WS63V100 安全软件

开发指南

文档版本　02

发布日期　2024-06-27

前言

概述

本文档主要描述了 CIPHER DRIVER 的结构及其软件接口的使用方法。

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
WS63	V100

读者对象

本文档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
危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危

符号	说明
	害。
⚠ 警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
⚠ 注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
02	2024-06-27	更新“1.2 使用流程”章节内容。 更新“2.1.4 错误类型”章节内容。
01	2024-04-10	第一次正式版本发布。
00B01	2024-03-29	第一次临时版本发布。

目 录

1 概述

1.1 功能描述

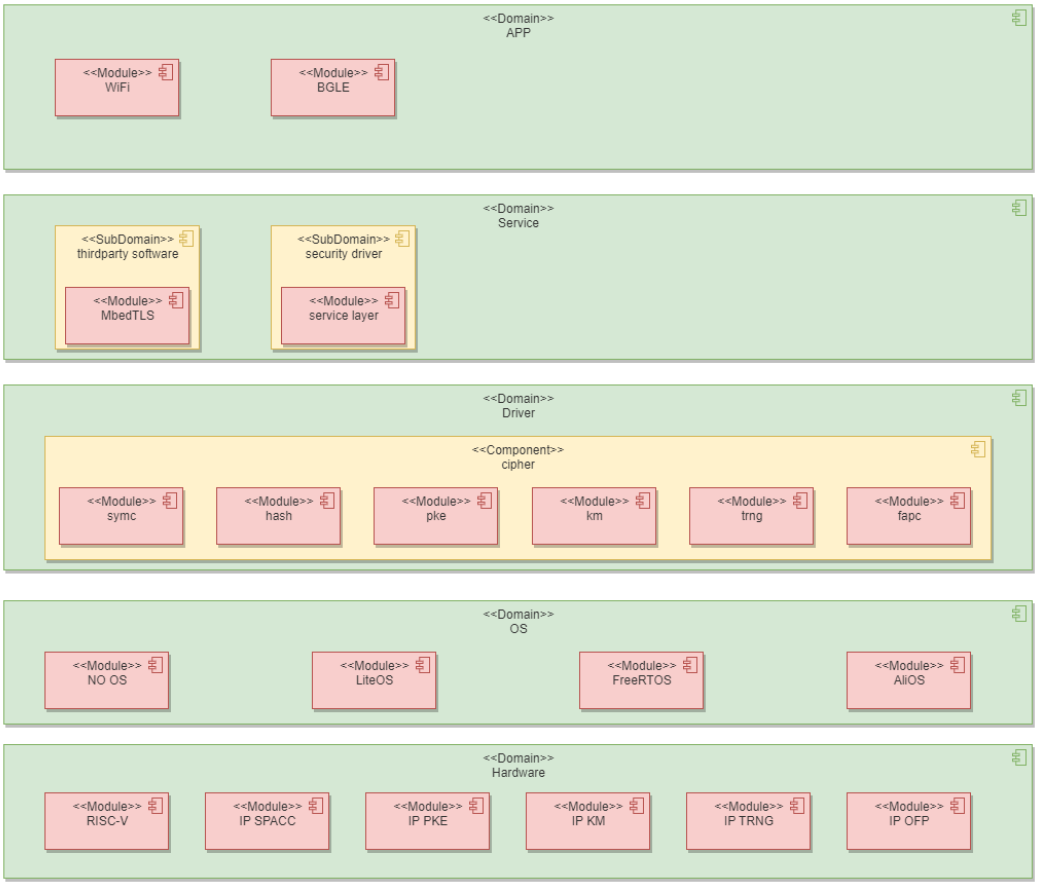
CIPHER DRIVER 为安全算法模块，对外提供 mbedtls API 和 service layer 两类接口，如图 1-1 所示。

- mbedtls API
  - 安全驱动对接开源第三方 mbedtls 接口，可以通过调用 mbedtls API 使用硬件安全能力。
  - 对于硬件支持的规格，均完成 mbedtls 对接，硬化范围及 API 使用具体参考《WS63V100 TLS & DTLS 开发指南》。
- service layer
  - 自研安全驱动接口，为上层业务提供直接使用硬件安全的能力，支持 LiteOS/FreeRTOS/AliOS 等 OS 环境，及 flashboot/bootrom 等无 OS 环境。
- CIPHER DRIVER 提供如下算法：
  - AES、SM4 等对称加解密算法；
  - HASH、SM3 及 HASH-MAC、SM3-MAC 摘要算法；
  - RSA、ECC、SM2 签名验签、密钥协商、加解密，大数运算等非对称算法；
  - 硬件密钥管理；
  - 随机数算法；
  - FLASH 在线解密；

- 主要用于 AIOT 产品安全启动、安全存储、安全通讯等场景。

- CIPHER DRIVER 包括 5 个子模块：SPACC（symc、hash）、PKE、KM、TRNG、FAPC。

图1-1 CIPHER DRIVER 上下文关系图



1.1.1 SPACC

SPACC（Security Protocol Accelerator，安全协议加速器）模块实现了 cipher 对称加解密算法、HASH 及 HMAC 摘要算法。

1.1.1.1 对称加解密算法

- AES：支持 ECB/CBC/CFB/OFB/CTR/CCM/GCM/CMAC/CBC_MAC 共 9 种工作模式。其中：
  - CFB 模式支持的加密数据位宽可为 8/128bit。
  - OFB 模式支持的加密数据位宽可为 128bit。
  - CCM/GCM 模式下，加解密结束后需获取一次 TAG 值。

- ECB、CBC、CBC_MAC 模式加密数据长度必须为 16Byte 对齐，CMAC 允许最后一块加密数据长度非 16Byte 对齐。
- 支持 AES-128、AES-192、AES-256（128、192、256 均指传入的密钥长度，单位为 bit)。

- SM4：支持 ECB/CBC/CTR/CFB/OFB 共 5 种工作模式。支持 SM4-128；除 CTR 模式外，加密数据长度须为 16Byte 对齐。

cipher 驱动支持软件多通道，各芯片可以根据自己需要在 porting 文件中配置；支持低功耗模式，默认开启低功耗；支持轮询和 wait_event 模式，若注册了 wait func 则走 wait_event 模式，否则走轮询模式。

说明

ECB 模式属于非安全算法，不建议使用。

1.1.1.2 摘要算法

- HASH：支持 SHA1/SHA224/SHA256/SHA384/SHA512/SM3。
- HMAC：支持 HMAC-SHA1/HMAC-SHA224/HMAC-SHA256/HMAC-SHA384/HMAC-SHA512/HMAC-SM3。

