GR5xx固件加密及应用介绍

版本： **3.0**

发布日期： **2023-03-30**

深圳市汇顶科技股份有限公司

商标声明

GOODiX 和其他汇顶商标均为深圳市汇顶科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人持有。

免责声明

本文档中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

深圳市汇顶科技股份有限公司（以下简称“GOODIX”）对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。GOODIX对因这些信息及使用这些信息而引起的后果不承担任何责任。

未经GOODIX书面批准，不得将GOODIX的产品用作生命维持系统中的关键组件。在GOODIX知识产权保护下，不得暗中或以其他方式转让任何许可证。

深圳市汇顶科技股份有限公司

总部地址：深圳市福田保税区腾飞工业大厦B座12-13层

电话：+86-755-33338828　　邮编：518000

网址：www.goodix.com

前言

编写目的

本文档介绍GR5xx系列芯片的安全模式所采用的固件加解密、消息认证和数字签名技术等，旨在帮助开发者了解并应用GR5xx系列芯片的安全模式。

读者对象

本文适用于以下读者：

- 芯片用户
- 开发人员
- 测试人员
- 开发爱好者
- 文档工程师

版本说明

本文档为第2次发布，对应的产品为低功耗蓝牙GR5xx系列。

修订记录

版本	日期	修订内容
1.0	2023-01-10	首次发布
3.0	2023-03-30	新增支持多款芯片的相关描述

目录

1 简介

物联网的快速发展对低功耗蓝牙芯片的安全性能提出更高的要求。GR5xx系列低功耗蓝牙芯片的安全性能包括保护芯片内部固件、数据以及系统功能。安全算法以及安全模块的结合，可对位于存储器上的用户数据以及应用固件进行保护，防止被窃听者获取。

本文档分为原理、实现及应用三大部分。

- 2 安全模块与算法章节主要介绍GR5xx系列芯片上的安全模块以及采用的安全算法。
- 3 系统配置区消息认证章节至5 数字签名技术章节主要介绍安全模式的构建方法。
- 6 安全模式应用章节主要介绍如何使用配套工具完成安全模式的应用。

1.1 安全解决方案

GR5xx系列芯片采用芯片内部加密引擎与外部配套工具相配合的方式，提供一套完整的安全解决方案。开发者可以通过工具（如GProgrammer）对芯片进行安全配置，随后通过芯片内部ROM发起的安全启动，实现对芯片的保护。

安全解决方案主要包括系统配置区（System Configuration Area，SCA）的消息认证、固件及数据加解密和数字签名。在3 系统配置区消息认证章节至5 数字签名技术章节中，将对以上内容进行详细说明。

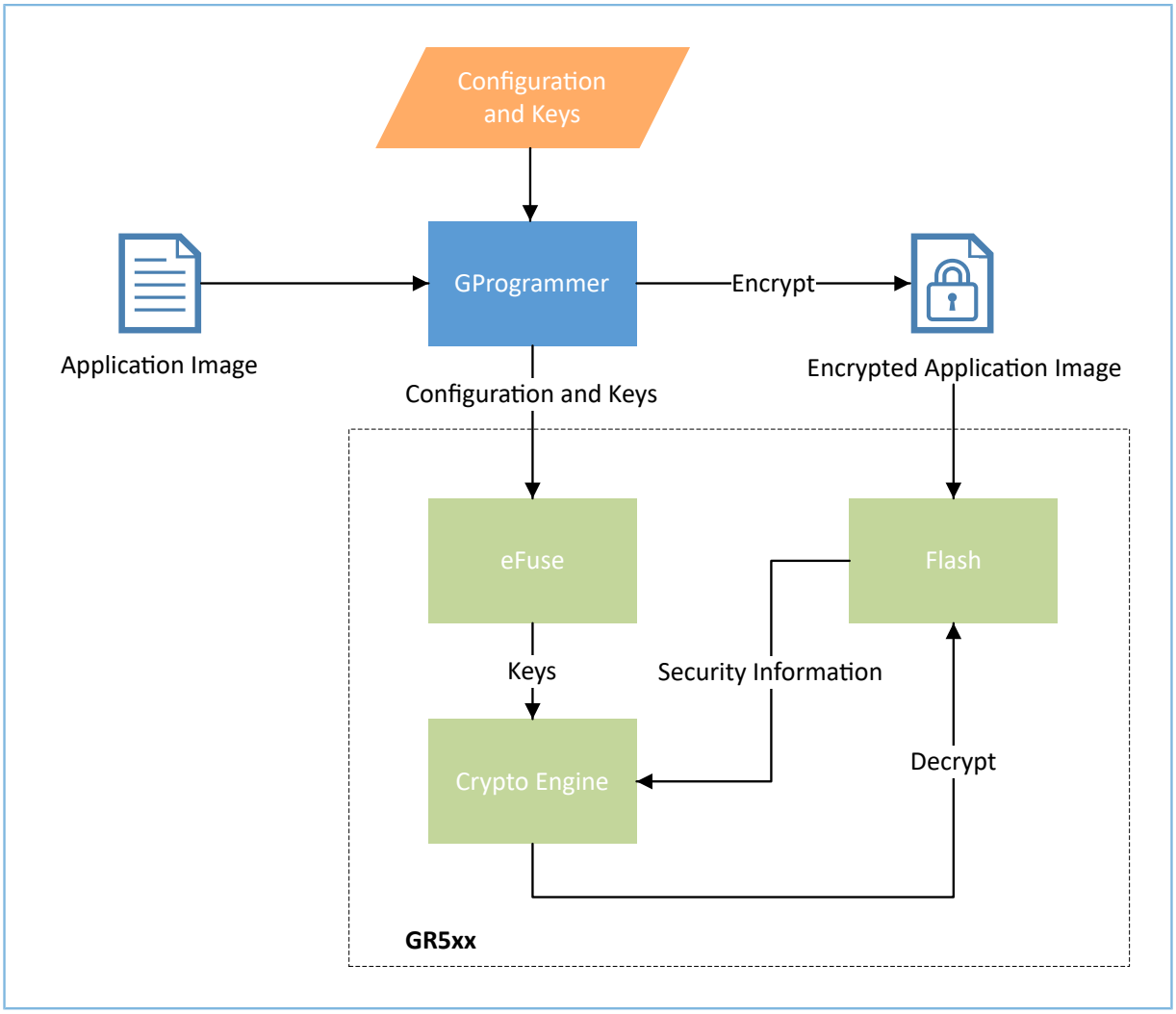

图 1-1 安全系统框图

- Application Image为编译生成的可执行的二进制文件，其格式通常为.hex或.bin文件。
- GProgrammer是一款支持GR5xx系列芯片的固件烧录工具，包含对芯片进行安全配置的功能。
- Encrypted Application Image为经过工具处理后的加密固件，其本身携带Security Information供芯片验证，其格式通常为.bin文件。

