life.augmented

# UM2262
# 用户手册

## X-CUBE-SBSFU STM32Cube扩展包入门

## 引言

本用户手册介绍了如何开始使用X-CUBE-SBSFU STM32Cube扩展包。

X-CUBE-SBSFU安全启动和安全固件更新解决方案使STM32微控制器内部固件可升级到新版本，添加新功能，和纠正潜在问题。升级过程是以安全的方式进行，以防未经授权的更新，并阻止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、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无法运行。

安全固件更新应用通过使用Ymodem协议的UART接口接收固件映像，检查其真实性，并在安装代码之前检查其完整性。固件更新可更新整个固件镜像，或者部分镜像。单槽示例适用于固件映像最小化的情况，而双槽示例则适用于确保映像安全安装的情况，且也适用于IoT设备中常用的无线固件更新功能。对于复杂系统，固件映像配置可扩展至最多三个映像。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。

安全密钥管理服务（KMS）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基于STM32Cube软件技术，易于在不同STM32微控制器之间移植。它以参考代码的形式提供，演示了STM32安全保护措施的最佳使用方式。

X-CUBE-SBSFU归类为ECCN 5D002。

# 目录

# 表格索引

# 图片索引

# 1 概述

X-CUBE-SBSFU扩展包附带在STM32F4系列、STM32F7系列、STM32G0系列、STM32G4系列、STM32H7系列、STM32L0系列、STM32L1系列、STM32L4系列、STM32L4+系列和STM32WB系列产品上运行的示例。此外，还为STM32L4+系列提供了一个结合了STM32微控制器和STSAFE-A110的示例。

X-CUBE-SBSFU以独立STM32系统解决方案示例的参考代码的形式提供，这些示例演示了用来保护资产免遭未授权外部和内部访问的STM32保护措施的最佳使用方式。X-CUBE-SBSFU还提供了一个结合了STM32和STSAFE-A110的系统解决方案示例，演示了面向安全身份验证服务和安全数据存储的硬件安全元件保护。

X-CUBE-SBSFU是OEM根据其产品的安全要求级别开发安全启动和安全固件更新的起点。

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STM3232位微控制器上运行。

## 1.1 术语和定义

*表 1*给出了相关的缩略语定义，帮助您更好地理解本文档。

**表1. 缩略语列表**

| 缩略语 | 说明 |
|---|---|
| AAD | 附加认证数据 |
| AES | 高级加密标准 |
| CBC | AES 密码块链接 |
| CKS | 客户密钥存储 |
| CTR | 基于AES计数器的密码模式 |
| DMA | 直接存储器访问 |
| DSA | 数字签名算法 |
| ECC | 椭圆曲线加密 |
| ECCN | 出口控制分类号 |
| ECDSA | 椭圆曲线数字签名算法 |
| FSM | 有限状态机 |
| GCM | AES Galois/计数器模式 |
| GUI | 图形用户界面 |
| HAL | 硬件抽象层 |
| HDP | 隐藏保护存储区（又名安全用户存储区） |
| IDE | 集成开发环境 |

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

表1. 缩略语列表（续）

| 缩略语 | 说明 |
|---|---|
| IV | 初始化向量 |
| IWDG | 独立看门狗 |
| FW | 固件 |
| FWALL | 防火墙 |
| KMS | 密钥管理服务 |
| MAC | 消息认证码 |
| MCU | 微控制器单元 |
| MPU | 存储器保护单元 |
| NONCE | 仅使用一次的数值 |
| OTFDEC | 动态解密 |
| PCROP | 专有代码读保护 |
| PEM | 隐私增强邮件 |
| RDP | 读保护 |
| SB | 安全启动 |
| SE | 安全引擎 |
| SFU | 安全固件更新 |
| SM | 状态机 |
| UART | 通用异步收发器 |
| UUID | 通用唯一标识符 |
| WRP | 写保护 |

*表 2*给出了相关的术语定义，帮助您更好地理解本文档。

表2. 术语列表

| 术语 | 说明 |
|---|---|
| 固件镜像 | 二进制映像（可执行文件），作为用户应用程序由设备来运行。 |
| 固件头文件 | 元数据包，描述要安装的固件镜像。它包含固件信息和密码信息。 |
| mbedTLS | TLS和SSL协议的mbed实现以及各自的密码算法。 |
| *sfb*文件 | 包含固件头文件和固件镜像的二进制文件。 |

## 1.2 参考

### 意法半导体的相关文档

意法半导体网站*www.st.com*在线提供了公开文档。如需详细信息，请与意法半导体联系。

1. 应用笔记*X-CUBE-SBSFU STM32Cube扩展包集成指南*（AN5056）
2. 应用笔记*STM32微控制器安全入门*（AN5156）
3. *STM32CubeProgrammer软件描述*用户手册(UM2237)
4. *认证、最先进的外设安全措施以及IoT设备*数据手册(DS12911)

### 其他文档

5. *PKCS #11密码令牌接口基本规范版本2.40 + 勘误表*
   http://docs.oasis-open.org/pkcs11/pkcs11-curr/v2.40/os/pkcs11-curr-v2.40-os.html

# 2 STM32Cube 概述

### STM32Cube是什么？

STM32Cube源自意法半导体，旨在通过减少开发工作量、时间和成本，明显提高设计人员的生产率。STM32Cube涵盖整个STM32产品系列。

STM32Cube 包括：

- 一套用户友好的软件开发工具，覆盖从概念到实现的整个项目开发过程，其中包括：
  - 图形软件配置工具STM32CubeMX，可通过图形向导自动生成初始化C代码
  - STM32CubeIDE，一种集外设配置、代码生成、代码编译和调试功能于一体的开发工具
  - STM32CubeProgrammer（STM32CubeProg），图形版本和命令行版本中可用的编程工具
  - STM32CubeMonitor（STM32CubeMonitor、STM32CubeMonPwr、STM32CubeMonRF和STM32CubeMonUCPD）是功能强大的监控工具，用于实时微调STM32应用的行为和性能
- STM32Cube MCU和MPU包，特定于每个微控制器和微处理器系列的综合嵌入式软件平台（如用于STM32L4系列和STM32L4+系列的STM32CubeL4），其中包含：
  - STM32Cube硬件抽象层（HAL），确保在STM32各个产品之间实现最大限度的可移植性
  - STM32Cube底层API，通过硬件提供高度用户控制，确保最佳性能和内存开销
  - 一组一致的中间件组件，如FAT文件系统、RTOS、OpenBootloader、USB主机、USB设备、TCP/IP、触摸感应库，以及图形库
  - 包含的软件覆盖了全套外设以及对应可用的示例
- STM32Cube扩展包，包含的软件组件是STM32Cube MCU和MPU包的功能补充：
  - 中间件扩展和应用层
  - 在特定的意法半导体开发板上运行的实现案例

### 此软件如何补充STM32Cube？

该软件基于STM32CubeHAL，即STM32微控制器的硬件抽象层。此软件包通过提供以下中间件组件扩展STM32Cube：

- 用于管理所有关键数据和操作（如访问固件加密密钥的加密操作等）的安全引擎
- 通过PKCS #11 API提供密码服务的密钥管理服务
- 用于管理硬件安全元件特性的STSAFE-A

该软件包包含有不同的示例应用程序，可提供完整的SBSFU解决方案：

- SE_CoreBin应用程序：提供包含所有“可信”代码的二进制文件。
- 安全启动和安全固件升级（SBSFU）应用程序：
  - 安全启动（可信根）
  - 本地下载通过串口
  - 固件安装管理
- 用户应用程序：
  - 在双插槽操作模式下下载新固件
  - 提供了测试保护机制的示例
  - 提供了使用KMS API的示例

示例应用提供了双插槽和单插槽操作模式，并且可以配置为不同加密方案。

*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。*

本用户手册介绍了软件包的典型用法：

- 基于NUCLEO-L476RG板
- 示例应用在双插槽模式下运行，并配置了非对称身份验证和对称固件加密

有关配置选项和单槽操作模式的更多信息，请参阅本文档的附录。

*注：* *STM32L4系列和STM32L4+系列提供KMS功能，所提供示例基于B-L475E-IOT01A和B-L4S5I-IOT01A板。*

*注：* *STM32L4+系列提供STSAFE-A110功能，所提供示例基于B-L4S5I-IOT01A板。*

# 3 安全启动和安全固件升级（SBSFU）

## 3.1 产品安全介绍

现场部署的设备在不受信任的环境中运行，因此会受到威胁和攻击。为了减轻受攻击风险，我们的目标是只在设备上运行可靠的固件。已连接的设备时常需要更新固件映像以修复错误，或引入新功能或对策，否则极易遭受攻击。

其后果可能是破坏性的，如固件克隆、恶意软件下载或设备损坏。因此必须设计出一套安全的解决方案来保护敏感数据（甚至可能是固件本身）和关键操作。

典型的对策基于加密技术（带有相关密钥）和内存保护：

- 密码可确保固件传输期间的完整性（确保数据未被破坏）、身份验证（确保某个实体确实符合其声明）以及机密性（确保只有经过授权的用户才能读取敏感数据）。
- 内存保护机制可以防止外部攻击（例如，通过JTAG物理访问设备）以及来自内部其它进程的攻击。

以下章节介绍实现机密性、完整性和身份验证服务的解决方案，以解决IoT终端节点设备最常见的威胁。

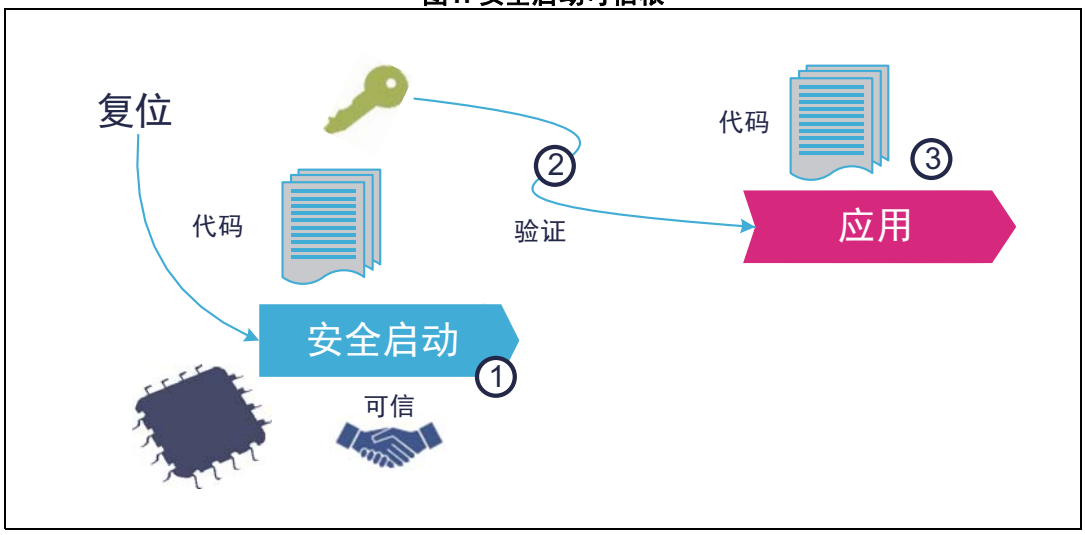
## 3.2 安全启动

安全启动（SB）确保所执行的用户应用程序映像的完整性和真实性：使用密码检查，防止运行未经授权或恶意修改的软件。安全启动过程实现可信根（参见*图 1*）：从该可信组件（1）开始，其他每个组件在其执行之前（3）都要经过认证（2）。

对**完整性**进行验证，以确保即将执行的映像未被破坏或恶意修改。

**可靠性**检查旨在验证固件映像是来自可信且已知的源，以防止未经授权的实体安装及执行代码。

图1. 安全启动可信根



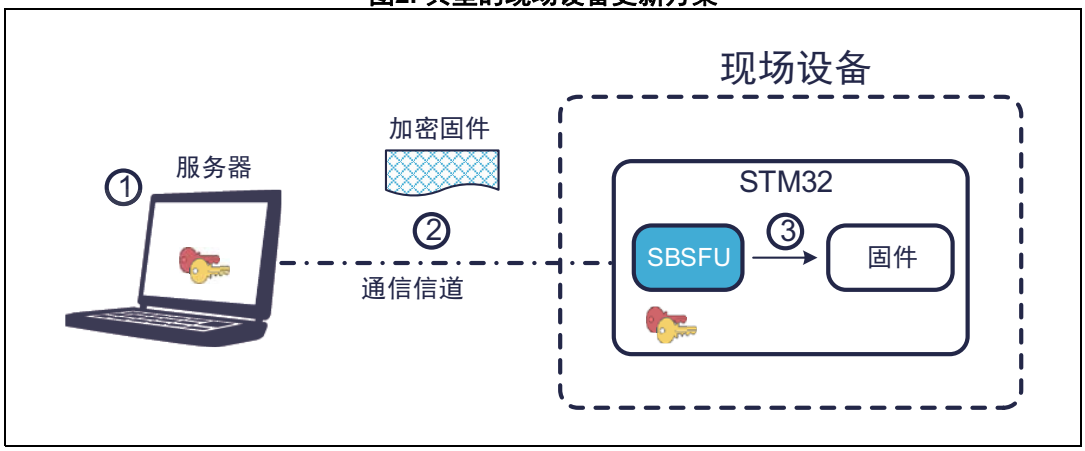

## 3.3 安全固件更新

安全固件更新（SFU）实现了安全的现场固件更新，可以安全地将新固件映像下载到设备。

如*图 2*中所示，通常有两个实体参与固件更新过程：

- 服务器
  - OEM制造商服务器/Web服务
  - 存储设备固件的新版本
  - 与设备通信，如果可用，则以加密形式发送该新映像版本
- 设备
  - 现场部署
  - 内置了执行固件更新过程的代码。
  - 与服务器通信并接收新的固件映像。
  - 验证、解密并安装新固件映像，然后执行它。

图2. 典型的现场设备更新方案



固件更新通过以下步骤进行：

1. 如果需要更新固件，则创建一个新的加密固件映像并将其存储在服务器中。
2. 新的加密固件映像通过不受信任的通道发送到现场部署的设备。
3. 下载、检查并安装新映像。

可对整个固件映像或仅固件映像的一部分（仅适用于双插槽配置）执行固件更新。

固件更新容易受到*第 3.1 节：产品安全介绍*中所示风险的影响：密码学技术用来确保机密性、完整性和身份验证。

**机密性**：用以保护固件映象，此固件映像可能是制造商的关键资产。它以加密的方式通过不可信通道发送，因此只有具备加密密钥访问权的设备才能解密固件包。

**完整性**：用来验证接收到的映象无任务损坏。

**可靠性**：检查旨在验证固件映像是来自可信且已知的源，以防止未经授权的实体安装及执行代码。

## 3.4 密码操作

X-CUBE-SBSFU STM32Cube扩展包提供了同时使用非对称和对称加密的四种加密方案。

默认的加密方案演示了用于固件验证的ECDSA非对称加密和用于固件解密的AES-CBC对称加密。利用非对称加密技术，固件验证可以通过公钥操作执行，因此设备中不需要任何保密信息。

X-CUBE-SBSFU扩展包中提供的其他加密方案是：

- 用于固件验证的ECDSA非对称加密技术，用AES-CBC或AES-CTR对称加密技术进行固件加密
- 用于明文固件验证的ECDSA非对称加密技术
- 用于明文固件验证的基于X509证书的ECDSA非对称加密技术
- 用于固件验证和加密的AES-GCM对称加密技术。

*表 3*介绍了每种密码方案相关的各种安全特性。

**表3. 密码方案比较**

| 特性 | 非对称<br>使用AES加密 | 非对称<br>不加密 | 基于X509证书的非对称<br>不加密 | 对称<br>（AES-GCM）(1) |
|---|---|---|---|---|
| 机密性 | AES-CBC加密，或者支持OTFDEC处理的STM32 MCU的AES-CTR加密（固件二进制文件） | 无，用户固件为明文格式。 | | AES-GCM加密（固件二进制文件） |
| 完整性 | SHA256（固件头文件和固件二进制文件） | | | AES-GCM标记（固件头文件和固件二进制文件） |
| 认证 | – 固件头文件的SHA256是ECDSA签名的<br>– 固件二进制文件的SHA256存储在固件头文件中 | | | |
| 设备中的密钥 | AES-CBC / AES-CTR私钥（秘密）<br>ECDSA公钥 | ECDSA公钥 | X509证书链中的ECDSA公钥（存储在STSAFE-A或KMS中） | AES-GCM私钥（秘密） |

1. 对于对称密码方案，强烈建议为每个产品配置唯一的对称密钥。

# 4 密钥管理服务(KMS)

密钥管理服务中间件通过标准PKCS #11 API（由OASIS指定）提供密码服务，这些API能够将密钥值提取给调用方（使用对象ID而非直接使用密钥值）。KMS运行在一个受保护/隔离的环境之内，主要是为了确保密钥值本身不能被未授权的外部代码访问。

KMS也可以提供一些密码学服务，且由STM32微控制器以外的元件来安全管理，比如STSAFE-A110（基于令牌ID）。

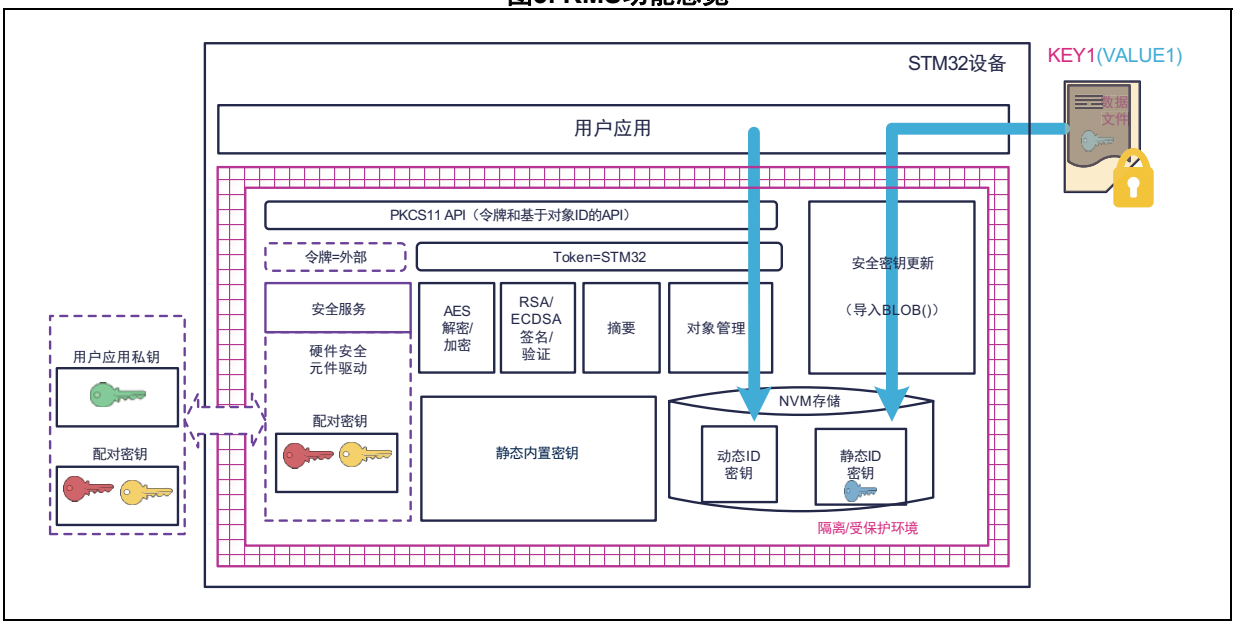
KMS只支持PKCS #11 API的子集：

- 对象管理功能：创建/更新/删除
- AES加密功能
- AES解密功能
- 摘要功能
- RSA和ECDSA签名/验证功能
- 密钥管理功能：密钥生成/派生

KMS管理三类密钥：

- 静态嵌入式密钥：
  - 在代码中嵌入预定义密钥。此类密钥不能修改。
- 具有静态ID的可更新密钥：
  - 在系统中预定义密钥ID
  - 可通过使用静态嵌入式根密钥的安全流程更新NVM存储器中的密钥值（真实性检查、数据完整性检查和数据解密）
  - 密钥无法删除
- 具有动态ID的可更新密钥：
  - 密钥ID在创建密钥时定义
  - 密钥值由使用内部函数来创建， 如DeriveKey()函数，可创建动态对象。
  - 密钥可以删除

**图3. KMS功能总览**




关于OASIS PKCS #11标准的更多信息，请参见*[5]*。

# 5 保护措施和安全策略

密码保证了完整性、真实性和机密性。但是，单独使用密码技术是不够的：需要一整套措施和系统级策略来保护关键操作和敏感数据（例如密钥），以及执行流程，以便抵御可能的攻击。

为抵御基本的故障注入攻击，实现了安全软件编码技术，如临界测试加倍、临界操作加倍、参数值检查和流控制机制测试。

安全策略基于以下理念：

- 确保复位时的单一入口点：通过强制代码从安全启动代码开始执行
- 确保SBSFU代码本身和SBSFU机密信息不可变：一旦安全保护完全激活，就不可能修改或更改它们
- 创建与SBSFU应用和用户应用相隔离的受保护隔离区，用于存储机密信息（如密钥）和运行关键操作（如密码算法）
- 限制应用代码仅可运行在SBSFU核心之外
- 免除对器件的JTAG访问
- 监控系统：入侵监测和SBSFU运行时间

*图 4*和*图 5*提供了每个STM32系列上激活的安全机制的高层视图。

**图4. SBSFU安全IP与STM32系列的对比（1/2）**

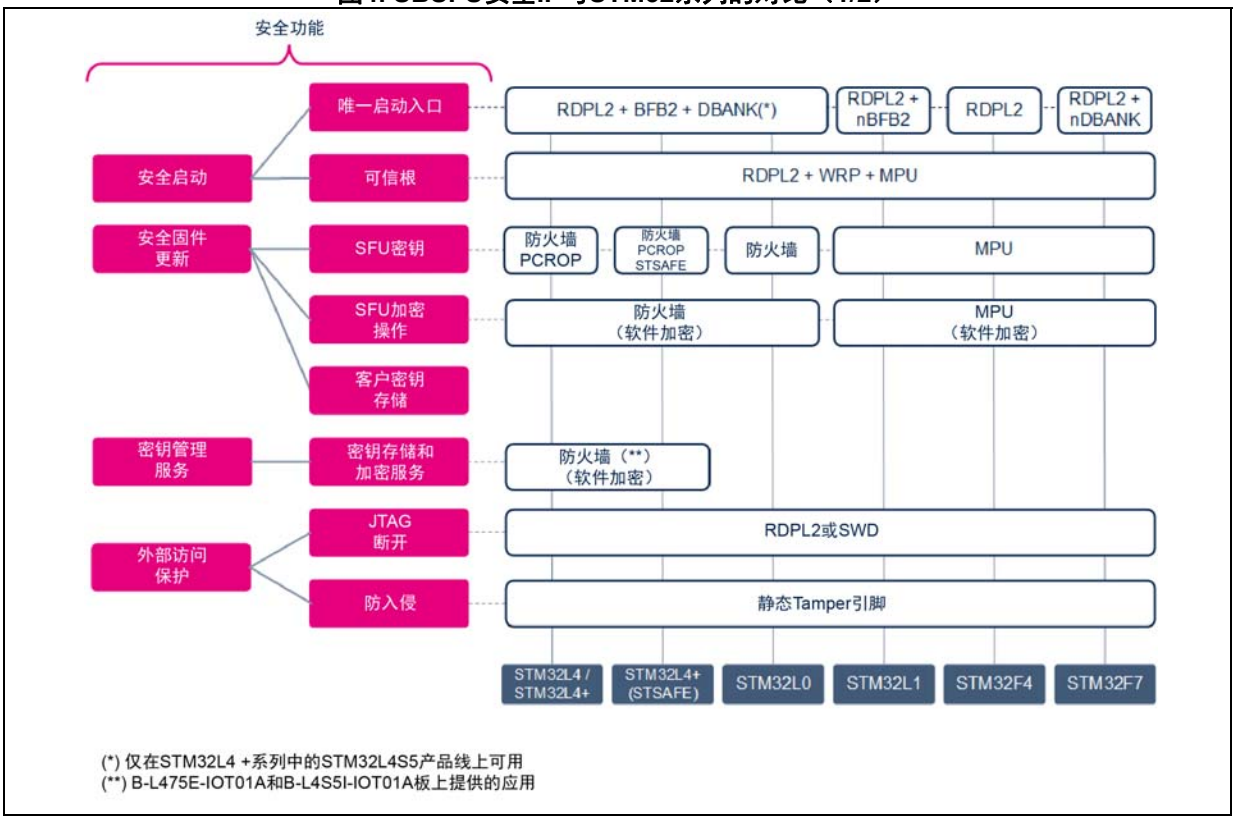
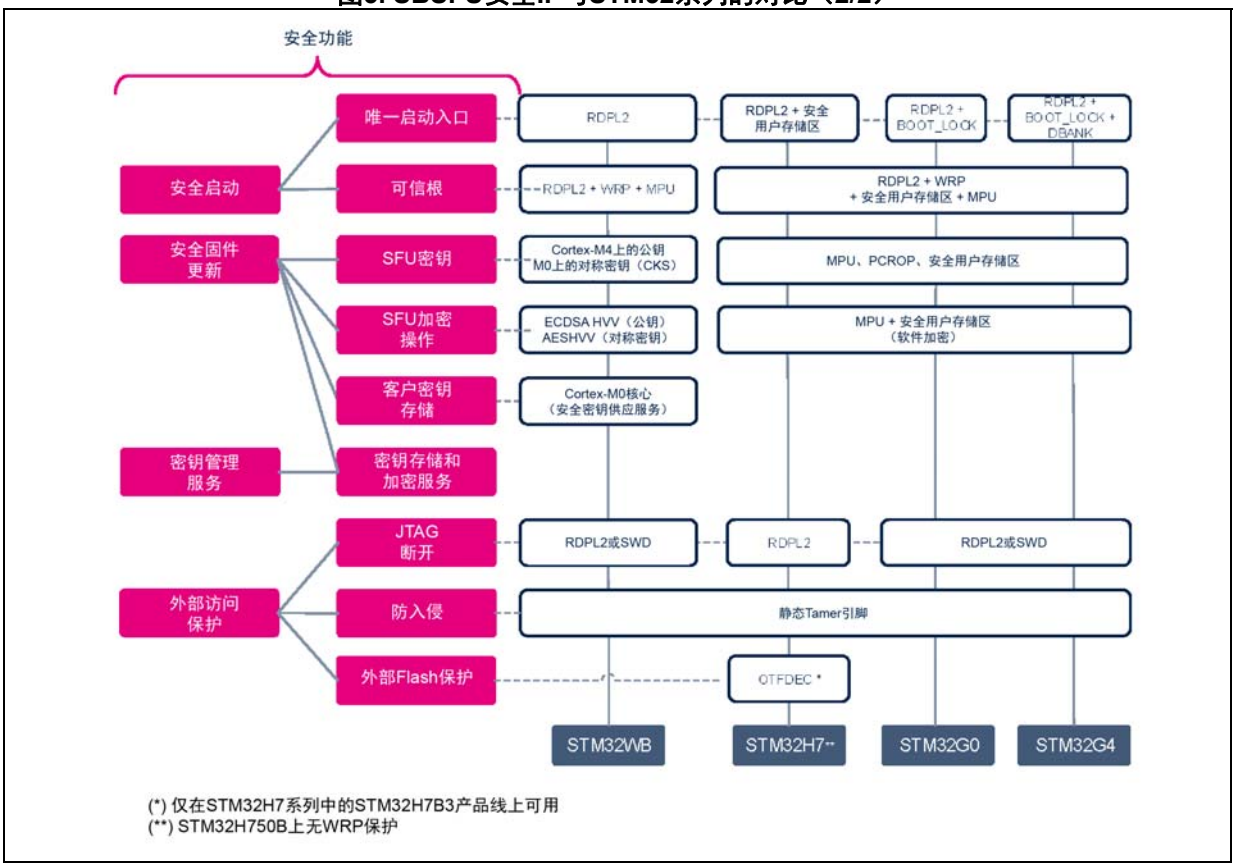


图5. SBSFU安全IP与STM32系列的对比（2/2）

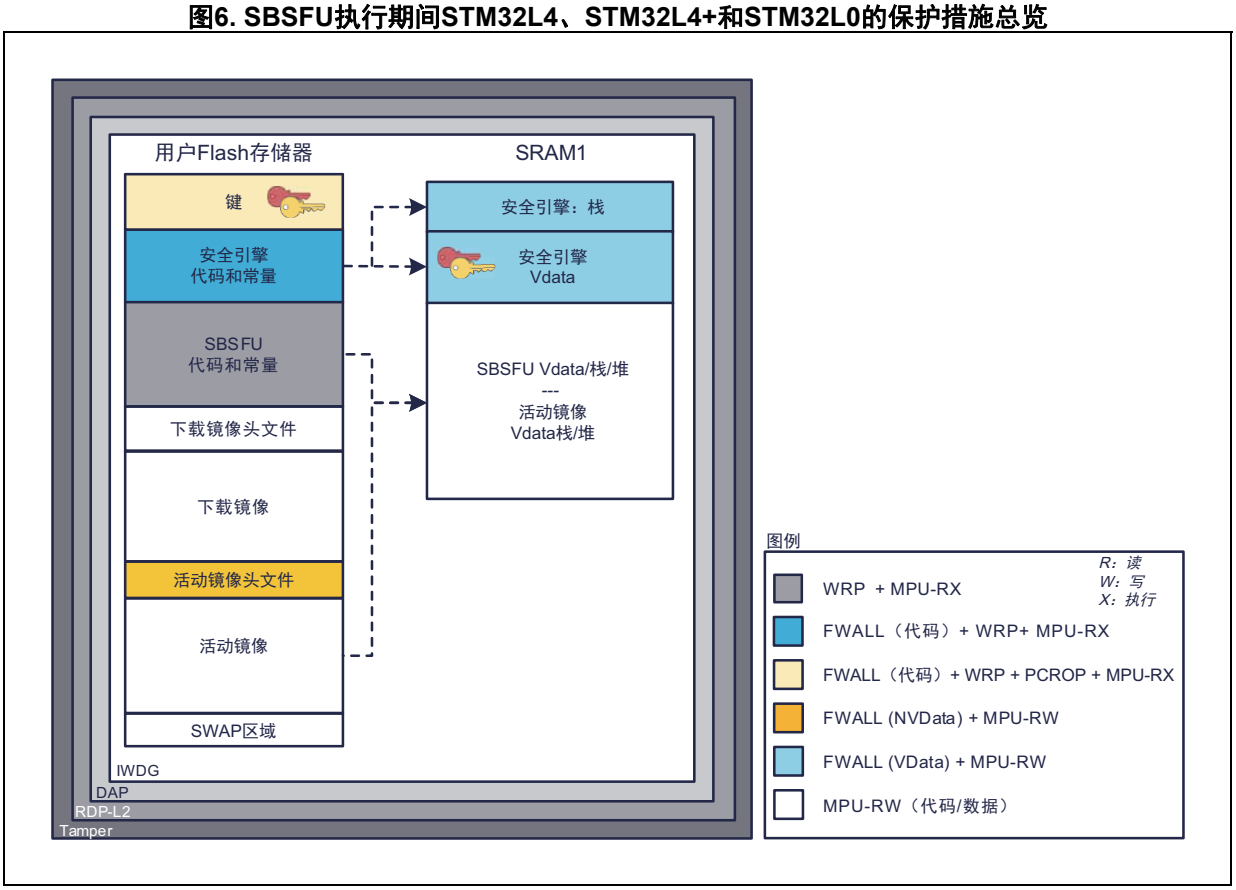



(*) 仅在STM32H7系列中的STM32H7B3产品线上可用
(**) STM32H750B上无WRP保护

## 5.1 STM32L4系列、STM32L4+系列和STM32L0系列

*图 6*说明了如何保护系统、代码和数据的STM32L4系列、STM32L4+系列和STM32L0系列的X-CUBE-SBSFU应用示例。

**图6. SBSFU执行期间STM32L4、STM32L4+和STM32L0的保护措施总览**



### 可抵御外部攻击的保护措施

外部攻击是指由外部工具触发的攻击，如调试器或探测器尝试访问设备时。在SBSFU应用示例中，RDP、tamper、DAP和IWDG保护用来保护产品免受外部攻击：

- **RDP**（读保护）：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：
  - 禁止通过JTAG硬件接口对RAM和Flash进行外部访问。这样可以防止改变SBSFU代码并因此来挖掘可信根的攻击。
  - 选项字节不能更改。这意味着其他保护措施如WRP和PCROP不能再进行更改。

  **注意** - 由于以下原因，不建议使用RDP级别1：

  1. 无法确保安全启动/可信根（单入口点和不可变代码），因为在RDP L1情况下可以修改选项字节（WRP）。

  2. 设备内部闪存可以利用新的固件进行完全重新编程（在通过RDP L0回归进行闪存整体擦除之后），无需任何安全措施。

  3. 系统复位时，通过JTAG硬件接口连接调试器，可以访问受防火墙保护的RAM中的秘密信息。

  如果客户产品上不能进行JTAG硬件接口访问，并且客户使用信任可靠的用户应用程序代码，则上述风险无效。
- **Tamper**：防篡改保护用于检测设备上的物理篡改行为并采取相应的应对措施。在篡改检测的情况下，SBSFU应用示例强制重启。
- **DAP**（调试访问端口）：DAP保护在于可以停用DAP（调试访问端口）。一旦停用，JTAG引脚不再连接到内部总线。使用RDP Level 2可自动禁用DAP。
- **IWDG**（独立看门狗）：IWDG是一个自由运行的减量计数器。一旦开始运行，它就不能再停止。在引起重置之前必须对其定期刷新。该机制能够控制SBSFU执行持续时间。

### 可抵御内部攻击的保护措施

内部攻击是指由STM32中运行的代码触发的攻击。攻击可能是由恶意固件利用错误或安全漏洞导致的，也可能由意外的操作引起。在SBSFU应用示例中，WRP、防火墙、PCROP和MPU保护可防止产品受到内部攻击：