HASH 和 HMAC 支持中途传入非 block size 对齐的数据块；支持软件多通道，各芯片可以根据自己需要在 porting 文件中配置；支持低功耗模式，默认开启低功耗；支持轮询和 wait_event 模式，若注册了 wait func 则走 wait_event 模式，否则走轮询模式。HASH 支持重试机制，即中间过程调用接口出错可重新调用该接口直至成功，而不需要从头开始整个计算流程，具体内容请参见《WS63V100 SECURITY DRIVER API manual》中 uapi_drv_cipher_hash_start、uapi_drv_cipher_hash_update 和 uapi_drv_cipher_hash_finish 接口使用说明。

说明

SHA1/SHA224 属于非安全算法，不建议使用。

1.1.2 PKE

PKE（Public Key Encryption，公钥加密)，即非对称加解密算法。常用于文件的签名和验签，以及对对称密钥的加解密，主要包括以下算法：

- RSA
  - 支持 PKCS#1 v1.5 和 PKCS#1 v2.1PSS 两种模式的私钥签名和公钥验签。
  - 支持 PKCS#1 v1.5 和 PKCS#1 v2.1OAEP 两种模式的公钥加密和私钥解密。
  - 支持密钥位宽 1024/2048/3072/4096。

- ECC
  - 支持 RFC 5639-Brainpool P256/384/512r1、NIST FIPS 186-4 P192/224/256/384/521 和 RFC 8032-ED25519（即 EDDSA）的私钥签名和公钥验签。
  - 支持密钥位宽 192/224/256/384/512(521)。
- SM2
  - 支持 SM2 公钥加密和私钥解密。
  - 支持 SM2 私钥签名和公钥验签。
- 密钥协商
  - 支持 RFC 5639-Brainpool P256/384/512r1、NIST FIPS 186-4 P192/224/256/384/521、RFC 7748-Curve25519/Curve448、 RFC 8032 - ED25519（即 EDDSA）和 SM2 的公私钥对生成，及由给定的私钥生成其对应的公钥。
  - 支持 RFC 5639-Brainpool P256/384/512r1、NIST FIPS 186-4 P192/224/256/384/521、RFC 7748-Curve25519/Curve448 和 SM2 的密钥交换（ECDH）。
  - 支持 192/224/256/384/512/521/1024/2048/3072/4096 比特 DH 密钥协商。
- 曲线上点的验证
  - 支持 RFC 5639-Brainpool P256/384/512r1、NIST FIPS 186-4 P192/224/256/384/521 和 SM2 的点是否在曲线上的验证。
- 大数运算
  - 支持小于等于 4096 位的模加、模减、模乘、模逆、模幂和模运算。
  - 支持小于等于 2048 位的常规大数乘运算。

PKE 支持低功耗模式，并默认开启低功耗。支持轮询和 wait_event 模式，若注册了 wait func 则走 wait_event 模式，否则走轮询模式。

说明

NIST FIPS 186-4 P192/224/256/384/521，PKCS#1 v1.5，RSA1024/2048，低于 256 比特的 ECDH 及低于 2048bit 的 DH 为不安全算法，不建议使用。

1.1.3 KM

KM 是密钥管理模块，增加密钥安全强度，用于将 key 加载到 keyslot 中，供 SPACC 和 Flash 在线解密使用。支持硬件 key 和软件 key；支持 8 个 cipher keyslot 通道和 2 个 hmac keyslot 通道。

1.1.4 TRNG

随机数获取模块，获取硬件产生的真随机数。

1.1.5 FAPC

通过 fapc 控制器，配置 flash 在线解密操作相关的参数。fapc 支持配置 4 个 region，支持 region 使用相同的 IV 值，flash 在线解密使用 AES-128-CTR 算法。

1.2 使用流程

系统初始化时会主动调用 uapi_drv_cipher_env_init 完成 CIPHER DRIVER 基础的初始化环境配置，用户不需要再单独调用。

1.2.1 密钥配置

1.2.1.1 场景说明

生成和配置 SPACC、在线解密模块所需要的软件 key 和硬件 Key，软件 key 可由 PBKDF2 和 HKDF 算法生成。keyslot 存放最终的软件或硬件 key，供 SPACC 和在线解密使用。支持 AES 128/192/256bits 加解密，支持 HMAC。

1.2.1.2 软件密钥配置工作流程

步骤 1 调用 uapi_drv_km_init 初始化 KM 模块。

步骤 2 创建 keyslot 通道，FLASH 在线解密不需要此操作，调用 uapi_drv_keyslot_create 接口完成。（若配置的 key 供 SPACC 模块调用，则该步骤在调用者调用 SPACC 模块接口绑定 keyslot 通道前完成）。

步骤 3 创建一路 klad，并获取 klad 句柄。调用 uapi_drv_klad_create 接口完成。

步骤 4 绑定 keyslot 通道与 klad 句柄，调用 uapi_drv_klad_attach 接口完成。步骤 2、步骤 4 的顺序可以调换。

步骤 5 配置 klad 属性。包含：

- 密钥对应的 root_key 类型。
- 密码可用于的算法。
- 密钥可用于加密还是解密操作。
- 密钥是安全还是非安全密钥。

- 密钥是否仅可被配置该密钥的 CPU 使用。
- 源和目的 buffer 类型是安全还是非安全（详见下方说明）。

调用 uapi_drv_klad_set_attr 接口完成。

步骤 6 配置软件 key，调用 uapi_drv_klad_set_clear_key 接口完成。SPACC

步骤 7 解绑 keyslot 与 klad 句柄，对于 FLASH 在线解密操作则是指定对应操作类型，调用 uapi_drv_klad_detach 接口完成。

步骤 8 销毁 klad 句柄，调用 uapi_drv_klad_destroy 接口完成。

步骤 9 销毁 keyslot 句柄，调用 uapi_drv_keyslot_destroy 接口完成。

步骤 10 调用 uapi_drv_km_deinit 去初始化 KM 模块。

说明

klad 配置的源和目的 buffer 类型与 SPACC 模块 buffer 类型相对应，如果此处配置源数据 buffer 为仅支持安全 buffer，而 SPACC 模块源 buffer 配置为非安全 buffer，则计算过程中会报错。步骤 9、步骤 10 的操作需要在 SPACC、HMAC、FLASH 在线解密使用完 keyslot 之后再执行，不然后导致计算报错。