1.2 安全特性

- 安全的密钥存储

  采用eFuse管理安全密钥，无法通过MCU直接读取eFuse存储的密钥信息。安全模式下，芯片会通过加密的方式把密钥信息加载到KEYRAM中，并产生真随机数作为掩码。MCU无法直接读取KEYRAM中

的密钥信息。当安全模块需要使用密钥信息时，芯片将通过内部独立的硬件单元实现密钥的自动导入。

- 防止固件窃取

  采用对称密码与公钥密码相结合的密码方式，解密时需要采用椭圆曲线密码（ECC）算法结合eFuse存储的私钥计算出PRESENT-128模块解密所需的密钥。窃听者即使获取Flash中存储的加密固件，但由于无法获取到密钥，依然无法使用该加密固件。

- 防止恶意攻击

  提供SWD锁和安全的DFU固件升级方式防止Flash存储的应用固件被恶意修改。在量产阶段，可以通过修改eFuse中存储的配置信息关闭SWD功能，防止攻击者读取或修改固件信息。当SWD功能关闭后，开发者依然可以通过DFU对固件进行升级。在DFU升级过程中配合使用App Bootloader流程，会对加密固件信息进行验签，以防止恶意修改Flash存储的固件信息。

- 支持一机一密

  将固件密钥和数据密钥分别存储在eFuse的不同区域，相同的应用固件可烧录在使用不同数据密钥的芯片中，开发者可利用数据密钥对存储的数据进行加密，以实现一台设备对应唯一数据密钥的目的。

2 安全模块与算法

本章主要介绍GR5xx系列芯片上的安全模块以及采用的安全算法。

2.1 基本概念

2.1.1 密码系统

- 对称密码系统：又称单密钥加解密系统，该系统中同一个密钥被用作信息的加密和解密，即信息的发送方和接收方需使用同一个密钥进行数据的加解密。
- 公钥密码系统：又称非对称加解密系统，它使用了一对密钥：公钥（Public Key）和私钥（Private Key）。在使用密钥对进行加解密时，若其中一个密钥用于加密，则另一个密钥用于解密。例如，如果加密时使用公钥（Public Key），那么解密时将使用私钥（Private Key）。
- 混合密码系统：将对称密码与公钥密码相结合的密码方式。使用对称密码提高加解密速度，使用公钥密码解决密钥配送问题。

2.1.2 消息认证

- 数字摘要：数字摘要是将任意长度的消息变成固定长度的短消息，通常使用单向散列函数（哈希函数）实现。
- 消息认证码：提供了一种消息认证机制，不仅可检验消息是否被篡改，还可确认消息是否来自所期望的通信对象。

2.1.3 数字签名

数字签名：只有信息的发送者才能产生的无法伪造的一段数字串，这段数字串同时也是对信息的发送者发送信息真实性的一个有效凭证。相较于消息认证，数字签名具有不可抵赖性（不可否认性）。

2.2 安全模块

安全模块主要包含TRNG、PRESENT-128、eFuse、KEYRAM、PKC、HMAC、XIP_DEC硬件模块。

- **TRNG（True Random Number Generator）模块**

  在安全模块的应用中，TRNG主要用于生成加密解密时使用的随机数掩码。为了保证随机数质量并修正TRNG的偏差，TRNG模块中增加了LFSR（Linear Feedback Shift Register）和Post-Process逻辑。

- **PRESENT-128模块**

  PRESENT是一种轻量级分组密码。该算法以其尺寸紧凑（比AES算法小约2.5倍）著称，适用于低功耗和高效率的应用场景。

  PRESENT-128加解密模块通过其内部的两个64位PRESENT内核，可以在单次运行中实现128位数据的加密或解密。

- **eFuse模块**

  eFuse是一个具有随机访问接口的一次性可编程存储器，用于存储安全密钥和芯片校准数据等。GR5xx系列eFuse容量为512 Bytes。

- **KEYRAM模块**

  KEYRAM主要用于芯片上电后的密钥推导和存储。在安全启动流程中，每次启动将生成真随机数作为掩码（MASK），用于防止外界穷举法破解。安全模块如AES、HMAC、XIP_DEC等，可以通过KeyPort总线接口实现加密读取密钥。KEYRAM中存储的密钥无法通过CPU及调试端口进行访问。

- **PKC（Public Key Cryptography）模块**

  PKC控制器模块用于完成公钥算法中的基本底层模运算和FIPS标准256点ECC（Elliptic Curve Cryptography）点乘计算。

- **HMAC（Hash Message Authentication Code）模块**

  基于哈希函数构建的消息认证码使用完全符合联邦信息处理标准（FIPS PUB 198-1）中定义的密钥哈希消息认证码（HMAC）的算法实现消息认证验证过程。该模块支持多种模式（SHA256、HMAC-SHA256）。

- **XIP_DEC模块**

  XIP_DEC模块内嵌硬件PRESENT-128子模块实现XIP模式下实时解密读取固件指令或数据。

说明:

详细的模块介绍请参考对应芯片Datasheet。

2.3 安全算法

安全算法包含ECIES（ECC P-256和PRESENT-128）、HMAC-SHA256、PKCS#1 V2.1 RSASSA-PSS。

表 2-1 安全算法应用概况

应用场景	安全算法	密钥长度（bit）
系统配置区消息校验	HMAC-SHA256	256
固件加解密	ECIES（ECC P-256）	私钥：256 公钥：256*2+(8)
	ECIES（PRESENT-128）	128
数据加解密	PRESENT-128	128
数字签名	PKCS#1 V2.1 RSASSA-PSS	私钥：2048 公钥：2048+32+(2048+32)

