高安全 CA

开发指南

文档版本　13

发布日期　2016-03-04

商标声明

 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：　深圳市龙岗区坂田华为基地华为总部　邮编：518129

网址：　http://www.hisilicon.com

客户服务邮箱：　support@hisilicon.com

前 言

概述

本文档主要介绍高安全 CA 芯片提供的安全方案，以及生产配置流程。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3716C 芯片	V200
Hi3716M 芯片	V300/V400/V310/V420/V410
Hi3719C 芯片	V100
Hi3719M 芯片	V100
Hi3796C 芯片	V100
Hi3798C 芯片	V100/V200
Hi3798M 芯片	V100
Hi3796M 芯片	V100
Hi3110E 芯片	V500

注意：

Hi3716MV300、Hi3716MV400、 Hi3716MV310、Hi3110EV500、Hi3716CV200、Hi3719CV100、Hi3719MV100、Hi3796CV100、Hi3798CV100、Hi3798CV200、Hi3798MV100、Hi3796MV100、Hi3716MV420、Hi3716MV410 芯片高安方案一致。

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
DANGER	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
WARNING	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
CAUTION	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
TIP	表示能帮助您解决某个问题或节省您的时间。
NOTE	表示是正文的附加信息，是对正文的强调和补充。

作者信息

章节号	章节名称	作者信息
全文	全文	L00168554

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2012-03-05	02	全面更新。
2012-07-17	03	再次全面更新。
2012-12-06	04	更新签名方案和 PVR 方案使用指南。
2013-03-20	05	按照 CA/STB 厂商要求更新名方案。
2013-9-16	06	支持 Hi3716CV200 / Hi3719CV100 / Hi3719MV100。

修订日期	版本	修订说明
2014-04-02	07	修改 4.3 章节；添加 4.4、4.5、4.6、4.7 章节。
2014-06-05	08	增加 Hi3716MV400/Hi3796CV100/Hi3798CV100 芯片支持，修改 3.6.8，4.5 章节。
2014-09-25	09	删除 Hi3110EV300、Hi3716CV100、Hi3716MV200、Hi3716HV100； 删除海思丝印为 H 的芯片描述； 增加 Hi3716MV310/Hi3798MV100，章节调整。
2015-03-03	10	增加海思丝印为 H 的芯片描述； 增加 Panaccess 高安芯片描述； 增加 Hi3110EV500，章节调整。 更正 2.4.2.2 非 boot 镜像通用签名的命令。
2015-05-25	11	增加 Hi3716MV420/Hi3716MV410。
2016-02-26	12	新增支持 Hi3798CV200。
2016-03-04	13	调整海思公共安全芯片章节位置；添加海思非安全芯片章节。

目 录

前 言 iii

1 高级安全芯片 1-1

1.1 高安芯片说明 1-1

1.2 高安芯片特性 1-2

2 高安 CA 方案说明 2-1

2.1 概述 2-1

2.2 安全启动整体流程 2-1

2.3 BOOT 镜像安全启动 2-3

2.3.1 BOOT 镜像分类 2-3

2.3.2 安全 BOOT 的 Flash 内部存储结构 2-4

2.3.3 BOOT 镜像设置 MSID 2-6

2.3.4 安全 BOOT 的开发流程 2-6

2.4 非 BOOT 软件安全校验 2-7

2.4.1 安全校验过程 2-7

2.4.2 非 BOOT 软件签名模式分类 2-8

2.4.3 参考代码 2-17

2.5 安全软件升级 2-17

2.6 TS 流解密 2-18

2.6.1 密文控制字 CW 使用流程 2-19

2.6.2 明文控制字 CW 使用流程 2-21

2.7 内容加解密方案 2-22

2.7.1 方案说明 2-22

2.7.2 使用随机数密钥对内存数据加解密 2-22

2.7.3 使用固定密钥对内存数据加解密 2-23

2.8 安全 PVR 方案 2-25

2.8.1 高安 PVR 录制 2-25

2.8.2 高安 PVR 播放 2-28

2.9 JTAG 保护方案 2-30

3 高安机顶盒量产 3-1

3.1 概述 3-1
3.2 高安 CA 机顶盒工厂生产基本步骤 3-1
3.2.1 高安 CA 芯片基本的设置 3-3
3.3 各 CA 芯片使用说明 3-6
3.3.1 Novel 高安芯片 3-6
3.3.2 Suma 高安芯片 3-7
3.3.3 CTI 高安芯片 3-8
3.3.4 Verimatrix 高安芯片 3-10
3.3.5 Nagra 高安芯片 3-16
3.3.6 Conax 高安芯片 3-16
3.3.7 Irdeto 高安芯片 3-16
3.3.8 Panaccess 高安芯片 3-17
3.3.9 海思非安全芯片 3-20
3.3.10 海思公共安全芯片 3-25

插图目录

图 2-1 机顶盒启动基本流程图.....2-2
图 2-2 安全 BOOT 在 Flash 的内部存储结构.....2-4
图 2-3 新 BOOT 在 Flash 的内部存储结构.....2-5
图 2-4 非 BOOT 文件特定签名的结构图.....2-9
图 2-5 非 BOOT 软件（单个）签名镜像结构图.....2-10
图 2-6 非 BOOT 软件（多个）签名镜像结构图.....2-11
图 2-7 非 BOOT 文件通用签名的结构图.....2-14
图 2-8 文件进行签名生成的签名镜像格式.....2-15
图 2-9 安全软件升级流程图.....2-17
图 2-10 DVB TS 流解密示意图.....2-19
图 2-11 3 级密文 CW 解密流程图.....2-20
图 2-12 数据加解密方案简图.....2-22
图 2-13 使用随机数密钥对内存数据加解密流程图.....2-23
图 2-14 使用固定密钥对内存数据加解密流程图.....2-25
图 2-15 高安 PVR 录制过程图.....2-26
图 2-16 高安 PVR 录制文件结构图.....2-27
图 2-17 高安 PVR 播放过程图.....2-29
图 3-1 高安 CA 机顶盒工厂生产基本流程图.....3-2
图 3-2 高安 CA 芯片基本的 PV 设置流程图.....3-4
图 3-3 Hi3716MV300/Hi3716CV200 Verimatrix Standard-STB 安全级别芯片 PV 设置流程图.....3-14
图 3-4 Hi3716MV310 Panaccess 安全级别芯片 PV 设置流程图.....3-19
图 3-5 典型锁定流程.....3-21
图 3-6 典型量产流程.....3-22
图 3-7 典型流程.....3-23
图 3-8 典型流程.....3-24

图 3-9 典型流程 3-25
图 3-10 Hi3716CV200 海思公用高安芯片 PV 设置流程图 3-27
图 3-11 CAS 私有数据设置 3-30
图 3-12 包含烧写 CAS 私有数据的量产流程 3-31

表格目录

表 1-1 海思高安芯片型号.....1-1
表 1-2 高安芯片和普通芯片之间安全特性差别.....1-2
表 2-1 3 种不同 BOOT 的适用范围。.....2-3
表 2-2 可用于验证非 BOOT 文件特定签名的命令.....2-12
表 2-3 可用于验证非 BOOT 文件通用签名的命令.....2-16
表 3-1 Novel 高安芯片密钥表.....3-6
表 3-2 Suma 高安芯片密钥表.....3-7
表 3-3 CTI 高安芯片密钥表.....3-9
表 3-4 Verimatrix Standard-STB 安全级别密钥表.....3-10
表 3-5 Panaccess 高安芯片密钥表.....3-17
表 3-6 STB 厂商需要设置的 key.....3-25
表 3-7 CAS 厂商私有数据.....3-29

1 高级安全芯片

1.1 高安芯片说明

机顶盒（STB）芯片用于开展与数字电视广播相关的运营业务。各地运营商使用机顶盒向广大客户提供数字电视服务时，要求芯片厂商，CA 厂商和机顶盒厂商联合开发的机顶盒必须具备一定的安全保护业务，确保运营商的合法收益。

现有的普通机顶盒芯片所提供的保护很难抵抗黑客的盗版攻击。

CA 厂商和芯片厂商由此需要共同合作开发可以提供更安全保护措施的方案，高安 CA 芯片由此而产生。

海思根据 CA 公司的要求定制的各类高安 CA 芯片，在芯片内部植入 CA 公司提供的各种密钥。所以 STB 厂商在申请高安芯片时，应该明确申请的芯片是用于哪家 CA 公司，从而订购对应型号的芯片。

如下以 Hi3716MV300 为例，海思可以提供的的芯片如表 1-1 所示。

表1-1 海思高安芯片型号

芯片型号	芯片描述	备注
Hi3716MRBCV3000x0	非高安芯片，芯片标志 **0**	已经量产
Hi3716MRBCV300Cx0	Conax 高安芯片，Conax 芯片标志 **C**	已经量产
Hi3716MRBCV300Rx0	Nagra 高安芯片，Nagra 芯片标志 **R**	已经量产
Hi3716MRBCV300Yx0	Novel 高安芯片，Novel 芯片标志 **Y**	已经量产
Hi3716MRBCV300Sx0	Suma 高安芯片，Suma 芯片标志 **S**	已经量产
Hi3716MRBCV300Tx0	CTI 高安芯片，CTI 芯片标志 **T**	已经量产
Hi3716MRBCV300Mx0	Verimatrix 高安芯片，Verimatrix 芯片标志 **M**	已经量产
Hi3716MRBCV300Ix0	Irdeto 高安芯片，Irdeto 芯片标志 **I**	已经量产
Hi3716MRBCV300Hx0	海思公共安全芯片，其芯片标志 **H**	已经量产

芯片型号	芯片描述	备注
Hi3716MRBCV300**K**x0	DCAS 芯片，其芯片标志 **K**	已经量产
Hi3716MRBCV310**P**x0	PANACCESS 芯片，其芯片标志 **P**	（Hi3716MV300 不支持， Hi3716MV310 以及后续芯片支持） 已经量产

1.2 高安芯片特性

高安全 CA 芯片提供以下功能：

- 芯片提供唯一标识符，即 CHIPID。
- 支持 BOOT 安全启动，提供 BOOT 签名校验和 BOOT 加密机制。
- 支持密文控制字解扰。
- 支持内容保护，支持 AES 和 3DES 加解密算法。
- 支持高安全 PVR。
- 内置安全 CPU。
- 支持 JTAG 保护。

表 1-2 列举出高安芯片和普通芯片之间安全特性差别。

表1-2 高安芯片和普通芯片之间安全特性差别

芯片可以提供的功能	普通芯片	高安芯片
使用明文 CW 解扰被加密的码流	提供	提供
使用密文 CW 解扰被加密的码流	不提供	提供
安全启动。机顶盒只允许运行合法的程序，未通过验证的程序无法运行	不提供	提供
安全 PVR 业务	不提供	提供
JTAG 等端口保护	不提供	提供

2 高安 CA 方案说明

2.1 概述

使用高安芯片的机顶盒可以提供实现如下的安全方案：

- 安全启动。
- 密文控制字解密 TS 流。
- 内容加解密方案。
- 安全 PVR 方案。
- JTAG 保护方案。

2.2 安全启动整体流程

正常运行的机顶盒会执行多种类型程序，包含运营商提供的各种服务。运行过程中，如果机顶盒执行了非法的程序，其所运行环境就受到威胁，运营商和客户的信息有可能被泄露。

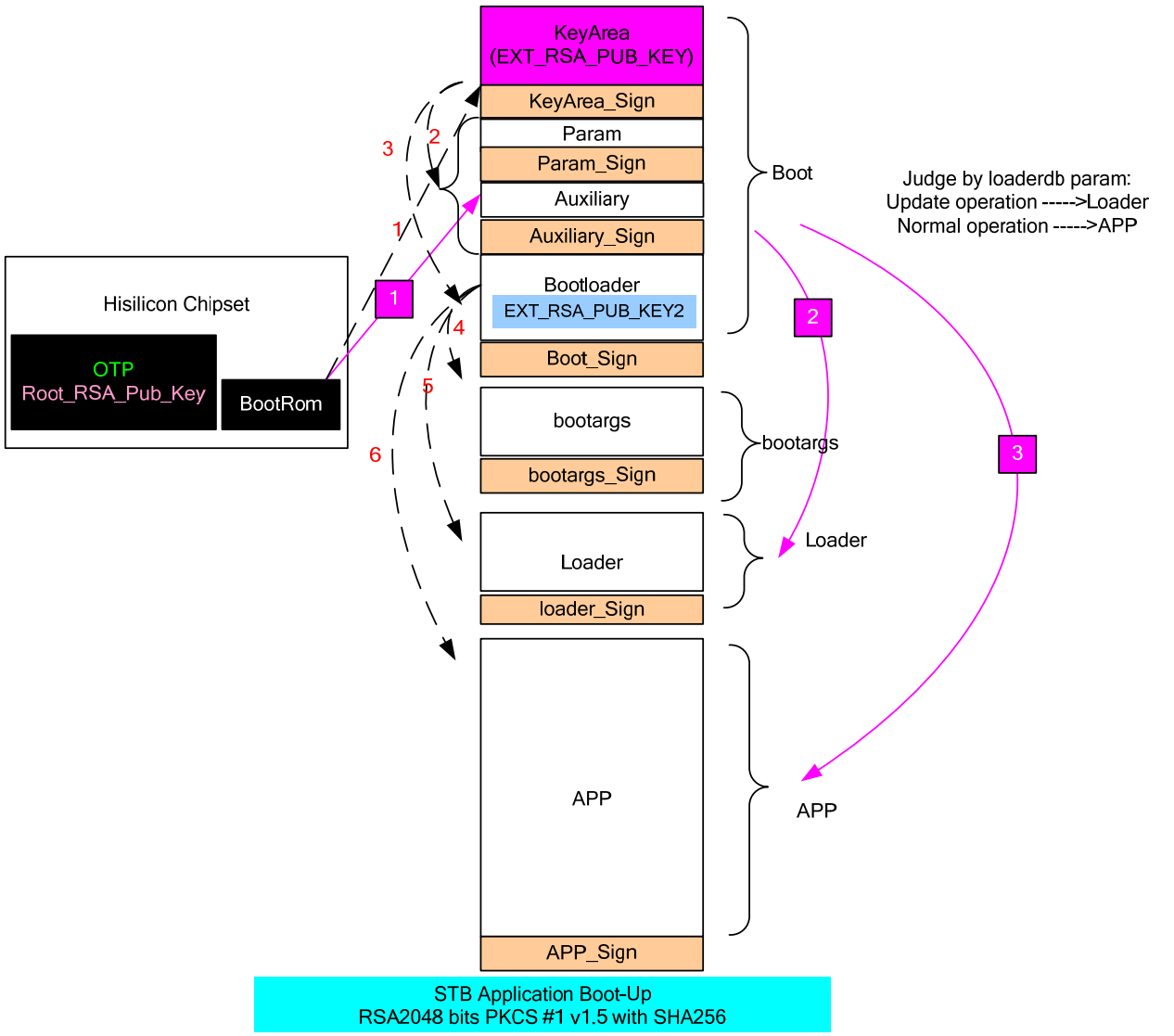
为了防止上述安全漏洞，机顶盒需要提供一种有效的保护系统不被非法篡改的方案。即安全启动方案。

海思的高安芯片提供完整的安全启动解决方案。安全启动方案能够保护机顶盒在运行中，只执行合法的程序。未经运营商或者机顶盒厂商签名的程序不允许在机顶盒的运行环境中执行起来。

业界对程序的校验是采用签名校验算法。海思推荐使用 RSA2048 bits PKCS #1 v1.5 with SHA256 算法执行数字签名与校验。

图 2-1 是机顶盒启动的流程图。

图2-1 机顶盒启动基本流程图



高安芯片出厂时，在 OTP 内保留一个由 CA 公司提供的 ROOT_RSA_PUB_KEY。在 Flash 上的 BOOT 保存 EXT_RSA_PUB_KEY，BOOT 和 APP 以及他们的签名数据。

高安芯片启动后，分别可能执行片外 BOOT 程序，执行 Loader 程序和执行 APP 程序。下面分为 3 个步骤对程序执行过程进行说明：

步骤 1 机顶盒上电启动，主芯片首先执行固化在芯片内的 BootRom 代码，BootRom 从 OTP 获取到 ROOT_RSA_PUB_KEY，校验 BOOT 代码的 KeyArea 区域的签名。校验通过后，可以获得 KeyArea 区域内的 EXT_RSA_PUB_KEY。BootRom 先使用 EXT_RSA_PUB_KEY 校验 Param 区和辅助代码区(Aux-Code)数据，然后使用 Param 数据和辅助代码区代码执行 DDR 初始化等操作，再把 Boot 区数据加载到 DDR 中用 EXT_RSA_PUB_KEY 校验，校验 OK 跳转到 Boot 程序执行。

步骤 2 BOOT 程序执行过程中，如果查找位于 Flash 的 loaderdb 分区（STB 厂商应自行定义该分区）的升级标志位，如果需要升级，校验 Loader 的签名，校验通过后跳转到 Loader 执行软件升级操作。校验用的密钥，可以是 boot 镜像 KeyArea 的

EXT_RSA_PUB_KEY，也可以是 boot 代码中的 EXT_RSA_PUB_KEY2，用哪个密钥取决于 CA 公司或运营商要求。。

步骤 3 BOOT 程序执行过程中，如果无法查找 loaderdb 的升级标识，或者升级标识未使能，BOOT 将校验 APP 的签名。校验通过后，程序跳转到 APP 执行。海思方案中，APP 包含了 kernel 和 FileSystem 等部分。校验用的密钥，可以是 boot 镜像 KeyArea 的 EXT_RSA_PUB_KEY，也可以是 boot 代码中的 EXT_RSA_PUB_KEY2，用哪个密钥取决于 CA 公司或运营商要求。

----结束

可见，安全启动方案分为 3 个部分：

- BOOT 镜像安全启动
- 非 BOOT 软件校验
- 安全软件升级

2.3 BOOT 镜像安全启动

本节描述高安芯片所使用的 BOOT 镜像结构，安全 BOOT 启动流程，以及打开安全启动等。这章节属于是安全启动方案的一部分。

2.3.1 BOOT 镜像分类

BOOT 是指主芯片上电后，直接执行的片外 Flash 上的第一个应用程序镜像。业界也将其命名为 BootLoader。

非高安芯片和高安芯片所使用的 BOOT 在结构上有差异。非高安芯片所用的 BOOT 称为普通 BOOT；高安芯片所用的 BOOT 称为安全 BOOT。

说明

如果没有特殊说明，本文中所述的 BOOT 特指高安芯片所用的安全 BOOT。

基于芯片对 BOOT 的要求来划分，芯片可分为 3 类：

- 普通芯片。直接使用 SDK 包中生成的普通 BOOT。
- 高安全系列（安全启动未使能）芯片。使用安全 BOOT，不用签名。
- 高安全系列（安全启动使能）芯片。使用安全 BOOT，必须签名（一般由 CA 厂商进行签名）。

表2-1 3 种不同 BOOT 的适用范围。

BOOT 类型	普通芯片	高安全芯片，安全启动未使能	高安全芯片，安全启动使能
普通 BOOT	√	×	×
安全 BOOT，未签名	×	√	×
安全 BOOT，已签名	×	√	√

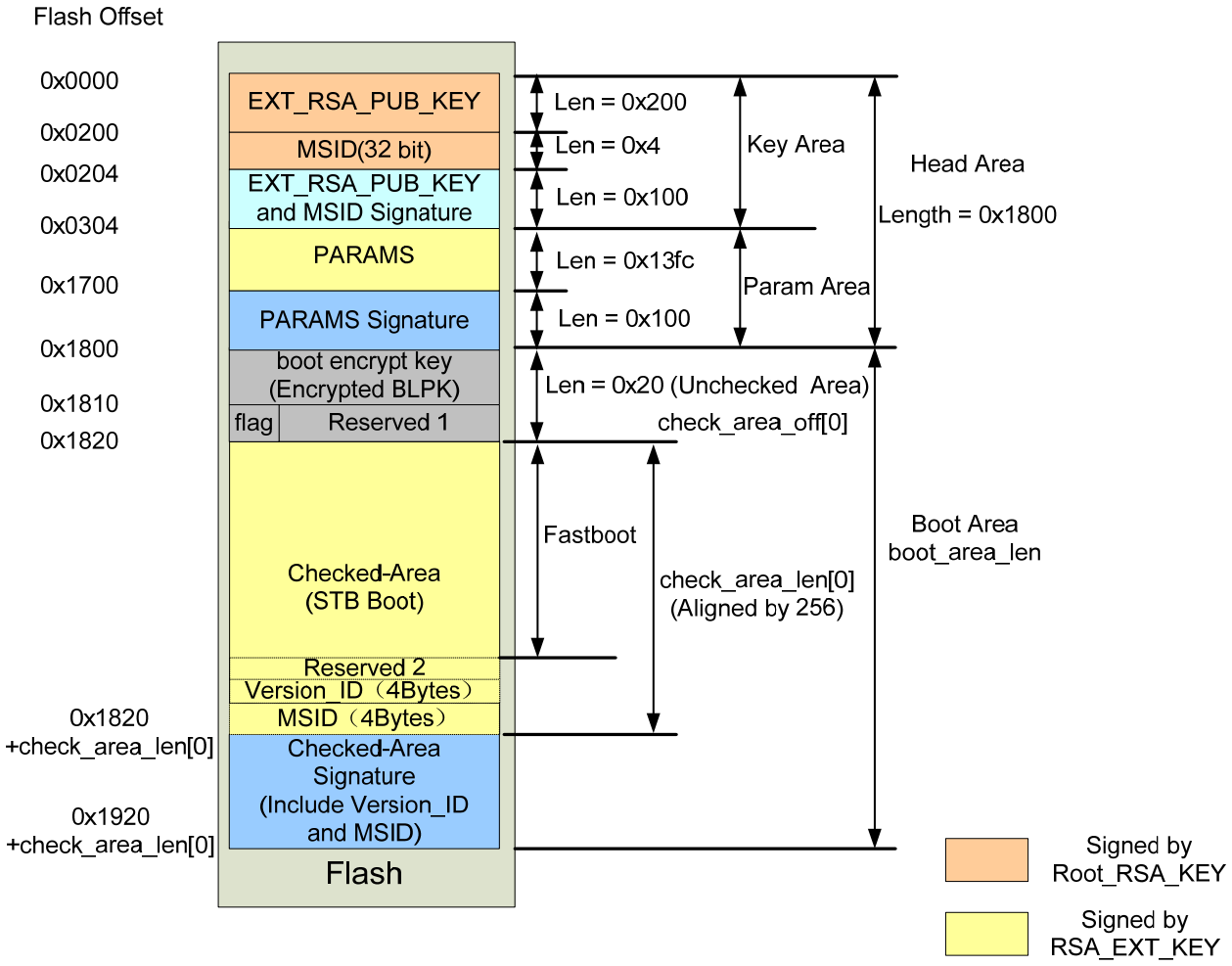
2.3.2 安全 BOOT 的 Flash 内部存储结构

说明

Hi3716MV300、Hi3716MV400、 Hi3716MV310、Hi3110EV500、Hi3716CV200、Hi3719CV100、Hi3719MV100、Hi3796CV100、Hi3798CV100、Hi3798MV100、Hi3796MV100、Hi3716MV420、Hi3716MV410、Hi3716MV330。**使用相同的存储结构。**

