

# 使用SPSDK更新KW45 NBU I-操作介绍

- 1 前言
- 2 SPSDK介绍
- 3 SPSDK安装
- 4 SPSDK示例
- 5 使用SPSDK更新NBU
  - 5.1 生成OEM 密钥和证书
  - 5.2 使用OEM keys 生成安全二进制文件SB
  - 5.3 使用OEM keys 对KW45样片进行编程
  - 5.4 将新的image发送到KW45

# 1 前言



在之前的介绍中，我们已经了解到KW45和以往的KW系列的产品有个很大的不同，即Radio固件有一个专有的核心Cortex-M3和专有闪存空间，应用程序跑在另一个核心Cortex-M33上。

这就导致了，在KW45上使用所有的wireless工程，都有一个前置条件，即保证烧录了正确的Radio固件，我们称之为NBU。

烧录NBU是通过SPSDK完成了，这篇文档是之前的延续，主要介绍如何使用SPSDK完成NBU更新。使用Jupyter Notebook作为脚本来完成安全更新NBU的一系列操作。

本文介绍的步骤如下：

1. 生成密钥Keys和证书Certificate；
2. 使用自定义生成的密钥keys烧录KW45 fuse；
3. 生成安全二进制文件secure binary，即SB3文件；
4. 通过 KW45 bootloader 将加密image NBU编程到radio闪存中；

## 2 SPSDK介绍

完整的操作所需要的软件功能，全部由SPSDK 提供。

- NOTE: KW45 EVK fuse 预烧录了 [generic keys](#)，便于开发。仍可使用 EVK 遵循本文档，但无法对其fuse进行编程。

本文档中提供的某些脚本会执行无法逆转的破坏性操作（烧fuse）。

SPSDK是一个统一、可靠、易用的 Python SDK 库，适用于恩智浦 MCU 产品组合。SPSDK加速了客户从原型开发到生产部署。允许用户与设备连接和通信，配置设备，准备、下载和上传数据，包括安全操作。

SPSDK结构：

- 应用程序接口（API），Python 库形式的对外接口。
- 应用程序，可使用 Python 虚拟环境从命令行调用。

下面列出了一些 SPSDK 应用。有关 SPSDK 支持的应用程序的完整列表，可以在[link](#)找到。

- [nxpimage](#)
  - 生成/解析 AHAB image
  - 生成 TrustZone image
  - 生成 MasterBootImage image
  - **生成 SecureBinary image;**
  - 生成 custom binaries;
- [nxpcrypto](#)
  - 生成具有各种密钥属性的 RSA/ECC 密钥对（私钥和公钥）
  - 验证密钥对
  - 转换密钥文件格式（PEM/DER/RAW）
  - 生成/验证 x509 证书
  - 生成/验证 hash digests

- `nxpdevscan` : 列出所有已连接的 USB 和 UART NXP 设备
- `blhost` : 用于与恩智浦设备上的 MCU bootloader通信的程序。它允许用户:
  - 根据内存 ID 擦除所有 flash/sections
  - 用 pattern 填充内存
  - 获取/设置 bootloader 特定属性
  - 写入/读取内存
  - 接收 SB 文件
  - 向设备加载 boot 映像
  - 密钥配置
  - 在地址处执行应用程序
  - 读取 flash 模块资源
  - 编程/读取 fuse
  - 列出所有内存
  - 执行可靠更新
- 其他

### 3 SPSDK 安装

请参考上一篇文章档，这里不再详细介绍。

### 4 SPSDK 示例

可以通过 SPSDK 中提供的 API，直接在命令行 CMD 中使用 SPSDK，也可以通过使用 Python 虚拟环境或 Jupyter Notebook，可视化运行 SPSDK 功能；在 SPSDK 的安装文件中提供了 API 和应用程序的示例。有关 Python 脚本示例，请查看本地安装文件中的示例文件夹：

