XR806 安全启动方案

开发指南

版本号：1.0

发布时间：2021-02-25

版本历史

版本	日期	责任人	版本描述
1.0	2021-02-25	AWA 1680	创建文档。

目录

图片目录

1 前言

1.1 文档简介

本文档介绍了 XR806 平台上安全启动方案的开发与使用方法。

1.2 目标读者

使用 XR806 SDK 的开发人员。

1.3 适用范围

此文档适用于 XR806 SDK，支持 XR806 系列芯片产品。

1.4 文档约定

1.4.1 标志说明

本文档采用各种醒目的标志来表示在操作过程中应该特别注意的地方，这些标志的含义如下：

标识	说明
警告	该标志后的说明应给予格外关注，如果不遵守，可能会导致人员受伤或死亡。
注意	提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。
说明	为准确理解文中指令、正确实施操作而提供的补充或强调信息。
窍门	一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

符号	例子	说明
0x	0x0200，0x79	地址或数据以 16 进制表示。
0b	0b010，0b00 000 111	数据采用二进制表示(寄存器描述除外)。
X	00X，XX1	数据描述中，X 代表 0 或 1。 例如，00X 代表 000 或 001；XX1 代表 001，011，101 或 111。

1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

类型	符号	对应数值
数据容量（如 NAND 容量）	1 K	1024
	1 M	1 048 576
	1 G	1 073 741 824
频率，数据速率等	1 k	1000
	1 M	1 000 000
	1 G	1 000 000 000

2 概述

2.1 背景说明

安全启动（Secure Boot）是软硬件相结合的安全保护机制，其目的是保证固件的合法性和完整性。所有合法固件均需要经过唯一的密钥签名，否则，都是非法固件。在支持安全启动的设备中，只有合法固件才可以运行。

XR806 的安全启动方案需要 Bootloader 的支持。支持安全启动和不支持安全启动的 Bootloader 不能相互替换，即：

1. 安全启动方案，必须使用支持安全启动的 Bootloader；
2. 非安全启动方案，必须使用不支持安全启动的 Bootloader。

2.2 规格特性

XR806 的安全启动方案支持以下功能特点：

- 支持 eFuse 存储 OEM Public Key SHA256 哈希值。
- 支持安全启动使能标志，在 eFuse 中提供。
- 支持椭圆曲线加密算法 ECC256 prime256v1，公钥和私钥由用户生成并保管。
- 支持镜像签名（对 bin 文件签名）、打包、验证，建立完整的安全信任链，保证固件合法性。

2.3 文件位置

以 SDK/project/sign_script 为根目录，本技术方案涉及到的主要文件位置如下。

```
.
└── project
    ├── sign_script
    │   ├── ecc_private_key.pem
    │   ├── ecc_public_key.pem
    │   ├── ecc_public_key.pem.der
    │   ├── gen_signature.sh
    │   ├── pk_sample.bin
    │   └── sha256.txt
...
    ├── bootloader
    │   └── main.c
```

关键文件说明如下。

表 2-1 安全启动方案文件说明

文件名	文件说明
ecc_private_key.pem	椭圆曲线加密算法生成的私钥文件，pem 格式
ecc_public_key.pem	椭圆曲线加密算法生成的公钥文件，pem 格式
ecc_public_key.pem.der	椭圆曲线加密算法生成的公钥证书，der 格式
gen_signature.sh	签名脚本，用于生成密钥文件和对输入的 bin 文件进行签名
pk_sample.bin	提取后的公钥文件（64-byte 公钥值）
sha256.txt	公钥文件的 Hash 值（pk_sample.bin 文件的 Hash 值）
main.c	Bootloader 源码，包含本方案中对所有 app bin 的签名验证过程

