



**CIPHER**

## **API 参考**

文档版本 01

发布日期 2019-05-15

Cogobuy Only For ShenZhen FuShi ChanJing Industrial Technology Co., Ltd.

版权所有 © 上海海思技术有限公司 2019。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海思技术有限公司的商标。

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。

## 上海海思技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<http://www.hisilicon.com/cn/>

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 AES、DES、3DES 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，部分芯片的 AES 算法还可支持 CCM、GCM，摘要算法也可支持 SHA224/384/512 算法类型。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，Hi3559CV100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3516DV300、Hi3559V200、Hi3556V200、Hi3516AV300 与 Hi3516CV500 内容一致。
- 未有特殊说明，Hi3516EV200、Hi3516EV300、Hi3518EV300 与 Hi3516DV200 内容一致。

## 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
Hi3559A	V100ES
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3516C	V500
Hi3516D	V300
Hi3559	V200
Hi3556	V200
Hi3516A	V300
Hi3516E	V200
Hi3516E	V300
Hi3518E	V300



产品名称	产品版本
Hi3516D	V200

## 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
	用于警示紧急的危险情形，若不可避免，将会导致人员死亡或严重的人身伤害。
	用于警示潜在的危险情形，若不可避免，可能会导致人员死亡或严重的人身伤害。
	用于警示潜在的危险情形，若不可避免，可能会导致中度或轻微的人身伤害。
	用于传递设备或环境安全警示信息，若不可避免，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。 不带安全警示符号的“注意”不涉及人身伤害。
	用于突出重要/关键信息、最佳实践和小窍门等。 “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 01 (2019-05-15)

第 1 次正式发布。



1.1 小节，涉及更新。

## 文档版本 00B09 (2019-03-30)

添加 Hi3516DV200 相关内容。

5.1 小节，涉及修改

## 文档版本 00B08 (2019-03-01)

添加 Hi3516AV300 相关内容

1.1 小节涉及修改

第 4 章，表 4-1 涉及修改

## 文档版本 00B07 (2018-11-23)

添加 Hi3516EV200、Hi3516EV300、Hi3518EV300 相关内容。

## 文档版本 00B06 (2018-10-30)

第 6 次临时版本发布

第 2 章，HI\_UNF\_CIPHER\_CreateHandle、HI\_UNF\_CIPHER\_RsaPrivateDecrypt、HI\_UNF\_CIPHER\_RsaPrivateEncrypt 涉及修改

第 3 章，HI\_UNF\_CIPHER\_CTRL\_S 的【成员】涉及修改

## 文档版本 00B05 (2018-09-29)

第 5 次临时版本发布

## 文档版本 00B04 (2018-09-04)

添加 Hi3516CV500/Hi3516DV300 相关内容

## 文档版本 00B03 (2018-04-11)

添加 Hi3519AV100 相关内容

第 4 章，表 4-1 涉及修改

## 文档版本 00B02 (2018-01-15)

添加 Hi3559AV100 和 Hi3559CV100 相关内容

## 文档版本 00B01 (2017-05-27)

第 1 次临时发布。



## 目 录

前 言.....	i
1 概述.....	7
1.1 概述.....	7
1.2 使用流程.....	9
1.2.1 单包数据加解密.....	9
1.2.2 多包数据加解密.....	11
1.2.3 HASH 计算.....	12
1.2.4 HMAC 计算.....	13
1.2.5 产生随机数.....	14
1.2.6 RSA 加解密操作步骤.....	14
1.2.7 RSA 签名及验签操作步骤.....	15
1.2.8 CCM/GCM 加解密操作步骤.....	16
2 API 参考.....	18
3 数据类型.....	42
4 错误码.....	67
5 Proc 调试信息.....	70
5.1 CIPHER 状态.....	70



## 插图目录

图 1-1 Cipher 应用场景 1，每次调用都需要更新 IV .....	10
图 1-2 Cipher 应用场景 2，只在第一次调用时配置 IV .....	11

Cogobuy Only For ShenZhen FuShi ChanJing Industrial Technology Co., Ltd.



## 表格目录

表 4-1 CIPHER 模块的错误码 .....	67
---------------------------	----

Cogobuy Only For ShenZhen FuShi ChanJing Industrial Technology Co., Ltd.





# 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 AES 和 DES/3DES 等对称加解密算法，RSA 不对称加解密算法、随机数生成，以及支持 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。各功能划分如下：

### 对称加解密算法

- AES：支持 ECB/CBC/CFB/OFB/CTR/CCM/GCM 等工作模式，其中 CCM/GCM 模式 Hi3516CV500、Hi3516DV300、Hi3516AV300、Hi3516EV200、Hi3516EV300、Hi3518EV300、Hi3516DV200 不支持，其它芯片均支持，该模式下，加解密结束后需获取一次 TAG 值。
- DES/3DES：ECB/CBC/CFB/OFB，其中 CFB 和 OFB 模式支持的位宽可为 1/8/64。Hi3516EV200、Hi3516EV300、Hi3518EV300、Hi3516DV200 不支持 DES/3DES 算法。

以上算法除了 CTR/CCM/GCM，其它算法、模式的数据长度必须按块大小对齐；CCM/GCM 的 N、A 需要靠软件按标准把各个字段封装成块大小对齐的数据块；各种工作模式支持一次实现多个分组的加解密运算，也支持一次实现单个分组的加解密运算，最多可以申请 7 个通道。Hi3516EV200、Hi3516EV300、Hi3518EV300、Hi3516DV200 只支持最多申请 2 通道。

DES/3DES 算法安全性低，即使是 3DES ( $K1 \neq K2 \neq K3$ ) 算法的安全性也不及 AES (位宽 128bit 及以上) 算法，推荐使用更安全的 AES 算法。

### 不对称加解密算法

RSA：支持密钥位宽 1024/2048/3072/4096，Hi3516CV500、Hi3516DV300、Hi3516AV300、Hi3516EV200、Hi3516EV300、Hi3518EV300、Hi3516DV200 不支持 3072 的密钥位宽，其它芯片均支持。

RSA 密钥位宽 1024 及以下算法为业界已知不安全算法，应禁止使用。



## 随机数生成

RNG: 支持 DRGB, 以更高速率获取随机数。

## 摘要算法

HASH: 支持 SHA1/SHA224/SHA256/SHA384/SHA512/SM3; 支持 HMAC1/HMAC224/HMAC256/HMAC384/HMAC512; 支持软件多通道, 最多可以申请 8 个通道。

其中, **Hi3516EV200、Hi3516EV300、Hi3518EV300、Hi3516DV200 不支持 SHA384、SHA512、HMAC384、HMAC512。**

SHA1 算法安全性较低, 不能应用在参与生成“数字签名”的场景, 推荐使用 SHA2 (256 位及以上) 算法。

以上各功能模块涉及的算法及工作模式符合以下标准:

- AES 算法的实现符合 FIPS 197 标准, 其支持的工作模式符合以下标准:
  - ECB、CBC、1/8/128-CFB、128-OFB、CTR 几种工作模式符合 NIST special800-38a 标准
  - CCM 工作模式符合 NIST special800-38e 标准
  - GCM 工作模式符合 NIST special800-38d 标准
- DES/3DES 算法的实现符合 FIPS46-3 标准, 工作模式符合的标准如下:
  - 支持 ECB、CBC、1/8/64-CFB、1/8/64-OFB 几种工作模式, 符合 FIPS-81 标准
- RSA 支持公钥加密私钥解密、私钥加密公钥解密、签名及验签等功能, 各种模式的数据填充方式符合 PKCS#1 标准
  - RSA 的加解密模式包括 NO\_PADDING、BLOCK\_YTPE\_0、BLOCK\_YTPE\_1、BLOCK\_YTPE\_2、RSAES\_OAEP\_SHA1、RSAES\_OAEP\_SHA224、RSAES\_OAEP\_SHA256、RSAES\_OAEP\_SHA384、RSAES\_OAEP\_SHA512、RSAES\_PKCS1\_V1\_5 等
  - RSA 的签名及验签模式包括 RSASSA\_PKCS1\_V15\_SHA1、RSASSA\_PKCS1\_V15\_SHA224、RSASSA\_PKCS1\_V15\_SHA256、RSASSA\_PKCS1\_V15\_SHA384、RSASSA\_PKCS1\_V15\_SHA512、RSASSA\_PKCS1\_PSS\_SHA1、RSASSA\_PKCS1\_PSS\_SHA224、RSASSA\_PKCS1\_PSS\_SHA256、RSASSA\_PKCS1\_PSS\_SHA384、RSASSA\_PKCS1\_PSS\_SHA512 等



### 说明

- 对称加解密算法的工作模式主要有:

ECB (Electronic codebook, 电子密码本模式)、CBC (Cipher-block chaining, 密码分组链接模式)、CFB (Cipher feedback, 密文反馈模式)、OFB (Output feedback, 输出反馈模式)、CTR (Counter mode, 计数器模式)、CCM (counter with CBC-MAC, CTR 加密模式和消息认证码 CMAC 算法的混合)、GCM (Galois/Counter Mode, 伽罗华域/计数器模式), 主要由工作在计数器模式下的分组密码和在伽罗华域 GF (2<sup>128</sup>) 上的哈希运算组成。CCM 和 GCM 在加解密的同时生成 CMAC 检验值, 解密时的 CMAC 要和解密时的 CMAC 一样才说明解密是正确的, 常用在需要同时加密和认证的领域, 欲了解算法的详细内容, 请参考相关文献。



- 块密码学中，将需要加密/解密的消息块分成数块：  
ECB 模式中，对每个块进行独立加密/解密，块与块之间没有依赖；非 ECB 模式中，块与块之间有依赖性，并且为了保证每条消息的唯一性，在第一个块中需要使用初始化向量 IV

## 1.2 使用流程

### 1.2.1 单包数据加解密

#### 场景说明

对单包数据进行加密或解密。当物理内存中有一段码流数据需要进行加/解密时，获取物理地址后在用户层调用 CIPHER 模块实现加/解密。

#### 工作流程

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](#) 或 [HI\\_UNF\\_CIPHER\\_ConfigHandleEx](#) 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

#### 注意事项

使用 CIPHER 模块时，请特别注意以下几点。

- 该接口支持 AES、DES/3DES、GCM、CMM 对称加解密算法。
- 各算法支持 ECB/CBC/CFB/OFB/CTR 等工作模式。
- 在进行加密、解密运算前必须先获取 CIPHER 句柄，当长时间不使用时可以释放，建议加密、解密各获取一个句柄，每个句柄只进行加密或者只进行解密操作。



- 只支持对物理空间连续的内存数据进行加密、解密。（用户可以通过海思的 HI\_MMZ\_New 接口获取到物理内存，并使用 HI\_MMZ\_Map 对物理内存进行虚地址映射）。
- CIPHER 内部采用 DMA 方式传输数据，所以调用 HI\_UNF\_CIPHER\_Encrypt 或 HI\_UNF\_CIPHER\_Decrypt 接口进行数据的加密或解密时，传入的地址参数为数据的物理地址。
- 加密、解密的源地址、目的地址可以相同，即可以在原地（密文、明文使用同一块 buffer）进行数据的加密和解密。
- 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。
- 使用非 ECB 模式进行 CIPHER 的加解密时，需要使用初始化向量 IV（Initial Vector）。
- 配置 IV 时有以下两个场景（以解密数据块为例）：

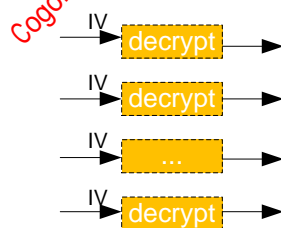
#### 【场景 1】

每次调用 CIPHER 时都需要更新 IV，此时请设置 stChangeFlags.bit1IV = 2，并正确配置 IV 值。

函数调用顺序请参考：

```
HI_UNF_CIPHER_ConfigHandle() //should set stChangeFlags.bit1IV = 2
and update u32IV
HI_UNF_CIPHER_Decrypt()
HI_UNF_CIPHER_ConfigHandle() //should set stChangeFlags.bit1IV = 2
and update u32IV
HI_UNF_CIPHER_Decrypt()
...
HI_UNF_CIPHER_ConfigHandle() //should set stChangeFlags.bit1IV = 2
and update u32IV
HI_UNF_CIPHER_Decrypt()
```

图1-1 Cipher 应用场景 1，每次调用都需要更新 IV



#### 【场景 2】

只需第一次调用 CIPHER 时设置 IV，此时请设置 stChangeFlags.bit1IV = 1，且配置 IV 值。

函数调用顺序请参考：

```
HI_UNF_CIPHER_ConfigHandle() //should set stChangeFlags.bit1IV = 1
```



```
and update u32IV
HI_UNF_CIPHER_Decrypt()HI_UNF_CIPHER_Decrypt()...
HI_UNF_CIPHER_Decrypt()
```

图1-2 Cipher 应用场景 2，只在第一次调用时配置 IV



请结合实际场景进行 IV 的配置。

- 单包加解密的 IV 向量可继承。创建一路 CIPHER，配置属性（假设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

- 如果结构体 HI\_UNF\_CIPHER\_CTRL 的成员 bKeyByCA 设置为 HI\_FALSE 时，这是普通的使用模式，表示需要手动配置 key 进行数据的加解密，例如：

```
memcpy(CipherCtrl.u32Key, u8KeyBuf, 32);
```