2.3.1 ECIES算法

集成加密方案（Integrated Encryption Scheme，IES）是一种混合加密系统。椭圆曲线集成加密方案（Elliptic Curve Integrated Encryption Scheme，ECIES）是IES的一种变体，该方案通常是利用椭圆曲线密钥协商算法（ECDH）获取共享密钥，然后使用密钥派生函数（KDF）派生出单独的对称加密密钥和MAC密钥进行加解密和消息认证。

安全算法采用通过P-256椭圆曲线获取PRESENT-128密钥的方式构建ECIES混合加密系统，目的是解决密钥配送问题和提高加解密的速度。

2.3.2 椭圆曲线密码（ECC）算法

椭圆曲线密码学（Elliptic Curve Cryptography，ECC）是一种建立公开密钥加密的演算法，基于椭圆曲线数学。其数学基础是利用椭圆曲线上的有理点构成Abel加法群上椭圆离散对数的计算困难性。ECC的主要优势是在某些情况下它比其他的方法使用更小的密钥提供相当的或更高等级的安全。

表 2-2 同等安全条件下的密钥长度（单位为bit）

RSA/DSA	512	768	1024	2048	21000
ECC	106	132	160	211	600

与RSA相比，ECC具有以下优势：

- 安全性能更高：160位ECC相当于1024位RSA、DSA的安全强度。
- 处理速度更快：在私钥的处理速度上，ECC远比RSA、DSA快。
- 带宽要求更低、所占存储空间更小：ECC的密钥大小和系统参数比RSA、DSA更小。

2.3.3 PRESENT-128算法

PRESENT-128是一个轻量级分组密码算法，采用SPN结构设计，分组长度为64 bits，密钥长度128 bits，一共迭代31轮。PRESENT-128密码算法与现有的轻量级分组密码算法TEA、MCRYPTON、HIGHT、SEA和CGEN相比，具有更简单的硬件实现和简洁的轮函数设计。

2.3.4 HMAC-SHA256算法

HMAC-SHA256算法，即基于SHA-256哈希函数构建的HMAC算法。根据HMAC算法和SHA-256算法的要求，HMAC-SHA256算法的明文分组长度为512 bits，可通过任意长度密钥（最小推荐长度为256 bits），计算出长度为256 bits的消息认证码。

2.3.5 RSASSA-PSS算法

RSA数字签名算法（RSASSA）的本质，仍然是RSA加密/解密算法。PSS（Probabilistic Signature Scheme）是私钥签名流程的一种填充模式。RSASSA-PSS算法本质就是在RSA算法的基础上叠加上一种填充算法。

- 填充性质：基于PKCS#1 V1.5签名算法是确定性签名算法，即对于同样的消息和私钥，输出是固定的；PKCS#1 PSS签名算法是概率签名算法，即使对于同样的消息和私钥，每次签名输出完全不一致（因为内部引入了随机数），但均可以通过对应公钥的验签。
- 安全性：对于PKCS#1 V1.5，公开指数过短（如e=3）会导致密文操纵攻击（即密文变换后仍符合填充规范），因此对于PKCS1#V1.5填充方案，公开指数e取值一般为65537；而采用PKCS#1 PSS算法即使公开指数e=3仍然安全。

3 系统配置区消息认证

系统配置区（System Configuration Area，SCA）位于Flash的前两个Sector，共8 KB，地址：0x0020_0000 ～ 0x0020_2000（GR551x为0x0100_0000 ～ 0x0100_2000）。SCA中存储了系统启动过程使用的标志以及其他系统配置参数。

本章将介绍GR5xx芯片对系统配置区的消息认证码（HMAC）进行验证的流程。

说明:

详细的SCA介绍请参考对应芯片开发者指南。

3.1 HMAC验证流程

在安全模式下，使用GProgrammer等工具更新系统配置区时，系统配置区中的HMAC会同步更新。安全启动时，芯片将首先对系统配置区的HMAC进行验证，若验证通过则继续进行其他安全启动流程（解密、验签），若验证不通过则芯片将进入DFU模式。