以 Hi3716MV310 为例，安全 BOOT 在 Flash 的内部存储结构如图 2-2 所示。

图2-2 安全 BOOT 在 Flash 的内部存储结构



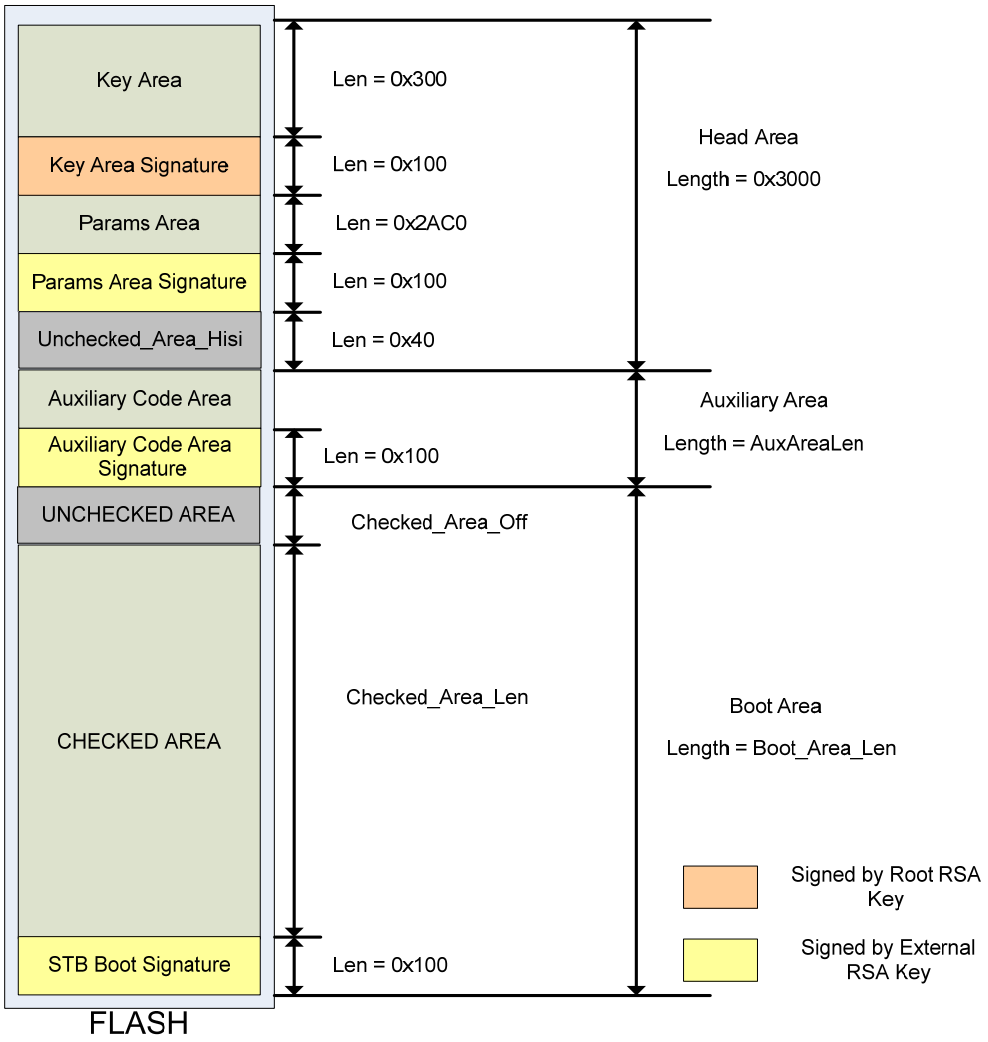
从图 2-2 中可以看出，安全 BOOT 主要分为 2 个部分：

- Head Area 区：
  - Key Area：包含 External RSA Public Key、MarketSegmentID 及其签名。
  - Param Area：包含 DDR 的一些配置数据及其签名。
- Boot Area 区：

- Checked Area：包含 Boot code 及其签名。
- Uncheck Area：用于保存 CA 厂商的一些私有数据，所在的 Flash 区域不需要签名。

Hi3798CV200 及其以后芯片采用新的 BOOT 结构，新 BOOT 在 Flash 的内部存储结构如图 2-3 所示。

图2-3 新 BOOT 在 Flash 的内部存储结构



从图 2-3 可以看出，新安全 BOOT 主要分为 3 个部分：

- Head Area 区：
  - Key Area：包含 External RSA Public Key、MarketSegmentID、辅助代码区地址和长度及其签名。
  - Param Area：包含一些配置数据及其签名。
- Auxiliary Code Area 区：
  - 包含辅助代码区的一些配置和信息。

- Boot Area 区：
  - Checked Area：包含 Boot code 及其签名。
  - Uncheck Area：用于保存 CA 厂商的一些私有数据，所在的 Flash 区域不需要签名。

2.3.3 BOOT 镜像设置 MSID

MSID（Market Segment ID）是特定 CA 厂商需要提供的 BOOT 启动安全控制位。MSID 主要用于区分不同的市场，MSID 的大小为 4 个字节。

MSID 存储于 OTP 和 Flash 中，存储于 OTP 中的 MSID 称为$(MSID)_{OTP}$，存储于 Flash 中的 MSID 称为$(MSID)_{Flash}$。在芯片安全启动过程中，$(MSID)_{OTP}$ 必须与$(MSID)_{Flash}$ 一致才能正常启动。

- $(MSID)_{OTP}$ 是产线生产时厂测程序调用对应 UNF 接口写入 OTP 中。
- $(MSID)_{Flash}$ 是使用 CASignTool 在生成安全 BOOT 时设置，写进安全 BOOT 中。

设置$(MSID)_{OTP}$ 方法如下：

步骤 1 调用 HI_UNF_ADVCA_Init 初始化高安全 CA 模块。

步骤 2 调用函数 HI_S32 HI_UNF_ADVCA_SetMarketId()设置 MSID。

步骤 3 调用 HI_UNF_ADVCA_DeInit 去初始化。

----结束

设置$(MSID)_{Flash}$ 的方法如下：

$(MSID)_{Flash}$ 的数值是通过签名工具 CASignTool 在生成安全 BOOT 时设置的。

说明

设置$(MSID)_{Flash}$ 时，需要关注该数据的字节序，请参考《CASignTool 使用说明》的“设置市场区域码“进行设置。

2.3.4 安全 BOOT 的开发流程

本节描述安全 BOOT 的开发，签名和使用。操作步骤如下：

步骤 1 生成不带签名的安全 BOOT。

1. 在编译 SDK 的 BOOT 时，通过命令”make menuconfig”,配置以下选项（如下操作将会修改$SDK/cfg.mak 中的配置），包括：

- 选项“CFG_HI_ADVCA_SUPPORT=y”。
- 设置的 CA 厂商类型。如果需要编译 ConaxCA，选择“CFG_ADVCA_CONAX=y”。

2. 在 SDK 执行编译 BOOT 的命令，可以直接产生 fastboot-burn.bin。
3. 使用 CASignTool 工具对上述 fastboot-burn.bin 执行签名，就可以产生高安 BOOT。

STB 厂商在开发 BOOT 阶段，如果安全启动标记未使能，可以选择使用自签名的高安 boot 开发调试。

说明

“CFG_FUNCTION_DEBUG”,” CFG_FUNCTION_RELEASE”和“CFG_FUNCTION_FINAL”是某些 CA 厂商对 SDK 编译的镜像特殊要求。目前只有 Conax，Nagra CA 在开发过程中可能需要使用上述编译选项。

编译 SDK 时，STB 厂商如果打开了上述 3 个选项中的其中一个。此时编译 BOOT 时所产生的 fastboot-burn.bin 文件不包括网络和自举功能。这时，STB 厂商不能使用 HiTool 工具直接下载该签名后的 BOOT。在这种情况下，STB 厂商可以在 SDK 目录直接使用“make advca_programmer”命令，编译出 advca_programmer.bin。该镜像签名后，自举所用的 programmer 文件。

Hi3798CV200 安全 BOOT 和非安全 BOOT 结构统一，SDK 编译出来的镜像即为不带签名的安全镜像，可直接用于烧写。

步骤 2 使用 HiTool 下载工具将高安 BOOT 下载到 Flash 中。安全 BOOT 已经正确的下载到 Flash 后，单板应该能够启动了。

说明

HiTool 使用时，芯片类型需要选择后缀名包含“_CA”的选项型号。此时高安芯片的安全启动标志位没有打开，芯片仍然不会校验 BOOT 签名。机顶盒厂商完成 BOOT 开发测试之后，才打开高安芯片的安全启动标志位。

Hi3798CV200 安全 BOOT 和非安全 BOOT 结构统一，芯片类型不区分 CA 和非 CA，只需要选择 Hi3798CV200 即可。

步骤 3 经过以上几个步骤，STB 厂商可以将 BOOT 的相关文件（如 KeyArea.bin、ParamArea.bin 和 BootArea.bin）提交给 CA 厂商签名，得到签名后，再使用 CASignTool 的 Merge Signed-BootImage 功能合并上述镜像，获取正式签名 BOOT(也称为 FinalBoot.bin)。

使用下载工具将步骤 3 生成的安全 BOOT 下载到 Flash 中。断电重启，单板就可以正常启动。

说明

此时芯片的安全启动功能尚未使能，经过步骤 2 的验证，可以确保此时芯片应该能够正常启动。

在执行下一步之前，STB 厂商可以将正式签名的 BOOT 发给海思，海思协助 STB 厂商确认改镜像是否已经正确签名。如果正确，可以执行下一步操作；否则，重新步骤 3。

步骤 4 使能芯片的安全启动功能，断电后重启，检查是否能够正常启动

----结束

2.4 非 BOOT 软件安全校验

2.4.1 安全校验过程

由于芯片启动时已经完成了对 BOOT 镜像的校验，所以在安全 BOOT 中，需要完成对非 BOOT 软件的签名校验后，再执行非 BOOT 软件。从而可以确保机顶盒所执行的程序都是合法的。

非 BOOT 软件也称之为系统软件，对应的镜像签名称之为非 BOOT 镜像签名，也称为系统软件签名。

非 BOOT 软件分为 3 类：

- 第一类：关键数据

  例如 BootArgs。BootArgs 是 Linux 操作系统环境下，用于 BOOT 启动中参数配置和启动 Kernel 所需参数的命令集合。BootArgs 虽属于非代码数据，但其决定系统启动的流程。如果存在，通常要求 BootArgs 必须签名。BOOT 运行中必须校验 BootArgs 的签名后，才能执行 BootArgs 的参数配置。BootArgs 的处理有 2 种方法：

  - 直接将 BootArgs 参数写入 BOOT 代码内。这种做法代码缺少灵活性，其最大好处省去校验该数据的操作。
  - 可以使用海思提供的 mkbootargs 工具，使用预先设定的 bootargs 参数产生 bootargs.bin 文件，再对该镜像执行签名。

- 第二类：普通系统软件

  此类镜像文件包括 kernel、Loader、application 等。

- 第三类：文件系统软件：

  文件系统对应的镜像通常比较大，按照安全启动要求，需要先校验其签名后，再挂载文件系统。STB 厂商应该优先使用只读文件系统，例如 quashfs 等。

  开发阶段不需要签名校验时，STB 厂商可以使用 jffs2、yaffs2 等可读写文件系统。。那么在配置 kernel 启动参数时，必须添加参数：“ro”。例如：

```
setenv bootargs 'mem=96M console=ttyAMA0,115200 root=/dev/mtdblock2
rootfstype=yaffs2 ro
mtdparts=hi_sfc:1M(Boot);hinand:5M(Boot),60M(rootfs)ro,-(others)
mmz=ddr,0,0x86000000,160M'
```

注意

如果使用可读写文件系统，不要在文件系统运行起来后，再使用 remount 等命令再修改文件系统。

非 BOOT 软件安全校验的过程为：

步骤 1 安全 BOOT 需要将非 BOOT 软件签名镜像读取到内存。如果非 BOOT 软件签名镜像是加密存储的，安全 BOOT 需要先解密该非 BOOT 软件镜像。

步骤 2 安全 BOOT 使用配置在 BOOT 代码中的 EXT_RSA_PUB_KEY，也可以是 boot 代码中的 EXT_RSA_PUB_KEY2 对步骤 1 中的镜像逐一校验。所有校验结束后，应用程序可以跳转到系统软件执行。

----结束

2.4.2 非 BOOT 软件签名模式分类

海思提供 2 种非 BOOT 软件签名模式：

- 模式 1：非 BOOT 文件特定签名。请参考《CASignTool 使用说明》“非 BOOT 文件特定签名模式”。
- 模式 2：非 BOOT 文件通用签名。请参考《CASignTool 使用说明》“非 BOOT 文件通用签名模式”。

2.4.2.1 非 BOOT 文件特定签名模式

应用场景

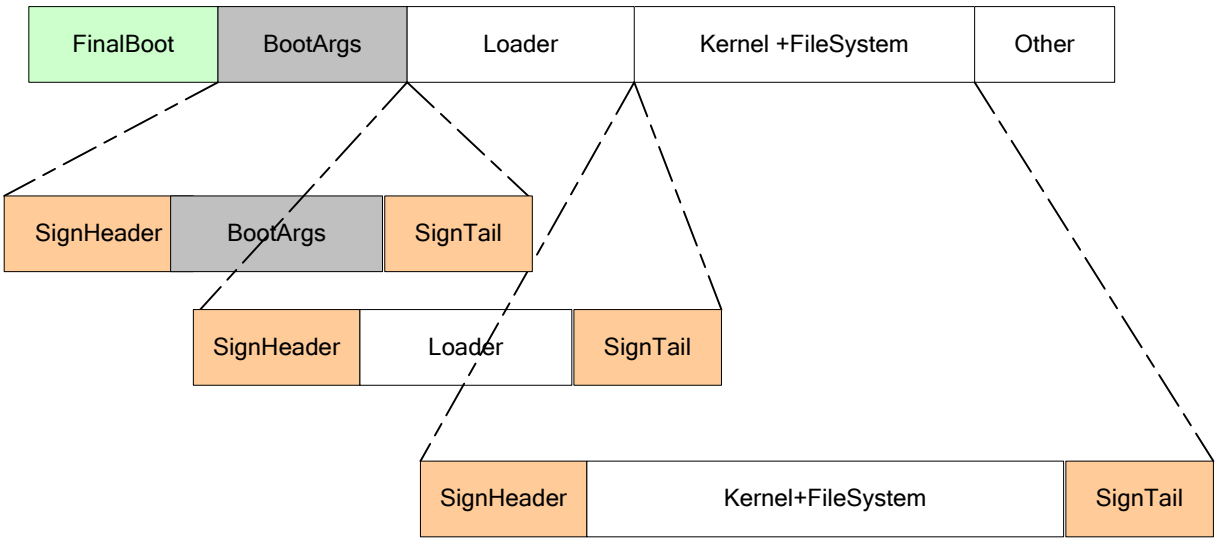
非 BOOT 文件特定模式签名通常用于 FLASH 存放的镜像需要加密的场景。

- 优点：快速查找到非 BOOT 软件的签名。快速判断该非 BOOT 软件是否被加密。
- 缺点：在非 BOOT 软件前添加一个数据头，影响 BOOT 参数段，内核和文件系统的使用。使用该镜像时需要考虑签名头部数据的偏移。

  例如，如果非 BOOT 文件镜像被拷贝到内存中的某个特定地址(ADD)，那么该镜像的实际地址为：ADD+0x2000。

非 BOOT 文件特定签名的结构如图 2-4 所示。

图2-4 非 BOOT 文件特定签名的结构图



以 Hi3716MV310 为例，系统安全启动的相关代码位于 $(SDK_Linux)\source\boot\product\driver\advca 下。请在 boot\product\driver\advca\common\auth 路径下，修改 ca_config.c 文件的内容。

其中：

- g_EnvFlashAddr：bootargs 等 BOOT 参数存放在 flash 中的位置。
- g_EnvBakFlashAddr：bootargs 备份区在 flash 中的位置。（如果不需要对 bootargs 进行校验，可以不配置 g_EnvFlashAddr 和 g_EnvBakFlashAddr）。
- g_customer_rsa_public_key_N[256]：客户使用的 RSA_public_key 的 N 值。
- g_customer_rsa_public_key_E：客户使用的 RSA_public_key 的 E 值，该参数建议配置为如下值：
  - g_customer_rsa_public_key_E = 0x10001；
  - g_customer_rsa_public_key_E = 0x3；
- isCipherkeyByCA：是否需要使用硬件 cipher 加密，也就是使用 R2R Key-Ladder 加密。
- g_CipherAlg：cipher 算法。
- g_AuthAlg：签名使用的 HASH 算法，请一定要与工具制作签名镜像时使用的算法配置成一致。

操作步骤

非 BOOT 文件特定签名模式的操作步骤如下：

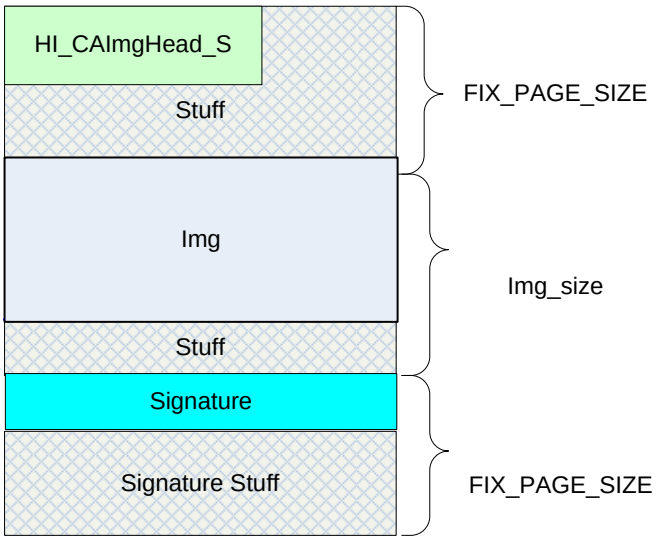
步骤 1 重新编译 BOOT，执行 make hiboot_prepare、make hiboot_clean、make hiboot_install。

步骤 2 使用 CASignTool 工具，制作签名软件镜像，烧写到对应的 flash 分区。即可对这些分区的软件做安全校验。

制作签名镜像的方法参考《CASignTool 使用说明》的描述。CASignTool 工具生成签名支持针对单个文件生成签名镜像，也支持把多个文件合并后，再生成最后的签名镜像。

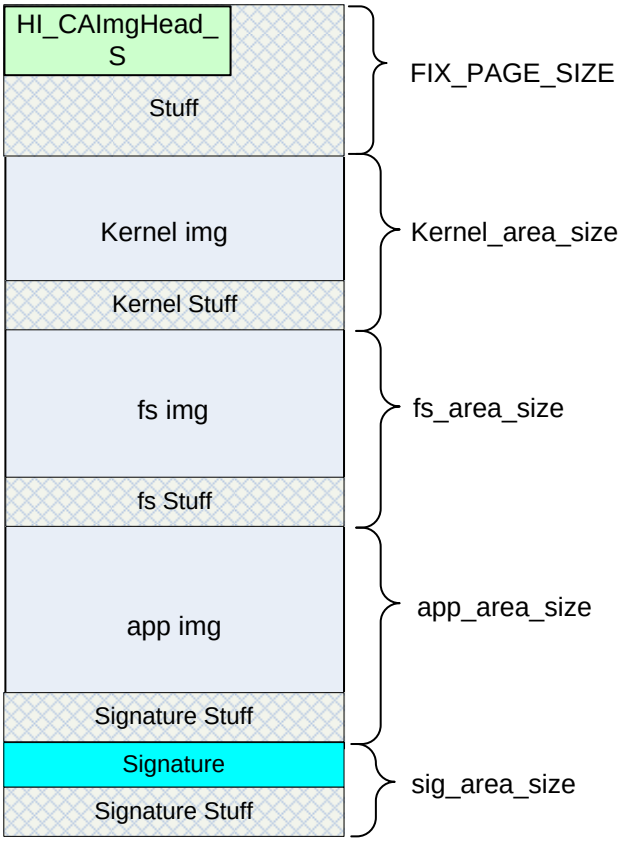
- 如果对单个文件进行签名，生成的签名镜像格式如图 2-5 所示。

图2-5 非 BOOT 软件（单个）签名镜像结构图



- 如果对多个文件合并后进行签名（以 kernel、fs、app 为例），生成的签名镜像格式如图 2-6 所示。

图2-6 非 BOOT 软件（多个）签名镜像结构图



步骤 3 挂载内存文件系统，比如 squashfs 内存文件系统（某些高安系统要求使用只读文件系统，并且只能使用内存文件系统）。

- 编译支持挂载内存文件系统的 kernel，修改内核配置“CONFIG_BLK_DEV_RAM=y”，再重新编译内核。
- 修改 bootargs 内容，使支持挂载内存文件系统，并且将 ramdisk size 改为 40MB，例如：

```
set bootargs 'mem=96M console=ttyAMA0,115200
initrd=0x82400000,0xc00000 root=/dev/ram ramdisk_size=40960
mtdparts=hinand:1M(Boot), 1M(bootpara),
1M(bootparabak),40M(sys),1M(baseparam),1M(logo),-(others)
mmz=ddr,0,0x86000000,
160M DmxPoolBufSize=0x200000';save
```

注意

红色部分根据实际烧写镜像的情况修改，initrd 是指文件系统在内存中的起始地址和长度。例如，实际制作签名系统镜像（sys）的时候，该镜像为 kernel 和 fs 的组合，如果 fs 的偏移地址为 0x400000,大小为 0xc00000，由于系统镜像会被加载到内存 0x82000000 处，因此为了启动内存文件系统，则 bootargs 应该写为:

initrd=0x82400000,0xc00000.

