**xilinx加密方案使用说明简化版V1.0**

|  |  |
| --- | --- |
| 拟 制 |  |
| 审 核 |  |
| 会签 |  |
| 批 准 |  |

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修订版本** | **日期** | **作者** | **修改描述** | **备注** |
| V1.0 | 2023.08.15 | 操正 | 初始版本 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1.1 项目背景和需求 4](#_Toc143272958)

[1.2 xilinx加密方案原理 4](#_Toc143272959)

[1.2.1 加密过程 4](#_Toc143272960)

[1.2.2 解密过程 6](#_Toc143272961)

[1.3 加密相关文件的准备 6](#_Toc143272962)

[1.2.1AES key的生成 7](#_Toc143272963)

[1.2.2生成AES key烧写程序 7](#_Toc143272964)

[1.2.3使用AES key加密Boot升级程序 9](#_Toc143272965)

[1.2.4使用AES key加密APP程序 11](#_Toc143272966)

[1.4 加密方案验证 12](#_Toc143272967)

[1.4.1 把key写入设备的eFUSEs 12](#_Toc143272968)

[1.4.2 AES key加密Boot升级程序 13](#_Toc143272969)

[1.4.3 烧写AES key加密后的APP程序 13](#_Toc143272970)

[2 Issue 14](#_Toc143272971)

[2.1 滑工：项目背景和需求补充 14](#_Toc143272972)

[2.2 滑工：解密流程存在问题 14](#_Toc143272973)

[2.3 吴经理：AES KEY程序如何解密 14](#_Toc143272974)

[2.4 吴经理：AES KEY程序烧写到板子后，再烧写不加密的boot或app能正常运行吗 15](#_Toc143272975)

[2.5 陈奕利，刘栩宏：efuse是否会被用户读取 15](#_Toc143272976)

[2.5.1 密钥或密码保护 15](#_Toc143272977)

[2.5.2 访问权限控制 15](#_Toc143272978)

[2.5.3 物理保护 16](#_Toc143272979)

[2.5.4 安全验证 16](#_Toc143272980)

[2.6 吴经理：生产流程指定，加密服务器搭建 16](#_Toc143272981)