`C:\nxp\spsdk\examples`

SPSDK 1.9.0 中的一些 SPSDK Python 脚本示例包括：

- crypto - 用于证书和密钥管理的示例

- dat - 用于调试凭证管理的示例
- image.py - 创建一个简单的可引导映像
- image\_dcd.py - 创建一个带有 DCD 数据的简单可引导映像
- image\_srk.py - 从证书创建fuse (SRK) 文件
- lpc55xx\_tz\_pfr.py - 为 LPC55xx 创建自定义TrustZone和受保护闪存区域数据
- mboot.py - 从目标的bootloader读取属性
- sbfile.py - **创建安全启动 (SB) 映像**
- sdp.py - 使用 SDP 读取内存
- sdp\_mboot.py - 将闪存下载到 i.MX RT10xx 设备并读取bootloader属性
- sdps.py - 使用 SDPS 写入内存

目前还没有 KW45 的 Python 脚本示例。

Jupyter Notebook 示例作为交互式文档，提供可视化运行。如果对于 jupyter 环境不了解，可以参考：<https://docs.jupyter.org/en/latest/start/index.html>。

KW45 Jupyter Notebook 示例位于 jupyter\_examples 中：

`C:\nxp\spsdk\examples\jupyter_examples`

## 5 使用SPSDK更新NBU

更新 KW45 Radio固件的某些预设任务是破坏性操作，无法逆转，例如烧毁样片fuse。密钥和证书的管理也很重要。如果 KW45 样片的fuse是用一组确定的密钥证书写入的，则密钥证书文件必须妥善保存。当密钥-证书丢失或被覆盖时，已编程的 KW45 样片的 NBU 将无法继续更新，因为需要这些文件来生成新的安全二进制文件。

更新 KW45 Radio固件之前，请查看 KW45 安全 bootloader 的基本知识：

- KW45 Security Reference Manual (document KW45SRM) -- Section 8 ROM bootloader
- KW45 Reference Manual (document KW45RM) -- Section 15 ROM bootloader

在这里，我们通过 ISP 更新Radio FW。KW45 ROM Bootloader 提供系统内编程（ISP）实用程序，可通过串口连接在 MCU 上运行。它能在整个产品生命周期（包括应用开发、最终产品制造等）内快速、轻松地对 MCU 进行编程。

Bllhost 是与 KW45 bootloader通信的PC端命令行工具。SPSDK 中包含 Bllhost。用户可以使用该工具上传/下载应用代码。

当 ROM bootloader 进入 ISP 模式时，它会自动检测 LPI2C/LPSPI/LPUART 或 CAN 接口上的活动。一旦收到正确形成的帧，它就会选择相应的接口。如果接收到无效帧，则丢弃数据并恢复扫描。

要制作 KW45 ROM bootloader，请进入 ISP，请按下 [BOOT\\_CONFIG](#) 引脚（KW45-EVK 中的 PTA4、SW4）并短接 [JP25](#) 的引脚 [2和3]（KW45 EVK）。