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

3 技术说明

3.1 数字签名验证典型流程

安全启动一般采用数字签名来保护数据的合法性。典型的数字签名包括：

1．密钥：一般采用非对称密钥，包括公钥和私钥。

2．签名：采用私钥对给定数据进行加密来生成签名。

3．签名验证：采用公钥对加密过的签名进行解密验证。

私钥的保密性是安全启动机制的保障。如果私钥泄漏，非法者即可采用私钥生成合法固件；如果私钥丢失，则无法生成合法固件。

数字签名一般包括以下步骤：

1．计算给定数据的摘要（哈希值）D1

2．采用私钥对摘要进行加密，生成 D1E

3．将 D1E 存储到指定区域

签名验证一般包括以下步骤：

1．计算给定数据的摘要（哈希值）D1

2．从指定区域获取加密后的摘要 D1E，采用公钥对 D1E 进行解密，生成摘要 D2

3．比较 D1、D2，若相等则验证通过，否则验证失败

摘要（哈希值）的使用保证了数据的完整性校验；非对称加解密则保证了数据的合法性。

3.2 XR806 签名验证流程

XR806 安全启动方案的关键要素包括：

1．eFuse：特定区域用于存储 OEM Public Key SHA256 哈希值，并且有标记位表明安全启动是否使能。

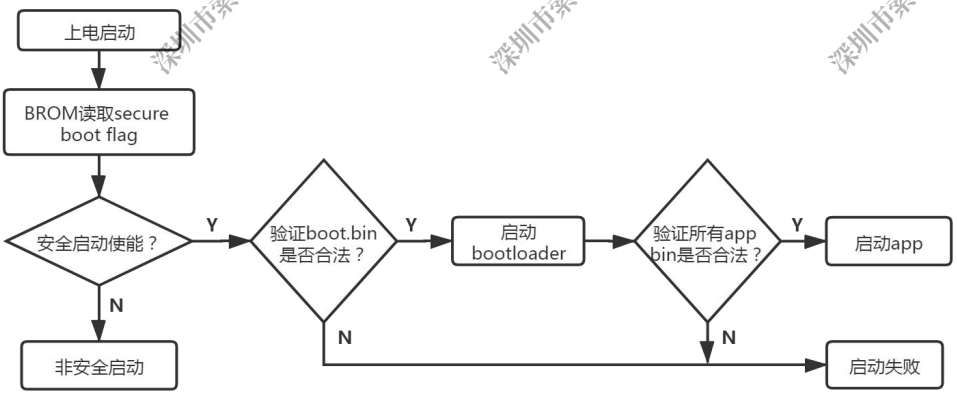
2．密钥：采用 ECC256 prime256v1 椭圆曲线生成公钥和私钥，由用户生成和保管。

3．签名：采用工具完成对 boot.bin 和所有 app bin 等数据的私钥签名。

4．BROM：采用公钥对 boot.bin 进行签名验证。

5．Bootloader：采用公钥对所有 app bin 进行签名验证。

图 3-1 XR806 安全启动流程



整体验证流程如下：

1．BROM 验证 Bootloader：当 eFuse 使能 secure boot flag 时，BROM 会验证 Bootloader 的合法性，包括公钥值和摘要签名值的合法性。

① 公钥值的验证：BROM 会读取固件中打包的公钥值并计算该公钥的摘要值，然后与 eFuse 中的公钥摘要值进行对比，验证通过，则进行 boot.bin 的摘要签名值验证。

② 摘要签名值的验证：BROM 读取 boot.bin 的有效数据并计算其摘要值 D1，通过前面验证过的公钥值将签名值解密为摘要值 D2，将 D1 与 D2 进行对比，验证通过，则加载启动 Bootloader，进行下一阶段 app bin 的验证。

2．Bootloader 验证所有 app bin：Bootloader 会按顺序验证 app.bin、app_xip.bin、app_psram.bin 等文件的合法性，验证方法与 BROM 验证 Bootloader 的摘要签名值一样，所有 app bin 文件验证通过后，才算整个验证流程通过。

注意

- BROM 启动时会检查 secure boot flag，若安全启动未使能，则 BROM 会进入非安全启动模式，直接加载 Bootloader 并运行。
- 验证过程中，若某一环节不通过，则验证失败，会进入 BROM 升级模式。