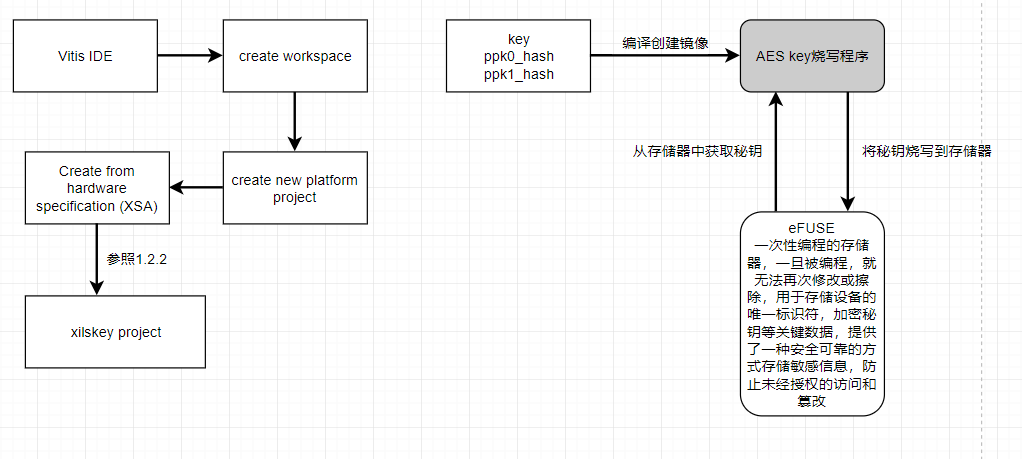
## 项目背景和需求

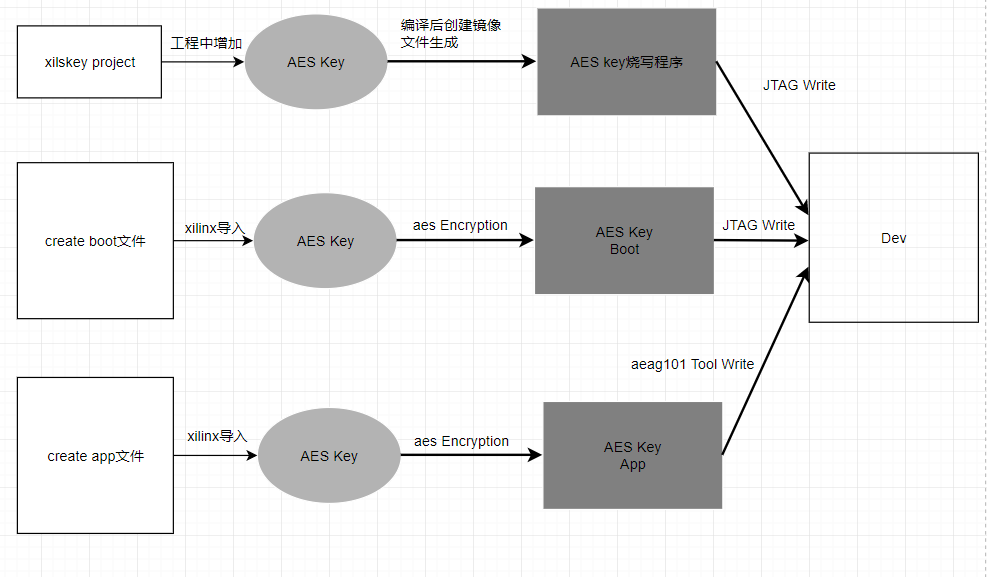
目前反制枪的版本未经过加密处理，给客户提供临时版本或进行工厂生产时，都没有对反制枪程序进行加密处理，应用程序很容易会被用户或其他竞争对手获取到，获取到未加密的程序后，直接进行反汇编就能得到应用程序的内部实现，从而导致应用程序的泄露，给公司造成重大损失。

对于反制枪程序的加密处理，被提上了日程。

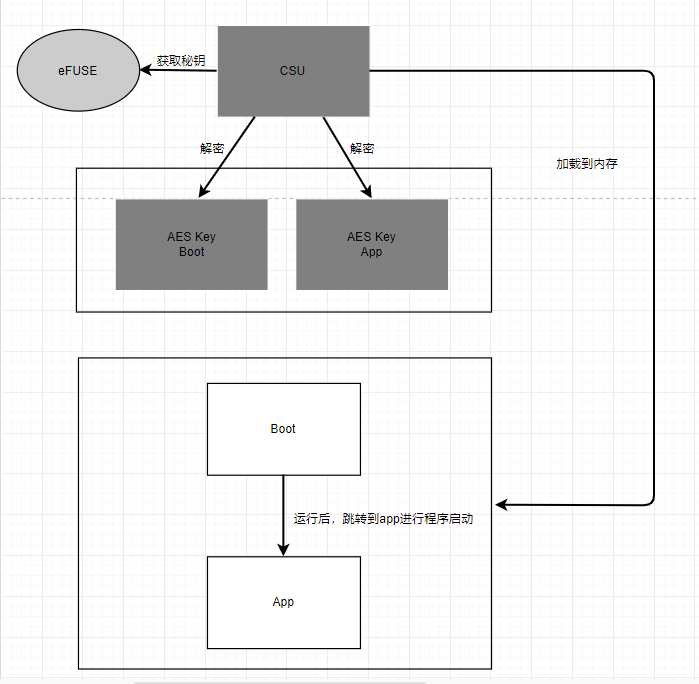
## xilinx加密方案原理

### 加密过程





### 解密过程



## 加密相关文件的准备

使用xilinx的加密方案需要准备下列内容：

***aes.nky***

AES key烧写程序

使用***aes.nky***加密的Boot升级程序

使用***aes.nky***加密的应用程序

### 1.2.1AES key的生成

可以选用不同的工具生成***AES key***，长度为256bit，包含大写字母和数字。

推荐一个生成算法：创建一个包含大写字母和数字的字符集，使用随机数生成器生成64个随机数，范围为0到字符集长度减1，将每个随机数作为索引，从字符集中取出对应的字符，再将每个随机数作为索引，从字符集中取出对应的字符，将64个字符连接起来，形成最终的64字节字符串。

