GR55xx固件加密及应用介绍

版本： **1.8**

发布日期： **2021-06-24**

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2层、13层

电话：+86-755-33338828　　传真：+86-755-33338099

网址：www.goodix.com

前言

编写目的

本文档介绍了GR55xx的安全模块所采用的加密和签名技术、数字签名的流程等，旨在帮助开发者了解并应用GR55xx芯片的安全模式。

读者对象

本文适用于以下读者：

- GR55xx用户
- GR55xx开发人员
- GR55xx测试人员
- 开发爱好者
- 文档工程师

版本说明

本文档为第6次发布，对应的产品系列为GR55xx。

修订记录

版本	日期	修订内容
1.0	2019-12-08	首次发布
1.3	2020-03-16	更新文档页脚版本时间
1.5	2020-05-30	更新“数字签名技术”章节中图片的格式
1.6	2020-06-30	更新“使用GProgrammer加密加签固件”
1.7	2021-02-26	更新“使用GProgrammer加密加签固件”
1.8	2021-06-24	更新GR551x为GR55xx。

目录

1 简介

GR55xx芯片安全加密模块能够将Flash存储器中的用户数据以及用户应用固件进行加密，以防止数据和应用固件被窃取，为用户产品提供安全保护。

1.1 安全基础概念

本章介绍了加密和签名的基础概念，以帮助用户更好地理解GR55xx芯片所采用的加解密和数字签名技术。

1.1.1 加密

- 对称密钥加密：又称私钥加密，即信息的发送方和接收方用一个密钥去加密和解密数据。
- 非对称密钥加密：又称公钥密钥加密，它使用了一对密钥：公钥（Public Key）和私钥（Private Key）。非对称加密和解密分别使用密钥对中的一个密钥。例如，如果加密时使用公钥（Public Key），那么解密时将使用另一个私钥（Private Key）。
- 消息完整性：是指消息没有被篡改，其检验通常使用单向散列表实现。
- 消息认证码：提供了一种认证机制，其不仅可检验消息是否被篡改，还可确认消息是否来自期望的通信对象。
- 混合密码系统：采用了对称密码与公钥密码相结合的密码方式。利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。

1.1.2 数字签名

数字签名是一种能够保证消息完整性，提供认证并且防止否认的密码技术。只有信息的发送者才能产生的一段数字串。该段数字串可作为信息的发送者发送信息真实性的有效凭证，他人无法伪造。

2 加解密技术

本章将介绍GR55xx芯片中的加解密模块、固件加解密所使用的算法以及数据加密解密流程。

2.1 加解密模块

GR55xx安全加密模块主要包含TRNG、PRESENT-128、eFuse、KEYRAM、PKC、HMAC、XIP_DEC等硬件模块。

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

  TRNG模块用于生成加密解密时使用的随机数。为了保证随机数质量并修正TRNG的偏差，TRNG模块中增加了LFSR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是一个IP核，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。PRESENT-128加解密模块的功能框图如图 2-1所示。