HMAC验证流程如图 3-1所示。

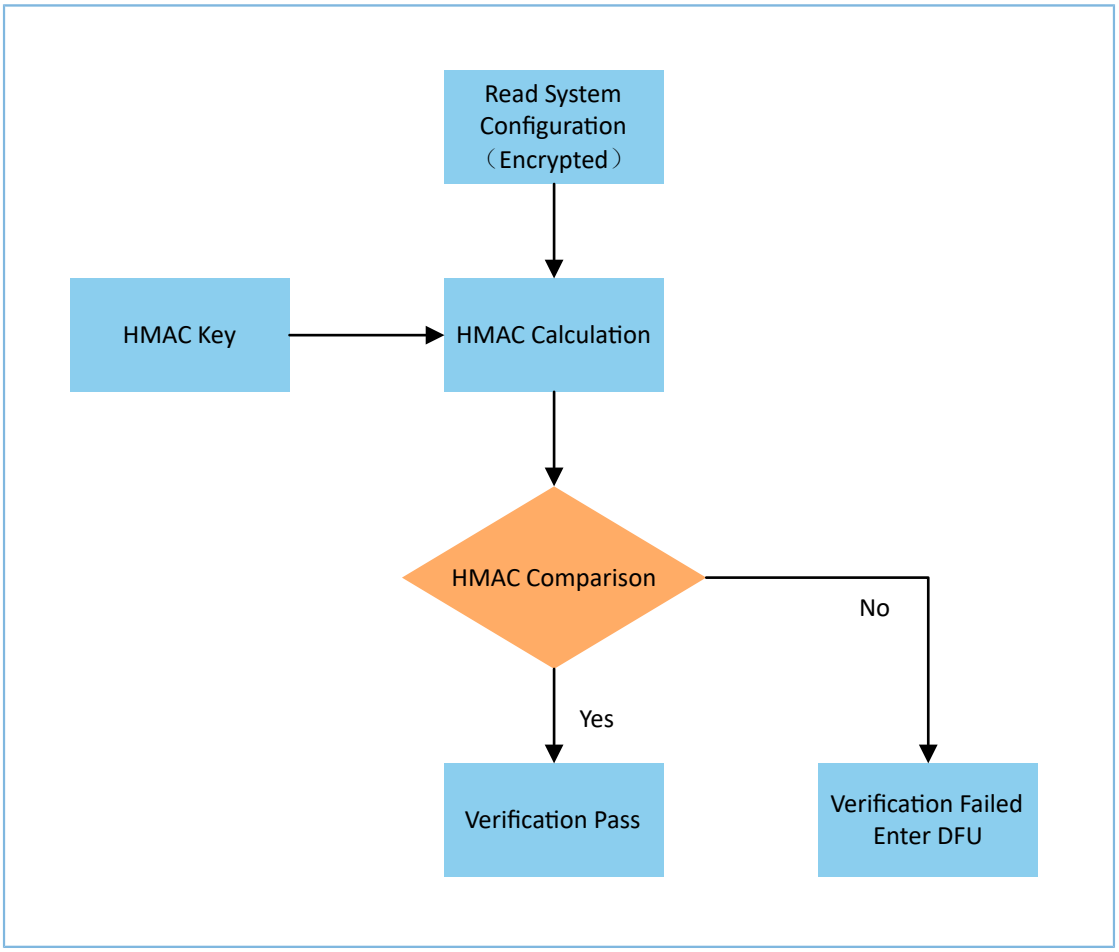

图 3-1 HMAC验证流程

HMAC验证流程具体步骤：

1. 读取系统配置区信息（该信息为加密信息）。
2. 使用存储在eFuse中的HMAC密钥对系统配置区信息（除去携带的HMAC）进行HMAC计算（采用HMAC-SHA256算法）。
3. 对比计算获得的HMAC和系统配置区信息携带的HMAC，若两者一致则验证通过。

4 固件和数据安全

4.1 固件加解密流程

GR5xx采用了椭圆曲线集成加密方案（ECIES），该方案属于混合密码系统。通过对称密码（PRESENT-128）和公钥密码（椭圆曲线密码ECC）两种算法混合来保证固件的安全性和高效性。通过使用混合密码系统，在通信中能够将对称密码和公钥密码的优势结合起来。

混合密码系统的加解密流程如下图所示。

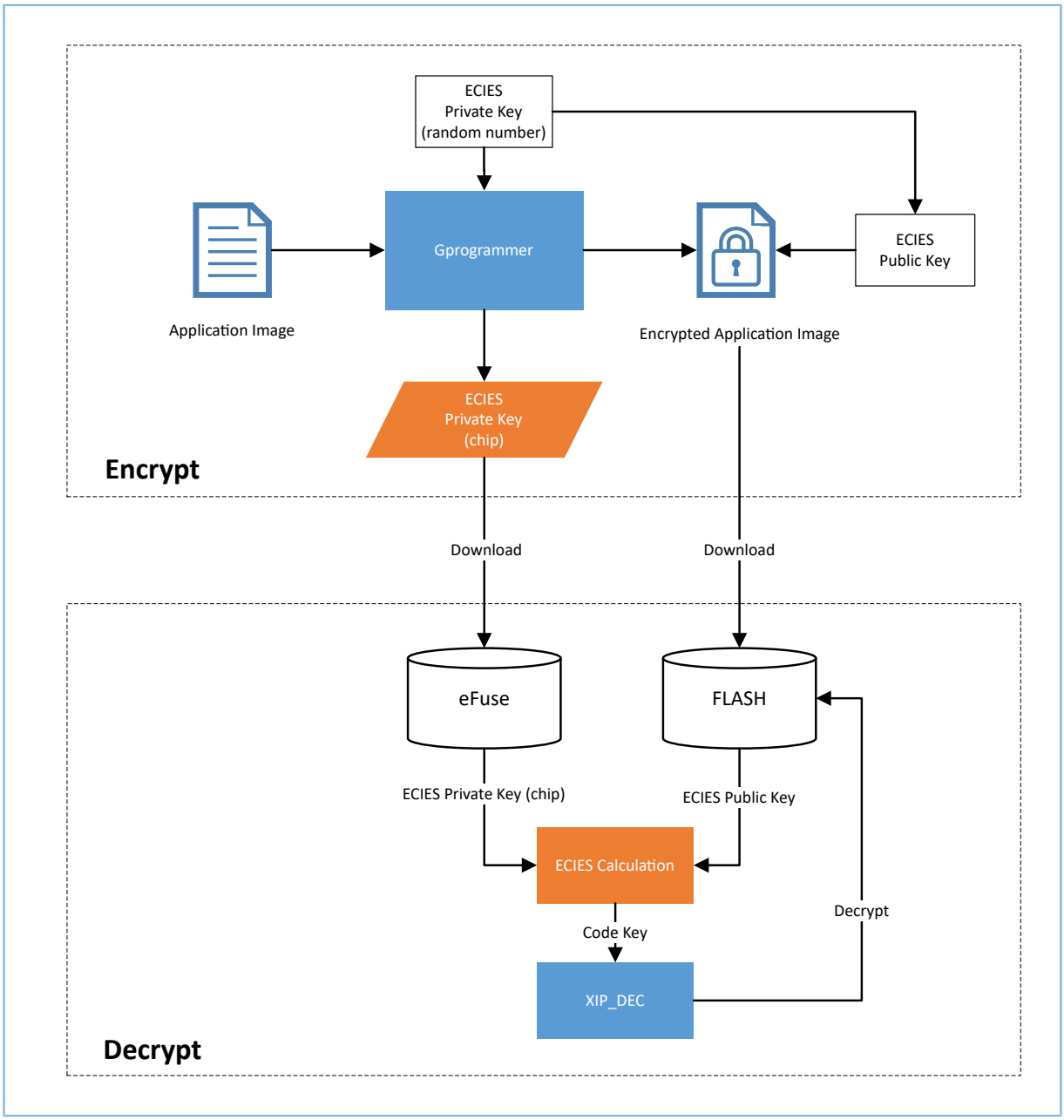

图 4-1 混合密码系统加解密流程

- 混合密码系统的加密流程

1. 生成ECIES随机私钥：通过工具（如GProgrammer）的随机数生成器生成ECIES加密中使用的随机私钥。
2. 获取会话密钥：GProgrammer工具使用生成的ECIES随机私钥进行ECIES计算，获取会话密钥（PRESENT-128对称密钥）和ECIES随机公钥。
3. 生成密文：GProgrammer工具使用会话密钥加密会话消息，并与ECIES随机公钥组合后生成会话消息密文。
4. 导入私钥及密文：通过GProgrammer工具可以将椭圆曲线ECC解密所用的FW Key烧录至eFuse中，并将会话消息密文烧录至Flash存储器中。