4 应用说明

4.1 应用简述

安全启动方案已经内嵌到 XR806 SDK，通过以下步骤即可启用。

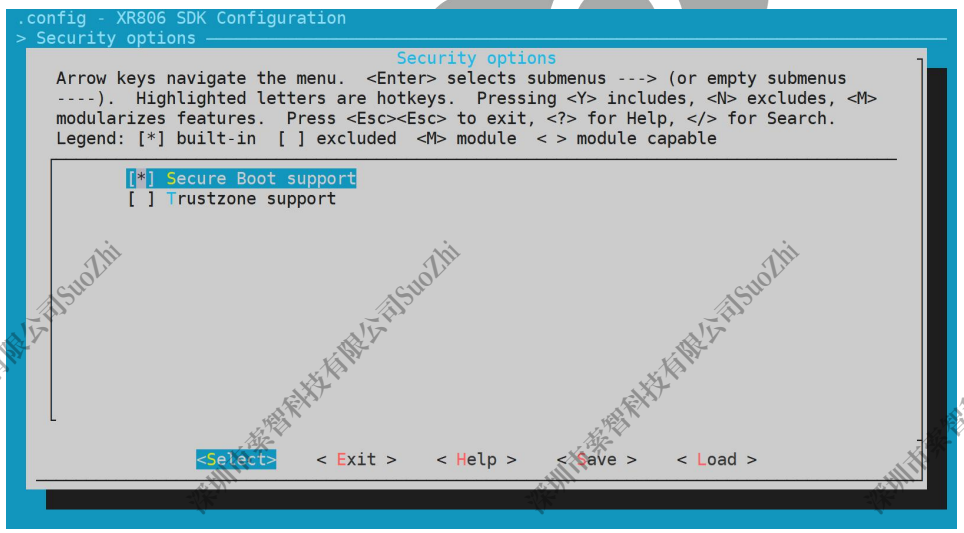
1. 检查/确认 SDK 中秘钥文件的配置，并进行备份保管，可参见“4.3 秘钥文件说明”章节
2. 编译安全 Bootloader，保证安全信任链的完整性，可参见“4.4 编译说明”章节
3. 在 menuconfig 中打开安全启动功能，保证编译固件时能对各 bin 文件进行签名与打包，可参见“4.2 配置说明”章节
4. 烧录 eFuse 中 secure boot 字段，开启 BROM 阶段的安全启动验证，可参见“4.5 烧录说明”章节

4.2 配置说明

开启 XR806 安全启动功能，需要进行 SDK 配置和 eFuse 特殊字段烧录，本章节主要介绍 SDK 中的配置介绍，eFuse 特殊字段烧录可参见“4.5 烧录说明”章节。

在 menuconfig 的 Security options 选项中选中 Secure Boot support，打开安全启动功能。

图 4-1 打开安全启动功能



说明

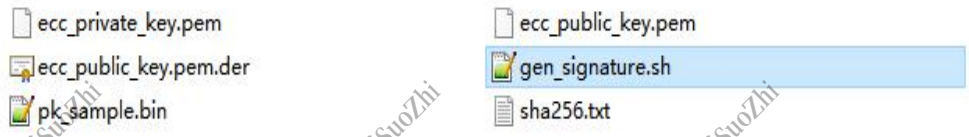
1. menuconfig 中开启“Secure Boot support”后，编译固件时会使用私钥自动对 boot.bin、app.bin、app_xip.bin、app_psram.bin 文件进行签名，生成对应的 xxx_sign.bin 文件，并一起打包至完整固件中。
2. 对 bin 文件的签名是在 SDK/project/sign_script/gen_signature.sh 脚本中完成，若为开启“Secure Boot support”后的第一次编译，SDK 会自动生成一对公、私钥文件，可参见“4.3 秘钥文件说明”章节。
3. 若用户开启了“Trustzone support”选项，则安全启动功能会被强制打开。

4.3 秘钥文件说明

秘钥文件的生成工具位于“SDK/project/sign_script”目录内，其文件内容说明可参见“2.3 文件位置”章节。

用户在编译最终的安全启动固件前，主要需要检查与确认文件夹中所包含的公钥文件（ecc_public_key.pem）、私钥文件（ecc_private_key.pem）、公钥提取值文件（pk_sample.bin）、公钥 Hash 值文件（sha256.txt）以及公钥证书文件（ecc_public_key.pem.der）是否存在，并将各文件自行备份保存，以防丢失。