----结束

1.2.1.3 硬件密钥配置工作流程

步骤 1 调用 uapi_drv_km_init 初始化 KM 模块。

步骤 2 创建 keyslot 通道，FLASH 在线解密不需要此操作，调用 uapi_drv_keyslot_create 接口完成。（若配置的 key 供 SPACC 模块调用，则该步骤在调用者调用 SPACC 模块接口绑定 keyslot 通道前完成）。

步骤 3 创建一路 klad，并获取 klad 句柄。调用 uapi_drv_klad_create 接口完成。

步骤 4 绑定 keyslot 通道与 klad 句柄，对于在 FLASH 线解密操作则是指定对应操作类型，调用 uapi_drv_klad_attach 接口完成。步骤 2、步骤 4 的顺序可以调换。

步骤 5 配置 klad 属性。包含：

- 密钥对应的 root_key 类型。
- 密码可用于的算法。
- 密钥可用于加密还是解密操作。
- 密钥是安全还是非安全密钥。

- 密钥是否仅可被配置该密钥的 CPU 使用。
- 源和目的 buffer 类型是安全还是非安全（详见下方说明）。

调用 uapi_drv_klad_set_attr 接口完成。

步骤 6 配置硬件 key，调用 uapi_drv_klad_set_effective_key 接口完成。

步骤 7 解绑 keyslot 与 klad 句柄，对于 FLASH 在线解密操作则是指定对应操作类型，调用 uapi_drv_klad_detach 接口完成。

步骤 8 销毁 klad 句柄，调用 uapi_drv_klad_destroy 接口完成。

步骤 9 销毁 keyslot 句柄，调用 uapi_drv_keyslot_destroy 接口完成。

步骤 10 调用 uapi_drv_km_deinit 去初始化 KM 模块。

说明

- klad 配置的源和目的 buffer 类型与 SPACC 模块 buffer 类型相对应，如果此处配置源数据 buffer 为仅支持安全 buffer，而 SPACC 模块源 buffer 配置为非安全 buffer，则计算过程中会报错。步骤 9、步骤 10 的操作需要在 SPACC、HMAC、FLASH 在线解密使用完 keyslot 之后再执行，不然后导致计算报错。
- 硬件 key 配置需要烧写对应 efuse 才可在 keyslot 通道中获得其对应的正确的工作 key，否则此 key 用于计算得到的结果是错误的，具体 efuse 烧写过程请参见“1.2.16 安全相关 efuse 烧写建议”小节。

----结束

1.2.1.4 注意事项

硬件密钥配置依赖对应 EFUSE，具体 efuse 烧写方式详见“1.2.16 安全相关 efuse 烧写建议”小节。

1.2.2 对称加解密

1.2.2.1 场景说明

对数据进行加密或解密，支持以下场景数据的加解密操作：

- 源数据所在位置为 DDR、RAM、flash。
- 目的数据所在位置为 DDR、RAM。

支持原地加解密操作，即输入和输出使用同一块 buffer 地址。

1.2.2.2 工作流程

步骤 1 调用 uapi_drv_cipher_symc_init 初始化 SYMC 模块

步骤 2 建一路 symc，并获取 symc 句柄。调用 uapi_drv_cipher_symc_create 接口完成。

步骤 3 创建 keyslot 通道，调用 uapi_drv_keyslot_create 接口完成。

步骤 4 绑定 keyslot 通道与 symc 句柄。调用 uapi_drv_cipher_symc_attach 接口完成。keyslot 通道创建可参考 KEYSLOT&KLAD 相关接口说明。

步骤 5 配置 symc 算法参数信息。包含：

- 密钥基本属性（长度、奇偶属性）。
- 初始向量。
- 加解密位宽（CFB/OFB 模式）。
- 密钥更新方式（每次重新设置 IV/使用上个数据包配置的 IV）。
- Additional Authenticated Data（GCM/CCM 模式）。

调用接口 uapi_drv_cipher_symc_set_config 接口完成。

步骤 6 对数据进行加解密。用户可以调用以下任一接口进行加解密。

- 加密：uapi_drv_cipher_symc_encrypt
- 解密：uapi_drv_cipher_symc_decrypt

步骤 7 如果是 CCM、GCM 模式，则需要继续调用 uapi_drv_cipher_symc_get_tag 接口获取 TAG 值，否则直接执行步骤 8。

步骤 8 解绑 keyslot 通道与 cipher 句柄，调用 uapi_drv_cipher_symc_detach 接口完成。

步骤 9 销毁 symc 句柄，调用 uapi_drv_cipher_symc_destroy 接口完成。

步骤 10 调用 uapi_drv_cipher_symc_deinit 去初始化 SYMC 模块

说明

步骤 5 配置好后，允许多包使用相同的配置信息，步骤 6 的操作可在此基础上连续进行。此时场景类似于步骤 5 配置 IV 更新方式为使用上个数据包配置的 IV（CRYPTO_SYMC_CCM_IV_DO_NOT_CHANGE/UAPI_DRV_CIPHER_SYMC_GCM_IV_DO_NOT_CHANGE/UAPI_DRV_CIPHER_SYMC_CCM_IV_DO_NOT_CHANGE）的场景。

----结束

1.2.2.3 注意事项

- 支持 AES ECB/CBC/CFB/OFB/CTR/CCM/GCM 共 7 种模式。
  - 对于 ECB/CBC/CFB/OFB/CTR 模式，支持重试，即在调用 uapi_drv_cipher_symc_encrypt/uapi_drv_cipher_symc_decrypt 接口失败时，仍可配置正确参数后继续重新调用该接口，完成后续数据加解密。
  - 对于 CCM/GCM 模式支持重试，即在调用 uapi_drv_cipher_symc_encrypt/uapi_drv_cipher_symc_decrypt 或 uapi_drv_cipher_symc_get_tag 接口失败时，仍可配置正确参数后继续重新调用该接口，完成后续数据加解密。
- 支持软件多通道。
  - 可同时进行多个数据加解密操作，即执行步骤 2 启动运算，在本次运算未完成（即未执行步骤 8）之前，可申请一个新通道，启动另一个数据加解密运算，直到申请不到通道为止。
  - 最多支持的软件通道数，可由各芯片在 porting 文件中自己配置。