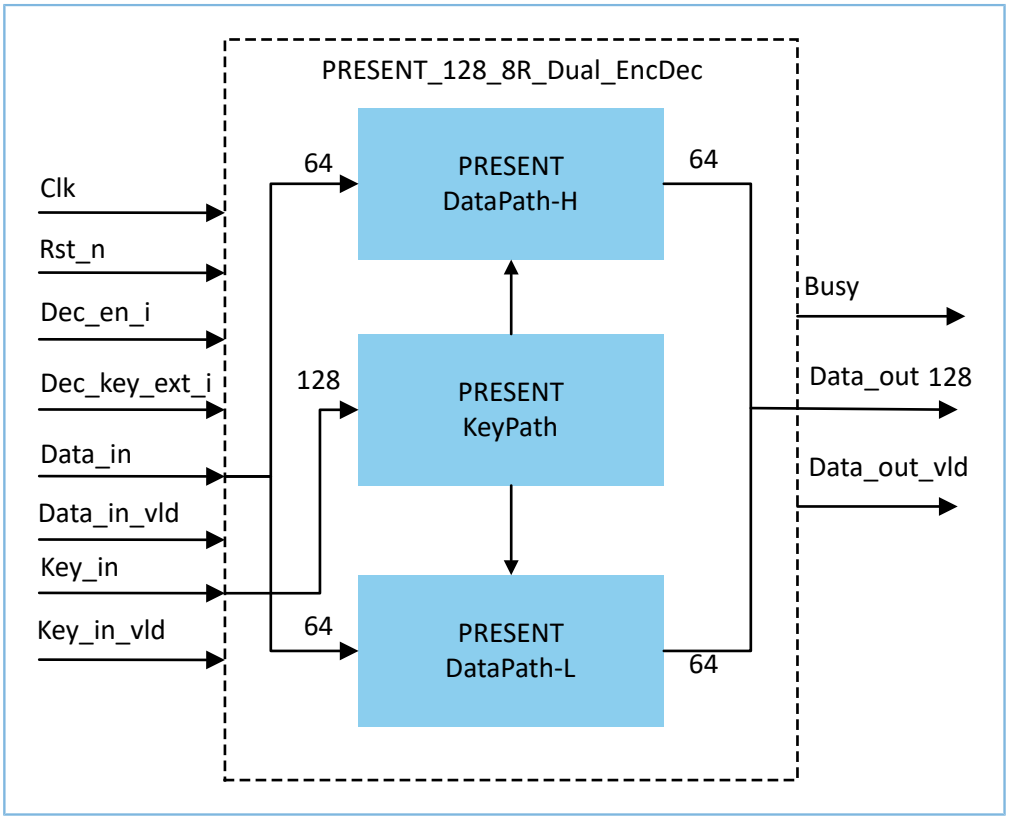

图 2-1 PRESENT-128加解密模块

- **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。其容量为512 Bytes，用于存储安全密钥和芯片校准数据等。

- **KEYRAM模块**

  KEYRAM主要用于芯片上电后的密钥推导。在安全启动流程中，每次冷启动生成真随机数作为掩码（MASK），用于防止外界穷举法破解。通过keyPort总线接口自动实现加密读取key到AES、HMAC、XIP_DEC等模块，硬件实现自动密钥导入。

- **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）点乘法。

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

  通过使用完全符合联邦信息处理标准（FIPS）出版物198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，哈希消息认证码（HMAC）协处理器能对数据进行认证。该模块支持多种算法（SHA256，HMAC-SHA256），密钥大小为256位。

- **XIP_DEC模块**

  在XIP模式下，实时解密时读取固件指令或数据需要调用XIP_DEC模块。XIP_DEC模块内嵌硬件PRESENT-128子模块实现加密固件的自动解密。XIP_DEC模块的功能框图如图 2-2所示。

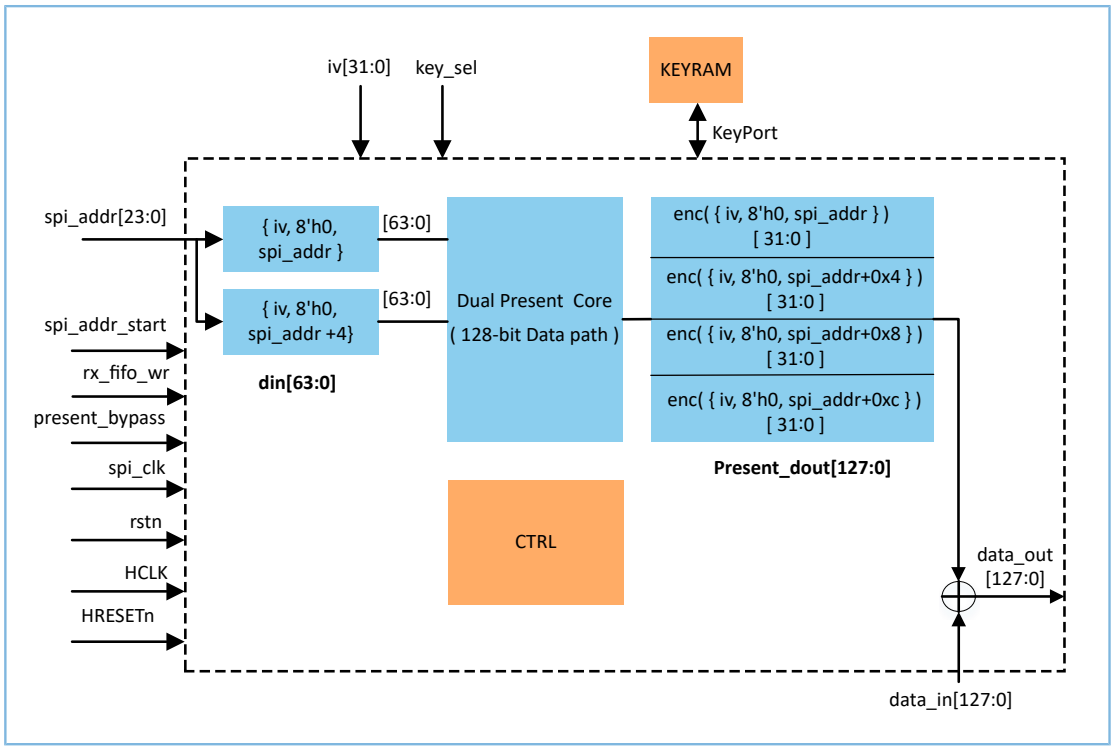

图 2-2 XIP_DEC模块

说明:

详细的模块介绍请参考对应GR55xx系列的Datasheet。

2.2 固件加解密流程

GR55xx混合使用了对称密码（PRESENT-128）和公钥密码（椭圆曲线密码算法ECC）两种算法来保证固件的安全性和高效性。

在加密过程中，GR55xx使用PRESENT-128对称密码加密明文，再使用ECC为对称加密中所使用的密钥进行加密。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。

2.2.1 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。

PRESENT-128模块的功能框图如图 2-3所示。

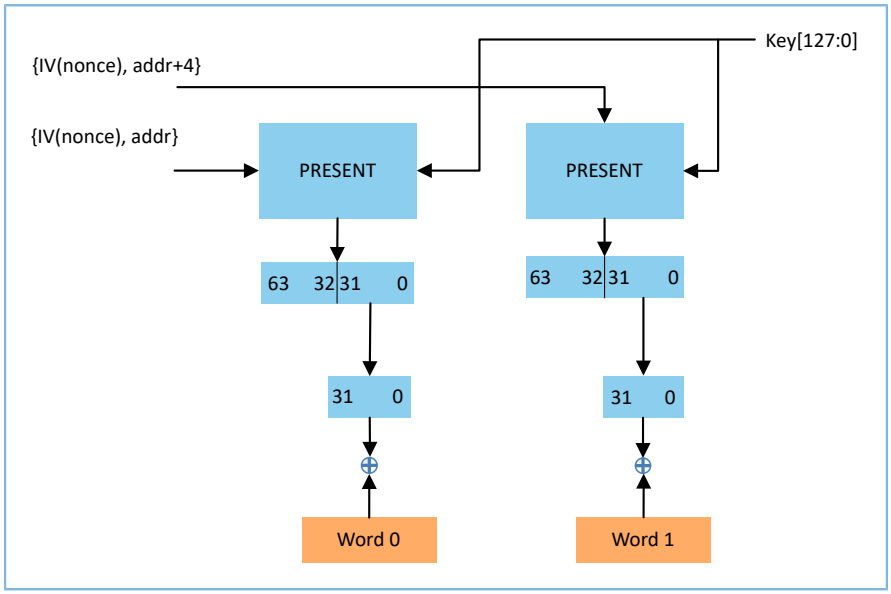

图 2-3 PRESENT-128模块功能框图

2.2.2 椭圆曲线密码（ECC）算法

ECC与传统的基于大质数因子分解困难性的加密方法不同，ECC通过椭圆曲线方程式的性质产生密钥。

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、存储空间更小：ECC的密钥大小和系统参数比RSA、DSA更小。

