



CIPHER

## API 参考

文档版本     00B05

发布日期     2016-05-10

**版权所有 © 深圳市海思半导体有限公司 2016。保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## **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电气生产中心                    邮编：518129

网址：                    <http://www.hisilicon.com>

客户服务电话：          +86-755-28788858

客户服务传真：          +86-755-28357515

客户服务邮箱：          [support@hisilicon.com](mailto:support@hisilicon.com)



# 前 言

## 概述

CIPHER 是海思数字媒体处理平台提供的安全算法模块，它提供了 DES、3DES 和 AES 三种对称加解密算法，HASH 及 HMAC 摘要算法，随机数算法，以及 RSA 不对称算法，主要用于对音视频码流进行加解密及数据合法性验证等场景。



说明

未有特殊说明，Hi3518EV201、Hi3516CV200 与 Hi3518EV200 完全一致。

## 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
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3519	V100
Hi3519	V101

## 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
 危险	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。
 警告	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。
 注意	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。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

## 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。

### 文档版本 00B05 (2016-05-10)

第 5 次临时发布

第 2 章，新增 HI\_UNF\_CIPHER\_KladEncryptKey。

第 3 章，新增 HI\_UNF\_CIPHER\_KLAD\_TARGET\_E，

HI\_UNF\_CIPHER\_KEY\_SRC\_E、HI\_UNF\_CIPHER\_RSA\_PRI\_ENC\_S 和  
HI\_UNF\_CIPHER\_RSA\_SIGN\_S 涉及修改

1.3 小节删除“HMAC 不支持多线程”

新增第 5 章“Proc 调试信息”

### 文档版本 00B04 (2015-12-28)

第 4 次临时发布

#### 第 1 章 概述

1.3 小节有修改

#### 第 2 章 API 参考

修改 HI\_UNF\_CIPHER\_EncryptMultiEx 和 HI\_UNF\_CIPHER\_DecryptMultiEx 中的【注意】，新增 HI\_UNF\_CIPHER\_WriteOTPKey

#### 第 3 章 数据类型

HI\_UNF\_CIPHER\_GCM\_INFO\_S 和 HI\_UNF\_CIPHER\_CTRL\_S 涉及修改



## 文档版本 00B03 (2015-11-20)

第 3 次临时发布。

### 第 1 章 概述

修改 1.2.1 小节，新增 1.2.2 和 1.2.3 节。

### 第 2 章 API 参考

新增 HI\_UNF\_CIPHER\_EncryptMultiEx、HI\_UNF\_CIPHER\_DecryptMultiEx、  
HI\_UNF\_CIPHER\_GetTag 和 HI\_UNF\_CIPHER\_Hash

### 第 3 章 数据类型

HI\_UNF\_CIPHER\_CCM\_INFO\_S 和 HI\_UNF\_CIPHER\_GCM\_INFO\_S 涉及修改。

## 文档版本 00B02 (2015-09-25)

第 2 次临时发布。

增加 Hi3519V100 相关内容。

## 文档版本 00B01 (2015-08-05)

第 1 次临时发布。



# 目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 使用流程.....	1
1.2.1 单包对称加解密.....	1
1.2.2 多包对称加解密.....	2
1.2.3 多包对称加解密扩展接口.....	2
1.2.4 不对称加解密.....	2
1.2.5 随机数生成.....	3
1.2.6 摘要计算.....	3
1.3 注意事项.....	3
2 API 参考 .....	5
3 数据类型.....	29
4 错误码.....	48
5 Proc 调试信息.....	49
5.1 CIPHER 状态.....	49



## 表格目录

表 4-1 CIPHER 模块的错误码 .....	48
---------------------------	----



# 1 概述

## 1.1 概述

CIPHER 是海思数字媒体处理平台提供的安全算法模块，其提供了包括 DES、3DES 和 AES 等对称加解密算法，RSA 不对称加解密算法，随机数生成，以及 HASH、HMAC 等摘要算法，主要用于对音视频码流进行加解密保护，用户合法性认证等场景。

## 1.2 使用流程

### 1.2.1 单包对称加解密

对数据进行对称的 DES/3DES/AES 加解密的过程如下：

- 步骤 1. CIPHER 设备初始化。调用接口 [HI\\_UNF\\_CIPHER\\_Init](#) 完成。
  - 步骤 2. 创建一路 CIPHER，并获取 CIPHER 句柄。调用接口 [HI\\_UNF\\_CIPHER\\_CreateHandle](#) 完成。
  - 步骤 3. 配置 CIPHER 控制信息，包含密钥、初始向量、加密算法、工作模式等信息。调用接口 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#) 完成。
  - 步骤 4. 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。
    - 单包加密--[HI\\_UNF\\_CIPHER\\_Encrypt](#)
    - 单包解密--[HI\\_UNF\\_CIPHER\\_Decrypt](#)
  - 步骤 5. 如果是 CCM、GCM 模式，调用 [HI\\_UNF\\_CIPHER\\_GetTag](#) 获取 TAG 值，否则执行下一步。
  - 步骤 6. 销毁 CIPHER 句柄。调用接口 [HI\\_UNF\\_CIPHER\\_DestroyHandle](#) 完成。
  - 步骤 7. 关闭 CIPHER 设备。调用接口 [HI\\_UNF\\_CIPHER\\_Deinit](#) 完成。
- 结束





## 1.2.2 多包对称加解密

对数据进行对称的 DES/3DES/AES 加解密的过程如下：

- 步骤 1. CIPHER 设备初始化。调用接口 [HI\\_UNF\\_CIPHER\\_Init](#) 完成。
- 步骤 2. 创建一路 CIPHER，并获取 CIPHER 句柄。调用接口 [HI\\_UNF\\_CIPHER\\_CreateHandle](#) 完成。
- 步骤 3. 配置 CIPHER 控制信息，包含密钥、初始向量、加密算法、工作模式等信息。调用接口 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#) 完成。
- 步骤 4. 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。
  - 多包加密--[HI\\_UNF\\_CIPHER\\_EncryptMulti](#)
  - 多包解密--[HI\\_UNF\\_CIPHER\\_DecryptMulti](#)
- 步骤 5. 销毁 CIPHER 句柄。调用接口 [HI\\_UNF\\_CIPHER\\_DestroyHandle](#) 完成。
- 步骤 6. 关闭 CIPHER 设备。调用接口 [HI\\_UNF\\_CIPHER\\_Deinit](#) 完成。

----结束

## 1.2.3 多包对称加解密扩展接口

对数据进行对称的 DES/3DES/AES 加解密的过程如下：

- 步骤 1. CIPHER 设备初始化。调用接口 [HI\\_UNF\\_CIPHER\\_Init](#) 完成。
- 步骤 2. 创建一路 CIPHER，并获取 CIPHER 句柄。调用接口 [HI\\_UNF\\_CIPHER\\_CreateHandle](#) 完成。
- 步骤 3. 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
  - 多包加密--[HI\\_UNF\\_CIPHER\\_EncryptMultiEx](#)
  - 多包解密--[HI\\_UNF\\_CIPHER\\_DecryptMultiEx](#)
- 步骤 4. 销毁 CIPHER 句柄。调用接口 [HI\\_UNF\\_CIPHER\\_DestroyHandle](#) 完成
- 步骤 5. 关闭 CIPHER 设备。调用接口 [HI\\_UNF\\_CIPHER\\_Deinit](#) 完成。