1.2.3 MAC 值获取

1.2.3.1 场景说明

获取数据 mac 值，支持以下场景：源数据所在位置为 DDR、RAM、flash。

1.2.3.2 工作流程

步骤 1 调用 uapi_drv_cipher_symc_init 初始化 SYMC 模块。

步骤 2 创建一路 symc，并获取 symc 句柄，配置 mac 计算需要的参数信息。包含：

- 密钥基本属性（长度、奇偶属性）。
- 加解密位宽。
- 使用的 keyslot 通道，keyslot 通道创建可参考 KEYSLOT&KLAD 相关接口说明。

调用 uapi_drv_cipher_mac_start 接口完成。

步骤 3 对数据进行加密，调用 uapi_drv_cipher_mac_update 接口完成（可调用一次或多次）。

步骤 4 获取 MAC 值，并销毁句柄，调用 uapi_drv_cipher_mac_finish 接口完成。

步骤 5 调用 uapi_drv_cipher_symc_deinit 接口，去初始化 SYMC 模块。

----结束

1.2.3.3 注意事项

- 支持 AES CBC_MAC 和 AES CMAC 两种模式。
- 支持软件多通道。
  - 可同时进行多个数据加解密操作。即执行步骤 2 启动运算，在本次运算未完成（即未执行步骤 4）之前，可申请一个新通道，启动另一个数据加解密运算，直到申请不到通道为止。
  - 最多支持的软件通道数，可由各芯片在 porting 文件中自己配置。

1.2.4 HASH 计算

1.2.4.1 场景说明

获取消息摘要，支持以下场景：源数据所在位置为 DDR、RAM、flash。

1.2.4.2 工作流程

步骤 1 调用 uapi_drv_cipher_hash_init 初始化 HASH 模块。

步骤 2 创建一路 hash，配置当前 hash 计算类型，并获取 hash 句柄。调用 uapi_drv_cipher_hash_start 接口完成。

步骤 3 传入消息数据，对消息进行摘要计算，调用 uapi_drv_cipher_hash_update 接口完成。

步骤 4 获取消息摘要，调用 uapi_drv_cipher_hash_finish 接口完成。

步骤 5 调用 uapi_drv_cipher_hash_deinit 去初始化 HASH 模块。

说明

步骤 4 如果成功会自己销毁 hash 句柄，不需要额外调用 hash 接口。步骤 2 启动成功后，无论后续步骤成功或失败，均可调用 uapi_drv_cipher_hash_destroy 接口销毁 hash 句柄（步骤 4 成功除外）。

----结束

1.2.4.3 注意事项

- 在调用 uapi_drv_cipher_hash_finish 接口获取消息摘要时，传入的 result buffer 需要调用者申请，其大小为 result_len，且 result_len 不应小于当前 hash 类型对应的摘要计算结果长度。
- 支持多段数据连续 update，结果与多段数据一次性 update 结果一致。

- 支持 hash clone 操作，即通过 uapi_drv_cipher_hash_get 接口拷贝当前 hash handle1 信息，通过 uapi_drv_cipher_hash_set 接口设置至另一个 hash handle2，由新的 hash handle2 继续完成后续计算操作。若 hash handle1 与 hash handle2 后续计算操作一致，则两者最终得到的 hash 摘要结果一致。
- 支持重试，即在调用 uapi_drv_cipher_hash_update/uapi_drv_cipher_hash_finish 接口失败时，仍可配置正确参数后继续重新调用该接口，完成后续摘要计算操作。

1.2.5 HMAC 计算

1.2.5.1 场景说明

依赖调用者传入的密钥数据，获取消息摘要，该密钥数据由消息交换双方提前协商好。支持以下场景：源数据所在位置为 DDR、RAM、flash。

1.2.5.2 工作流程

步骤 1 调用 uapi_drv_cipher_hash_init 初始化 HASH 模块。

步骤 2 创建一路 hash，配置当前 hash 计算类型和密钥所在 keyslot 通道，并获取 hash 句柄。调用 uapi_drv_cipher_hash_start 接口完成。

步骤 3 传入消息数据，对消息进行摘要计算，调用 uapi_drv_cipher_hash_update 接口完成。

步骤 4 获取消息摘要，调用 uapi_drv_cipher_hash_finish 接口完成。

步骤 5 调用 uapi_drv_cipher_hash_deinit 去初始化 HASH 模块。

说明

步骤 4 如果成功会自己销毁 hash 句柄，不需要额外调用 hash 接口。步骤 2 启动成功后，无论后续步骤成功或失败，均可调用 uapi_drv_cipher_hash_destroy 接口销毁 hash 句柄（步骤 5 成功除外）。

----结束

1.2.5.3 注意事项

在调用 uapi_drv_cipher_hash_finish 接口获取消息摘要时，传入的 result buffer 需要调用者申请，其大小为 result_len，且 result_len 不应小于当前 hash 类型对应的摘要计算结果长度。

调用者传入的密钥数据，如果是明文密钥，则要求密钥长度不能大于当前 hash 类型的 block_size 大小，如果密钥长度大于当前 hash 类型的 block_size 大小，则需要调用者

先对密钥进行 hash 计算，使得其长度小于 block_size。如果是密文密钥，则密钥长度只能为 16 bytes、24bytes 或 32 bytes。

1.2.6 RSA 加解密

1.2.6.1 场景说明

RSA 加解密应用场景为公钥加密私钥解密。

1.2.6.2 工作流程

步骤 1 公钥加密，调用 uapi_drv_cipher_pke_rsa_public_encrypt 接口完成。

步骤 2 私钥解密，调用 uapi_drv_cipher_pke_rsa_private_decrypt 接口完成。

说明

调用加密接口时，存储加密结果的 buffer，其缓冲区大小不能小于密钥 N 的位宽。

----结束

1.2.6.3 注意事项

支持 PKCS#1 v1.5 和 PKCS#1 v2.1 OAEP 两种模式，且用户标签数据（label）可选，该 label 仅在 PKCS#1 v2.1 OAEP 模式下使用。PKCS#1 v1.5 为不安全算法，不建议使用。