图 4-2 生成签名所需的秘钥文件



其中，sha256.txt 文件主要存放 public key 的 SHA256 哈希值，该值为 32 字节 16 进制数据，此处转换成 64 个可读字符进行显示，举例如下（红色字体部分）：

```
SHA256(pk_sample.bin)= c7ecb9d555c998c7a34477acbc1e6f7f2b95a355840c41c7e707d73052f3d422
```

该哈希值将用于的 eFuse 中 Secure Boot 字段的烧录，烧录方法见“4.5 烧录说明”章节。

说明

1. 若用户在一个新工程中使用自己的秘钥对，则只需要将上述保存的文件替换到新工程的 SDK/project/sign_script 目录中即可。
2. 基于 ECC256 prime256v1 算法，生成的公钥值为 64-byte，对 bin 文件的签名值也为 64-byte。

注意

由于“SDK/project/sign_script”目录下用于签名的密钥属于机密信息，所以，开发和量产可能需要分别使用两套不同的密钥，即：

1. 开发过程中，使用临时密钥进行功能开发和验证。
2. 量产时使用保密的量产密钥进行签名，生成量产镜像。

4.4 编译说明

4.4.1 编译 bootloader

XR806 安全启动功能需要编译支持安全启动的 Bootloader，步骤如下：

1．在 SDK 根目录下执行：

```
cp ./project/bootloader/gcc/defconfig .config
```

2．在 menuconfig 中检查/确认 Security options 下 Secure Boot support 选项已打开

3．执行如下命令，进行编译：

```
make build
```

生成的新 boot.bin 会被自动拷贝到 SDK/bin/xradio_v3/boot/xr806 目录，下次编译 image 时，会自动打包。

注意

编译 Bootloader 步骤需要放在编译 image 之前进行，以保证 image 打包所用的 boot.bin 为最新。

4.4.2 编译 image

安全启动方案需要对“boot.bin”、“app.bin”等文件进行签名并打包生成 image，编译步骤如下：

1. menuconfig 中打开安全启动功能，打开方式可参见“4.2 配置说明”章节
2. 修改工程对应的 image.cfg 文件，检查/确认“boot.bin”和各 app bin 文件的证书名字是否填入（“cert”字段），并检查/确认“attr”属性赋值是否正确（可参照 SDK/project/image_cfg/readme.md 文档中的定义），即“boot.bin”、“app.bin”、“app_psram.bin”的“attr”属性值应为 0x5，“app_xip.bin”的“attr”属性值应为 0x6。

图 4-3 image config 检查

```
{
    "magic"   : "AWIH",
    "version" : "0.5",
    "OTA"     : {"addr": "1024K", "size": "4K"},
    "image"   : {"max_size": "1020K"},
    "section" :
    [
        {"id": "0xa5ff5a00", "bin": PRJ_IMAGE_BOOT_BIN, "cert": PRJ_BOOT_CERT, "max_len": "16K", "sram_offs": PRJ_IMAGE_BOOT_SRAM_OFFS, "ep": PRJ_IMAGE_BOOT_EP, "attr": PRJ_IMAGE_BOOT_ATTR},
        {"id": "0xa5fe5a01", "bin": "app.bin",          "cert": "app_sign.bin",  "max_len": "150K", "sram_offs": PRJ_IMAGE_APP_SRAM_OFFS,  "ep": PRJ_IMAGE_APP_EP,  "attr": "0x5"},
#if (defined(CONFIG_XIP))
        {"id": "0xa5fd5a02", "bin": "app_xip.bin",    "cert": "app_xip_sign.bin", "max_len": "700K", "sram_offs": "0xffffffff", "ep": "0xffffffff",  "attr": "0x6"},
#endif
#if (defined(CONFIG_PSRAM))
        {"id": "0xa5f65a09", "bin": "app_psram.bin", "cert": "app_psram_sign.bin", "max_len": "110K",  "sram_offs": PRJ_PSRAM_START_OFFS,  "ep": "0x00008000",  "attr": "0x5"},
#endif
#if (defined(CONFIG_WLAN))
        {"id": "0xa5fa5a05", "bin": "wlan_bl.bin",      "cert": "null",    "max_len": "4K", "sram_offs": "0xffffffff",   "ep": "0xffffffff",   "attr": "0x1"},
        {"id": "0xa5f95a06", "bin": "wlan_fw.bin",      "cert": "null",    "max_len": "128K", "sram_offs": "0xffffffff",  "ep": "0xffffffff",   "attr": "0x1"},
#endif
#if (defined(CONFIG_WLAN) || defined(CONFIG_BLE))
        {"id": "0xa5f85a07", "bin": PRJ_IMAGE_SYS_SDD_BIN,  "cert": "null",    "max_len": "1K", "sram_offs": "0xffffffff",  "ep": "0xffffffff",   "attr": "0x1"},
#endif
        {}
    ],
}
```