----结束

## 1.2.4 不对称加解密

对数据进行不对称的 RSA 加解密的过程如下：

- 步骤 1. CIPHER 设备初始化。调用接口 [HI\\_UNF\\_CIPHER\\_Init](#) 完成。
- 步骤 2. 对数据进行加解密或签名验证。根据使用的密钥不同，分为 6 个接口，用户可以调用以下任一接口进行加解密、签名验证、生成密钥对等。
  - 公钥加密--[HI\\_UNF\\_CIPHER\\_RsaPublicEnc](#)
  - 私钥解密-- [HI\\_UNF\\_CIPHER\\_RsaPrivateDec](#)



- 私钥加密--[HI\\_UNF\\_CIPHER\\_RsaPrivateEnc](#)
- 公钥解密--[HI\\_UNF\\_CIPHER\\_RsaPublicDec](#)
- 私钥签名--[HI\\_UNF\\_CIPHER\\_RsaSign](#)
- 公钥验证--[HI\\_UNF\\_CIPHER\\_RsaVerify](#)
- 生成密钥--[HI\\_UNF\\_CIPHER\\_RsaGenKey](#)

步骤 3. 关闭 CIPHER 设备。调用接口 [HI\\_UNF\\_CIPHER\\_Deinit](#) 完成。

----结束

## 1.2.5 随机数生成

生成随机数据的过程如下：

步骤 1. CIPHER 设备初始化。调用接口 [HI\\_UNF\\_CIPHER\\_Init](#) 完成。

步骤 2. 生成随机数。调用接口 [HI\\_UNF\\_CIPHER\\_GetRandomNumber](#) 完成。

步骤 3. 关闭 CIPHER 设备。调用接口 [HI\\_UNF\\_CIPHER\\_Deinit](#) 完成。

----结束

## 1.2.6 摘要计算

步骤 1. CIPHER 设备初始化。调用接口 [HI\\_UNF\\_CIPHER\\_Init](#) 完成。

步骤 2. 创建一路 HASH，并获取 HASH 句柄。调用接口 [HI\\_UNF\\_CIPHER\\_HashInit](#) 完成。

步骤 3. 计算 HASH，HMAC 等摘要。调用接口 [HI\\_UNF\\_CIPHER\\_HashUpdate](#) 完成。

步骤 4. 如果摘要未计算完成，再次执行步骤 3。

步骤 5. 完成摘要计算，获取计算结果。调用接口 [HI\\_UNF\\_CIPHER\\_HashFinal](#) 完成。

步骤 6. 关闭 CIPHER 设备。调用接口 [HI\\_UNF\\_CIPHER\\_Deinit](#) 完成。

----结束

## 1.3 注意事项

- 对称加解密操作中每个数据包的长度必须小于 1MB。如果数据长度大于或等于 1M，请拆分成多个数据包进行处理。进行多包加解密时，最多支持同时加解密 128 个包。
- 单包加解密的 IV 向量可继承。创建一路 CIPHER，配置属性（假设配置的工作模式需要使用 IV 向量）之后，每次调用单包加解密接口时，IV 向量会依次轮流使用。

例如：用户需依次加密数据 0，数据 1。向量为 a,b,c,d。用户加密完数据 0 之后，数据 0 的最后一个分块数据使用了 IV 向量中的 b 进行加密处理；此时，用户再加



密数据 1 时，数据 1 的第一个分块数据将会使用 IV 向量 c 进行加密，然后依次为 d,a,b,c,d...

因此在加解密时，必须要保证两次向量使用的一致性。重新配置 CIPHER 控制信息将设置 IV 向量从第一个开始使用。

多包加解密的 IV 向量不可继承。

- 建议每次加解密之前都进行配置 CIPHER 控制信息操作，以使每次加解密操作都将从 IV 向量起始位置开始执行。
- 最多可同时创建 7 个 Cipher 通道，8 个 HASH 通道。



## 2 API 参考

CIPHER 提供以下 API:

- [HI\\_UNF\\_CIPHER\\_Init](#): 初始化 CIPHER 设备。
- [HI\\_UNF\\_CIPHER\\_Deinit](#): 去初始化 CIPHER 设备。
- [HI\\_UNF\\_CIPHER\\_CreateHandle](#): 创建一路 CIPHER, 并获取 CIPHER 句柄。
- [HI\\_UNF\\_CIPHER\\_DestroyHandle](#): 销毁某路 CIPHER。
- [HI\\_UNF\\_CIPHER\\_ConfigHandle](#): 配置 CIPHER 控制信息。
- [HI\\_UNF\\_CIPHER\\_Encrypt](#): 对数据进行加密。
- [HI\\_UNF\\_CIPHER\\_Decrypt](#): 对数据进行解密。
- [HI\\_UNF\\_CIPHER\\_EncryptMulti](#): 进行多个包数据加密。
- [HI\\_UNF\\_CIPHER\\_DecryptMulti](#): 进行多个包数据解密。
- [HI\\_UNF\\_CIPHER\\_EncryptMultiEx](#): 进行多个包数据加密。
- [HI\\_UNF\\_CIPHER\\_DecryptMultiEx](#): 进行多个包数据解密。
- [HI\\_UNF\\_CIPHER\\_GetTag](#): 获取 TAG 值。
- [HI\\_UNF\\_CIPHER\\_HashInit](#): 初始化 hash 模块。
- [HI\\_UNF\\_CIPHER\\_HashUpdate](#): 计算 hash 值。
- [HI\\_UNF\\_CIPHER\\_HashFinal](#): 获取 hash 值。
- [HI\\_UNF\\_CIPHER\\_Hash](#): 计算 hash 值。
- [HI\\_UNF\\_CIPHER\\_GetRandomNumber](#): 生成随机数据。
- [HI\\_UNF\\_CIPHER\\_RsaPublicEnc](#): 公钥加密。
- [HI\\_UNF\\_CIPHER\\_RsaPrivateDec](#): 私钥解密。
- [HI\\_UNF\\_CIPHER\\_RsaPrivateEnc](#): 私钥加密。
- [HI\\_UNF\\_CIPHER\\_RsaPublicDec](#): 公钥解密。
- [HI\\_UNF\\_CIPHER\\_RsaSign](#): 私钥签名。
- [HI\\_UNF\\_CIPHER\\_RsaVerify](#): 公钥验证。
- [HI\\_UNF\\_CIPHER\\_RsaGenKey](#): 生成密钥。
- [HI\\_UNF\\_CIPHER\\_WriteOTPKKey](#): 烧写 Key 到 OTP 区域。
- [HI\\_UNF\\_CIPHER\\_KladEncryptKey](#): 使用 KLAD 对透明密钥进行加密。



## HI\_UNF\_CIPHER\_Init

### 【描述】

初始化 CIPHER 设备。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_Init(HI_VOID);
```

### 【参数】

无。

### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

### 【注意】

无。

### 【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_Deinit

### 【描述】

去初始化 CIPHER 设备。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_DeInit(HI_VOID);
```

### 【参数】

无。

### 【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

无。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_CreateHandle

【描述】

创建一路 CIPHER，并获取 CIPHER 句柄。

【语法】

```
HI_S32 HI_UNF_CIPHER_CreateHandle(HI\_HANDLE* phCipher);
```

【参数】

参数名称	描述	输入/输出
phCipher	CIPHER 句柄指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

- phCipher 不能为空。
- 该句柄将用于数据加解密时的输入。



- 最大支持 7 路 cipher。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_DestroyHandle