生成的字**符串例如**：

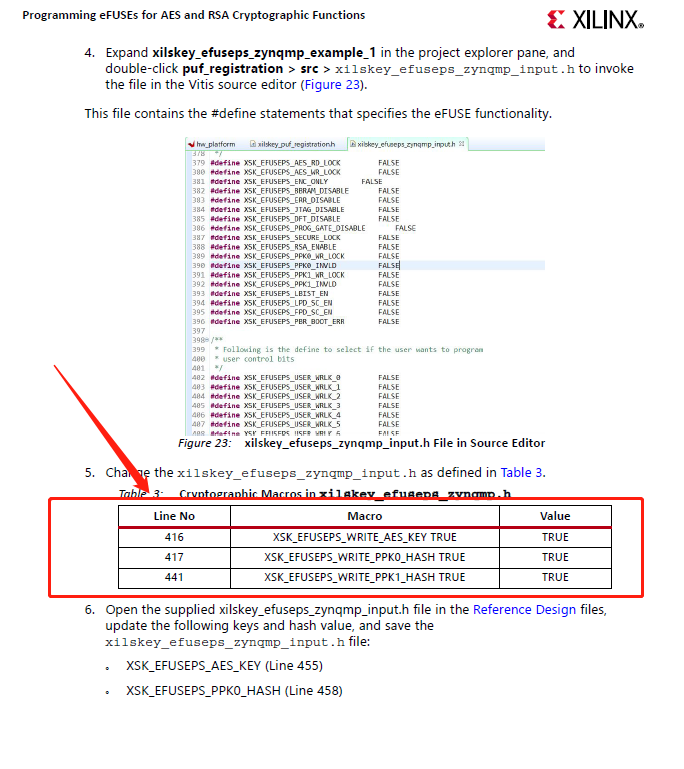
45195DE9B5B80119D8DD4E7DF032736D53CF75AD1DC1E61C5BA681CF4A0724E8