2.2.3 混合密码系统

在具备同等机密性的密钥长度的情况下，公钥密码的处理速度只有对称密码的几百分之一。因此，公钥密码无法适用于加密较长的消息内容。

下面以加密过程为例简单介绍混合密码系统的加解密流程，如图 2-4所示。

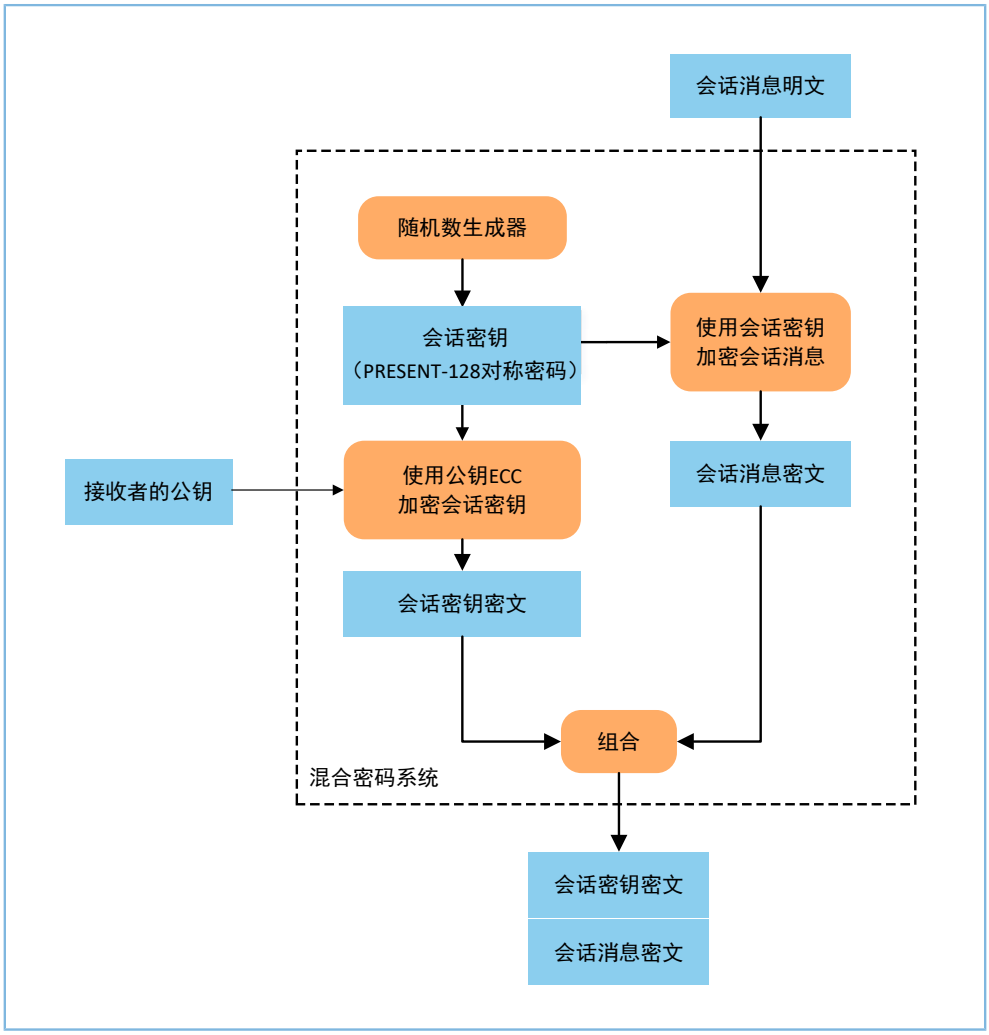

图 2-4 混合加密系统加密流程

该流程具体描述如下：

1. 生成会话密钥：通过随机数生成器生成对称密码加密中使用的会话密钥（PRESENT-128对称密码）。
2. 会话密钥加密消息：使用会话密钥加密会话消息，生成会话消息密文。
3. 公钥加密会话密钥：从混合密码系统外部获得公钥ECC对会话密钥加密，得到会话密钥密文。
4. 组合密文：组合会话密钥密文和会话消息密文。

2.3 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，GR55xx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。

数据加密解密流程如图 2-5所示。

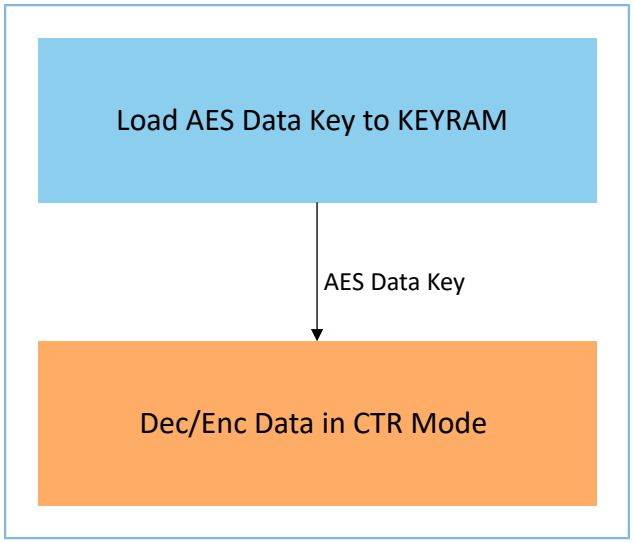

图 2-5 数据加密解密流程

该流程具体描述如下：

1. AES算法所用的对称密钥存储于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的加密配合使得对称密钥难以被外部获取。
2. AES的算法的加密、解密流程基本一致。在AES加密解密流程中，采用CTR分组模式。

说明:

GR55xx采用的固件密钥（非对称密钥）和数据密钥（对称密钥）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。