FIX_PAGE_SIZE 为固定的 0x2000。

----结束

命令说明

安全 BOOT 镜像中已经提供了如表 2-2 所示的测试命令，可以在单板上用于验证非 BOOT 软件数据签名。

表2-2 可用于验证非 BOOT 文件特定签名的命令

BOOT 程序内包含的命令	命令介绍	用法
CX_EncryptBurnImage	该命令使用 R2R KeyLadder，对内存中存放的可执行的代码镜像数据执行加密，加密后的镜像写入指定 FLASH 分区。 上述的可执行的代码镜像必须是非 BOOT 文件特定签名镜像的结构。 该命令与前期 SDK 版本中的 ca_special_burnflashname 相同。	CX_EncryptBurnImage DDRAddress FlashPartition 参数 DDRAddress 表示内存中存放的可执行的代码镜像数据的初始地址； 参数 FlashPartition 表示加密后的镜像写入到指定的 FLASH 分区名。
CX_EncryptBurnData	该命令使用 R2R KeyLadder，对内存中存放的数据镜像数据执行加密，加密后的镜像写入 FLASH。 上述的数据镜像可以是内存中的一块数据。 该命令与前期 SDK 版本中的 ca_special_burnflashnamebylen 相同。	CX_EncryptBurnData DDRAddress ImageLen FlashPartition 参数 DDRAddress 表示内存中存放的数据镜像数据的初始地址； 参数 ImageLen 表示内存中存放的数据镜像数据的大小； 参数 FlashPartition 表示加密后的镜像写入到指定的 FLASH 分区名。
cx_decrypt_partition	该命令将指定 FLASH 分区的镜像数据解密到固定内存地址。 该命令与前期 SDK 版本 ca_decryptflashpartition 相同	cx_decrypt_partition FlashPartition 参数 FlashPartition 表示指定 FLASH 分区名
cx_verify	该命令用于校验指定 FLASH 分区的镜像。 该命令与前期 SDK 版本 ca_special_verify 相同。	cx_verifyFlashPartition 参数 FlashPartition 表示指定 FLASH 分区名
verify_test_ex	该命令用于校验指定 FLASH 偏移量地址的镜像。 该命令与前期 SDK 版本 ca_special_verifyaddr 相同。	verify_test_exFlashAddOffset 参数 FlashAddOffset 表示指定 FLASH 偏移量地址。

BOOT 程序内包含的命令	命令介绍	用法
cx_verify_bootargs	该命令用于校验 bootargs 分区。 该命令与前期 SDK 版本 ca_special_verifybootargs 相同	cx_verify_bootargs 该命令没有参数。

2.4.2.2 非 BOOT 文件通用签名模式

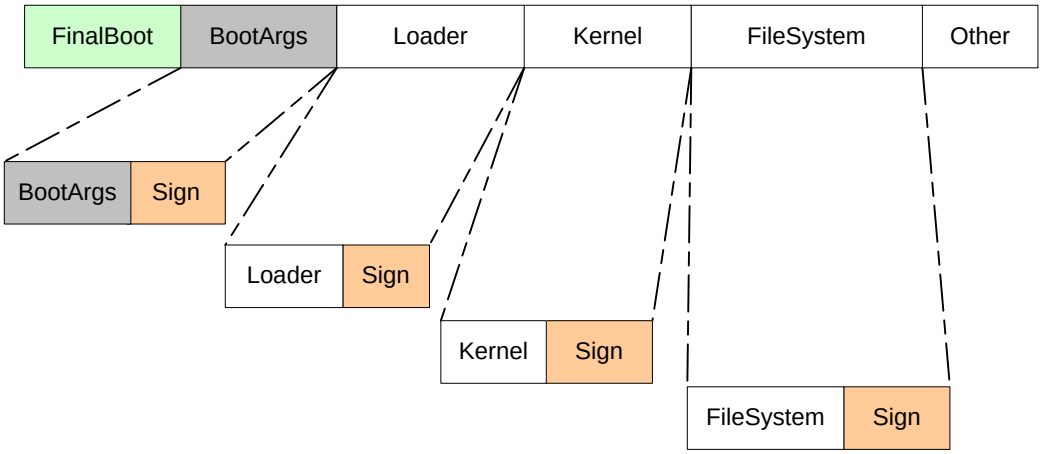
应用场景

该模式主要运用于对于系统安全要求不高的运营商市场。此种场景下，如果允许非 BOOT 软件可以从 Flash 直接加载运行，或者允许使用明文的非 BOOT 软件镜像，STB 厂商可以使用此模式。

- 优点：调试和实际使用可以使用同样的 Flash 存储格式。可以灵活的放置数据签名。
- 缺点：查找非 BOOT 软件的签名需要特别处理。

非 BOOT 文件通用签名的结构如图 2-7 所示。

图2-7 非 BOOT 文件通用签名的结构图



以 Hi3716MV310 为例，系统安全启动的相关代码位于 boot\product\driver\advca 下。请在 boot\product\driver\advca\common\auth 路径下，修改 ca_config.c 文件的内容。

其中：

- g_EnvFlashAddr：bootargs 等 BOOT 参数存放在 flash 中的位置。
- g_EnvBakFlashAddr：bootargs 备份区在 flash 中的位置。
  如果不需要对 bootargs 进行校验，可以不配置 g_EnvFlashAddr 和 g_EnvBakFlashAddr。
- g_customer_rsa_public_key_N[256]：客户使用的 RSA_public_key 的 N 值。

- g_customer_rsa_public_key_E：客户使用的 RSA_public_key 的 E 值，请将该值配置为以下两种中的一种。
  - g_customer_rsa_public_key_E = 0x10001；
  - g_customer_rsa_public_key_E = 0x3；
- isCipherkeyByCA：表示是否需要使用硬件 cipher 加密，也就是使用 R2R 加密。如果 STB 厂商同时希望使用加密功能，我们建议使用 2 个分区分别存放加密镜像文件和签名。
- g_CipherAlg：cipher 算法。
- g_AuthAlg：签名使用的 HASH 算法，请一定要与工具制作签名镜像时使用的算法配置成一致。

操作步骤

非 BOOT 文件通用签名模式的操作步骤如下：

步骤 1 重新编译 BOOT，执行 make hiboot_prepare、make hiboot_clean、make hiboot_install。

步骤 2 使用 CASignTool 工具，制作签名软件镜像，烧写到对应的 flash 分区。即可对这些分区的软件做安全校验。

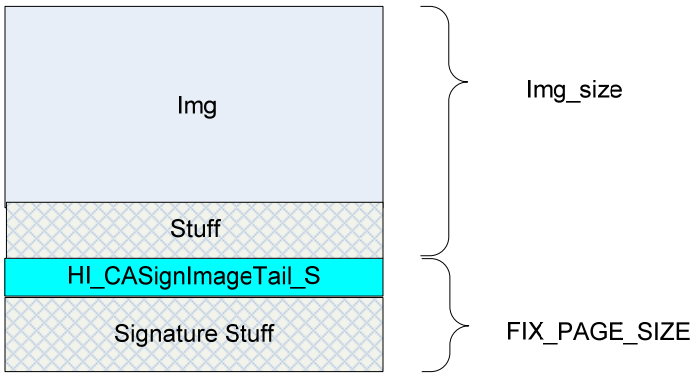
制作签名镜像的方法参考《CASignTool 使用说明》的描述。CASignTool 工具一次可产生文件与签名合并的镜像和独立的签名镜像。

注意

如果使用 CASignTool 工具对 yaffs 文件镜像进行签名，其签名结果仅产生签名镜像。

文件进行签名生成的签名镜像格式如图 2-8 所示。

图2-8 文件进行签名生成的签名镜像格式



----结束

为了快速获得镜像文件的数据签名。BOOT 代码提供了一个变量 signature_check，其格式为：

```
sign:Image,Image_Signature, Image_SignatureOffset
```

- 如果变量 Image_Signature 与 Image 值相同，表示被签名镜像与签名被放在同一个分区；
- 如果变量 Image_Signature 与 Image 值不相同，表示被签名镜像与签名被放在不同一个分区。

SignatureOffset 用于快速查找出 Image_Signatur 分区中偏移为 SignatureOffset 的后面的签名数据，数据按照 0x2000 对齐方式查找。例如：

```
setenv signature_check 'sign:kernel,kernel_sign,0x0 sign:fs,fs_sign,0x00
sign:subfs,subfs_sign,0x00'
setenv signature_check 'sign:kernel,kernel,0x28000 sign:fs,fs,0x600000
sign:subfs,subfs,0x400000'
```

使用方法请参考下面的“ca_common_verify_signature_check”。

命令说明

安全 BOOT 镜像中已经提供了如下的测试命令，可以在单板上用于验证非 BOOT 软件数据签名。

表2-3 可用于验证非 BOOT 文件通用签名的命令

BOOT 程序内包含的命令	命令介绍	用法
ca_verify	该命令用于验证指定 FLASH 分区镜像的签名，并把该镜像数据拷贝到固定内存地址。 该命令与前期 SDK 版本中的 ca_common_verify_image_signature 相同	ca_verify image_partition_name sign_partition_name sign_offset_partition 参数 image_partition_name 表示将被验证的 FLASH 分区镜像名； 参数 sign_partition_name 表示签名数据存放的 FLASH 分区名； 参数 sign_offset_partition 表示签名数据在 FLASH 分区的地址偏移量

BOOT 程序内包含的命令	命令介绍	用法
ca_verify_by_env	该命令是一个批处理执行多个镜像签名校验的命令； STB 厂商应该使用”signature_check”变量先设置需要批处理执行多个镜像签名校验的各项参数。 该命令与前期 SDK 版本中的 ca_common_verify_signature_check 相同	ca_verify_by_env 该命令没有参数。
ca_verify_bootargs_by_env	该命令用于校验 bootargs 分区签名。 该命令与前期 SDK 版本中的 ca_common_verify_bootargs 相同。	ca_verify_bootargs_by_env 该命令没有参数。
advca_verify	该命令用于校验指定的分区的签名。	advca_verifypartition_name 参数 partition_name 表示待校验分区的名字。

2.4.3 参考代码

SDK 中非 BOOT 软件签名的参考代码位于 Boot 目录下，不同 SDK 版本代码存放目录有所差别。

以 Hi3716MV310 为例，BOOT 下的安全校验接口定义在 ca_authenticate.h 文件中。

如果要进行非 BOOT 软件安全校验，修改 boot\product\main.c 文件，在 fastapp_entry () 函数内添加安全校验接口。

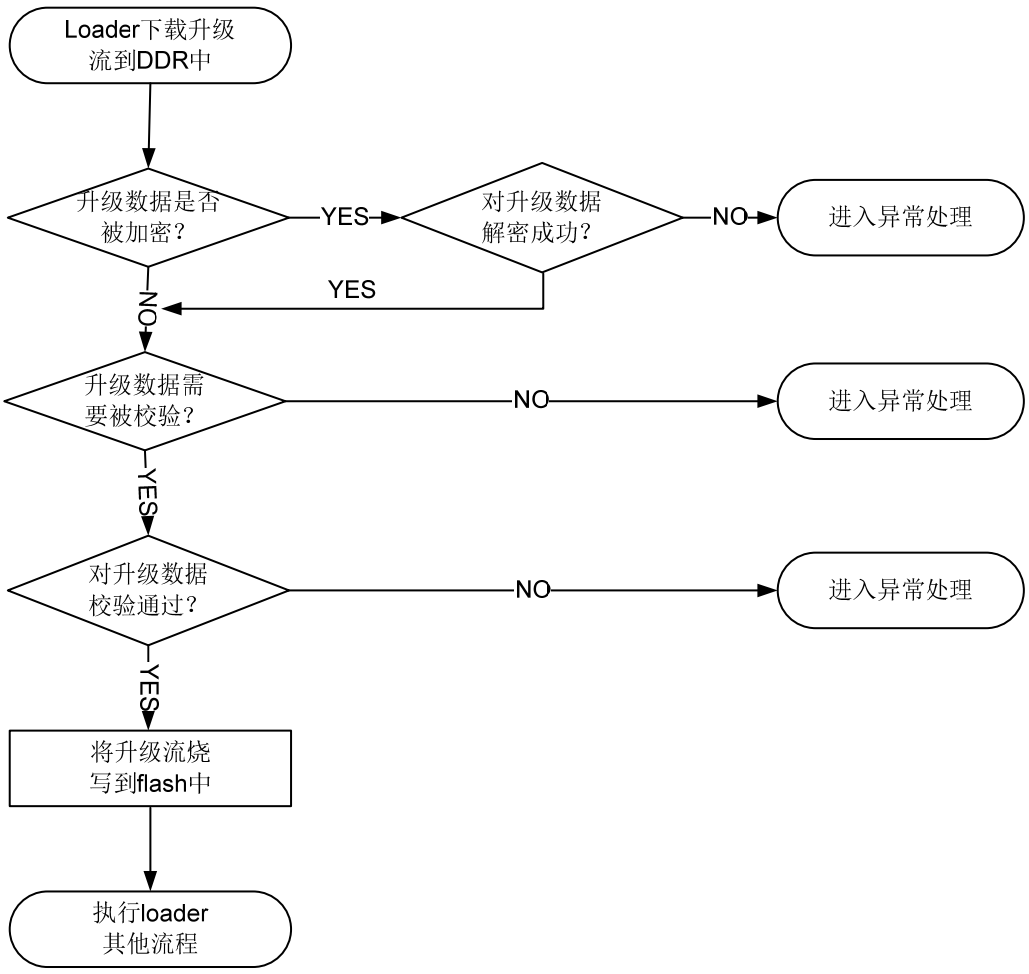
2.5 安全软件升级

安全软件升级属于安全启动方案的一部分。

由于高安对软件的安全性要求，在升级中需要对升级软件的执行校验。如果运营商或者 CA 厂商还要求对升级镜像执行加密传输，则需要对升级镜像进行加密处理。

安全软件升级过程如图 2-9 所示。

图2-9 安全软件升级流程图



以 Hi3716MV310 为例，安全软件升级步骤如下：

步骤 2 配置安全校验环境。

请修改 loader 下的 cx_config.c 文件的内容。

注意

安全软件升级和“2.4 非 BOOT 软件安全校验

安全校验”章节描述的非 BOOT 软件安全校验需要配合使用，两种场景下使用的校验算法和加密解密算法必须一致。

- g_customer_rsa_public_key_N[256]：客户使用的 RSA_public_key 的 N 值。
- g_customer_rsa_public_key_E：客户使用的 RSA_public_key 的 E 值，该值建议配置为如下两种中的一种。
  - g_customer_rsa_public_key_E = 0x10001;

- g_customer_rsa_public_key_E = 0x3；

- isCipherkeyByCA：是否需要使用硬件 cipher 加密，也就是使用 R2R 加密。
- g_CipherAlg：cipher 算法。
- g_AuthAlg：签名使用的 HASH 算法，请一定要与工具制作签名镜像时使用的算法配置成一致。

步骤 3 编译带高安功能的 loader。

- 修改 cfg.mk 文件，确保 CFG_ADVCA_TYPE 有定义(非 nagra)。
- 编译：在 SDK 顶层路径执行 make hiloader_build。

步骤 4 使用 CASignTool 工具，制作签名软件镜像，然后再用 loader 的打包工具制作升级流。制作签名镜像的方法参考《CASignTool 使用说明》中的描述。

----结束

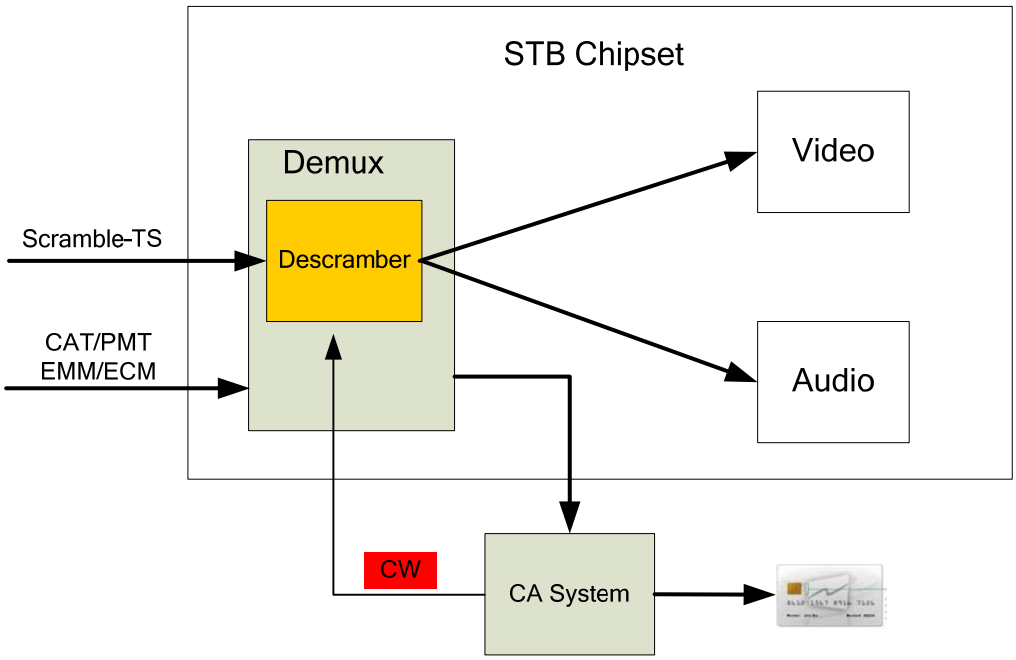
2.6 TS 流解密

TS 码流解密是 DVB 核心业务。TS 码流经过调制解调器(Tuner)将模拟信号调制为数字信号，通过解复用模块（Demux）过滤出有效的音视频 TS 流数据，传给音视频解码器，最后将音视频信号输出到电视机。

如果输入到 Demux 的 TS 流是加密的数据。该数据需要经过解扰器(Descrambler)解扰为明文的 TS 流。该过程由条件接收系统(CA)控制。该过程的核心操作为：CA 系统通过接收和解析 CAT、ECM、EMM 私有数据流，将这类数据流传递给智能卡(SmartCard)解析后，获取到加扰 TS 流的控制字(CW)。

CA 系统再将 CW 设置到 Descrambler，从而解出明文的 TS 流。

图2-10 DVB TS 流解密示意图



当前机顶盒业务所面临的安全威胁主要是：用于解密码流的控制字(CW)被黑客盗取，并通过各种手段分发到非法市场上，扰乱正常的运营市场。

所以，需要采取更有效的手段保护用于解密码流的控制字(CW)。这是高安 CA 最核心的业务就是实现采用 CA 厂商特有的加密方法对 CW 执行加密保护。

可见，高安 CA 与非高安 CA 的本质区别是：

- 非高安 CA 采用明文方式将加密 TS 码流的控制字(CW)传递到芯片的解扰器模块。

  普通芯片和高安芯片均可以实现明文 TS 流解密特性。
- 高安 CA 采用密文方式将加密 TS 码流的控制字(CW)传递到芯片的解扰器模块。

  只有高安芯片实现密文 TS 流解密特性。

2.6.1 密文控制字 CW 使用流程

场景说明

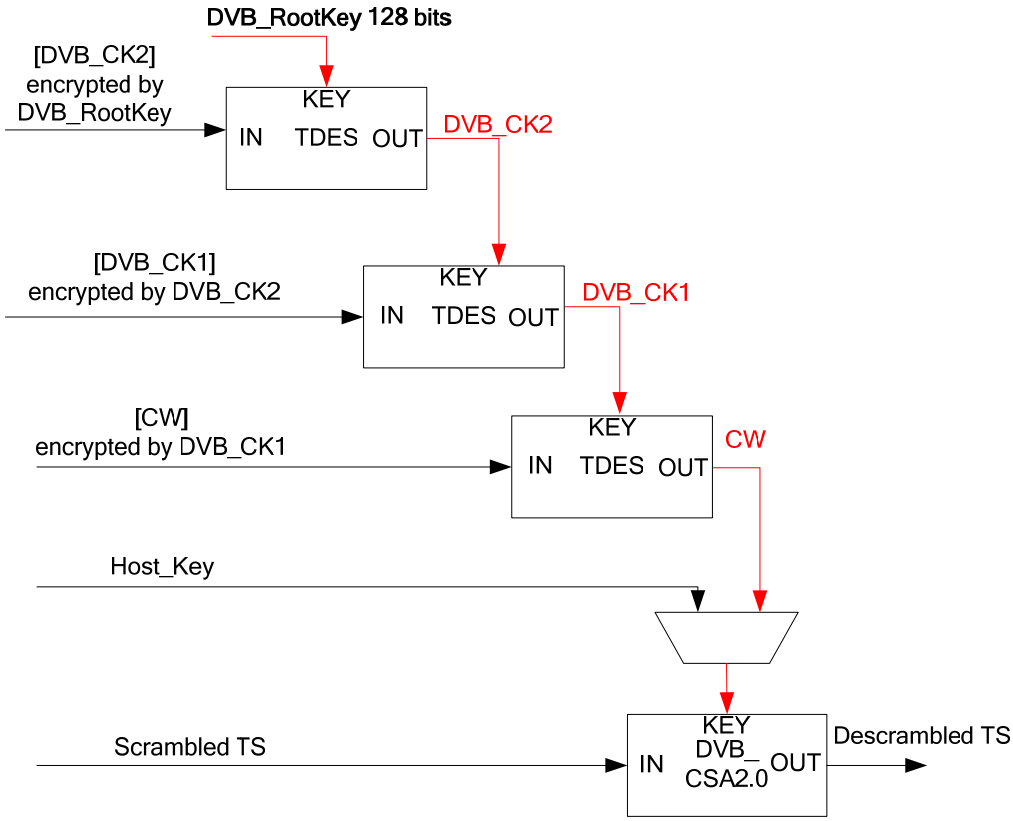
高安芯片内置一个 CW 字解密的根密钥（例如：DVB_RootKey）。控制字 CW 是经过多级密钥加密传输，加密级数可能是 2 级或 3 级，加密算法一般是 TDES 算法。