- **FWALL**（防火墙）：对防火墙进行配置以保护代码、易失性和非易失性数据。受保护的代码可以通过单入口点进行访问（调用门机制在*附录A*中有说明）。任何试图跳过并尝试不通过入口点而执行代码段中函数的操作都会导致系统复位。
  在KMS示例中，密钥和密码服务是在受防火墙保护的隔离环境中执行的。

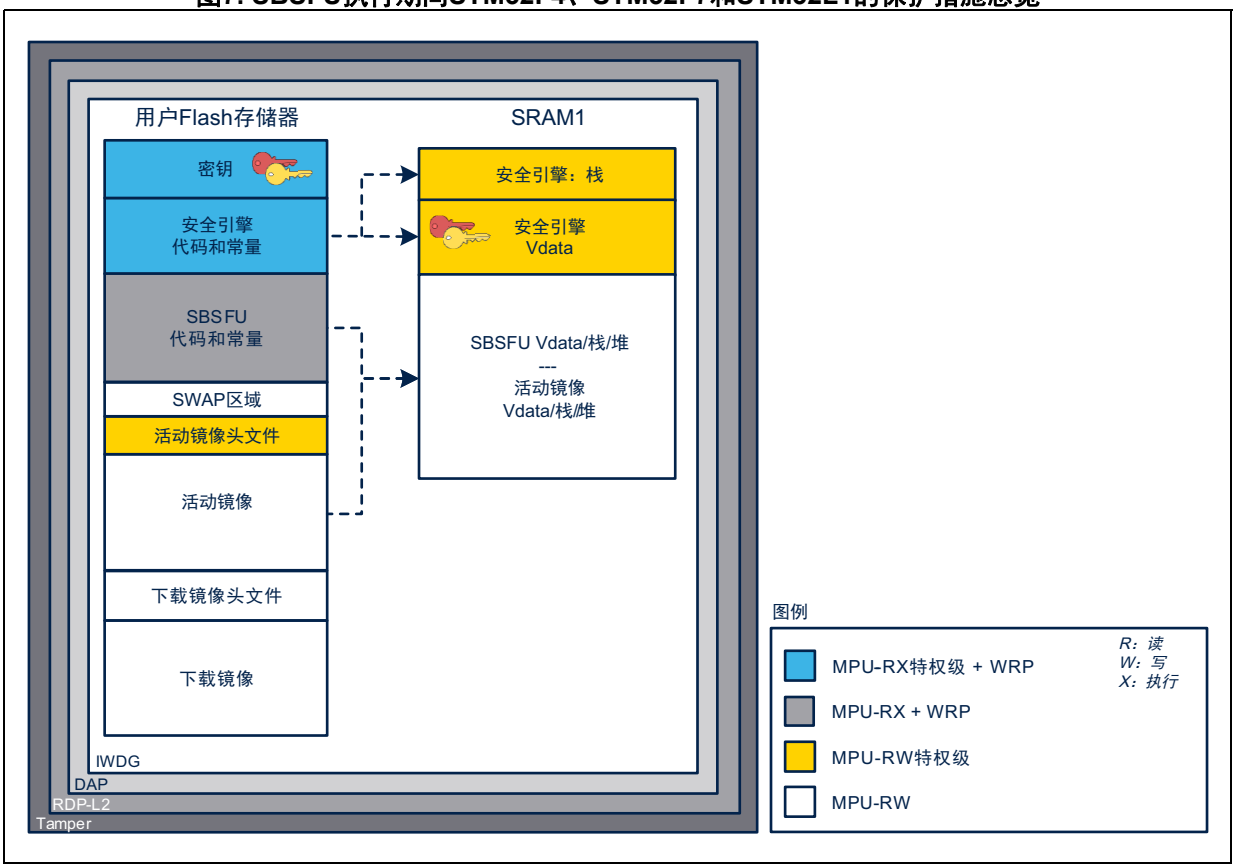
- **PCROP**[1]（专有代码读出保护）：通过PCROP保护，将Flash中的一个扇区定义为只执行区域后， 此区域不再被允许读取和写入访问。将密钥转化为一段代码并放置于只执行区域，密钥将受到PCROP的保护：执行此段代码将转移密钥到RAM中指定位置，同时将此置于防火墙内，以防从防火墙外部运行此段代码可能。
- **WRP**（写保护）：写保护用于保护可信代码免受外部攻击或甚至内部修改，如对关键代码/数据意外的写入/擦除操作。
- **MPU**（存储器保护单元）：通过将Flash和SRAM的存储器映射划分为具有访问权限的区域，MPU可使嵌入式系统更加稳健。在SBSFU应用实例中，配置MPU是为了确保在SBSFU代码执行期间，不执行任何其它代码。在离开SBSFU应用时，更新MPU配置以便授权用户应用代码的执行。

1. *STM32L0系列的读保护与写保护紧耦合：激活后，任何受读保护的扇区也受写保护。因此，不能激活读保护。*

## 5.2 STM32F4系列、STM32F7系列和STM32L1系列

*图 7*说明了如何保护系统、代码和数据的STM32F4系列、STM32F7系列和STM32L1系列的X-CUBE-SBSFU应用示例。

**图7. SBSFU执行期间STM32F4、STM32F7和STM32L1的保护措施总览**



### 可抵御外部攻击的保护措施

外部攻击是指由外部工具触发的攻击，如调试器或探测器尝试访问设备时。在SBSFU应用示例中，RDP、tamper、DAP和IWDG保护用来保护产品免受外部攻击：

- **RDP**（读保护）：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：
  - 禁止通过JTAG硬件接口对RAM和Flash进行外部访问。这样可以防止改变SBSFU代码并因此来挖掘可信根的攻击。
  - 选项字节不能更改。这意味着其他保护措施如WRP和PCROP不能再进行更改。

  **注意** - 由于以下原因，不建议使用RDP级别1：

  1. 无法确保安全启动/可信根（单入口点和不可变代码），因为在RDP L1情况下可以修改选项字节（WRP）。

  2. 设备内部闪存可以利用新的固件进行完全重新编程（在通过RDP L0回归进行闪存整体擦除之后），无需任何安全措施。

  3. 系统复位时，通过JTAG硬件接口连接调试器，可以访问受防火墙保护的RAM中的秘密信息。

  如果客户产品上不能进行JTAG硬件接口访问，并且客户使用信任可靠的用户应用程序代码，则上述风险无效。
- **Tamper**：防篡改保护用于检测设备上的物理篡改行为并采取相应的应对措施。在篡改检测的情况下，SBSFU应用示例强制重启。
- **DAP**（调试访问端口）：DAP保护在于可以停用DAP（调试访问端口）。一旦停用，JTAG引脚不再连接到内部总线。 使用RDP Level 2可自动禁用DAP。
- **IWDG**（独立看门狗）：IWDG是一个自由运行的减量计数器。一旦开始运行，它就不能再停止。在引起重置之前必须对其定期刷新。该机制能够控制SBSFU执行持续时间。

### 可抵御内部攻击的保护措施

内部攻击是指由STM32中运行的代码触发的攻击。攻击可能是由恶意固件利用错误或安全漏洞导致的，不需要的操作。在SBSFU应用示例中，WRP和MPU保护可防止产品受到内部攻击：

- **WRP**（写保护）：写保护用于保护可信代码免受外部攻击或甚至内部修改，如对关键代码或数据进行意外的写入或擦除操作。
- **MPU**（存储器保护单元）：利用MPU将管理所有关键数据和操作的受保护环境与其他软件组件隔离开来。安全引擎代码和数据只能通过特权级别的软件执行进行访问。因此，以非特权级别运行的软件不能调用安全引擎服务或访问关键数据。这种对安全引擎服务和资源的严格访问控制是通过定义特定的MPU区域来实现的，如*表 4*所示。

表4. STM32F4系列、STM32F7系列和STM32L1系列中的MPU区域

| 区域内容 | 特权权限 | 非特权权限 |
|---|---|---|
| 安全引擎代码和常量 | 只读<br>（允许执行） | 无访问权限 |
| 安全引擎堆栈和VDATA | 读/写<br>（不可执行） | 无访问权限 |

此外，MPU还确保在安全启动和安全固件更新进程运行时，只有获得授权的代码才被授予执行权限。正因为如此，才在启动用户应用之前更新MPU配置，以便授权其执行。不过，在运行用户应用时（不仅仅是在运行SBSFU代码时），安全引擎隔离设置和监控器调用机制仍然适用。

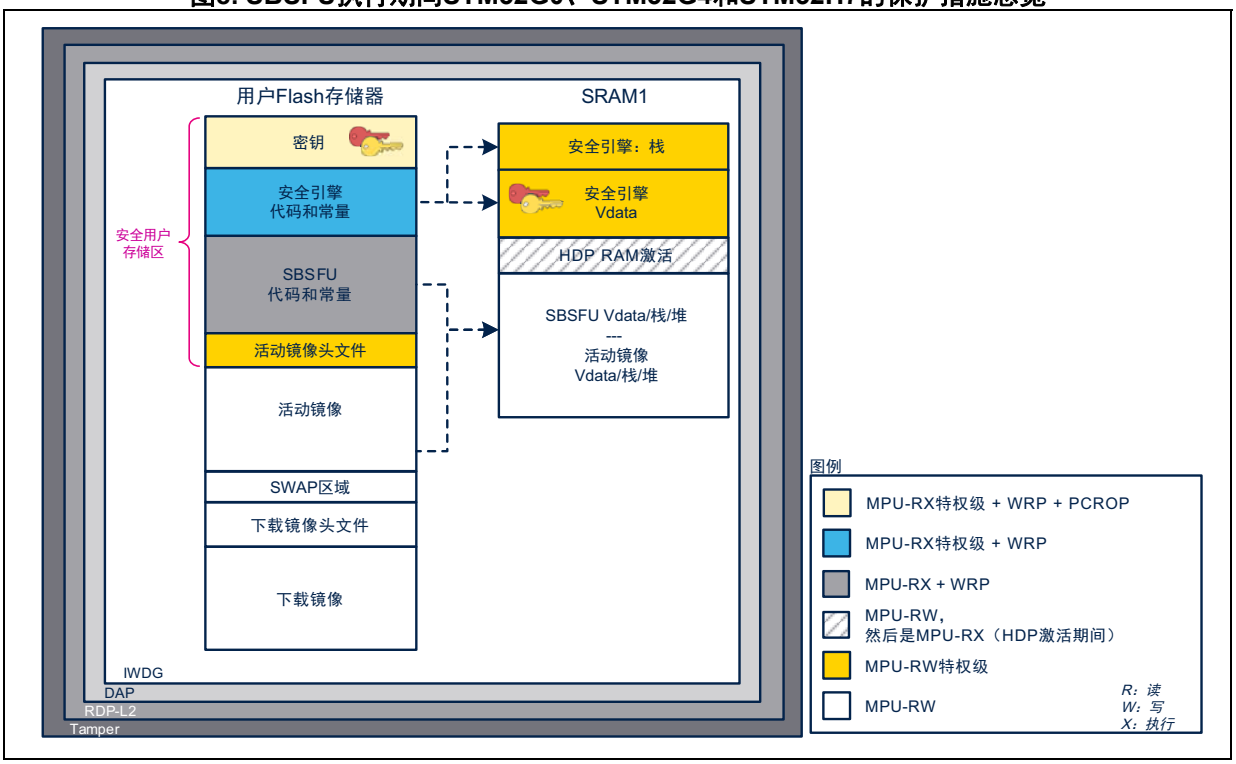
## 5.3 STM32G0系列、STM32G4系列和STM32H7系列

*图 8*说明了如何保护系统、代码和数据的STM32G0系列、STM32G4系列和STM32H7系列的X-CUBE-SBSFU应用示例。

关于STM32H7B3器件的特殊性，请参见附录*I.2: STM32H7B3器件上的外部Flash*。

关于STM32H750器件的特殊性，请参见附录*I.3: STM32H750B器件特殊性*。

图8. SBSFU执行期间STM32G0、STM32G4和STM32H7的保护措施总览



### 可抵御外部攻击的保护措施

外部攻击是指由外部工具触发的攻击，如调试器或探测器尝试访问设备时。在SBSFU应用示例中，RDP、tamper、DAP和IWDG保护用来保护产品免受外部攻击：

**RDP（读保护）：**

- 读保护级别2可实现最高级别的保护并实现可信根。
  - 禁止通过JTAG W接口对RAM和Flash进行外部访问。这样可以防止改变SBSFU代码并因此来挖掘可信根的攻击。
  - 选项字节不能更改。这意味着其他保护措施如WRP和PCROP不能再进行更改。
- 由于以下原因，读保护级别1实现的保护级别低于RDP 2级：
  - 在选项字节（WRP）重置后，可修改Flash中存储的代码/数据（去掉了不可变性）（在RDP 2级时不可能）。
  - 处于RDP 1级时，设备内部Flash可以利用新的固件进行完全重新编程（在通过RDP L0回归进行Flash整体擦除之后），无需任何安全措施。
  - 系统复位时，通过JTAG硬件接口连接调试器，可以访问RAM中的机密信息(1)。
- 读保护级别0不支持任何保护，可以访问整个产品。

**安全启动/可信根：**在复位后，使用*BOOT_Lock*配置强制首先运行这部分客户代码。使用可安全存储区保护将其与运行时间固件的其余部分隔离开来。

**篡改**：防篡改保护用于检测设备上的物理篡改行为并采取相应的应对措施。在篡改检测的情况下，SBSFU应用示例强制重启。

**DAP（调试访问端口）：**DAP保护在于可以停用DAP（调试访问端口）。禁用后，JTAG引脚不再连接到STM32内部总线。RDP 2级自动禁用DAP。

**IWDG（独立看门狗）：**IWDG是一个自由运行的减量计数器。一旦开始运行，它就不能再停止。在引起重置之前必须对其定期刷新。该机制能够控制SBSFU执行持续时间。

1. 在STM32H7系列上不可能实现。更多详细信息，请参见*附录I*。

**可抵御内部攻击的保护措施：**内部攻击是指由STM32中运行的代码触发的攻击。攻击可能 是由恶意固件利用错误或安全漏洞导致的，也可能由不需要的操作引起。

在SBSFU应用示例中，PCROP、WRP和MPU保护可防止产品受到内部攻击：

- **PCROP**（专有代码读出保护）：通过PCROP保护，将Flash的一个扇区定义为只执行区。在读取或写入时不能访问此扇区。作为一个只执行区域，只有当它“嵌入”一段代码时，密钥才受到PCROP保护：执行此代码会将该密钥移动到RAM中的特定位置。由于它在防火墙之后，因此不可能从外部执行。
- **WRP**（写保护）：写保护用于保护可信代码免受外部攻击或甚至内部修改，如对关键代码/数据进行不需要的写入/擦除操作。
- **MPU**（存储器保护单元）：利用MPU将管理所有关键数据和操作的受保护环境与其他软件组件隔离开来。安全引擎代码和数据只能通过特权级别的软件执行进行访问。因此，以非特权级别运行的软件不能调用安全引擎服务或访问关键数据。这种对安全引擎服务和资源的严格访问控制通过定义特定的MPU区域实现，如*表 5*所示。

**表5. STM32G0系列、STM32G4系列和STM32H7系列中的MPU区域**

| 区域内容 | 特权权限 | 非特权权限 |
|---|---|---|
| 安全引擎代码和常量 | 只读<br>（允许执行） | 无访问权限 |
| 安全引擎堆栈和VDATA | 读/写<br>（不可执行） | 无访问权限 |

此外，MPU还确保在安全启动和安全固件更新进程运行时，只有获得授权的代码才被授予执行权限。

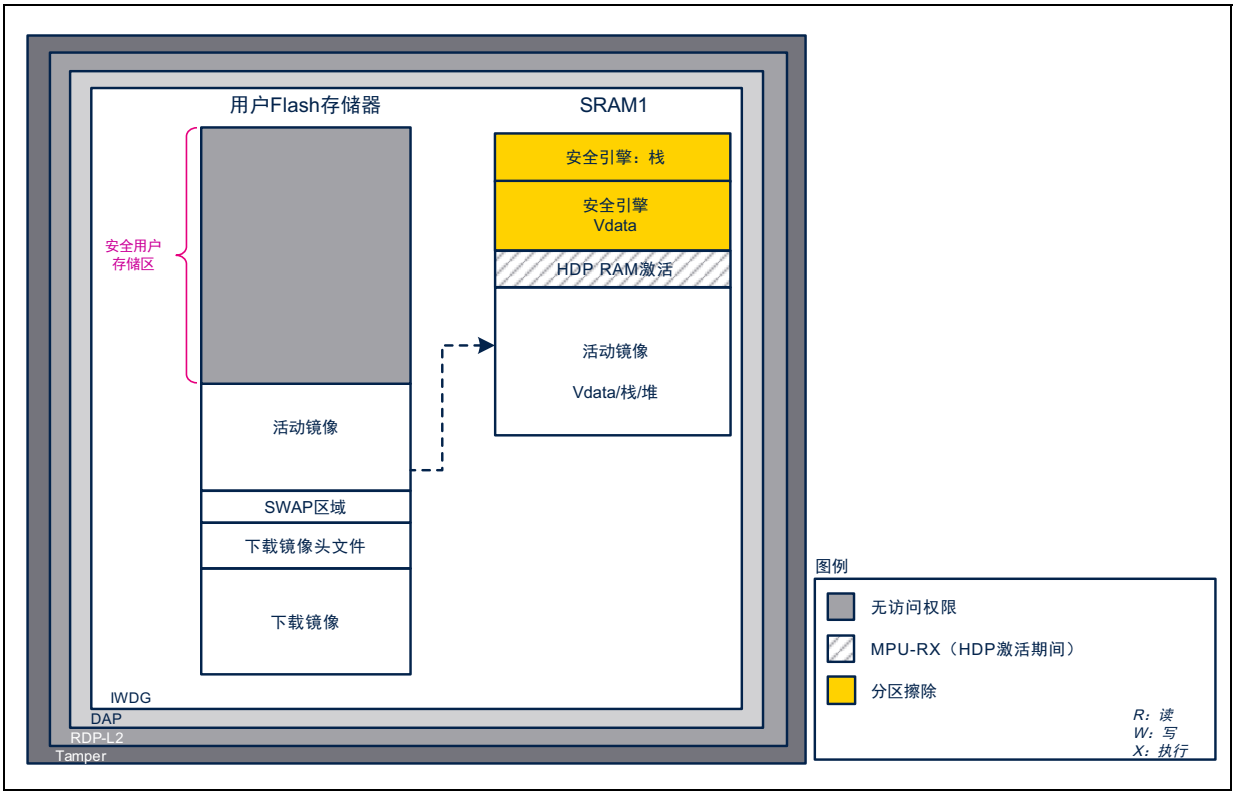
在启动用户应用之前，MPU保护被禁用但安全用户存储区保护被激活。

- **安全用户存储区**：在安全用户存储区保护被激活后，对可安全存储区的任何访问（提取、读取、编程和擦除）都会被拒绝，生成总线错误。安全用户存储区（受保护环境）中的所有代码和秘密信息均完全隐藏。由于不受安全用户存储区保护，在启动用户应用时，安全引擎堆栈和数据会被清除。

为激活安全用户存储区而执行的代码必须位于受保护存储区之外。在SBSFU中，这些代码位于RAM中。

*图 9*描述了在启动用户应用时安全用户存储区的关闭。

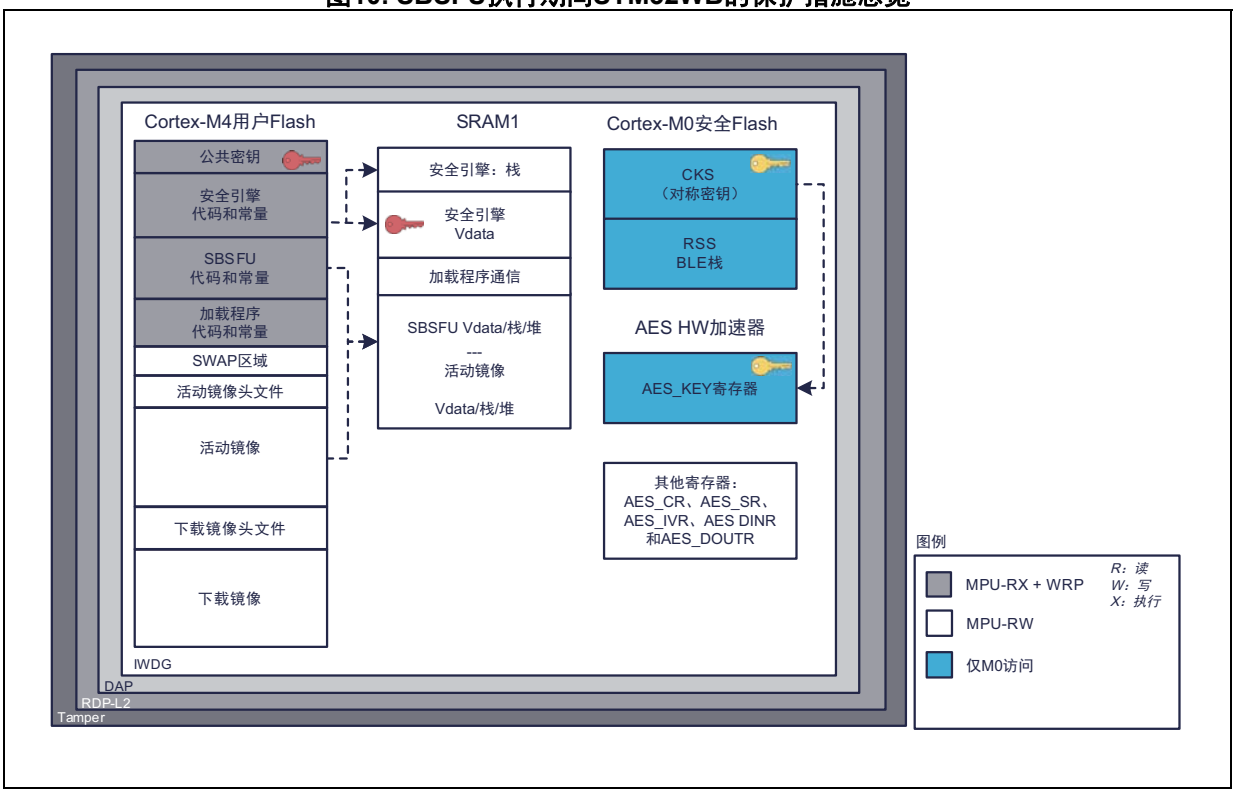
**图9. 用户应用执行期间STM32G0、STM32G4和STM32H7的保护措施总览**



## 5.4 STM32WB系列

*图 10*说明了如何保护系统、代码和数据的STM32WB系列的X-CUBE-SBSFU应用示例。

**图10. SBSFU执行期间STM32WB的保护措施总览**



### 可抵御外部攻击的保护措施

外部攻击是指由外部工具触发的攻击，如调试器或探测器尝试访问设备时。在SBSFU应用示例中，RDP、tamper、DAP和IWDG保护用来保护产品免受外部攻击：

- **RDP**（读保护）：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：
  - 禁止通过JTAG硬件接口对RAM和Flash进行外部访问。这样可以防止改变SBSFU代码并因此来挖掘可信根的攻击。
  - 选项字节不能更改。这意味着其他保护措施如WRP和PCROP不能再进行更改。

  **注意** - 由于以下原因，不建议使用RDP级别1：

  1. 无法确保安全启动/可信根（单入口点和不可变代码），因为在RDP L1情况下可以修改选项字节（WRP）。

  2. 设备内部闪存可以利用新的固件进行完全重新编程（在通过RDP L0回归进行闪存整体擦除之后），无需任何安全措施。

  3. 系统复位时，通过JTAG硬件接口连接调试器，可以访问受防火墙保护的RAM中的秘密信息。

  如果客户产品上不能进行JTAG硬件接口访问，并且客户使用信任可靠的用户应用程序代码，则上述风险无效。
- **Tamper**：防篡改保护用于检测设备上的物理篡改行为并采取相应的应对措施。在篡改检测的情况下，SBSFU应用示例强制重启。
- **DAP**（调试访问端口）：DAP保护在于可以停用DAP（调试访问端口）。一旦停用，JTAG引脚不再连接到内部总线。 使用RDP Level 2可自动禁用DAP。
- **IWDG**（独立看门狗）：IWDG是一个自由运行的减量计数器。一旦开始运行，它就不能再停止。在引起重置之前必须对其定期刷新。该机制能够控制SBSFU执行持续时间。

### 可抵御内部攻击的保护措施

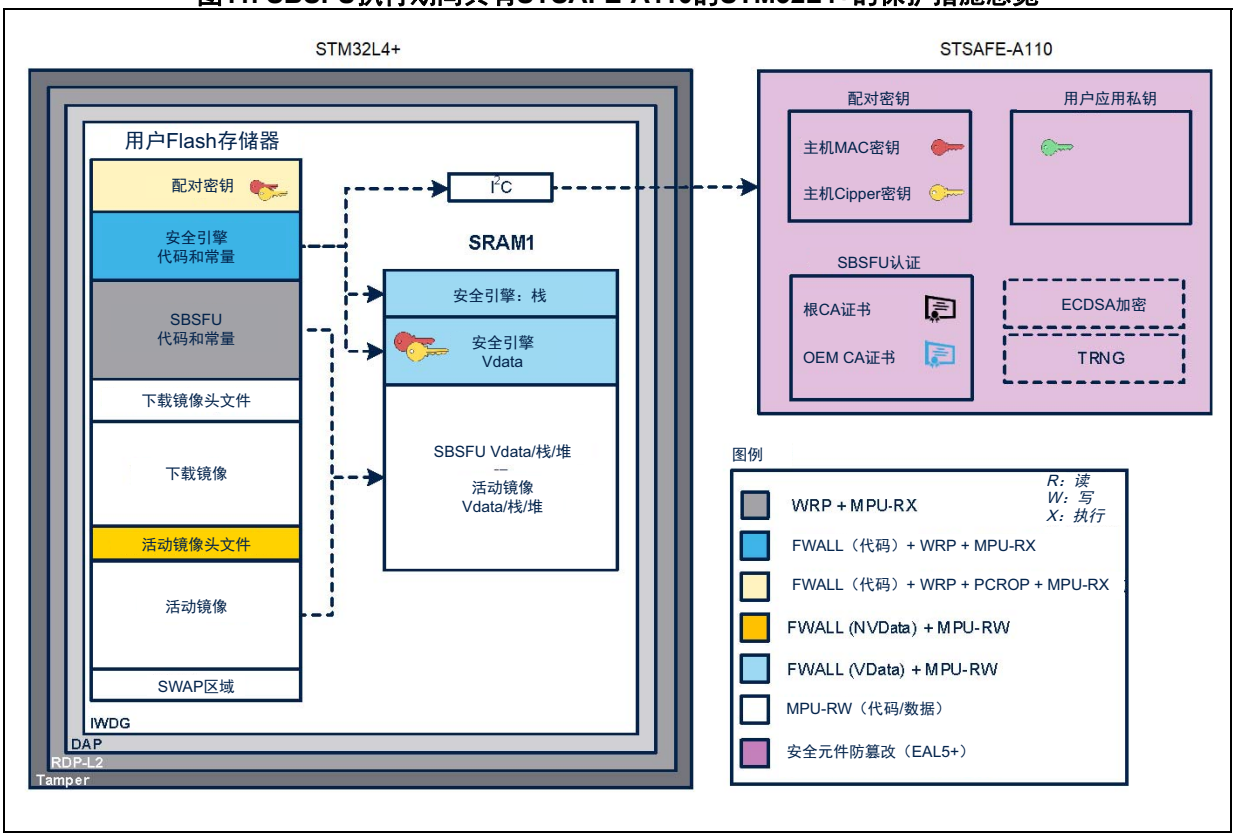
内部攻击是指由STM32中运行的代码触发的攻击。攻击可能是由恶意固件利用错误或安全漏洞导致的，不需要的操作引起。在SBSFU应用示例中，CKS、WRP和MPU保护可防止产品受到内部攻击：

- **CKS**（客户密钥存储）：SBSFU对称密钥位于Cortex®-M0+核心安全Flash中，因此，不能从Cortex®-M4核心访问。在每次AES密码解密/加密前，Cortex®-M4核心请求Cortex®-M0+代码将密钥加载到AES硬件加速器密钥寄存器中（只能从Cortex®-M0+核心访问）。
- **WRP**（写保护）：写保护用于保护可信代码免受外部攻击或甚至内部修改，如对关键代码/数据意外的写入/擦除操作。此外，WRP可以保护SBSFU公钥。
- **MPU**（存储器保护单元）：通过将Flash和SRAM的存储器映射划分为具有访问权限的区域，MPU可使嵌入式系统更加稳健。在SBSFU应用实例中，配置MPU是为了确保在SBSFU代码执行期间，不从任何存储器执行其他代码。在离开SBSFU应用时，更新MPU配置以便授权用户应用代码的执行。

## 5.5 结合了STSAFE-A110的STM32L4+系列

*图 11*说明了如何保护系统、代码和数据的X-CUBE-SBSFU结合了STSAFE-A110的STM32L4+系列的应用示例。

**图11. SBSFU执行期间具有STSAFE-A110的STM32L4+的保护措施总览**



### 可抵御外部攻击的STM32微控制器保护措施

外部攻击是指由外部工具触发的攻击，如调试器或探测器尝试访问设备时。在SBSFU应用示例中，RDP、tamper、DAP和IWDG保护用来保护产品免受外部攻击：

- **RDP**（读保护）：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：
  - 禁止通过JTAG硬件接口对RAM和Flash进行外部访问。这样可以防止改变SBSFU代码并因此来挖掘可信根的攻击。
  - 选项字节不能更改。这意味着其他保护措施如WRP和PCROP不能再进行更改。

  **注意** - 由于以下原因，不建议使用RDP级别1：

  1. 无法确保安全启动/可信根（单入口点和不可变代码），因为在RDP L1情况下可以修改选项字节（WRP）。

  2. 设备内部闪存可以利用新的固件进行完全重新编程（在通过RDP L0回归进行闪存整体擦除之后），无需任何安全措施。

  3. 系统复位时，通过JTAG硬件接口连接调试器，可以访问受防火墙保护的RAM中的秘密信息。

  如果客户产品上不能进行JTAG硬件接口访问，并且客户使用信任可靠的用户应用程序代码，则上述风险无效。
- **Tamper**：防篡改保护用于检测设备上的物理篡改行为并采取相应的应对措施。在篡改检测的情况下，SBSFU应用示例强制重启。
- **DAP**（调试访问端口）：DAP保护在于可以停用DAP（调试访问端口）。一旦停用，JTAG引脚不再连接到内部总线。 使用RDP Level 2可自动禁用DAP。
- **IWDG**（独立看门狗）：IWDG是一个自由运行的减量计数器。一旦开始运行，它就不能再停止。在引起重置之前必须对其定期刷新。该机制能够控制SBSFU执行持续时间。

### 可抵御内部攻击的STM32微控制器保护措施

内部攻击是指由STM32中运行的代码触发的攻击。攻击可能是由恶意固件利用错误或安全漏洞导致的，不需要的操作。在SBSFU应用示例中，WRP、防火墙、PCROP和MPU保护可防止产品受到内部攻击：

- **FWALL**（防火墙）：对防火墙进行配置以保护代码、易失性和非易失性数据。受保护的代码可以通过单入口点进行访问（调用门机制在*附录A*中有说明）。任何试图跳过并尝试不通过入口点而执行代码段中函数的操作都会导致系统重置。

- **PCROP**（专有代码读出保护）：通过PCROP保护，将Flash的一个扇区定义为只执行区。在读取或写入时不能访问此扇区。作为一个只执行区域，只有当它“嵌入”一段代码时，密钥才受到PCROP保护：执行此代码会将该密钥移动到RAM中的特定指针。由于它在防火墙之后，因此不可能从外部执行。
- **WRP**（写保护）：写保护用于保护可信代码免受外部攻击或甚至内部修改，如对关键代码/数据意外的写入/擦除操作。
- **MPU**（存储器保护单元）：通过将Flash和SRAM的存储器映射划分为具有访问权限的区域，MPU可使嵌入式系统更加稳健。在SBSFU应用实例中，配置MPU是为了确保在SBSFU代码执行期间，不执行任何其他代码。在离开SBSFU应用时，更新MPU配置以便授权用户应用代码的执行。

### STSAFE-A安全元件保护

STSAFE-A110是一种高度安全的解决方案，其安全操作系统运行于最新一代安全微控制器上：

- **安全特性**：芯片经CC EAL5+ AVA_VAN5 Common Criteria认证，提供下列保护措施。
  - 主动屏蔽
  - 监控环境参数
  - 故障保护机制
  - 每块晶片上标有唯一序列号
  - 侧信道攻击保护
- **安全操作系统**：STSAFE-A110运行安全操作系统，提供针对逻辑和物理攻击的保护。
- **安全信道和设备绑定**：STSAFE-A110支持用STM32设置安全信道，以防I²C线路上的敏感信息被窃听，并确保特定STM32与特定STSAFE-A110的配对（防止克隆）。