1.2.7 RSA 签名验签

1.2.7.1 场景说明

RSA 签名验签主要应用场景是私钥签名公钥验签。

1.2.7.2 工作流程

步骤 1 私钥签名，调用 uapi_drv_cipher_pke_rsa_sign 接口完成。

步骤 2 公钥验签，调用 uapi_drv_cipher_pke_rsa_verify 接口完成。

----结束

1.2.7.3 注意事项

支持 PKCS#1 v1.5 和 PKCS#1 v2.1 PSS 两种模式的私钥签名和公钥验签。PKCS#1 v1.5 为不安全算法，不建议使用。

1.2.8 SM2 加解密

1.2.8.1 场景说明

SM2 加解密应用场景是公钥加密私钥解密。

1.2.8.2 工作流程

步骤 1 公钥加密，调用 uapi_drv_cipher_pke_sm2_public_encrypt 接口完成。

步骤 2 私钥解密，调用 uapi_drv_cipher_pke_sm2_private_decrypt 接口完成。

说明

加解密所需要的公私钥对可调用 uapi_drv_cipher_pke_ecc_gen_key 生成。

----结束

1.2.9 SM2 签名验签

1.2.9.1 场景说明

SM2 签名验签主要应用场景是私钥签名公钥验签。

1.2.9.2 工作流程

步骤 1 生成消息对应的 hash 摘要，调用 uapi_drv_cipher_pke_sm2_dsa_hash 接口完成。

步骤 2 私钥签名，调用 uapi_drv_cipher_pke_ecdsa_sign 接口完成。

步骤 3 公钥验签，调用 uapi_drv_cipher_pke_ecdsa_verify 接口完成。

说明

签名验签所需要的公私钥对可调用 uapi_drv_cipher_pke_ecc_gen_key 接口生成，可以调用 uapi_drv_cipher_pke_check_dot_on_curve 接口以确认公钥确实是曲线上的点。

----结束

1.2.10 ECC 签名验签

1.2.10.1 场景说明

ECC 签名验签主要应用场景是私钥签名公钥验签。

1.2.10.2 工作流程

步骤 1 生成消息对应的 hash 摘要，调用 HASH/HMAC 相应接口完成。

步骤 2 私钥签名，调用 uapi_drv_cipher_pke_ecdsa_sign 接口完成。

步骤 3 公钥验签，调用 uapi_drv_cipher_pke_ecdsa_verify 接口完成。

说明

签名验签所需要的公私钥对可调用 uapi_drv_cipher_pke_ecc_gen_key 接口生成，可以调用 uapi_drv_cipher_pke_check_dot_on_curve 接口以确认公钥确实是曲线上的点，签名需要先计算 hash 摘要，签名过程是对摘要进行签名。

----结束

1.2.10.3 注意事项

支持 RFC 5639 - Brainpool P256/384/512r1、NIST FIPS 186 - 4 P192/224/256/384/521 的私钥签名和公钥验签。

1.2.11 EDDSA 签名验签

1.2.11.1 场景说明

EDDSA 签名验签主要应用场景是私钥签名公钥验签，是使用 ed25519 这种特殊曲线签名验签。

1.2.11.2 工作流程

步骤 1 私钥签名，调用 uapi_drv_cipher_pke_eddsa_sign 接口完成。

步骤 2 公钥验签，调用 uapi_drv_cipher_pke_eddsa_verify 接口完成。

说明

签名验签所需要的公私钥对可调用 uapi_drv_cipher_pke_ecc_gen_key 接口生成，可以调用 uapi_drv_cipher_pke_check_dot_on_curve 接口以确认公钥确实是曲线上的点，EDDSA 签名不需要提前计算 hash，直接传入消息进行签名即可。

----结束

1.2.11.3 注意事项

支持 RFC 8032 - ED25519 的私钥签名和公钥验签。

1.2.12 大数运算

1.2.12.1 场景说明

大数运算主要用于非对称加解密和签名验签的计算过程。

1.2.12.2 工作流程

直接调用大数计算接口完成相应类型的大数计算即可，例如模加可直接调用 uapi_drv_cipher_pke_add_mod 接口完成。

1.2.13 ECDH 密钥协商

1.2.13.1 场景说明

密钥协商主要用于消息传递双方在生成各自的公私钥对之后进行密钥交换。

1.2.13.2 工作流程

步骤 1 消息传递者 a 生成自己的公私钥对，调用 uapi_drv_cipher_pke_ecc_gen_key 接口完成。

步骤 2 消息接收者 b 生成自己的公私钥对，调用 uapi_drv_cipher_pke_ecc_gen_key 接口完成。

步骤 3 密钥协商，消息传递者 a/消息接收者 b 使用自己的私钥和对方的公钥生成共享密钥，调用 uapi_drv_cipher_pke_ecc_gen_ecdh_key 接口完成。

说明

- 对于给定的私钥 uapi_drv_cipher_pke_ecc_gen_key 生成的公钥总是固定的；对于未给定私钥的情况下，uapi_drv_cipher_pke_ecc_gen_key 生成的公私钥对是随机的。
- 共享密钥生成后，消息传递者双方可对生成的共享密钥做比较，以验证共享密钥的正确性。

----结束

1.2.13.3 注意事项

- 支持 RFC 5639 - Brainpool P256/384/512r1、NIST FIPS 186 - 4 P192/224/256/384/521、RFC 7748 - Curve448/Curve25519、RFC 8032 - ED25519（即 EDDSA）和 SM2 的公私钥对生成，及由给定的私钥生成其对应的公钥。
- 支持 RFC 5639 - Brainpool P256/384/512r1、NIST FIPS 186 - 4 P192/224/256/384/521、RFC 7748 - Curve448/Curve25519 和 SM2 的密钥交换（ECDH），低于 256bit 的 ECDH 及 NIST FIPS 对应的曲线为不安全算法，不建议使用。

1.2.14 DH 密钥协商

1.2.14.1 场景说明

密钥协商主要用于消息传递双方在生成各自的公私钥对之后进行密钥交换，DH 密钥协商用于非曲线类型。

1.2.14.2 工作流程

步骤 1 消息传递者 a 生成自己的公私钥对，调用 uapi_drv_cipher_pke_dh_gen_key 接口完成。

步骤 2 消息接收者 b 生成自己的公私钥对，调用 uapi_drv_cipher_pke_dh_gen_key 接口完成。

步骤 3 密钥协商，消息传递者 a/消息接收者 b 使用自己的私钥和对方的公钥生成共享密钥，调用 uapi_drv_cipher_pke_dh_compute_key 接口完成。

说明

- 对于给定的私钥 uapi_drv_cipher_pke_dh_gen_key 生成的公钥总是固定的；对于未给定私钥的情况下，uapi_drv_cipher_pke_dh_gen_key 生成的公私钥对是随机的。
- 共享密钥生成后，消息传递者双方可对生成的共享密钥做比较，以验证共享密钥的正确性。

