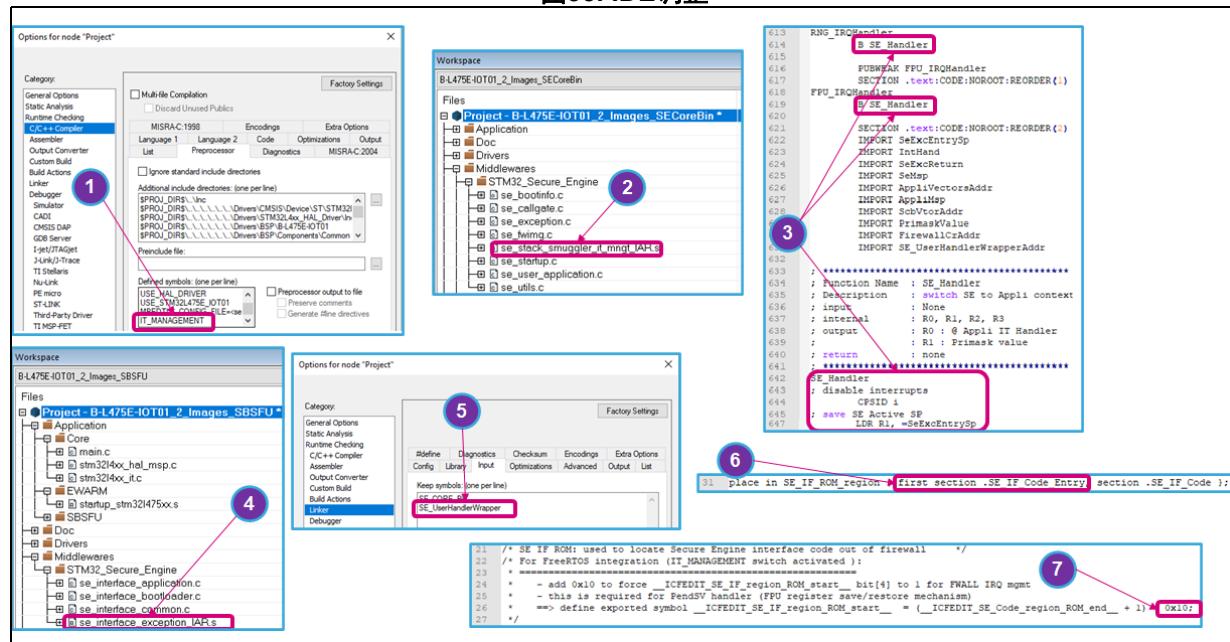
7.3 如何激活隔离防火墙内部的中断管理环境

当需要低延迟的中断处理时，可以激活防火墙隔离环境内部的中断管理。在32L496GDISCOVERY和B-L475E-IOT01A板的2_Images_OSC变体中提供了示例。

图 33 显示了激活此选项所需的不同步骤：

1. 在SECOREBin和SBSFU IDE配置中添加作为预处理指令的IT_MANAGEMENT
2. 在SECOREBin IDE配置中选择se_stack_smuggler_it_mngt_IAR.c, 而非se_stack_smuggler_IAR.c
3. 修改startup_xxx.s文件，以划分SE_Handler上所需的中断处理程序
4. 将se_interface_exception_IAR.s添加到SBSFU IDE配置中
5. 修改SBSFU链接选项，以保留SE_UserHandlerWrapper符号
6. 修改SBSFU xxx_flash.icf链接文件，将SE_IF_Code_Entry符号(SE_UserHandlerWrapper)放在SE_IF_ROM_region的开头。
7. FreeRTOS相关：通过添加0x10将__ICFEDIT_SE_IF_region_ROM_start_bit[4]强制为1来修改mapping_sbsfu.icf。这是PendSV处理程序的要求(FPU寄存器保存/恢复机制)。