下面以 3 层加密为例，描述密文 CW 解密过程。

图中的密文 DVB_CK1/密文 DVB_CK2/密文 CW 统称为会话密钥(SessionKey)，由应用程序通过特定的函数接口设置到主芯片内。

解密流程图如图 2-11 所示。

图2-11 3 级密文 CW 解密流程图



高安 CA 设置控制字(CW)的流程如下：

步骤 1 STB 厂商向主芯片设置密文 DVB_CK2，主芯片读取 DVB_RootKey 解密获得明文 DVB_CK2。

步骤 2 STB 厂商向主芯片设置密文 DVB_CK1，主芯片使用明文 DVB_CK2 解密获得明文 DVB_CK1。

步骤 3 STB 厂商向主芯片设置密文 CW，主芯片使用明文 DVB_CK1 解密获得明文 CW。

步骤 4 主芯片得到明文 CW 后，将其设置进解扰器中，启动解扰。

----结束

操作步骤

创建播放器播放 TS 流的步骤请参考《HMS 开发指南》。这里重点介绍为视频通道配置高安全 CA 解扰器的步骤：

步骤 1 调用 HI_UNF_ADVCA_Init 初始化高安全 CA 模块。

步骤 2 调用 HI_UNF_AVPLAY_GetDmxVidChnHandle 获取视频 Demux 通道。

调用 HI_UNF_DMX_CreateDescramblerExt 创建使用高安全 CA 的解扰器。设置该函数的参数 enCaType 时，请选择 HI_UNF_DMX_CA_ADVANCE。

步骤 3 调用 HI_UNF_DMX_AttachDescrambler 将解扰器与视频通道绑定。

步骤 4 调用 HI_UNF_ADVCA_SetDVBAlg 设置控制字加密算法，默认为 TDES。

步骤 5 如果密钥级别是 3 级，调用 2 次 HI_UNF_ADVCA_SetDVBSessionKey 设置前 2 级加密的会话密钥。如果密钥级别是 2 级，调用一次 HI_UNF_ADVCA_SetDVBSessionKey 设置前 1 级加密的会话密钥。

步骤 6 调用 HI_UNF_DMX_SetDescramblerOddKey 设置密文的奇控制字。

调用 HI_UNF_DMX_SetDescramblerEvenKey 设置密文的偶控制字。

----结束

参考代码

请参考 advca 提供的用例：sample_ca_dvbplay.c

2.6.2 明文控制字 CW 使用流程

场景说明

高安芯片默认情况下可以支持明文控制字(CW)解密码流。解密流程请参考图 2-11 所示中使用 HostKey 的流程。

操作步骤

创建播放器播放 TS 流的步骤请参考《HMS 开发指南》。这里重点介绍为视频通道配置高安全 CA 解扰器的步骤：

步骤 1 调用 HI_UNF_AVPLAY_GetDmxVidChnHandle 获取视频 Demux 通道。

调用 HI_UNF_DMX_CreateDescramblerExt 创建使用高安全 CA 的解扰器。设置该函数的参数 enCaType 时，请选择 HI_UNF_DMX_CA_NORMAL。

步骤 2 调用 HI_UNF_DMX_AttachDescrambler 将解扰器与视频通道绑定。

步骤 3 调用 HI_UNF_DMX_SetDescramblerOddKey 设置明文的奇控制字。

调用 HI_UNF_DMX_SetDescramblerEvenKey 设置明文的偶控制字。

----结束

参考代码

请参考 advca 提供的用例：sample_ca_dvbplay.c

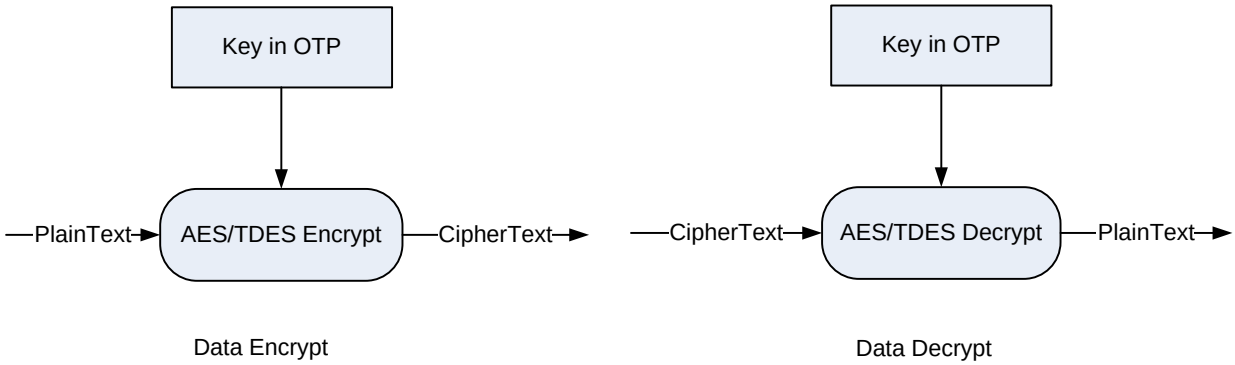
2.7 内容加解密方案

2.7.1 方案说明

机顶盒的存储空间可能保存着运营商/客户的私有数据。这类私有数据如果泄露，将会损坏运营商/客户利益。为了防止黑客通过非法途径获取到机顶盒信息，存放在 Flash 中的数据要求被加密存储。

如图 2-12 所示，是比较安全的一种数据加解密方案简图。

图2-12 数据加解密方案简图



STB 厂商可以选择将加密数据所用的密钥存放在主芯片的 OTP 区域内。位于 OTP 用于加解密所用的 Key 包括主 CPU 可以访问和主 CPU 不可以访问 2 种类型。

海思提供 2 种安全的内存数据加解密方式。分别描述如下：

- 使用随机数密钥对内存数据加解密。
- 使用固定密钥对内存数据加解密。

2.7.2 使用随机数密钥对内存数据加解密

场景说明

如果 STB 厂商要求每台机顶盒加密数据所用的密钥应该是唯一的。他们可以选择使用 R2R_Rootkey 或者 STB_RootKey 实现上述需求。

操作步骤

使用随机数密钥对内存数据加解密的步骤如下：

步骤 1 调用 HI_UNF_ADVCA_Init 初始化高安全 CA 模块，调用 HI_UNF_CIPHER_Open 打开 CIPHER 设备，调用 HI_MMZ_Malloc 为输入缓冲区和输出缓冲区申请物理连续内存，输入缓冲区存储加密前数据，输出缓冲区存储加密后数据。

步骤 2 调用 HI_UNF_CIPHER_CreateHandle 创建 CIPHER 句柄。

步骤 3 调用 HI_UNF_ADVCA_SetR2Ralg 设置 R2R 加解密算法。

步骤 4 调用 HI_UNF_ADVCA_SetR2RsessionKey 设置会话随机数密钥。海思安全芯片默认会话密钥级别为 2 级，上述函数只需要执行一次。特定 CA 芯片可能将会话密钥级别设置为 3 级，这种情况下，上述函数需要执行二次。

步骤 5 调用 HI_UNF_CIPHER_ConfigHandle 设置 CIPHER 加密参数。该参数中的 bKeyByCA 应该设置为 HI_TRUE。

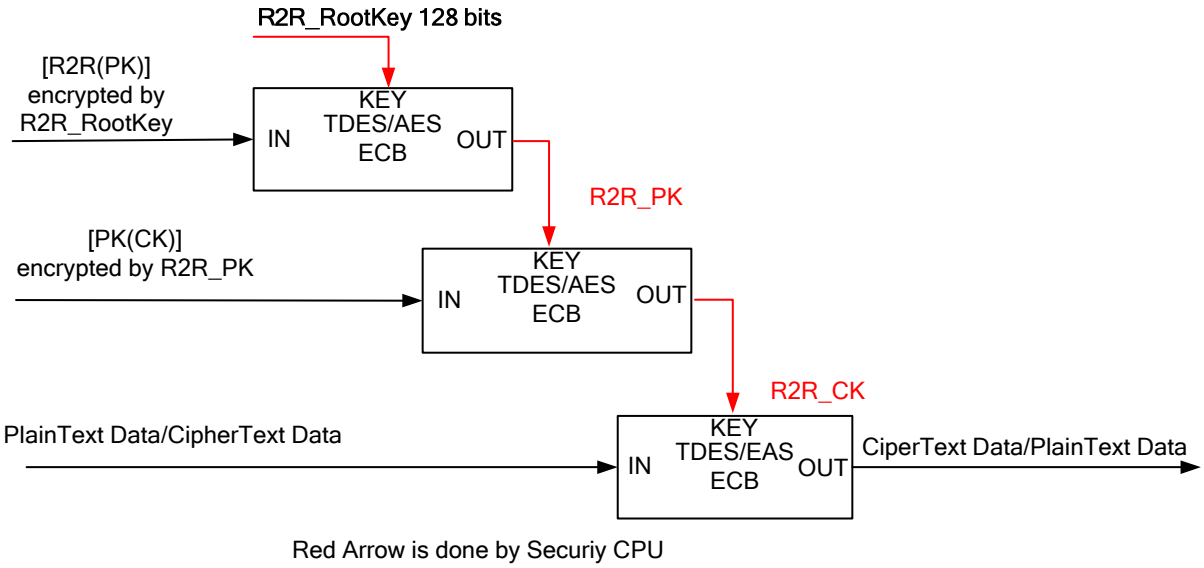
步骤 6 调用 HI_UNF_CIPHER_Encrypt 进行加密运算。

步骤 7 调用 HI_UNF_CIPHER_Close/HI_UNF_ADVCA_DeInit 关闭操作。

----结束

使用随机数密钥对内存数据加解密流程图如图 2-13 所示。

图2-13 使用随机数密钥对内存数据加解密流程图



参考代码

请参考 advca 提供的用例：sample_ca_crypto.c

2.7.3 使用固定密钥对内存数据加解密

场景说明

STB 厂商需要使用一个 Key 加密数据，并要求 Key 不能以明文方式存放在 Flash 中。这就需要被芯片用 R2R_RootKey 加密后存储在 Flash 中。

假如加密后的密钥为 R2R_RootKey(Key)。客户可以使用这个 Key 加密数据，通过 USB，网络等方式将加密数据传递到机顶盒。

机顶盒使用 R2R_RootKey(Key)通过 SWPK key ladder 解密数据，在解密数据的过程中，密钥 Key 是保密的，不能通过 CPU 读取，从而可以保证密钥 Key 的安全。

本模式适合于客户需要对数据加密，同时加密数据所用的密钥不能以明文形式存在。

说明

不同型号的海思安全芯片提供有差异的“使用固定密钥对内存数据加解密”实现，如果 STB 厂商需要使用该特性，请与海思协商具体细节。

操作步骤

使用固定密钥对内存数据加解密的操作分为 2 个部分。包括机顶盒量产前执行数据加密和机顶盒出厂后执行数据解密。

对内存中指定数据加密，其步骤如下：

步骤 1 STB 厂商前期已经定义了一个全局的 GolbalKey。

步骤 2 STB 厂商前期使用 GolbalKey 对内存数据进行 AES/TDES 加密。加密后的数据存放在机顶盒的 FLASH 内。

步骤 3 调用 HI_UNF_ADVCA_Init 初始化高安全 CA 模块。

步骤 4 调用 HI_UNF_ADVCA_EncryptSWPK 对 GolbalKey 加密。将加密的 GolbalKey 存放在机顶盒的 FLASH 内。

步骤 5 调用 HI_UNF_ADVCA_DeInit 退出。

----结束

对内存中指定数据解密，其步骤如下：

步骤 1 调用 HI_UNF_ADVCA_Init 和 HI_UNF_CIPHER_Open 初始化高安全 CA 和 CIPHER 模块。

步骤 2 调用 HI_UNF_CIPHER_CreateHandle 创建 CIPHER 句柄。

步骤 3 调用 HI_UNF_ADVCA_SWPKKeyLadderOpen 打开 SWPK 解密接口。

步骤 4 调用 HI_UNF_CIPHER_ConfigHandle 设置 CIPHER 加密参数。该参数中的 bKeyByCA 应该设置为 HI_TRUE。

步骤 5 调用 HI_UNF_CIPHER_Decrypt 进行解密运算。

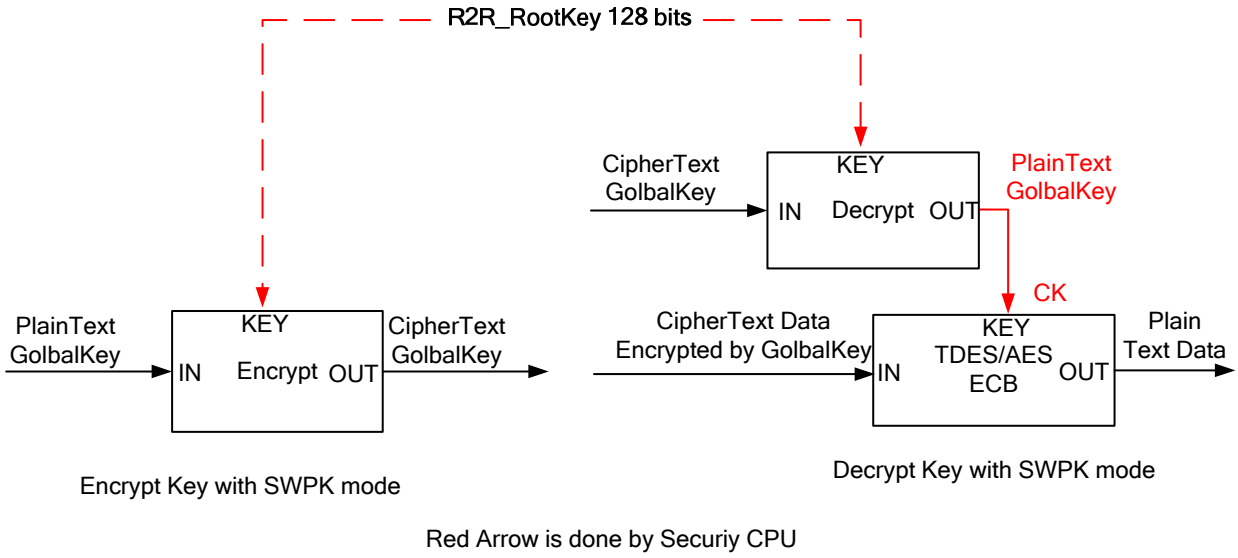
步骤 6 调用 HI_UNF_ADVCA_SWPKKeyLadderClose 关闭 SWPK 解密处理。

步骤 7 调用 HI_UNF_CIPHER_Close/ HI_UNF_ADVCA_DeInit 关闭操作。

----结束

使用固定密钥对内存数据加解密的流程图如图 2-14 所示。

图2-14 使用固定密钥对内存数据加解密流程图



参考代码

请参考 advca 提供的用例：sample_ca_swpk_keyladder.c

2.8 安全 PVR 方案

现有 PVR 方案是将经过解扰器解密后的码流录制到硬盘内。录制中可能会采用一个已知的密钥对码流进行再次加密。这种使用很容易被黑客破解，从而有可能泄露录制内容。运营商/用户需要更安全的录制功能。要求录制的内容只允许他们授权的用户才可以使用。

由此产生安全 PVR 方案。简单来说，该方案使用机顶盒内唯一的密钥对解密后的码流进行加密。这样录制的码流只有这台机顶盒可以解密出来。机顶盒内唯一的密钥应该是存放在主芯片的 OTP 区域。

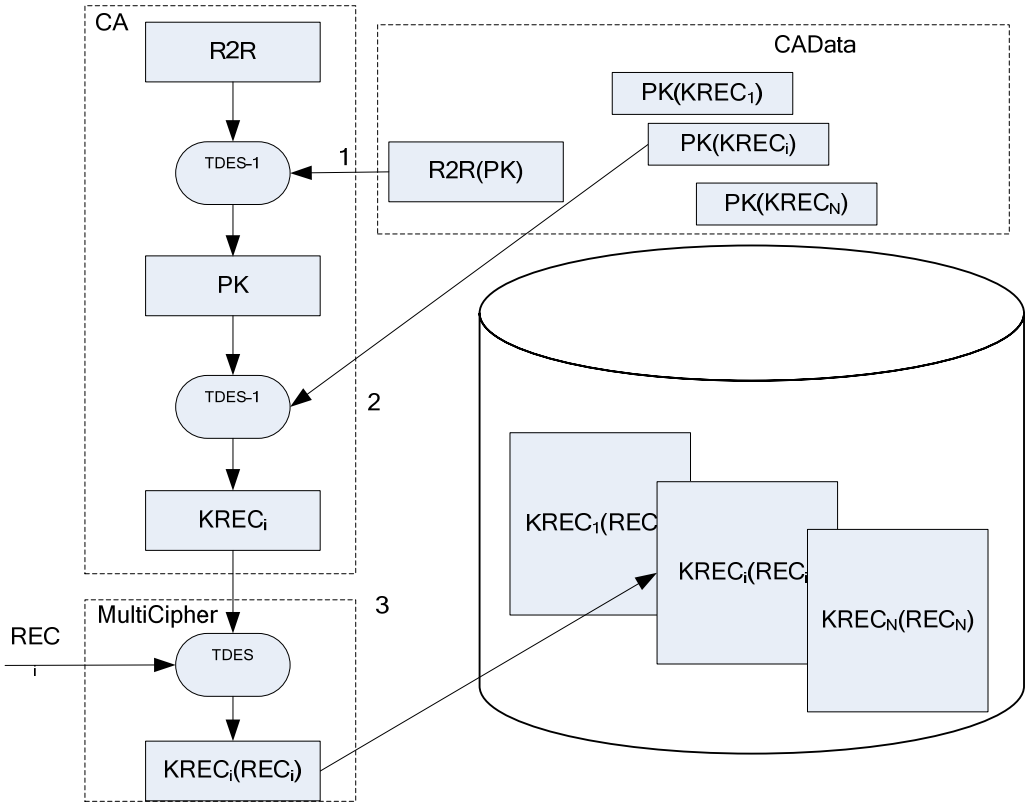
2.8.1 高安 PVR 录制

场景说明

高安全 PVR 录制需要使用芯片内置唯一的密钥作为根密钥，通过 KeyLadder 将加密码流所用的密钥(KPECi)计算出来。机顶盒再使用这个密钥(KPECi)对码流进行加密。

高安 PVR 录制原理图如图 2-15 所示。

图2-15 高安 PVR 录制过程图



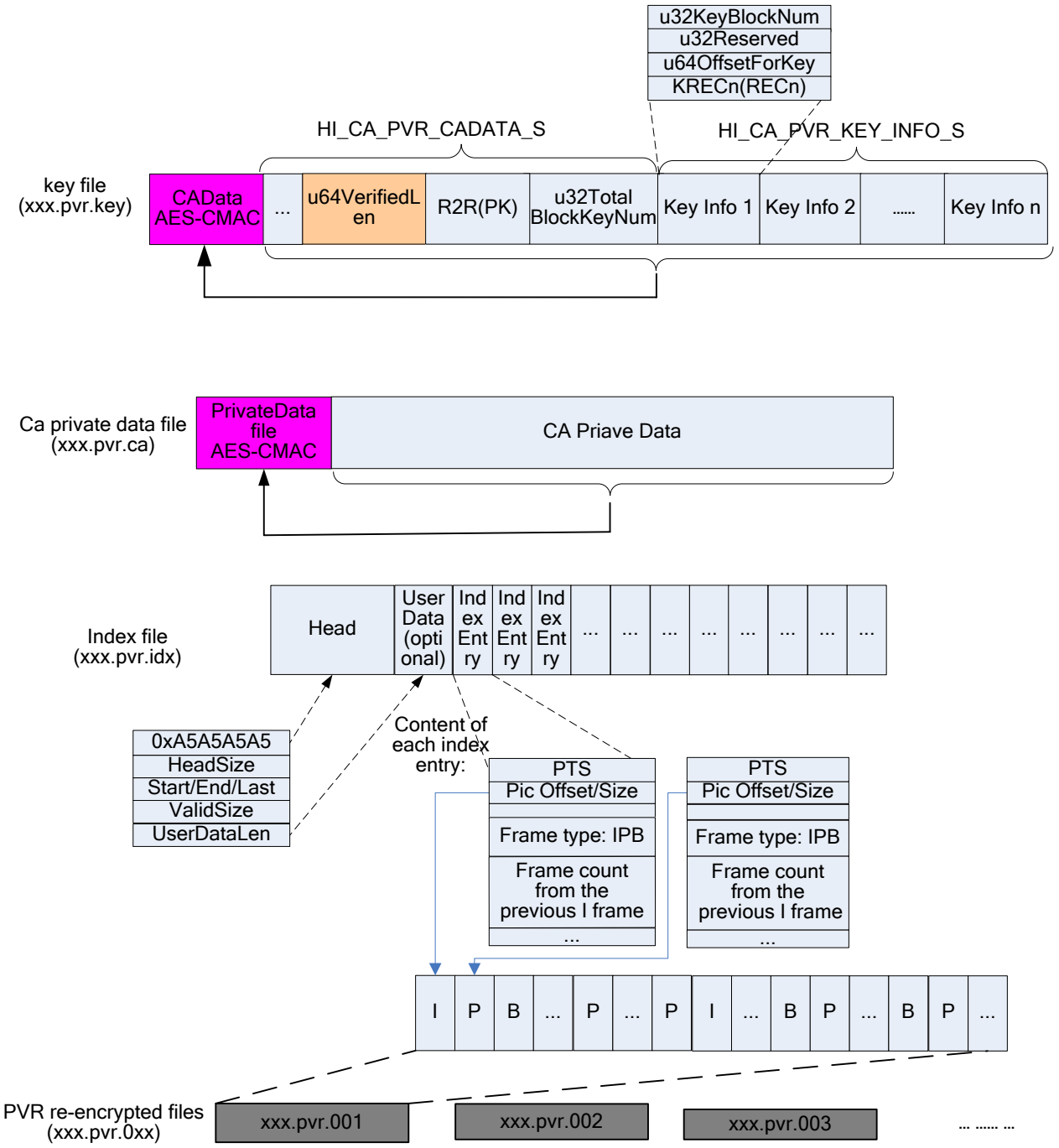
使用过程如下：

R2R(PK)是由 STB 程序启动后产生的随机数的数据。机顶盒使用 R2R_RootKey 对 R2R(PK)进行解密，获取 PK，PK 被存在了 CA 模块的内部寄存器中。