安全信道基于对称加密：用两个AES 128位密钥（即主机配对密钥）实现诸如指令授权、指令数据加密、响应数据加密和响应验证等服务。

# 6 软件包说明

本节详细介绍了X-CUBE-SBSFU包的内容和使用方法。

## 6.1 概述

X-CUBE-SBSFU是用于STM32微控制器的软件包。

它提供了构建安全启动和安全固件更新应用的完整解决方案：

- 可支持对称和非对称加密方法，利用用于解密或验证的AES-GCM、AES-CBC和ECDSA算法，或用来同时解密和验证的X-CUBE-CRYPTOLIB
- 固件映像和固件更新的X509证书链验证支持[a]
- 两种操作模式：
  - 双插槽配置包含一个活动插槽和一个下载插槽，能够进行安全的映像编程，在安装过程中断后可继续安装
  - 单插槽配置包含一个活动插槽，可使用户应用大小最大化
- 将安全外设和机制整合集成，以实现SBSFU可信根。RDP、WRP、PCROP、防火墙、MPU、安全用户内存、tamper和IWDG结合使用，实现最高的安全级别[b]。
- 使用安全引擎（SE）模块作为中间件的一部分，以提供管理所有关键数据和操作（如安全密钥存储，加密操作等）的受保护环境
- 集成了安全密钥管理服务（KMS），可通过PKCS #11 API提供对称和非对称密码服务，并提供安全密钥存储和更新服务
- 集成了STSAFE-A110安全元件，以便为系统提供防篡改可信根（CC EAL5+ AVA_VAN5 Common Criteria认证），并减轻ECDSA加密操作的主机MCU负载。关于STSAFE-A110的更多信息，请访问www.st.com/stsafe-a110。
- 可使用用户应用程序示例源代码。
- 对于有多种固件（如协议栈、中间件和用户应用）的复杂系统，固件映像配置可扩展至最多三个映像
- 当新映像通过“自检”验证成功时，用户应用可验证新的活动映像的安装。
- 现在，对于中断处理要求低延迟的应用，支持防火墙以内代码执行期间的中断管理。
- 可使用固件映像准备工具，它以可执行代码和源代码的方式提供。

a. 特定于STSAFE-A110。

b. 安全IP的可用性取决于STM32系列。

X-CUBE-SBSFU被移植到STM32F4系列、STM32F7系列、STM32G0系列、STM32G4系列、STM32H7系列、STM32L0系列、STM32L1系列、STM32L4系列、STM32L4+系列和STM32WB系列上。X-CUBE-SBSFU还被移植到结合了STSAFE-A110（安装在B-L4S5I-IOT01A板上）的STM32L4+系列上。

该软件包包含了应用示例，开发人员可以将其用于试验代码。

该软件包以zip文档的形式提供，其中包含源代码。

可支持以下集成开发环境：

- IAR Systems® - IAR Embedded Workbench®(a)
- Keil® - MDK-ARM(a)
- 意法半导体 - STM32CubeIDE

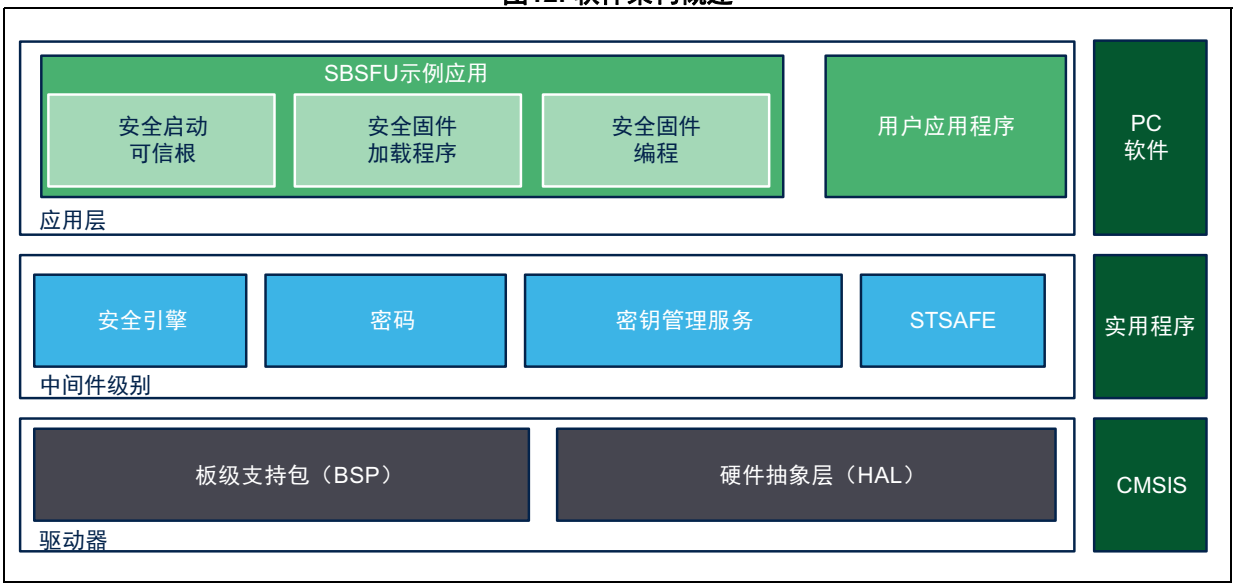
*注：* *STM32L4系列和STM32L4+系列提供KMS功能，所提供示例基于B-L475E-IOT01A和B-L4S5I-IOT01A板。*
*STM32L4+系列提供STSAFE-A110功能，所提供示例基于B-L4S5I-IOT01A板。*

a. 仅限于Windows® PC。

## 6.2 架构

本节描述X-CUBE-SBSFU包的软件组成部分，如*图 12*中所示。

**图12. 软件架构概述**




### 6.2.1 STM32CubeHAL

HAL驱动层提供通用的多实例简单API组（应用程序编程接口），以便与上层（应用、库和协议栈）交互。它由通用和扩展API构成。它直接围绕通用架构构建，允许在其基础上构建层，例如中间件层，实现了它的功能又无需依赖给定微控制器单元（MCU）的特定硬件配置。

此结构可提高库代码的可复用性，并确保可向其他设备轻松移植。

### 6.2.2 板级支持包（BSP）

除MCU之外，软件包需支持STM32板上的外设。板级支持包（BSP）中包含此软件。这是一个有限的API集，为板特有的某些外设（例如LED和用户按钮等）提供编程接口。

### 6.2.3 加密库

支持三种不同的密码中间件：

- X-CUBE-CRYPTOLIB可支持对称和非对称密钥方法（AES-GCM，AES-CBC，ECDSA）以及用于解密和验证的哈希运算（SHA256）。使用软件加密函数，以避免将密钥存储在不受保护的硬件加密IP寄存器中。
- mbedTLS：以开源代码的形式提供的密码服务。
  与X-CUBE-CRYPTOLIB类似，支持对称和非对称密钥方法（AES-GCM、AES-CBC和ECDSA）以及用于解密和验证的哈希运算（SHA256）。提供了32L496GDISCOVERY、B-L475E-IOT01A, STM32F413HDISCOVERY, STM32F769IDISCOVERY, P-NUCLEO-WB55和NUCLEO-H753ZI板件的示例（文件夹*2_images_OSC*中）。
- mbed-crypto：以开源代码的形式提供的密码服务。此中间件提供PSA密码API实现。为B-L4S5I-IOT01A板提供的示例位于*2_Images_KMS*和*2_Images_STSAFE*文件夹中。

### 6.2.4 安全引擎（SE）中间件

安全引擎中间件提供了用于管理所有关键数据和操作（例如访问固件密钥的操作，等等）的受保护环境。受保护的代码和数据可通过一个单入口点（调用门机制）进行访问，因此不能在不经过单入口点的情况下运行或访问任何SE代码或数据，否则会产生系统重置（请参阅*附录A*获取有关调用门机制的详细信息）。

*注：* *根据用户应用程序的需要，安全引擎关键操作可以扩展其他功能。只有可信代码才能被添加到安全引擎环境中，因为它有访问秘密信息的权限。*

### 6.2.5 密钥管理服务（KMS）中间件

安全密钥管理服务通过PKCS #11 API（基于KEY ID的API）向用户应用提供密码服务，这些API在安全隔区中执行。用户应用密钥存储在安全隔区中，可进行安全更新（更新前进行真实性检查、解密和完整性检查）。

### 6.2.6 STSAFE-A中间件

STSAFE-A中间件提供了一整套API，用于从STM32微控制器访问所有STSAFE-A110器件功能。

它集成了底层通信驱动和更高层的处理，前者用于连接STSAFE-A110硬件，后者用于导出一组指令API，以便从STM32微控制器访问器件功能。

### 6.2.7 安全启动和安全固件升级（SBSFU）应用程序

#### 安全启动（可信根）

- 检查并应用STM32平台的安全机制以保护关键操作和机密免受攻击
- 每次执行前认证并核验用户应用程序

#### 通过UART虚拟COM进行本地下载

- 检测固件下载请求
- 通过使用Ymodem协议的UART虚拟COM和Tera Term工具，在STM32Flash中下载新的加密固件映像（头文件和加密固件）（参见*注释*）

#### 固件安装管理

- 检测要安装的新固件版本
  - 通过UART接口从本地下载服务
  - 通过用户应用下载（仅适用于双插槽变体）
- 安全固件升级：
  - 身份验证和完整性检查
  - 固件解密
  - 固件安装
  - 防回滚机制，避免重新安装之前的固件版本
- 支持多个映像：
  - 对于有多种固件（如协议栈、中间件和用户应用）的复杂系统，支持最多三个活动插槽和三个下载插槽
  - 每个插槽特定的加密密钥
  - 同步安装映像，以确保固件之间彼此兼容
- 支持单插槽配置，使用户应用大小最大化
- 支持双插槽配置，实现安全的映像编程
  - 带固件镜像验证的安装过程。如果新的固件映像未通过用户应用的验证，则在下一次复位时将触发向上一固件映像的回滚。此选项在ENABLE_IMAGE_STATE_HANDLING编译开关下提供。请参考*附录J*获取关于固件映像验证的详细信息。
  - 继续安装固件：如果安装过程中断电，将在下一次上电时继续安装。
  - 有SWAP区的安装，可限制需要的存储器开销。然后，可实现回滚选项。（请参考*附录B*获取关于多插槽管理的详细信息）。
  - 无SWAP区的安装，下载区可保持加密状态。当下载插槽位于无OTFDEC IP的外部Flash中时这是必需的（就B-L475E-IOT01A和STM32WB5MM-DK板的2_Images_ExtFlash而言）。
  - 部分更新：可灵活地选择更新整个固件映像或其一部分。

*注 STM32H7B3I-DK和STM32H750B-DK板的2_Images_Ext变体中提供了独立加载程序的示例。有关详细信息，请参见附录I。*

*对于STM32WB系列，用BLE_Ota加载程序替代独立的YMODEM加载程序。有关详细信息，请参见附录H。*

### 6.2.8 用户应用程序

- 提供了一个使用Ymodem协议通过UART下载用户应用的示例。可在用户应用中实现无线下载机制，如BLE、Wi-Fi®或其他。例如，YMODEM加载程序被STM32WB系列（具有P-NUCLEO-WB55 Nucleo板或STM32WB5MM-DK探索板）的BLE_Ota加载程序取代。
- 提供了测试保护机制的示例。
- 提供了使用由SE导出的某些功能的示例，例如获取有关当前固件映像的信息。
- 提供了使用以下KMS导出服务（通过标准PKCS #11接口）的示例：AES-GCM/CBC加密/解密，RSA签名/验证，密钥配置，AES ECB密钥推导，ECDSA密钥对生成，以及ECDH Diffie-Hellman密钥推导。
- 提供了演示以下STSAFE-A导出服务（通过标准PKCS #11接口）的示例：使用器件唯一私钥生成ECDSA签名，器件证书读取，签名验证，ECDSA密钥对生成，以及ECDH Diffie-Hellman密钥推导。这些服务通常用在TLS交换的背景下，是用于开发IoT节点（通过基于云的服务进行连接）的候选构建模块。

## 6.3 文件夹结构

*图 13*和*图 14*中显示了文件夹结构的顶层视图。

**图13. 项目文件夹结构（1/2）**

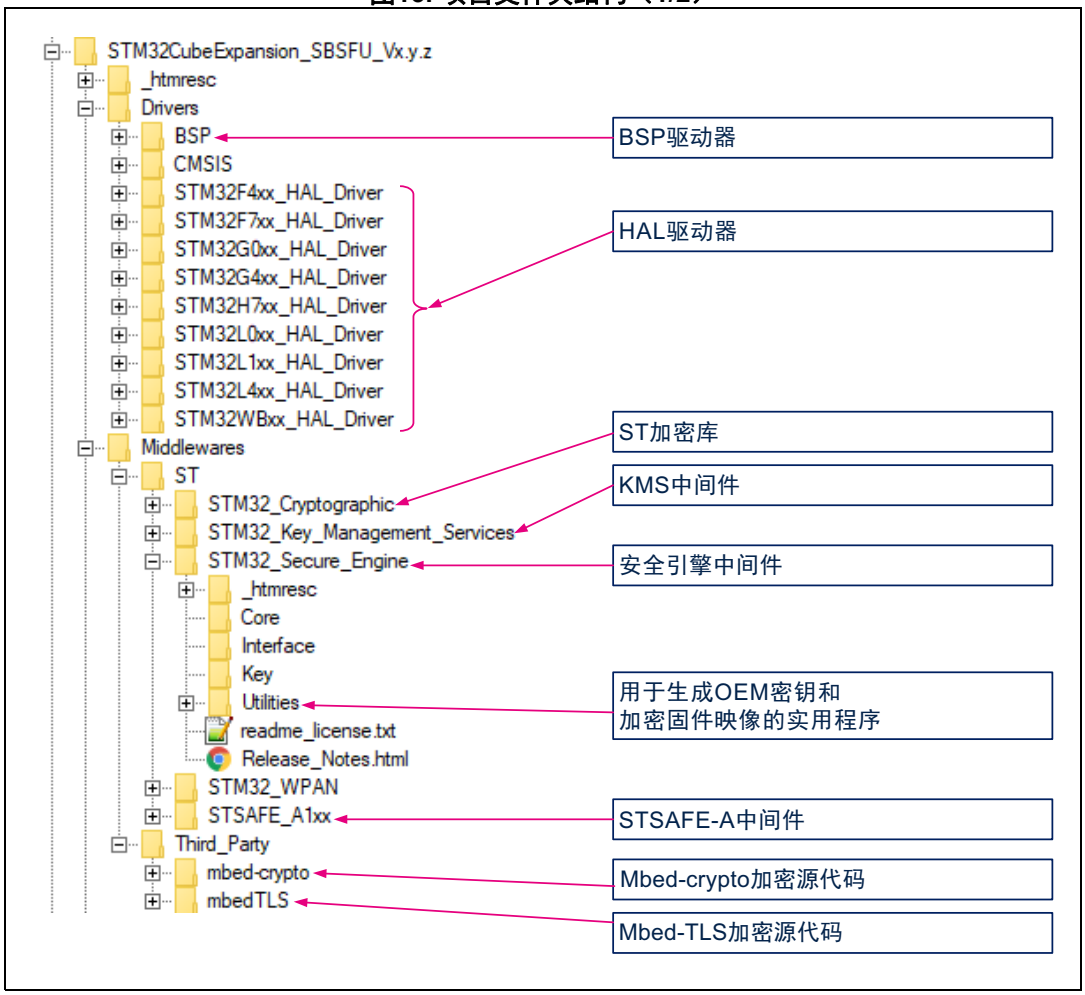
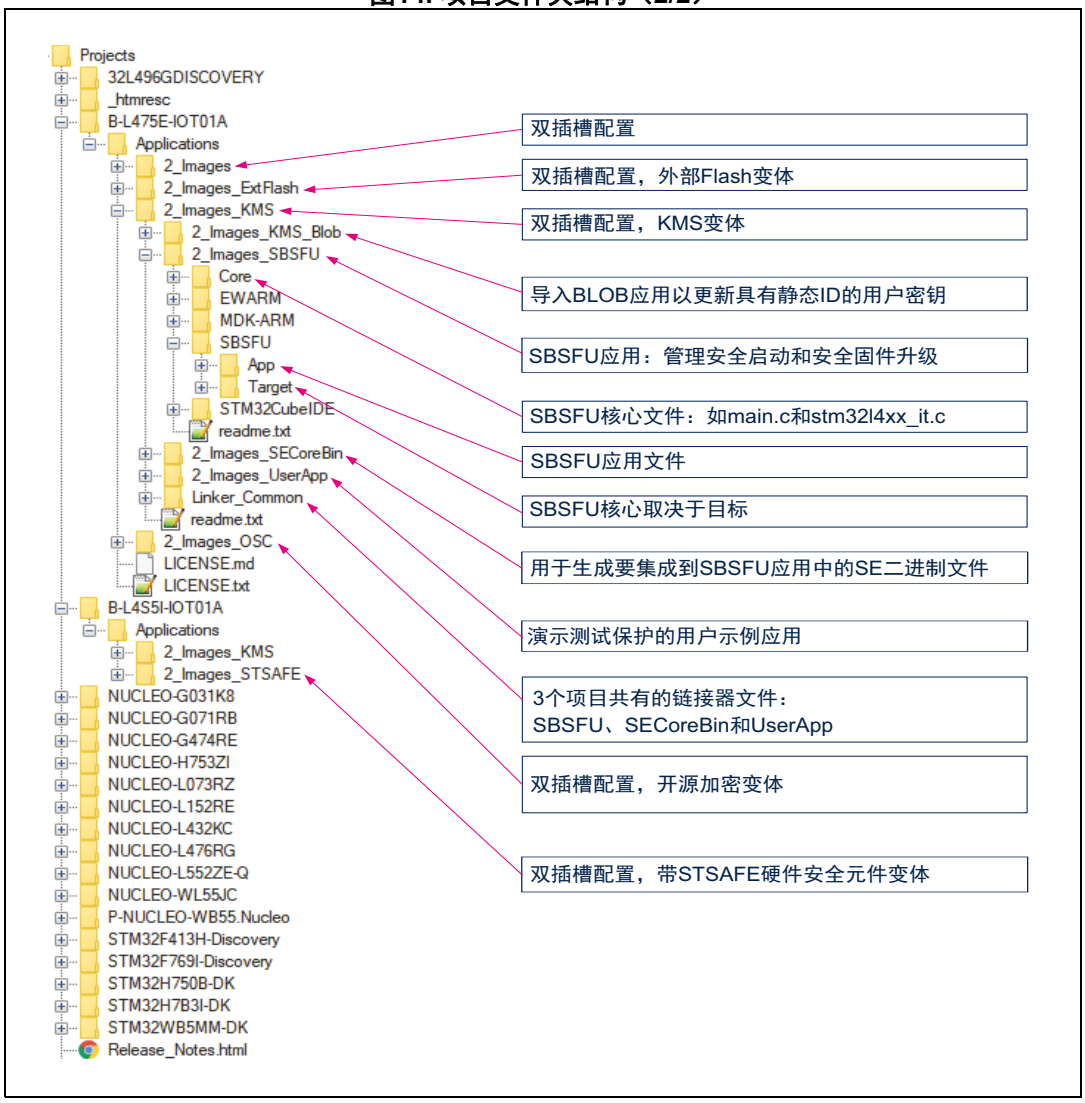


图14. 项目文件夹结构（2/2）




*注：* *在名为1_Image的应用中对单映像配置进行了演示。*

*在名为2_Images的示例中对双映像配置进行了演示。*

## 6.4 API

软件包的*STM32_Secure_Engine*、*STM32_Key_Management_Services*和*STSAFE_A1xx*文件夹中的已编译HTML文件中提供了关于API的详细技术信息，该软件包中介绍了所有函数和参数。

## 6.5 使用IAR Systems®工具链时的应用编译过程

*图 15*概述了构建应用程序以及演示安全启动和安全固件更新所需的步骤：

- 第1步：准备内核二进制文件

  创建安全引擎内核二进制文件需要执行此步骤，该代码中包括了映射到防火墙代码段的所有“可信”代码和密钥。SE callgate函数被指定为该二进制文件的入口点。步骤2中，该二进制文件链接到SBSFU代码。

- 第2步：SBSFU

  此步骤编译SBSFU源代码，实现状态机并配置保护。此外，它将该代码与步骤1中生成的SECore二进制文件链接起来，生成单独的包含SE可信代码的SBSFU二进制文件。它还生成一个包含符号的文件，以供用户应用程序调用SE接口方法，这是一组包装单个SE调用门API的用户友好API。

- 步骤3：用户应用程序示例

  它会生成：

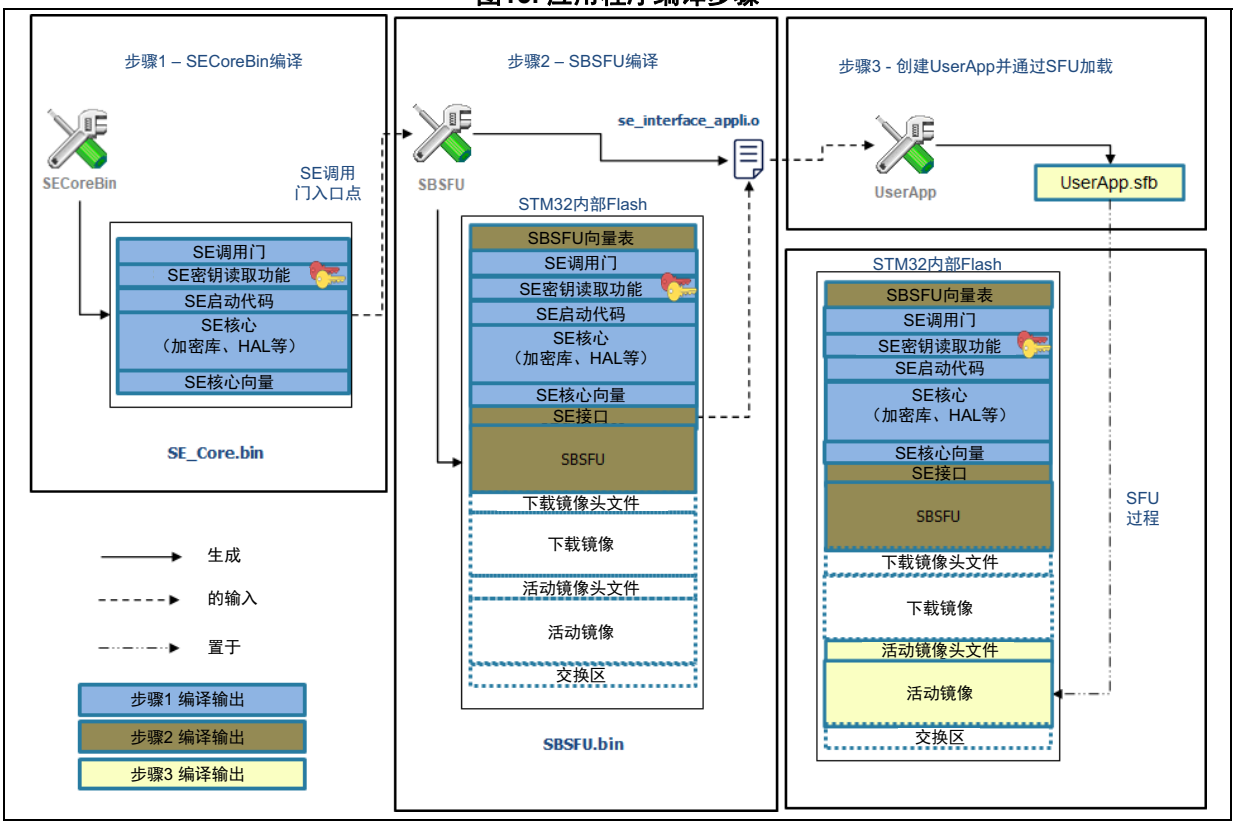
  – 用户应用程序二进制文件，使用安全固件更新程序（*UserApp.sfb*）可将其上传到设备。

  – 一个二进制文件，连接了SBSFU二进制文件、明文格式的用户应用程序二进制文件和对应固件头文件。

  当该二进制文件通过烧录工具烧录到设备中时，正确安置这三个元素，以便使SBSFU和用户应用程序运行。因此，SBSFU不需要固件安装程序来启动和引导用户应用程序。这是使用单个闪存级测试用户应用程序的便捷方式。

*注：* *请参考附录H 了解STM32WB系列的特殊性，并参考附录I 了解STM32H7B3I-DK和STM32H750B-DK探索板的2_Images_ExtFlash变体。*

**图15. 应用程序编译步骤**



# 7 硬件和软件环境设置

本节说明了硬件和软件设置过程。

## 7.1 硬件设置

要设置硬件，必须利用USB线缆将某种支持的板（如*第 6.1 节：概述*中所介绍）插入个人计算机。与PC的连接允许用户：

- 烧写板
- 通过UART控制台与板进行交互
- 在禁用保护时进行调试

## 7.2 软件设置

本节为开发人员列出了设置SDK、运行示例场景以及自定义应用的最低要求。

### 7.2.1 开发工具链和编译器

选择STM32Cube扩展包支持的其中一个集成开发环境。

考虑所选IDE供应商提供的系统要求和设置信息。

### 7.2.2 编程STM32微控制器的软件工具

#### ST-LINK Utility

STM32ST-LINK实用程序（STSW-LINK004）是用于编程STM32微控制器的全功能软件接口。它为读取、写入和验证存储设备提供了一个易用高效的环境。

参考*www.st.com*上的STSW-LINK004 STM32 ST-LINK实用软件。

**注意：** 确保使用了最新版本的ST-LINK。

#### STM32CubeProgrammer

STM32CubeProgrammer（STM32CubeProg）是一款用于编程STM32微控制器的全功能多操作系统软件工具。它通过调试接口（JTAG和SWD）和bootloader接口（UART和USB）提供了一个易用高效的环境，用于读取、写入和验证设备内存。

STM32CubeProgrammer提供了广泛的功能，可编程STM32微控制器内部存储器（如Flash、RAM和OTP）以及外部存储器。STM32CubeProgrammer还允许选择编程和上传、编程内容验证以及通过脚本自动编程微控制器。

STM32CubeProgrammer提供了GUI（图形用户界面）和CLI（命令行界面）版本。

参考*www.st.com*上的STM32CubeProgrammer软件（STM32CubeProg）

### 7.2.3 终端仿真器

运行演示需要终端仿真器软件。

本文档中的示例基于Tera Term，这是一款开源免费软件终端仿真器，可从https://osdn.net/projects/ttssh2/网页下载。可以使用任何其他类似的工具（需要支持Ymodem协议）。

### 7.2.4 X-CUBE-SBSFU固件映像准备工具

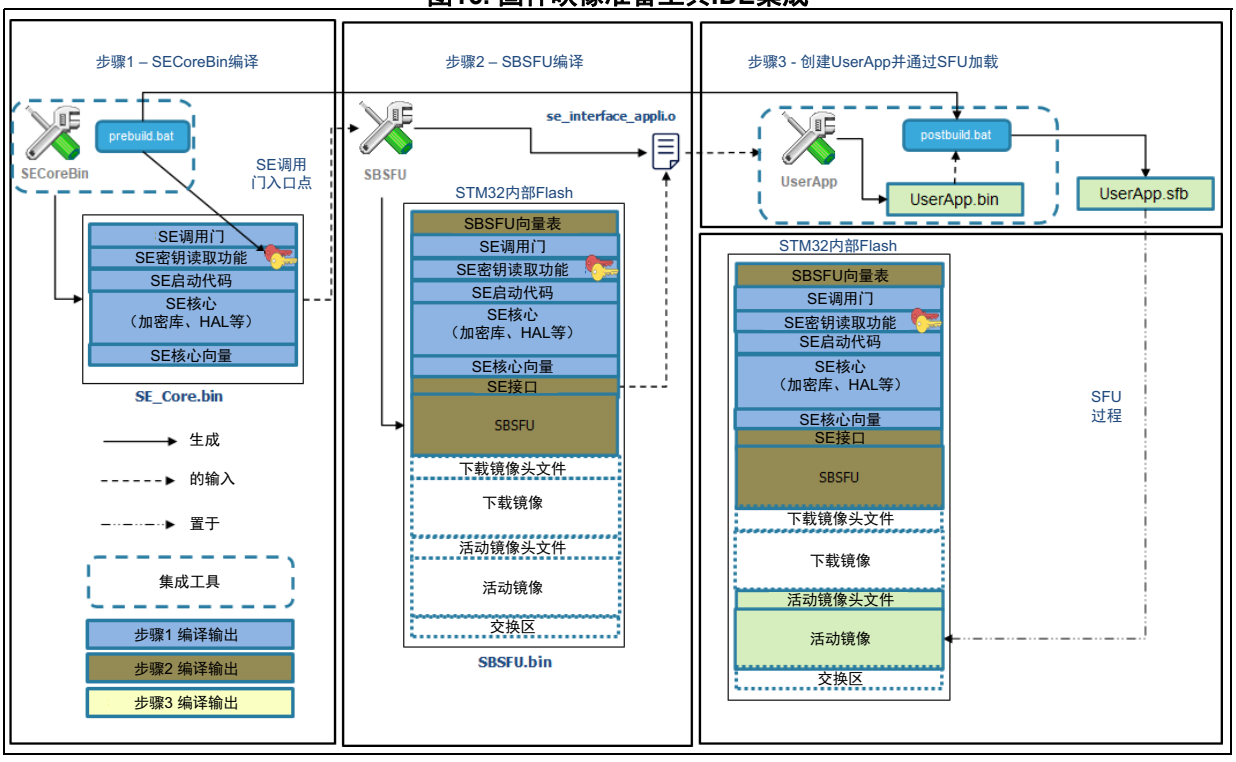
用于STM32Cube的X-CUBE-SBSFU扩展软件包随*prepareimage*工具一起提供，用于处理加密密钥和固件映像准备。

*prepareimage*工具以两种格式提供：

- Windows®可执行文件：需要标准Windows®命令解释器
- Python™脚本：Python™解释器以及*Middlewares\ST\STM32_Secure_Engine\Utilities\KeysAndImages\readme.txt*中列出的组件是必需的

Windows®可执行文件完全集成在支持的IDE和编译过程中，如*图 16*中所示。

**图16. 固件映像准备工具IDE集成**




有关准备工具的更多信息，请参阅*附录E: 固件映像准备工具*。

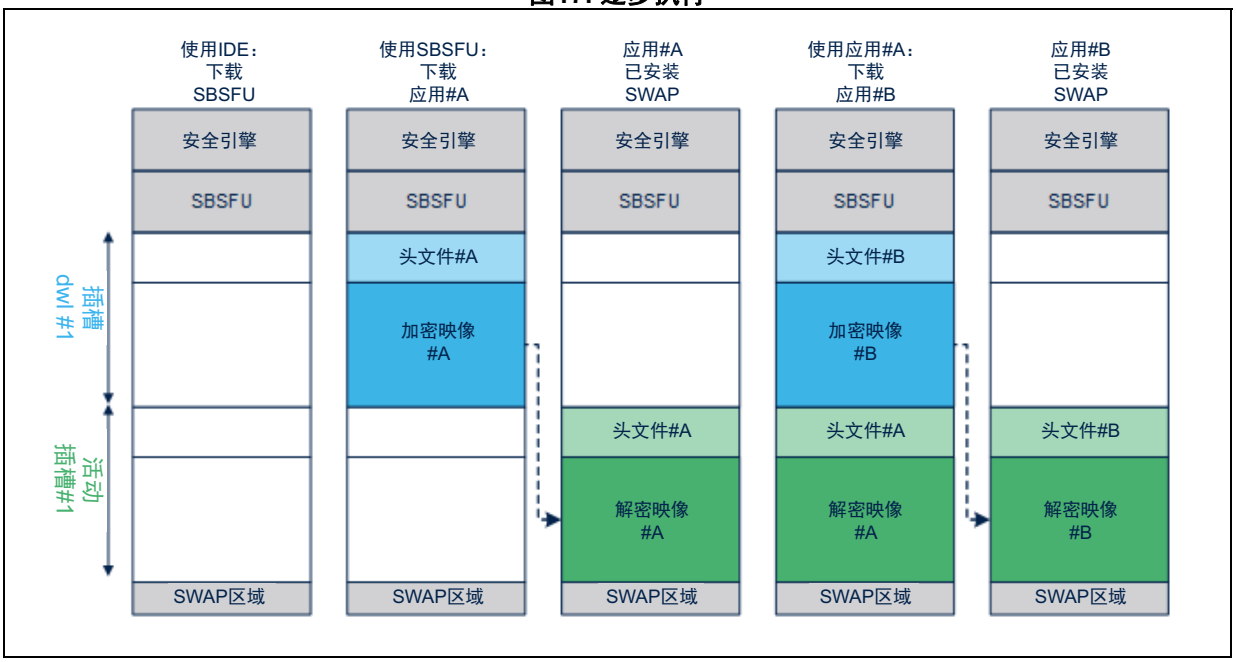
# 8 逐步执行

以下步骤描述了在NUCLEO-L476RG板上执行的采用默认加密方案的双映像SBSFU场景，进一步的说明如*图 17*所示：

1. 下载SBSFU应用程序
2. SBSFU正在运行：下载UserApp #A
3. UserApp #A已安装
4. UserApp #A正在运行，下载UserApp #B
5. UserApp #B安装并运行

UserApp#A和UserApp#B二进制文件基于用户应用程序示例项目而生成。将应用程序定义为#A或#B是通过更改应用程序*main.c*中声明的*UserAppId*变量的值来完成的。

**图17. 逐步执行**



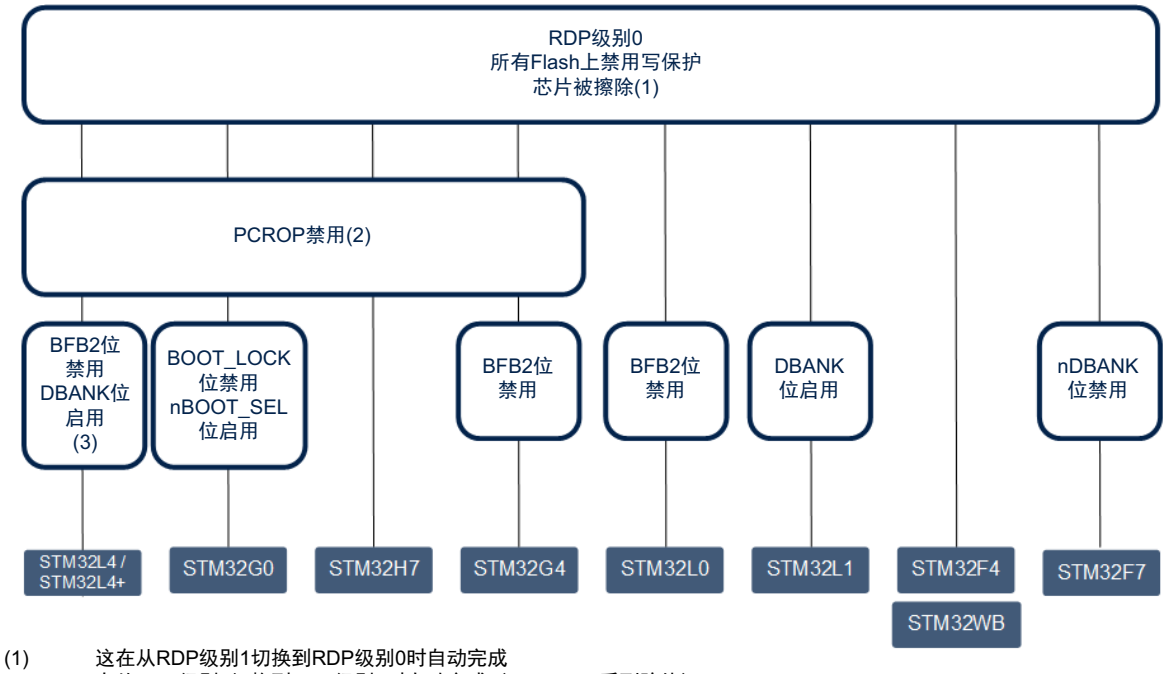
## 8.1 STM32板子准备

NUCLEO-L476RG板的目标选项字节设置如下：

- 设置RDP Level 0
- 所有闪存页面上禁用写保护
- 禁用PCROP保护(a)
- BFB2位禁用
- 芯片被擦除(b)