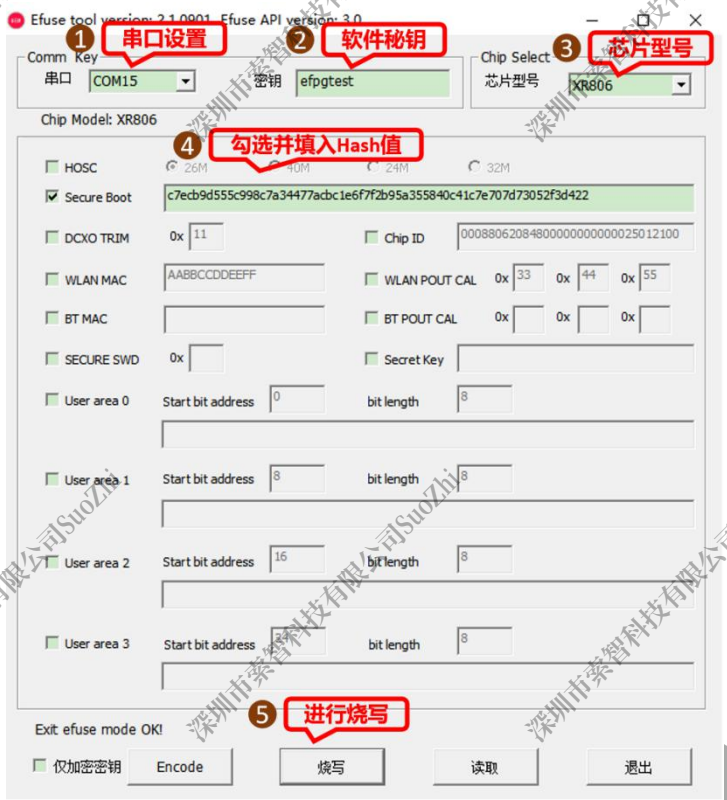
3. 执行“make build”命令，编译用户自己的工程，生成新的 image 固件。

4.5 烧录说明

要使完整的安全验证链生效，必须把正确的 ECC256 Public Key 的 SHA256 哈希值（“4.3 秘钥文件说明”章节提到的保存在文件“sha256.txt”中的哈希值）烧录到芯片的 eFuse 芯片中，具体方法如下：

1. 打开“efuse_tool.exe”工具，选择“Secure Boot”选择框，并填入需要写入的 ECC256 Public Key 的 SHA256 哈希值（64 字符表示），如下图所示。

图 4-4 烧录 eFuse Secure Boot 字段



2. 点击“烧写”按钮进行 eFuse Secure Boot 字段烧写。

注意

1. 待烧写的工程板应保证其运行固件支持 eFuse 的测试命令。
2. 向 eFuse 中烧录 Secure Boot 字段后，其 eFuse 内部的 secure boot flag 会被自动使能。

5 示例说明

安全启动方案主要用于保护固件的合法性与完整性，防止固件被他人替换、篡改，本示例主要演示 XR806 安全启动功能的开启以及该功能的有效性。

5.1 示例简介

本示例工程主要展示安全启动在 XR806 芯片的有效性：

1. 当开启安全启动功能后（烧写了 eFuse 中 Secure Boot 字段），使用自有的秘钥文件编译出的固件在 XR806 芯片上可正常运行。
2. 当使用其他秘钥文件编译生成的固件或者其他任何非法固件进对此芯片进行烧写后，均无法正常运行。

5.1.1 获取方式

安全启动示例有示例工程代码，位于 XR806 SDK 的/project/example/secure_boot 目录，以下此示例工程简称为 secure_boot 示例工程。