当录制节目时，STB 程序随机生成一个数据 PK(KRECi)，它经过 PK 解密后得到 KRECi 并送给 MultiCipher，KRECi 用来对节目进行加密录制。PK(KRECi)被保存在 PVR CA 私有数据中。每次当录制内容大小超过一定值的时候（默认是 1G），会重新随机生成 PK(KRECi)，用于录制新的内容。

在高安 PVR 录制过程中，由于会生成一些随机密钥（R2R(PK)、PK(KRECi)）等，这些数据需要被保存下来用于 PVR 播放，这些数据被保存在 CA 私有密钥文件(xxx.pvr.key)中。此外，由于不同 CA 厂商对 PVR 的方案需求不一致，高安 PVR 录制时还会包含一个私有的数据文件(xxx.pvr.ca)用于存放特性 CA 厂商私有数据，如下图所示。

图2-16 高安 PVR 录制文件结构图



操作步骤

操作步骤如下：

步骤 1 调用 HI_S32 HI_UNF_ADVCA_PVR_RecOpen()，打开高安 PVR 录制通道。

步骤 2 调用 HI_UNF_ADVCA_PVR_GetCAPrivateFileName()和 HI_UNF_ADVCA_PVR_CreateCAPrivateFile()创建 PVR 私有数据文件。

步骤 3 调用 HI_S32 HI_UNF_PVR_RegisterExtraCallback()，注册高安 PVR 录制回调函数到该录制通道。PVR 录制时，函数参数 enExtraCallbackType 为 HI_UNF_PVR_EXTRA_WRITE_CALLBACK，回调函数可参考 WriteCallBack 函数。

步骤 4 在 WriteCallBack 函数中，调用 HI_UNF_ADVCA_PVR_WriteCallBack()接口，该接口对录制码流执行加密处理。

步骤 5 启动高安 PVR 录制的时候，调用 HI_UNF_ADVCA_PVR_SetStreamInfo()，保存该录制文件的一些信息。**（此步骤不是必须）**

步骤 6 录制结束时候，在 HI_UNF_PVR_RecDestroyChn 之前先调用 HI_UNF_PVR_UnRegisterExtraCallBack()，去除注册在该录制通道上的回调函数。

步骤 7 录制结束，调用 HI_S32 HI_UNF_ADVCA_PVR_RecClose()，关闭该高安录制通道。

步骤 8 调用 HI_UNF_ADVCA_DeInit 关闭操作。

----结束

参考代码

请参考 advca 提供的用例：ca_pvr_rec.c、sample_ca_adp_pvr.c

2.8.2 高安 PVR 播放

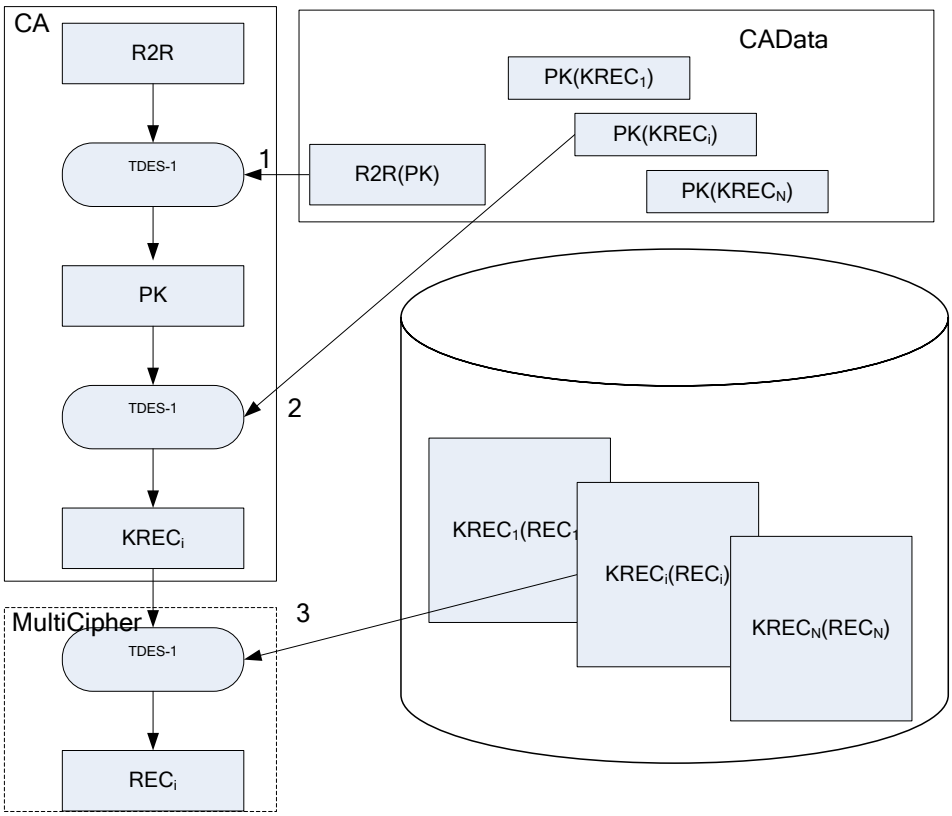
场景说明

使用前面高安 PVR 录制的码流，可以使用高安 PVR 播放出来。R2R(PK)是从录制私有文件读取出来。机顶盒使用 R2R_RootKey 对 R2R(PK)进行解密，获取 PK，PK 被存在了 CA 模块的内部寄存器中。

当播放高安 PVR 录制的节目时，机顶盒先将 PK(KRECi)送给 CA 模块，经过 PK 解密后得到加密码流的密钥(KRECi)。密钥(KRECi)再传递到 MultiCipher，就可以用来对节目进行解密。

解密过程如图 2-17 所示。

图2-17 高安 PVR 播放过程图



操作步骤

操作步骤如下：

步骤 1 调用 HI_S32 HI_UNF_ADVCA_PVR_PlayOpen(HI_U32 u32PlayChnID)，打开高安 PVR 播放通道。

步骤 2 调用 HI_UNF_ADVCA_PVR_GetCAPrivateFileName()和 HI_UNF_ADVCA_PVR_CheckCAPrivateFileMAC()验证 PVR 私有数据的合法性。

步骤 3 调用 HI_S32 HI_UNF_PVR_RegisterExtraCallback()，注册高安 PVR“播放”回调函数到该播放通道。函数参数 enExtraCallbackType 为 HI_UNF_PVR_EXTRA_READ_CALLBACK，回调函数可参考 ReadCallBack 函数。

步骤 4 在 ReadCallBack 函数中，调用 HI_UNF_ADVCA_PVR_ReadCallBack()，该接口对录制码流执行解密处理。

步骤 5 启动高安 PVR 播放的时候，调用 HI_UNF_ADVCA_PVR_GetStreamInfo)可以获取该播放文件的一些私有信息。**（此步骤不是必须）**

步骤 6 播放结束时候，在 HI_UNF_PVR_PlayDestroyChn 之前，请先调用 HI_UNF_PVR_UnRegisterExtraCallBack()，去除注册在该播放通道上的回调函数。

步骤 7 播放结束，调用 HI_S32 HI_UNF_ADVCA_PVR_PlayClose()关闭该高安播放通道。

步骤 8 调用 HI_UNF_ADVCA_DeInit 关闭操作。

----结束

参考代码

请参考 advca 提供的用例：ca_pvr_play.c、sample_ca_adp_pvr.c

2.9 JTAG 保护方案

场景说明

JTAG 是芯片提供的调试接口，用于机顶盒进行开发和测试验证。

海思 SDK 开发包中包含了 HiWorkbench 工具。机顶盒厂商可以使用该软件和 JTAG 连接器，通过 JTAG 端口连接机顶盒，从而实现调试机顶盒的功能。

从安全要求看来，正式量产的机顶盒应限制 JTAG 的使用。

高安 CA 芯片可以控制 JTAG 端口工作模式，将该接口设置为密码保护模式(Password 模式)和永久关闭模式(Close 模式)。

高安芯片在出厂前，每个芯片都设置了由 CA 公司提供的唯一的 JTAG-Password。STB 厂商在进行 STB 产品 CA 认证和 STB 产品生产中。应该按照 CA 公司的要求将 JTAG 设置为 Password 模式或者 Close 模式。

操作步骤

设置 JTAG 保护的操作步骤如下：

步骤 1 初始化 CA 模块。

步骤 2 设置 JTAG 访问控制模式，调用 HI_UNF_ADVCA_SetJtagMode()。

- 如果运营商要求将 JTAG 设置为密码保护模式，机顶盒厂商需要将参数设置为 HI_UNF_ADVCA_JTAG_MODE_PROTECT，JTAG 将被设置为密钥保护模式。
- 如果运营商要求将 JTAG 设置为永久关闭模式，机顶盒厂商需要将设置为 HI_UNF_ADVCA_JTAG_MODE_CLOSED，JTAG 将被设置为密钥保护模式。

----结束

3 高安机顶盒量产

3.1 概述

机顶盒厂商采购海思高安 CA 芯片，集成 CA 中间件，并进行机顶盒整机认证。通过 CA 厂商的认证后，机顶盒就进入了量产阶段。

本章主要描述高安全机顶盒整机生产的一般流程。高安全机顶盒的生产跟普通机顶盒的生产差别主要体现在 PV 的设置和量产报告。本章节将会按照不同 CA 厂商的要求，分别列举对应的 PV 设置指导步骤。

推荐的高安机顶盒生产基本流程如下：

步骤 1 从海思购买高安芯片，明确使用哪个 CA 厂商，哪款芯片型号等信息。

步骤 2 机顶盒的生产线执行生产测试，验证硬件和软件方面的功能。

步骤 3 设置高安芯片的 PV(Permanent Value)。

步骤 4 向 CA 厂商申请用于高安全机顶盒测试的码流以及智能卡等，测试机顶盒能否正常解扰。

步骤 5 读出机顶盒内芯片的 ChipID，按照 CA 厂商的格式要求生成相应的 ChipID 报告，并将该报告发给运营商。

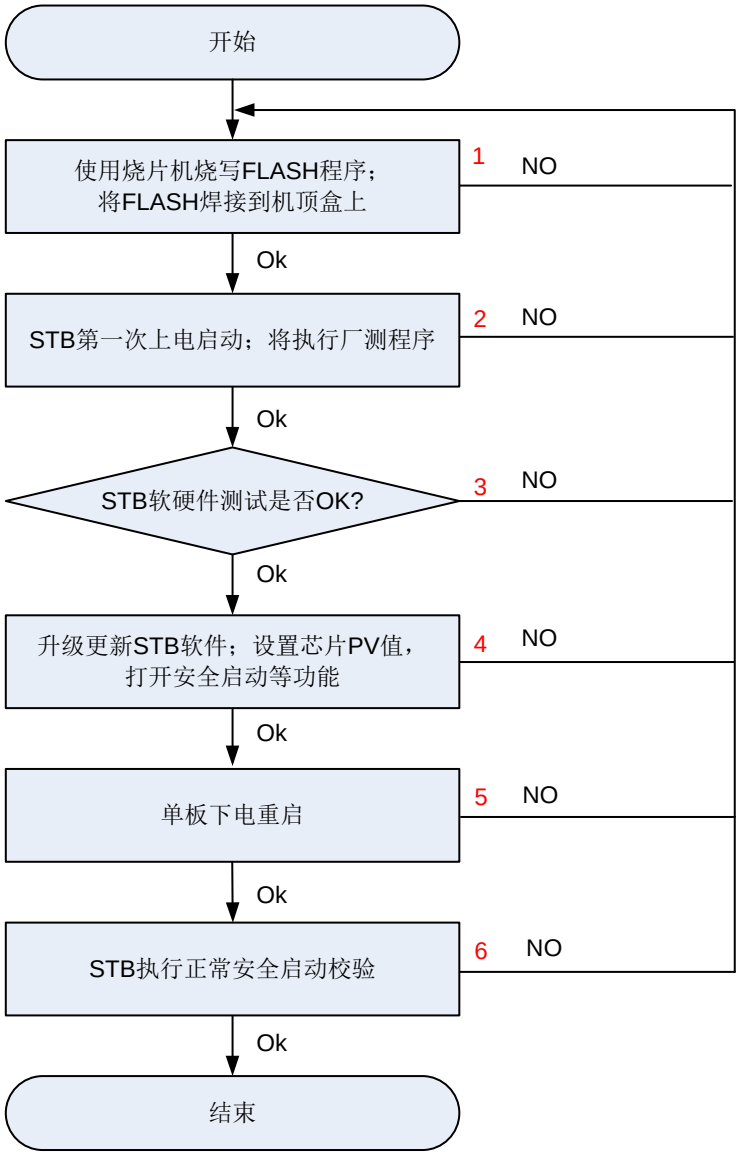
----结束

3.2 高安 CA 机顶盒工厂生产基本步骤

高安 CA 机顶盒在生产中，与普通机顶盒比较，增加设置芯片安全控制位 PV(Permanent Value)的操作。如下是基本的生产步骤。

图 3-1 是高安 CA 机顶盒工厂生产基本流程图。

图3-1 高安 CA 机顶盒工厂生产基本流程图



操作步骤

具体操作步骤如下：

步骤 1 使用 flash 烧片机将 Flash 主要的镜像文件(BOOT、BootArgs、Loader、Kernel、FileSystem 等)统一烧写到 Flash 上。将该 Flash 焊接到 STB 单板上。

步骤 2 第一次启动单板，安全启动标志位(请参考“getSecureBootEn”命令)没有设置，BOOT 程序可以判断该标志位。此时，该标志应该为 0，BOOT 将跳转到 STB 生产测试程序。

说明

- 步骤 2 执行的 BOOT 一般不是正式量产 BOOT。该镜像可以包含多种生产调试手段，以方便量产。正式 BOOT 应该通过步骤 4 的升级完成。
- STB 生产测试程序是一个独立的由 STB 厂商定制化的小应用程序，可以作为 Loader 的一部分，也可以是附属于主程序，并存放在文件系统内的子程序。
- STB 生产测试程序应该包括 STB 软硬件检测功能，网络下载数据等功能。
- 本文不涉及 STB 生产测试程序实现。

步骤 3 检测 STB 软硬件是否正常。

步骤 4 下载更新代码镜像（包括正式 BOOT，非 BOOT 镜像等），设置与机顶盒相关的数据和序列号。设置高安 CA PV 值，设置安全启动。

步骤 5 在确认上面步骤完成后，再次重启 STB。

步骤 6 STB 将按照高安 CA 要求执行安全启动校验。工厂验证 STB 运行正常后，生产流程结束。

----结束

注意事项

用于保护 HDMI 端口的 HDCP key 属于所有高清机顶盒芯片共有的密钥。请客户参考 HDCP 相关使用文档。如果客户需要在高安芯片使用 HDCP 保护 HDMI。其处理的流程是先执行高安芯片 PV 烧写后，再向芯片内写入 HDCP 相关的密钥，按照芯片型号分别为：

- Hi3716CV100/Hi3716MV200/Hi3716MV300，客户需要设置 HDCP KEY。
- Hi3716CV200/Hi3798MV100/Hi3716MV310/Hi3716MV420/Hi3716MV410 以及后续芯片，客户需要设置 HDCP_ROOT_KEY。

PV 值烧写不可逆，一旦写入数据出现失败，该芯片无法再次使用，需要更换芯片。STB 厂商可以在校验上一次操作成功后，再执行下一步操作。

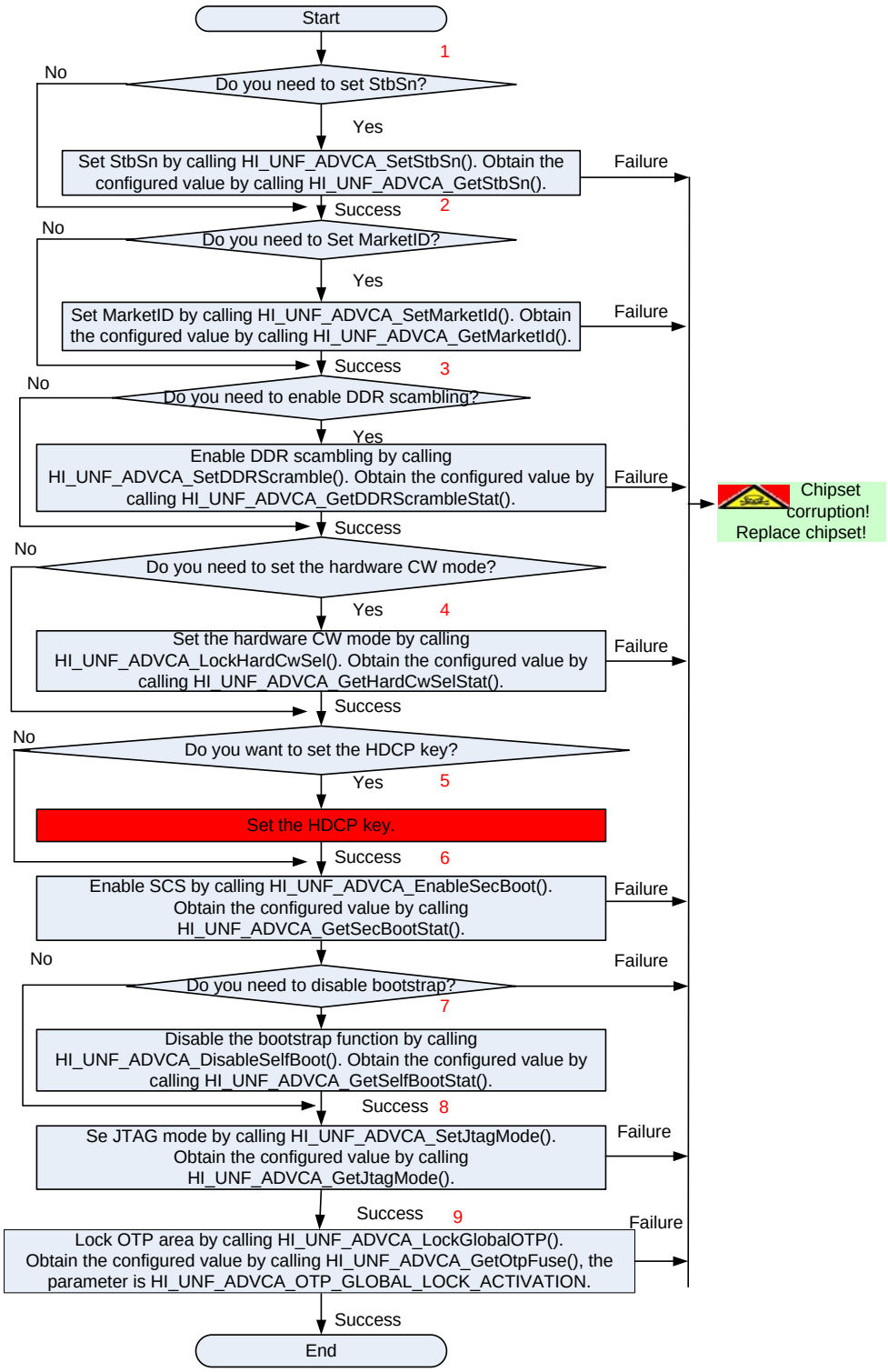
3.2.1 高安 CA 芯片基本的设置

不同的 CA 厂商对机顶盒的安全要求各有不同。除了下文单独列出的 CA 厂商外，CA 厂商要求 STB 厂商使用高安 CA 芯片时，需要打开芯片内置的各种安全控制功能。

本文称这个操作为设置芯片 PV(Permanent Value)值。

STB 厂商可以参考下面的 PV 设置流程。除了安全启动和 JTAG 保护模式一定要设置外，其他选项，如设置 STBSN、MarketID、Enable DDRScramble、DisableSelfboot、LockHardCw 等标志位由 STB 厂商按照 CA 公司要求设置。

图3-2 高安 CA 芯片基本的 PV 设置流程图



PV 设置的是芯片内部 OTP 的数据，OTP 大部分数据必须单板重启后才能获得设置状态，在生产中，只需要判断设置 PV 的函数返回值是否为 HI_SUCCESS，即可以判断设置是否成功。

操作步骤

具体操作步骤如下：

步骤 1 依据 CA 厂商需求，STB 厂商需要确定是否设置 STBSN。如果 CA 厂商需要设置 STBSN，设置 STBSN。设置接口为：HI_UNF_ADVCA_SetStbSn()；获取设置值的接口为：HI_UNF_ADVCA_GetStbSn()。如果 CA 厂商没有此类要求，请直接跳过该步骤。

步骤 2 依据 CA 厂商需求，STB 厂商需要确定是否设置 MarketID。如果 CA 厂商需要设置 MarketID，设置 MarketID。设置接口为：HI_UNF_ADVCA_SetMarketId()；获取设置值接口为：HI_UNF_ADVCA_GetMarketId()。请参考《Common CASignTool 使用说明》中配置市场区域码的字节序设置。如果 CA 厂商没有此类要求，请直接跳过该步骤。