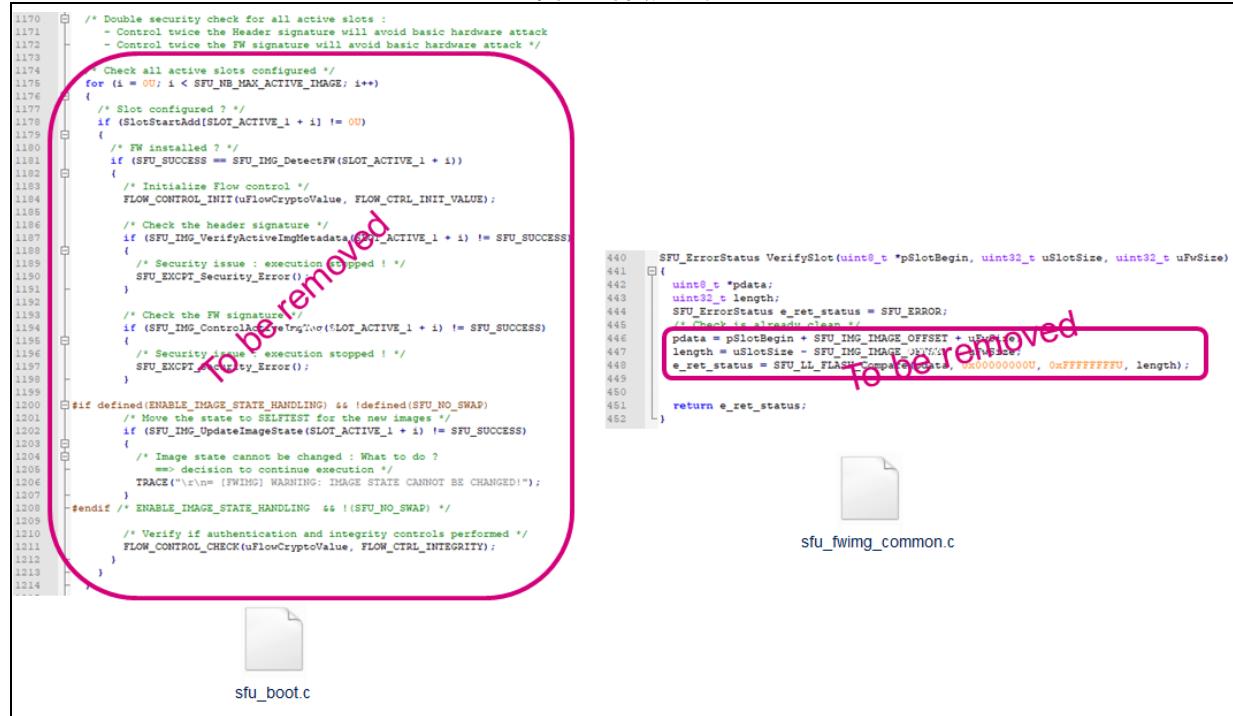
图33. IDE调整



7.4 如何改善启动时间

为了抵御基本故障注入攻击，会重复一些关键操作，从而会影响用户应用程序的启动时间。如果无需此类保护（例如，没有对设备的物理访问），则可以删除相应处理，如图 34 中所示。

图34. 启动时间



```

1170  /* Double security check for all active slots :
1171   - Control twice the Header signature will avoid basic hardware attack
1172   - Control twice the FW signature will avoid basic hardware attack */
1173
1174  /* Check all active slots configured */
1175  for (i = 0U; i < SFU_NB_MAX_ACTIVE_IMAGE; i++)
1176  {
1177    /* Slot configured ? */
1178    if (SlotStartAdd(SLOT_ACTIVE_1 + i) != 0U)
1179    {
1180      /* FW installed ? */
1181      if (SFU_SUCCESS == SFU_IMG_DetectFW(SLOT_ACTIVE_1 + i))
1182      {
1183        /* Initialize Flow control */
1184        FLOW_CONTROL_INIT(uFlowCryptoValue, FLOW_CTRL_INIT_VALUE);
1185
1186        /* Check the header signature */
1187        if (SFU_SUCCESS == SFU_IMG_VerifyActiveImgMetadata(SLOT_ACTIVE_1 + i) != SFU_SUCCESS)
1188        {
1189          /* Security issue : execution stopped ! */
1190          SFU_EXCEPT_Security_Error();
1191        }
1192
1193        /* Check the FW signature */
1194        if (SFU_IMG_ControlFWImg((SLOT_ACTIVE_1 + i) != SFU_SUCCESS)
1195        {
1196          /* Security issue : execution stopped ! */
1197          SFU_EXCEPT_Security_Error();
1198        }
1199
1200 #if defined(ENABLE_IMAGE_STATE_HANDLING) && !defined(SFU_NO_SWAP)
1201   /* Move the state to SELFTEST for the new images */
1202   if (SFU_IMG_UpdateImageState(SLOT_ACTIVE_1 + i) != SFU_SUCCESS)
1203   {
1204     /* Image state cannot be changed : What to do ?
1205      => decision to continue execution */
1206     TRACE("\r\n(FWHS) WARNING: IMAGE STATE CANNOT BE CHANGED!");
1207   }
1208 #endif /* ENABLE_IMAGE_STATE_HANDLING && !(SFU_NO_SWAP) */
1209
1210   /* Verify if authentication and integrity controls performed */
1211   FLOW_CONTROL_CHECK(uFlowCryptoValue, FLOW_CTRL_INTEGRITY);
1212 }
1213
1214