详细用法请参考 Cipher 相关 sample。

- 如果 bKeyByCA 设置为 HI\_TRUE，表示使用芯片内置的 Key 进行数据的加解密。
- AES-CCM，AES-GCM 只能使用 HI\_UNF\_CIPHER\_ConfigHandleEx 进行配置，CCM、GCM 在计算完成后需要获取 TAG 值，解密的 TAG 要和加密时一样解密才成功。

## 示例

具体示例请参见发布包 sample: sample\_cipher.c。

## 1.2.2 多包数据加解密

### 场景说明

对多个数据包进行加密或解密。当物理内存中有多段码流数据需要进行加/解密时，获取物理地址后在用户层调用 CIPHER 模块实现加/解密。

### 工作流程

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](#) 或 [HI\\_UNF\\_CIPHER\\_ConfigHandleEx](#) 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

## 注意事项

- 进行多包加解密时，最多支持同时加解密 128 个包。
- 对于多个包的操作，每个包都使用 [HI\\_UNF\\_CIPHER\\_ConfigHandle](#) 或 [HI\\_UNF\\_CIPHER\\_ConfigHandleEx](#) 配置的向量进行运算，IV 作用域是可配置的，前一个包的向量运算结果可以作为下一个包的 IV，或者每个包 IV 都是独立运算的（前一次函数调用的结果不会影响后一次函数调用的运算结果）。
- 其它注意事项同“单包数据加解密”章节。

## 示例

具体示例请参见发布包 sample：sample\_multicipher.c。

## 1.2.3 HASH 计算

### 场景说明

计算数据的 HASH 值，可选择 SHA1、SHA224、SHA256、SHA384、SHA512。

### 工作流程

- 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，获取 HASH 句柄，选择 HASH 算法。调用接口 [HI\\_UNF\\_CIPHER\\_HashInit](#) 完成。
- 步骤 3 输入数据，逐个数据块依次计算 HASH 值。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，结束输入，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

## 注意事项

支持软件多通道，可同时进行多个 HASH 运算，即执行步骤 2 启动一个 HASH 运算，在本次 HASH 计算未完成（即未执行步骤 5）之前，可申请一个新通道启动另一个 HASH 运算，直到申请不到通道为止。

最多支持 8 个 HASH 软件通道，8 个通道可同时都被打开，但同一时间内只有一个通道在进行运算。

## 示例

具体示例请参见发布包 sample: sample\_hash.c。

## 1.2.4 HMAC 计算

### 场景说明

计算数据的 HMAC 值。基于的 HASH 算法为 SHA1、SHA224、SHA256、SHA384 或 SHA512。

### 工作流程

HMAC 运算开发操作步骤如下：

- 步骤 1 调用 `HI_UNF_CIPHER_Init` 初始化 CIPHER 模块。
- 步骤 2 调用 `HI_UNF_CIPHER_HashInit` 选择使用的 HASH 算法，并配置 HMAC 计算的密钥，初始化 HASH 模块。
- 步骤 3 调用 `HI_UNF_CIPHER_HashUpdate` 输入数据，可以一个 BLOCK 接一个 BLOCK 输入。
- 步骤 4 调用 `HI_UNF_CIPHER_HashFinal` 结束输入，并输出 HMAC 值。
- 步骤 5 调用 `HI_UNF_CIPHER_Deinit` 去初始化 CIPHER 设备。

----结束

## 注意事项

支持软件多通道，可同时进行多个 HMAC 运算，即执行步骤 2 启动一个 HMAC 运算，在本次 HMAC 计算未完成（即未执行步骤 5）之前，可申请一个新通道启动另一个 HMAC 运算，直到申请不到通道为止。

HMAC 和 HASH 共用 8 个软件通道，8 个通道可同时都被打开，但同一时间内只有一个通道在进行运算。

## 示例

具体示例请参见发布包 sample: sample\_hash.c。





## 1.2.5 产生随机数

### 场景说明

获取硬件产生的真随机数。

### 工作流程

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 获取 32bits 的随机数。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

### 注意事项

无。

### 示例

具体示例请参见发布包 sample: sample\_rng.c。

## 1.2.6 RSA 加解密操作步骤

### 场景说明

对数据进行 RSA 不对称算法进行加解密，当使用公钥加密的数据，必须使用私钥进行解密，反之，使用私钥加密的数据，必须使用公钥进行解密。

该算法请参考：rfc3447. RSA Cryptography Specifications。

### 工作流程

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\\_RsaPublicEncrypt](#)
  - 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](#)





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

----结束

## 注意事项

RSA 密钥位宽可选 1024、2048、3072 及 4096。根据 RSA 算法原理，明文和密文都必须比公钥 N 小，所以待加解密的数据长度必须小于或等于密钥的长度，惯用作法是在待加解密的数据的高位补 0 等，使其长度和公钥 N 相等，但其值比公钥 N 小，PKCS#1 标准定义了几种填充数据的方式，分别是 Block Type 0, Block Type 1, Block Type 2, RSAES-OAEP 和 RSAES-PKCS1-v1\_5 等。

## 示例

具体示例请参见发布包 sample: sample\_rsa\_enc.c。

## 1.2.7 RSA 签名及验签操作步骤

### 场景说明

对数据进行 RSA 签名及验签时，使用私钥进行数据签名，使用公钥进行数据验签。

该算法请参考：rfc3447. RSA Cryptography Specifications。

### 工作流程

对数据进行不对称的 RSA 签名及验签的过程如下：

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，调用以下接口签名验证。

- 私钥签名 [HI\\_UNF\\_CIPHER\\_RsaSign](#)
- 公钥验证 [HI\\_UNF\\_CIPHER\\_RsaVerify](#)

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

----结束

## 注意事项

RSA 密钥位宽可选 1024、2048、3072 及 4096。根据 RSA 算法原理，明文和密文都必须比公钥小，所以待加解密的数据长度必须小于或等于密钥的长度，惯用作法是先计算待签名数据的 HASH 值，接着将 HASH 值填充成长度和公钥 N 相等但其值比公钥 N 小的数据，然后再进行加密，PKCS#1 标准定义了几种填充数据的方式，分别是 RSASSA-PSS 和 RSAES-PKCS1-v1\_5 等。

## 示例

具体示例请参见发布包 sample: sample\_rsa\_sign.c。



## 1.2.8 CCM/GCM 加解密操作步骤

### 场景说明

对数据进行 CCM 加解密。该算法请参考：SP800-38C\_updated-July20\_2007\_CCM. The CCM Mode for Authentication and Confidentiality。