【描述】

销毁一路 CIPHER。

【语法】

```
HI_S32 HI_UNF_CIPHER_DestroyHandle(HI_HANDLE hCipher);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

支持重复销毁。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_ConfigHandle

【描述】

配置 CIPHER 控制信息。详细配置请参见结构体 [HI\\_UNF\\_CIPHER\\_CTRL\\_S](#)。

【语法】

```
HI_S32 HI_UNF_CIPHER_ConfigHandle(HI_HANDLE hCipher,  
HI_UNF_CIPHER_CTRL_S* pstCtrl);
```



【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
pstCtrl	控制信息指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

控制信息指针不能为空。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_Encrypt

【描述】

对数据进行加密。

【语法】

```
HI_S32 HI_UNF_CIPHER_Encrypt(HI\_HANDLE hCipher, HI_U32 u32SrcPhyAddr,  
HI_U32 u32DestPhyAddr, HI_U32 u32ByteLength);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
u32SrcPhyAddr	源数据（待加密的数据）的物理地址。	输入
u32DestPhyAddr	存放加密结果的物理地址。	输出
u32ByteLength	数据的长度（单位：字节）。	输入





【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 数据的长度至少为 16 字节，且不能大于或等于 1024\*1024 字节。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_Decrypt

【描述】

对数据进行解密。

【语法】

```
HI_S32 HI_UNF_CIPHER_Decrypt(HI_HANDLE hCipher, HI_U32 u32SrcPhyAddr,  
HI_U32 u32DestPhyAddr, HI_U32 u32ByteLength);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
u32SrcPhyAddr	源数据（待解密的数据）的物理地址。	输入
u32DestPhyAddr	存放解密结果的物理地址。	输出
u32ByteLength	数据的长度（单位：字节）。	输入

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 数据的长度至少为 16 字节，且不能大于或等于 1024x1024 字节。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_EncryptMulti

【描述】

进行多个包数据的加密。

【语法】

```
HI_S32 HI_UNF_CIPHER_EncryptMulti(HI_HANDLE hCipher, HI_UNF_CIPHER_DATA_S  
*pstDataPkg, HI_U32 u32DataPkgNum);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
*pstDataPkg	待加密的数据包。	输入
u32DataPkgNum	待加密的数据包个数。	输入

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】



- 头文件: hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件: libhi\_cipher.a

#### 【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 每次加密的数据包个数最多不超过 128 个。
- 对于多个包的操作, 每个包都使用 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#) 配置的向量进行运算, 前一个包的向量运算结果不会作用于下一个包的运算, 每个包都是独立运算的。前一次函数调用的结果也不会影响后一次函数调用的运算结果。

#### 【举例】

参考 sample\_multiciphe.c。

## HI\_UNF\_CIPHER\_DecryptMulti

#### 【描述】

进行多个包数据的解密。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_DecryptMulti(HI_HANDLE hCipher, HI_UNF_CIPHER_DATA_S
*pstDataPkg, HI_U32 u32DataPkgNum);
```

#### 【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
*pstDataPkg	待解密的数据包。	输入
u32DataPkgNum	待解密的数据包个数。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件: hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件: libhi\_cipher.a



#### 【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 每次加密的数据包个数最多不超过 128 个。
- 对于多个包的操作，每个包都使用 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#) 配置的向量进行运算，前一个包的向量运算结果不会作用于下一个包的运算，每个包都是独立运算的。前一次函数调用的结果也不会影响后一次函数调用的运算结果。

#### 【举例】

参考 sample\_multiciphe.c。

## HI\_UNF\_CIPHER\_EncryptMultiEx

#### 【描述】

进行多个包数据的加密。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_EncryptMultiEx(HI_HANDLE hCipher,  
HI\_UNF\_CIPHER\_CTRL\_S* pstCtrl, HI\_UNF\_CIPHER\_DATA\_S *pstDataPkg, HI_U32  
u32DataPkgNum);
```

#### 【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
pstCtrl	控制信息指针。	输入
*pstDataPkg	待加密的数据包。	输入
u32DataPkgNum	待加密的数据包个数。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

#### 【注意】



- CIPHER 句柄必须已创建。
- 调用前无需调用 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#)。
- 可多次调用。
- 每次加密的数据包个数最多不超过 128 个。
- 对于多个包的操作，每个包都使用 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#) 的配置进行运算，前一个包的向量运算结果不会作用于下一个包的运算，每个包都是独立运算的。前一次函数调用的结果也不会影响后一次函数调用的运算结果。
- CCM 和 GCM 模式不支持多包加解密。

#### 【举例】

参考 sample\_multiciphe.c。

## HI\_UNF\_CIPHER\_DecryptMultiEx

#### 【描述】

进行多个包数据的解密。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_DecryptMulti(HI_HANDLE hCipher,  
HI_UNF_CIPHER_CTRL_S* pstCtrl, HI_UNF_CIPHER_DATA_S *pstDataPkg, HI_U32  
u32DataPkgNum);
```

#### 【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
pstCtrl	控制信息指针。	输入
*pstDataPkg	待解密的数据包。	输入
u32DataPkgNum	待解密的数据包个数。	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a



#### 【注意】

- CIPHER 句柄必须已创建。
- 调用前无需调用 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#)。
- 可多次调用。
- 每次解密的数据包个数最多不超过 128 个。
- 对于多个包的操作，每个包都使用 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#) 的配置进行运算，前一个包的向量运算结果不会作用于下一个包的运算，每个包都是独立运算的。前一次函数调用的结果也不会影响后一次函数调用的运算结果。
- CCM 和 GCM 模式不支持多包加解密。

#### 【举例】

参考 sample\_multiciphe.c。

## HI\_UNF\_CIPHER\_GetTag

#### 【描述】

CCM/GCM 模式加解密后获取 TAG 值。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_GetTag(HI\_HANDLE hCipher, HI_U8 *pstTag);
```

#### 【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
pstTag	TAG 值	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

#### 【注意】

只有在 CCM、GCM 模式下此接口才有效。



本函数对 Hi3518EV200 无效。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_HashInit

【描述】

初始化 HASH 模块。

【语法】

```
HI_S32 HI_UNF_CIPHER_HashInit(HI_UNF_CIPHER_HASH_ATTRS_S *pstHashAttr,  
HI_HANDLE *pHashHandle);
```

【参数】

参数名称	描述	输入/输出
pstHashAttr	用于计算 hash 的结构体参数	输入
pHashHandle	输出的 hash 句柄	输出

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

如果有其他程序正在使用 HASH 模块，返回失败状态。

【举例】

无。

## HI\_UNF\_CIPHER\_HashUpdate

【描述】

计算 hash 值。

【语法】



```
HI_S32 HI_UNF_CIPHER_HashUpdate(HI_HANDLE hHashHandle, HI_U8  
*pu8InputData, HI_U32 u32InputDataLen);
```

#### 【参数】

参数名称	描述	输入/输出
hHashHandle	Hash 句柄	输入
pu8InputData	输入数据缓冲	输入
u32InputDataLen	输入数据的长度	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

#### 【注意】

- 输入数据块的长度必须是 64 字节对齐，最后一个 block 无此限制。
- Hash 句柄必须已经创建。
- 可以分多次调用，每次计算若干个 block。

#### 【举例】

参考 sample\_hash.c。

## HI\_UNF\_CIPHER\_HashFinal

#### 【描述】

获取 hash 值。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_HashFinal(HI_HANDLE hHashHandle, HI_U8  
*pu8OutputHash);
```

#### 【参数】





参数名称	描述	输入/输出
hHashHandl	Hash 句柄	输入
pu8OutputHash	输出的 hash 值	输出

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

无

【举例】

参考 sample\_hash.c。

## HI\_UNF\_CIPHER\_Hash

【描述】

一次性完成 hash 计算并输出结果。

【语法】