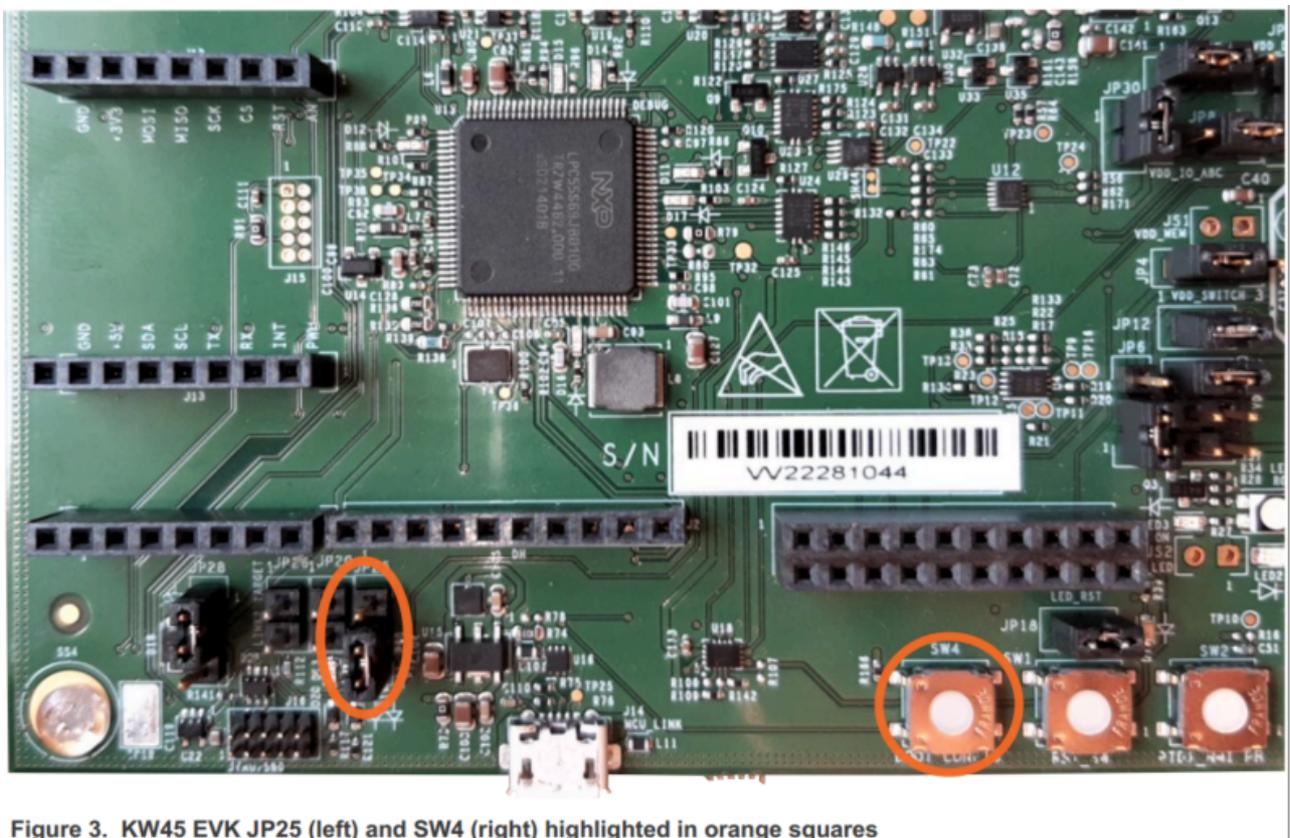


Figure 3. KW45 EVK JP25 (left) and SW4 (right) highlighted in orange squares

Bllhost [receive-sb-file](#) 命令用于通过 ISP 更新 CM33 闪存或radio (CM3) 闪存上的映像。通过该命令，KW45 设备接收安全二进制 (SB) 文件，解密、验证并将映像编入目标存储器。

通过 ISP 向 KW45 发送安全二进制文件以更新Radio FW 需要一系列步骤。完整的操作步骤，请参考 [AN\\_SPSDK.zip](#)。

## 5.1 生成OEM 密钥和证书

1. 使用 SPSDK `npxcrypto` 应用程序创建第一个Root of Trust Keys (`RoTKs`) 和可选的 Image Signing Certificate (`ISK`)。
2. 之后，用户应该修改证书配置文件，以生成包含公钥为私钥的自签名x509证书。
3. 然后，还要创建一个随机的 SB3 密钥衍生密钥 (`SB3KDK`)。
4. 由 RoTKs 和 SB3KDK 生成的信任根密钥表哈希 (`RoTKTH`) 将在下一步的 KW45 样片 fuse 中编程。

这些相同的证书和密钥将用于生成安全二进制文件，并发送给 KW45 以更新其 NBU。

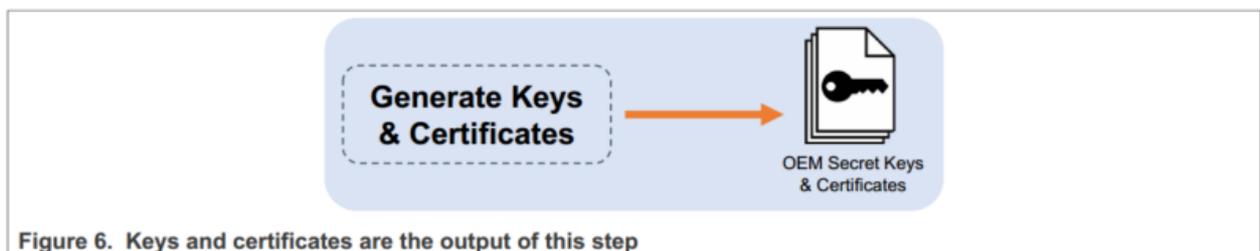


Figure 6. Keys and certificates are the output of this step

NOTE: 通过使用相同的脚本生成的文件会存放在同一个文件夹中，多次操作可能会覆盖和丢失预先存在的文件。所以，一定要注意，这个步骤执行完要保存好文件!!! 否则可能造成之前的Key丢失无法找回。