对数据进行 GCM 加解密。该算法请参考：SP-800-38D-GCM. Galois/Counter Mode (GCM) and GMAC。

### 工作流程

对数据进行对称的 CCM/GCM 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 调用 [HI\\_UNF\\_CIPHER\\_CreateHandle](#) 获取 CIPHER 句柄。
- 步骤 3 调用 [HI\\_UNF\\_CIPHER\\_ConfigHandleEx](#) 配置 CIPHER 参数。
- 步骤 4 对数据进行加解密。用户可以调用以下接口进行加解密。
  - 加密--[HI\\_UNF\\_CIPHER\\_Encrypt](#)
  - 解密--[HI\\_UNF\\_CIPHER\\_Decrypt](#)
- 步骤 5 调用 [HI\\_UNF\\_CIPHER\\_GetTag](#) 获取 CCM/GCM 的 TAG 数据。
- 步骤 6 调用 [HI\\_UNF\\_CIPHER\\_DestroyHandle](#) 释放 CIPHER 句柄。
- 步骤 7 调用 [HI\\_UNF\\_CIPHER\\_Deinit](#) 去初始化 CIPHER 设备。

----结束

### 注意事项

- CCM/GCM 密钥长度可选 128bit、192bit 及 256bit。CCM/GCM 解密产生的 TAG 值必须和加密时一样，解密结果才是正确的
- AES-CCM 模式由 AES CTR 和 AES CBC 模式构成，既可以保证数据的保密性，也能保证数据的完整性
  - 根据 CCM 算法原理，向量 IV 长度 u32IVLen 可取{7, 8, 9, 10, 11, 12, 13} byte，IV 存放算法标准中的 Nonce 数据 N，加密数据的长度用 n 个 Byte 表示，且应满足条件：u32IVLen+n=15，，所以 u32IVLen 为 13 时，n 为 2，此时加密数据长度最长为 65536byte，其它以此类推。
  - CCM 加密时的向量 N、关联数据 A 取值必须与解密时保持一致。
- AES-GCM 模式由 AES CTR 和 GHASH 构成，既可以保证数据的保密性，也能保证数据的完整性
  - 根据 GCM 算法原理，GCM 的向量 IV 长度 u32IVLen 可取范围为[1~16]。
  - GCM 加密时的关联数据 A 取值必须与解密时保持一致。



## 示例

具体示例请参见发布包 sample: sample\_cipher.c。

Cogobuy Only For ShenZhen FuShi ChanJing Industrial Technology Co., Ltd.



## 2 API 参考

CIPHER 提供以下 API:

- [HI\\_UNF\\_CIPHER\\_Init](#): 初始化 CIPHER 模块。
- [HI\\_UNF\\_CIPHER\\_Deinit](#): 去初始化 CIPHER 模块。
- [HI\\_UNF\\_CIPHER\\_Open](#): 打开 CIPHER 模块。
- [HI\\_UNF\\_CIPHER\\_Close](#): 关闭 CIPHER 模块。
- [HI\\_UNF\\_CIPHER\\_CreateHandle](#): 创建一路的 Cipher 句柄。
- [HI\\_UNF\\_CIPHER\\_DestroyHandle](#): 销毁已存在的 CIPHER 句柄。
- [HI\\_UNF\\_CIPHER\\_ConfigHandle](#): 配置 CIPHER 控制信息。
- [HI\\_UNF\\_CIPHER\\_ConfigHandleEx](#): 配置 CIPHER 控制信息（扩展）。
- [HI\\_UNF\\_CIPHER\\_GetHandleConfig](#): 获取 CIPHER 通道对应的配置信息。
- [HI\\_UNF\\_CIPHER\\_Encrypt](#): 单包数据加密功能。
- [HI\\_UNF\\_CIPHER\\_Decrypt](#): 单包数据解密功能。
- [HI\\_UNF\\_CIPHER\\_EncryptVir](#): 对数据进行加密。
- [HI\\_UNF\\_CIPHER\\_DecryptVir](#): 对数据进行解密。
- [HI\\_UNF\\_CIPHER\\_EncryptMulti](#): 多包数据加密功能。
- [HI\\_UNF\\_CIPHER\\_DecryptMulti](#): 多包数据解密功能。
- [HI\\_UNF\\_CIPHER\\_HashInit](#): HASH、HMAC 计算初始化功能。
- [HI\\_UNF\\_CIPHER\\_HashUpdate](#): HASH、HMAC 计算数据输入功能。
- [HI\\_UNF\\_CIPHER\\_HashFinal](#): HASH、HMAC 计算最终结果输出功能。
- [HI\\_UNF\\_CIPHER\\_GetRandomNumber](#): 获取随机数功能。
- [HI\\_UNF\\_CIPHER\\_GetTag](#): 获取 TAG 值。
- [HI\\_UNF\\_CIPHER\\_RsaPublicEncrypt](#): 使用公钥对明文进行加密。
- [HI\\_UNF\\_CIPHER\\_RsaPrivateDecrypt](#): 使用私钥对密文进行解密。
- [HI\\_UNF\\_CIPHER\\_RsaPrivateEncrypt](#): 使用私钥对明文进行加密。
- [HI\\_UNF\\_CIPHER\\_RsaPublicDecrypt](#): 使用公钥对密文进行解密。
- [HI\\_UNF\\_CIPHER\\_RsaSign](#): 使用私钥对用户数据进行签名。



- [HI\\_UNF\\_CIPHER\\_RsaVerify](#): 使用公钥对用户数据进行合法性及完整性验证。
- [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 模块。

### 【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 模块。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_DeInit(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\_Open

【描述】

打开 CIPHER 模块。

【语法】

```
#define HI_UNF_CIPHER_Open(HI_VOID) 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\_Close

### 【描述】

关闭 CIPHER 模块。

### 【语法】

```
#define HI_UNF_CIPHER_Close(HI_VOID) HI_UNF_CIPHER_DeInit(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\_CreateHandle

### 【描述】

创建一路的 Cipher 句柄。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_CreateHandle(HI_HANDLE* phCipher, const  
HI_UNF_CIPHER_ATTRS_S *pstCipherAttr);
```

### 【参数】

参数名称	描述	输入/输出
phCipher	CIPHER 句柄指针。	输出
pstCipherAttr	CIPHER 属性指针。	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】

- phCipher、pstCipherAttr 不能为空。
- 句柄 phCipher 将用于数据加解密时的输入。
- 最大支持 7 路 cipher。
- 使用完通道后，应销毁对应的通道。

#### 【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】

创建与销毁通道成对使用。

【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\_ConfigHandleEx

### 【描述】

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

### 【语法】

```
HI_S32 HI_UNF_CIPHER_ConfigHandleEx(HI_HANDLE hCipher,  
HI\_UNF\_CIPHER\_CTRL\_EX\_S* pstExCtrl);
```

### 【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
pstExCtrl	CIPHER 控制扩展信息指针。	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】

CIPHER 控制扩展信息指针不能为空。

### 【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_GetHandleConfig

### 【描述】

获取 CIPHER 通道对应的配置信息。

### 【语法】