不同STM32系列的选项字节设置可能不同，如*图 18*所示。

**图18. STM32板子准备**




(1) 这在从RDP级别1切换到RDP级别0时自动完成
(2) 在从RDP级别1切换到RDP级别0时自动完成（STM32H7系列除外）
(3) 仅在STM32L4+系列中的STM32L4S5产品线上可用

a. 在从RDP级别1切换到RDP级别0时自动完成（STM32H7系列除外）。

b. 这在从RDP级别1切换到RDP级别0时自动完成。

使用STM32CubeProgrammer，通过以下四个步骤验证选项字节设置：

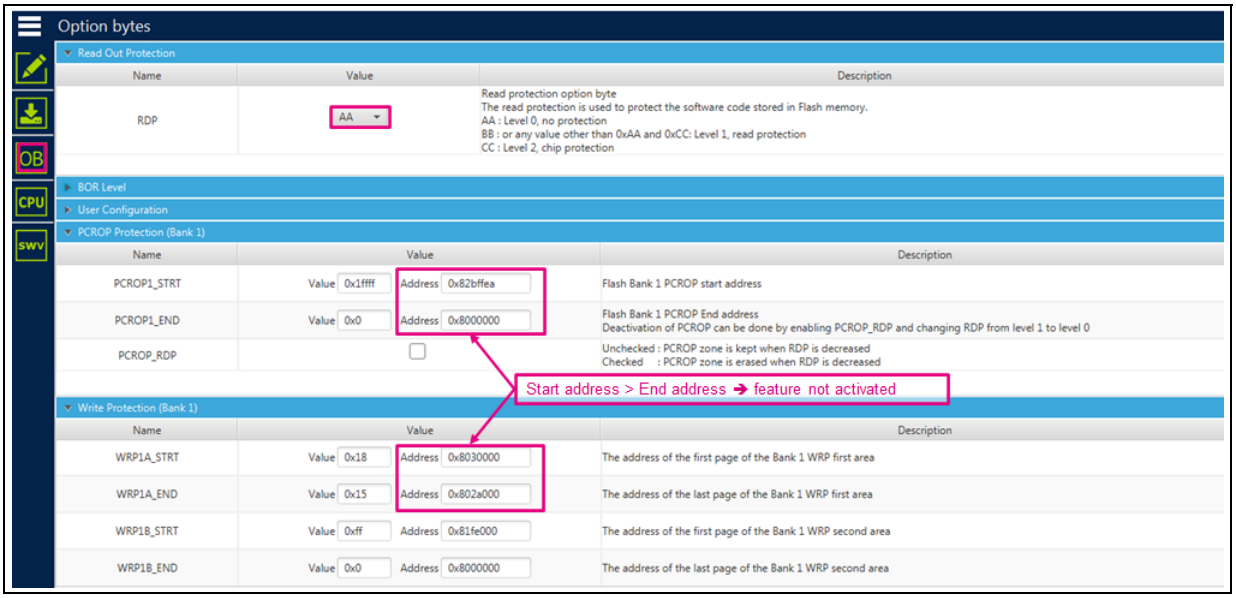
**1. 在“Under reset”模式下连接板（参见*图 19*）**

**图19. STM32CubeProgrammer连接菜单**


建议通过“Firmware upgrade”按钮升级ST-LINK的固件版本

2. 确认“OB”配置（参见*图 20*）

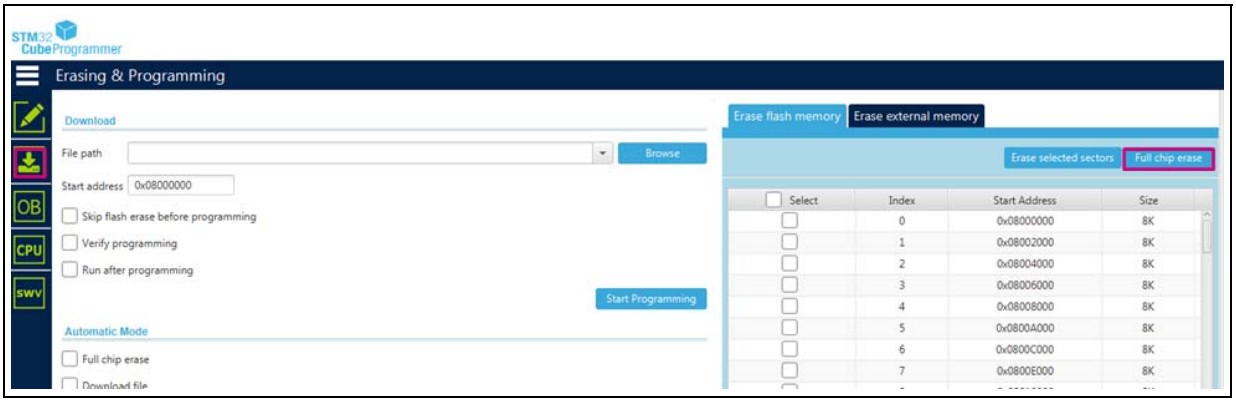
**图20. STM32CubeProgrammer*选项字节*界面**




STM32CubeProgrammer*选项字节*界面是STM32微控制器系列特有的。

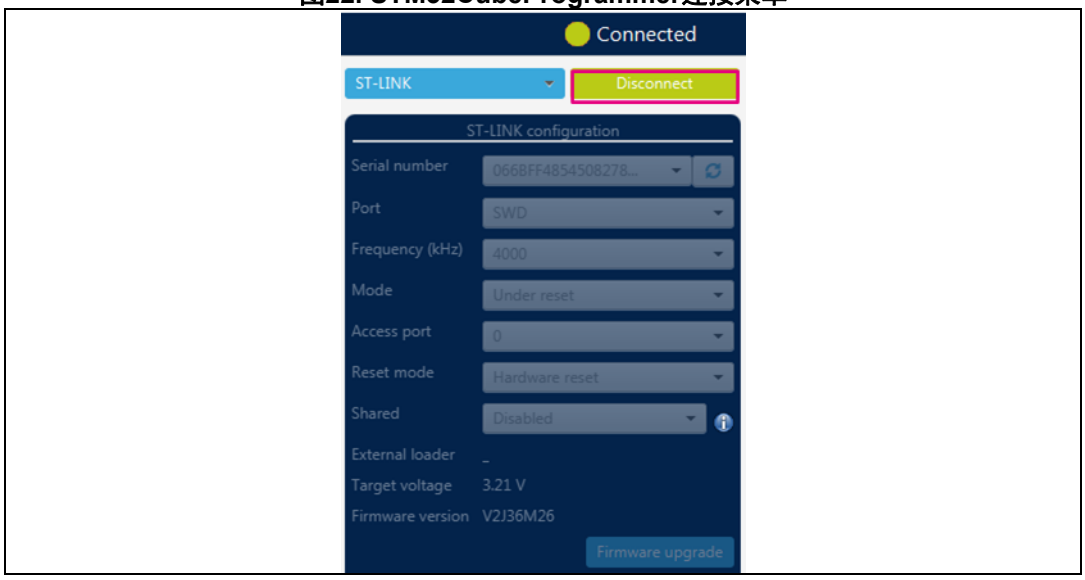
3. 擦除芯片：菜单目标 / 擦除芯片

**图21. STM32CubeProgrammer擦除**



4. 断开

图22. STM32CubeProgrammer连接菜单




## 8.2 应用程序编译

使用选定的工具链（IAR Embedded Workbench®、MDK-ARM或STM32CubeIDE）重新构建所有项目，如*第 6.5节：使用IAR Systems®工具链时的应用编译过程*中所述。

可将SB SFU项目软件下载到目标，无需启动调试会话（由SBSFU管理的安全保护会禁止JTAG连接，因为它被解释为外部攻击）。

## 8.3 Tera Term连接

Tera Term连接通过依次应用从*第 8.3.1 节*到*第 8.3.4 节*所述的步骤来实现。

### 8.3.1 禁止 ST-LINK

SBSFU管理的安全机制会禁止JTAG连接（解释为外部攻击）。必须禁用ST-LINK才能建立Tera Term连接。以下程序适用于从ST-LINK固件版本V2J29开始的更高版本[a]：

a) 在烧录SBSFU后重启板子（拔下/插入USB电缆）。

b) SBSFU应用程序以开发模式启动并配置安全机制。在产品模式下，仅检查安全机制是否处于正确值。

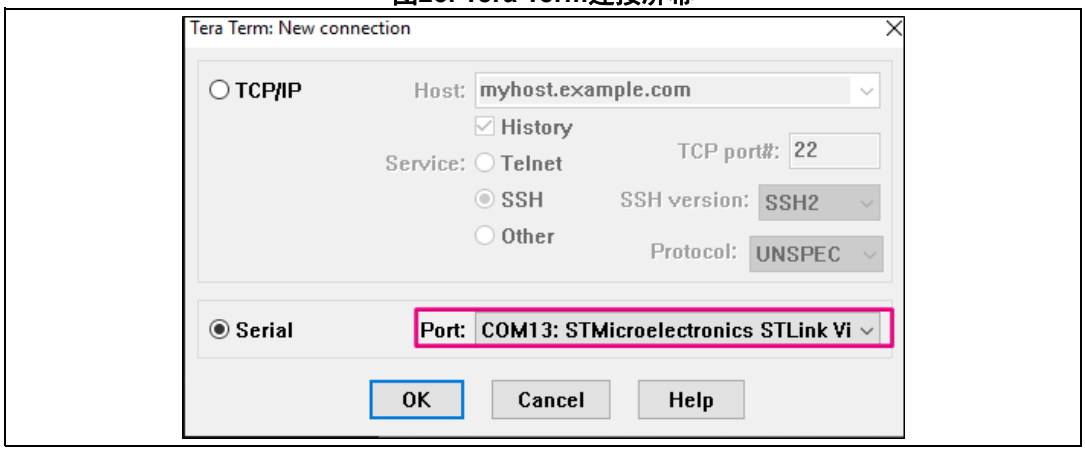
c) 再一次重启板子（拔下/插入USB电缆）：SBSFU应用程序启动，所配置安全措施打开，可以连接Tera Term。

### 8.3.2 Tera Term启动

Tera Term启动要求端口选择为*COMxx：STMicroelectronics STLink虚拟COM端口*。

*图 23*说明了基于端口COM54选择的示例。

**图23. Tera Term连接屏幕**



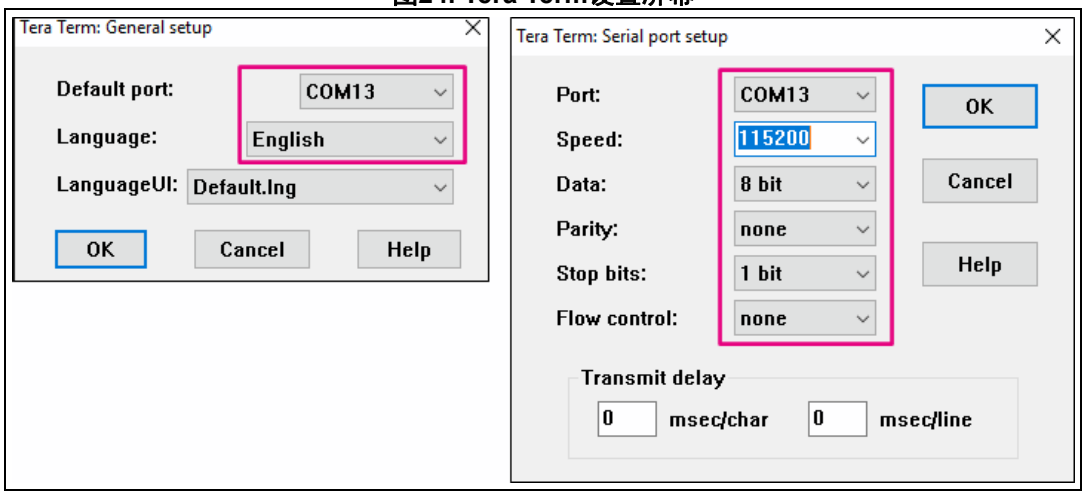

### 8.3.3 Tera Term配置

Tera Term配置通过*General*和*Serial port*设置菜单来实现。

*图 24*说明了*General setup*和*Serial port setup*菜单。

a. 确保嵌入在板上的ST-LINK调试器/编程器运行正确的固件版本（V2J29或更高版本）。如果版本情况不是这样，请首先升级此固件。

图24. Tera Term设置屏幕




利用*Menu Setup / Save Setup*来保存设置。

**注意：** 在每次插 / 拔USB电缆后，必须再次验证*Tera Term Serial port setup*菜单才能重新启动连接。**按下*Reset*按钮来显示欢迎屏幕**。

### 8.3.4 欢迎屏幕显示

欢迎屏幕显示在Tera Term中，如*图 25*所示。

**图25. SBSFU欢迎屏幕显示**


## 8.4 SBSFU应用程序执行

在每次重启时，应用程序都会检查用户是否通过按住用户按钮请求了新的固件下载。

如果没有下载请求，则应用程序将检查用户固件的状态

- 由于板子被擦除，所以没有固件可用。
- 应用程序不能跳转到固件，并返回来检查是否有下载请求。

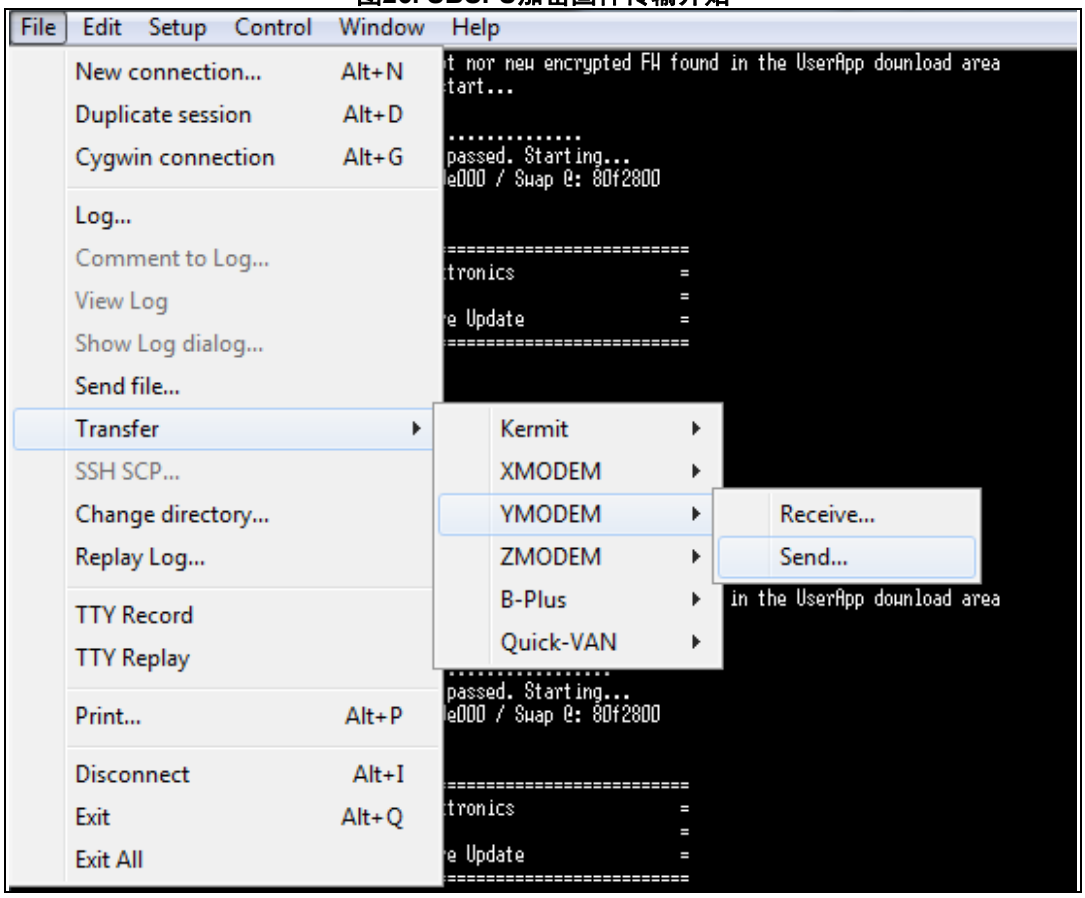
### 8.4.1 下载请求

当没有用户固件存在时，SBSFU自动等待下载程序来启动。否则，通过在STM32Nucleo板上按住用户按钮来获得下载请求。

### 8.4.2 发送固件

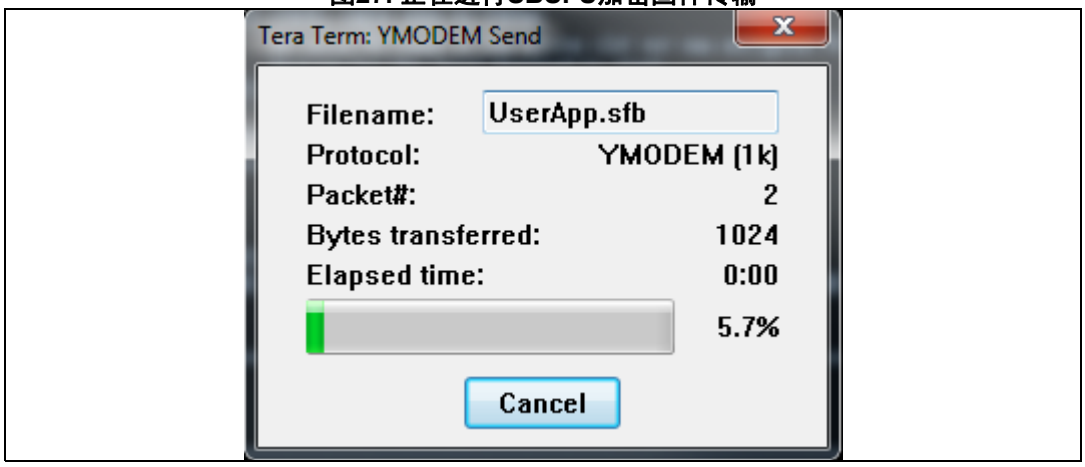
为了发送固件（*.sfb*），需要使用Tera Term中的*File* > *Transfer* > *YMODEM* > *Send*菜单，如*图 26*所示。

图26. SBSFU加密固件传输开始




当选择了*UserApp.sfb* 文件时，Ymodem传输开始。报告传输进度如*图 27*中所示。

图27. 正在进行SBSFU加密固件传输

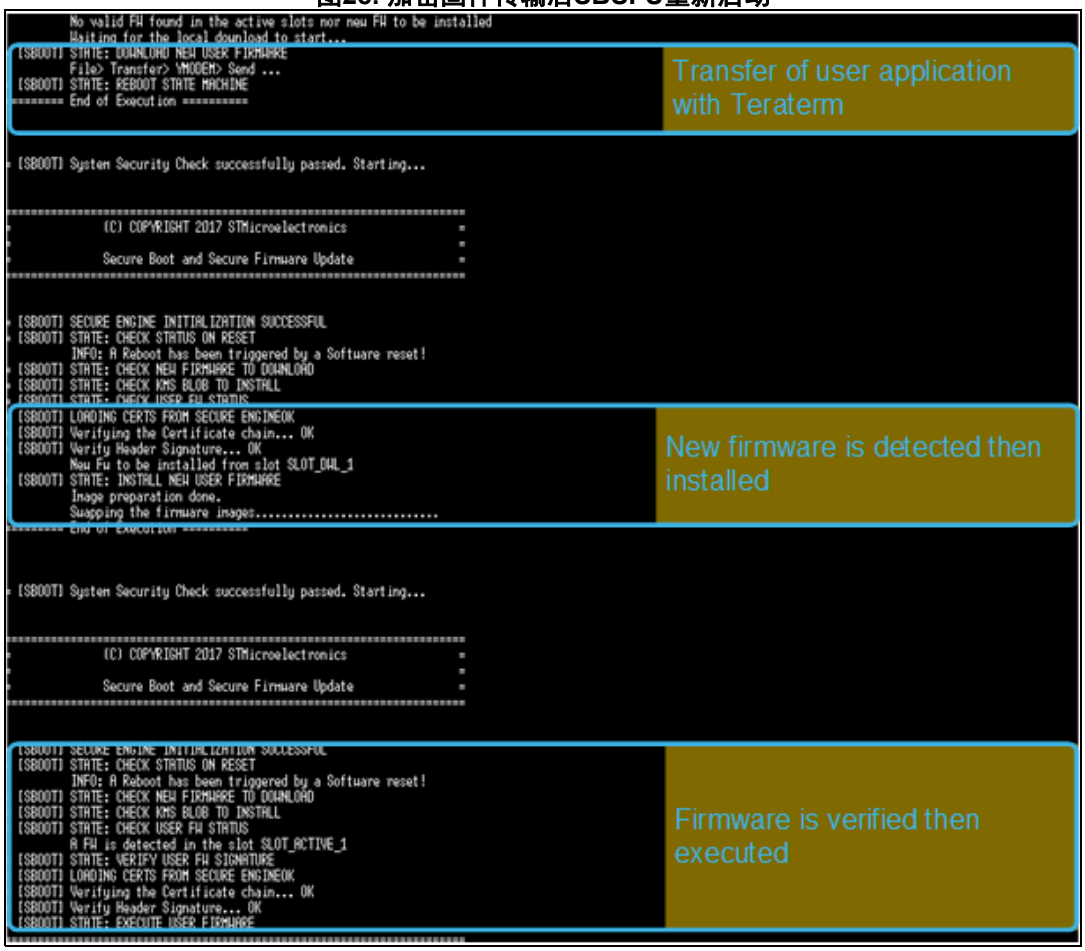



在程序开始时，进度表会暂停一会儿，此时SBSFU将验证固件头文件的有效性并擦除下载固件映像的Flash插槽。

### 8.4.3 文件传输完成

文件传输完成后，系统会强制重新启动，如*图 28*所示。

**图28. 加密固件传输后SBSFU重新启动**




随后*图 28*中显示的系统状态将提供以下信息：

- 没有固件可供下载。
- 固件被检测为已加密。用户固件被解密。
- 如果解密完成，则安装用户固件。
- 如果安装完成，则验证用户固件签名。
- 如果验证完成，则执行用户固件。

### 8.4.4 系统重启

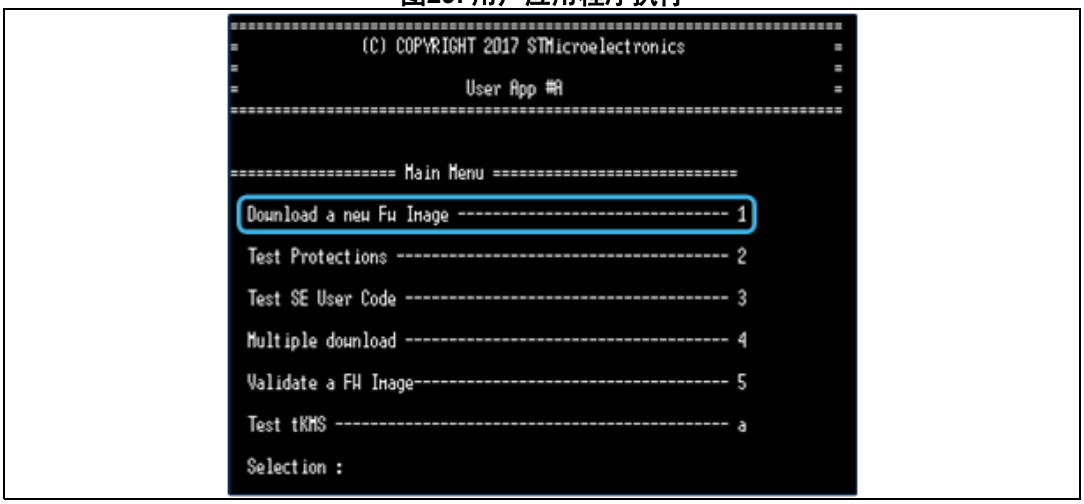
按下复位按钮强制系统重启：用户应用程序由SBSFU启动。

*注：* *在复位期间按住用户按钮，会触发强制下载状态，而不是执行用户应用程序。*

## 8.5 用户应用程序执行

用户应用程序根据*图 29*中所示选择来执行，并进一步按照从*第 8.5.1 节*到*第 8.5.3 节*所述来执行。

**图29. 用户应用程序执行**



### 8.5.1 下载新固件映像

下载新的固件映像的过程与*第 8.4 节*中SBSFU的步骤相同。

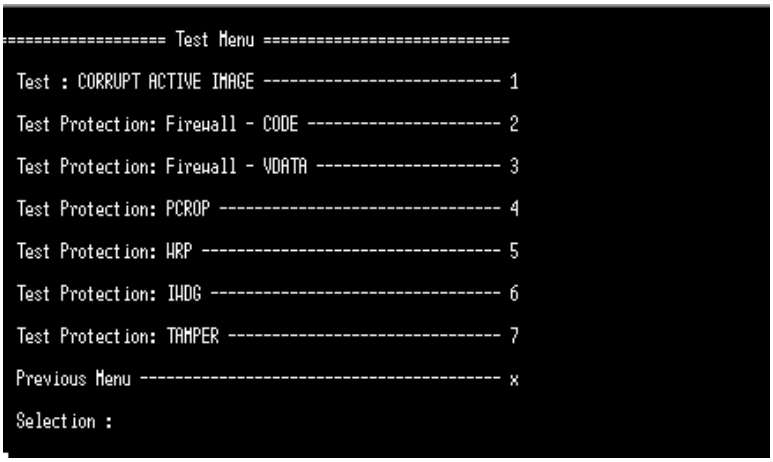
1. 发送固件：
   – 在Tera Term中，点击*File>Transfer>YMODEM>Send*
   – 选择*UserApp.sfb*（编译为*UserApp#B*）
2. 系统重启。
3. 安全启动状态机处理新映像：
   – 验证固件头文件
   – 解密固件
   – 安装固件
   – 验证固件签名
   – 执行固件

**图30. 通过用户应用程序下载加密的固件**

### 8.5.2 测试保护

测试保护菜单的示例如*图 31*所示。实际菜单取决于STM32系列。

**图31. 用户应用程序测试保护菜单**




作为测试运行功能，在每次禁止操作或错误注入的测试尝试时都会打印测试保护菜单：

- CORRUPT IMAGE测试（#1）
  - 在下次启动时会导致签名验证失败。
- 固件测试（#2, #3）
  - 尝试访问受保护的代码或数据（在RAM或Flash中）会导致重置
- PCROP测试（#4）
  - 尝试访问保护密钥的PCROP区域时会导致错误
- WRP测试（#5）
  - 尝试擦除写保护的代码时会导致错误
- IWDG测试（#6）
  - 不刷新看门狗会引起一个模拟死锁的重置
- TAMPER测试（#7）
  - 检测到篡改事件会导致重置
  - 为了产生篡改事件，用户必须将PA0（CN7.28）连接到GND（将手指靠近CN7.28可能就足够了）。

按下*x*键可以返回上一级菜单。

### 8.5.3 测试安全引擎用户代码

当前用户固件的版本和大小利用安全引擎服务进行检索，并打印在控制台中。

### 8.5.4 多重下载

在多映像配置中，可使用此菜单为下载选择目标插槽。可成功下载多个插槽。然后，通过选择#4功能触发安装。

### 8.5.5 固件映像验证

此功能仅在定义了ENABLE_IMAGE_STATE_HANDLING编译开关时可用。

选择的插槽识别参数可以是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。更多详细信息，请参见*附录J*。

## 8.6 在安全特性激活时进行新软件的编程

在所有安全特性启用的情况下将二进制文件刷写到Flash中后，不再能够直接更新SBSFU。您需要首先禁用以防SBSFU被擦除的安全配置。

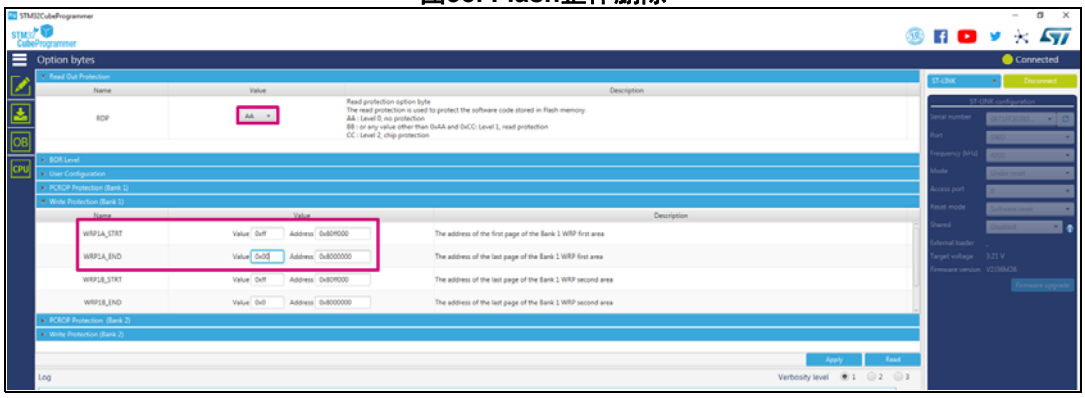
启动Cube Programmer应用程序，并打开*选项字节*菜单。

*选项字节*菜单如下图所示（RDP 1级和WRP保护）：

**图32. *选项字节*菜单**


对字段进行如下更新。将RDP设置为级别0，将WRP大小设置为0，并将起始地址设置为0xff。然后，点击*应用*以将Flash整体删除：

**图33. Flash整体删除**




在该步骤之后，Flash已完全擦除，可以对软件进行重新编程。

# 9 了解启动时的最后执行状态消息

*表 6*列出了主要错误信息及其解释。

**表6. 启动时的错误消息**

| 错误消息 | 意义 |
|---|---|
| No error. Success | 没有遇到问题。 |
| Memory fault | 内存故障由MPU故障处理程序报告。 |
| Hard fault | Arm® Cortex®-M硬性故障异常 |
| Tampering fault | TAMPER检测报告 |
| SE lock cannot be set. | 在启动活动固件之前尝试在*固件执行*（非特权）模式下配置安全引擎时遇到错误 |
| FW too big | 此错误意味着在本地下载过程中，头文件中指定的固件大小比下载插槽的容量要大。 |
| File not correctly received | 在本地下载过程中，下载操作未成功完成（Ymodem协议问题）。 |
| Firmware header authentication error | 在本地下载过程中，头文件无法成功认证。只有当存储在RAM中的头文件被更改时才会发生此错误（否则，下载会被忽略而不会触发严重故障）。 |
| Firmware decryption error | 在解密下载插槽的内容时遇到错误。此错误报告解密或认证问题，因为解密的最后阶段是对签名的检查。 |
| Firmware signature check error | 在安装过程中验证解密固件的签名时遇到错误。在示例代码中，不会发生此错误，因为在解密阶段就会捕获签名问题（报告“解密失败”）。 |
| Flash error | 安装过程中遇到闪存错误。 |
| External Flash configuration error | 在存储器映射模式下切换外部Flash时发生错误。 |
| Header erasing error | 在尝试擦除头文件时遇到Flash错误。 |
| Header validation error | 在尝试更新头文件以便验证安装时遇到Flash错误。 |
| Additional code detected beyond firmware | 插槽的未使用部分不为空（疑似恶意代码） |
| Error during swap process | 在安装过程中遇到错误：交换映像时出现故障（以前的固件和解密的固件）。 |
| Error during rollback process | 在安装过程回滚到上一固件映像时遇到错误。 |
| Backed-up firmware not identified | 未识别已备份固件：上一固件映像的指纹不正确。 |
| Backup copy error | 在备份插槽中保存下载插槽时检测到Flash错误 |
| Download slot erasing error | 在安装后擦除下载插槽时检测到Flash错误 |
| Trailer erasing error | 在擦除尾标区时检测到Flash错误 |
| Trailer update error | 在更新尾标区时检测到Flash错误 |
| Magic tag update error | 在更新尾标区中的干净标记时检测到Flash错误 |
| Firmware version rejected (anti-rollback) | 在安装过程中遇到错误：该固件版本不能被接受（新版固件已经安装或者版本低于允许的最低版本） |
| CM0 update process error | 在CM0+执行固件升级服务（FUS）或无线栈更新期间检测到错误 |
| Unknown error. | 未记录的错误（意外的异常或意外的状态机问题） |