```

```

440 SFU_ErrorStatus VerifySlot(uint8_t *pSlotBegin, uint32_t uSlotSize, uint32_t uFwSize)
441 {
442   uint8_t *pdata;
443   uint32_t length;
444   SFU_ErrorStatus e_ret_status = SFU_ERROR;
445
446   /* Check if already clean */
447   pdata = pSlotBegin + SFU_IMG_IMAGE_OFFSET + uFwSize;
448   length = uSlotSize - SFU_IMG_IMAGE_OFFSET - uFwSize;
449
450   e_ret_status = SFU_LL_FLASH_CompareData((uint32_t)0x00000000U, 0xFFFFFFFFU, length);
451
452   return e_ret_status;

```

sfu_boot.c

sfu_fwimg_common.c

8 调整用户应用程序

8.1 如何使应用程序兼容SBSFU

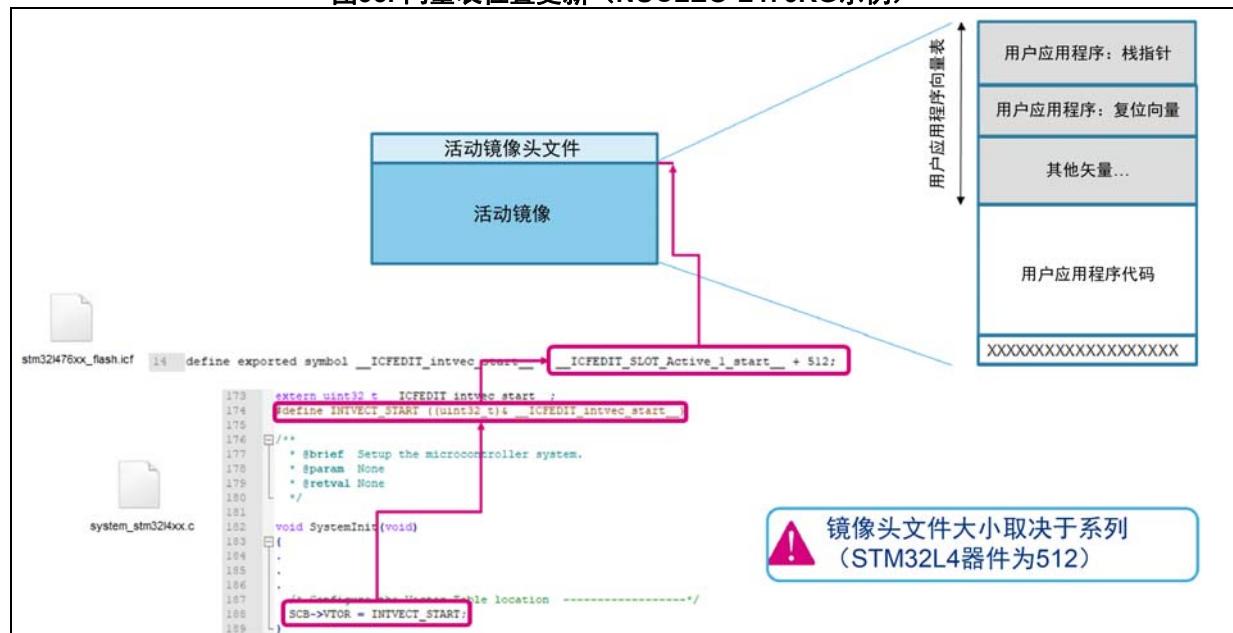
首先，必须修改用户应用程序的映射，以允许应用程序在激活插槽#1中运行。在多镜像配置中，这同样适用于活动插槽#2或#3：