```
HI_S32 HI_UNF_CIPHER_GetHandleConfig(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	CIPHER 通道的配置信息。	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\_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。	输入
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。

【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。	输入
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。

#### 【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_EncryptVir

#### 【描述】

对数据进行加密。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_EncryptVir(HI_HANDLE hCipher, const HI_U8  
*pu8SrcData, HI_U8 *pu8DestData, HI_U32 u32ByteLength);
```

#### 【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
*pu8SrcData	源数据（待加密的数据）的虚拟地址。	输入
*pu8DestData	存放加密结果的虚拟地址。	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。
- 模式为 CTR/CCM/GCM 是数据的长度为任意长度，其他模式要求 block 对齐。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_DecryptVir

【描述】

对数据进行解密。

【语法】

```
HI_S32 HI_UNF_CIPHER_DecryptVir(HI_HANDLE hCipher, const HI_U8  
*pu8SrcData, HI_U8 *pu8DestData, HI_U32 u32ByteLength);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄。	输入
*pu8SrcData	源数据（待解密的数据）的虚拟地址。	输入
*pu8DestData	存放解密结果的虚拟地址。	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。
- 模式为 CTR/CCM/GCM 是数据的长度为任意长度，其他模式要求 block 对齐。



### 【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\_multiphe.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\_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	参见错误码。

#### 【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



参数名称	描述	输入/输出
pu8InputData	输入数据缓冲。	输入
u32InputDataLen	输入数据的长度，单位：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】

- 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
hHashHandle	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\_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\_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。

【举例】

参考 sample\_cipher.c。

## HI\_UNF\_CIPHER\_RsaPublicEncrypt

【描述】

使用 RSA 公钥加密一段明文。

【语法】

```
HI_S32 HI_UNF_CIPHER_RsaPublicEncrypt(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，单位：byte。	输入
pu8Output	加密结果数据。	输出
pu32OutLen	加密结果数据长度，单位：byte。	输出

#### 【返回值】

返回值	描述
0	成功
非 0	参见错误码。

#### 【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\_RsaPrivateDecrypt

#### 【描述】

使用 RSA 私钥解密一段密文。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaPrivateDecrypt(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，单位：byte。	输入
pu8Output	解密结果数据。	输出
pu32OutLen	解密结果数据长度，单位：byte。	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\_RsaPrivateEncrypt

**【描述】**

使用 RSA 私钥加密一段明文。

**【语法】**

```
HI_S32 HI_UNF_CIPHER_RsaPrivateEncrypt(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



参数名称	描述	输入/输出
u32InLen	待加密的数据长度，单位：byte。	输入
pu8Output	加密结果数据。	输出
pu32OutLen	加密结果数据长度，单位：byte。	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\_RsaPublicDecrypt

#### 【描述】

使用 RSA 公钥解密一段密文。

#### 【语法】

```
HI_S32 HI_UNF_CIPHER_RsaPublicDecrypt(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，单位：byte。	输入
pu8Output	解密结果数据。	输出



参数名称	描述	输入/输出
pu32OutLen	解密结果数据长度，单位：byte。	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
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，单位：byte。	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





参数名称	描述	输入/输出
pu8OutSign	签名结果数据。	输出
pu32OutSignLen	签名结果数据长度，单位：byte。	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  
*pRsaVerify, 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，单位：byte。	输入



参数名称	描述	输入/输出
pu8HashData	待验证文本的 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，单位：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】

无

#### 【举例】

参考 sample\_rsa\_sign.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_CA_TYPE_E enRootKey,  
HI_UNF_CIPHER_KLAD_TARGET_E enTarget, HI_U8 *pu8CleanKey,  
HI_U8* pu8EcncryptKey, 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



参数名称	描述	输入/输出
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】

无

【举例】

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

Cogobuy Only For ShenZhen FuShi ChangJing Industrial Technology Co., Ltd.



# 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\\_CTRL\\_CHANGE\\_LAG\\_S](#): 定义 CIPHER CCM 模式的信息结构体。
- [HI\\_UNF\\_CIPHER\\_CA\\_TYPE\\_E](#): 定义 CIPHER key 的来源。
- [HI\\_UNF\\_CIPHER\\_KLAD\\_TARGET\\_E](#): 定义 Klad 产生的 Key 送达的目标选择。
- [HI\\_UNF\\_CIPHER\\_TYPE\\_E](#): 定义 CIPHER 加解密类型选择。
- [HI\\_UNF\\_CIPHER\\_ATTS\\_S](#): 定义 CIPHER 加解密类型结构。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_S](#): 定义 CIPHER 控制信息结构体。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_AES\\_S](#): AES 加密控制信息结构扩展。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_AES\\_CCM\\_GCM\\_S](#): AES-CCM、AES-GCM 加密控制信息结构。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_DES\\_S](#): DES 加密控制信息结构扩展。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_3DES\\_S](#): 3DES 加密控制信息结构。
- [HI\\_UNF\\_CIPHER\\_CTRL\\_EX\\_S](#): 加密控制信息扩展结构作为算法的专用参数。
- [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 数据签名策略。
- [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。
- [CIPHER\\_IV\\_CHANGE\\_ONE\\_PKG](#): CIPHER 为数据包设置向量时, 仅更新一个数据包的 IV。
- [CIPHER\\_IV\\_CHANGE\\_ALL\\_PKG](#): CIPHER 为数据包设置向量时, 更新所有数据包的 IV。

## 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,
    HI_UNF_CIPHER_WORK_MODE_INVALID = 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。
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。
HI_UNF_CIPHER_WORK_MODE_INVALID	非法值。

### 【注意事项】

无

### 【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_SM1           = 0x3,
    HI_UNF_CIPHER_ALG_SM4           = 0x4,
    HI_UNF_CIPHER_ALG_DMA           = 0x5,
    HI_UNF_CIPHER_ALG_BUTT          = 0x6,
    HI_UNF_CIPHER_ALG_INVALID       = 0xffffffff,
```



```
}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
HI_UNF_CIPHER_ALG_SM1	SM1 算法
HI_UNF_CIPHER_ALG_SM4	SM4 算法
HI_UNF_CIPHER_ALG_DMA	DMA 直接拷贝，不做加解密运算
HI_UNF_CIPHER_ALG_BUTT	无效算法
HI_UNF_CIPHER_ALG_INVALID	非法值

#### 【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_DEFAULT = 0x0,
    HI_UNF_CIPHER_KEY_INVALID = 0xffffffff,
}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
HI_UNF_CIPHER_KEY_DEFAULT	默认 key 长度，DES：8 byte，SM1：48 byte，SM4：16 byte
HI_UNF_CIPHER_KEY_INVALID	非法值

#### 【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_INVALID = 0xffffffff,
}HI_UNF_CIPHER_BIT_WIDTH_E;
```