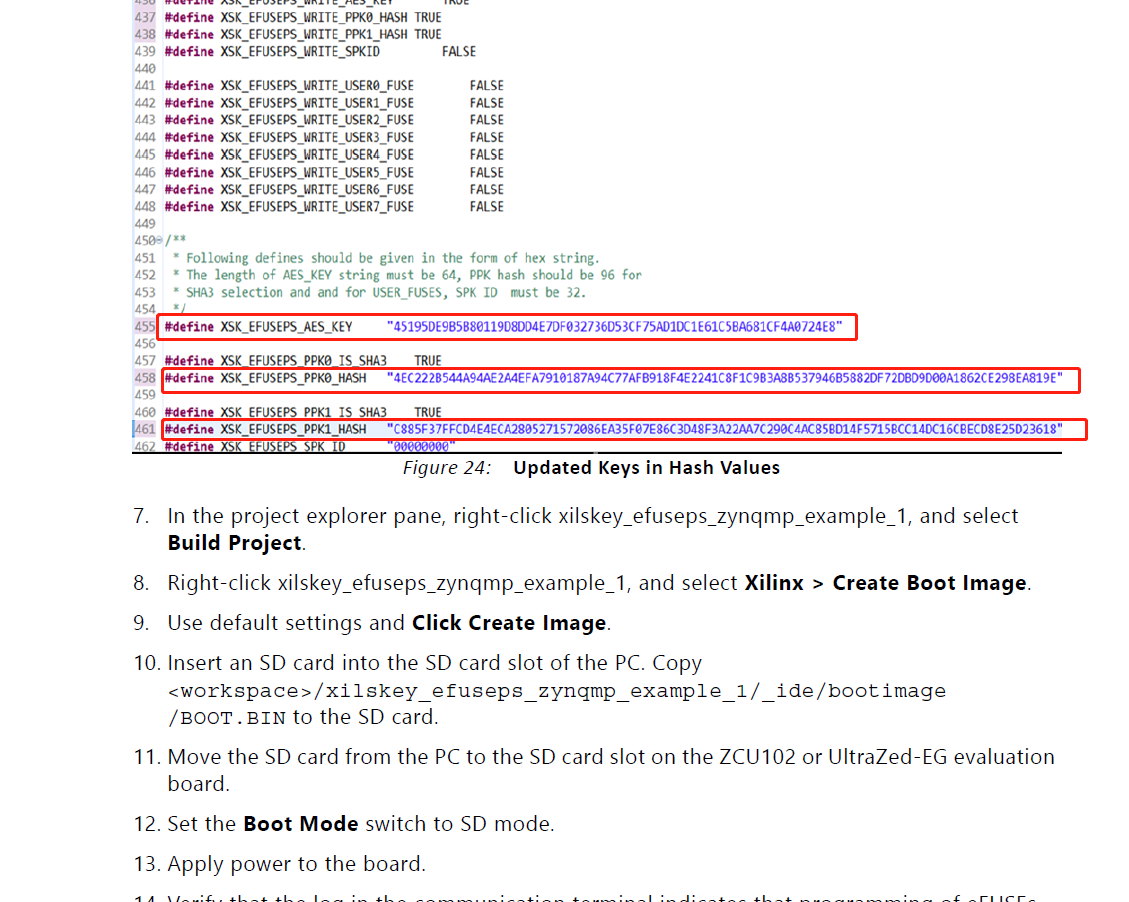
### 1.2.2生成AES key烧写程序

程序项目的生成和编译参见文档《xapp1319-zynq-usp-prog-nvm.pdf》

【Programming the AES Key in BBRAM】的【Launch Vitis and create a platform project】章节【Programming eFUSEs for AES and RSA Cryptographic Functions】章节

主要修改六个地方