- 由向量表启动的代码段必须配置为从激活插槽#1运行，并跟在镜像头文件之后：
__ICFEDIT_SLOT_Active_1_start__ + 512 (SFU_IMG_OFFSET = STM32L4系列为512)
- 数据段必须在安全引擎保护区域之后启动：(__ICFEDIT_SE_region_SRAM1_end__ + 1)

请参考[第 3.2 节：内存映射定义第 10 页](#)获取更多关于内存约束的详细信息。

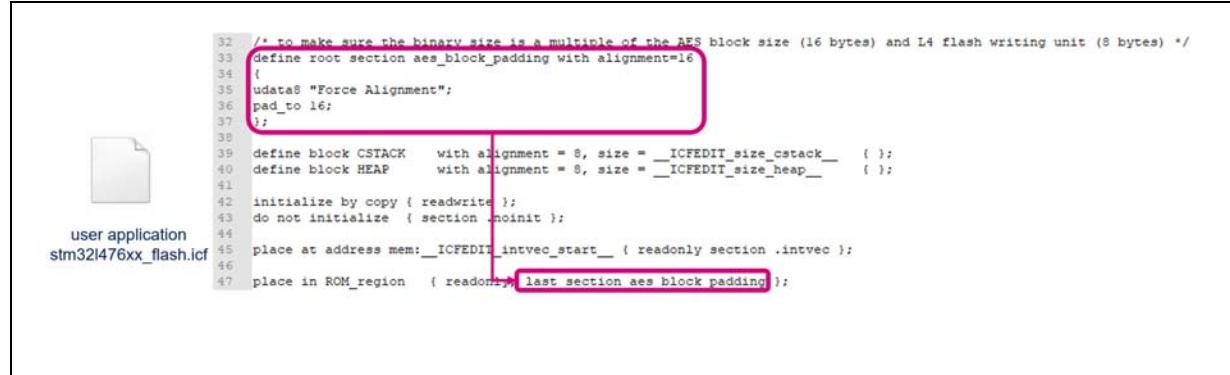
然后，在系统初始化时，必须将VTOR设置为向量表的新位置，如[图 35](#)中所示。

图35. 向量表位置更新 (NUCLEO-L476RG示例)



对于用户应用程序加密，用户应用程序二进制文件长度必须是16字节的倍数。图 36 演示如何更新链接器文件以验证此约束。

图36. 用户应用程序二进制文件长度



```

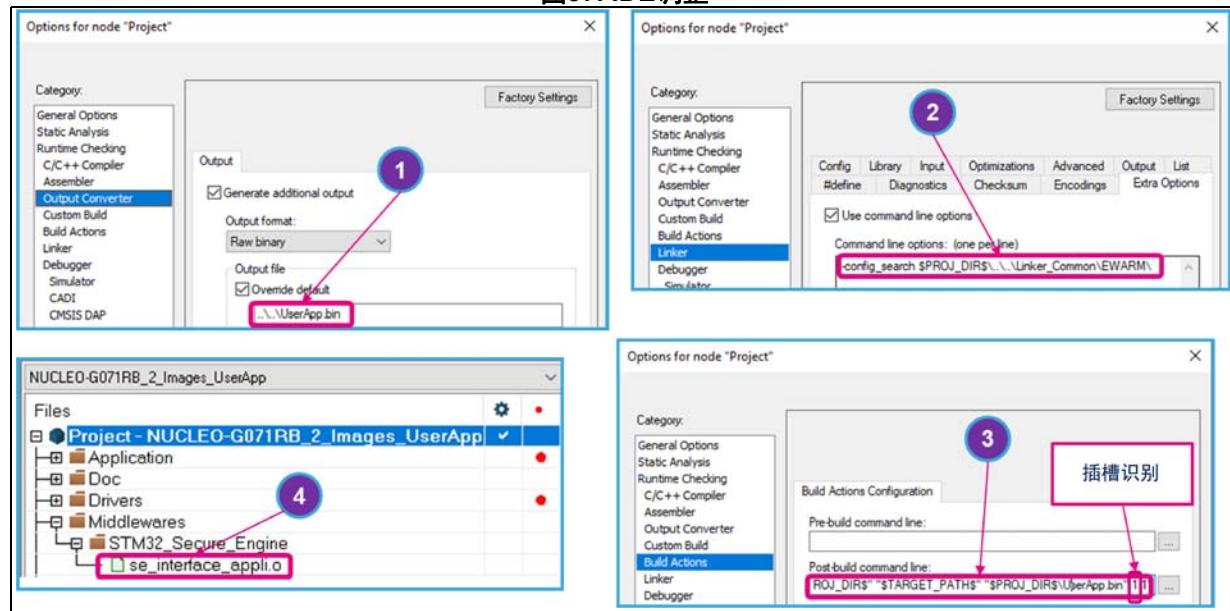
32 /* to make sure the binary size is a multiple of the AES block size (16 bytes) and L4 flash writing unit (8 bytes) */
33 define root section aes_block_padding with alignment=16
34 {
35     udata8 "Force Alignment";
36     pad_to 16;
37 }
38
39 define block CSTACK with alignment = 8, size = __ICFEDIT_size_cstack__ { };
40 define block HEAP with alignment = 8, size = __ICFEDIT_size_heap__ { };
41
42 initialize by copy { readwrite };
43 do not initialize { section .noinit };
44
45 place at address mem:_ICFEDIT_intvec_start_ { readonly section .intvec };
46
47 place in ROM_region { readonly last section aes block padding };