步骤 3 依据 CA 厂商需求，STB 厂商需要确定是否设置 DDR 加扰标志位。如果 CA 厂商需要设置 DDR 加扰。设置接口为：HI_UNF_ADVCA_SetDDRScramble()；获取设置值的接口为：HI_UNF_ADVCA_GetDDRScrambleStat()。如果 CA 厂商没有此类要求，请直接跳过该步骤。

步骤 4 依据 CA 厂商需求，STB 厂商需要确定是否设置硬件 CW 字模式。如果 CA 厂商需要设置硬 CW 字模式，设置硬 CW 字模式。设置接口为：HI_UNF_ADVCA_LockHardCwSel()；获取设置值的接口为：HI_UNF_ADVCA_GetHardCwSelStat ()。如果 CA 厂商没有此类要求，请直接跳过该步骤。

步骤 5 如果客户需要使用 HDCP key，请在这个阶段烧写。Hi3716CV100，Hi3716MV300, Hi3716CV200 的使用方法有差异。详情请参考相关的 HDCP Key 使用指南。

步骤 6 设置安全启动 SCS 模式。设置接口为：HI_UNF_ADVCA_EnableSecBoot()；获取设置值的接口为：HI_UNF_ADVCA_GetSecBootStat()。机顶盒需要断电重启，才可以验证该功能。

步骤 7 依据 CA 厂商需求，STB 厂商需要确定是否设置关闭自举功能。如果 CA 厂商需要禁止使用自举(selfboot)功能，可以关闭自举功能。设置接口为：HI_UNF_ADVCA_DisableSelfBoot()；获取设置值的接口为：HI_UNF_ADVCA_GetSelfBootStat()。如果 CA 厂商没有此类要求，请直接跳过该步骤。

步骤 8 设置 JTAG 为 password 模式。设置接口为：HI_UNF_ADVCA_SetJtagMode ()；获取设置值的接口为：HI_UNF_ADVCA_GetJtagMode()。

步骤 9 设置 OTP 全局锁定控制位(可选项)。设置接口为：HI_UNF_ADVCA_LockGlobalOTP ()；获取设置值的接口为：HI_UNF_ADVCA_GetOtpFuse()，其参数为 HI_UNF_ADVCA_OTP_GLOBAL_LOCK_ACTIVATION。

警告

OTP 全局锁定控制位一旦设置，芯片 OTP 区间将无法改变，这个操作必须放在所有 OTP 操作（比如烧写 PV 值，烧写 HDCP_Root_Key，STB_Root_Key 等）的最后。

----结束

3.3 各 CA 芯片使用说明

3.3.1 Novel 高安芯片

永新视博（Novel）CA 专用芯片上的高安位域的标识符为“Y”。

3.3.1.1 Novel 高安芯片密钥组成

按照 Novel 要求，STB 厂商可能使用如表 3-1 所示的密钥。

表3-1 Novel 高安芯片密钥表

Key	Key-owner	Location	Description
CHIPID	Novel	OTP	Chipset unique ID. This value is defined by Novel. This value has already burned into chipset.
CSA2_RootKey	Novel	OTP	Novel secret key. This value has already burned into chipset.
R2R_RootKey	Novel	OTP	Novel secret key. This value has already burned into chipset.
MISC_RootKey	Novel	OTP	Novel secret key. This value has already burned into chipset.
JTAG_Key	Novel	OTP	Novel secret key. This value has already burned into chipset.
ROOT_RSA_PUB_KEY	Novel	OTP	Security SCS root key. This value has already burned into chipset.
EXT_RSA_PUB_KEY	STB Vendor	Flash	Secure SCS external key. Use by BootRom to verify “param” and check-area(BOOT) of BOOT in Flash.
EXT_RSA_PUB_KEY2	STB Vendor	Flash	Secure SCS second external key. Its public key is stored in BOOT code. Use to verify the signature of software images(bootargs, system, loader, stbid). Its private key is managed by STB vendor. It can be the same value of EXT_RSA_PUB_KEY.
MarketID	Novel	OTP/Flash	This value is defined by Verimatrix.
HDCP_Key	STB Vendor	OTP	It is used to protect HDMI.

3.3.1.2 Novel 码流解密

Novel 在码流解密中，要求 STB 厂商使用 DVB_RootKey(即 CSA2.0_RootKey)作为 DVB 业务根密钥，STB 厂商需要使用 3 级 Key-Ladder 对 TS 进行解密。

Hi3716CV200 以及后续芯片，Novel 还要求 STB 厂商使用 MISC_RootKey 作为 DVB 业务备选根密钥，STB 厂商需要使用 3 级 Key-Ladder 对 TS 进行解密。

海思为 Novel 提供专用的 R2R 加解密接口。

请参考 advca 的用例：sample_ca_novel_dvbplay.c、sample_ca_novel_dvbplay_misc.c、ca_novel_crypto.c。

3.3.1.3 Novel 镜像签名

签名镜像包括如下 2 类：

- 对于 BOOT 镜像。

  STB 厂商需要从 Novel 获取到市场区域码(MarketID)。

  STB 厂商需要将整个 BOOT 源代码提交 NOVEL 审核，由 NOVEL 编译出量产 BOOT，并完成 BOOT 签名。

  STB 厂商最后可以获得签名的 BOOT。

- 对于非 BOOT 镜像，比如 bootargs、kernel、FS 等。

  Novel 发布非 BOOT 镜像签名规范，STB 厂商需要与 Novel 协商最终的实施方案。

  2.4 中定义的非 BOOT 软件安全校验可以作为 STB 厂商的一个参考方案。

3.3.2 Suma 高安芯片

数码视讯（SUMA）CA 专用芯片上的高安位域的标识符为“S”。

3.3.2.1 Suma 高安芯片密钥组成

按照 Suma 要求，STB 厂商可能使用如表 3-2 所示的密钥。

表3-2 Suma 高安芯片密钥表

Key	Key-owner	Location	Description
CHIPID	Suma	OTP	Chipset unique ID. This value is defined by Suma. This value has already burned into chipset.
CSA2_RootKey	Suma	OTP	Suma secret key. This value has already burned into chipset.
R2R_RootKey	Suma	OTP	Suma secret key. This value has already burned into chipset.
JTAG_Key	Suma	OTP	Suma secret key. This value has already burned into chipset.

Key	Key-owner	Location	Description
ROOT_RSA_PUB_KEY	Suma	OTP	Security SCS root key. This value has already burned into chipset.
EXT_RSA_PUB_KEY	STB Vendor	Flash	Secure SCS external key. Use by BootRom to verify “param” and check-area(BOOT) of BOOT in Flash.
EXT_RSA_PUB_KEY2	STB Vendor	Flash	Secure SCS second external key. Its public key is stored in BOOT code. Use to verify the signature of software images(bootargs, system, loader, stbid). Its private key is managed by STB vendor. It can be the same value of EXT_RSA_PUB_KEY.
HDCP_Key	STB Vendor	OTP	It is used to protect HDMI.

3.3.2.2 Suma 码流解密

Suma 在码流解密中，要求 STB 厂商使用 DVB_RootKey(即 CSA2.0_RootKey)作为 DVB 业务根密钥。STB 厂商需要根据 Suma 要求，选择使用 2 级或者 3 级 Key-Ladder 对 TS 进行解密。

请参 advca 提供的用例：sample_ca_suma_tsplay.c、sample_ca_crypto.c。

3.3.2.3 Suma 镜像签名

签名镜像包括如下 2 类：

- 对于 BOOT 镜像。STB 厂商可以使用 SDK 编译产生安全 BOOT 相关的镜像，包括：cfg.bin 和 fastboot-burn.bin。

  STB 厂商将上述镜像提交 Suma 签名，可以获得签名的 BOOT。
- 对于非 BOOT 镜像，比如 bootargs、kernel、FS 等。

  “2.4 非 BOOT 软件安全校验”中定义的非 BOOT 软件安全校验可以作为 STB 厂商的一个参考方案。

3.3.3 CTI 高安芯片

算通（CTI）CA 专用芯片上的高安位域的标识符为“T”。

3.3.3.1 CTI 高安芯片密钥组成

按照 CTI 要求，STB 厂商可能使用如表 3-3 所示的密钥。

表3-3 CTI 高安芯片密钥表

Key	Key-owner	Location	Description
CHIPID	CTI	OTP	Chipset unique ID. This value is defined by CTI. This value has already burned into chipset.
CSA2_RootKey	CTI	OTP	CTI secret key. This value has already burned into chipset.
R2R_RootKey	CTI	OTP	CTI secret key. This value has already burned into chipset.
JTAG_Key	CTI	OTP	CTI secret key. This value has already burned into chipset.
ROOT_RSA_PUB_KEY	CTI	OTP	Security SCS root key. This value has already burned into chipset.
STB_ROOT_KEY	CTI	OTP	CTI static secret key. This value has already burned into chipset. It is used to encrypt the Boot-Image.
EXT_RSA_PUB_KEY	STB Vendor	Flash	Secure SCS external key. Use by BootRom to verify “param” and check-area(BOOT) of BOOT in Flash.
EXT_RSA_PUB_KEY2	STB Vendor	Flash	Secure SCS second external key. Its public key is stored in BOOT code. Use to verify the signature of software images(bootargs, system, loader, stbid). Its private key is managed by STB vendor. It can be the same value of EXT_RSA_PUB_KEY.
HDCP_Key	STB Vendor	OTP	It is used to protect HDMI.

3.3.3.2 CTI 码流解密

CTI 在码流解密中，要求 STB 厂商使用 DVB_RootKey(即 CSA2.0_RootKey)作为 DVB 业务根密钥，STB 厂商需要使用 2 级 Key-Ladder 对 TS 进行解密。

请参考 advca 提供的用例：sample_ca_cti_tsplay.c、sample_ca_crypto.c。

3.3.3.3 CTI 镜像签名

签名镜像包括如下 2 类：

- 对于 BOOT 镜像。STB 厂商可以使用 SDK 编译产生安全 BOOT 相关的镜像，包括：cfg.bin 和 fastboot-burn.bin。

  STB 厂商将上述镜像提交 CTI 签名，可以获得签名的 BOOT。

- 对于非 BOOT 镜像，比如 bootargs, kernel, FS 等。

  “2.4 非 BOOT 软件安全校验”中定义的非 BOOT 软件安全校验可以作为 STB 厂商的一个参考方案。

3.3.4 Verimatrix 高安芯片

Verimatrix CA 专用芯片上的高安位域的标识符为”M”。

3.3.4.1 Verimatrix 安全要求分类

Verimatrix 支持 2 种不同认证等级，不同等级对机顶盒产品的要求如下：

- Standard-STB Security

  当前市场通常使用 Standard STB 级别。该安全级别的要求如下：

  - 机顶盒必须提供基于硬件实现的安全启动方案。算法为 RSA2048-HASH256。
  - 机顶盒必须支持安全升级方案，升级数据必须先验证其合法性再更新程序。
  - 机顶盒必须支持唯一 CHIPID。CHIPID 长度至少 32bit。
  - 机顶盒必须支持内存加密和 DDR 加扰。
  - 机顶盒必须 JTAG 保护，支持 Secure JTAG。

  STB 厂商可以使用 Verimatrix Standard 安全级别芯片实现上述特性。这个类型的芯片需要 STB 厂商自行设置密钥，包括 CHIPID, Root_RSA_Public_Key 等密钥，并在最终量产时，按照后面章节的建议设置芯片对应的 PV 值。

- Advcanced-STB Security

  Hi3716CV200 以及后续芯片可以支持级别。该安全级别的要求如下：

  - 机顶盒必须满足 Standard STB Security 所要求的特性。
  - 机顶盒必须支持密文 CW 字的 DVB Key-Ladder 解密方案。

  STB 厂商必须使用 Verimatrix Advcanced 安全级别芯片实现上述特性。这个类型的芯片出厂前，芯片内部已经写入 Verimatrix 的密钥，包括 CHIPID, Root_RSA_Public_Key 等密钥。所以，STB 厂商只需要按照后面章节的建议设置芯片对应的剩余的 PV 值。

3.3.4.2 Verimatrix Standard-STB 安全级别芯片

Verimatrix Standard-STB 密钥组成

按照 Verimatrix 要求，STB 厂商可能使用如表 3-4 所示的密钥。

表3-4 Verimatrix Standard-STB 安全级别密钥表

Key	Key-owner	Location	Description
CHIPID	STB vendor	OTP	Chipset unique ID, burnt by the STB vendor.
R2R_RootKey	STB Vendor	OTP	R2Rroot key.This key has the size of 16 bytes. The first 8 bytes and the last 8 bytes of the key can not be equal.
JTAG_Key	STB Vendor	OTP	JTAG password, burnt by the STB Vendor.

Key	Key-owner	Location	Description
ROOT_RSA_PUB_KEY	Verimatrix	OTP	Security SCS root key, STB Vendor require this key from Verimatrix, burnt by the STB Vendor. Only use by BootRom to verify Key-Area of Boot in Flash.
EXT_RSA_PUB_KEY	STB Vendor	Flash	Secure SCS external key. Use by BootRom to verify “param” and check-area(Boot) of BOOT in Flash.
EXT_RSA_PUB_KEY2	STB Vendor	Flash	Secure SCS second external key. Its public key is stored in BOOT code. Use to verify the signature of software images(bootargs, system, loader, stbid). Its private key is managed by STB vendor. It can be the same value of EXT_RSA_PUB_KEY.
HDCP_Key	STB Vendor	OTP	It is used to protect HDMI.

Standard-STB BOOT 签名流程

海思 Verimatrix Standard-STB 高安芯片在出厂时，默认没有写入用于安全启动所需的 ROOT_RSA_PUB_KEY，所以在打开安全启动之前，芯片需要写入 ROOT_RSA_PUB_KEY。

海思用于安全启动的 FinalBoot 包含 2 级签名方案。Verimatrix 到目前为止只愿意维护一组 RSA Key，可以是 ROOT_RSA_PUB_KEY 或者是 EXT_RSA_PUB_KEY，并提供镜像签名。

下面的操作流程是假设 Verimatrix 维护 ROOT_RSA_PUB_KEY 为例进行说明。在这种情况下，客户直接从 Verimatrix 申请 ROOT_RSA_PUB_KEY，而 Flash 上的 External RSA Public Key 可以由 STB 厂商产生和维护。

STB 厂商签名和使用 BOOT 的工作流程如下：

步骤 1 参考《CASignTool 使用说明》，STB 厂商可以使用 CASignTool 工具，产生一对 Extrenal RSA Key。

步骤 2 使用 CASignTool 对 BOOT 签名，请参考《CASignTool 使用说明》中的“2.3 生成自签名的安全 BOOT 功能”。将产生的 KeyArea.bin 文件提交给 Verimatrix 签名。

步骤 3 Verimatrix 返回 KeyArea.bin 文件对应的签名文件（假设该镜像为 KeyArea.bin.sign），同时包含一个 ROOT_RSA_PUB_KEY。

步骤 4 使用 CASignTool 的“Merge Signed-BootImage”功能，将多个 BOOT 相关镜像合并为最终的 FinalBoot.bin。

步骤 5 使用下载工具将 FinalBoot.bin 写入 FLASH。Verimatrix 要求 BOOT 应该是加密存放在 FLASH。STB 厂商需要对 BOOT 进行加密。请参考高安用例中的 sample_product_encrypt_boot.c 对 BOOT 进行加密。

步骤 6 向芯片 OTP 内写入 ROOT_RSA_PUB_KEY。使用 HI_UNF_ADVCA_SetRSAKey()设置。请参考 sample_ca_writeRSAkey.c。

步骤 7 打开安全启动模式。请参考高安用例中的 sample_ca_opensecboot.c 打开安全启动。

----结束

VerimatrixCA Standard-STB 安全级别芯片解密需求

Verimatrix DRM，也可以被称为 Verimatrix IPTV。

Verimatrix 加密码流中的 PMT 包含 CA-System-Descriptor 中，有如下的规定：

- 如果 CA-System-Descriptor 中不包含私有数据段，码流将使用 CSA2.0 算法加扰。
- 如果 CA-System-Descriptor 中包含私有数据段，第一个字节表示加密算法。STB 厂商可以向 CA 公司获得更详细的信息。

Verimatrix DRM 除了 CSA2.0 外，还会使用的是 AES-ECB 和 AES-CBC 这 2 种加密算法加密数据流。所以在码流在传递给芯片 Demux 之前，需要使用 CIPHER 模块解密每个 TS 块。

对内存中指定数据使用 AES 解密的步骤如下，请参考高安用例中 sample_ca_cipher.c：

步骤 1 调用 HI_UNF_CIPHER_Open 初始化 CIPHER 模块。

步骤 2 调用 HI_UNF_CIPHER_CreateHandle 获取 CIPHER 句柄。

步骤 3 用 HI_UNF_CIPHER_ConfigHandle 配置加/解密选项。

该配置参数中：

- 如果 CIPHER 采用 AES-ECB，请配置 CipherCtrl.enAlg = HI_UNF_CIPHER_ALG_AES 和 CipherCtrl.enWorkMode = HI_UNF_CIPHER_WORK_MODE_ECB。
- 如果 CIPHER 采用 AES-CBC，请配置 CipherCtrl.enAlg = HI_UNF_CIPHER_ALG_AES 和 CipherCtrl.enWorkMode = HI_UNF_CIPHER_WORK_MODE_CBC。
- 如果 IV 向量需要被更新，请设置 stChangeFlags.bit1IV = 1。否则，如果 stChangeFlags.bit1IV = 0，IV 向量使用上一次调用 CIPHER 的计算值。

步骤 4 调用 HI_MMZ_New()等函数分配 DDR 数据的物理内存，并将需要解密的数据拷贝到该物理内存中。

步骤 5 调用 HI_UNF_CIPHER_ Decrypt 解密物理内存中的数据。返回值即为解密后的数据。

步骤 6 用 HI_UNF_CIPHER_DestroyHandle 释放 CIPHER 句柄。

步骤 7 调用 HI_UNF_CIPHER_Close 关闭 CIPHER 设备。

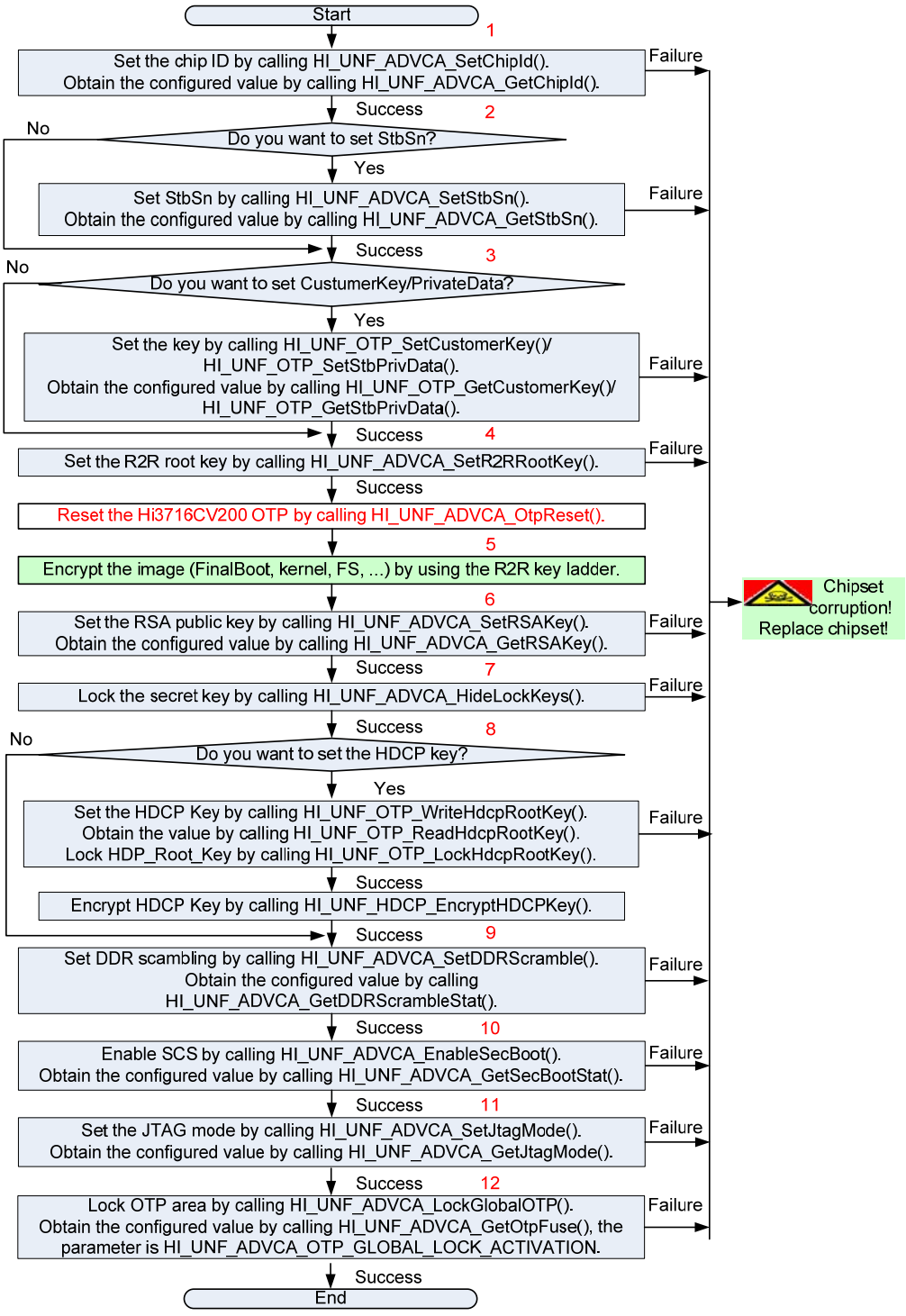
----结束

Verimatrix Standard-STB 安全级别的芯片 PV 设置

支持 Verimatrix Standard-STB 安全级别的芯片包括：Hi3716MV300，Hi3716CV200、Hi3716MV400、 Hi3716MV310、Hi3719CV100、Hi3719MV100、Hi3796CV100、Hi3798CV100、Hi3798MV100、Hi3796MV100 、Hi3716MV420、Hi3716MV410、Hi3798CV200 等。