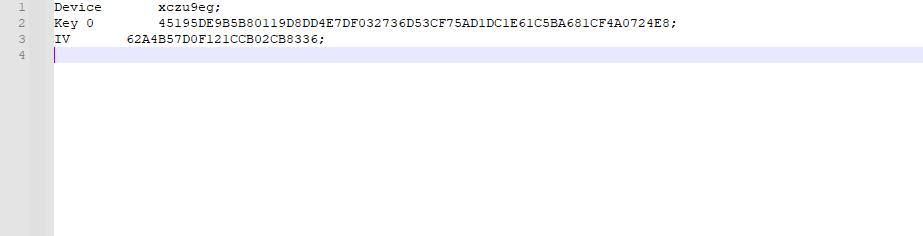


在本方案中使用eFUSEs保存***AES key***，eFUSEs只能烧写一次（eFUSEs是赛灵思提供的参数存储方式，只能写一次，写一次后就不允许再写）。

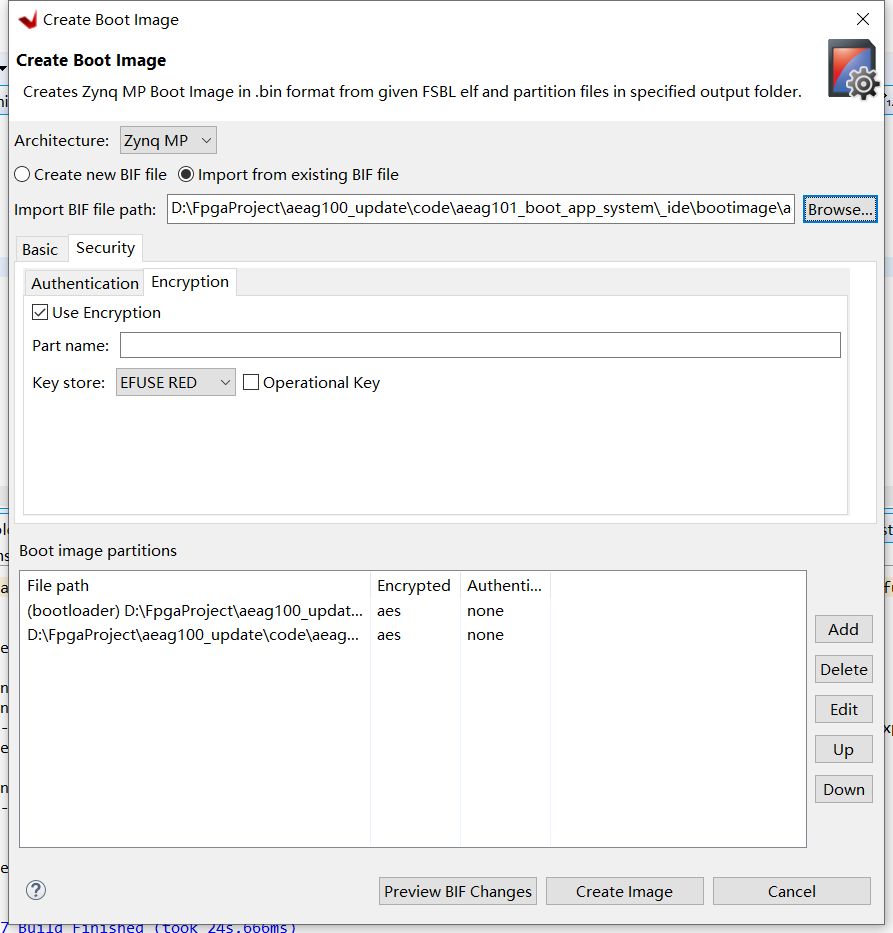
名词解释：**eFUSE**是一种一次性编程的存储器，一旦被编程，就无法再次修改或擦除。它通常用于存储设备的唯一标识符、加密密钥、许可证信息和其他关键配置数据。由于其不可逆的特性，eFUSE提供了一种安全和可靠的方式来存储敏感信息，防止未经授权的访问和篡改。