#### 【成员】

成员名称	描述
HI_UNF_CIPHER_BIT_WIDTH_64BIT	64bit 位宽





成员名称	描述
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
HI_UNF_CIPHER_BIT_WIDTH_INVALID	非法值

【注意事项】

无。

【相关数据类型及接口】

无。

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

【说明】

定义 CIPHER CCM 模式的信息结构体。

【定义】

```
typedef struct hiUNF_CIPHER_CTRL_CHANGE_FLAG_S
{
    HI_U32    bit1IV:    2;
    HI_U32    bitsResv:  30;
} HI_UNF_CIPHER_CTRL_CHANGE_FLAG_S;
```

【成员】

成员名称	描述
bit1IV	向量变更:0 不变更; 1 只为第一个变更; 2 为每个包变更
bitsResv	保留

【注意事项】

无。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_CA\_TYPE\_E

【说明】



定义 CIPHER key 的来源。

#### 【定义】

```
typedef enum hiHI_UNF_CIPHER_CA_TYPE_E
{
    HI_UNF_CIPHER_KEY_SRC_USER = 0x0,
    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_INVALID = 0xffffffff,
} HI_UNF_CIPHER_CA_TYPE_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_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
HI_UNF_CIPHER_KEY_SRC_INVALID	非法值

#### 【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】

无

#### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_TYPE\_E

#### 【说明】

定义 CIPHER 加解密类型选择。

#### 【定义】

```
typedef enum  
{  
    HI_UNF_CIPHER_TYPE_NORMAL = 0x0,  
    HI_UNF_CIPHER_TYPE_COPY_AVOID,  
    HI_UNF_CIPHER_TYPE_BUTT,  
    HI_UNF_CIPHER_TYPE_INVALID = 0xffffffff,  
} HI_UNF_CIPHER_TYPE_E;
```

#### 【成员】

成员名称	描述
HI_UNF_CIPHER_TYPE_NORMAL	1-7 通道 DMA 方式
HI_UNF_CIPHER_TYPE_COPY_AVOID	0 通道的 CPU 拷贝方式
HI_UNF_CIPHER_TYPE_BUTT	无效类型
HI_UNF_CIPHER_TYPE_INVALID	非法值



【注意事项】

无。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_ATTS\_S

【说明】

定义 CIPHER 加解密类型结构。

【定义】

```
typedef struct
{
    HI_UNF_CIPHER_TYPE_E enCipherType;
}HI_UNF_CIPHER_ATTS_S;
```

【成员】

成员名称	描述
enCipherType	加密类型结构体变量

【注意事项】

无。

【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_BOOL bKeyByCA;
    HI_UNF_CIPHER_CA_TYPE_E enCaType;
    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_CTRL_S;
```

#### 【成员】

成员名称	描述
u32Key[8]	密钥
u32IV[4]	初始向量
bKeyByCA	是否使用 CA key 进行加解密
enCaType	CA 类型
enAlg	加密算法
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

#### 【注意事项】

ECB 模式下不需要初始向量。

#### 【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_CTRL\_AES\_S

#### 【说明】

AES 加密控制信息结构扩展。

#### 【定义】

```
typedef struct hiHI_UNF_CIPHER_CTRL_AES_S  
{  
    HI_U32 u32EvenKey[8];  
    HI_U32 u32OddKey[8];  
    HI_U32 u32IV[4];  
    HI_UNF_CIPHER_BIT_WIDTH_E enBitWidth;  
    HI_UNF_CIPHER_KEY_LENGTH_E enKeyLen;  
    HI_UNF_CIPHER_CTRL_CHANGE_FLAG_S stChangeFlags;  
} HI_UNF_CIPHER_CTRL_AES_S;
```

#### 【成员】



成员名称	描述
u32EvenKey	偶 key(默认使用偶 key)
u32OddKey	奇 key
u32IV	向量
enBitWidth	加密位宽
enKeyLen	加密 key 长度
stChangeFlags	向量变更标志

#### 【注意事项】

AES 支持的工作模式为：ECB/CBC/CFB/OFB/CTR，其中 CFB 支持的位宽可以为 1、8、128bit，OFB 模式仅支持 128bit 位宽。

#### 【相关数据类型及接口】

无

## HI\_UNF\_CIPHER\_CTRL\_AES\_CCM\_GCM\_S

#### 【说明】

AES-CCM、AES-GCM 加密控制信息结构

#### 【定义】

```
typedef struct hiHI_UNF_CIPHER_CTRL_AES_CCM_GCM_S
{
    HI_U32 u32Key[8];
    HI_U32 u32IV[4];
    HI_UNF_CIPHER_KEY_LENGTH_E enKeyLen;
    HI_U32 u32IVLen;
    HI_U32 u32TagLen;
    HI_U32 u32ALen;
    HI_U32 u32APhyAddr;
} HI_UNF_CIPHER_CTRL_AES_CCM_GCM_S;
```

#### 【成员】

成员名称	描述
u32Key	偶 key(默认使用偶 key)
u32IV	向量
enKeyLen	加密 key 长度



成员名称	描述
IVLen	向量 IV 长度
u32TagLen	Tag 长度标志
u32ALen	关联数据 A 的长度
u32APhyAddr	关联数据 A 的物理地址

#### 【注意事项】

- 对于 CCM：向量 IV 长度 u32IVLen 可取{7, 8, 9, 10, 11, 12, 13} byte，IV 存放算法标准中的 Nonce 数据 N。加密数据的长度用 n 个 Byte 表示，且应满足条件： $u32IVLen+n=15$ 。因此，u32IVLen 为 13 时，n 为 2，此时加密数据长度最长为 65536byte，其它以此类推。Tag 的长度 u32TagLen 可取{4, 6, 8, 10, 12, 14, 16}byte，CCM 加密时的向量 N、关联数据 A 取值必须与解密时保持一致。
- 对于 GCM：向量 IV 长度 u32IVLen 可取范围为[1, 16]byte，Tag 的长度 u32TagLen 可为{12, 13, 14, 15, 16}byte，特殊情况可取 4、6byte，GCM 加密时的关联数据 A 取值必须与解密时保持一致。

#### 【相关数据类型及接口】

无

## HI\_UNF\_CIPHER\_CTRL\_DES\_S

#### 【说明】

DES 加密控制信息结构扩展。

#### 【定义】

```
typedef struct hiHI_UNF_CIPHER_CTRL_DES_S
{
    HI_U32 u32Key[2];
    HI_U32 u32IV[2];
    HI_UNF_CIPHER_BIT_WIDTH_E enBitWidth;
    HI_UNF_CIPHER_CTRL_CHANGE_FLAG_S stChangeFlags;
} HI_UNF_CIPHER_CTRL_DES_S;
```