----结束

1.2.14.3 注意事项

- 支持 192/224/256/384/512/521/1024/2048/3072/4096 比特 DH 密钥协商，低于 3072bits 的 DH 为不安全算法，不建议使用。

1.2.15 FLASH 在线解密

1.2.15.1 场景说明

由于运行在 CPU 上的 OS 和应用程序，都是加密且运行在 FLASH 上。所以，安全启动过程中， Boot 需要配置 OS 镜像的 FLASH 在线解密。该模块主要暴露给 boot 使用，上层业务不使用。

1.2.15.2 工作流程

步骤 1 配置要执行在线解密操作的 region 及其对应 flash 区域的起始地址，调用 drv_fapc_set_config 接口完成。

步骤 2 配置在线解密操作使用的 iv，调用 drv_fapc_set_iv 接口完成。

步骤 3 配置在线解密操作使用的 key，详见“1.2.1 密钥配置”小节。

步骤 4 直接读取已配置在线解密操作的 flash 区域内容，读取结果即为解密之后的结果。

----结束

1.2.15.3 注意事项

FLASH 在线解密使用的算法是 AES_128_CTR，因此在使用该功能时，对应 flash 区域的加密方式也应是 AES_128_CTR。

boot 配置镜像的 FLASH 在线解密时，使用的 root key 类型由对应芯片自行选择。

1.2.16 安全相关 efuse 烧写建议

安全特性相关的 efuse 位较多，涉及到安全特性的正确使用，此处将逐一列出相关 efuse 及对应使用场景和烧写建议供参考，各项目根据具体需求及实际 efuse 排布表单进行合理配置。表 1-1 是 OEM 不同场景下使用的 efuse，及其建议烧写阶段和建议烧写值。

表1-1 安全特性 efuse 使用场景及烧写建议

使用场景	关键项	烧写阶段	烧写完是否需要 lock	烧写方式及烧写值
安全启动	sec_verify_enable	OEM	否	Burntool 烧写工具，根据是否开启安全启动功能确定。

使用场景	关键项	烧写阶段	烧写完是否需要 lock	烧写方式及烧写值
	mcu_ver	OEM	否	若支持防回滚功能，则通过 Burntool 烧写工具，根据实际情况烧写正确的镜像版本号信息。
	flashboot_ver	OEM	否	若支持防回滚功能，则通过 Burntool 烧写工具，根据实际情况烧写正确的镜像版本号信息。
	params_ver	OEM	否	若支持防回滚功能，则通过 Burntool 烧写工具，根据实际情况烧写正确的镜像版本号信息。
	Hash_root_public_key	OEM	是	Burntool 烧写工具，根公钥镜像 HASH 值。
	MSID	OEM	否	Burntool 烧写工具，市场 ID。
安全驱动	otp_crc_rd_disable (KM)	OEM	否	Burntool 烧写工具，根据是否保留 CRC debug 功能确定烧写值。
	sha1_disable	OEM	否	Burntool 烧写工具，根据是否关闭 SHA1 来确定。
	rkp_deob_alg_sel	OEM	否	Burntool 烧写工具，配置 DEOB 算法。
	sm_disable	OEM	否	Burntool 烧写工具，根据是否关闭国密来确定。
AES/SM4 加解密，HMAC	obfu_mrk1_owner_id	OEM	否	Burntool 烧写工具，16bit 随机值。
	obfu_mrk	OEM	否	Burntool 烧写工具，128bit 随机值，加解密和 HMAC 硬

使用场景	关键项	烧写阶段	烧写完是否需要 lock	烧写方式及烧写值
				件密钥派生。
安全存储	obfu_rusk	OEM	否	Burntool 烧写工具，128bit 随机值，用于派生安全存储密钥。

2 错误码

说明

本章描述了 CIPHER DRIVER 的错误码，便于开发人员理解与定位所发生的错误。

2.1 CIPHER DRIVER 错误码结构与说明

Cipher 驱动返回值通常有成功和失败两种类型，如表 2-1 所示。

表2-1 Cipher 驱动返回值

成员名称	枚举值
CRYPTO_SUCCESS	0
CRYPTO_FAILURE	0xFFFFFFFF

为了通过错误码更加清晰的标识错误发生的具体信息，采用了层级结构信息拼接。其结构为：

Env(4 bits) | Layer(4 bits) | Modules(4 bits) | Reserved(12 bits) | Error Code(8 bits)

2.1.1 ENV

【描述】

错误码发生的操作系统环境。

【定义】

```
#define ERROR_ENV_LINUX          0x1
```

```
#define ERROR_ENV_ITRUSTEE          0x2
#define ERROR_ENV_OPTEE             0x3
#define ERROR_ENV_LITEOS            0x4
#define ERROR_ENV_SELITEOS          0x5
#define ERROR_ENV_NOOS              0x6
```

【成员】

成员名称	枚举值
ERROR_ENV_LINUX	LINUX
ERROR_ENV_ITRUSTEE	ITRUSTEE
ERROR_ENV_OPTEE	OPTEE
ERROR_ENV_LITEOS	LITEOS
ERROR_ENV_SELITEOS	SELITEOS
ERROR_ENV_NOOS	无 OS 环境

【注意】

无

2.1.2 LAYER

【描述】

错误码发生的代码层级。

【定义】

```
enum {
    ERROR_LAYER_UAPI = 0x1,
    ERROR_LAYER_DISPATCH,
    ERROR_LAYER_KAPI,
    ERROR_LAYER_DRV,
    ERROR_LAYER_HAL,
};
```

【成员】

成员名称	枚举值
ERROR_LAYER_UAPI	用户接口层

成员名称	枚举值
ERROR_LAYER_DISPATCH	指令分发层
ERROR_LAYER_KAPI	内核接口层
ERROR_LAYER_DRV	驱动逻辑层
ERROR_LAYER_HAL	硬件适配层

【注意】

无

2.1.3 MODULE

【描述】

错误码发生的 CIPHER 模块。

【定义】

```
enum {
    ERROR_MODULE_SYMC = 0x1,
    ERROR_MODULE_HASH,
    ERROR_MODULE_PKE,
    ERROR_MODULE_TRNG,
    ERROR_MODULE_OTHER
};
```

【成员】

成员名称	枚举值
ERROR_MODULE_SYMC	对称加解密模块
ERROR_MODULE_HASH	摘要算法模块
ERROR_MODULE_PKE	非对称加解密模块
ERROR_MODULE_TRNG	随机数模块
ERROR_MODULE_OTHER	其他模块