### 1.2.3使用AES key加密Boot升级程序

使用<https://adasgitlab.autel.com/aeag/aeag100_upgrade.git>中的dev\_embed\_boot分支，编译生成boot程序后，使用vitis的create boot image生成boot的image文件，image的所有文件都需要使用***aes.nky***均进行加密，文件加密时需要选取***aes.nky***文件，***aes.nky***文件格式如下：

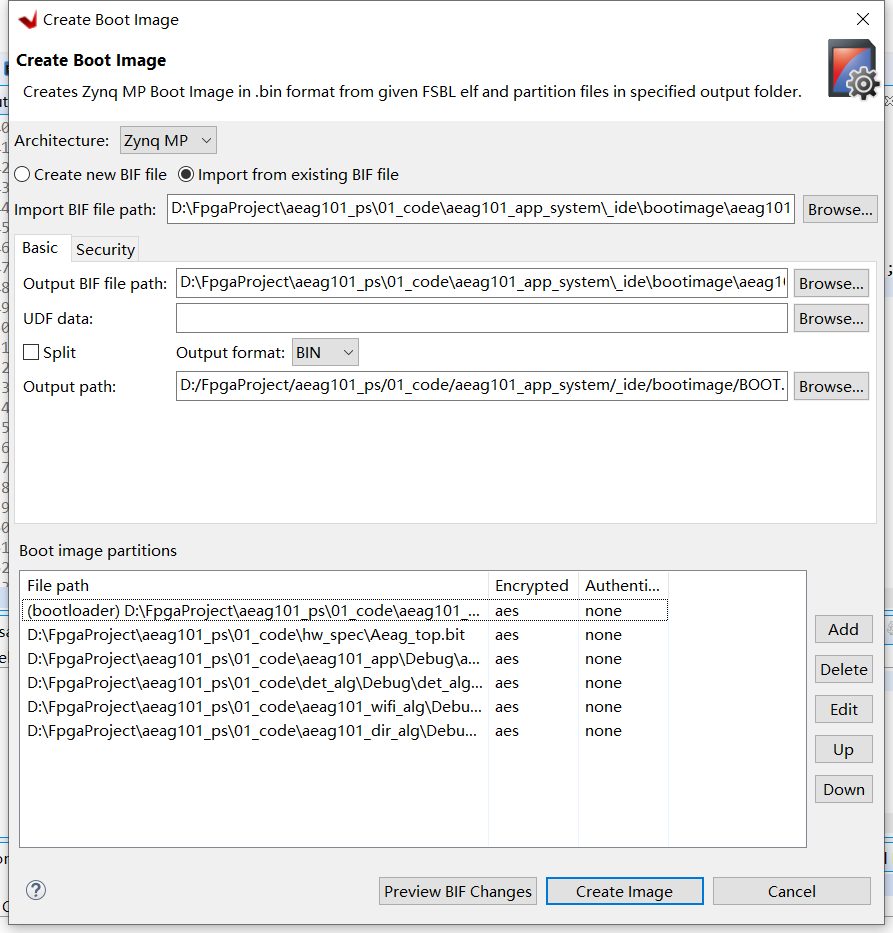


不同文件key文件的内容相同、key同***AES key***烧写程序中的key一致，但key文件名不同，fsbl（bootloader）文件必须加密，界面如下所示：