```
HI_S32 HI_UNF_CIPHER_Hash(HI_UNF_CIPHER_HASH_ATTRS_S *pstHashAttr, HI_U32  
u32DataPhyAddr, HI_U32 u32ByteLength, HI_U8 *pu8OutputHash)
```

【参数】

参数名称	描述	输入/输出
pstHashAttr	用于计算 hash 的结构体参数	输入
u32DataPhyAddr	输入数据物理地址	输入
u32ByteLength	输入数据的长度	输入
pu8OutputHash	输出的 hash 值	输出

【返回值】



返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

无。

【举例】

参考 sample\_hash.c。

## HI\_UNF\_CIPHER\_GetRandomNumber

【描述】

生成随机数。

【语法】

```
HI_S32 HI_UNF_CIPHER_GetRandomNumber(HI_U32 *pu32RandomNumber);
```

【参数】

参数名称	描述	输入/输出
pu32RandomNumber	输出的随机数	输出

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

无



### 【举例】

参考 sample\_rng.c。

## HI\_UNF\_CIPHER\_RsaPublicEnc

### 【描述】

使用 RSA 公钥加密一段明文。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaPublicEnc(HI_UNF_CIPHER_RSA_PUB_ENC_S *pstRsaEnc,  
HI_U8 *pu8Input, HI_U32 u32InLen, HI_U8 *pu8Output, HI_U32 *pu32OutLen);
```

### 【参数】

参数名称	描述	输入/输出
pstRsaEnc	公钥加密属性结构体。	输入
pu8Input	待加密的数据。	输入
u32InLen	待加密的数据长度。	输入
pu8Output	加密结果数据。	输出
pu32OutLen	加密结果数据长度	输出

### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

### 【注意】

无

### 【举例】

参考 sample\_rsa\_enc.c。



## HI\_UNF\_CIPHER\_RsaPrivateDec

### 【描述】

使用 RSA 私钥解密一段密文。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaPrivateDec (HI_UNF_CIPHER_RSA_PRI_ENC_S  
*pstRsaDec, HI_U8 *pu8Input, HI_U32 u32InLen, HI_U8 *pu8Output, HI_U32  
*pu32OutLen);
```

### 【参数】

参数名称	描述	输入/输出
pstRsaDec	私钥解密属性结构体。	输入
pu8Input	待解密的数据。	输入
u32InLen	待解密的数据长度。	输入
pu8Output	解密结果数据。	输出
pu32OutLen	解密结果数据长度	输出

### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

### 【注意】

无

### 【举例】

参考 sample\_rsa\_enc.c。

## HI\_UNF\_CIPHER\_RsaPrivateEnc

### 【描述】

使用 RSA 私钥加密一段明文。



### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaPrivateEnc (HI_UNF_CIPHER_RSA_PRI_ENC_S  
*pstRsaEnc, HI_U8 *pu8Input, HI_U32 u32InLen, HI_U8 *pu8Output, HI_U32  
*pu32OutLen);
```

### 【参数】

参数名称	描述	输入/输出
pstRsaEnc	私钥加密属性结构体。	输入
pu8Input	待加密的数据。	输入
u32InLen	待加密的数据长度。	输入
pu8Output	加密结果数据。	输出
pu32OutLen	加密结果数据长度	输出

### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

### 【注意】

无

### 【举例】

参考 sample\_rsa\_enc.c。

## HI\_UNF\_CIPHER\_RsaPublicDec

### 【描述】

使用 RSA 公钥解密一段密文。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaPublicDec (HI_UNF_CIPHER_RSA_PUB_ENC_S  
*pstRsaDec, HI_U8 *pu8Input, HI_U32 u32InLen, HI_U8 *pu8Output, HI_U32  
*pu32OutLen);
```



【参数】

参数名称	描述	输入/输出
pstRsaDec	公钥解密属性结构体。	输入
pu8Input	待解密的数据。	输入
u32InLen	待解密的数据长度。	输入
pu8Output	解密结果数据。	输出
pu32OutLen	解密结果数据长度	输出

【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

【注意】

无

【举例】

参考 sample\_rsa\_enc.c。

## HI\_UNF\_CIPHER\_RsaSign

【描述】

使用 RSA 私钥签名一段文本。

【语法】

```
HI_S32 HI_UNF_CIPHER_RsaSign(HI_UNF_CIPHER_RSA_SIGN_S *pstRsaSign, HI_U8  
*pu8InData, HI_U32 u32InDataLen, HI_U8 *pu8HashData, HI_U8 *pu8OutSign,  
HI_U32 *pu32OutSignLen);
```

【参数】

参数名称	描述	输入/输出
pstRsaSign	签名属性结构体。	输入



参数名称	描述	输入/输出
pu8InData	待签名的数据, 如果 pu8HashData 不为空, 则使用 pu8HashData 进行签名, 该参数将被忽略。	输入
u32InDataLen	待签名的数据长度。	输入
pu8HashData	待签名文本的 HASH 摘要, 如果为空, 则自动计算 pu8InData 的 HASH 摘要进行签名。	输入
pu8OutSign	签名结果数据	输出
pu32OutSignLen	签名结果数据长度	输出

#### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件: hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件: libhi\_cipher.a

#### 【注意】

无

#### 【举例】

参考 sample\_rsa\_sign.c。

## HI\_UNF\_CIPHER\_RsaVerify

#### 【描述】

使用 RSA 公钥签名验证一段文本。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaVerify(HI_UNF_CIPHER_RSA_VERIFY_S  
*pstRsaVerify, HI_U8 *pu8InData, HI_U32 u32InDataLen, HI_U8  
*pu8HashData, HI_U8 *pu8InSign, HI_U32 u32InSignLen);
```

#### 【参数】



参数名称	描述	输入/输出
pstRsaVerify	签名验证属性结构体。	输入
pu8InData	待验证的数据, 如果 pu8HashData 不为空, 则使用 pu8HashData 进行验证, 该参数将被忽略。	输入
u32InDataLen	待验证的数据长度。	输入
pu8HashData	待验证文本的的 HASH 摘要, 如果为空, 则自动计算 pu8InData 的 HASH 摘要进行验证。	输入
pu8InSign	待验证的签名数据	输入
u32InSignLen	待验证的签名数据长度	输入

#### 【返回值】

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件: hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件: libhi\_cipher.a

#### 【注意】

无

#### 【举例】

参考 sample\_rsa\_sign.c。

## HI\_UNF\_CIPHER\_RsaGenKey

#### 【描述】

生成一个 RSA 私钥。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaGenKey(HI_U32 u32NumBits, HI_U32 u32Exponent,  
HI_UNF_CIPHER_RSA_PRI_KEY_S *pstRsaPriKey);
```

#### 【参数】





参数名称	描述	输入/输出
u32NumBits	RSA 密钥 N 的位宽。	输入
u32Exponent	RSA 密钥 E 值。	输入
pstRsaPriKey	RSA 私钥。	输出

**【返回值】**

返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

**【注意】**

无

**【举例】**

无。

## HI\_UNF\_CIPHER\_WriteOTPKey

**【描述】**

烧写 Key 到 OTP 区域。

**【语法】**