【注意】

无

2.1.4 错误类型

【描述】

错误码发生的具体错误类型。

【定义】

```
/* Common Error Code. 0x00 ~ 0x3F. */
ERROR_INVALID_PARAM = 0x0,          /* return when the input param's value is not in the valid
range. */
ERROR_PARAM_IS_NULL,                /* return when the input param is NULL and required not
NULL. */
ERROR_NOT_INIT,                     /* return when call other functions before call init function. */
ERROR_UNSUPPORT,                    /* return when some configuration is unsupport. */
ERROR_UNEXPECTED,                   /* reture when unexpected error occurs. */
ERROR_CHN_BUSY,                     /* return when try to create one channel but all channels
are busy. */
ERROR_CTX_CLOSED,                   /* return when using one ctx to do something but has
been closed. */
ERROR_NOT_SET_CONFIG,               /* return when not set_config but need for symc. */
ERROR_NOT_ATTACHED,                 /* return when not attach but need for symc. */
ERROR_NOT_MAC_START,                /* return when not mac_start but need for symc. */
ERROR_INVALID_HANDLE,               /* return when pass one invalid handle. */
ERROR_GET_PHYS_ADDR,                /* return when transfer from virt_addr to phys_addr
failed. */
ERROR_SYMC_LEN_NOT_ALIGNED,         /* return when length isn't aligned to 16-Byte except
CTR/CCM/GCM.   */
ERROR_SYMC_ADDR_NOT_ALIGNED,        /* return when the phys_addr writing to register is not
aligned to 4-Byte. */
ERROR_PKE_RSA_SAME_DATA,            /* return when rsa exp_mod, the input is equal to output.
*/
ERROR_PKE_RSA_CRYPTO_V15_CHECK, /* return when rsa crypto v15 padding check failed. */
ERROR_PKE_RSA_CRYPTO_OAEP_CHECK,    /* return when rsa crypto oaep padding check
failed. */
ERROR_PKE_RSA_VERIFY_V15_CHECK,     /* return when rsa verify v15 padding check failed.
*/
ERROR_PKE_RSA_VERIFY_PSS_CHECK,     /* return when rsa verify pss padding check
failed. */
ERROR_PKE_RSA_GEN_KEY,              /* return when rsa generate key failed. */
ERROR_PKE_ECDSA_VERIFY_CHECK,   /* return when ecdsa verify check failed. */
/* Outer's Error Code. 0x40 ~ 0x5F. */
```

```
ERROR_MEMCPY_S          = 0x40,        /* return when call memcpy_s failed. */
ERROR_MALLOC,                          /* return when call xxx_malloc failed. */
ERROR_MUTEX_INIT,                      /* return when call xxx_mutex_init failed. */
ERROR_MUTEX_LOCK,                      /* return when call xxx_lock failed. */
/* Specific Error Code for UAPI. 0x60 ~ 0x6F. */
ERROR_DEV_OPEN_FAILED,                 /* return when open dev failed. */
ERROR_COUNT_OVERFLOW,                  /* return when call init too many times. */
/* Specific Error Code for Dispatch. 0x70 ~ 0x7F. */
ERROR_CMD_DISMATCH    = 0x70,          /* return when cmd is dismatched. */
ERROR_COPY_FROM_USER,                  /* return when call copy_from_user failed. */
ERROR_COPY_TO_USER,                    /* return when call copy_to_user failed. */
ERROR_MEM_HANDLE_GET,                  /* return when parse user's mem handle to kernel's
mem handle failed. */
ERROR_GET_OWNER,                       /* return when call crypto_get_owner failed. */
/* Specific Error Code for KAPI. 0x80 ~ 0x8F. */
ERROR_PROCESS_NOT_INIT = 0x80,    /* return when one process not call kapi_xxx_init first. */
ERROR_MAX_PROCESS,                     /* return when process's num is over the limit. */
ERROR_MEMORY_ACCESS,                   /* return when access the memory that does not
belong to itself.   */
ERROR_INVALID_PROCESS,                 /* return when the process accesses resources of other
processes. */
/* Specific Error Code for DRV. 0x90 ~ 0x9F. */
/* Specific Error Code for HAL. 0xA0 ~ 0xAF. */
ERROR_HASH_LOGIC        = 0xA0,        /* return when hash logic's error occurs. */
ERROR_PKE_LOGIC,                       /* return when pke logic's error occurs. */
ERROR_INVALID_CPU_TYPE,                /* return when logic get the invalid cpu type. */
ERROR_INVALID_REGISTER_VALUE,     /* return when value in register is invalid. */
ERROR_INVALID_PHYS_ADDR,               /* return when phys_addr is invalid. */
/* Specific Error Code for Timeout. 0xB0 ~ 0xBF. */
ERROR_GET_TRNG_TIMEOUT = 0xB0,    /* return when logic get rnd timeout. */
ERROR_HASH_CLEAR_CHN_TIMEOUT,     /* return when clear hash channel timeout. */
ERROR_HASH_CALC_TIMEOUT,               /* return when hash calculation timeout. */
ERROR_SYMC_CLEAR_CHN_TIMEOUT,     /* return when clear symc channel timeout. */
ERROR_SYMC_CALC_TIMEOUT,               /* return when symc crypto timeout. */
ERROR_SYMC_GET_TAG_TIMEOUT,        /* return when symc get tag timeout. */
ERROR_PKE_LOCK_TIMEOUT,                /* return when pke lock timeout. */
ERROR_PKE_WAIT_DONE_TIMEOUT,       /* return when pke wait done timeout. */
ERROR_PKE_ROBUST_WARNING,       /* return when pke get robust warning. */
```