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

5.1.2 准备工作

secure_boot 示例工程的硬件准备有如下。

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 UART0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。

secure_boot 示例工程的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》。

5.2 操作步骤

3. 检查 menuconfig 中“Secure Boot support”功能是否打开，可参考“4.2 配置说明”章节。
4. 执行“make build”命令编译此工程，并烧写进开发板中运行。
5. 查看 SDK/project/sign_script 目录下“sha256.txt”文件中的 Hash 值。
6. 使用“efuse_tool.exe”工具将 64 个可读字符显示的 Hash 值（256bit）烧写进 XR806 芯片的 eFuse 的中，可参考“4.5 烧录说明”章节操作。
7. 重启开发板，查看控制台输出。

8. 在 menuconfig 关闭“Secure Boot support”功能，重新编译固件，烧写运行，查看运行状态。
9. 备份 SDK/project/sign_script 目录下的各秘钥文件后将其删除（除 gen_signature.sh 外），在 menuconfig 中打开“Secure Boot support”功能，再次编译固件并烧写运行，查看运行状态。
10. 将备份的各秘钥文件重新拷贝到 SDK/project/sign_script 目录下，保持 menuconfig 中“Secure Boot support”功能为打开状态，编译固件烧写运行，并查看运行状态。

5.3 效果展示

上述步骤 2 中烧写固件后，可正常运行，因为此时 eFuse 中的 Secure Boot 未被烧写，BROM 默认采用非安全启动方式运行。此时控制台可正常输出，符合预期。

烧写完 eFuse 的 Secure Boot 字段后，重启开发板，可成功运行，控制台输出如下，符合预期。

```
use default flash chip mJedec 0x0
[FD I]: mode: 0x10, freq: 96000000Hz, drv: 0
[FD I]: jedec: 0x0, suspend_support: 1
mode select:e

platform information ============================================
XR806 SDK v0.3.0    Feb 26 2021 16:50:22

heap space [0x205080, 0x24bc00), size 289664

cpu    clock 160000000 Hz
HF     clock    40000000 Hz

sdk option:
    XIP                : enable
    Security Boot : enable
    INT LF OSC       : enable
    SIP flash          : enable

====================================================================
```

步骤 6 中，当关闭 menuconfig 中“Secure Boot support”功能后，编译固件时不会进行 public_key 文件和签名文件的打包，因此生成的固件烧写到开发板后会在 BROM 阶段校验失败，原因是找不到 public_key，向控制台键入“U”后，控制台返回 OK，表示已进入 BROM 升级模式，符合预期，如下所示：

```
OKOKOKOKOKOKOKOK
```

步骤 7 中表示更换了一套新的秘钥文件进行 bin 文件签名与打包，生成的固件，烧写入开发板必然无法校验通过，向控制台键入“U”后，控制台返回 OK，表示已进入 BROM 升级模式，符合预期，如下：

```
OKOKOKOKOK
```

步骤 8 中将备份的各秘钥文件重新拷贝至 SDK/project/sign_script 目录后，生成的固件为合法固件，其 public_key 的 Hash 值会与 eFuse 中的 Secure Boot 字段内容成功匹配，后面的校验也会顺利通过，开发板可正常运行此固件，如下：

```
use default flash chip mJedec 0x0
[FD I]: mode: 0x10, freq: 96000000Hz, drv: 0
[FD I]: jedec: 0x0, suspend_support: 1
mode select:e

platform information ============================================
XR806 SDK v0.3.0    Feb 26 2021 17:09:32

heap space [0x205080, 0x24bc00), size 289664

cpu    clock 160000000 Hz
HF     clock    40000000 Hz

sdk option:
       XIP                 : enable
       Security Boot : enable
       INT LF OSC        : enable
       SIP flash          : enable

============================================================
```

附录 A： 术语表

表 A-1 术语表

E		
ECC	Elliptic curve cryptography	椭圆曲线密码学
O		
OEM	Original Equipment Manufacturer	原始设备制造商
P		
PSRAM	Pseudo Static Random Access Memory	伪随机静态存取存储器
X		
XIP	eXecute In Place	在 Flash 中执行代码

著作权声明

商标声明

免责声明