按照 Verimatrix 要求，STB 厂商使用 Verimatrix 高安芯片，进行 STB 整机认证和生产时，机顶盒需要设置的 PV 值如图 3-3 所示。

图3-3 Hi3716MV300/Hi3716CV200 Verimatrix Standard-STB 安全级别芯片 PV 设置流程图



STB 厂商应按照上面的流程设置 PV 值，上面的设置可以分多次进行，请按照上面设置顺序依次执行。

PV 设置的是 OTP 的数据，OTP 有部分数据必须单板重启后才能获得设置状态，在生产中，只需要判断设置 PV 的函数返回值是否为 HI_SUCCESS，即可以判断设置是否成功。

具体操作如下：

步骤 1 设置 CHIPID。设置接口为：HI_UNF_ADVCA_SetChipId()；获取设置值接口为：HI_UNF_ADVCA_GetChipId()。

NOTE

- CHIPID 是定义芯片标示唯一的 ID。
- 这个 ID 一共有 4 个字节。VerimatrixS tandard-STB 安全级别芯片内出厂前默认没有烧写此 ID。
- 调用 HI_UNF_ADVCA_SetChipId ()时，如果设置的 CHIPID 是 0x12345678，那么通过 HI_UNF_ADVCA_SetChipId ()读出的 CHIPID 为 0x78563412。

步骤 2 如果 STB 厂商需要使用 STBSN 保留私有数据，可以设置 StbSn。否则，直接跳过该步骤。

StbSn 是 CHIPID 的扩展，如果客户要设置大于 32bit 的芯片标志 ID，可以先使用 CHIPID 设置低 32bit 的数据，其余数据可以写入 StbSn。设置接口为：HI_UNF_ADVCA_SetStbSn()；获取设置值接口为：HI_UNF_ADVCA_GetStbSn()。

步骤 3 如果 STB 厂商需要使用 CustomerKey 或者 StbPrivateData 存储空间保留私有数据，可以设置 CustomerKey/StbPrivateData。否则，直接跳过该步骤。

此类接口使用与存储客户私有数据，CustomerKey 和 StbPrivateData 的大小分别为 16 个字节。设置接口为：HI_UNF_OTP_SetCustomerKey()/HI_UNF_OTP_SetStbPrivData(), 获取设置值的接口为：HI_UNF_OTP_GetCustomerKey()/HI_UNF_OTP_GetStbPrivData()。

步骤 4 设置 R2R_Root_Key。

VerimatrixCA 建议机顶盒镜像需要加密存放在 FLASH 中。。STB 厂商可以选择使用 R2R_Root_Key 作为根密钥用 R2R Key-Ladder 对非 BOOT 软件加密后再存到 Flash。

NOTE

设置了 R2R_Root_Key 后，Hi3716CV200 以及后续芯片，无法读取写入的密钥。可以选择上面的断电重新启动或者调用 HI_UNF_ADVCA_OtpReset()使得密钥生效。

R2R_Root_Key 由 16 个字节组成，前后 8 个字节的数据不能相等。

步骤 5 R2R_Root_Key 生效后。STB 厂商可以使用 R2R_Root_Key 加密 BOOT, kernel，FS 等。

加密 BOOT 时，Hi3716CV200 采用的是先签名后加密 BOOT 的方案。请先使用 SignTool 制作已被签名的 BOOT(FinalBoot.bin)，将 FinalBoot.bin 下载到 Flash 内，再使用 CIPHER 对 Finalboot 区的代码使用 AES 算法加密。最后使用 HI_UNF_ADVCA_EncryptSWPK()对 FinalBoot.bin 的加密密钥再次加密。请参考 ca_blpk.c。

加密 kernel 和文件系统等，可以参考 BOOT 下的命令：common_verify_encryptimage 等验证。

步骤 6 需要设置 ROOT_RSA_PUB_KEY。设置接口为：HI_UNF_ADVCA_SetRSAKey()；获取设置值的接口为：HI_UNF_ADVCA_GetRSAKey()。

步骤 7 锁定 R2R_RootKey，ROOT_RSA_PUB_KEY 等密钥，需要调用 HI_UNF_ADVCA_HideLockKeys()。

步骤 8 如果客户需要使用 HDCP key，请在这个阶段设置。

步骤 9 设置 DDR 加扰，设置接口为：HI_UNF_ADVCA_SetDDRScramble()；获取设置值的接口为：HI_UNF_ADVCA_GetDDRScrambleStat()。

步骤 10 设置安全启动 SCS 模式。设置接口为：HI_UNF_ADVCA_EnableSecBoot()；获取设置值的接口为：HI_UNF_ADVCA_GetSecBootStat()。机顶盒需要断电重启，才可以验证该功能。

步骤 11 设置 JTAG 为 password 模式。设置接口为：HI_UNF_ADVCA_SetJtagMode ()；获取设置值的接口为：HI_UNF_ADVCA_GetJtagMode()。

步骤 12 设置 OTP 全局锁定控制位(可选项)。设置接口为：HI_UNF_ADVCA_LockGlobalOTP ()；获取设置值的接口为：HI_UNF_ADVCA_GetOtpFuse()，其参数为 HI_UNF_ADVCA_OTP_GLOBAL_LOCK_ACTIVATION。

警告

OTP 全局锁定控制位一旦设置，芯片 OTP 区间将无法改变，这个操作必须放在所有 OTP 操作（比如烧写 PV 值，烧写 HDCP_Root_Key，STB_Root_Key 等）的最后。

----结束

3.3.4.3 Verimatrix Advcanced-STB 安全级别芯片

有关 Verimatrix Advcanced 高安的整机开发请参考 Verimatrix 高安对应的开发文档。包括《Verimatrix advanced CA 开发指南补充说明》等。

3.3.5 Nagra 高安芯片

Nagra CA 专用芯片上的高安位域的标识符为“R”。

有关 Nagra 高安的整机开发请参考 Nagra 高安对应的开发文档。包括《Nagra CASignTool 使用说明》和《Nagra 高级安全 CA Boot 开发指南》等。

3.3.6 Conax 高安芯片

Conax CA 专用芯片上的高安位域的标识符为“C”。

有关 Conax 高安的整机开发请参考 Conax 高安对应的开发文档。包括《Conax 高安全 CA 开发指南补充说明.pdf》等。

3.3.7 Irdeto 高安芯片

Irdeto CA 专用芯片上的高安位域的标识符为“I”。

海思与 Irdeto 共同完成《Supplement to Advanced Secure CA Development Guide for Irdeto.pdf》。该文档主要描述了 STB 厂商如何向 Irdeto 申请签名高安 Boot。STB 厂商可以联系 Irdeto 获取。

3.3.8 Panaccess 高安芯片

海思正在和 Panaccess 合作，在 Hi3716MV310 上提供支持 Panaccess CA 的高安芯片。

3.3.8.1 Panaccess 高安芯片密钥组成

按照 Panaccess 要求，STB 厂商可能使用如表 3-5 所示的密钥。

表3-5 Panaccess 高安芯片密钥表

Key	Key-owner	Location	Description
CHIPID	Panaccess	OTP	Chipset unique ID. This value is defined by Panaccess. This value has already burned into chipset. It is the unique value of 8 bytes.
CSA2_RootKey	Panaccess	OTP	Panaccess secret key. This value has already burned into chipset.
R2R_RootKey	Panaccess	OTP	Panaccess secret key. This value has already burned into chipset.
JTAG_Key	Panaccess	OTP	Panaccess secret key. This value has already burned into chipset.
STB_ROOT_KEY	Panaccess	OTP	Panaccess static secret key. This value has already burned into chipset. It is used to encrypt the Boot-Image.
ROOT_RSA_PUB_KEY	Panaccess	OTP	Security SCS root key. This value has already burned into chipset.
EXT_RSA_PUB_KEY	STB Vendor	Flash	Secure SCS external key. Use by BootRom to verify “param” and check-area(BOOT) of BOOT in Flash.
EXT_RSA_PUB_KEY2	STB Vendor	Flash	Secure SCS second external key. Its public key is stored in BOOT code. Use to verify the signature of software images(bootargs, system, loader, stbid). Its private key is managed by STB vendor. It can be the same value of EXT_RSA_PUB_KEY.
HDCP_Key	STB Vendor	OTP	It is used to protect HDMI.

3.3.8.2 Panacces 码流解密

Panaccess 在码流解密中，要求 STB 厂商使用 DVB_RootKey(即 CSA2.0_RootKey)作为 DVB 业务根密钥，STB 厂商需要使用 3 级 Key-Ladder 对 TS 进行解密。

请参考 advca 提供的用例：sample_ca_panaccess_tsplay.c、sample_ca_crypto.c。

3.3.8.3 Panaccess 镜像签名

Panaccess 要求在机顶盒的 Flash 中保存的可执行镜像需要加密存储。按照这个要求，海思推荐如下的签名方案。

签名镜像包括如下 2 类：

- 对于 BOOT 镜像。STB 厂商可以使用 SDK 编译产生安全 BOOT 相关的镜像，包括：cfg.bin 和 fastboot-burn.bin。

  STB 厂商将上述镜像提交 Panaccess 签名，可以获得 Panaccess 签名的 BOOT。

  注意，Panaccess 返回的签名的 BOOT 是已经完成了签名和加密的二进制镜像。
- 对于非 BOOT 镜像，比如 bootargs、kernel、FS 等。

  “2.4 非 BOOT 软件安全校验”中定义的非 BOOT 软件安全校验可以作为 STB 厂商的一个参考方案。STB 厂商可以考虑使用“Specail CA Signature”模式对非 BOOT 软件镜像进行签名。

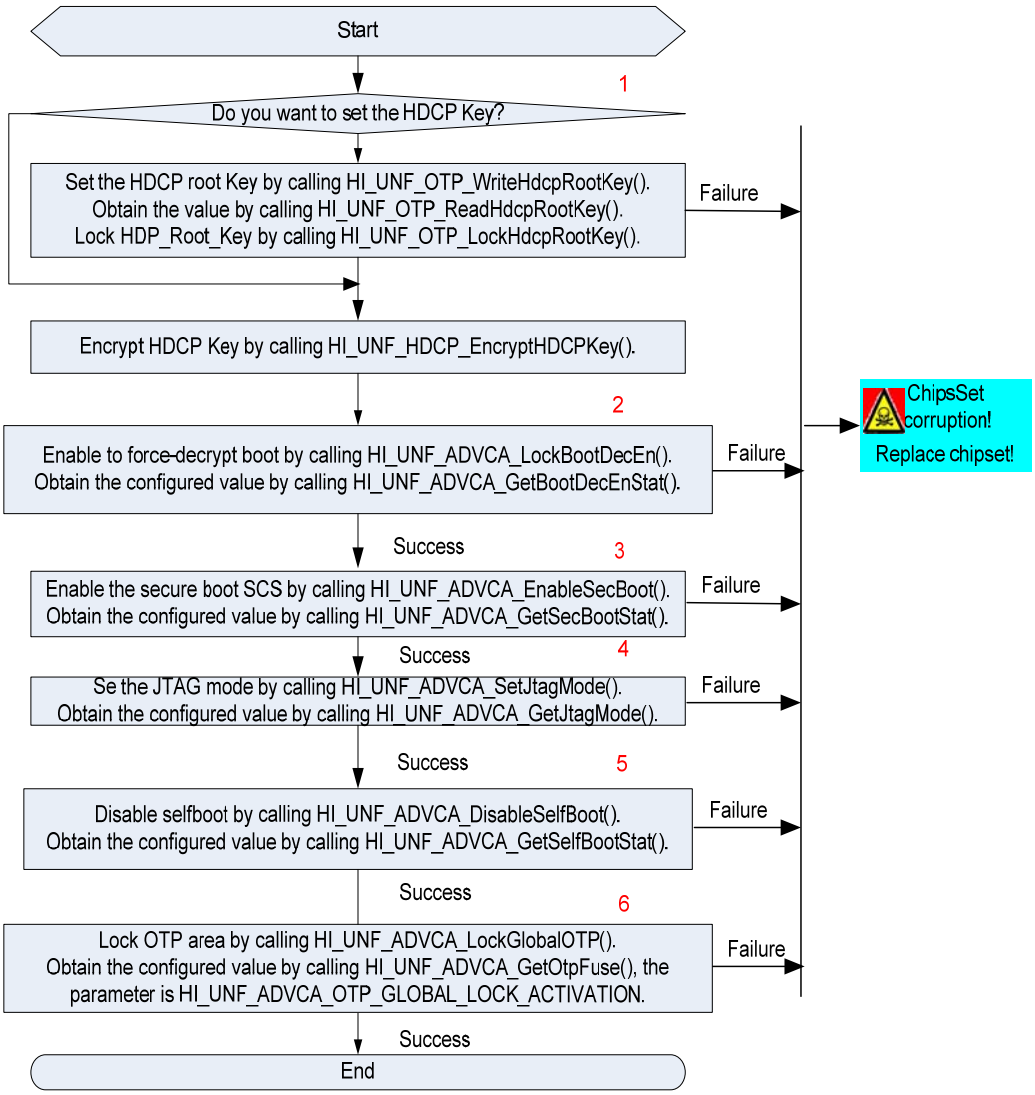
3.3.8.4 Panaccess 芯片 PV 设置

支持 Panaccess 安全级别的芯片为 Hi3716MV310。

按照 Panaccess 要求，Panaccess 高安芯片在芯片出厂前，就已将在芯片内置 Panaccess 密。包括 CHIPID，ROOT_RSA_PUB_KEY 等。

STB 厂商使用 Panaccess 高安芯片，进行 STB 整机认证和生产时，机顶盒需要设置的 PV 值如图 3-4 所示。

图3-4 Hi3716MV310 Panaccess 安全级别芯片 PV 设置流程图



STB 厂商应按照上面的流程设置 PV 值，上面的设置可以分多次进行，请按照上面设置顺序依次执行。

PV 设置的是 OTP 的数据，OTP 有部分数据必须单板重启后才能获得设置状态，在生产中，只需要判断设置 PV 的函数返回值是否为 HI_SUCCESS，即可以判断设置是否成功。

具体操作如下：

步骤 1 如果客户需要使用 HDCP key，请在这个阶段设置。详情请参考相关的 HDCP Key 使用指南。

步骤 2 设置强制 BOOT 解密功能。设置接口为：HI_UNF_ADVCA_LockBootDecEn()；获取设置值的接口为：HI_UNF_ADVCA_GetBootDecEnStat()。

步骤 3 设置安全启动 SCS 模式。设置接口为：HI_UNF_ADVCA_EnableSecBoot()；获取设置值的接口为：HI_UNF_ADVCA_GetSecBootStat()。机顶盒需要断电重启，才可以验证该功能。

步骤 4 禁止使用自举升级功能。设置接口为：HI_UNF_ADVCA_DisableSelfBoot()；获取设置值的接口为：HI_UNF_ADVCA_GetSelfBootStat()。

步骤 5 设置 JTAG 为 password 模式。设置接口为：HI_UNF_ADVCA_SetJtagMode ()；获取设置值的接口为：HI_UNF_ADVCA_GetJtagMode()。

步骤 6 设置 OTP 全局锁定控制位(可选项)。设置接口为：HI_UNF_ADVCA_LockGlobalOTP ()；获取设置值的接口为：HI_UNF_ADVCA_GetOtpFuse()，其参数为 HI_UNF_ADVCA_OTP_GLOBAL_LOCK_ACTIVATION。

警告

OTP 全局锁定控制位一旦设置，芯片 OTP 区间将无法改变，这个操作必须放在所有 OTP 操作（比如烧写 PV 值，烧写 HDCP_Root_Key 等）的最后。

------结束

3.3.9 海思非安全芯片

除了以上介绍的多种 CA 芯片，海思芯片还有非安全芯片和海思公共安全芯片（原”H”Mark 芯片），从 2015 开始，为解决机顶盒厂商统一备货的问题，海思芯片出厂时不再区分这两种芯片，机顶盒厂商可以根据市场需要，自行烧写为非安全芯片或海思公共安全芯片。

支持出厂后烧写的芯片包括 Hi3796MV100、Hi3798MV100、Hi3716MV410、Hi3716MV420、Hi3798CV200。

本节重点介绍出厂芯片如何烧写为非安全芯片或海思公共安全芯片，及其典型的量产流程。

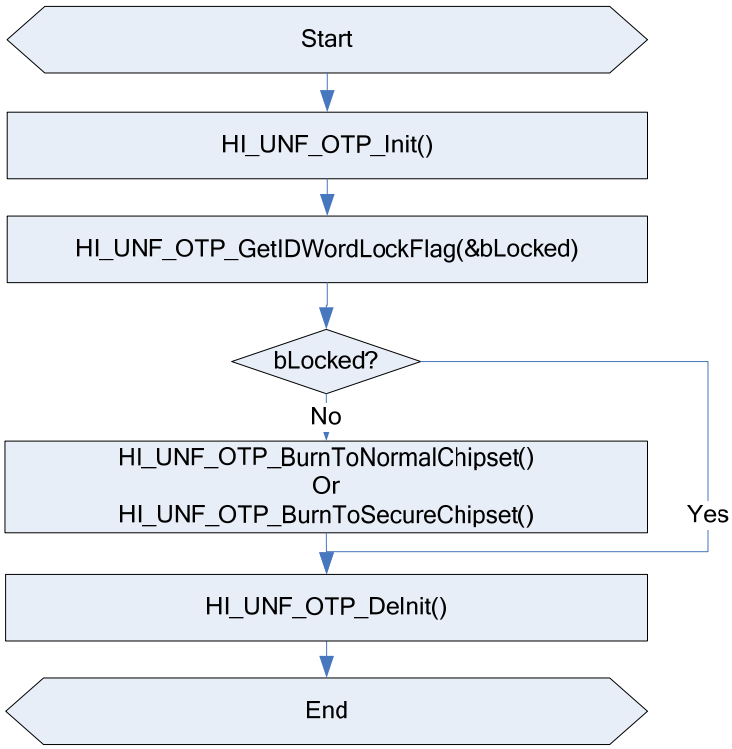
3.3.9.1 芯片烧写

SDK 提供了 UNF 接口：

- HI_S32 HI_UNF_OTP_BurnToNormalChipset(HI_VOID)
  将芯片烧写为非安全芯片；
- HI_S32 HI_UNF_OTP_BurnToSecureChipset(HI_VOID)
  将芯片烧写为海思公共安全芯片；
- HI_S32 HI_UNF_OTP_GetIDWordLockFlag(HI_BOOL *pbLockFlag)
  获取芯片烧写锁定状态。

典型锁定流程如图 3-5 所示。

图3-5 典型锁定流程



说明

开发调试阶段，为了避免跟您的其他烧写动作互相影响，建议烧写完成后重启单板；量产阶段原则上不需要重启。

Boot 下和系统下都提供了这三个接口，厂商可以根据自己生产方便，灵活选择在 boot 下烧写或在厂测软件中烧写。海思推荐在厂测软件中烧写的方式，原因如下：

- 如果在 boot 下自动烧写，开发调试只要单板启动一遍就会被烧写，需要管理安全、非安全单板，使用不方便；
- 因为第一次启动时芯片还是非安全芯片，所以 Flash 烧片时需要烧写高安未签名 boot，厂测完成后再烧签了名的高安 boot 镜像。如果在 boot 启动时自动烧写为海思公共安全芯片，一旦在烧 Finalboot 前单板重启，就会出现 boot 和芯片不匹配的情况，单板将无法启动，维修也比较麻烦。

所以推荐在厂测软件中烧写，将没有以上问题。

说明

Hi3798CV200 使用了统一 boot 方案，不会出现 boot 和芯片不匹配的情况，具体参考“Hi3798CV200 海思公共安全芯片量产”章节。

Hi3796MV100/ Hi3798MV100 会默认在 boot 下自动烧写（如果编译的是非高安 boot，会烧写为非安全芯片，如果编译的是高安 boot，会烧写为海思公共安全芯片），如果想关闭自动烧写功能，请注掉 Code/source/boot/product/main.c 中 HI_OTP_LockIdWord() 语句。

Hi3716MV410、Hi3716MV420、Hi3798CV200 默认不在 boot 下烧写，请厂商在厂测软件中烧写。

烧写成功之后，非安全芯片在启动时 CPU 类型打印如下：

```
CPU: Hi3796Mv100
```

烧写成为海思公共安全芯片启动后 CPU 类型打印如下：

```
CPU: Hi3796Mv100(CA)
```

3.3.9.2 Hi3796MV100、Hi3798MV100 非安全芯片量产

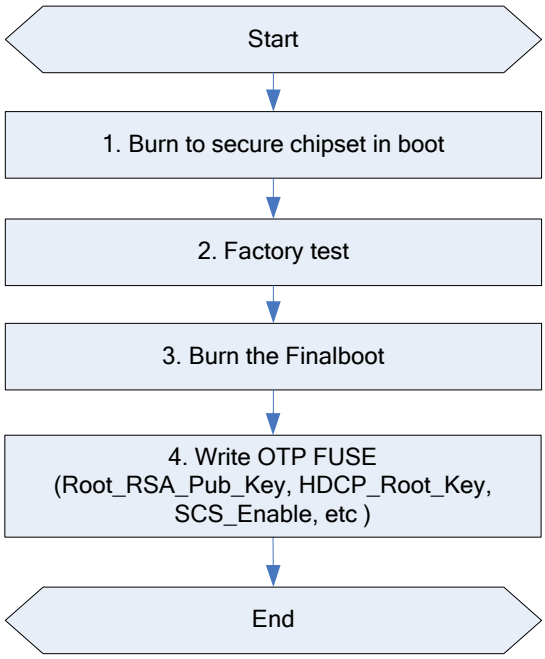
Hi3796MV100、Hi3798MV100 默认在 boot 启动时完成烧写，如果编译的是非安全 boot，会默认烧写为非安全芯片，对原有生产流程无影响。