```
HI_S32 HI_UNF_CIPHER_WriteOTPKey(HI_U32 u32OptId, const HI_U8 *pu8Key,
HI_U32 u32KeyLen);
```

**【参数】**

参数名称	描述	输入/输出
u32OptId	烧写的 OTP 区域，取值范围[0, 3]	输入
pu8Key	待烧写的 Key 数据	输入
u32KeyLen	Key 数据长度，最大不超过 16，单位是 byte	输入

**【返回值】**



返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

#### 【需求】

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

#### 【注意】

- 调用此接口之前，cipher 必须已初始化。
- 可烧写的 OTP 区域共有 4 个，分别为 0，1，2，3。
- 每个 OTP 区域只能烧写一次，且不能读取。烧写的 Key 长度最大不超过 16 个字节。

#### 【举例】

请参考 cipher sample 目录下的 sample\_anticopy.c。

## HI\_UNF\_CIPHER\_KladEncryptKey

#### 【描述】

使用 KLAD 对透明密钥进行加密。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_KladEncryptKey(HI_UNF_CIPHER_KEY_SRC_E enRootKey,  
HI_UNF_CIPHER_KLAD_TARGET_E enTarget, HI_U8 *pu8CleanKey,  
HI_U8* pu8EcnryptKey, HI_U32 u32KeyLen);
```

#### 【参数】

参数名称	描述	输入/输出
enRootKey	KLAD 根密钥选择，只能选择 EFUSE Key。	输入
enTarget	使用该密钥的模块。	输入
pu8CleanKey	透明密钥。	输入
pu8EcnryptKey	加密密钥。	输出
u32KeyLen	密钥的长度，必须是 16 整数倍。	输入

#### 【返回值】



返回值	描述
0	成功。
非 0	参见 <a href="#">错误码</a> 。

**【需求】**

- 头文件：hi\_error\_mpi.h、hi\_type.h、hi\_unf\_cipher.h
- 库文件：libhi\_cipher.a

**【注意】**

该函数仅对 Hi3519V101 有效。

**【举例】**

请参考 cipher sample 目录下的 sample\_rsa\_enc.c



# 3 数据类型

相关数据类型、数据结构定义如下：

- [HI\\_HANDLE](#)：定义 CIPHER 的句柄类型。
- [HI\\_UNF\\_CIPHER\\_WORK\\_MODE\\_E](#)：定义 CIPHER 工作模式。
- [HI\\_UNF\\_CIPHER\\_ALG\\_E](#)：定义 CIPHER 加密算法。
- [HI\\_UNF\\_CIPHER\\_KEY\\_LENGTH\\_E](#)：定义 CIPHER 密钥长度。
- [HI\\_UNF\\_CIPHER\\_BIT\\_WIDTH\\_E](#)：定义 CIPHER 加密位宽。
- [HI\\_UNF\\_CIPHER\\_KEY\\_SRC\\_E](#)：定义 CIPHER key 的来源。
- [HI\\_UNF\\_CIPHER\\_CCM\\_INFO\\_S](#)：定义 CIPHER CCM 模式的信息结构体。
- [HI\\_UNF\\_CIPHER\\_GCM\\_INFO\\_S](#)：定义 CIPHER GCM 模式的信息结构体。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_S](#)：定义 CIPHER 控制信息结构体。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_CHANGE\\_FLAG\\_S](#)：定义 CIPHER 控制参数变更标志。
- [HI\\_UNF\\_CIPHER\\_DATA\\_S](#)：定义 CIPHER 加解密数据。
- [HI\\_UNF\\_CIPHER\\_HASH\\_TYPE\\_E](#)：定义 CIPHER 哈希算法类型。
- [HI\\_UNF\\_CIPHER\\_HASH\\_ATTS\\_S](#)：定义 CIPHER 哈希算法初始化输入结构体。
- [HI\\_UNF\\_CIPHER\\_RSA\\_ENC\\_SCHEME\\_E](#)：定义 RSA 算法数据加密填充方式。
- [HI\\_UNF\\_CIPHER\\_RSA\\_SIGN\\_SCHEME\\_E](#)：定义 RSA 数据签名算法。
- [HI\\_UNF\\_CIPHER\\_RSA\\_PUB\\_KEY\\_S](#)：定义 RSA 公钥结构体。
- [HI\\_UNF\\_CIPHER\\_RSA\\_PRI\\_KEY\\_S](#)：定义 RSA 私钥结构体。
- [HI\\_UNF\\_CIPHER\\_RSA\\_PUB\\_ENC\\_S](#)：定义 RSA 公钥加解密算法参数结构体。
- [HI\\_UNF\\_CIPHER\\_RSA\\_PRI\\_ENC\\_S](#)：定义 RSA 私钥解密算法参数输入结构体。
- [HI\\_UNF\\_CIPHER\\_RSA\\_SIGN\\_S](#)：定义 RSA 签名算法参数输入结构体。
- [HI\\_UNF\\_CIPHER\\_RSA\\_VERIFY\\_S](#)：定义 RSA 签名验证算法参数输入结构体。
- [HI\\_UNF\\_CIPHER\\_KLAD\\_TARGET\\_E](#)：定义 Klad 目标选择

## HI\_HANDLE

### 【说明】

定义 CIPHER 的句柄类型。



【定义】

```
typedef HI_U32 HI_HANDLE;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_WORK\_MODE\_E

【说明】

定义 CIPHER 工作模式。

【定义】

```
typedef enum hiHI_UNF_CIPHER_WORK_MODE_E
{
    HI_UNF_CIPHER_WORK_MODE_ECB,
    HI_UNF_CIPHER_WORK_MODE_CBC,
    HI_UNF_CIPHER_WORK_MODE_CFB,
    HI_UNF_CIPHER_WORK_MODE_OFB,
    HI_UNF_CIPHER_WORK_MODE_CTR,
    HI_UNF_CIPHER_WORK_MODE_CCM,
    HI_UNF_CIPHER_WORK_MODE_GCM,
    HI_UNF_CIPHER_WORK_MODE_CBC_CTS,
    HI_UNF_CIPHER_WORK_MODE_BUTT    = 0xffffffff
}HI_UNF_CIPHER_WORK_MODE_E;
```

【成员】

成员名称	描述
HI_UNF_CIPHER_WORK_MODE_ECB	ECB（Electronic CodeBook）模式
HI_UNF_CIPHER_WORK_MODE_CBC	CBC（Cipher Block Chaining）模式
HI_UNF_CIPHER_WORK_MODE_CFB	CFB（Cipher FeedBack）模式
HI_UNF_CIPHER_WORK_MODE_OFB	OFB（Output FeedBack）模式
HI_UNF_CIPHER_WORK_MODE_CTR	CTR（Counter）模式
HI_UNF_CIPHER_WORK_MODE_CCM	CCM（Counter with Cipher Block Chaining-Message Authentication）模式



成员名称	描述
HI_UNF_CIPHER_WORK_MODE_GCM	GCM（Galois/Counter Mode）模式
HI_UNF_CIPHER_WORK_MODE_CBC_CTS	CBC CTS（Community Tissue Services）模式
HI_UNF_CIPHER_WORK_MODE_BUTT	无效模式

**【注意事项】**

当前版本不支持 CBC-CTS 模式。

**【相关数据类型及接口】**

无。

## HI\_UNF\_CIPHER\_ALG\_E

**【说明】**

定义 CIPHER 加密算法。

**【定义】**