当使用Flash中加密的数据时，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GR55xx系列的开发者指南。

---

3 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。

GR55xx安全模式下的数字签名过程（见图 3-1）包括：

- 加签：用户可通过GProgrammer工具将签名所需要的有关信息下载到eFuse和Flash中，以实现对固件的数字签名。
- 验签：GR55xx 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。

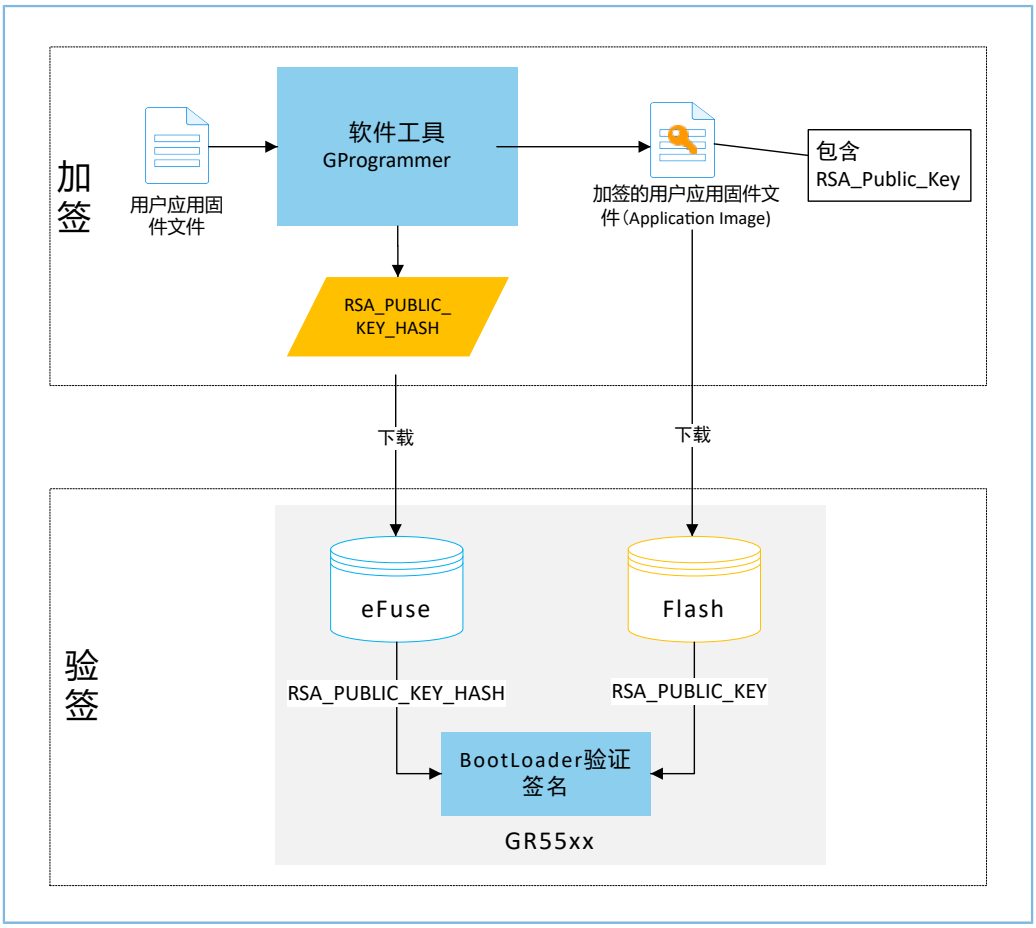

图 3-1 GR55xx数字签名过程

3.1 加签流程

固件文件的加签流程如图 3-2所示。

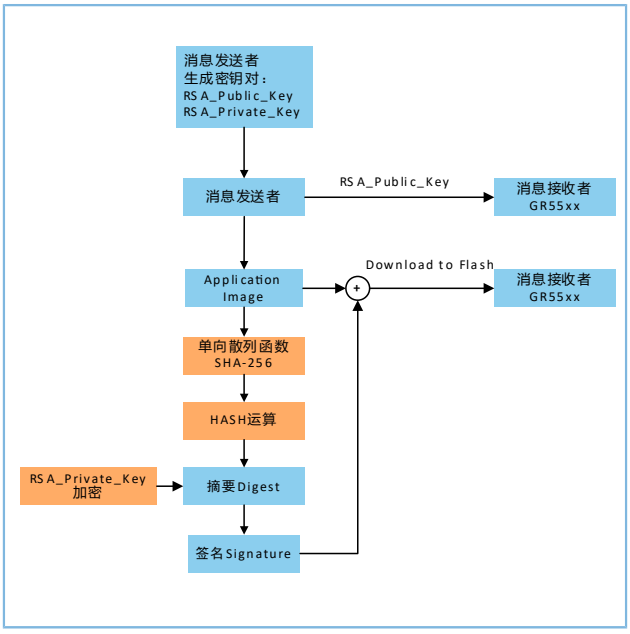

图 3-2 加签流程

该流程具体描述如下：