- 混合密码系统的解密流程

1. 获取ECC私钥：椭圆曲线ECC解密所用的FW Key存储在eFuse中。在系统启动初始化过程中，该密钥将被加载到KEYRAM模块中。随机数生成器模块与eFuse的配合使得该密钥难以被外部获取。
2. 获取会话密钥：通过PKC模块使用ECIES随机公钥和FW Key进行ECIES计算，获取PRESENT-128算法所需的对称密钥FW Code Key，并将该密钥加载到KEYRAM模块中。
3. 解密密文：将对称密钥FW Code Key加载到XIP_DEC硬件模块实现Flash存储器上Code的自动解密。

4.2 数据加解密流程

为保证Flash存储器所存用户数据（比如需安全保存的用户信息、Sensor采样数据等）的安全性，安全模块支持对该类数据进行基于PRESENT-128的轻量级对称加密。

数据加解密的流程如图 4-2所示。

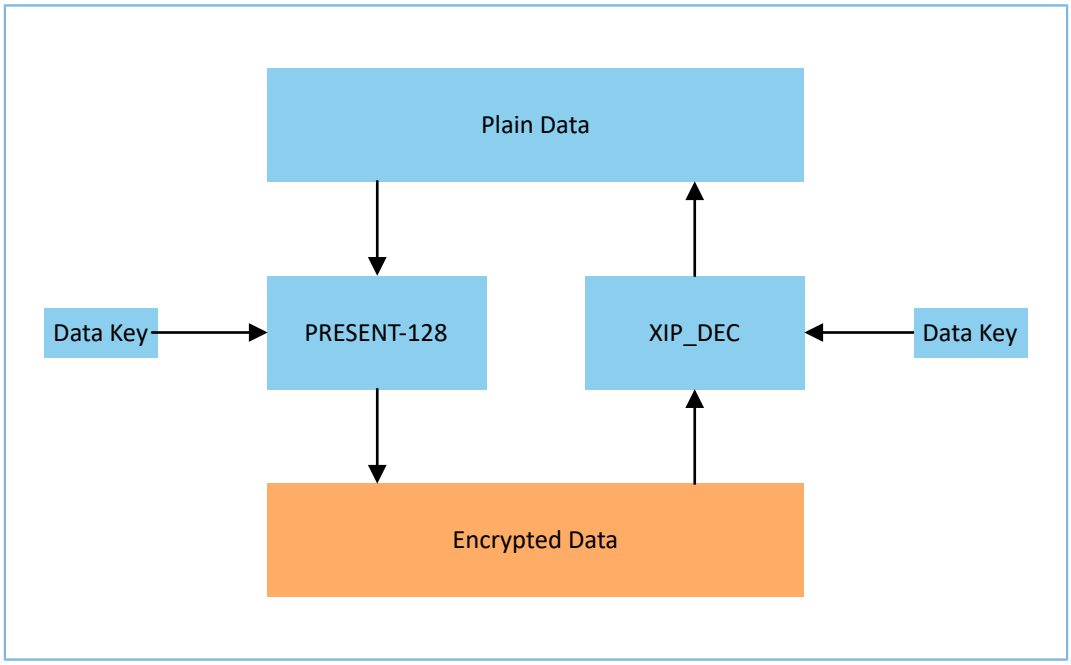

图 4-2 数据加解密流程

该流程具体描述如下：

1. PRESENT-128算法所用的对称密钥Data Key存储于eFuse中。在系统启动初始化过程中，该对称密钥将被加载到KEYRAM模块中。随机数生成器模块与eFuse的配合使得对称密钥难以被外部获取。
2. 在数据加密流程中，将对称密钥Data Key加载到PRESENT-128模块实现将数据加密存储在Flash存储器中。
3. 在数据解密流程中，将对称密钥Data Key加载到XIP_DEC硬件模块实现存储器上数据的自动解密。

说明:

- 固件密钥和数据密钥在eFuse中是分开存储的，某个加密固件想在不同芯片上运行，需要将同个固件密钥烧录在不同的芯片中，但数据密钥允许存在差异，以实现一机一密的功能。
- Flash接口支持加密读写，该接口会自动判断芯片是否进入了安全模式，然后使用数据密钥对写入/读取的数据进行加/解密。开发者仍可调用相应接口禁用加密读写，从而向Flash中读写明文。
- Flash接口暂不支持使用固件密钥进行加密读写，固件密钥当前仅用于XIP_DEC解密Flash中Code区域的内容。
- 当使用Flash加密存储数据时，也可以直接调用相应的Non-volatile Data Storage（NVDS）接口。关于NVDS的详细描述，请参考对应芯片开发者指南。

---

5 数字签名技术

数字签名是一种提供认证并防止否认的密码技术，能够确保消息的完整性和真实性。

安全模式下的数字签名流程（见图 5-1）包括：

- 加签：用户可通过GProgrammer对固件进行数字签名，并且可以将验签需要的相关信息下载到eFuse和Flash中。
- 验签：Bootloader在启动过程中从eFuse和Flash中拿到有关信息，以验证固件的签名。