```
typedef enum hiHI_UNF_CIPHER_ALG_E
{
    HI_UNF_CIPHER_ALG_DES    = 0x0,
    HI_UNF_CIPHER_ALG_3DES   = 0x1,
    HI_UNF_CIPHER_ALG_AES    = 0x2,
    HI_UNF_CIPHER_ALG_BUTT   = 0x3
}HI_UNF_CIPHER_ALG_E;
```

**【成员】**

成员名称	描述
HI_UNF_CIPHER_ALG_DES	DES 算法
HI_UNF_CIPHER_ALG_3DES	3DES 算法
HI_UNF_CIPHER_ALG_AES	AES 算法

**【注意事项】**

无。

**【相关数据类型及接口】**

无。



## HI\_UNF\_CIPHER\_KEY\_LENGTH\_E

### 【说明】

定义 CIPHER 密钥长度。

### 【定义】

```
typedef enum hiHI_UNF_CIPHER_KEY_LENGTH_E
{
    HI_UNF_CIPHER_KEY_AES_128BIT = 0x0,
    HI_UNF_CIPHER_KEY_AES_192BIT = 0x1,
    HI_UNF_CIPHER_KEY_AES_256BIT = 0x2,
    HI_UNF_CIPHER_KEY_DES_3KEY = 0x2,
    HI_UNF_CIPHER_KEY_DES_2KEY = 0x3,
}HI_UNF_CIPHER_KEY_LENGTH_E;
```

### 【成员】

成员名称	描述
HI_UNF_CIPHER_KEY_AES_128BIT	AES 运算方式下采用 128bit 密钥长度
HI_UNF_CIPHER_KEY_AES_192BIT	AES 运算方式下采用 192bit 密钥长度
HI_UNF_CIPHER_KEY_AES_256BIT	AES 运算方式下采用 256bit 密钥长度
HI_UNF_CIPHER_KEY_DES_3KEY	3DES 运算方式下采用 3 个 key
HI_UNF_CIPHER_KEY_DES_2KEY	3DES 运算方式下采用 2 个 key

### 【注意事项】

- AES 的密钥长度可以为 128bit, 192bit 或 256bit。
- 3DES 算法的密钥长度可以为 2 个或 3 个 key，一个 key 指 DES 加密所用的密钥，它的长度为 64bit。
- DES 算法该项无效。

### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_BIT\_WIDTH\_E

### 【说明】

定义 CIPHER 加密位宽。

### 【定义】

```
typedef enum hiHI_UNF_CIPHER_BIT_WIDTH_E
{
```



```
HI_UNF_CIPHER_BIT_WIDTH_64BIT = 0x0,  
HI_UNF_CIPHER_BIT_WIDTH_8BIT  = 0x1,  
HI_UNF_CIPHER_BIT_WIDTH_1BIT  = 0x2,  
HI_UNF_CIPHER_BIT_WIDTH_128BIT = 0x3,  
}HI_UNF_CIPHER_BIT_WIDTH_E;
```

#### 【成员】

成员名称	描述
HI_UNF_CIPHER_BIT_WIDTH_64BIT	64bit 位宽
HI_UNF_CIPHER_BIT_WIDTH_8BIT	8bit 位宽
HI_UNF_CIPHER_BIT_WIDTH_1BIT	1bit 位宽
HI_UNF_CIPHER_BIT_WIDTH_128BIT	128bit 位宽

#### 【注意事项】

无。

#### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_KEY\_SRC\_E

#### 【说明】

定义 CIPHER key 的来源。

#### 【定义】

```
typedef enum hiHI_UNF_CIPHER_KEY_SRC_E  
{  
    HI_UNF_CIPHER_KEY_SRC_USER = 0x0,  
    HI_UNF_CIPHER_KEY_SRC_EFUSE_0,  
    HI_UNF_CIPHER_KEY_SRC_EFUSE_1,  
    HI_UNF_CIPHER_KEY_SRC_EFUSE_2,  
    HI_UNF_CIPHER_KEY_SRC_EFUSE_3,  
    HI_UNF_CIPHER_KEY_SRC_KLAD_1,  
    HI_UNF_CIPHER_KEY_SRC_KLAD_2,  
    HI_UNF_CIPHER_KEY_SRC_KLAD_3,  
    HI_UNF_CIPHER_KEY_SRC_BUTT  
} HI_UNF_CIPHER_KEY_SRC_E;
```

#### 【成员】





成员名称	描述
HI_UNF_CIPHER_KEY_SRC_USER	用户配置的 Key
HI_UNF_CIPHER_KEY_SRC_EFUSE_0	Efuse 的第 0 组 Key
HI_UNF_CIPHER_KEY_SRC_EFUSE_1	Efuse 的第 1 组 Key
HI_UNF_CIPHER_KEY_SRC_EFUSE_2	Efuse 的第 2 组 Key
HI_UNF_CIPHER_KEY_SRC_EFUSE_3	Efuse 的第 3 组 Key
HI_UNF_CIPHER_KEY_SRC_KLAD_1	KLAD 的第 1 组 Key, 其 Root Key 为 Efuse 的第 1 组 Key
HI_UNF_CIPHER_KEY_SRC_KLAD_2	KLAD 的第 2 组 Key, 其 Root Key 为 Efuse 的第 2 组 Key
HI_UNF_CIPHER_KEY_SRC_KLAD_3	KLAD 的第 3 组 Key, 其 Root Key 为 Efuse 的第 3 组 Key
HI_UNF_CIPHER_KEY_SRC_BUTT	无效类型

【注意事项】

- Hi3518EV200 和 Hi3519V100 数据加解密支持 EFUSE Key，不支持 KLAD Key。
- Hi3519V101 数据加解密支持 KLAD Key，不支持 EFUSE Key。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_CCM\_INFO\_S

【说明】

定义 CIPHER CCM 模式的信息结构体。

【定义】

```
typedef struct hiUNF_CIPHER_CCM_INFO_S
{
    HI_U8  u8Nonce[16];
    HI_U8  *pu8Aad;
    HI_U32 u32ALen;
    HI_U32 u32MLen;
    HI_U8  u8NLen;
    HI_U8  u8TLen;
    HI_U8  u8Reserve[2];
}HI_UNF_CIPHER_CCM_INFO_S;
```



【成员】

成员名称	描述
u8Nonce	CCM 模式下的 NONCE 数据
pu8Aad	CCM 模式下的额外数据 A 的指针
u32ALen	CCM 模式下的额外数据的长度
u32MLen	消息数据的长度
u8NLen	NONCE 数据的长度
u8TLen	CCM 模式下的校验值 TAG 的长度
u8Reserve	保留字段

【注意事项】

- 额外数据 A 只参与校验值的生成，不参与数据的加解密，即其仅影响 TAG 的值，不会影响加解密结果。
- HI3518EV200 不支持本结构体。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_GCM\_INFO\_S

【说明】

定义 CIPHER GCM 模式的信息结构体。

【定义】

```
typedef struct hiUNF_CIPHER_GCM_INFO_S
{
    HI_U8 *pu8Aad;
    HI_U32 u32ALen;
    HI_U32 u32MLen;
    HI_U32 u32IVLen;
}HI_UNF_CIPHER_GCM_INFO_S;
```

【成员】

成员名称	描述
pu8Aad	GCM 模式下的额外数据 A 的指针
u32ALen	GCM 模式下的额外数据的长度
u32MLen	消息数据的长度



成员名称	描述
u32IVLen	GCM 模式下的 IV 的长度

【注意事项】

- 额外数据 A 只参与校验值的生成，不参与数据的加解密，即其仅影响 TAG 的值，不会影响加解密结果。
- u32IVLen 的值必须大于或等于 1，且小于或等于 16。
- Hi3518EV200 不支持本结构体。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_CTRL\_S

【说明】

定义 CIPHER 控制信息结构体。

【定义】