要使用SPSDK生成密钥和证书，使用Jupyter打开第一个笔记本并执行每个单元格。

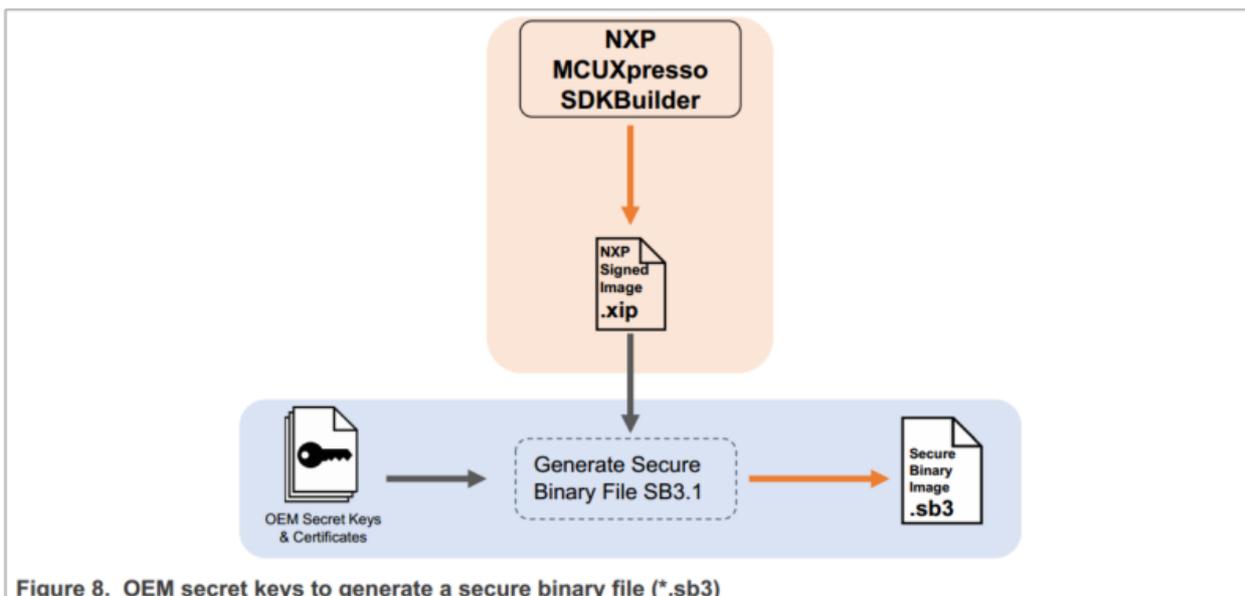
根据需要修改 `*.yaml` 配置文件。文件在新的文件夹workspace中生成。

## 5.2 使用OEM keys 生成安全二进制文件SB

注意：KW45 EVK fuse 已用通用keys预编程，以便在开发过程中使用。您仍可使用 EVK 遵循本指南，但无法对其fuse进行编程。对于 EVK，应使用NXP SDK Keys来生成安全二进制文件SB3。

安全二进制文件的生成始于 OEM 产生并控制的密钥和证书。这些密钥和证书在 SB 配置文件中用于加密映像。另一个必需组件是NXP release的不同版本SDK中提供的不同签名 image。

KW45 ROM bootloader 会检查image中的恩智浦签名，因为所有 KW45 样片都预编程了恩智浦认证密钥。这保证了 KW45 Radio 只能运行源自恩智浦的image。输出是一个安全的二进制文件 (\*.sb3)，可随时发送到 OEM KW45。