1. 消息发送者（用户）使用GProgrammer工具生成密钥对（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（GR55xx）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GR55xx，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. 使用单向散列函数计算出消息的散列值，然后对散列值进行HASH运算生成摘要，再使用私钥RSA_Private_Key加密摘要，以生成签名。

3.2 验签流程

验证签名流程如图 3-3所示。

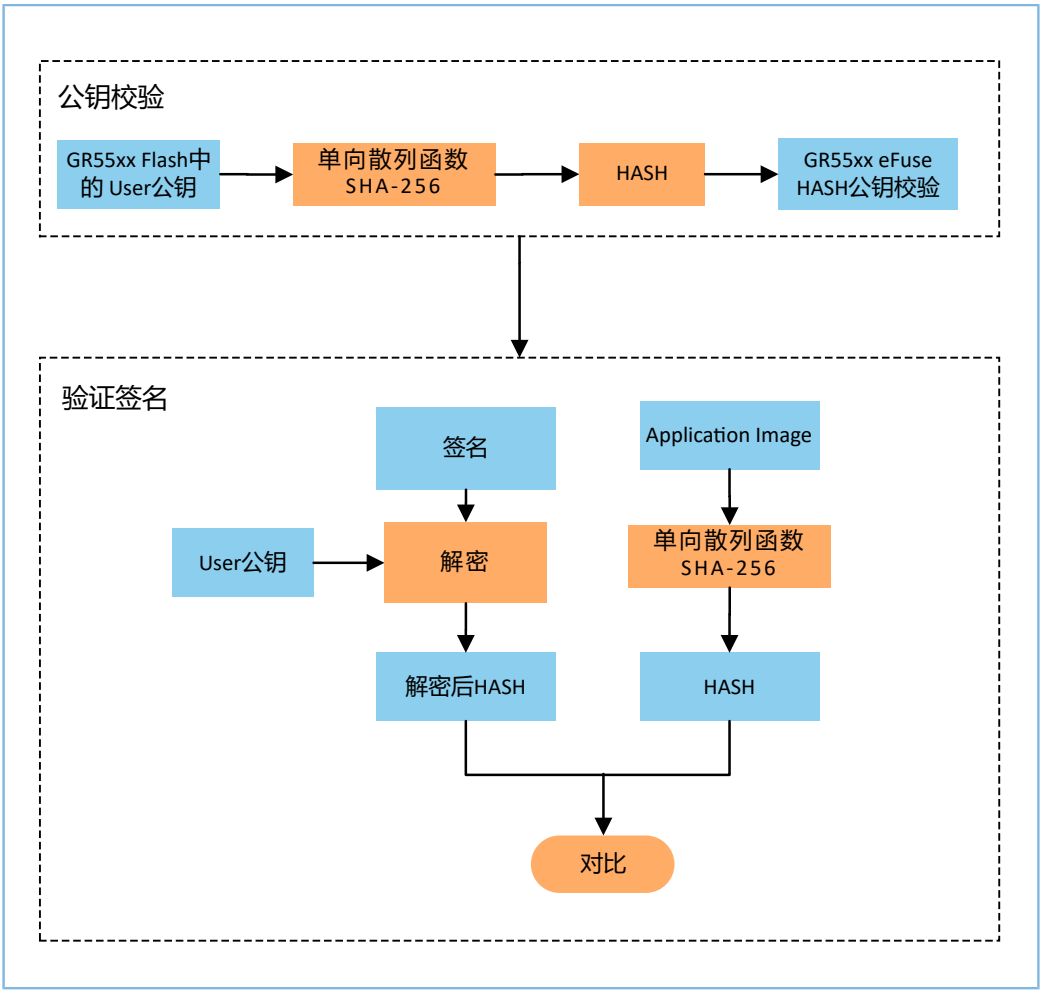

图 3-3 验签流程

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。Application Image下载到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。