```

最后，如UserApp示例中所述，必须更新IDE配置，以便：

1. 生成UserApp.bin文件
2. 包括链接器通用文件的搜索路径
3. 调用postbuild.bat，以生成具有正确插槽标识(1/2/3)的UserApp.sfb和SBU_UserApp.bin
4. 集成se_interface_appli.o，以访问安全引擎实时服务（如果有）

图37. IDE调整



如用户手册UM2262中的解释，还存在一些额外的约束，具体取决于STM32系列：

- STM32F4系列、STM32F7系列和STM32L1系列：基于MPU的安全引擎隔离完全依赖于这样一个事实 - 访问安全引擎服务需要软件执行于特权级。用户应用程序必须考虑这种情况，并信任以特权模式运行的任何代码段。
- STM32G0系列、STM32G4系列、以及STM32H7系列：在安全状态下，任何对安全内存区域的访问（取、读、编程、擦除）都被拒绝，并生成一个总线错误。因此，用户应用程序不支持安全引擎实时服务。

注： *IWDG在SBSFU执行期间启动。必须对其进行定期刷新。*

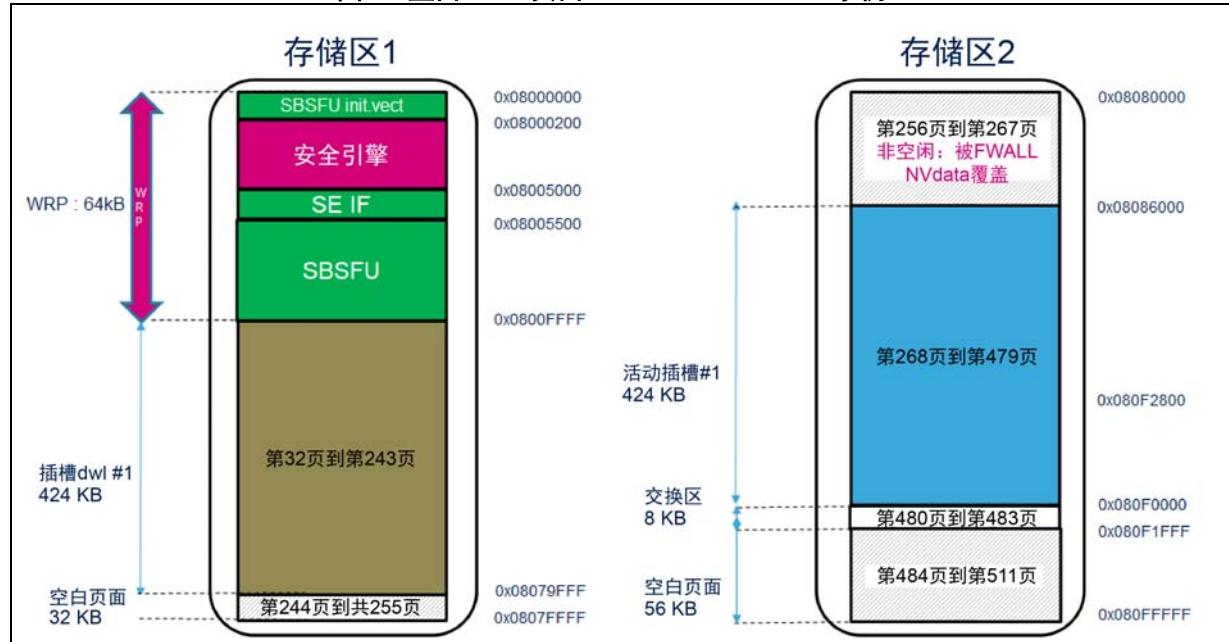
8.2 使用闪存存储用户数据

在闪存页面或扇区存储用户数据可能有一些限制：

- 在SBSFU代码区域之外
- 不在镜像插槽中
- 不在交换区域中

图 38 提供一个基于 NUCLEO-L476RG 的内存映射示例，此处的闪存（从 489 页到 511 页）供用户存储数据、安装文件系统或模拟 EEPROM。

图38. 空白Flash页面 (NUCLEO-L476RG示例)

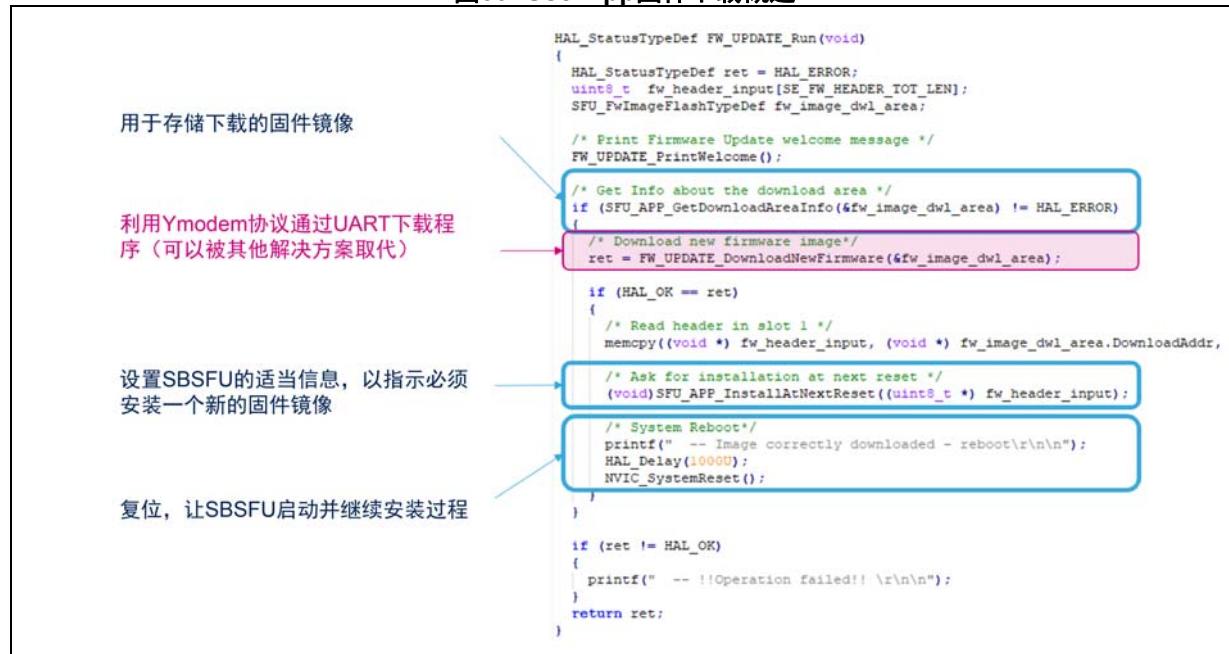


8.3 更改用户应用程序中的固件下载功能

只有在双插槽操作模式下才存在这种可能性。

X-CUBE-SBSFU UserApp项目中提供了一段基于YMODEM协议（通过UART）的示例代码。下载流程在文件fw_update_app.c中，如图 39中所示。