【成员】

类型	描述

类型	描述	
COMMON	ERROR_INVALID_PARAM	输入参数在无效范围
	ERROR_PARAM_IS_NULL	输入非法空指针参数
	ERROR_NOT_INIT	未调用初始化函数
	ERROR_UNSUPPORT	算法或配置未支持
	ERROR_UNEXPECTED	预期之外的错误
	ERROR_CHN_BUSY	通道均被占用时尝试创建通道
	ERROR_CTX_CLOSED	尝试使用已关闭的 CTX
	ERROR_NOT_SET_CONFIG	SYMC 未调用 set_config 函数
	ERROR_NOT_ATTACHED	SYMC 未绑定 KEYSLOT
	ERROR_NOT_MAC_START	SYMC 未调用 mac_start 函数
	ERROR_INVALID_HANDLE	传入了无效的 handle
	ERROR_GET_PHYS_ADDR	获取物理地址失败
	ERROR_SYMC_LEN_NOT_ALIGNED	SYMC 长度未对其到 16Byte 除了（CTR/CCM/GCM）模式
	ERROR_SYMC_ADDR_NOT_ALIGNED	SYMC 写到寄存器的物理地址没有按照 4Byte 对齐
	ERROR_PKE_RSA_SAME_DATA	RSA 模密运算时，输入和输出相等
	ERROR_PKE_RSA_CRYPTO_V15_CHECK	RSA PKCSV15 解密 Padding 校验失败
	ERROR_PKE_RSA_CRYPTO_OAEP_CHECK	RSA OAEP 解密 Padding 校验失败
	ERROR_PKE_RSA_VERIFY	RSA PKCSV15 验签

类型	描述	
	_V15_CHECK	Padding 校验失败
	ERROR_PKE_RSA_VERIFY_PSS_CHECK	RSA PSS 验签 Padding 校验失败
	ERROR_PKE_RSA_GEN_KEY	RSA 密钥生成失败 该功能暂不支持
	ERROR_PKE_ECDSA_VERIFY_CHECK	ECDSA 验签失败
OUTER	ERROR_MEMCPY_S	调用 memcpy_s 失败
	ERROR_MALLOC	调用 xxx_malloc 失败
	ERROR_MUTEX_INIT	调用 xxx_mutex_init 失败
	ERROR_MUTEX_LOCK	调用 xxx_lock 失败
SPECIFIC UAPI	ERROR_DEV_OPEN_FAILED	打开设备失败
	ERROR_COUNT_OVERFLOW	调用太多次初始化
SPECIFIC DISPATCH	ERROR_CMD_DISMATCHED	CMD 无匹配项
	ERROR_COPY_FROM_USER	从用户态拷贝发生错误
	ERROR_COPY_TO_USER	向用户态拷贝发生错误
	ERROR_MEM_HANDLE_GET	获取句柄发生错误
	ERROR_GET_OWNER	获取使用者发生错误
SPECIFIC KAPI	ERROR_PROCESS_NOT_INIT	进程未初始化
	ERROR_MAX_PROCESS	数值超过了最大值
	ERROR_MEMORY_ACCESS	没有该内存的访问权限
	ERROR_INVALID_PROCESS	没有访问资源权限的进程访问了该资源
SPECIFIC HAL	ERROR_HASH_LOGIC	HASH 发生硬件逻辑错误

类型	描述	
	ERROR_PKE_LOGIC	PKE 发生硬件逻辑错误
	ERROR_INVALID_CPU_TYPE	逻辑拿到了无效的 CPU 类型
	ERROR_INVALID_REGISTER_VALUE	寄存器中的值是无效的
	ERROR_INVALID_PHYS_ADDR	无效的物理地址
TIMEOUT	ERROR_GET_TRNG_TIMEOUT	获取随机数超时
	ERROR_HASH_CLEAR_CHN_TIMEOUT	HASH 清通道超时
	ERROR_HASH_CALC_TIMEOUT	HASH 计算超时
	ERROR_SYMC_CLEAR_CHN_TIMEOUT	SYMC 清通道超时
	ERROR_SYMC_CALC_TIMEOUT	SYMC 计算超时
	ERROR_SYMC_GET_TAG_TIMEOUT	获取 tag 值超时
	ERROR_PKE_LOCK_TIMEOUT	PKE 锁超时
	ERROR_PKE_WAIT_DONE_TIMEOUT	PKE 计算超时
	ERROR_PKE_ROBUST_WARNING	PKE 鲁棒性告警

说明

security_unified 对错误码进行了层级设计，方便 debug 时快速定位。用户调用 service 层接口时返回的错误的错误码不带层级信息，其错误含义与上述表格中的错误码一一对应，具体参见 API 手册 security_unified 模块错误码相关定义，错误码段为 0x80001500~0x8001600.

3 网络安全注意事项

3.1 安全驱动

- Cipher 驱动实现了标准的对称加密 AES/ SM4 算法，非对称 RSA、ECDSA/SM2/ED25519 算法，摘要算法 SHA/SM3/HMAC_SHA/SM3，密钥协商算法 ECDH，密钥派生算法 PBKDF2，没有使用任何私有算法。
- 对称算法使用建议：
  - AES/SM4 算法 ECB 模式为非安全算法，不建议使用。
  - 对称算法的密钥的长度越长，安全等级越高，建议用户使用 AES 128bit 及以上的密钥。
- 非对称算法使用建议：
  - RSA 算法建议使用密钥长度 3072bit 及以上的密钥。
  - RSA 签名算法 padding 方式建议使用 PSS padding 方式。
  - RSA 加解密算法 padding 方式建议使用 OAEP padding 方式。
  - ECDSA 算法建议使用密钥长度 256bit 及以上的密钥。
  - ECDSA 算法曲线不建议使用 NIST P256/384/521 曲线。
- 摘要算法使用建议：
  - SHA1/SHA224 算法安全性低，不建议用户使用。
  - SHA/HMAC_SHA 算法建议使用 SHA256/HMAC_SHA256 及以上的算法。
- 密钥协商算法使用建议：

  - ECDH 算法建议使用密钥长度 256bits 及以上的密钥。
  - DH 算法建议使用密钥长度 3072bits 及以上的密钥。
- 密钥派生算法使用建议：
  - 迭代次数和安全性成正比例，也与计算时间成正比例，迭代次数越大，意味着计算密钥花费时间越长，同时抗暴力破解能力越强。对于性能不敏感或高安全性要求场景推荐迭代次数至少需要 10000000 次，其他场景迭代次数默认推荐至少 10000 次；对于性能有特殊要求的产品最低可以迭代 1000 次（NIST SP 800 132）。
  - 哈希函数推荐选择 SHA256 或更安全的哈希算法。
  - 盐值至少 16Byte，应使用安全的随机数。
  - 用于口令单向哈希时，其输出长度应该不小于 256bit。

3.2 其他使用安全注意事项

3.2.1 JTAG 接口

恶意者可通过 JTAG 接口，篡改系统和配置，恶意破坏系统。

建议用户采用以下措施：

- 产品出厂时，将 JTAG 接口从物理上删除。
- 芯片提供 JTAG disable 功能，软件通过写 EFUSE，可从芯片层面直接永久关闭 JTAG。