#### 【成员】

成员名称	描述
u32Key	密钥
u32IV	向量
enBitWidth	加密位宽
stChangeFlags	向量变更标志



【注意事项】

该算法不安全，不建议产品使用该种加解密算法。

【相关数据类型及接口】

无。

## HI\_UNF\_CIPHER\_CTRL\_3DES\_S

【说明】

3DES 加密控制信息结构

【定义】

```
typedef struct hiHI_UNF_CIPHER_CTRL_3DES_S
{
    HI_U32 u32Key[6];
    HI_U32 u32IV[2];
    HI_UNF_CIPHER_BIT_WIDTH_E enBitWidth;
    HI_UNF_CIPHER_KEY_LENGTH_E enKeyLen;
    HI_UNF_CIPHER_CTRL_CHANGE_FLAG_S stChangeFlags;
} HI_UNF_CIPHER_CTRL_3DES_S;
```

【成员】

成员名称	描述
u32Key	密钥
u32IV	向量
enBitWidth	加密位宽
enKeyLen	加密 key 长度
stChangeFlags	向量变更标志

【注意事项】

- 3DES 加密过程：使用 3 个不同的 K1、K2、K3 依次进行加密、解密、加密操作，当第一次加密和最后解密时使用的 key 相等并且不等于第二次解密的 key 时，即 $(K1=K3) \neq K2$ ，此时为使用 2Key 的情况，仅需设置 K1，K2 即可。
- DES/3DES 支持的工作模式有：ECB、CBC、CFB、OFB，其中 CFB 和 OFB 支持的位宽可为 1、8、64bit

【相关数据类型及接口】

无。





## HI\_UNF\_CIPHER\_CTRL\_EX\_S

### 【说明】

加密控制信息扩展结构作为算法的专用参数，可适应各种不同类型算法的加解密场合。新增的 CCM、GCM 等算法，不适合用 [HI\\_UNF\\_CIPHER\\_CTRL\\_S](#) 进行参数配置。

### 【定义】

```
typedef struct hiHI_UNF_CIPHER_CTRL_EX_S
{
    HI_UNF_CIPHER_ALG_E enAlg;
    HI_UNF_CIPHER_WORK_MODE_E enWorkMode;
    HI_BOOL bKeyByCA;
    HI_VOID *pParam;
} HI_UNF_CIPHER_CTRL_EX_S;
```

### 【成员】

成员名称	描述
enAlg	加解密算法
enWorkMode	工作模式
bKeyByCA	是否使用硬件 key
pParam	指向各种算法的 CIPHER 控制信息结构体

### 【注意事项】

- 作为 [HI\\_UNF\\_CIPHER\\_ConfigHandleEx](#) 接口的输入参数，不同算法类型，[HI\\_VOID \\*pParam](#) 参数分别对应以下参数：
  - 对于 AES，指针应指向 [HI\\_UNF\\_CIPHER\\_CTRL\\_AES\\_S](#)
  - 对于 AES\_CCM 或 AES\_GCM，指针应指向 [HI\\_UNF\\_CIPHER\\_CTRL\\_AES\\_CCM\\_GCM\\_S](#)
  - 对于 DES，指针应指向 [HI\\_UNF\\_CIPHER\\_CTRL\\_DES\\_S](#)
  - 对于 3DES，指针应指向 [HI\\_UNF\\_CIPHER\\_CTRL\\_3DES\\_S](#)

### 【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_SIZE_T szSrcPhyAddr;
    HI_SIZE_T szDestPhyAddr;
    HI_U32 u32ByteLength;
    HI_BOOL bOddKey;
} 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
bOddKey	是否使用奇偶 key(默认使用偶 key)

### 【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_SHA224,
    HI_UNF_CIPHER_HASH_TYPE_SHA256,
    HI_UNF_CIPHER_HASH_TYPE_SHA384,
    HI_UNF_CIPHER_HASH_TYPE_SHA512,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA1,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA224,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA256,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA384,
    HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA512,
```



```
HI_UNF_CIPHER_HASH_TYPE_SM3,  
HI_UNF_CIPHER_HASH_TYPE_BUTT,  
HI_UNF_CIPHER_HASH_TYPE_INVALID = 0xffffffff,  
}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_SHA224	SHA224 哈希算法
HI_UNF_CIPHER_HASH_TYPE_SHA256	SHA256 哈希算法
HI_UNF_CIPHER_HASH_TYPE_SHA384	SHA384 哈希算法
HI_UNF_CIPHER_HASH_TYPE_SHA512	SHA512 哈希算法
HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA1	HMAC_SHA1 哈希算法
HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA224	HMAC_SHA224 哈希算法
HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA256	HMAC_SHA256 哈希算法
HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA384	HMAC_SHA384 哈希算法
HI_UNF_CIPHER_HASH_TYPE_HMAC_SHA512	HMAC_SHA512 哈希算法
HI_UNF_CIPHER_HASH_TYPE_SM3	SM3 杂凑算法
HI_UNF_CIPHER_HASH_TYPE_BUTT	无效算法
HI_UNF_CIPHER_HASH_TYPE_INVALID	非法值

#### 【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_SHA224,  
    HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256,  
    HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA384,  
    HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA512,  
    HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5,  
    HI_UNF_CIPHER_RSA_ENC_SCHEME_BUTT,  
    HI_UNF_CIPHER_RSA_ENC_SCHEME_INVALID = 0xffffffff,  
}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_SHA224	PKCS#1 的 RSAES-OAEP-SHA224 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_OAEP_SHA384	PKCS#1 的 RSAES-OAEP-SHA384 填充方式
HI_UNF_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA512	PKCS#1 的 RSAES-OAEP-SHA512 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_SCHEME_BUTT	无效值
HI_UNF_CIPHER_RSA_ENC_SCHEME_INVALID	非法值

【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_SHA224,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA384,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA512,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA224,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA384,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA512,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_BUTT,
    HI_UNF_CIPHER_RSA_SIGN_SCHEME_INVALID = 0xffffffff,
}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_SHA224	PKCS#1 RSASSA_PKCS1_V15_SHA224 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_V15_SHA384	PKCS#1 RSASSA_PKCS1_V15_SHA384 签名算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA512	PKCS#1 RSASSA_PKCS1_V15_SHA512 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_SHA224	PKCS#1 RSASSA_PKCS1_PSS_SHA224 签名算法



成员名称	描述
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_RSASSA_PKCS1_PSS_SHA384	PKCS#1 RSASSA_PKCS1_PSS_SHA384 签名算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA512	PKCS#1 RSASSA_PKCS1_PSS_SHA512 签名算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_BUTT	无效算法
HI_UNF_CIPHER_RSA_SIGN_SCHEME_INVALID	非法值

【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



成员名称	描述
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





成员名称	描述
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_CA_TYPE_E enCaType;
}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
enCaType	RSA 使用的私钥来源选择