要使用SPSDK生成安全的二进制文件，请使用Jupyter打开第三个笔记本并执行每个单元。

这些文件在workspace文件夹中生成，文件名为“sb3.sb3”。

### 5.3 使用OEM keys 对KW45样片进行编程

恩智浦在OEM-open 生命周期中提供KW45样片，因此必须写入一些fuse才能使样片可用。

在此步骤中，使用之前生成的密钥对两个KW45 fuse进行编程：

- `CUST_PROD_OEMFW_ENC_SK`: 256位加密密钥，用于保护OEM固件的机密性。通常需要使用 `sb3` (`SB3KDK` ---SB3 key派生key)进行固件更新。
- `CUST_PROD_OEMFW_AUTH_PUK`: 256位 `RoTKTH`，通常用于 `CM-33` 主闪存image认证。

在此操作之后，已编程的KW45样片将永久链接到写入其fuse中的密钥，即链接到上一步中生成的密钥和证书。一旦被编程，KW45只能用安全的二进制文件更新，这些二进制文件是用同一组密钥和证书生成的。

要使用上一节中生成的密钥对KW45样片进行永久编程，请使用Jupyter打开第二个笔记本。然后，在修改最后一个单元格并将所需的keys添加到图12中突出显示的命令之后，执行每个单元格。

Note: KW45 EVK fuse 用图12中突出显示的通用key预先编程。检查以 `#example line` 开始的注释行：

### Program device fuses with keys/RoTKTH generated in previous steps

To program fuses blhost tool is used. Device needs to be in ISP mode, where it can communicate with blhost and process blhost commands. To serve the purpose of this document, ISP communication only over UART peripheral is considered for scripts. Also, accurate COMx port must be used.

**WARNING!!! This step is destructive operation (burning fuses), be sure that you set values of SB3KDK and RoTKH correctly in script as printed in output from nxpimage**

```
In [6]: # Increase voltage for fuse burning
%! blhost $UART_CONNECTION set-property 0x16 1

# program SB3KDK (CUST_PROD_OEMFW_ENC_SK)
# put value SB3KDK generated by nxpimage
#! blhost $UART_CONNECTION fuse-program 0x20 ["Substitute this comment by the SB3KDK generated key output in section SB3.1 gene
# example line : %! blhost $UART_CONNECTION fuse-program 0x20 [[7aa7ef9813b3561257b8837dab26225301df3511217f2733c71dadcd447722d1

# program RoTKTH (CUST_PROD_OEMFW_AUTH_PUK)
# put value RoTKTH generated by nxpimage
#! blhost $UART_CONNECTION fuse-program 0x1F ["Substitute this comment by the RoTKTH generated key output in section SB3.1 gene
# example line : %! blhost $UART_CONNECTION fuse-program 0x1F [[650d8097079ff27a3e8a2da14781b922fd8295b6c00bfa067f00e87f1a16b8b3

# Program TZM_EN fuse, this fuse was missed during manufacturing of first KW45 samples. Without TZM_EN fuse set, the S3MUA semaph
%! blhost $UART_CONNECTION fuse-program 0xD [[01]]

# Set voltage to normal value
%! blhost $UART_CONNECTION set-property 0x16 0
```

Figure 12. The highlighted commands must be updated with keys from step 2

### SB3.1 generation

We have created certificates and keys required for the creation of SB3.1 file. Let's create a SB3.1.

```
In [7]: %! nxpimage $VERBOSITY sb31 export $SB31_TEMPLATE_PATH
assert _exit_code == 0
assert os.path.exists(WORKSPACE+SB31_FILE_PATH)

nxpimage sb31 export workspace/sb31 config.vml
SB3KDK: 1d5a43bc0adb4cf6c1e6c642ea5b8cb2fa4f1297017fc94d703f00f07dc7e41f
RoTKTH: 9cafdb4417941784fd0754b08238bae5793ab8074a6f9df7b93114d01102434a356fabb761bd303773c6c6880895a643
Success. (Secure binary 3.1: C:/Users/ )
```

Figure 13. Keys generated in step 2

## 5.4 将新的image发送到KW45

这最后一步是通过Blhost `receive-sb-file` 命令完成的。



Figure 14. In this step, the generated \*.sb3 file is programmed in KW45

要将生成的安全二进制文件发送到KW45 EVK，请使用Jupyter打开第四个笔记本。按照笔记本中的描述，在ISP模式下初始化板子，并执行每个cell。 `blhost receive-sb-file` 命令需要几秒钟才能完成。