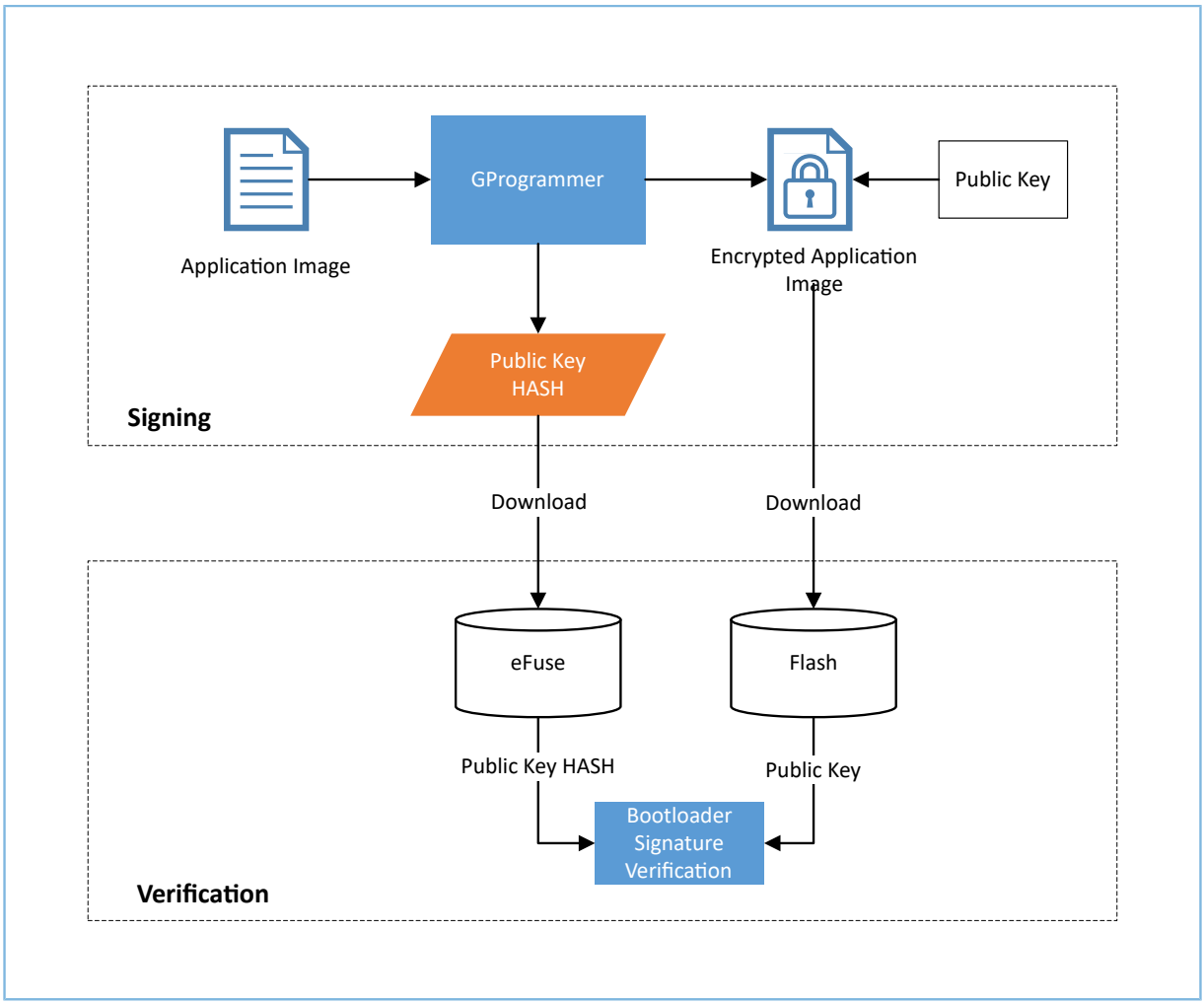

图 5-1 数字签名流程

5.1 加签流程

固件的加签流程如下图所示。

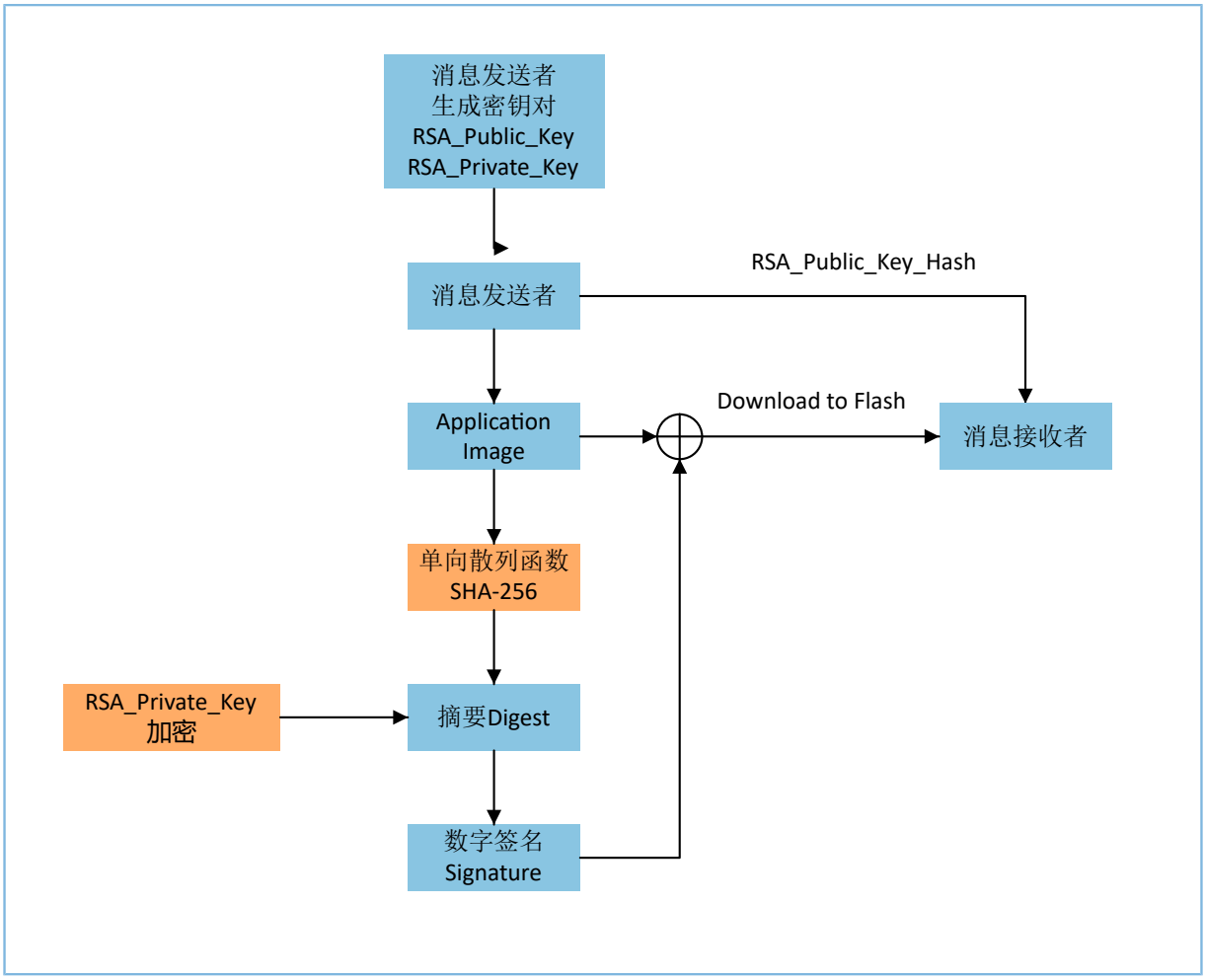

图 5-2 固件加签流程

该流程具体描述如下：

1. 消息发送者（用户）使用GProgrammer工具生成公私钥对（RSA_Public_Key、RSA_Private_Key），用于添加签名、验证签名。消息发送者使用私钥RSA_Private_Key生成加签信息。消息接收者（芯片）用公钥RSA_Public_Key验证签名。