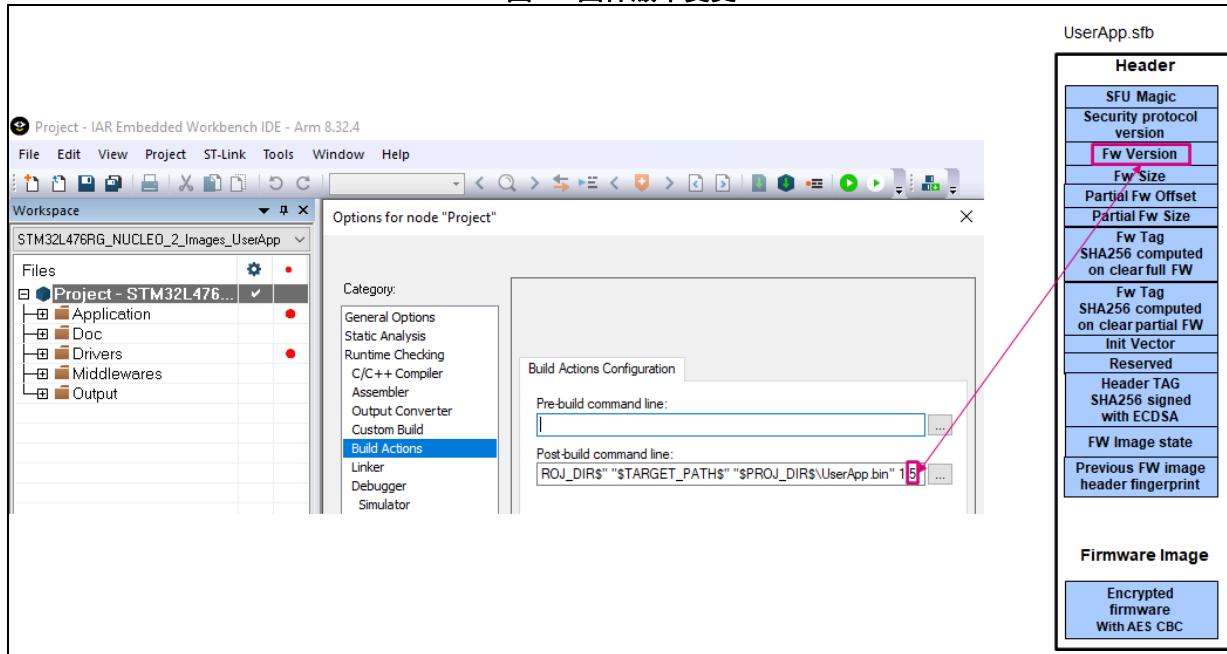
图39. UserApp固件下载概述



8.4 如何变更固件版本

固件版本是用*postbuild.bat*脚本生成的固件头文件的一部分。在下面的示例中，版本为5。

图40. 固件版本变更



注意：当安装的镜像的头文件无效时，版本为SFU_FW_VERSION_INIT_NUM *app_sfu.h*的固件是唯一允许安装的固件。出现这种情况的原因可能是未安装固件（开发阶段）或攻击尝试。保持此类固件的私密性很重要，因为此类固件的唯一目的是分析和维修从现场返回的设备。

8.5 如何验证固件镜像

首先，必须在SECoreBin、SBSFU和UserApp IDE配置中定义ENABLE_IMAGE_STATE_HANDLING编译开关。

在用户应用程序首次启动时，如果执行正确（如在首次测试执行后），用户应用程序必须调用运行服务*SE_APP_Validate(slot_id)*（如果可用）或更新RAM中的专用标志，以验证固件镜像。如果未执行该操作，则会在下次复位时通过SBSFU回滚至上一固件镜像。

在用户应用程序中通过菜单*FW_VALIDATE_RunMenu()*提供了一个示例，如图41中所示。在多镜像配置中，插槽识别参数可能为1、2、3或255。值255表示通过单个请求来验证所有新固件镜像。目标为在验证阶段出现中断时确保所有新镜像之间的固件兼容性。

图41. 验证菜单

```
===== Validation of FW Image =====  
Validate all firmwares ----- 0  
Validate firmware of SLOT_ACTIVE_1 ----- 1  
Validate firmware of SLOT_ACTIVE_2 ----- 2  
Validate firmware of SLOT_ACTIVE_3 ----- 3  
Previous Menu ----- x  
Selection :
```