# 附录A 安全引擎保护的环境

安全引擎（SE）概念定义了一个受保护的区域，用来输出可信环境中执行的一组安全功能。

SE向SBSFU应用示例提供以下功能：

- 安全引擎初始化功能
- 安全加密功能
  - AES-GCM和AES-CBC 解密
  - SHA256哈希和ECDSA验证
  - 敏感数据（密钥，AES上下文）不能离开受保护的环境，并且不能被未受保护的代码访问
- 对固件映像信息的安全读/写访问
  - 受保护Flash区的读和写操作
  - 只有受保护的代码才能访问此区域
- 安全映像状态处理
- 锁定安全引擎中一些功能的安全服务
  - 单向锁定机制：一旦锁定，除非通过系统重置，否则无法解锁
  - 一旦锁定，就不能再通过调用门机制来执行功能了
  - 在安全引擎示例中通过锁定机制锁定的功能为：
    - 安全引擎初始化功能
    - 带OEM密钥的安全加密功能
    - 对固件映像信息的安全读/写访问
    - 锁定安全引擎中一些功能的安全服务

*注：* *SE导出的功能可以根据最终用户应用程序的需要进行扩展。*

对于STM32WB系列，SE提供两种额外服务：

- 驱动由CM0+核心执行的无线栈/FUS更新。
- 在离开SBSFU执行程序之前锁定CM0+Flash区中存储的SBSFU对称密钥。

在KMS变体中，用安全密钥管理服务扩展了SE功能，它通过PKCS #11 API（基于KEY ID的API）向用户应用提供密码服务，这些API在受安全引擎保护的环境中执行。用户应用密钥存储在此受保护/隔离环境中。

在STSAFE-A变体中，用STSAFE-A中间件扩展了KMS，以便提供对密钥和服务的访问，这些密钥和服务由安全元件通过标准接口提供。通过存储在受保护/隔离环境中的对称密钥保护与STSAFE-A110的通信。

为了处理防火墙调用门机制并为用户提供一组安全的API，SE设计了由SE内核和SE接口组成的两级体系结构。

在使用MPU保护安全引擎时，调用门理念和两级体系结构也适用，如附录*A.2: 基于MPU的安全引擎隔离*所述。

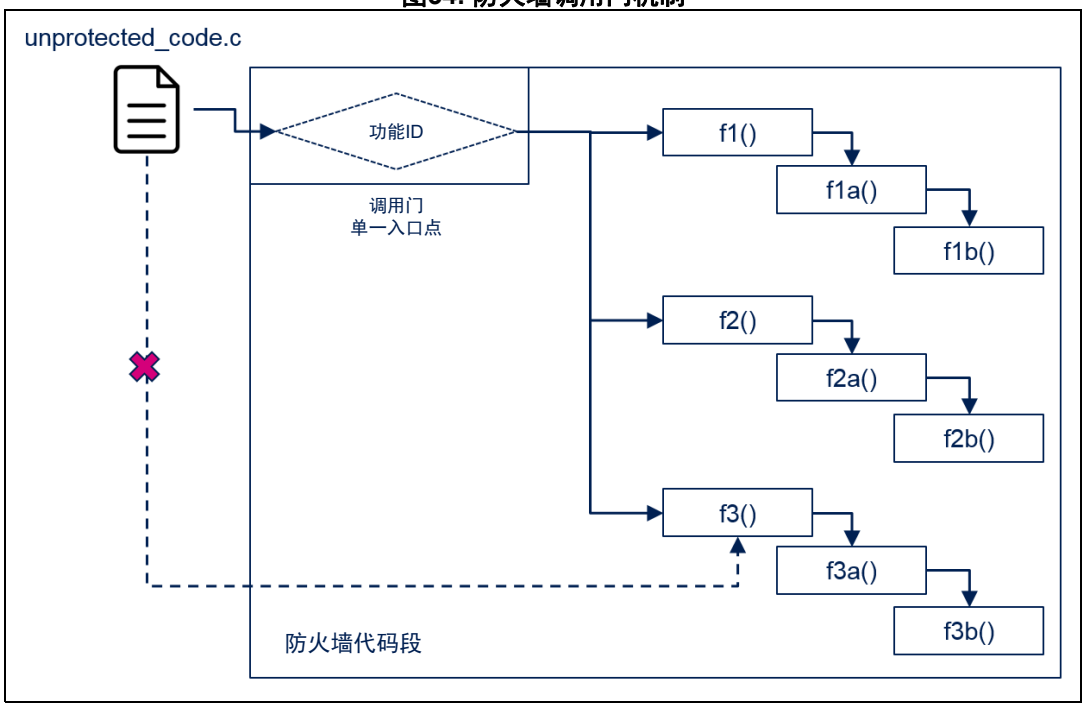
# A.1 基于防火墙的安全引擎隔离

## A.1.1 SE内核调用门机制

防火墙使用特定的“调用门”机制打开或关闭：必须使用单入口点（位于代码段基址的第二个字）来打开门并执行由防火墙保护的代码。如果访问受保护的代码而不通过调用门机制，则会产生系统重置。

因为对应调用门序列的唯一方法是通过单个调用门入口点，因此，假如外部应用程序需要调用受防火墙保护的多个功能（例如，加密和解密功能），则需要选择执行哪个内部函数。一种解决方案是，使用一个参数来指定执行哪个函数，例如CallGate(F1_ID)、CallGate(F2_ID)等等。根据参数，来调用内部正确的函数。

**图34. 防火墙调用门机制**



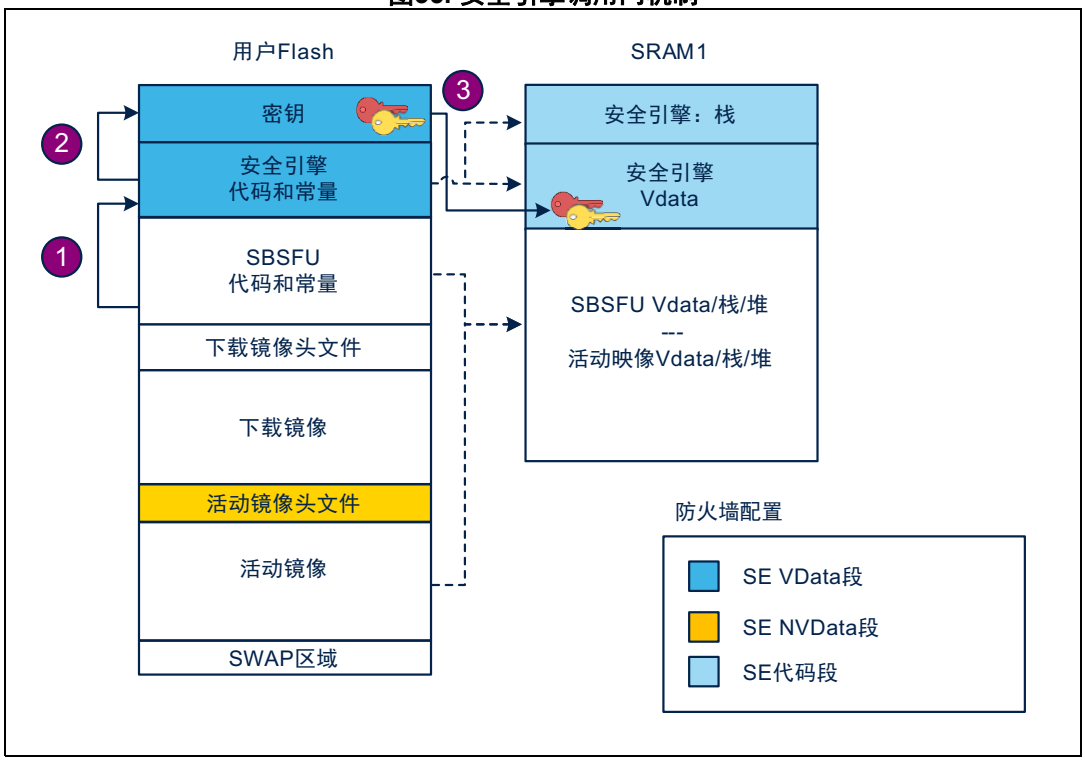

**注意：** 代码段必须包含防火墙打开时执行的所有代码。例如，如果调用序列是callgate->f1()->f1a()->f1b()，则全部三个函数f1()、f1a()和f1b()必须包含在代码段中。

*图 35*显示了执行加密操作（需要访问密钥）的步骤，以遵守调用门机制。

对于加密函数：

1. SBSFU代码调用调用门函数，打开防火墙并执行受保护的代码
2. 调用门函数检查参数和安全性，然后调用请求的Crypto函数
3. SE Crypto函数调用内部ReadKey函数，此函数可将密钥移入SRAM1的受保护部分，然后在加密操作中使用它们。

图35. 安全引擎调用门机制

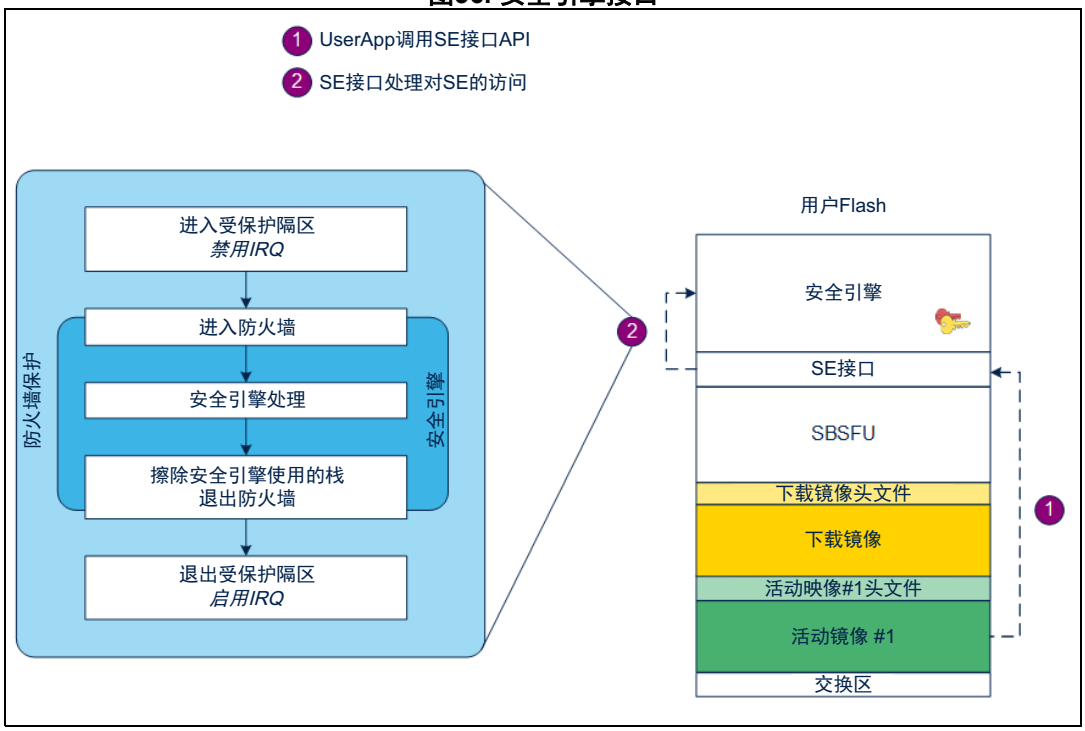



## A.1.2 SE接口

由防火墙保护的代码必须是不能中断的，在打开防火墙之前，是否禁用中断取决于用户代码。

SE接口提供了一个用户友好的封装，用来处理受保护区域的入口和出口，并在该区域中执行实际的SE调用门函数，如*图 36*中所示。

图36. 安全引擎接口




SE接口机制简化了对调用门的访问控制，可独立于用户实现。SE接口API与用户应用程序共享，因此可使用SE提供的服务以安全方式执行敏感操作（如果未被安全引擎锁定服务锁定）。

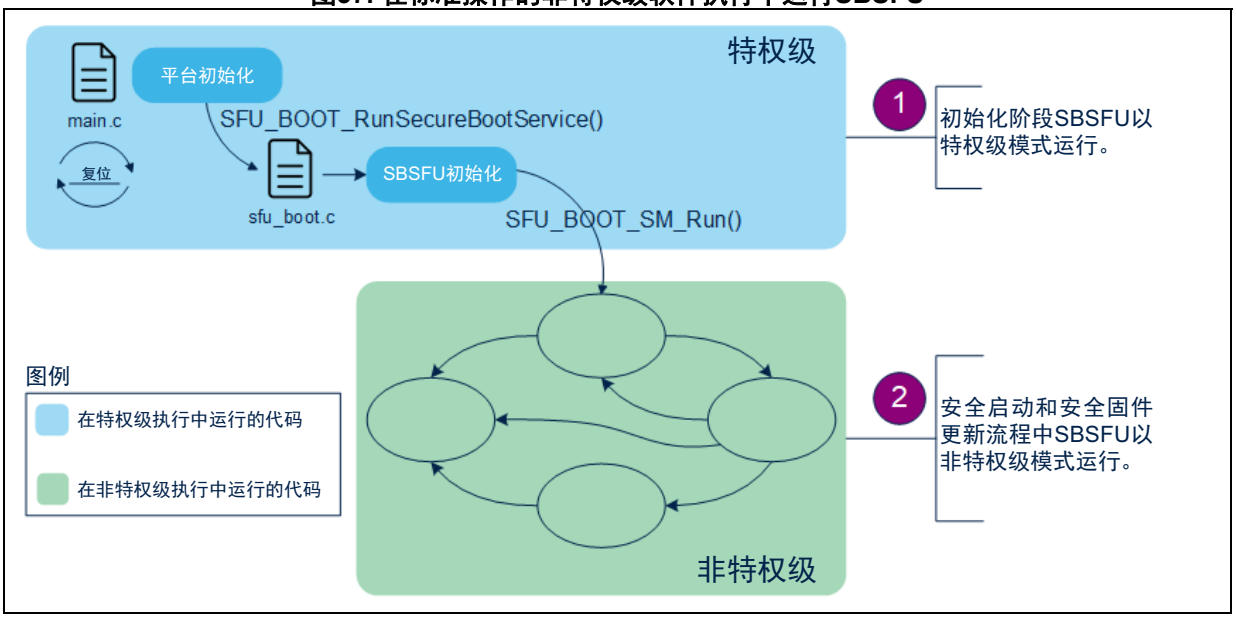
当需要低延迟的中断处理时，可以激活防火墙隔离环境内部的中断管理。在32L496GDISCOVERY和B-L475E-IOT01A板的*2_Images_OSC*变体中提供了示例。应用笔记*如何在用防火墙隔离的环境中激活中断管理*（AN5056）的7.3节详细描述了激活此选项所需的所有步骤。

## A.2 基于MPU的安全引擎隔离

### A.2.1 原理

基于MPU的安全引擎隔离依赖于软件执行的特权级和非特权级理念。默认必须以非特权级执行运行软件（在SBSFU或用户应用运行时），除非极特殊的操作，如平台初始化或中断处理。相关内容在*图 37*中介绍。

图37. 在标准操作的非特权级软件执行中运行SBSFU




当软件以非特权模式运行时，对访问安全引擎代码或数据的任何尝试都会导致MPU故障：如此可确保关键资产的隔离。

安全引擎的此类隔离是通过特定的MPU区域来实现的，如*表 7*所示。

表7. 用于安全引擎隔离的MPU区域

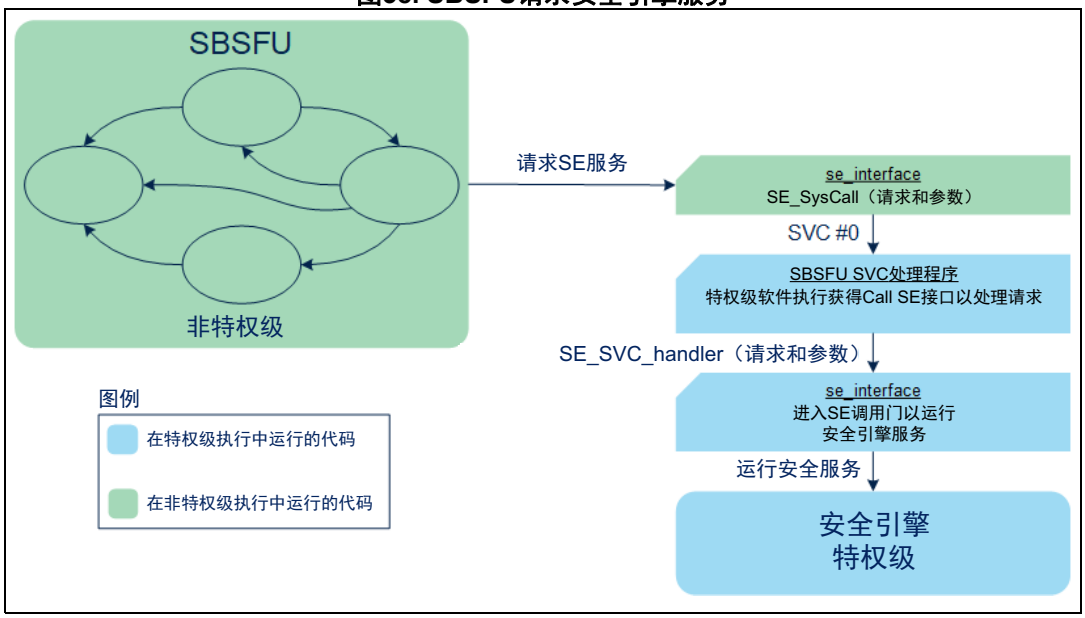
| 区域内容 | 特权权限 | 非特权权限 |
|---|---|---|
| 安全引擎代码和常量 | 只读<br>（允许执行） | 无访问权限 |
| 安全引擎堆栈和VDATA | 读/写<br>（不可执行） | 无访问权限 |

为了运行安全引擎服务，调用方必须首先通过受控接入点进入特权级软件执行。这是使用SE接口的理念来实现的（参见*第 A.1.2 节：SE接口*，注意，MPU保护替代了防火墙保护）。它提取特权级软件执行的请求：此请求包括触发监控器（SVC）调用。

在X-CUBE-SBSFU扩展包提供的SBSFU示例中，SBSFU应用实现了一个SVC处理程序，用于捕获此SVC调用并用另一个SE接口服务对其进行处理，以便通过其调用门进入安全引擎，如*图 38*所示。

*注：* *作为安全引擎访问控制的关键因素，SVC处理程序必须可信。*

图38. SBSFU请求安全引擎服务

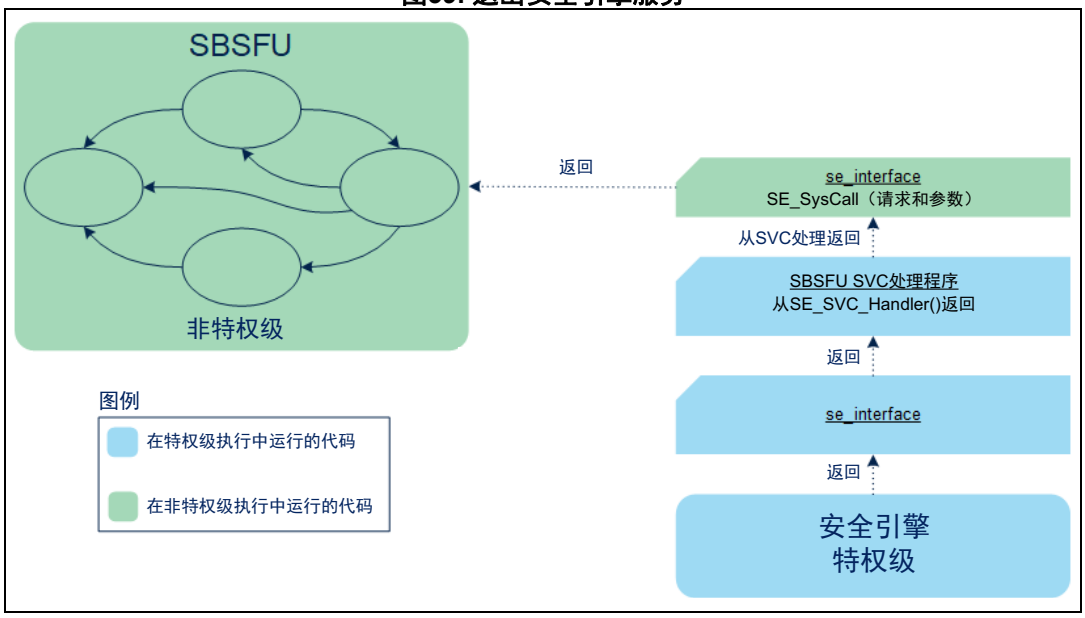



SE调用门机制使用*第 A.1.1 节：SE内核调用门机制*中描述的理念提供安全引擎服务的唯一入口。与*第 A.1.1 节*之间的区别在于，MPU保护替代了防火墙保护；调用门代码的位置限制只有MPU区域限制（调用门必须位于特权级代码区）。

当安全引擎处理结束时，调用门理念提供一个出口点，SVC调用返回流程适用。此返回流程使软件回到非特权级执行。然后，对安全引擎代码和数据的任何后续的直接访问都会导致MPU故障。

*图 39*中描述了安全引擎服务的退出。

图39. 退出安全引擎服务



### A.2.2 约束

访问安全引擎服务需要软件执行的特权级别时，才需要进行基于MPU的安全引擎隔离。SVC处理程序是特权级执行的受控接入点（必须是可信代码）。此外，在特权模式下运行的任何一段代码也都必须是可信的（中断例程、初始化代码及其他），因此受控接入点不会被绕过。对软件进行仔细分割并避免在不需要时授予特权级执行是十分重要的（软件必须尽可能以非特权模式运行）。

MPU控制Cortex®-M对存储器的访问。总线上任何充当主设备的外设都可以访问安全引擎代码和数据，而不会触发MPU故障（如DMA外设）。因此，需确保只有可信代码才能对这些外设进行编程。例如，在32F413HDISCOVERY的X-CUBE-SBSFU示例中，MPU区域涵盖DMA寄存器，以确保不能以非特权级执行对这些外设进行编程：只有特权级代码才能配置DMA。

最后不能不提的是，对于STM32F4系列和STM32F7系列，只有维持需要的MPU设置，才能确保安全引擎保护。如果用户应用代码不完全可信或故障可能被黑客利用，则在用户应用执行期间，必须维持MPU配置不变。

对于具有安全用户存储区的STM32系列，不存在后面这种限制。在启动用户应用之前，MPU保护被禁用且安全用户存储区保护被激活。激活后，对可安全存储区的任何访问（提取、读取、编程和擦除）都会被拒绝，生成总线错误。安全用户存储区（受保护环境）中的所有代码和秘密信息均完全隐藏。

# 附录B 双插槽配置

一些SBSFU应用示例处理位于内部Flash中的两个插槽。

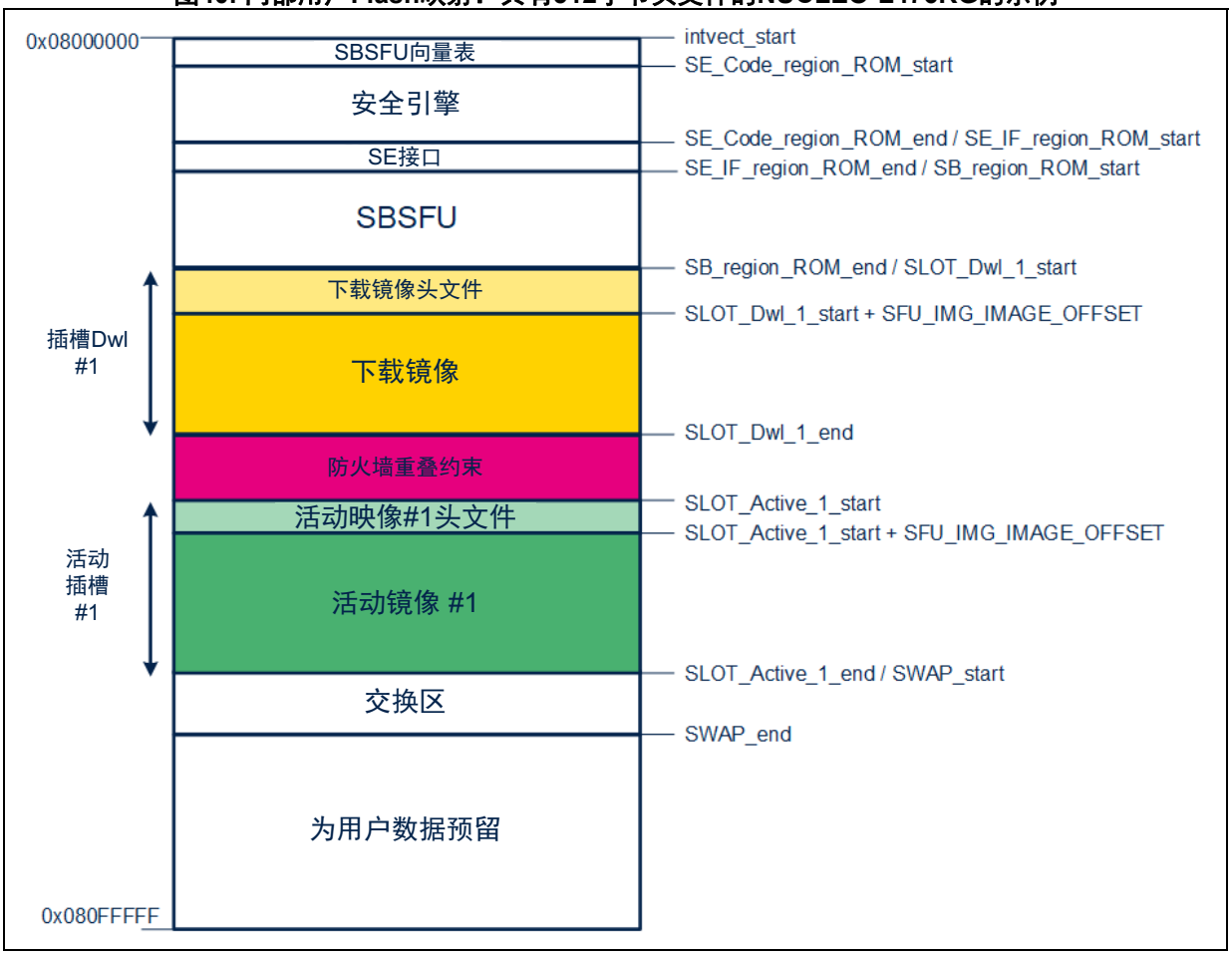
## B.1 元件和角色

- 活动插槽：
  - 该插槽包含了有效固件（固件头文件和固件）。这是SBSFU在启动时启动的用户应用程序（验证其有效性后）。
- 下载插槽：
  - 此插槽用于存储下载的固件（固件头文件和加密固件），以便在下次重启时安装。
  - 对于部分映像，此插槽的大小可能小于活动插槽的大小（包含完整映像）。可根据可能的最大部分映像确定下载插槽的大小。
- 交换区域：
  - 这是用来交换活动插槽和下载插槽的内容的Flash区。
  - 不过，此区域并不是每次Flash区交换都要使用的缓冲区。它用于移动第一个扇区，因此会在闪存中产生移位，从而可以逐扇区交换两个插槽。

*图 40*显示了NUCLEO-L476RG上的映射。插槽和交换元素的映射顺序取决于STM32系列：

- 对于具有安全用户存储区的STM32系列，必须在SBSFU代码受到安全用户存储区保护后立即映射活动插槽头文件。
- 对于STM32L4系列和STM32L4+系列，每个区中的防火墙代码和数据（活动插槽头文件）段与基址之间的偏移量必须相同（确保即使交换了区，秘密信息也始终受到保护）。

**图40. 内部用户Flash映射：具有512字节头文件的NUCLEO-L476RG的示例**

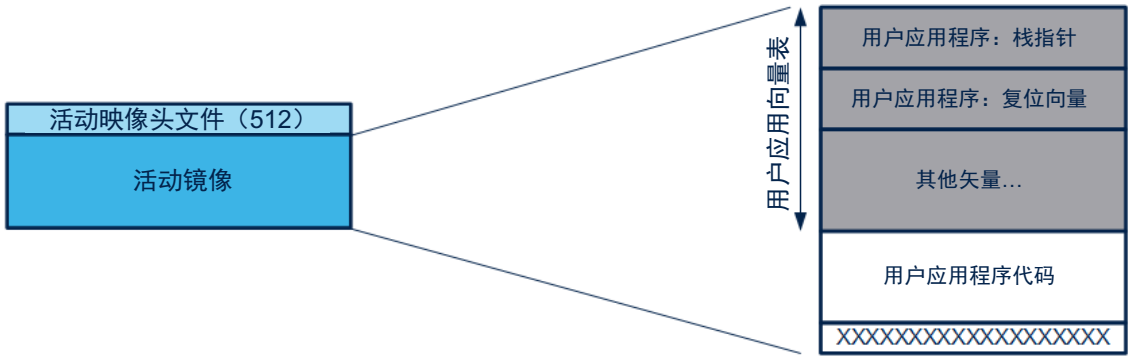


## B.2 映射定义

映射定义位于每个示例的*Linker_Common*文件夹中。

要启动应用程序，SBSFU使用用户应用程序堆栈指针值初始化SP寄存器，然后跳转到用户应用程序重置向量。请参见*图 41*。

**图41. 用户应用向量表（STM32L4系列的示例）**




源代码：sfu_fwing_common.c和system_stm32l4xx.c

```
1. End of SBSFU execution: SFU_IMG_ExecuteUserFW(void)
   /* Initialize user application's Stack Pointer */
  __set_MSP(*(__IO uint32_t *)(SlotStartAdd [SlotNumber] + SFU_IMG_IMAGE_OFFSET));
  /* Jump to user application */
  jump_address = *(__IO uint32_t *)((SlotStartAdd[SlotNumber] + SFU_IMG_IMAGE_OFFSET + 4));

2. Start of USER application: SystemInit(void)
   SCB->VTOR =(uint32_t)&__ICFEDIT_intvec_start__ ;   /* __ICFEDIT_SLOT_Active_1_start__ + 512 */
```

# 附录C 单插槽配置

一些SBSFU应用示例处理位于内部Flash中的一个插槽。

这种操作模式允许通过以下方式最大化用户固件大小：

- 减少闪存中的SBSFU占用空间
- 为用户应用程序分配更多的Flash空间

这些好处是以一些功能为代价的：

- 无法确保安全的固件映像编程：安装中断（断电）后，必须立即重新开始固件更新过程（包括下载阶段）。
- 如果新的固件映像不正确，将无法回滚。
- 用户应用程序无法下载新的固件映像：本地下载过程是更新活动用户代码的唯一方法。

## C.1 元件和角色

活动插槽：

- 该插槽包含了有效固件（固件头文件和固件）。这是SBSFU在启动时启动的用户应用程序（验证其有效性后）。
- 当下载并安装新的固件映像时（固件头文件验证之后），该插槽会直接更新

## C.2 映射定义

映射定义位于每个示例的*Linker_Common*文件夹中。

要启动应用程序，SBSFU使用用户应用程序堆栈指针值初始化SP寄存器，然后跳转到用户应用程序重置向量，请参考*图 41：用户应用向量表（STM32L4系列的示例）*。

# 附录D 加密方案处理

提供了四种密码方案作为例子来说明加密操作。默认的加密方案同时使用对称（AES-CBC）和非对称（ECDSA）加密。因此，它可以处理私钥（AES128私钥）以及公钥（ECC密钥）。

提供了两种候补方案。可使用SECoreBin编译器开关（名为“SECBOOT_CRYPTO_SCHEME”）进行选择。

在B-L4S5I-IOT01A板的STSAFE-A变体和KMS变体中配置基于X509证书的非对称方案。

## D.1 此软件包中包含的加密方案

*表 8*显示了使用SECBOOT_CRYPTO_SCHEME编译器开关选择的加密方案。

**表8. 加密方案列表**

| SECBOOT_CRYPTO_SCHEME值 | 身份验证 | 机密性 | 完整性 |
|---|---|---|---|
| SECBOOT_ECCDSA_WITH_AES128_CBC_SHA256（默认） | ECDSA | AES128-CBC | SHA256 |
| SECBOOT_ECCDSA_WITH_AES128_CTR_SHA256(1) | ECDSA | AES128-CTR | SHA256 |
| SECBOOT_ECCDSA_WITHOUT_ENCRYPT_SHA256 | ECDSA | 无(2) | SHA256 |
| SECBOOT_AES128_GCM_AES128_GCM_AES128_GCM(3) | AES-GCM | | |
| SECBOOT_X509_ECDSA_WITHOUT_ENCRYPT_SHA256 | ECDSA | 无(2) | SHA256 |