2. RSA_Public_Key被存储于Application Image布局中并传递给芯片，eFuse中存储着公钥的HASH值。基于RSA_Public_Key计算生成的HASH值与eFuse中存储的RSA_PUBLIC_KEY_HASH须一致。

3. 使用单向散列函数生成固件的摘要，再使用私钥RSA_Private_Key加密该摘要，以生成数字签名。

5.2 验签流程

固件的验签流程如下图所示。

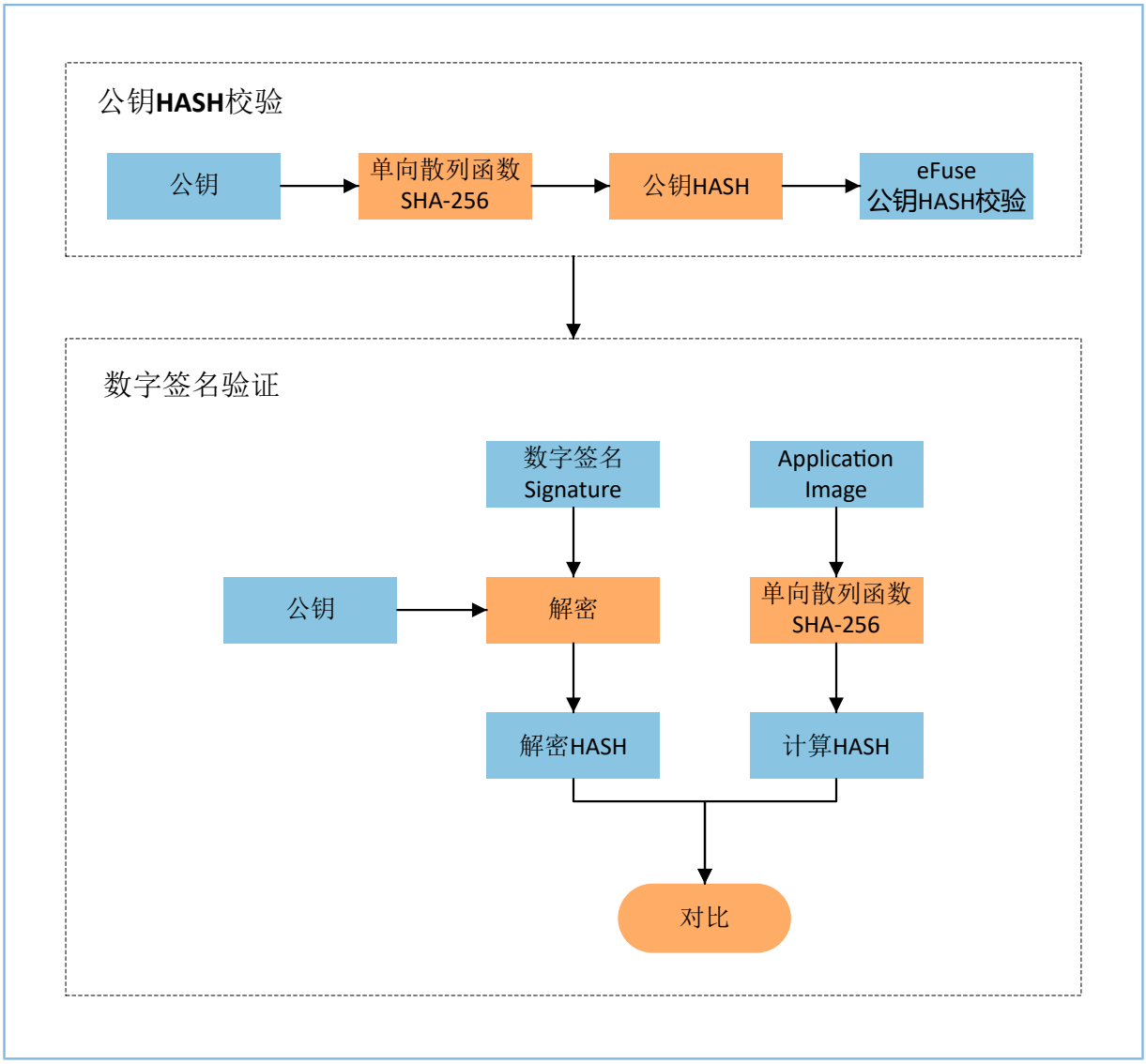

图 5-3 固件验签流程

该流程具体描述如下：

1. 公钥校验：用户通过GProgrammer将RSA_Public_Key传递给Application Image。下载Application Image到Flash后，RSA_Public_Key将与eFuse中的RSA_PUBLIC_KEY_HASH进行校验。eFuse具有一次性烧写特性，因此在更新、升级等过程中，Application Image中的RSA_Public_Key生成的HASH值与eFuse中存储的RSA_PUBLIC_KEY_HASH必须保持一致，否则校验将无法通过。

2. 验证签名：芯片使用RSA_Public_Key对固件的签名进行解密，以获得解密后的HASH值，并对Application Image进行单向散列函数计算得到HASH值。对比解密后的HASH值和单向散列函数计算得到的HASH值，如果结果一致则验签通过。

6 安全模式应用

在安全模式下，开发者需要利用eFuse来存储产品配置信息、安全模式控制信息以及用于加密加签的各种密钥信息等。开发者需要注意eFuse的一次性编程特性，从而避免一些不必要的损失。