3.3.9.3 Hi3796MV100、Hi3798MV100 海思公共安全芯片量产

Hi3796MV100、Hi3798MV100 默认在 boot 启动时完成烧写，如果编译的是高安 boot，会烧写为海思公共安全芯片。因为第一次启动时芯片还是非安全芯片，所以 Flash 烧片时需要烧写高安未签名 boot，boot 起来后自动烧写为海思公共安全芯片，厂测完成后再烧 Finalboot， 然后再烧写包括 Root_RSA_Pub_ Key 和安全启动使能在内的 OTP FUSE，重新启动后整套环境即为安全环境。

典型量产流程如图 3-6 所示。

图3-6 典型量产流程



具体说明：

步骤 1 高安未签名 boot 启动，会自动烧写为海思公共安全芯片；

步骤 2 出厂测试，基本功能、输入输出端口测试等；

步骤 3 烧写 Finalboot；

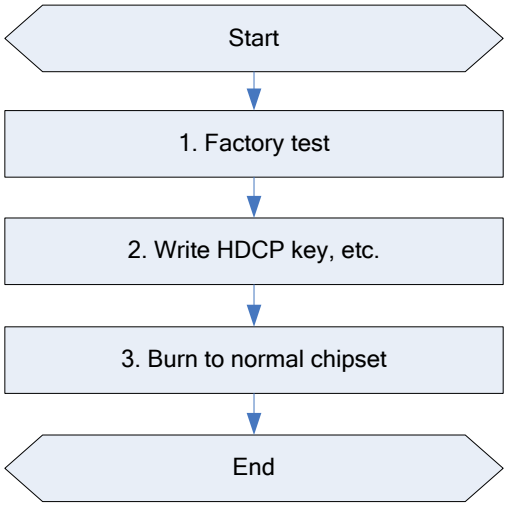
步骤 4 烧写 root key， HDCP key，PV 值等。，具体请参考 3.3.10 海思公共安全芯片。

----结束

3.3.9.4 Hi3716MV410、420 非安全芯片量产

跟 Hi3796MV100、Hi3798MV100 不同在于，Hi3716MV410、Hi3716MV 420 没有在 boot 中自动烧写，需要厂商在厂测软件中烧写，典型流程如图 3-7 所示。

图3-7 典型流程



具体说明：

步骤 1 出厂测试，基本功能、输入输出端口测试等；

步骤 2 烧写 HDCP key 等，视需要确定；

步骤 3 烧写为非安全芯片。

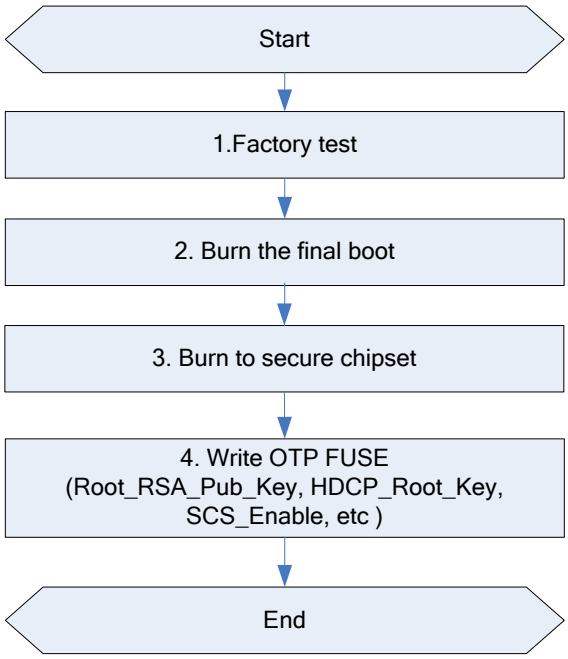
----结束

3.3.9.5 Hi3716MV410、420 海思公共安全芯片量产

跟 Hi3796MV100、Hi3798MV100 不同在于，Hi3716MV410、420 没有在 boot 中自动烧写，需要厂商在厂测软件中烧写。因为第一次启动时芯片还是非安全芯片，所以 Flash 烧片时需要烧写高安未签名 boot，厂测完成后再烧 Finalboot，然后再烧写包括 Root_RSA_Pub_ Key 和安全启动使能在内的 OTP FUSE，重新启动后整套环境即为安全环境。

典型流程如图 3-8 所示。

图3-8 典型流程



具体说明：

步骤 1 出厂测试，基本功能、输入输出端口测试等；

步骤 2 烧写 Finalboot；

步骤 3 烧写成为海思公共安全芯片。

步骤 4 烧写 root key，HDCP key，PV 值等。具体请参考 3.3.10 海思公共安全芯片。

----结束

3.3.9.6 Hi3798CV200 非安全芯片量产

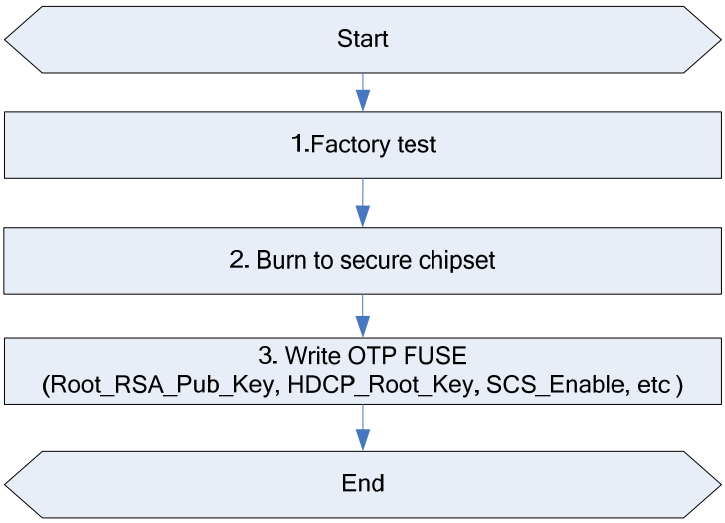
同 Hi3716MV410、Hi3716MV420 非安全量产流程。

3.3.9.7 Hi3798CV200 海思公共安全芯片量产

Hi3798CV200 芯片进行了统一 boot 设计，即非高安 boot、高安未签名 boot、高安签名 Finalboot 可以兼容，非安全芯片烧写高安未签名 boot、高安签名 Finalboot 也可以启动。所以量产时 Flash 烧片阶段就可以直接烧高安签名 Finalboot，量产阶段不再需要烧 boot 动作。

典型流程如图 3-9 所示。

图3-9 典型流程



具体说明：

步骤 1 出厂测试，基本功能、输入输出端口测试等；

步骤 2 烧写为海思公共安全芯片；

步骤 3 烧写 root key，HDCP key，PV 值等。具体请参考 3.3.10 海思公共安全芯片。

----结束

3.3.10 海思公共安全芯片

海思公共安全芯片是指原"H" Mark 芯片，STB 厂商可用此类芯片实现部分安全功能，如安全启动和防抄板等。

Hi3716CV200/Hi3719CV100/Hi3716MV310//Hi3110EV500 等型号海思提供"H" Mark 芯片；

Hi3796MV100/Hi3798MV100/Hi3716MV410/Hi3716MV420/Hi3798CV200 等型号海思不再提供"H" Mark 芯片，厂商可根据需要将出厂芯片烧写为海思公共安全芯片，具体可参考 3.3.9 章节。烧写后的海思公共安全芯片也需参考本节描述根据需要设置 PV 值。

3.3.10.1 海思公用高安芯片 STB 厂商密钥

实现安全启动要求，STB 厂商需要设置如表 3-6 所示 key。

表3-6 STB 厂商需要设置的 key

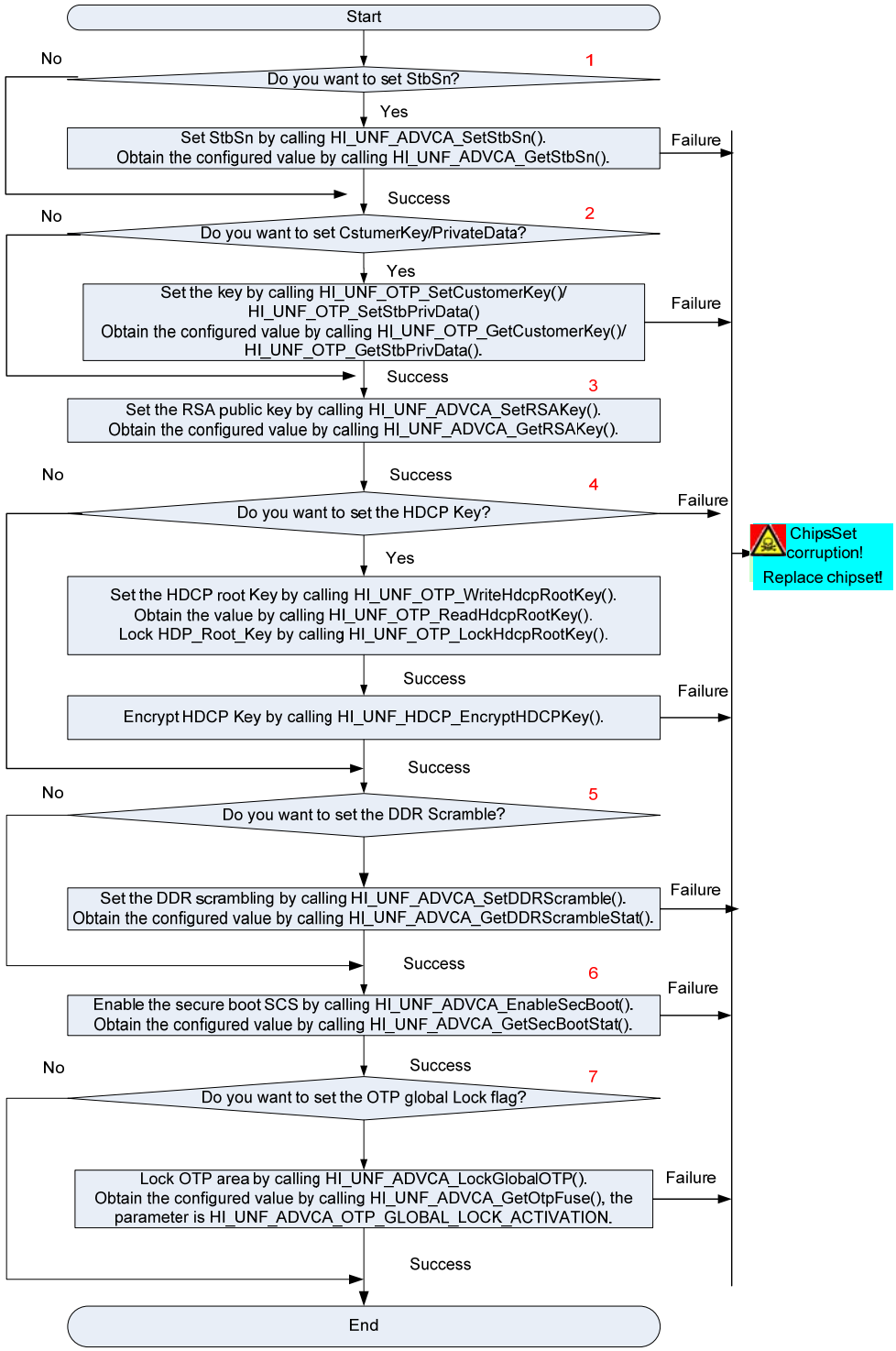
Key	Key-owner	Location	Decription
Root_RSA_Pub_ Key	STB Vendor	OTP	Secure SCS root key. Burnt by the STB Vendor. Only use by BootRom to verify Key-Area of Bootloader in Flash.

Key	Key-owner	Location	Decription
Ext_RSA_Pub_Key	STB Vendor	FALSH	Secure SCS external key. Only use by BootRom to verify “param” and check-area(boot) of Bootloader in Flash.
Ext_RSA_Pub_Key2	STB Vendor	FLASH	Secure SCS second external Key. Its public key is stored in Finalboot. Use to verify the signature of software images(bootargs, system, loader, stbid). Its private key is managed by STB vendor. It can be the same value of Ext_RSA_Pub_Key.
HDCP_Root_Key	STB Vendor	OTP	It is used to protect HDCP key of HDMI.
CustomerKey	STB Vendor	OTP	STB private data, burnt by the STB Vendor.(Optional)
StbPrivateData	STB Vendor	OTP	STB private data, burnt by the STB Vendor. (Optional)

3.3.10.2 PV 设置

STB 厂商使用海思公用安全芯片启动安全特性，如图 3-10 所示。

图3-10 Hi3716CV200 海思公用高安芯片 PV 设置流程图



STB 厂商应按照上面的流程设置 PV 值，上面的设置可以分多次进行，但是不要尝试改动设置顺序。

PV 设置的是 OTP 的数据，OTP 有部分数据必须单板重启后才能获得设置状态，在生产中，只需要判断设置 PV 的函数返回值是否为 HI_SUCCESS，即可以判断设置是否成功。

具体操作如下：

步骤 1 设置 StbSn。

- 设置接口为：HI_UNF_ADVCA_SetStbSn()；
- 获取设置值接口为：HI_UNF_ADVCA_GetStbSn()。

步骤 2 设置 CustomerKey/StbPrivateData。此类接口使用与存储客户私有数据，CustomerKey 和 StbPrivateData 的大小分别为 16 个字节。

- 设置接口为：

  HI_UNF_OTP_SetCustomerKey()/HI_UNF_OTP_SetStbPrivData()；
- 获取设置值的接口为：

  HI_UNF_OTP_GetCustomerKey()/HI_UNF_OTP_GetStbPrivData()。

步骤 3 需要设置 Root RSA Public Key。

- 设置接口为：HI_UNF_ADVCA_SetRSAKey()；
- 锁定 RSAKey 的接口为：HI_UNF_ADVCA_ConfigLockFlag()。
- 获取设置值的接口为：HI_UNF_ADVCA_GetRSAKey()。

步骤 4 如果客户需要烧写 HDCP_Root_key，请在这个阶段烧写。Hi3716CV200 以及后续芯片在使用 HDCP 的方式已该变,详情请参考 HDCP KEY 使用文档。

- 设置 HDCP Root Key 接口为：HI_UNF_OTP_WriteHdcpRootKey ();
- 获取设置值的接口为：HI_UNF_OTP_ReadHdcpRootKey ();
- 锁定该密钥的接口为：HI_UNF_OTP_LockHdcpRootKey();
- 使用 HDCP_Root_key 加密 HDCP Key 的接口为：HI_UNF_HDCP_EncryptHDCPKey().

步骤 5 设置 DDR 加扰功能。

- 设置接口为：HI_UNF_ADVCA_SetDDRScramble()；
- 获取设置值的接口为：HI_UNF_ADVCA_GetDDRScrambleStat()。

步骤 6 设置安全启动 SCS 模式。

- 设置接口为：HI_UNF_ADVCA_EnableSecBootEx()
- 获取设置值的接口为：HI_UNF_ADVCA_GetSecBootStat()。

该控制位设置后,机顶盒厂商应该将机顶盒下电重启验证该功能后,再执行下一步设置 JTAG 控制操作。

步骤 7 设置 OTP 全局锁定控制位(可选项)。设置接口为：HI_UNF_ADVCA_LockGlobalOTP ()；获取设置值的接口为：HI_UNF_ADVCA_GetOtpFuse()，其参数为 HI_UNF_ADVCA_OTP_GLOBAL_LOCK_ACTIVATION。

警告

OTP 全局锁定控制位一旦设置，芯片 OTP 区间将无法改变，这个操作必须放在所有 OTP 操作（比如烧写 PV 值，烧写 HDCP_Root_Key，STB_Root_Key 等）的最后。

----结束

3.3.10.3 海思公共安全芯片 CAS 厂商密钥

某些情况下机顶盒厂商会使用海思公共安全芯片支持特定的 CAS 功能，可能需要设置如表 3-4 所示的 CAS 私有数据。

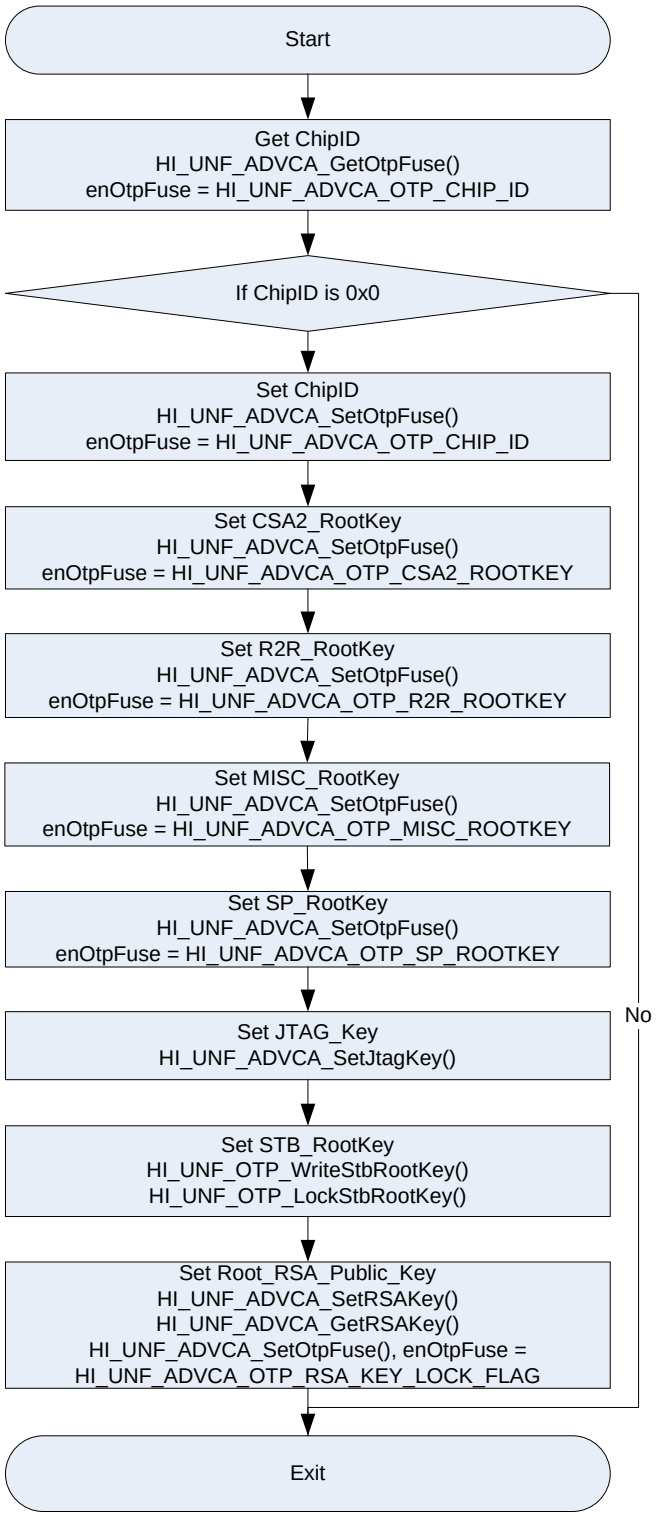
表3-7 CAS 厂商私有数据

Key	Key-owner	Location	Decription
CHIPID	CA	OTP	Chipset unique ID
Root_RSA_Pub_ Key	CA	OTP	Secure SCS root key. Burnt by the STB Vendor. Only use by BootRom to verify Key-Area of Bootloader in Flash.
CSA2_RootKey	CA	OTP	Root key for CSA2 key ladder
CSA3_RootKey	CA	OTP	Root key for CSA3 key ladder()
R2R_RootKey	CA	OTP	Root key for R2R key ladder
SP_RootKey	CA	OTP	Root key for SP key ladder
MISC_RootKey	CA	OTP	Root key for MISC key ladder
JTAG_Key	CA	OTP	Jtag password
STB_RootKey	CA	OTP	Root key for encrypt boot

3.3.10.4 CAS 厂商私有数据设置

CAS 私有数据设置如图 3-11 所示。

图3-11 CAS 私有数据设置



具体操作如下：

步骤 1 获取 ChipID，如果为 0x0，则设置 ChipID，否则退出。

步骤 2　设置 CSA2_RootKey。

步骤 3　设置 R2R_RootKey。

步骤 4　设置 MISC_RootKey。

步骤 5　设置 SP RootKey。

步骤 6　设置 Jtag RootKey。

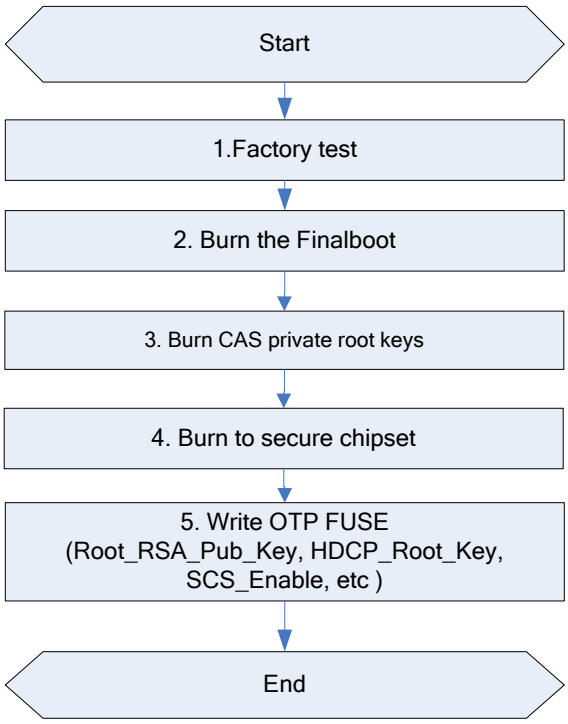
步骤 7　设置 STB RootKey。

步骤 8　设置 Root RSA Public Key。

----结束

CAS 私有数据的烧写处于将芯片烧写成为海思公共安全芯片之前，以 Hi3716MV410 为例，流程如图 3-12 所示。

图3-12 包含烧写 CAS 私有数据的量产流程



具体操作如下：

步骤 1　出厂测试，基本功能、输入输出端口测试等。

步骤 2　烧写 Finalboot。

步骤 3　烧写 CAS 私有数据。

步骤 4　烧写成为海思公共安全芯片。

步骤 5 烧写 root key，HDCP key，PV 值等。具体请参考 3.3.10 海思公共安全芯片。

----结束