### 1.2.4使用AES key加密APP程序

APP项目编译完成后，同样使用vitis的create boot image生成APP的image文件，image的所有文件均进行加密，文件加密时需要选取key文件，不同文件key文件的内容相同、key同***AES key***烧写程序中的key一致，但***aes.nky***文件名不同，fsbl（bootloader）文件必须加密，需要注意，如果生成的APP image文件大于16M，有可能不能启动，界面如下所示：



## 加密方案验证

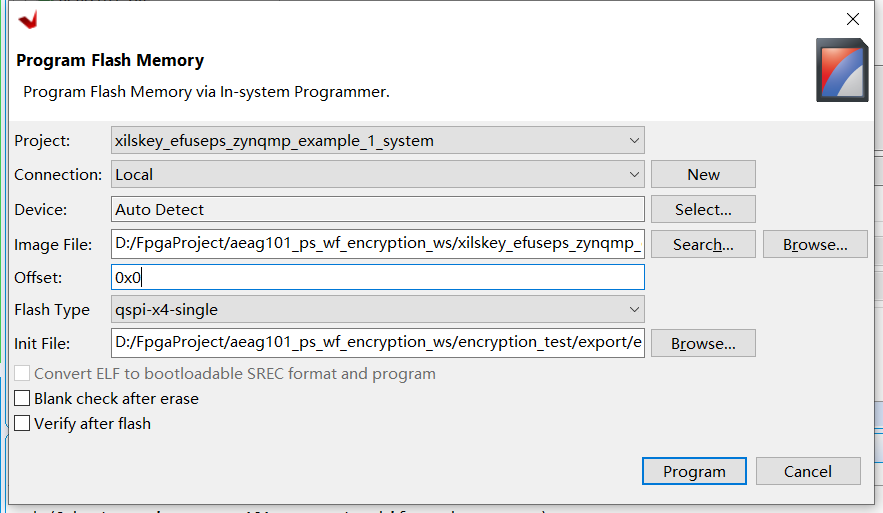
按照下列步骤验证加密程序是否正常运行：

### 把key写入设备的eFUSEs

eFUSE只能烧写一次，烧写成功后再次烧写会提示烧写失败。把key写入设备的eFUSE可以使用两种方式：

第一种方式：把编译好的***AES key***烧写程序通过JTAG口下载到设备内存运行，串口会打印运行结果。

第二种方式：把生成的***AES key***烧写程序在vitis内通过JTAG口烧写到设备FLASH，烧写成功后，设备重新上电，串口会打印运行结果。

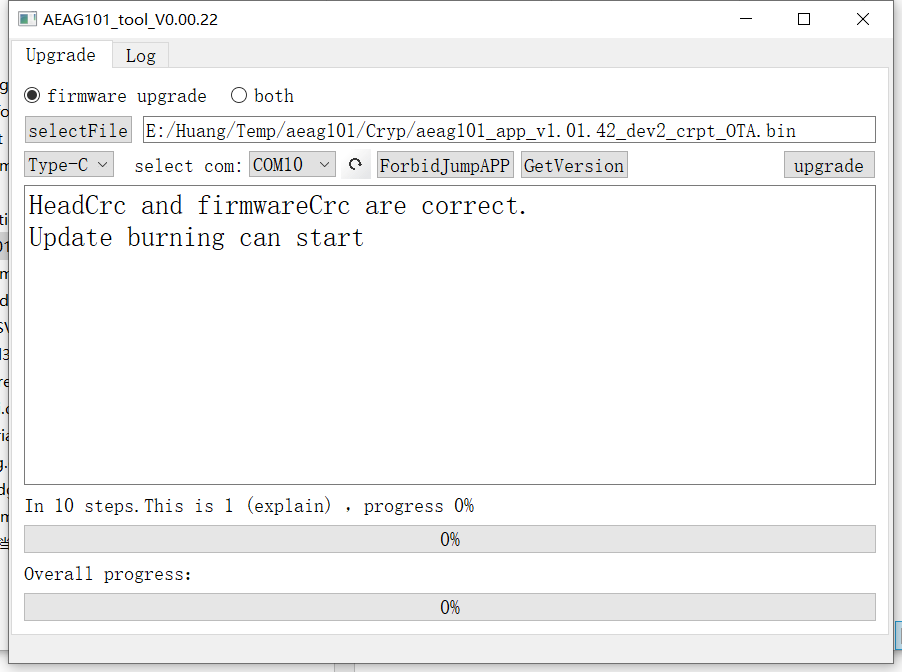


### AES key加密Boot升级程序

把***AES key***加密的boot程序在vitis内通过JTAG口烧写到设备FLASH，烧写成功后，设备重新上电，串口会打印运行结果，并提示可以烧写APP程序。

### 烧写AES key加密后的APP程序

使用公司自研的上位机烧写程序烧写***AES key***加密后的APP程序到flash，烧写成功后，设备重新上电，设备如果正常工作，且版本号应该同烧写程序的版本号一致，说明烧写的加密软件成功运行。



# Issue

## 滑工：项目背景和需求补充

答：背景和需求已补充。

## 滑工：解密流程存在问题

答：解密流程已重新修改。

## 吴经理：AES KEY程序如何解密

答：cpu里有一个解密模块CSU（Configuration and security unit）

CSU是安全引导过程的中心。它强制硬件根的信任或仅在启用安全引导步骤时加密。犯罪现场组还负责维护安全通过禁止从安全状态过渡到不安全状态来改变设备的状态，或者

从不安全状态到安全状态，没有完整的POR。一旦FSBL(和，如果适用的，PMUFW)已安全加载，CSU归零的存储元素并将复位释放到指定的处理单元(APU或)RPU)。

## 吴经理：AES KEY程序烧写到板子后，再烧写不加密的boot或app能正常运行吗

答：不能正常运行，eFUSE是一次性编程存储器，没有提供加密开启关闭的控制开关，通过AES Key烧写程序把秘钥写到eFUSE后，运行boot和app时，PMU模块会获取对加密程序通过PPK HASH做完成性检查，然后CSU解密模块需要去加密程序中获取秘钥进行解密操作，将获取到的秘钥和eFUSE中的秘钥做比较，验证通过后解密成功，加载到内存运行程序，未加密的程序没有秘钥和PPK HASH等信息，无法执行解密流程，程序不能正常运行。

## 陈奕利，刘栩宏：efuse是否会被用户读取

答：Xilinx eFUSE是一种用于存储配置和密钥信息的可编程电子熔丝。通常情况下，Xilinx eFUSE是由芯片制造商在生产过程中进行编程的，以确保配置和密钥信息的安全性。一般情况下，用户无法直接获取或修改Xilinx eFUSE中的信息。

然而，一些特定的Xilinx芯片型号可能提供了一些用户可编程的eFUSE位，允许用户在特定条件下进行一些配置或密钥信息的修改。这些用户可编程的eFUSE位通常受到严格的访问控制和安全保护机制的限制，以确保只有授权用户才能进行修改。

常见的授权和安全机制示例：

### 密钥或密码保护

用户需要提供正确的密钥或密码才能访问和修改eFUSE中的信息。这可以防止未经授权的用户获取敏感信息。

### 访问权限控制

芯片制造商可以设置访问权限，只有特定的用户或设备才能访问和修改eFUSE。这可以确保只有授权用户才能进行操作。

### 物理保护

eFUSE位可能被物理上保护，例如使用特殊的封装或封装技术，以防止非授权的物理访问。

### 安全验证

在进行eFUSE位的修改之前，可能需要进行安全验证，例如使用数字签名或加密算法来验证用户的身份和操作的合法性。

## 吴经理：生产流程指定，加密服务器搭建