   GR55xx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，如果对比结果一致则验签通过。

4 GR55xx安全模式应用

GR55xx是一款支持多种主流密码算法的系统级低功耗蓝牙芯片（SoC），可对位于存储器上的用户数据以及用户固件进行保护，防止被窃听者获取。

GR55xx的安全功能具有以下特点：

1. 安全的密钥存储。

   GR55xx采用eFuse存储器管理安全密钥，无法通过MCU直接读取eFuse存储的密钥信息。当加密模块需要使用密钥信息时，由芯片内部独立的硬件单元实现从eFuse自动导入密钥到加密模块。

2. 防止固件窃取。

   GR55xx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，但由于无法获取到私钥，依然无法使用该加密固件。

3. 防止恶意攻击。

   GR55xx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，防止攻击者操控GR55xx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，会验签加密固件信息，以防止恶意修改Flash存储的固件信息。

4. 支持一机一密。

   GR55xx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GR55xx芯片中，以实现一台设备对应唯一数据密钥的目的。

4.1 使用GProgrammer加密加签固件

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。

在GR55xx安全模式下，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。

在GProgrammer工具中，用户可通过指定产品的“Name”、“ID”、“Firmware Key”、“Security Mode”和“SWD”接口来生成存储在eFuse中的相关文件。

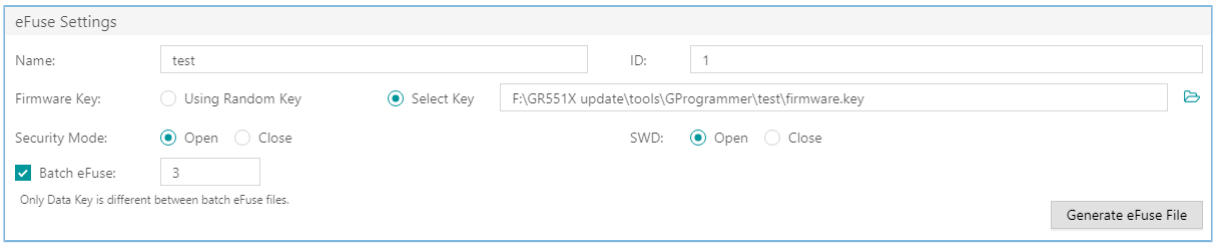

图 4-1 生成eFuse相关文件

说明:

- eFuse是GR55xx芯片内部的一个具有随机访问接口的一次性可编程存储器。
- Firmware Key可使用软件自动生成的随机Key，也可使用用户自行添加的Key文件。
- Batch eFuse可指定生成相应数量的数据密钥文件，以支持一机一密的使用场景。

生成的文件包括：

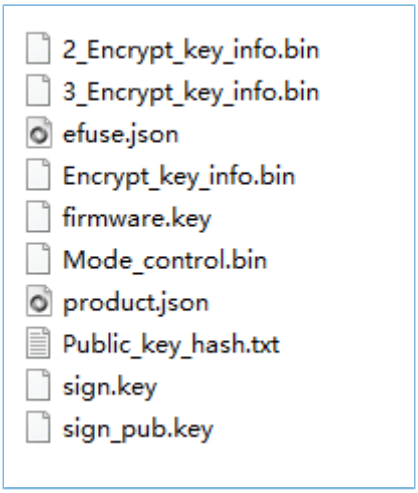

图 4-2 生成的文件

- *efuse.json*：生成的临时文件。
- *Encrypt_key_info.bin*：生成的eFuse下载文件，包含了产品以及加密加签的信息。此文件需下载至eFuse。
- *firmware.key*：用于加密固件的私钥。
- *Mode_control.bin*：生成的eFuse下载文件，包含了Security Mode和SWD信息。此文件需下载至eFuse。
- *product.json*：生成的产品信息文件。当固件加密加签时，需导入此文件。
- *sign.key*：用于生成签名的私钥。
- *sign_pub.key*：用于验证签名的公钥。
- *Public_key_hash.txt*：用于验证签名的公钥Hash。

说明:

以上文件请用户妥善保存，切忌泄露、丢失。后续的“烧写eFuse”和“固件加密加签”操作需使用这些文件。

为了方便用户下载文件到eFuse或加密加签固件，生成的*Encrypt_key_info.bin*和*Mode_control.bin*文件的路径将默认被添加到下载区域，产品信息文件*product.json*的路径被默认添加到加密加签的“Product Info”栏中。点击“Download to eFuse”按钮可将安全模式控制信息及各种密钥信息烧录至eFuse中，切记不应该向eFuse烧录2个不同的密钥信息。