注意： 仅在选择互换安装过程的双插槽配置示例中才能激活此功能。

9 版本历史

表4. 文档版本历史

日期	版本	变更
2017年12月20日	1	初始版本。
2018年8月31日	2	文件结构和内容全部更新： – 重新关注 简介 中的集成主题 – 适应于非对称和对称加密方案 – 适应于单镜像和双镜像模式
2018年12月18日	3	产品范围扩展到STM32F4系列、 STM32F7系列和STM32G0系列： – 更新了 第1章：概述 、 第2章：相关文档 、 第3.2节：内存映射定义 、 第4.3节：安全配置 、 节：图15显示文件app_sfu.h中针对STM32WB系列提供的各种安全配置解决方案 、以及 第8.1节：如何使应用程序SBSFU兼容 – 添加了 第7章：调整SBSFU 安全库提供扩展到mbedTLS： – 更新了 第4.1节：要配置的特性
2019年9月6日	4	更新了 简介 。 产品范围扩展到STM32H7系列、 STM32G4系列、STM32L0系列、STM32L1系列、以及 STM32WB系列。 更新了 第2章：相关文档 。 更新了 第3.1节：硬件适应 更新了 第3.2节：内存映射定义 修改了 第3.2.1节：SBSFU区域定义参数 和 第3.2.2节：固件镜像插槽定义参数 更新了 第17页上的第4.1节 更新了 第4.3章节：安全配置 （更新了图形并添加了 图18：STM32WB系列安全配置 (app_sfu.h) 在 第18页上的第4.2节 中添加了注释。 在 第4.4节：开发或生产模式配置 中修改了选项字节配置。 添加了 第5.3节：STM32WB系列特殊性 、 第5.4节：KMS特殊性 、以及 第5.5节：STSAFE-A100特殊性 。 在 第7.2节：优化内存映射 中更新了 表3 添加了 第8.4节：如何用BLE OTA加载器替换独立加载器 和 第8.5节：如何变更固件版本 。

表4. 文档版本历史 (续)

日期	版本	变更
2020年7月9日	5	<p>在第4.1节：要配置的功能和第4.2节：加密方案选择中新增了OTFDEC信息（新增了一个注释）</p> <p>更新了第3.2.2节：固件镜像插槽定义参数。</p> <p>新增了图8：双存储区产品的防火墙配置约束和图9：交换存储区之后的防火墙配置。</p> <p>更新了图11：特定于SBSFU的链接器文件、图12：特定于UserApp的链接器文件 (NUCLEO-L476RG示例)和图13：SBSFU配置。</p> <p>更新了第4.4节：开发或生产模式配置、第6.2节：内存映射调整、第7.2节：优化内存映射</p> <p>移除了图28：NUCLEO-G031K8上的内存映射优化示例—1个镜像。</p>
2020年9月1日	6	<p>增加了：</p> <ul style="list-style-type: none"> – 第3.2.4节：多镜像配置 – 第3.3节：双核适配 – 第7.3节：如何激活隔离防火墙环境内部的中断管理 – 第7.4节：如何改善启动时间 – 第8.6节：如何验证固件镜像 <p>更新了：</p> <ul style="list-style-type: none"> – 将安全元件STSAFE-A100替换为STSAFE-A110
2021年7月22日	7	<p>增加了：</p> <ul style="list-style-type: none"> – 第3.2.2节：固件镜像插槽定义参数 STM32H7系列的SFU_IMAGE_OFFSET值 – 第4.1节：要配置的特性 不支持动态解密的外部Flash存储器 <p>更新了：</p> <ul style="list-style-type: none"> – 第3.2节：内存映射定义和第5.1节：生成UM2262新的固件AES加密密钥参考 – 第5.3节：STM32WB系列特殊性，新增了图24：密钥配置 <p>删除了：</p> <ul style="list-style-type: none"> – 先前的8.4节 如何用BLE OTA加载器替换独立加载器37
2021年12月14日	8	<p>更新了：</p> <ul style="list-style-type: none"> – 面向STM32WB5MM-DK探索板的第 4.1节：要配置的特性 – 用于在P-NUCLEO-WB55 Nucleo和STM32WB5MM-DK探索板上选择启动模式的第 5.3节：STM32WB系列特殊性 – 第 8.5节：如何验证固件镜像

表5. 中文文档版本历史

日期	版本	变更
2022年9月5日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST产品和/或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST产品的最新信息。ST产品的销售依照订单确认时的相关ST销售条款。

买方自行负责对ST产品的选择和使用，ST概不承担与应用协助或买方产品设计相关的任何责任。

ST不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST产品如有不同于此处提供的信息的规定，将导致ST针对该产品授予的任何保证失效。

ST和ST徽标是ST的商标。若需ST商标的更多信息，请参考 www.st.com/trademarks。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。

© 2022 STMicroelectronics - 保留所有权利