```
typedef struct hiHI_UNF_CIPHER_CTRL_S
{
    HI_U32                u32Key[8];
    HI_U32                u32IV[4];
    HI_UNF_CIPHER_ALG_E   enAlg;
    HI_UNF_CIPHER_BIT_WIDTH_E enBitWidth;
    HI_UNF_CIPHER_WORK_MODE_E enWorkMode;
    HI_UNF_CIPHER_KEY_LENGTH_E enKeyLen;
    HI_UNF_CIPHER_CTRL_CHANGE_FLAG_S stChangeFlags;
    HI_UNF_CIPHER_KEY_SRC_E enKeySrc;
    union
    {
        HI_UNF_CIPHER_CCM_INFO_S stCCM;
        HI_UNF_CIPHER_GCM_INFO_S stGCM;
    } unModeInfo;
} HI_UNF_CIPHER_CTRL_S;
```

【成员】

成员名称	描述
u32Key[8]	密钥
u32IV[4]	初始向量
enAlg	加密算法



成员名称	描述
enBitWidth	加密或解密的位宽
enWorkMode	工作模式
enKeyLen	密钥长度
stChangeFlags	更新标志位，表示 IV 等是否需要更新
enKeySrc	密钥的来源
unModeInfo	模式信息

【注意事项】

- ECB 模式下不需要初始向量。
- 结构体中的成员 unModeInfo 对 HI3518EV200 无效。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_CTRL\_CHANGE\_FLAG\_S

【说明】

定义 CIPHER 控制参数变更标志。

【定义】

```
typedef struct hiUNF_CIPHER_CTRL_CHANGE_FLAG_S
{
    HI_U32    bit1IV:    1;
    HI_U32    bitsResv:  31;
} HI_UNF_CIPHER_CTRL_CHANGE_FLAG_S;
```

【成员】

成员名称	描述
bit1IV	向量变更
bitsResv	保留

【注意事项】

无。

【相关数据类型及接口】

无。



## HI\_UNF\_CIPHER\_DATA\_S

### 【说明】

定义 CIPHER 加解密数据。

### 【定义】

```
typedef struct hiHI_UNF_CIPHER_DATA_S
{
    HI_U32 u32SrcPhyAddr;
    HI_U32 u32DestPhyAddr;
    HI_U32 u32ByteLength;
} HI_UNF_CIPHER_DATA_S;
```

### 【成员】

成员名称	描述
u32SrcPhyAddr	源数据物理地址
u32DestPhyAddr	目的数据物理地址
u32ByteLength	加解密数据长度

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_HASH\_TYPE\_E

### 【说明】

定义 CIPHER 哈希算法类型。

### 【定义】

```
typedef enum hiHI_UNF_CIPHER_HASH_TYPE_E
{
    HI_UNF_CIPHER_HASH_TYPE_SHA1,
    HI_UNF_CIPHER_HASH_TYPE_SHA256,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA1,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA256,
    HI_UNF_CIPHER_HASH_TYPE_IRDETO_CBCMAC,
    HI_UNF_CIPHER_HASH_TYPE_BUTT,
} HI_UNF_CIPHER_HASH_TYPE_E;
```

### 【成员】



成员名称	描述
HI_UNF_CIPHER_HASH_TYPE_SHA1	SHA1 哈希算法
HI_UNF_CIPHER_HASH_TYPE_SHA256	SHA256 哈希算法
HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA1	HMAC_SHA1 哈希算法
HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA256	HMAC_SHA256 哈希算法
HI_UNF_CIPHER_HASH_TYPE_BUTT	-

【注意事项】

无。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_HASH\_ATTS\_S

【说明】

定义 CIPHER 哈希算法初始化输入结构体。

【定义】

```
typedef struct
{
    HI_U8 *pu8HMACKey;
    HI_U32 u32HMACKeyLen;
    HI_UNF_CIPHER_HASH_TYPE_E eShaType;
}HI_UNF_CIPHER_HASH_ATTS_S;
```

【成员】

成员名称	描述
pu8HMACKey	HAMC 密钥
u32HMACKeyLen	HAMC 密钥长度
eShaType	选择哈希算法类型

【注意事项】

无。

【相关数据类型及接口】

无。



## HI\_UNF\_CIPHER\_RSA\_ENC\_SCHEME\_E

### 【说明】

定义 RSA 算法数据加密填充方式。

### 【定义】

```
typedef enum hiHI_UNF_CIPHER_RSA_ENC_SCHEME_E
{
    HI_UNF_CIPHER_RSA_ENC_SCHEME_NO_PADDING,
    HI_UNF_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_0,
    HI_UNF_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_1,
    HI_UNF_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_2,
    HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA1,
    HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256,
    HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5,
    HI_UNF_CIPHER_RSA_ENC_SCHEME_BUTT,
}HI_UNF_CIPHER_RSA_ENC_SCHEME_E;
```

### 【成员】

成员名称	描述
HI_UNF_CIPHER_RSA_ENC_SCHEME_NO_PADDING	不填充
HI_UNF_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_0,	PKCS#1 的 block type 0 填充方式
HI_UNF_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_1	PKCS#1 的 block type 1 填充方式
HI_UNF_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_2	PKCS#1 的 block type 2 填充方式
HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA1	PKCS#1 的 RSAES-OAEP-SHA1 填充方式
HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256	PKCS#1 的 RSAES-OAEP-SHA256 填充方式
HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5	PKCS#1 的 PKCS1_V1_5 填充方式
HI_UNF_CIPHER_RSA_ENC_SCHEME_BUTT	-

### 【注意事项】

无。



【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_RSA\_SIGN\_SCHEME\_E

【说明】

定义 RSA 数据签名策略。

【定义】

```
typedef enum hiHI_UNF_CIPHER_RSA_SIGN_SCHEME_E
{
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA1 = 0x100,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_BUTT,
}HI_UNF_CIPHER_RSA_SIGN_SCHEME_E;
```

【成员】

成员名称	描述
HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA1	PKCS#1 RSASSA_PKCS1_V15_SHA1 签名算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256	PKCS#1 RSASSA_PKCS1_V15_SHA256 签名算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1	PKCS#1 RSASSA_PKCS1_PSS_SHA1 签名算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256	PKCS#1 RSASSA_PKCS1_PSS_SHA256 签名算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_BUTT	-

【注意事项】

无。

【相关数据类型及接口】

无。





## HI\_UNF\_CIPHER\_RSA\_PUB\_KEY\_S

### 【说明】

定义 RSA 公钥结构体。

### 【定义】

```
typedef struct
{
    HI_U8  *pu8N;
    HI_U8  *pu8E;
    HI_U16 u16NLen;
    HI_U16 u16ELen;
}HI_UNF_CIPHER_RSA_PUB_KEY_S;
```

### 【成员】

成员名称	描述
pu8N	指向 RSA 公钥 N 的指针
pu8E	指向 RSA 公钥 E 的指针
u16NLen	RSA 公钥 N 的长度
u16ELen	RSA 公钥 E 的长度

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_RSA\_PRI\_KEY\_S

### 【说明】

定义 RSA 私钥结构体。

### 【定义】

```
typedef struct
{
    HI_U8  *pu8N;
    HI_U8  *pu8E;
    HI_U8  *pu8D;
    HI_U8  *pu8P;
    HI_U8  *pu8Q;
    HI_U8  *pu8DP;
```