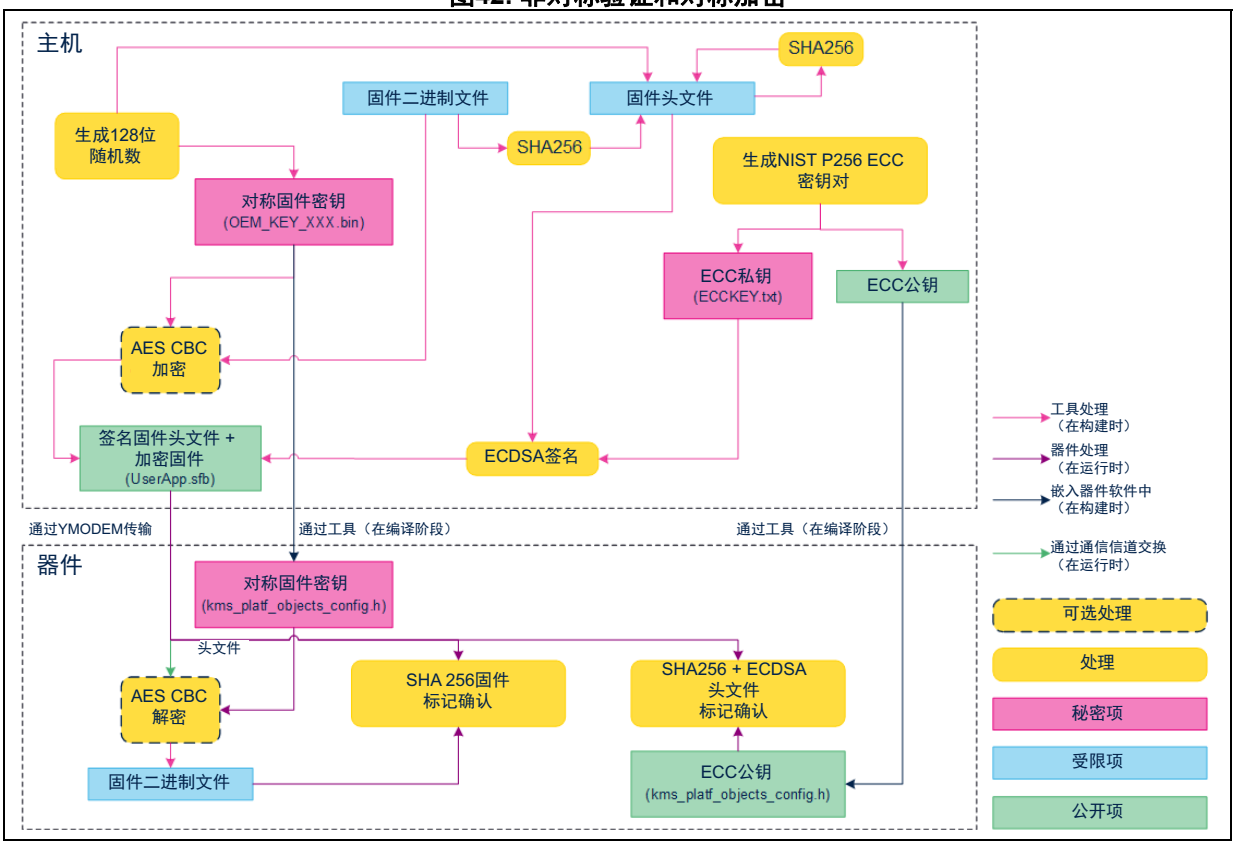
1. 为具有外部Flash和OTFDEC支持的产品选择此加密方案。
2. 还必须通过将编译器开关SFU_IMAGE_PROGRAMMING_TYPE设置为值SFU_CLEAR_IMAGE，对SBSFU项目进行配置，来处理明文固件映像。
3. 对于对称密码方案，强烈建议为每个产品配置唯一的对称密钥。

## D.2 非对称验证和对称加密方案

### D.2.1 具有完整软件实现的加密方案

这些方案（SECBOOT_ECCDSA_WITH_AES128_CBC_SHA256，SECBOOT_ECCDSA_WITHOUT_ENCRYPT_SHA256）用于固件解密和验证，如*图 42*中所示。

**图42. 非对称验证和对称加密**

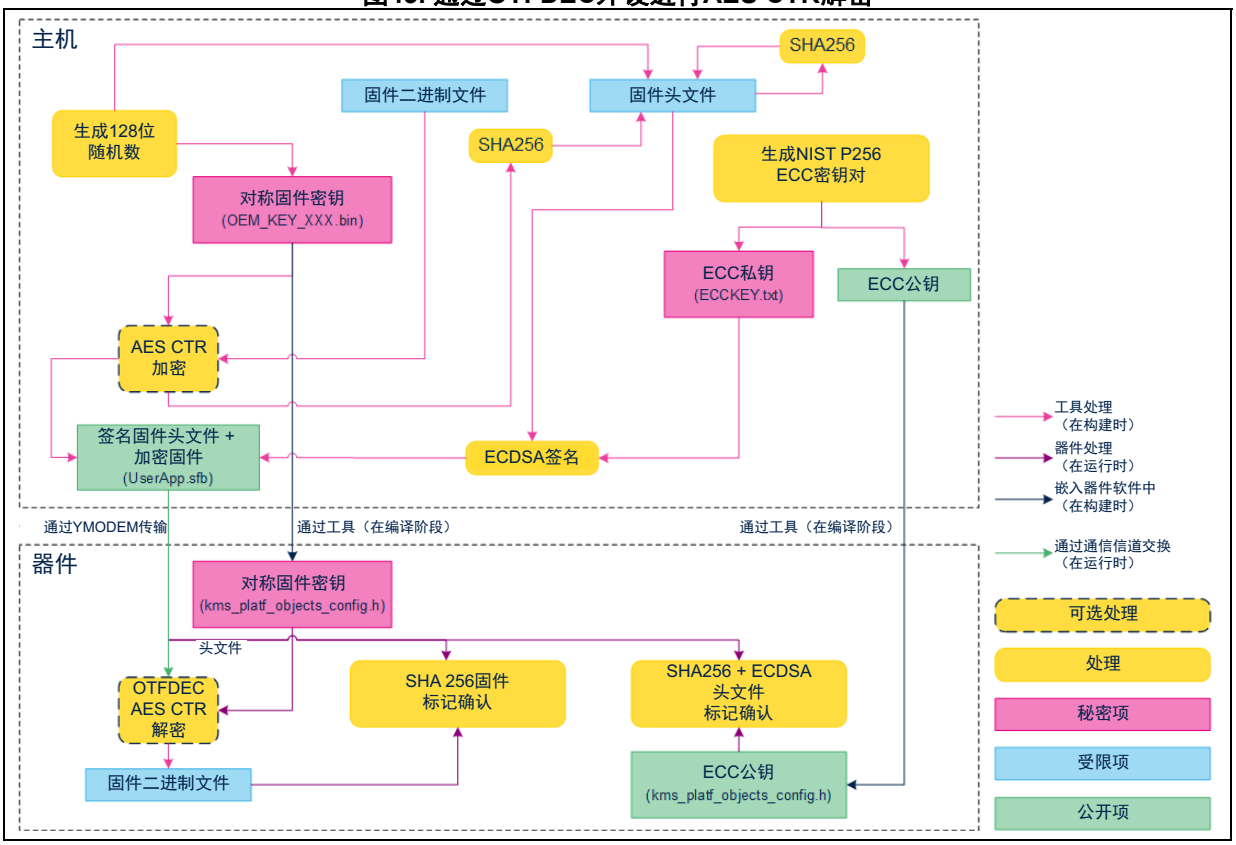


### D.2.2 通过OTFDEC外设进行AES CTR解密

STM32H7B3系列产品中嵌入的OTFDEC外设能够以低延迟损失直接解密内容，无需分配SRAM。OTFDEC是一个可以基于读取请求地址信息动态解密的硬件模块。

方案（SECBOOT_ECCDSA_WITH_AES128_CTR_SHA256）用于固件解密和验证，如*图 43*中所示。

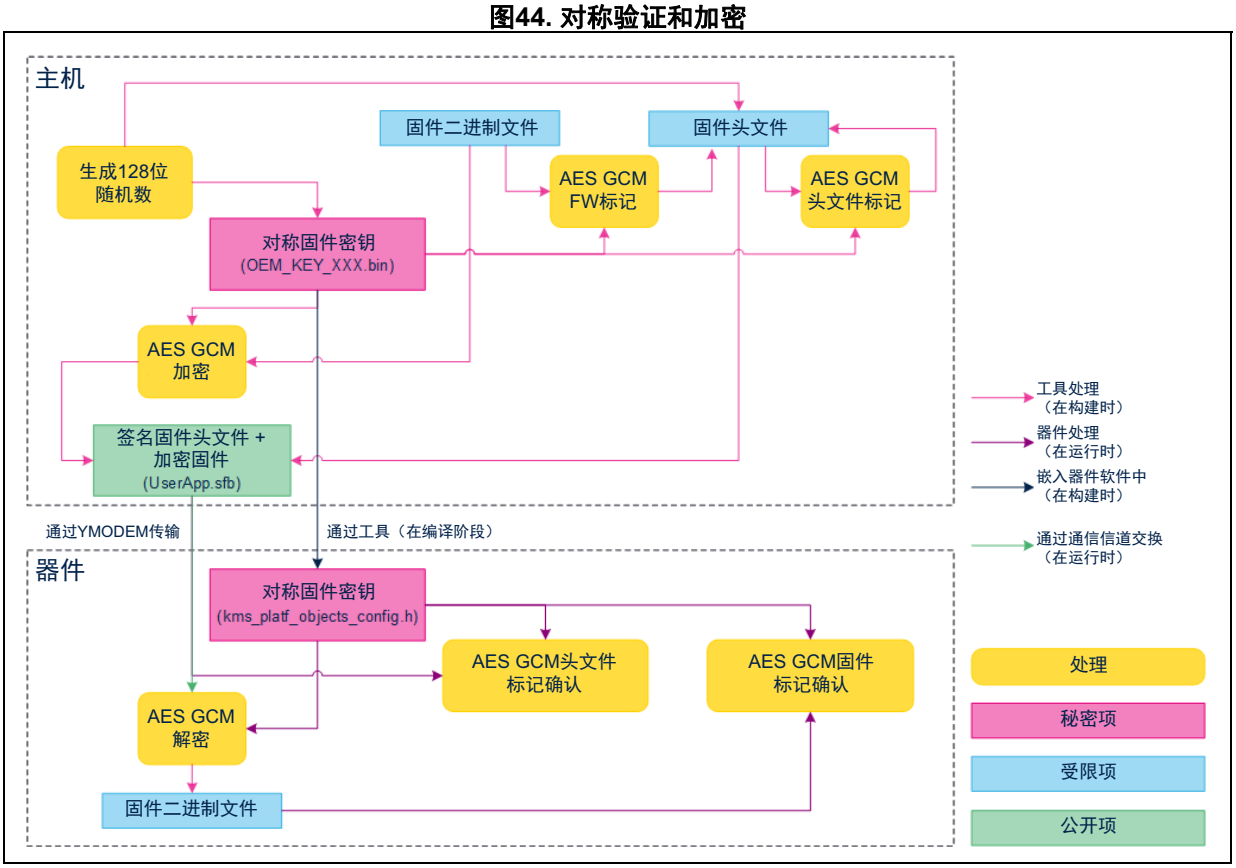
**图43. 通过OTFDEC外设进行AES CTR解密**



## D.3 对称验证和加密方案

此方案（SECBOOT_AES128_GCM_AES128_GCM_AES128_GCM）用于固件解密和验证，如*图 44*中所示。

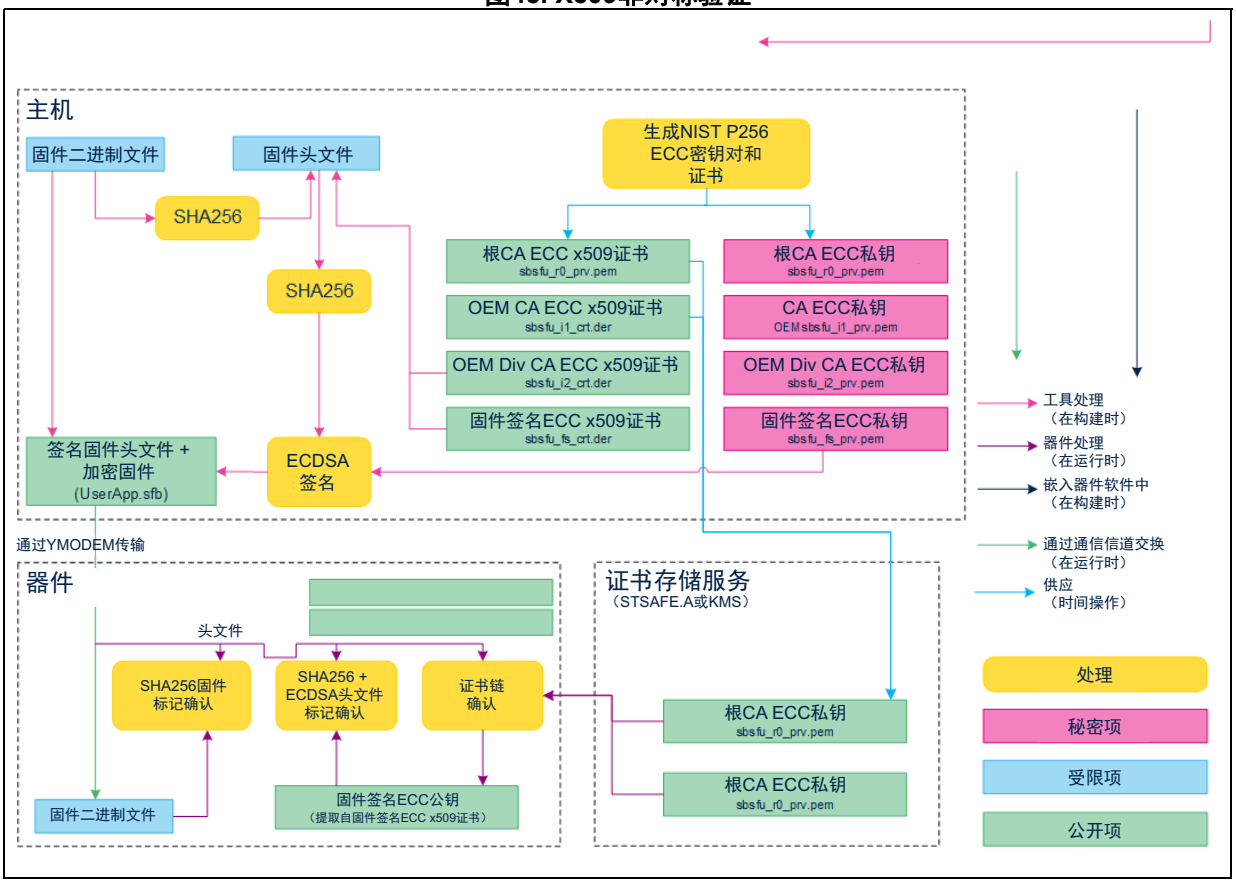
**图44. 对称验证和加密**



## D.4 基于X509证书的无固件加密的非对称方案

此方案（SECBOOT_X509_ECDSA_WITHOUT_ENCRYPT_SHA256）用于固件验证，如*图 45*中所示。

**图45. X509非对称验证**

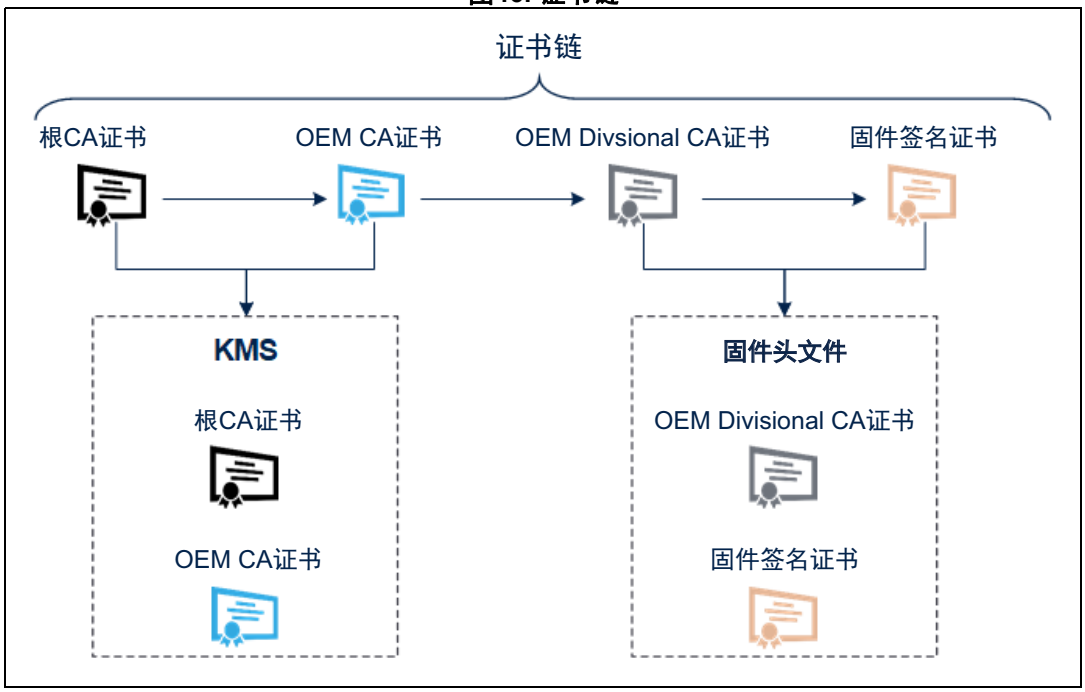



基于X509证书的非对称方案利用X509证书链提供公钥，用于验证固件头文件签名。

在提供的示例中，安全元素STSAFE-A110或KMS非易失性存储器中嵌入了两个证书（根CA和OEM CA（第一个中间CA）），而第二个中间CA和叶证书（固件签名证书）则作为固件头文件的一部分来提供。

## D.5 非对称验证和对称加密方案

图46. 证书链




为了使用叶证书中包含的公钥，SBSFU代码首先验证证书链，以确保提供的固件签名公钥是真实的。在验证证书链后，固件签名公钥被用于验证固件头文件签名。

## D.6 安全启动和安全固件更新流程

*图 47*和*图 48*表明了加密操作（采用固件加密的非对称加密方案）如何集成到SBSFU执行启动流程中。

**图47. SBSFU双插槽启动流程**

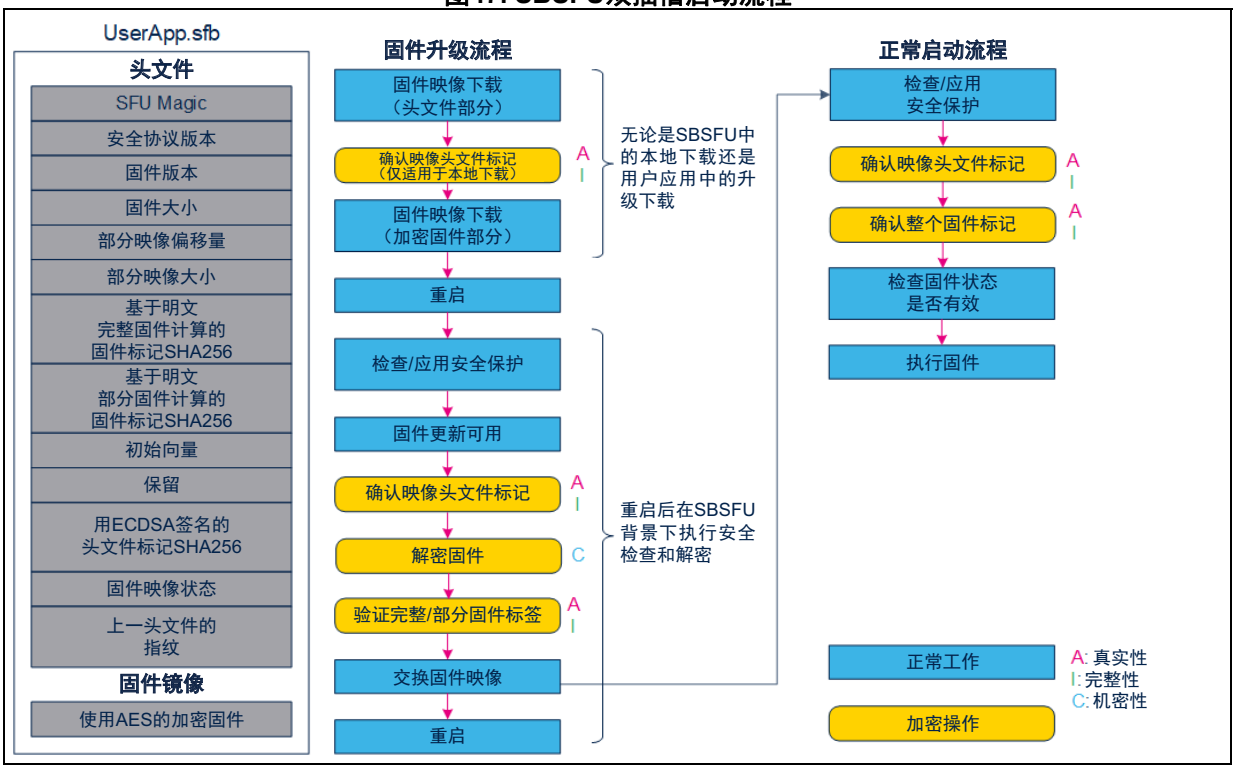
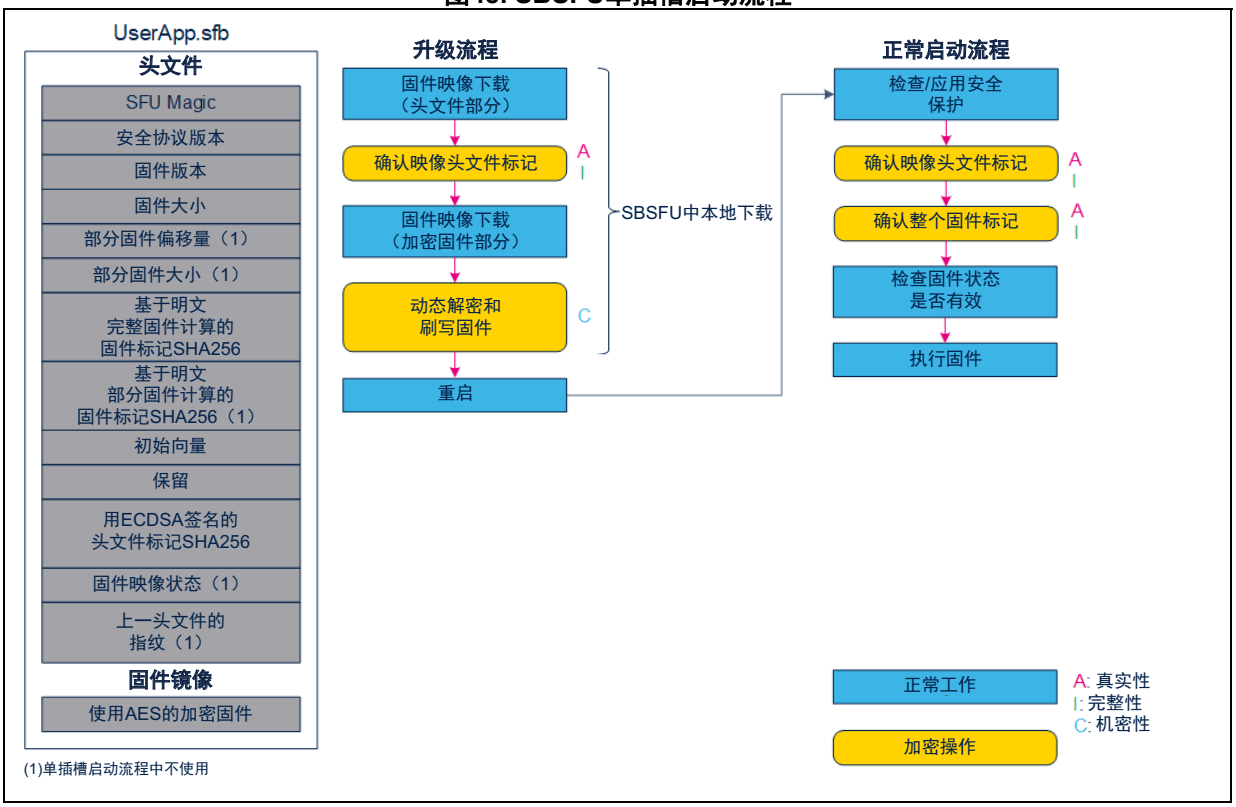


图48. SBSFU单插槽启动流程



# 附录E 固件映像准备工具

X-CUBE-SBSFU STM32Cube扩展包随附*prepareimage*固件映像准备工具，可以：

- 考虑到选定的加密方案和密钥
- 有需要时加密固件映像
- 通过提取两个完整映像之间的二进制文件差异，生成部分固件映像
- 生成固件头文件，其具有验证和完整性检查所需的所有数据

*prepareimage*工具以两种格式提供：

- Windows®可执行文件：需要标准Windows®命令解释器
- Python™脚本：Python™解释器以及*Middlewares\ST\STM32_Secure_Engine\Utilities\KeysAndImages\readme.txt*中列出的组件是必需的。

Windows®可执行文件可以快速轻松地使用该软件包，其中有全部三种预定义的加密方案。作为源代码提供的Python™脚本提供了以灵活方式定义其他加密方案的可能性。

*注：* *参见附录F和附录G了解KMS和STSAFE-A的特殊性。*

## E.1 工具位置

Python™脚本以及Windows®可执行文件位于安全引擎组件的文件夹*Middlewares\ST\STM32_Secure_Engine\Utilities\KeysAndImages*。

## E.2 输入

软件包随附了一些默认密钥和加密设置，在文件夹*Applications\2_Images\2_Images SECoreBin\Binary*中。

以下每个文件都可以这样使用，或者考虑用户设置而修改：

- *ECCKEY1.txt*：PEM格式的ECC私钥。它用来签名固件头文件。此密钥**未**嵌入到*SECoreBin*中，而对应的公钥工具则将其生成在*se_key.s*文件中
- *nonce.bin*：这是一个随机数（使用AES-GCM时）或IV（使用AES-CBC时）。该值由工具自动添加到固件头文件中。
- *OEM_KEY_COMPANY1_key_AES_CBC.bin*：对称AES-CBC密钥。此密钥用于AES-CBC加密和解密操作，并嵌入在*se_key.s*文件中。*OEM_KEY_COMPANY1_key_AES_GCM.bin*文件类似，对应AES-GCM的密钥，取决于用户选择哪种方案
- *OEM_KEY_COMPANY1_key_AES_GCM.bin*：对称AES-GCM密钥。此密钥可用于所有AES-GCM操作，并嵌入在*se_key.s*文件中。*OEM_KEY_COMPANY1_key_AES_CBC.bin*文件类似，对应AES-CBC的密钥，取决于用户选择哪种方案

该工具根据文件Applications\2_Images\2_Images SECoreBin\Inc\se_crypto_config.h中的SECBOOT_CRYPTO_SCHEME选择的加密方案，使用适当的一组文件*Applications\2_Images\2_Images SECoreBin\Inc\se_crypto_config.h*。

## E.3 输出

该工具会生成：

- *SECoreBin*项目中编译的*se_key.s*文件：该文件包含嵌入在设备中的密钥（适时使用的对称私钥和ECC公钥）以及用于访问它们的代码。在从IDE运行工具时，此文件位于*Applications\2_Images\2_Images SECoreBin\EWARM*中。如果是多映像配置，则为每个映像生成一组密钥。
- *.sfb*文件，其中包含用户固件头文件和加密的用户固件映像（当所选加密方案对用户固件进行加密时）。当从IDE运行该工具时，该文件将在*Applications\2_Images\2_Images UserApp\Binary*。
- *bin*文件合并了SBSFU、UserApp和活动固件映像头文件。使用烧录工具将此文件烧录到设备中使得UserApp安全过程变得简单化，因为固件映像和其头文件已经包含在内并已正确安装。安装UserApp不需要额外通过SBSFU来进行。

  对于具有OTFDEC支持和外部Flash的STM32器件，生成两个独立的二进制文件：

  – 第一个二进制文件合并了SBSFU二进制文件和活动固件映像头文件（*SBSFU_UserApp_Header.bin*），将刷写到内部Flash中。
  – 第二个二进制文件（*UserApp.sfb*）将刷写到外部存储器中的活动插槽起始地址。

  参见*附录I*和*附录H*了解STM32H7B3、STM32H750、STM32WB55和STM32WB5M器件的特殊性。

  **注意：**

  – 在将*bin*文件编程到器件中之前，必须执行整体删除。为了检测是否有任何恶意软件，SBSFU会在启动时检测活动插槽中的UserApp后面无其他代码。
  – 当活动插槽位于外部Flash中时，必须在对*bin*文件进行编程之前擦除活动插槽。
- 两个日志文件（*output.txt*）位于
  *Applications\2_Images\2_Images SECoreBin\EWARM*
  和
  *Applications\2_Images\2_Images UserApp\EWARM*
  中，用于跟踪*prebuild.bat*和*postbuild.bat*的执行情况。

## E.4 IDE集成

*prepareimage*工具与IDE集成，作为Windows®批处理文件，用于：

- *SECoreBin*应用程序的预生成操作：在此阶段，管理加密密钥
- *UserApp*应用程序的后生成操作：在此阶段，构建固件映像

使用IDE进行编译时，会处理密钥和固件映像。用户不需要额外操作。编译步骤结束时：

- 所需密钥嵌入到*SECoreBin*二进制文件中
- 生成可安装的固件镜像文件，*sfb*格式文件。*sfb*文件包含固件头，SBSFU通过YModem协议来传输并安装它。
- *bin*文件可以用于UserApp测试（*SBFU_UserApp.bin*）。

批处理文件，在IDE中集成该工具，位于文件夹*Applications\2_Images\2_Images SECoreBin\EWARM*：

- *prebuild.bat*：在编译*SECoreBin*项目时调用该工具来执行预生成操作
- *prebuild.bat*：在编译*UserApp*项目时调用该工具来执行后生成操作

这些批处理文件允许从Windows® 可执行变体无缝切换到 *prepareimage*工具的Python™脚本变体。该过程在文件本身中进行了描述。

## E.5 部分映像

相比于活动固件映像，部分映像只包含要安装的新固件映像的二进制部分。

使用部分映像具有多种优势：

- 要下载的固件映像更小（缩短下载时间）
- 固件映像的安装速度更快
- 存储器映射优化，可能出现更小的下载插槽（部分映像的最大大小）和更大的活动插槽（完整映像的最大大小）

只有双插槽变体才会有此特性（在单插槽变体上不可用）。

对于部分映像，头文件结构包含两个固件标记实例：

- 部分固件标签：仅部分映像的标签。在映像安装阶段，SBSFU会检查此标签。
- 固件标签：完整映像的标签（在安装部分映像后）。在映像启动阶段，SBSFU会检查此标签。

*prepareimage*工具可用于生成部分固件映像，从活动映像和要安装的新固件映像开始：

1. 首先执行两个完整映像明文的二进制比较
2. 然后，执行常规的映像准备流程

请参考*prepareimage*工具的*readme.txt文件*（*Middlewares\ST\STM32_Secure_Engine\Utilities\KeysAndImages\readme.txt*）中描述的*部分更新的示例*。

# 附录F KMS

## F.1 密钥更新过程说明

PKCS #11 API通过对象管理密钥，这些对象包含不同类型的信息：

- 对象头文件：提供关于对象本身的信息，如属性大小、属性数量和对象ID
- 对象属性：如类型、大小和值

静态嵌入式密钥被嵌入代码中，不能修改。相反地，受保护/隔离环境中的NVM存储器中的可更新密钥是可以修改的：

- 具有动态ID的可更新密钥可通过安全对象创建流程进行创建，该流程在受保护/隔离环境中进行，确保密钥仍然处于受保护/隔离环境中（密钥值存储在NVM存储器中，只返回对象ID给应用）。
- 通过使用静态嵌入式根密钥的安全更新流程，可以更新NVM存储器中具有静态ID的可更新密钥（真实性检查、数据完整性检查和数据解密）。这意味着为确保密钥的真实性、完整性和机密性，密钥必须以特定格式提供给KMS。为KMS示例提供的工具能够基于ECDSA非对称加密技术（适用于数据真实性/完整性验证）和AES-CBC对称加密技术（适用于数据机密性）自动生成加密对象。在加密对象下载到设备中后，SBSFU应用在下一次系统复位时检测该对象，并通过KMS安全更新流程（C_STM_ImportBlob()函数）处理该对象。