安全模式一般用于产品量产阶段，在开发阶段建议使用非安全模式。

本章主要介绍如何使用配套工具GProgrammer和GRPLT Lite配置工具完成安全模式的应用。

6.1 使用GProgrammer烧录加密固件

开发者可以点击GProgrammer工具左侧工具栏中的“Encrypt & Sign”图标进入芯片加密操作界面，如图6-1所示。

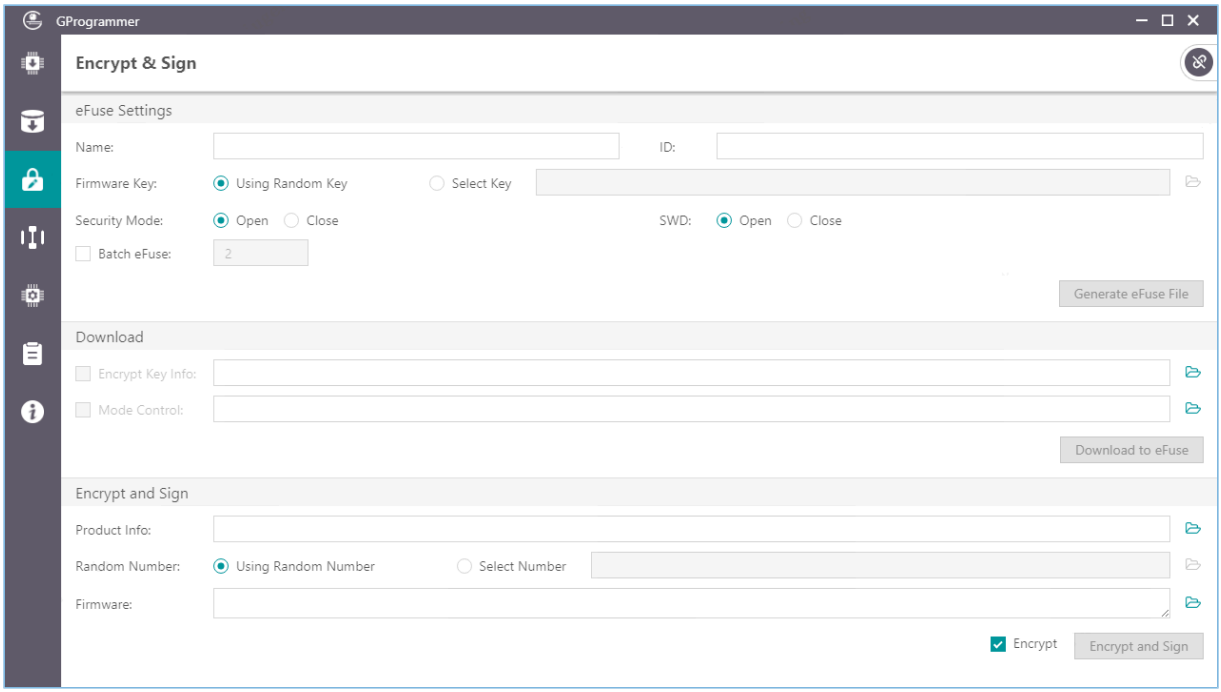

图 6-1 Encrypt & Sign界面

使用GProgrammer设置和下载eFuse，以及固件加密加签的详细操作说明，请参考《GProgrammer用户手册》。

6.2 使用GRPLT Lite配置工具自定义加密

用户可运行GRPLT Lite配置工具，进入“可选功能配置 > 加密算法配置”面板，勾选“用户自定义加密方式”后，进行自定义固件加密。

图 6-2 自定义固件加密面板

GRPLT Lite配置工具以及使用工具进行自定义固件加密的详细操作说明，请参考《GRPLT Lite配置工具用户手册》和《GRPLT Lite配置工具自定义固件加密及应用介绍》。

6.3 硬件SWD接口

GR5xx具有SWD接口。通过禁用该接口，可防止从外部入侵芯片。

在安全模式应用中，用户一旦将配置了禁用SWD接口的安全模式控制文件下载到eFuse中，SWD接口将被禁用。用户可以在GProgrammer工具生成eFuse下载文件（*Mode_control.bin*）时选择关闭SWD接口。当SWD接口关闭后，开发者依然可以通过DFU对固件进行升级。开发者也可以通过应用程序控制对应的寄存器，从而使能SWD接口。

说明:

因为eFuse的特性，烧录禁用SWD接口的安全模式控制文件的行为将是不可恢复的。

6.3.1 寄存器

表 6-1 SWD控制寄存器

位域	字段名	RW	复位值	说明
GR551x芯片系列 寄存器地址：0xA000C504				
17	SWD_ENABLE	RW	0x0	启用SWD调试 值： • 0x0：禁用 • 0x1：启用

位域	字段名	RW	复位值	说明
其他芯片系列 寄存器地址：0x4000A004				
8	SWD_ENABLE	RW	0x0	启用SWD调试 值： • 0x0：禁用 • 0x1：启用

6.3.2 函数接口

表 6-2 sys_swd_enable接口

函数原型	void sys_swd_enable(void)
功能说明	启用SWD接口
输入参数	无
返回值	无
备注	

表 6-3 sys_swd_disable接口

函数原型	void sys_swd_disable(void)
功能说明	禁用SWD接口
输入参数	无
返回值	无
备注	

7 常见问题解答

本章描述在使用安全功能时，可能出现的问题、原因及处理方法。

7.1 固件无法执行

- 问题描述

  使用Keil工具编译生成的.bin或者.hex文件，无法在已加密芯片上执行。

- 问题分析

  Keil工具编译生成的.bin或者.hex文件为非加密固件。

- 处理方法

  将编译生成的.bin或者.hex文件通过GProgrammer工具生成带后缀名为_encryptedandsign（加密加签）或者_sign（加签）的固件，然后通过GProgrammer或DFU方式烧录到Flash中。

说明:

GProgrammer工具加密固件使用的*product.json*文件必须与加密芯片上eFuse存储的信息相匹配。

7.2 固件加密加签失败

- 问题描述

  使用GProgrammer工具进行加密加签时，无法导入.hex格式文件或者弹出“Encrypt and sign “xxx.hex” failed”。

- 问题分析

  V 1.2.25之前版本的GProgrammer工具不支持hex格式文件导入。

  待加密的固件为经过*after_build.bat*脚本处理后的固件。

- 处理方法

  获取并安装1.2.25或更高版本的GProgrammer工具。

  停止使用并关闭*after_build.bat*脚本，使用当前工程重新生成的.hex或者.bin文件。

7.3 如何管理密钥信息和加密固件

- 问题描述

  将密钥文件和加密固件提供给产线，是否会有安全隐患？

- 问题分析

  密钥文件和加密固件都是在量产阶段烧录，保护文件不被泄露具有重要意义。

- 处理方法

  客户需要严格把控产线的安全性，防止GProgrammer生成的密钥文件泄露。