```
HI_U8 *pu8DQ;  
HI_U8 *pu8QP;  
HI_U16 u16NLen;  
HI_U16 u16ELen;  
HI_U16 u16DLen;  
HI_U16 u16PLen;  
HI_U16 u16QLen;  
HI_U16 u16DPLen;  
HI_U16 u16DQLen;  
HI_U16 u16QPLen;  
}HI_UNF_CIPHER_RSA_PRI_KEY_S;
```

#### 【成员】

成员名称	描述
pu8N	指向 RSA 公钥 N 的指针
pu8E	指向 RSA 公钥 E 的指针
pu8D	指向 RSA 公钥 D 的指针
pu8P	指向 RSA 公钥 P 的指针
pu8Q	指向 RSA 公钥 Q 的指针
pu8DP	指向 RSA 公钥 DP 的指针
pu8DQ	指向 RSA 公钥 DQ 的指针
pu8QP	指向 RSA 公钥 QP 的指针
u16NLen	RSA 公钥 N 的长度
u16ELen	RSA 公钥 E 的长度
u16DLen	RSA 公钥 D 的长度
u16PLen	RSA 公钥 P 的长度
u16QLen	RSA 公钥 Q 的长度
u16DPLen	RSA 公钥 DP 的长度
u16DQLen	RSA 公钥 DQ 的长度
u16QPLen	RSA 公钥 QP 的长度

#### 【注意事项】

无。

#### 【相关数据类型及接口】



无。

## HI\_UNF\_CIPHER\_RSA\_PUB\_ENC\_S

### 【说明】

定义 RSA 公钥加解密算法参数结构体。

### 【定义】

```
typedef struct
{
    HI_UNF_CIPHER_RSA_ENC_SCHEME_E enScheme;
    HI_UNF_CIPHER_RSA_PUB_KEY_S stPubKey;
}HI_UNF_CIPHER_RSA_PUB_ENC_S;
```

### 【成员】

成员名称	描述
enScheme	RSA 数据加解密算法策略
stPubKey	RSA 公钥结构体

### 【注意事项】

无。

### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_RSA\_PRI\_ENC\_S

### 【说明】

定义 RSA 私钥解密算法参数结构体。

### 【定义】

```
typedef struct
{
    HI_UNF_CIPHER_RSA_ENC_SCHEME_E enScheme;
    HI_UNF_CIPHER_RSA_PRI_KEY_S stPriKey;
    HI_UNF_CIPHER_KEY_SRC_E enKeySrc;
}HI_UNF_CIPHER_RSA_PRI_ENC_S;
```

### 【成员】

成员名称	描述
enScheme	RSA 数据加解密算法策略



成员名称	描述
stPriKey	RSA 私钥结构体
enKeySrc	RSA 使用的私钥来源选择

【注意事项】

enKeySrc 只对 Hi3519V101 有效，只能选择 CPU Key 或 Klad Key。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_RSA\_SIGN\_S

【说明】

定义 RSA 签名算法参数输入结构体。

【定义】

```
typedef struct
{
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_E enScheme;
    HI_UNF_CIPHER_RSA_PRI_KEY_S stPriKey;
    HI_UNF_CIPHER_KEY_SRC_E enKeySrc;
}HI_UNF_CIPHER_RSA_SIGN_S;
```

【成员】

成员名称	描述
enScheme	RSA 数据加解密算法策略
stPriKey	RSA 私钥结构体
enKeySrc	RSA 使用的私钥来源选择

【注意事项】

enKeySrc 只对 Hi3519V101 有效，只能选择 CPU Key 或 Klad Key。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_RSA\_VERIFY\_S

【说明】

定义 RSA 签名验证算法参数输入结构体。



#### 【定义】

```
typedef struct
{
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_E enScheme;
    HI_UNF_CIPHER_RSA_PUB_KEY_S stPubKey;
} HI_UNF_CIPHER_RSA_VERIFY_S;
```

#### 【成员】

成员名称	描述
enScheme	RSA 数据加解密算法策略
stPubKey	RSA 公钥结构体

#### 【注意事项】

无。

#### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_KLAD\_TARGET\_E

#### 【说明】

定义 Klad 产生的 Key 送达的目标选择。

#### 【定义】

```
typedef struct
{
    HI_UNF_CIPHER_KLAD_TARGET_AES,
    HI_UNF_CIPHER_KLAD_TARGET_RSA,
    HI_UNF_CIPHER_KLAD_TARGET_BUTT,
} HI_UNF_CIPHER_KLAD_TARGET_E;
```

#### 【成员】

成员名称	描述
HI_UNF_CIPHER_KLAD_TARGET_AES	Klad 产生的 Key 送到 AES
HI_UNF_CIPHER_KLAD_TARGET_RSA	Klad 产生的 Key 送到 RSA
HI_UNF_CIPHER_KLAD_TARGET_BUTT	无数参数

#### 【注意事项】



Hi3516CV300 和 Hi3519V101 支持 KLAD，其它芯片不支持。

**【相关数据类型及接口】**

无。



# 4 错误码

CIPHER 提供的错误码如表 4-1 所示。

表4-1 CIPHER 模块的错误码

错误代码	宏定义	描述
0x804D0001	HI_ERR_CIPHER_NOT_INIT	设备未初始化
0x804D0002	HI_ERR_CIPHER_INVALID_HANDLE	Handle 号无效
0x804D0003	HI_ERR_CIPHER_INVALID_POINT	参数中有空指针
0x804D0004	HI_ERR_CIPHER_INVALID_PARA	无效参数
0x804D0005	HI_ERR_CIPHER_FAILED_INIT	初始化失败
0x804D0006	HI_ERR_CIPHER_FAILED_GETHANDLE	获取 handle 失败
-1	HI_FAILURE	操作失败



# 5 Proc 调试信息

## 5.1 CIPHER 状态

### 【调试信息】

```
# cat /proc/driver/hi_cipher
-----CIPHER STATUS-----
Chnid  Status  Decrypt  Alg  Mode  KeyLen
0      close   0        DES  ECB   000
1      close   0        DES  ECB   000
2      close   0        DES  ECB   000
3      close   0        DES  ECB   000
4      close   0        DES  ECB   000
5      close   0        DES  ECB   000
6      close   0        DES  ECB   000
7      close   0        DES  ECB   000

Phy-Addr in/out  KeyFrom  INT-RAW in/out  INT-EN in/out  INT_OCNTC
fe0c1014/fe0c0000  SW       0/0      0/0      00000000
84001800/84002000  SW       0/0      0/0      00000000
84003000/84003800  SW       0/0      0/0      00000000
84004800/84005000  SW       0/0      0/0      00000000
84006000/84006800  SW       0/0      0/0      00000000
84007800/84008000  SW       0/0      0/0      00000000
84009000/84009800  SW       0/0      0/0      00000000
8400a800/8400b000  SW       0/0      0/0      00000000
```

### 【调试信息分析】

记录当前 CIPHER 各个通道的配置信息。

### 【参数说明】





参数		描述
CIPHER 基本属性	Chnid	通道号
	Status	打开/关闭
	Decrypt	加密/解密
	Alg	算法, AES/DES/3DES 等
	Mode	模式, ECB/CBC/CFB/CTR 等
	KeyLen	密钥长度, 128/192/256 等
	Phy-Addr	输入/输出物理地址
	KeyFrom	密钥来源, CPU 或 EFUSE
	INT-RAW	是否有原始中断
	INT-EN	是否中断使能
	INT_OCNTCFG	是否有产生中断