*图 49*和*图 50*显示了密钥的创建和更新过程。

**图49. 加密对象的创建**

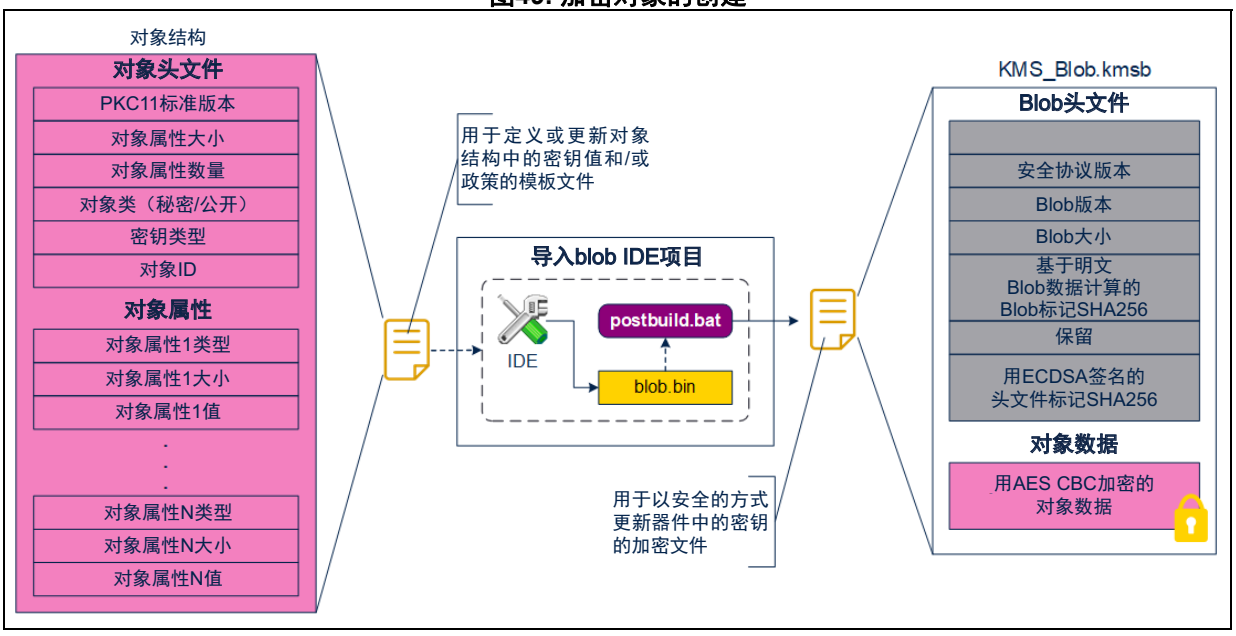
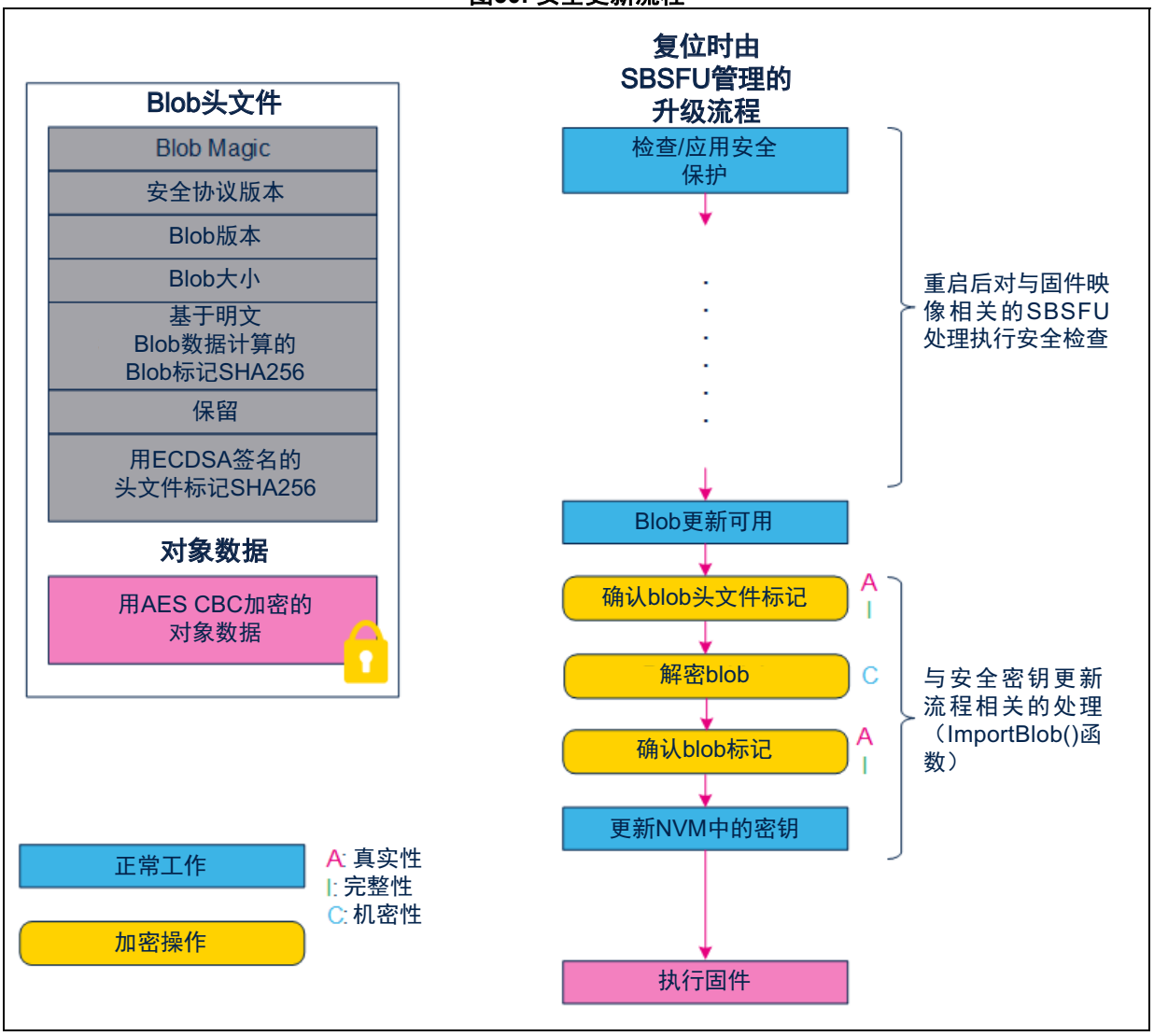


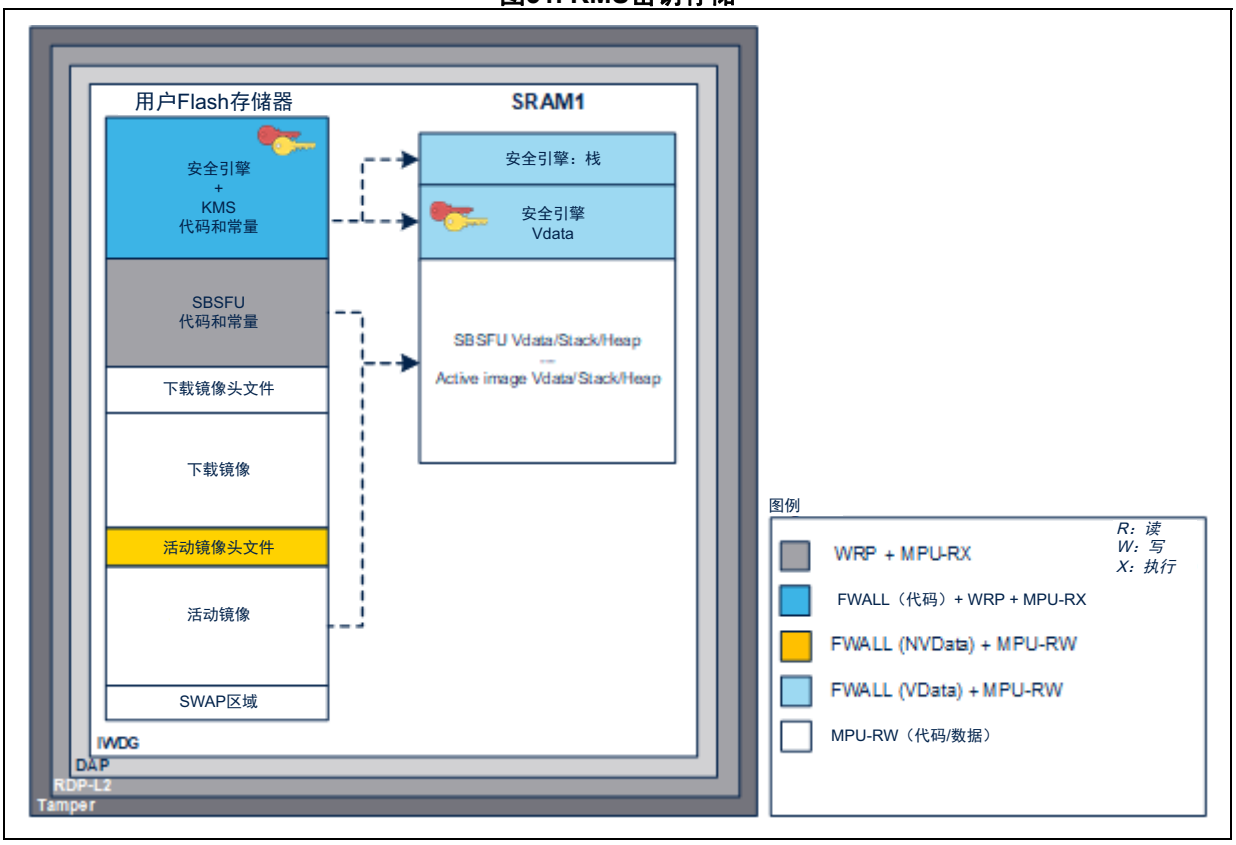
图50. 安全更新流程




## F.2 SBSFU静态密钥的生成

有了KMS中间件集成技术，SBSFU密钥不再存储在受PCROP保护的区域，而是位于在安全隔区中运行的KMS代码中，如*图 51*所示。在SECoreBin编译阶段，*prebuild.bat*更新文件*kms_platf_objects_config.h*（位于*Applications\2_Images\2_Images SECoreBin\EWARM*中）中的SBSFU静态嵌入式密钥（而不是标准变体中执行的*se_key.s*文件更新）。

图51. KMS密钥存储

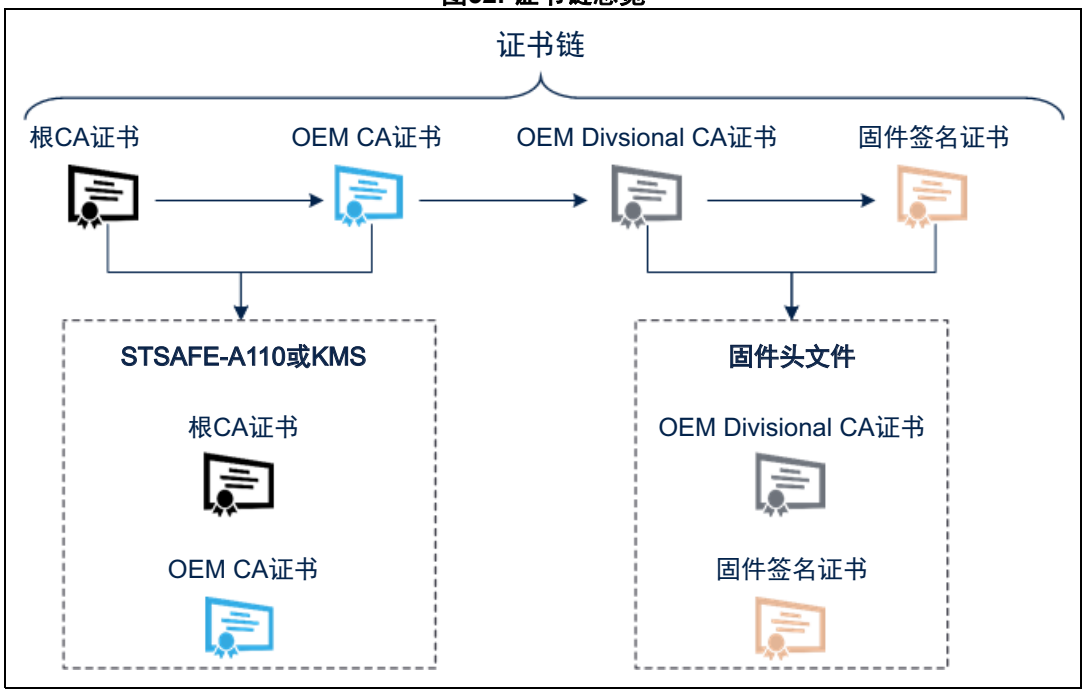



## F.3 使用KMS和X509加密方案

*图 52*所示加密方案基于四证书链原则：

1. 根CA证书：将在KMS嵌入式密钥中提供的根证书（如*第 F.2 节*所述）
2. OEM CA证书：将在KMS嵌入式密钥中提供的来自OEM的第一个中间证书（如*第 F.2 节*所述）
3. OEMDivisionalCA证书：来自OEM的第二个中间证书，将被插入每个新固件映像的头文件中
4. 固件签名证书：来自OEM的固件签名证书，将被插入每个新固件映像的头文件中

图52. 证书链总览




*注：* *关于证书生成的详细信息，请参考第 G.2 节。*

## F.4 UserApp菜单

添加的特定菜单提供了使用以下KMS服务导出服务（通过标准PKCS #11接口）的示例：AES-GCM/CBC加密/解密，RSA签名/验证，密钥配置，AES ECB密钥推导，ECDSA密钥对生成，以及ECDH Diffie-Hellman密钥推导。

图53. KMS菜单

# 附录G 使用STM32和STSAFE-A110时的SBSFU

## G.1 STSAFE-A110简介

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。

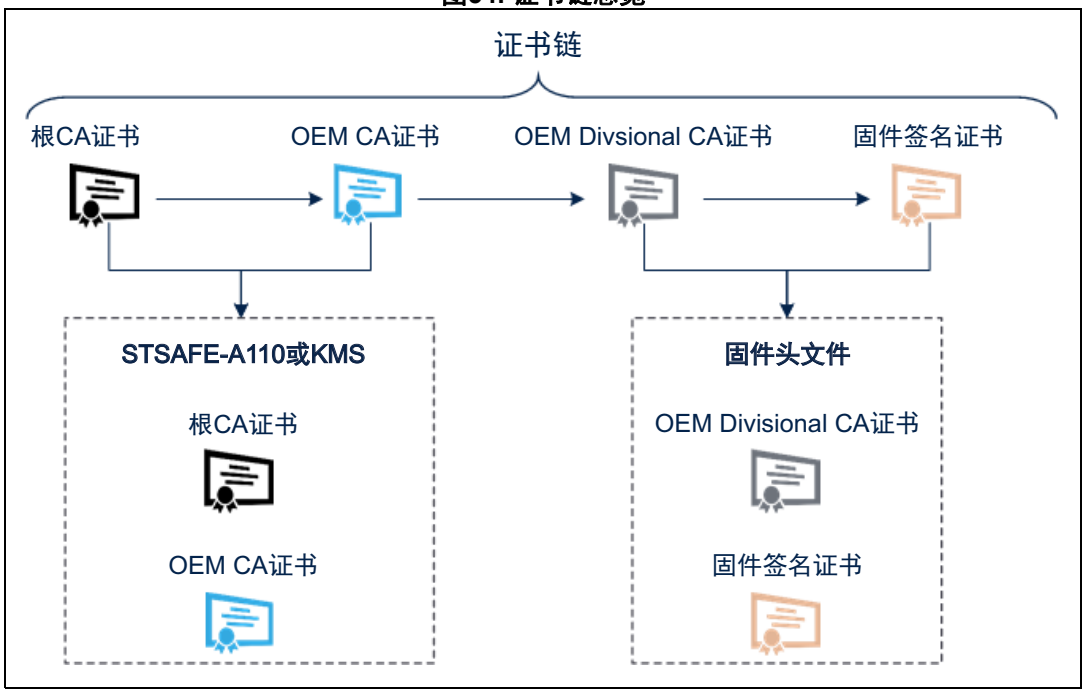
STSAFE-A110通过$I^2C$硬件接口连接到STM32。为了保护系统，必须在STSAFE-A110和STM32中提供配对密钥：

- Host_Mac_Key：对称密钥，用于将特定STM32与特定STSAFE-A110配对，以便使用CMAC计算（基于分组密码的MAC）在STM32和STSAFE-A110之间建立安全通信
- Host_Cipher_Key：对称密钥，用于加密STM32和STSAFE-A110之间的$I^2C$通信，以便建立安全的通信信道

为了结合STSAFE-A110和STM32以用于SBSFU应用，使用加密方案*基于X509证书的无固件加密的非对称方案*（参见*附录D: 加密方案处理*了解更多信息）。此加密方案基于四证书链原则：

- 根CA证书：将在STSAFE-A110中提供的根证书
- OEM CA证书：将在STSAFE-A110中提供的来自OEM的第一个中间证书
- OEMDivisionalCA证书：来自OEM的第二个中间证书，将被插入每个新固件映像的头文件中
- 固件签名证书：来自OEM的固件签名证书，将被插入每个新固件映像的头文件中

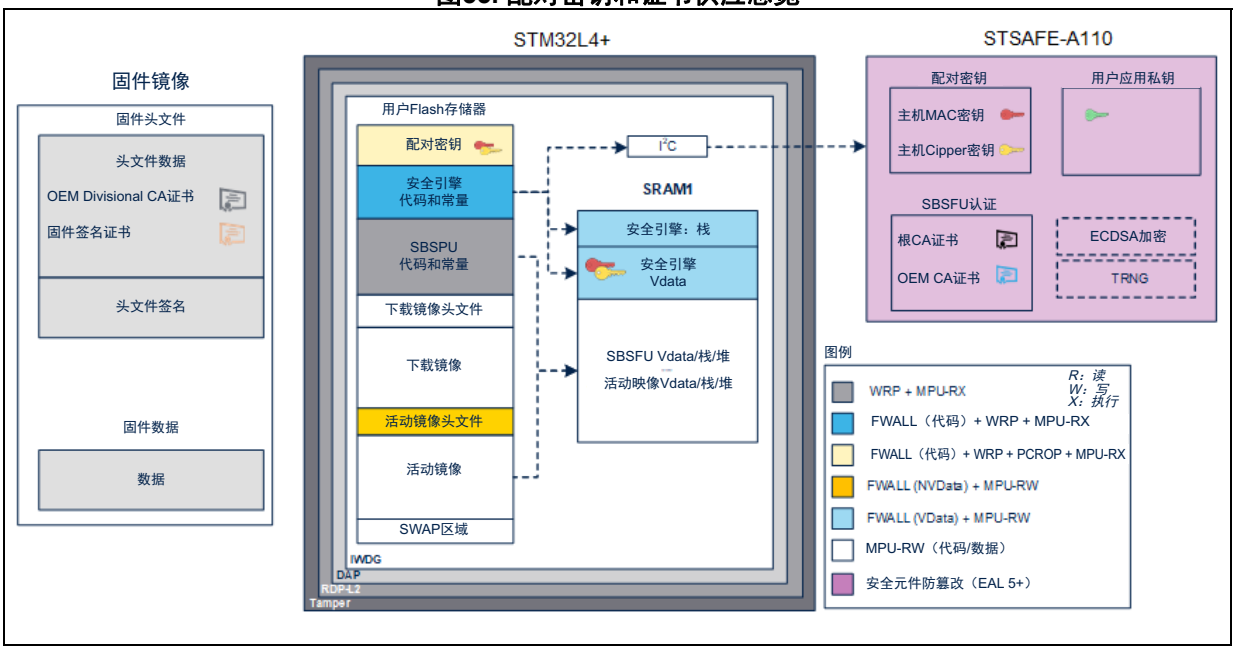
**图54. 证书链总览**



如上文所述，STM32、STSAFE-A110和固件映像必须与一些密钥和/或证书一起供应：

- STM32：为了能够与STSAFE-A110组件安全地通信，必须将配对密钥插入SBSFU应用代码（在受保护环境中执行的那部分）(a)。
- STSAFE-A110：
  - 为了能够与STM32组件安全地通信，必须在STSAFE-A110中提供配对密钥(a)。配对密钥以可执行代码的形式存储在项目*2_Images_STSAFE\2_Images_SECoreBin*中的文件*se_key.s*（SE_ReadKey部分）中。
  - 为了能够验证OEM Divisional CA和固件签名证书（作为要在STM32上安装的新固件映像的头文件的一部分接收到的），必须在STSAFE-A110中提供根CA证书和OEM CA证书(a)。
- 固件映像：必须将OEMDivisionalCA和固件签名证书插入要在STM32上安装的新固件映像的头文件中(a)。

**图55. 配对密钥和证书供应总览**



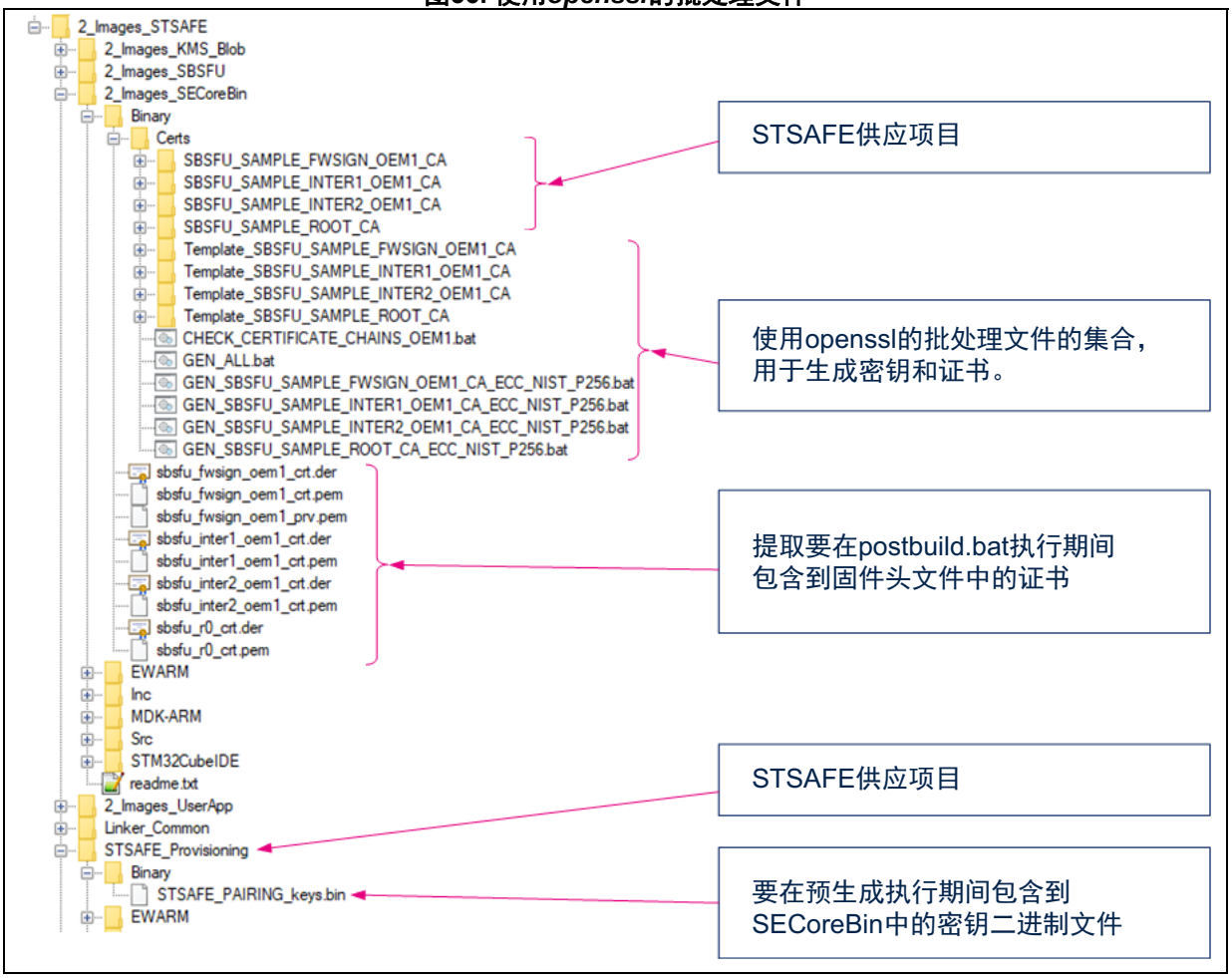

a. 提供了在STSAFE-A110内部提供密钥和证书的完整代码示例。项目名称为“STSAFE-A Provisioning”。更多信息见*第 G.3 节*。

## G.2 证书生成

使用*openssl*，通过STSAFE-A110项目示例变体中提供的一组批处理文件生成证书，如*图 56*所示：

- GEN_SBSFU_SAMPLE_ROOT_CA_ECC_NIST_P256.bat：生成根CA公钥、ECC私钥和自签名根证书。
- GEN_SBSFU_SAMPLE_INTER1_OEM1_CA_ECC_NIST_P256.bat：生成第一个中间CA（OEM CA）ECC密钥对和由RootCA签名的证书。
- GEN_SBSFU_SAMPLE_INTER2_OEM1_CA_ECC_NIST_P256.bat：生成第二个中间CA（OEM Divisional CA）ECC密钥对和由OEM CA签名的证书。
- GEN_SBSFU_SAMPLE_FWSIGN_OEM1_CA_ECC_NIST_P256.bat：生成固件签名ECC密钥对和由OEM Divisional CA签名的证书。

**图56. 使用*openssl*的批处理文件**



## G.3 STSAFE-A110供应

为了为STSAFE-A110供应SBSFU应用示例场景中使用的配对密钥和证书，X-CUBE-SBSFU包中以示例的形式提供了一个供应工具应用项目（*2_Images_STSAFE\STSAFE_Provisioning*）。

必须更新*STSAFE_PAIRING_keys.bin*，以便在STM32中提供相同密钥。

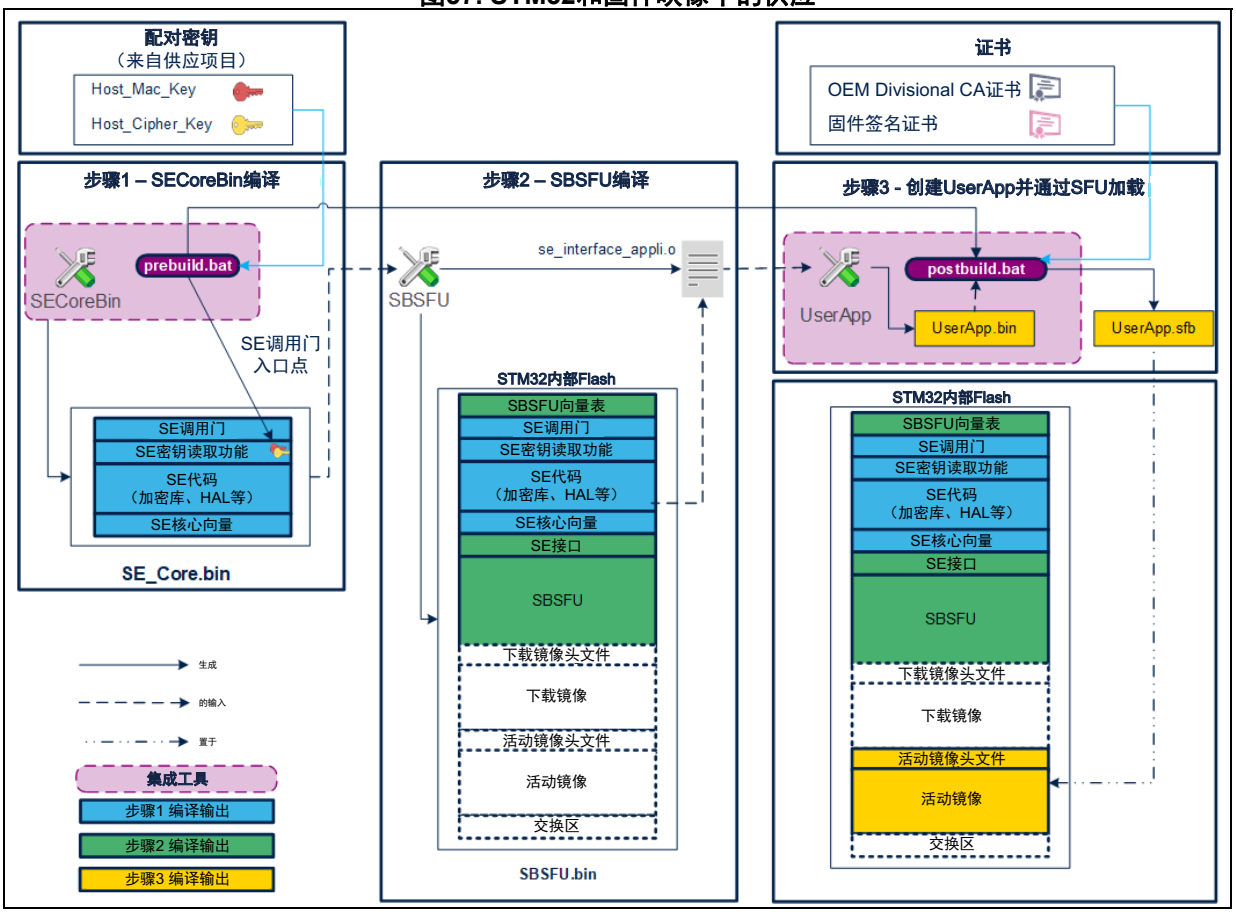
请参考该项目中的*readme*文件，以获取关于该项目执行的程序和步骤的详细信息。

## G.4 STM32和固件映像供应

为了给STM32提供配对密钥并将证书插入固件映像头文件，使用了X-CUBE-SBSFU扩展包的准备映像工具链（请参考*附录E: 固件映像准备工具*了解更多信息）：

- 使用IDE预生成脚本在SBSFU代码中插入配对密钥
- 使用IDE后期生成脚本在固件映像头文件中插入证书

**图57. STM32和固件映像中的供应**



## G.5 STSAFE-A110的订购

为了在SBSFU应用示例的背景下使用STSAFE-A110，必须在ST生产阶段用名为“Generic sample profile or SPL2”的配置文件对应的一些特定数据（如私钥）对STSAFE-A110进行个性化处理。应用笔记*STSAFE-A110通用示例配置文件说明*（AN5435）（https://www.st.com/en/secure-mcus/stsafe-a110.html#resource）中描述了此个性化配置文件。

B-L4S5I-IOT01A板（https://www.st.com/en/evaluation-tools/b-l4s5i-iot01a.html）已包含板上焊接的STSAFE-A110，并加载了AN5435所述的通用示例配置文件。

SPL02配置文件包含：

STSAFE-A110的订购代码（销售参考）：

- STSAFA110S8SPL02（SO8N封装）
- STSAFA110DFSPL02（UFDFPN8封装）。

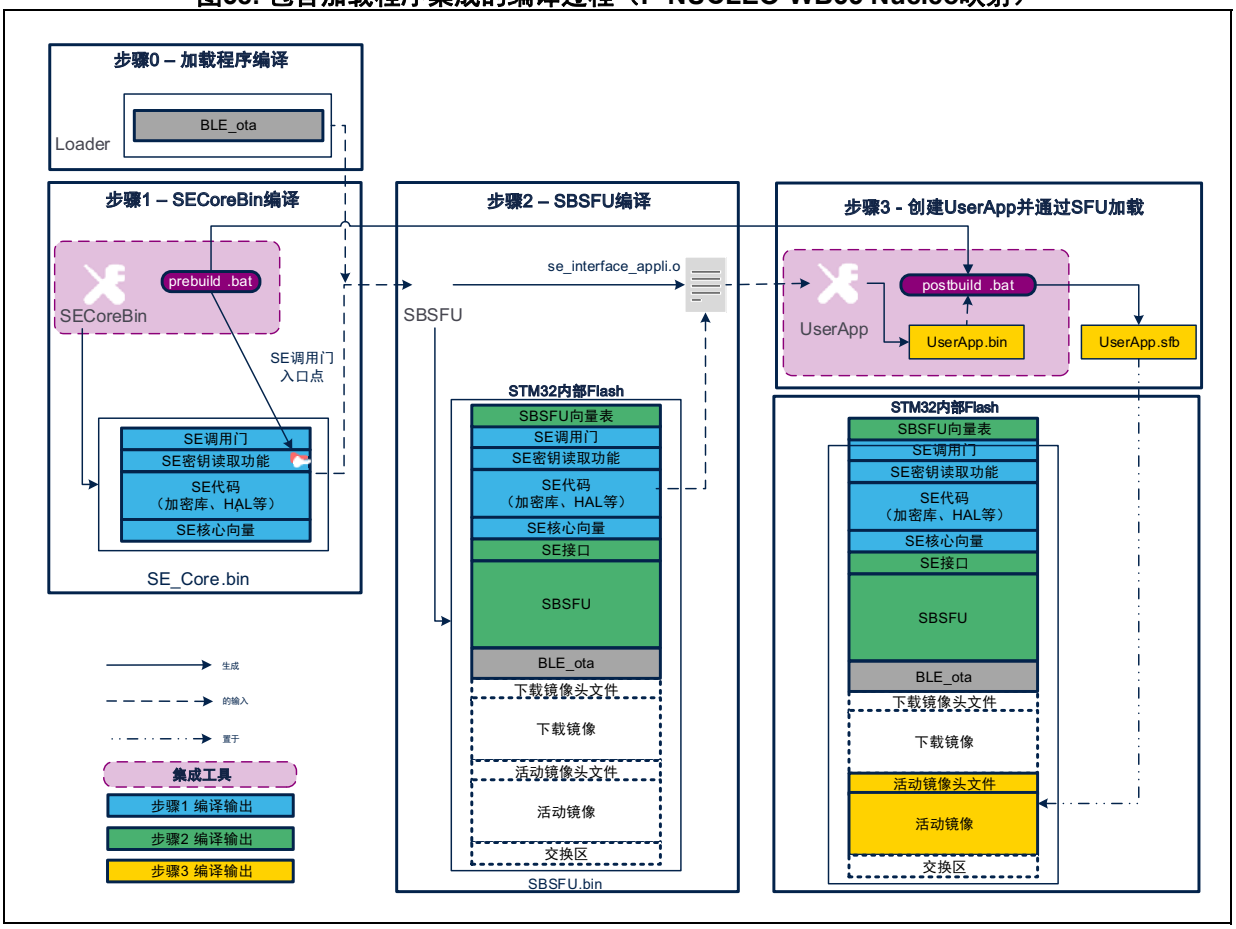
# 附录H STM32WB系列特殊性

## H.1 编译过程

提供的特定项目（BLE_Ota）实现了基于Bluetooth® Low Energy协议的固件下载特性。

*图 58*描述了额外的编译步骤0和在编译过程中将加载程序集成到SBSFU的情形。

**图58. 包含加载程序集成的编译过程（P-NUCLEO-WB55 Nucleo映射）**



## H.2 密钥配置

在首次执行SBSFU示例前，对AES加密函数使用的对称密钥必须供应到Cortex® M0+中。遵循SECoreBin *readme.txt*文件中的指令列表。

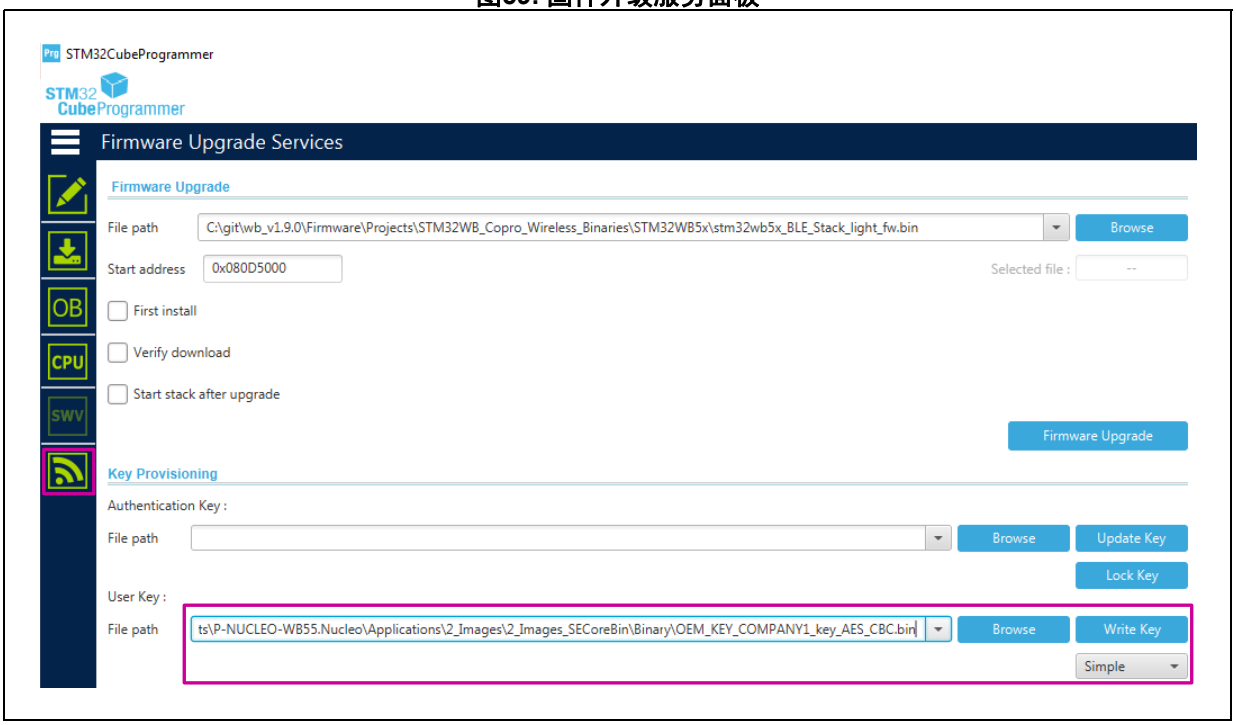
配置AES密钥的另一种方法为使用最新的STM32CubeProgrammer版本。自STM32CubeProgrammer V2.5.0起，Cortex-M0+密钥供应作为固件升级服务（FUS）提供。