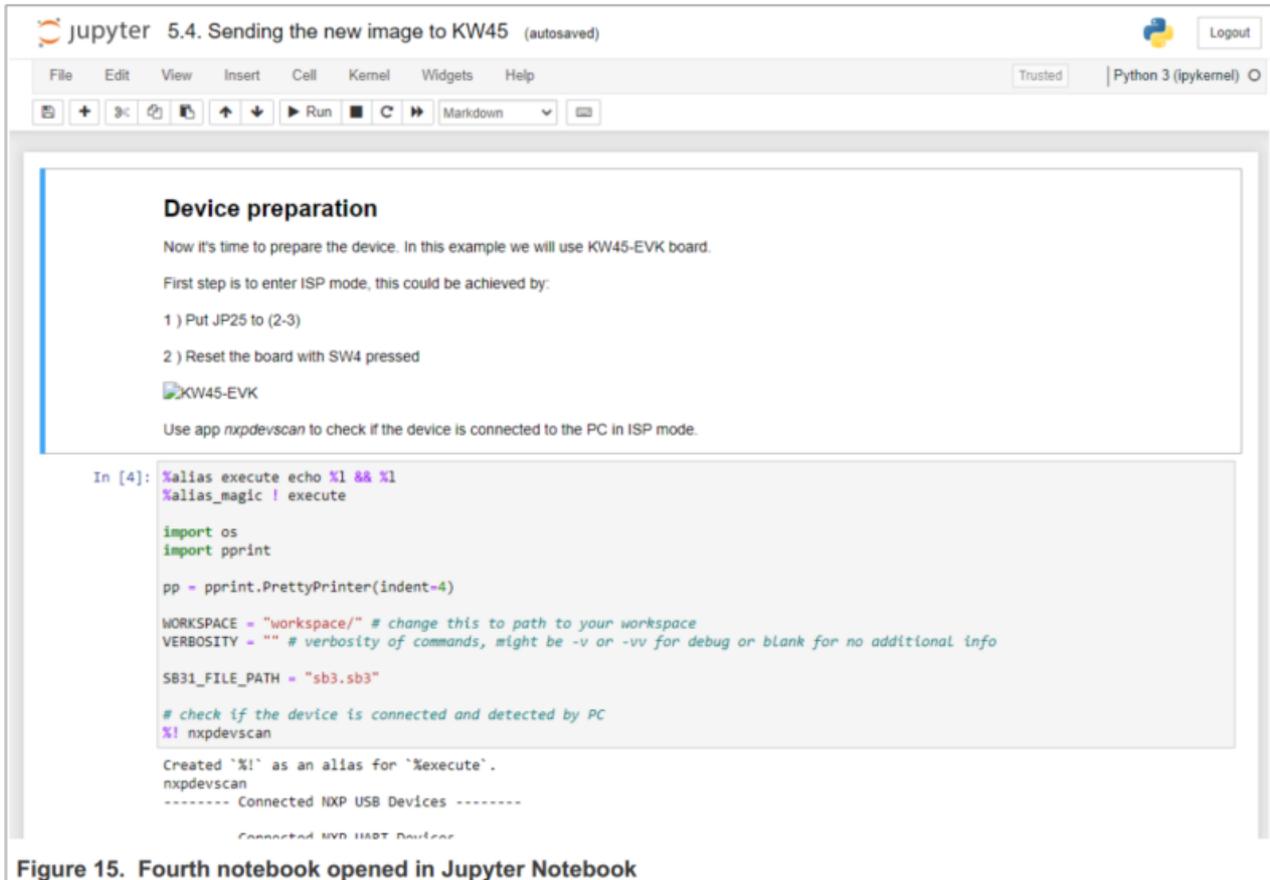


Figure 15. Fourth notebook opened in Jupyter Notebook

要使用恩智浦提供的新releases更新NXP KW45 radio，步骤5.2和步骤5.4必须用新的\*.xip文件执行。