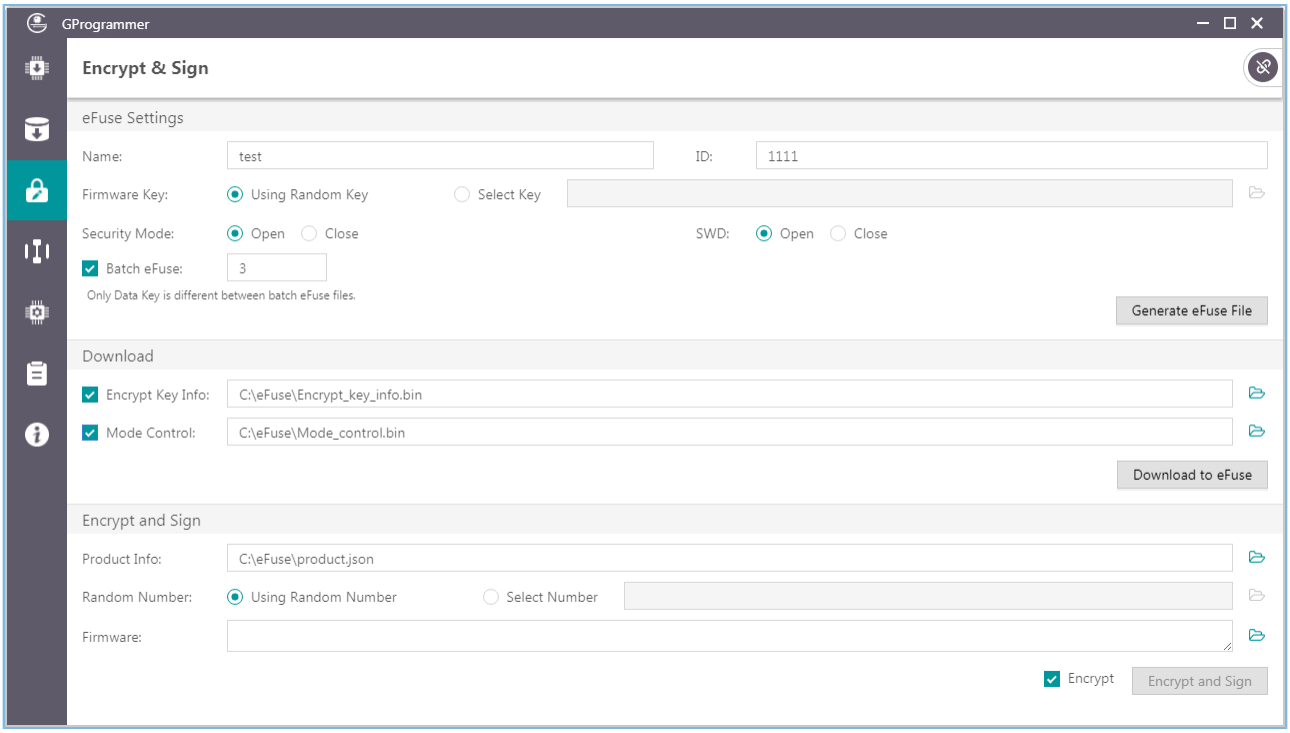

图 4-3 自动加载文件路径

在“Firmware”栏中选择需要加密加签或者仅加签的非加密固件，如果勾选“Encrypt”复选框，则操作按钮变成“Encrypt and Sign”；如果不勾选“Encrypt”复选框，则操作按钮变成“Sign”。点击操作按钮，选择路径后，即可生成加密加签或者仅加签的固件。

GProgrammer的详细操作说明，请参考《GProgrammer用户手册》。

说明:

不要修改eFuse中生成的任何文件，否则可能造成固件加密加签失败。

4.2 硬件SWD接口

GR55xx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接口将永远无法启用。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。

5 常见问题解答

本章描述了在使用安全功能时，可能出现的问题、原因及处理方法。

5.1 测试时，固件无法执行

- 问题描述

  使用Keil工具编译生成的.bin文件，无法在加密芯片上执行。

- 问题分析

  Keil工具编译生成的.bin文件为非加密固件。

- 处理方法

  1. 将编译生成的.bin文件通过GProgrammer工具生成带后缀名为_encrypted的加密固件。
  2. 通过GProgrammer或DFU方式升级到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。

5.2 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生成的密钥文件和加密固件泄露。