【注意事项】

enCaType 可选择 CPU Key 或 Klad Key。

【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_CA_TYPE_E enCaType;
}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
stPriKey	RSA 私钥结构体
enCaType	RSA 使用的私钥来源选择

【注意事项】

enCaType 可选择 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_CA_TYPE_E enCaType;
} HI_UNF_CIPHER_RSA_SIGN_S;
```

【成员】

成员名称	描述
enScheme	RSA 签名算法策略
stPriKey	RSA 私钥结构体
enCaType	RSA 使用的私钥来源选择

【注意事项】

enCaType 可选择 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。

## CIPHER\_IV\_CHANGE\_ONE\_PKG

**【说明】**

CIPHER 为数据包设置向量时，仅更新一个数据包的 IV。

**【定义】**

```
#define CIPHER_IV_CHANGE_ONE_PKG (1)
```

**【注意事项】**

无

**【相关数据类型及接口】**

无。

## CIPHER\_IV\_CHANGE\_ALL\_PKG

**【说明】**

CIPHER 为数据包设置向量时，更新所有数据包的 IV。

**【定义】**

```
#define CIPHER_IV_CHANGE_ALL_PKG (2)
```

**【注意事项】**

仅在多包加密时使用该宏。

**【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_PARAM	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
0x804D0007	HI_ERR_CIPHER_FAILED_RELEASEHANDLE	释放 handle 失败
0x804D0008	HI_ERR_CIPHER_FAILED_CONFIGAES	AES 配置无效
0x804D0009	HI_ERR_CIPHER_FAILED_CONFIGDES	DES 配置无效
0x804D000A	HI_ERR_CIPHER_FAILED_ENCRYPT	加密失败
0x804D000B	HI_ERR_CIPHER_FAILED_DECRYPT	解密失败
0x804D000C	HI_ERR_CIPHER_BUSY	忙状态
0x804D000D	HI_ERR_CIPHER_NO_AVAILABLE_RNG	没有可用的随机数
0x804D000E	HI_ERR_CIPHER_FAILED_MEM	内存申请错误
0x804D000F	HI_ERR_CIPHER_UNAVAILABLE	不可用
0x804D0010	HI_ERR_CIPHER_OVERFLOW	数据溢出
0x804D0011	HI_ERR_CIPHER_HARD_STATUS	硬件状态错误
0x804D0012	HI_ERR_CIPHER_TIMEOUT	等待超时
0x804D0013	HI_ERR_CIPHER_UNSUPPORTED	不支持的配置
0x804D0014	HI_ERR_CIPHER_REGISTER_IRQ	中断号无效
0x804D0015	HI_ERR_CIPHER_ILLEGAL_UUID	非法 UUID



错误代码	宏定义	描述
0x804D0016	HI_ERR_CIPHER_ILLEGAL_KEY	非法 key
0x804D0017	HI_ERR_CIPHER_INVALID_ADDR	无效地址
0x804D0018	HI_ERR_CIPHER_INVALID_LENGTH	无效长度
0x804D0019	HI_ERR_CIPHER_ILLEGAL_DATA	无效数据
0x804D001A	HI_ERR_CIPHER_RSA_SIGN	RSA 签名失败
0x804D001B	HI_ERR_CIPHER_RSA_VERIFY	RSA 校验失败
0x804D001E	HI_ERR_CIPHER_RSA_CRYPT_FAILED	RSA 加解密失败
-1	HI_FAILURE	操作失败
0x004D0001	HI_LOG_ERR_MEM	内存操作失败
0x004D0002	HI_LOG_ERR_SEM	Semaphore 操作失败
0x004D0003	HI_LOG_ERR_FILE	文件操作失败
0x004D0004	HI_LOG_ERR_LOCK	锁操作失败
0x004D0005	HI_LOG_ERR_PARAM	参数无效
0x004D0006	HI_LOG_ERR_TIMER	计时器错误
0x004D0007	HI_LOG_ERR_THREAD	线程失败
0x004D0008	HI_LOG_ERR_TIMEOUT	超时
0x004D0009	HI_LOG_ERR_DEVICE	Device 操作失败
0x004D0010	HI_LOG_ERR_STATUS	状态出错
0x004D0011	HI_LOG_ERR_IOCTL	IO 操作失败
0x004D0012	HI_LOG_ERR_INUSE	资源使用中
0x004D0013	HI_LOG_ERR_EXIST	退出失败
0x004D0014	HI_LOG_ERR_NOEXIST	资源未退出
0x004D0015	HI_LOG_ERR_UNSUPPORTED	不支持
0x004D0016	HI_LOG_ERR_UNAVAILABLE	不可用
0x004D0017	HI_LOG_ERR_UNINITED	未初始化
0x004D0018	HI_LOG_ERR_DATABASE	数据库出错
0x004D0019	HI_LOG_ERR_OVERFLOW	溢出
0x004D0020	HI_LOG_ERR_EXTERNAL	外部出错



错误代码	宏定义	描述
0x004D0021	HI_LOG_ERR_UNKNOWNED	位置错误
0x004D0022	HI_LOG_ERR_FLASH	Flash 操作失败
0x004D0023	HI_LOG_ERR_ILLEGAL_IMAGE	非法镜像
0x004D0023	HI_LOG_ERR_ILLEGAL_UUID	非法 UUID
0x004D0023	HI_LOG_ERR_NOPERMISSION	操作不允许

Cogobuy Only For ShenZhen FuShi ChanJing Industrial Technology Co., Ltd.



# 5 Proc 调试信息

## 5.1 CIPHER 状态

### 【调试信息】

```
-----CIPHER STATUS-----
Chnid  Status  Decrypt  Alg  Mode  KeyLen
0      close   1        DES  ECB   008
1      close   1        DES  ECB   008
2      close   0        DES  ECB   008
3      close   0        DES  ECB   008
4      close   0        DES  ECB   008
5      close   0        DES  ECB   008
6      close   0        DES  ECB   008
7      close   0        DES  ECB   008

Phy-Addr in/out  KeyFrom  INT-RAW in/out  INT-EN in/out  INT_OCNTCFG
00000420/00000080  HW       0/0       0/0       00000000
00000000/00000000  SW       0/0       1/0       00000001
00000000/00000000  SW       0/0       1/0       00000001
00000000/00000000  SW       0/0       1/0       00000001
00000000/00000000  SW       0/0       1/0       00000001
00000000/00000000  SW       0/0       1/0       00000001
00000000/00000000  SW       0/0       1/0       00000001
00000000/00000000  SW       0/0       1/0       00000001
00000000/00000000  SW       0/0       1/0       00000001
```

### 【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	Chnid	通道号





参数		描述
本属性	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

Cogobuy Only For ShenZhen FuShi ChanJing Industrial Technology Co., Ltd.