首先，连接到bootloader程序USB接口：

- 勾选nBOOT1和nSWBOOT0
- 拉高BOOT0引脚来选择正确的启动模式：
  - 对于P-NUCLEO-WB55Nucleo板：CN7.5(VDD)与CN7.7(Boot0)之间的跳线接通。
  - 对于STM32WB5MM-DK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。
- 将USB线缆连接到USB_USER接口
- 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 59*中所示。

**图59. 固件升级服务面板**



## H.3 无线栈/FUS更新

由在CM0+上运行的FUS执行无线栈/FUS更新。在CM4上运行的SBSFU触发并控制此操作。

包含要更新的新无线栈/FUS的下载区必须位于内部Flash中，靠近安全Flash，原因有两个：

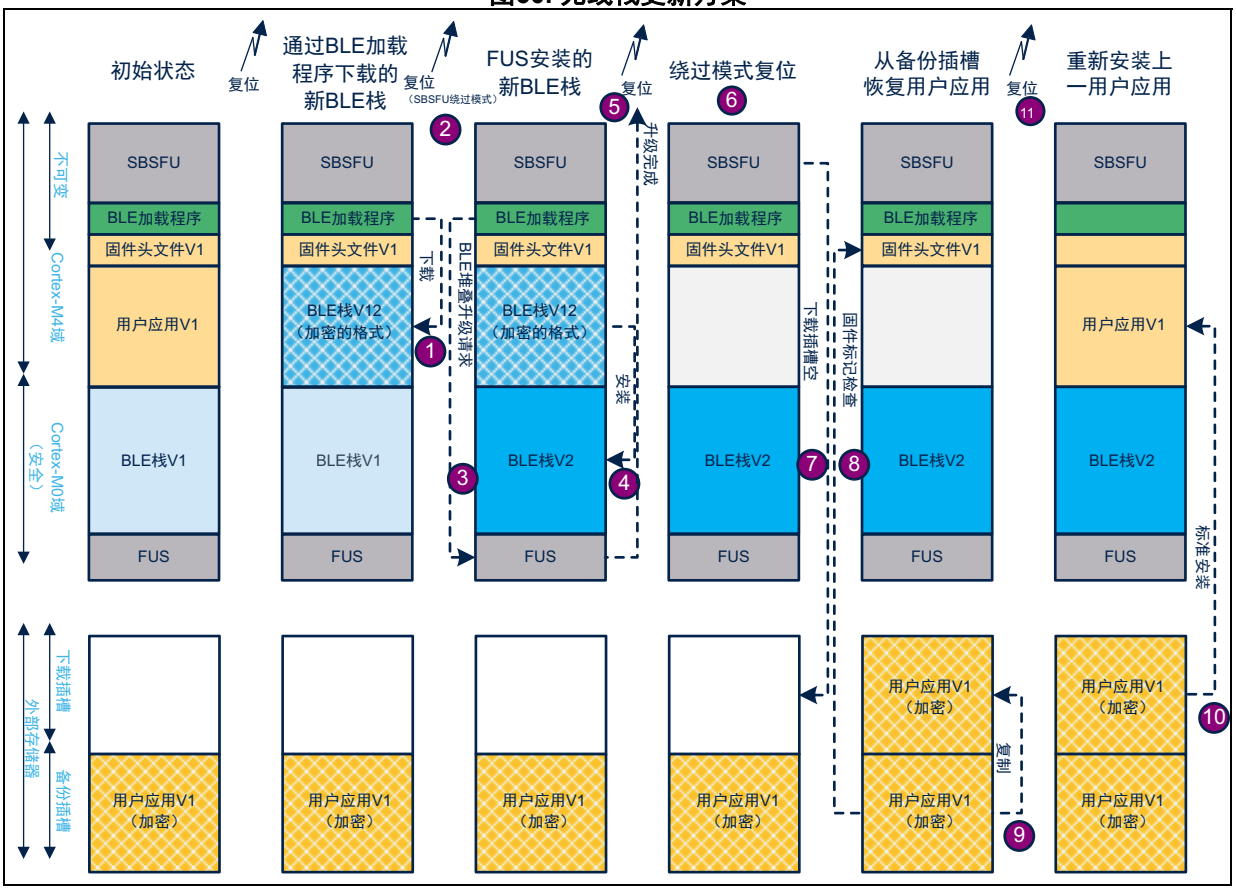
1. CM0+不能访问外部Flash。
2. 在更新过程结束时，CM0+会自动擦除下载区起始位置与安全Flash起始位置之间的区域。

根据需要的固件映像大小，下载插槽可能需要外部Flash。在这种情况下，也使用活动插槽下载和更新新的无线栈/FUS，并在外部Flash中配置额外备份插槽，以便在无线栈/FUS更新完成后重新安装固件映像。在*2_Images_ExtFlash*变体中提供了STM32WB5MM-DK板的此类配置示例。

*图 60*显示了无线栈更新方案：

1. BLE_Ota应用在内部Flash中下载新的无线栈，重写用户应用。
2. BLE_Ota应用设置*SBSFU bypass*模式并将设备复位。
3. 在*SBSFU bypass*模式下，SBSFU请求CM0+ FUS应用管理固件升级。
4. FUS应用检测内部Flash中的新CM0+固件，并安全地管理固件更新。
5. 完成后，FUS应用将设备复位。
6. 复位时，SBSFU检测到升级已完成，并重置*SBSFU bypass*模式。
7. SBSFU检测到下载插槽中无有效固件，并检测到活动插槽为空。
8. SBSFU检测备份插槽中的有效固件版本（与活动头文件相同的固件标记）。
9. SBSFU将备份插槽复制到下载插槽中，并复位设备以触发安装。
10. 继续执行标准固件安装过程。

图60. 无线栈更新方案


初始状态
复位
通过BLE加载程序下载的新BLE栈
复位
(SBSFU绕过模式)
FUS安装的新BLE栈
复位
绕过模式复位
从备份插槽恢复用户应用
复位
重新安装上一用户应用
不可变
Cortex-M4域
Cortex-M0域（安全）
SBSFU
BLE加载程序
固件头文件V1
用户应用V1
BLE栈V12（加密的格式）
BLE栈V1
BLE栈V2
FUS
下载
BLE堆叠升级请求
安装
升级完成
下载插槽清空
固件标记检查
标准安装
复制
下载插槽
备份插槽
外部存储器
用户应用V1（加密）
1
2
3
4
5
6
7
8
9
10
11

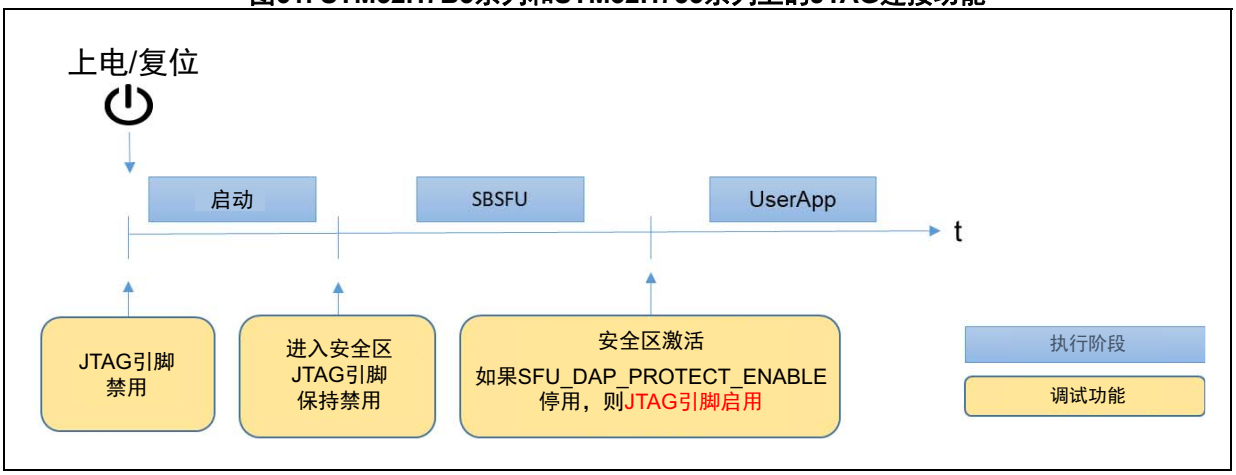
# 附录I STM32H7系列特殊性

## I.1 配置了安全存储器时的JTAG连接功能

如*图 61*所示，当配置了安全存储器时，在SBSFU执行过程中，JTAG引脚被禁用，无法连接STM32CubeProgrammer工具（STM32CubeProg）。在UserApp启动后，可以进行连接（必须在ST-LINK配置面板中选择热插拔模式）。

如果因SBSFU应用执行期间的任何潜在问题而不能执行或启动用户应用，则无法连接STM32CubeProgrammer工具（STM32CubeProg）。为了缓解此风险，仅在开发阶段后通过激活SFU_FINAL_SECURE_LOCK_ENABLE开关启用SFU_SECURE_USER_PROTECT_ENABLE开关。

**图61. STM32H7B3系列和STM32H753系列上的JTAG连接功能**

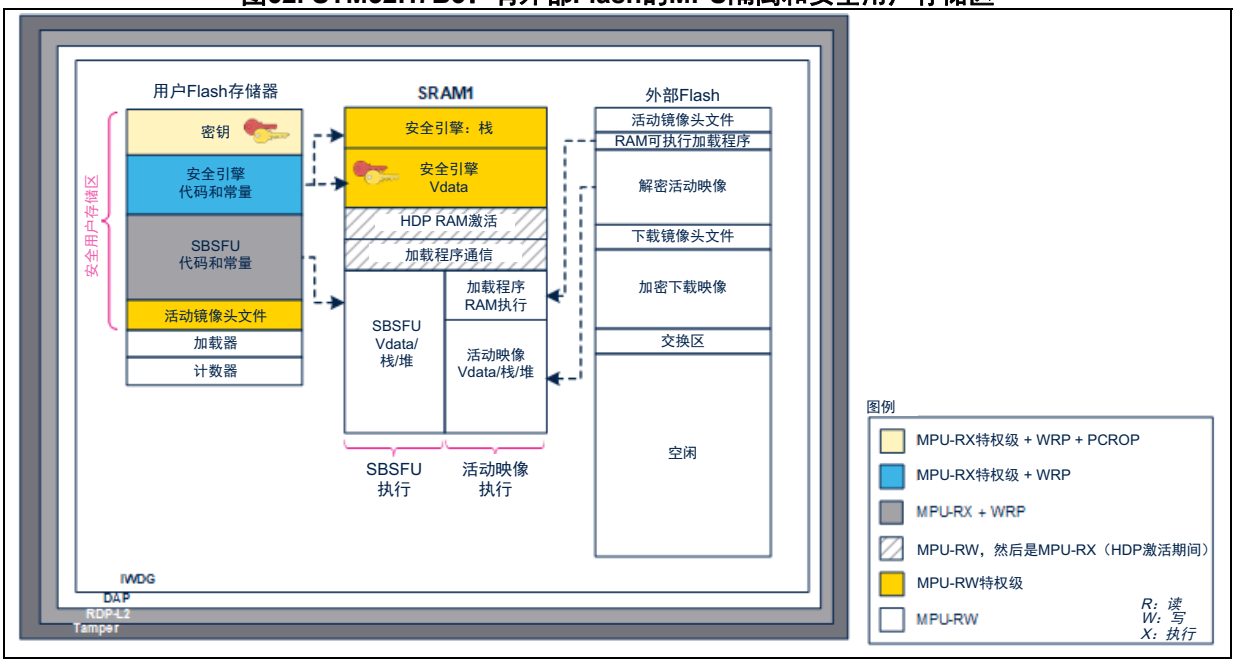



## I.2 STM32H7B3器件上的外部Flash

当用户应用需要的存储空间大于内部Flash的可用空间时，STM32H7B3xxx微控制器能够在外部存储器中存储固件映像。

在启动用户应用前，通过安全引擎受保护隔区配置OTFDEC只写密钥寄存器，如*图 62*所示。

**图62. STM32H7B3：有外部Flash的MPU隔离和安全用户存储区**




在安装过程中，为确保机密性，活动插槽固件保持加密状态。在从外部Flash执行用户应用的过程中，OTFDEC外设执行动态解密。

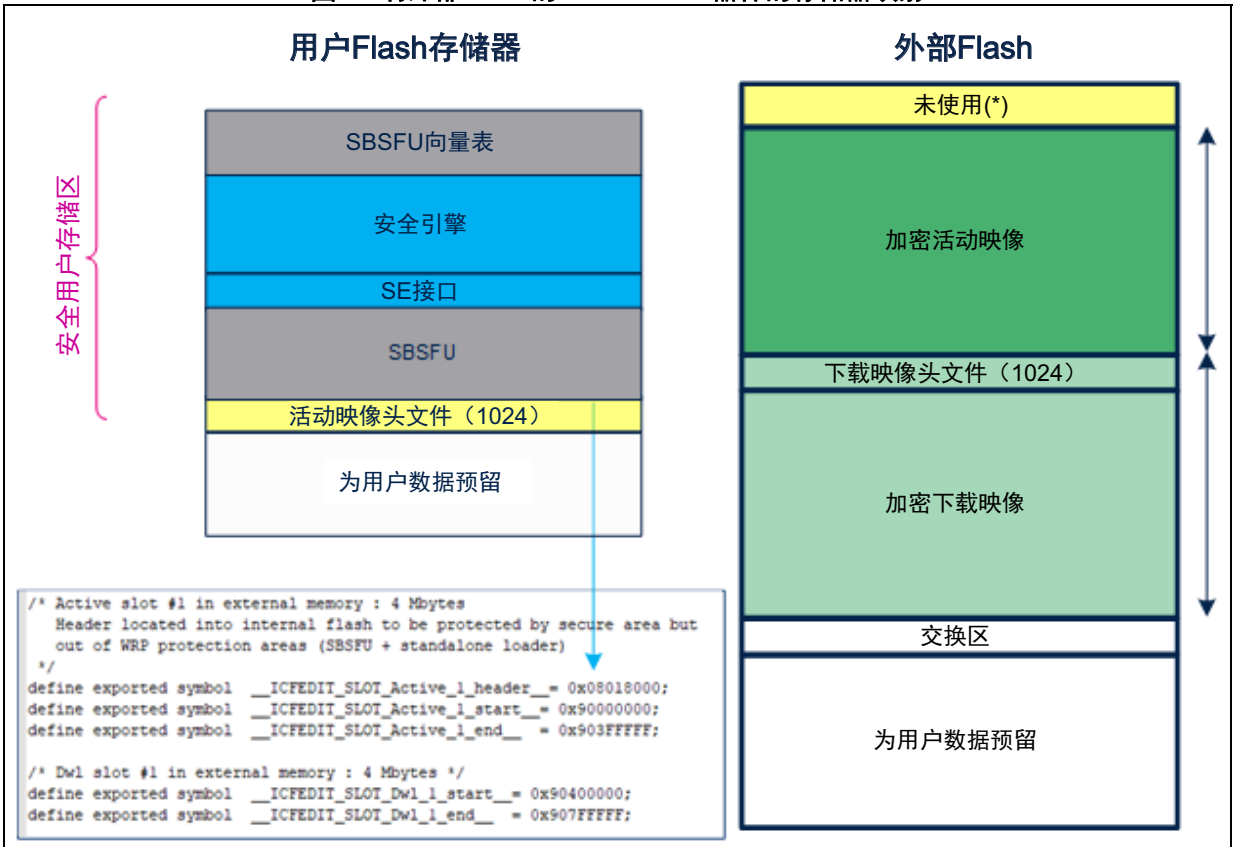
为了说明此配置需要的加密操作，提供了特定的加密方案（SECBOOT_ECCDSA_WITH_AES128_CTR_SHA256）：此加密方案使用AES-CTR加密和非对称（ECDSA）加密。

在具有外部Flash的此类配置中，在程序执行期间，可从内部Flash或从RAM执行加载程序功能，以便将下载的固件存储在外部Flash中。

为了提供与其他STM32微控制器相同的安全级别，在用户应用执行过程中，活动插槽的头文件必须不可访问。因此，活动插槽的头文件被存储在安全用户存储区保护的内部Flash中。

为了解释此功能，为X-CUBE-SBSFU扩展包的STM32H7B3I-DK板提供了名为*2 images ExtFlash*的应用。*图 63*显示了此类配置的典型存储器映射。

图63. 有外部Flash的STM32H7B3器件的存储器映射




*：为了维持所有应用（有或没有外部Flash和OTFDEC）之间的一致性，对活动插槽使用相同存储器结构（头文件和映像是相邻的）。对于STM32H7B3，不使用外部Flash中的头文件。

结果，在用户应用编译结束时生成了两个独立的二进制文件：

- 第一个二进制文件合并了SBSFU二进制文件和活动固件映像头文件（*SBSFU_UserApp_Header.bin*），将编程到内部Flash中。
- 第二个二进制文件（*UserApp.sfb*）将编程到外部存储器中的活动插槽起始地址。

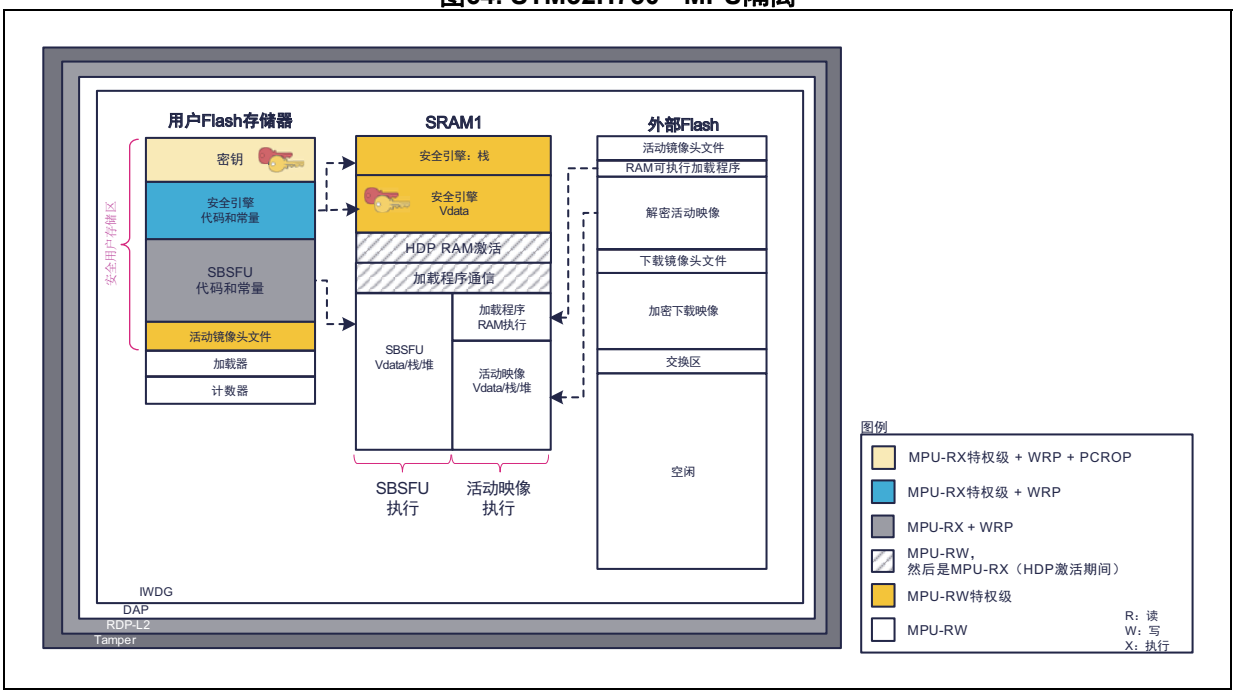
为了启动已安装了固件映像的SBSFU，必须烧写两个二进制文件。

## I.3 STM32H750B器件特殊性

STM32H750B器件在一个扇区中提供128KB的内部Flash。如*图 64*所示，活动插槽和下载插槽被映射到外部Flash中。因此，一些特性不可用：

- 由于用户应用在执行前进行了解密，因此无法确保固件机密性。
- 由于防回滚机制基于内部Flash中存储的计数器，因此不能对内部Flash设置WRP保护。在用户应用执行过程中，通过HDP保护保证SBSFU不可变代码的安全性，但在启动执行过程中无法保证。
- 用户应用中提供的加载程序功能必须从RAM执行。考虑到这个新应用，对编译过程进行了修改，如*图 65*所示。

**图64. STM32H750 - MPU隔离**

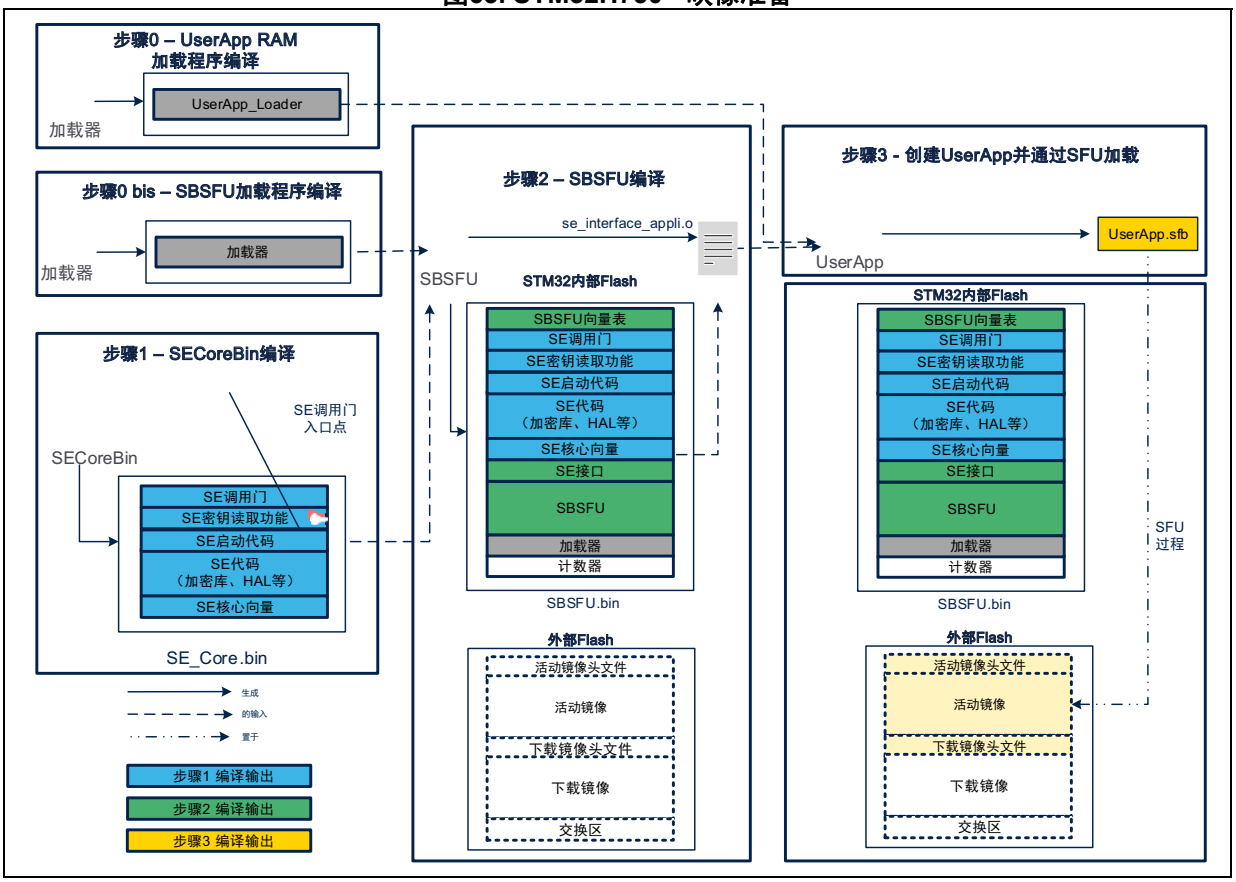



结果，在用户应用编译结束时生成了两个独立的二进制文件：

- 要编程到内部Flash中的SBSFU二进制文件
- 第二个二进制文件合并了活动固件映像头文件和用户应用二进制文件，将被编程到外部存储器中的活动插槽起始地址

为了启动已安装了固件映像的SBSFU，必须刷写两个二进制文件。

图65. STM32H750 - 映像准备



# 附录J 新固件映像的验证

在固件更新过程完成之前，用户可能需要通过执行自检来控制新映像的有效性。为了启用此机制，必须激活ENABLE_IMAGE_STATE_HANDLING开关。

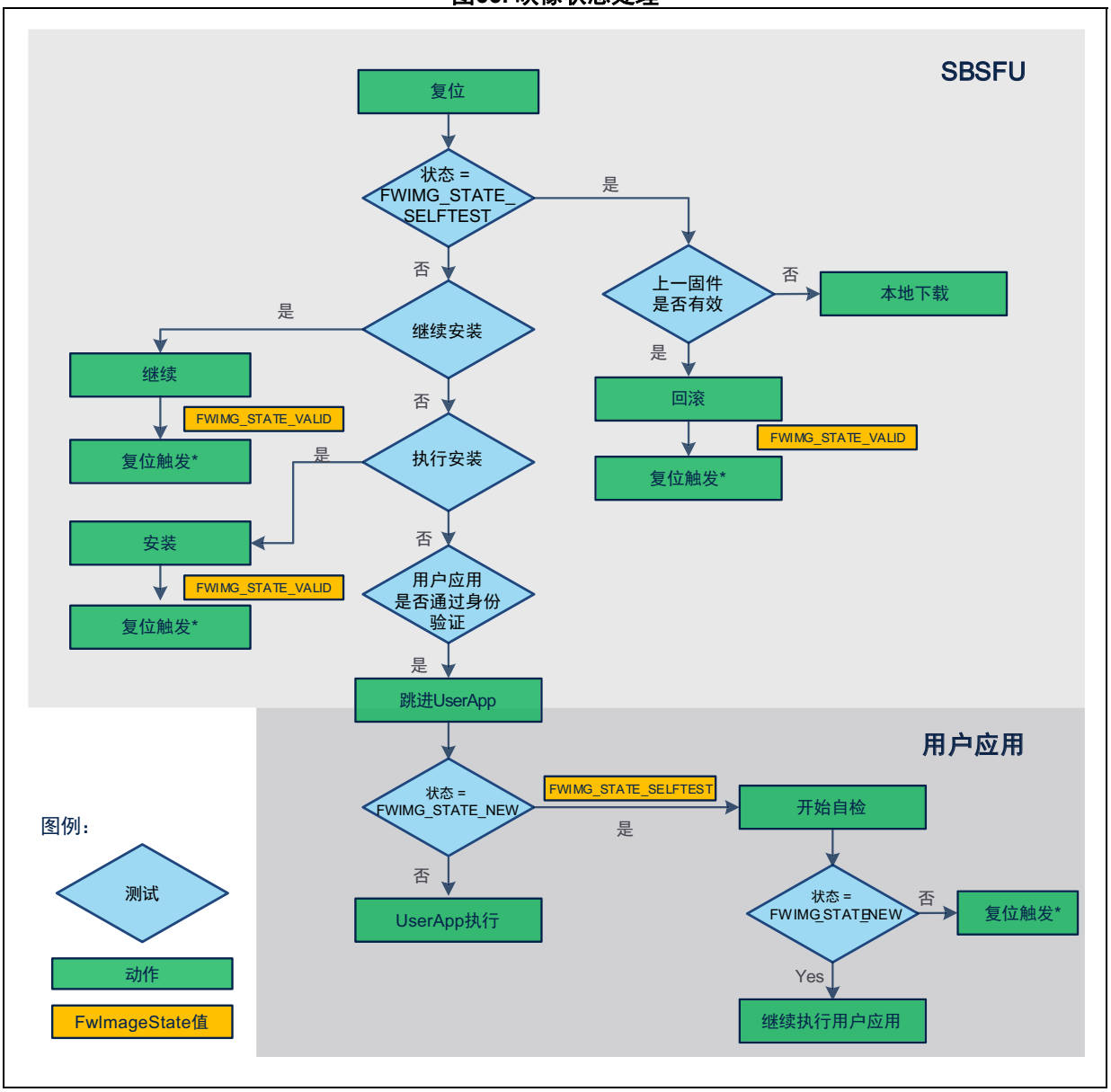
仅在选择互换安装过程的双插槽配置示例中才能激活此功能。

名为*FwImageState*的参数存储在映像的头文件中，用于指示固件映像的状态，并根据新映像的安装和验证过程进行更新。

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。

流程如*图 66*所示。

图66. 映像状态处理




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。

# 版本历史

表9. 文档版本历史

| 日期 | 版本 | 变更 |
|---|---|---|
| 2017年12月7日 | 1 | 初始版本。 |
| 2017年12月20日 | 2 | 删除了*第7章：了解启动时的最后执行状态消息*和*B.2 映射定义*中对集成指南的引用。更新了*表4：启动时的错误消息*和*5.2.2 节：编程STM32微控制器的软件工具*。 |
| 2018年4月20日 | 3 | 文档范围扩展到非对称和对称加密方案。添加了单映像模式。扩展了STM32L4系列支持的功能：<br>– 更新了所有章节<br>– 更新了*附录A 安全引擎保护的环境*和*附录B 双映像处理*<br>– 增加了*附录C 单映像处理*、*附录D 加密方案处理*和*附录E 固件映像准备工具*<br>– 删除了MSC附录 |
| 2018年12月18日 | 4 | 产品范围扩展到了STM32F4系列、STM32F7系列和STM32G0系列：<br>– 更新了*5节：保护措施和安全策略*、*第8章：逐步执行*和*B.1 节：元件和角色*<br>– 增加了*A.2 节：基于MPU的安全引擎隔离*<br>安全库提供扩展到mbedTLS：<br>– 更新了*6.2.3 节：加密库*和*6.3 节：文件夹结构* |
| 2019年7月22日 | 5 | 更新了整个文档：<br>– 产品范围扩展到了STM32G4系列、STM32H7系列、STM32L0系列、STM32L1系列和STM32WB系列<br>– 增加了STSAFE-A100的集成<br>– 增加了使用PKCS #11 API的安全密钥管理服务<br>– 增加了*附录FKMS*、*附录G使用STM32和STSAFE-A100时的SBSFU*、*附录H STM32WB系列特殊性*和*附录I STM32H7系列特殊性*<br>– 删除了*附录 SBSFU应用状态机* |
| 2020年2月4日 | 6 | 增加了支持OTFDEC处理的微控制器的外部FlashAES-CTR加密：<br>– 增加了*I.2：STM32H7B3器件的JTAG连接*和*I.3：STM32H7B3器件上的外部Flash*<br>– 更新了*E.3：输出*<br>– 更新了*图5：SBSFU安全IP与STM32系列的对比（2/2）*、*图14：项目文件夹结构（2/2）*、*图39：非对称验证和对称加密*和*图43：SBSFU双映像启动流程*<br>– 更新了*表3：加密方案比较*和*表8：加密方案列表* |

表9. 文档版本历史（续）

| 日期 | 版本 | 变更 |
|---|---|---|
| 2020年9月3日 | 7 | 更新了：<br>– 安全元件STSAFE-A110替代原来的STSAFE-A100。<br>增加了：<br>– *简介*和*概述*中的多映像支持（可配置最多三个活动映像）<br>– 映像状态处理：为每个活动映像增加了状态信息，用于指示安装流程的进度（*附录A*和*附录J*）<br>– 新板件B-L4S5I-IOT01A，具有安全元件STSAFE-A110（*STM32Cube总览*、*概述*、*6.2.3节：加密库*、*附录D*和*附录G*）<br>– 用于B-L475E-IOT01A的应用2 images External Flash（*6.2.7节：安全启动和安全固件升级（SBSFU）应用程序*）<br>– 防火墙以内代码执行期间的中断管理（*概述*和*附录A*）<br>– *8.6节：在安全特性激活时进行新软件的编程* |
| 2020年10月19日 | 8 | 更新了*5.3节*中的*可抵御外部攻击的保护措施* |
| 2021年6月22日 | 9 | 更新了：<br>– 将之前的图更新为更清晰的版本<br>增加了：<br>– *8.5.4节：多重下载*<br>– *8.5.5节：固件映像验证*<br>– *I.3节：STM32H750B器件特殊性*<br>– *图59：固件升级服务面板*<br>– *图62：STM32H7B3：有外部Flash的MPU隔离和安全用户存储区*<br>– *图64：STM32H750 - MPU隔离*<br>– *图65：STM32H750 - 映像准备* |
| 2021年12月14日 | 10 | 更新了：<br>– *第 2节：STM32Cube 概述*<br>– *图 4*、*图 5*、*图 11*、*图 14*、*图 18*、*图 55*和*图 56*<br>– *图 58*标题<br>增加了：<br>– 对STM32L4+系列的引用<br>– *图 60：无线栈更新方案*<br>– *第 H.3节：无线栈/FUS更新* |

表10. 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。