

CIPHER

API 参考

文档版本 V1.1

发布日期 2025-12-17

前言

概述

CIPHER 是数字媒体处理平台提供的安全算法模块，它提供了 AES 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

未经特殊说明，以 XM7606V13 指代 XM76 系列所有芯片。

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

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	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不可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	表示如不可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。如不可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。

修订日期	版本	修订说明
2024-12-21	V1.0	DI 版本发布。
2025-12-17	V1.1	3 数据类型 更新 xmedia_cipher_rsa_pri_key 类型成员说明。

目 录

前 言.....	i
1 概述.....	1
1.1 概述.....	1
1.2 使用流程.....	2
1.2.1 单包数据加解密.....	2
1.2.2 多包数据加解密.....	5
1.2.3 HASH 计算.....	6
1.2.4 HMAC 计算.....	7
1.2.5 产生随机数.....	8
1.2.6 RSA 加解密操作步骤.....	9
1.2.7 RSA 签名及验签操作步骤.....	10
2 API 参考.....	11
3 数据类型.....	36
4 Proc 调试信息.....	55
4.1 CIPHER 状态.....	55

插图目录

图 1-1 Cipher 应用场景 1, 每次调用都需要更新 IV	4
图 1-2 Cipher 应用场景 2, 只在第一次调用时配置 IV	4

1 概述

1.1 概述

CIPHER 是数字媒体处理平台提供的安全算法模块，其提供了包括 AES 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 工作模式。

以上算法除了 CTR，其它算法、模式的数据长度必须按块大小对齐；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 2 个通道。

不对称加解密算法

RSA：支持密钥位宽 2048/4096。

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 DRBG，以更高速率获取随机数。

摘要算法

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

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
- 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_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_PSS_SHA1、RSASSA_PKCS1_PSS_SHA224、RSASSA_PKCS1_PSS_SHA256 等。

说明

- 对称加解密算法的工作模式有：

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)。
- 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 AES 加解密的过程如下：

- 步骤 1 CIPHER 设备初始化。调用接口 `xmedia_cipher_init` 完成。
- 步骤 2 创建一路 CIPHER，并获取 CIPHER 句柄。调用接口 `xmedia_cipher_create_handle` 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 `xmedia_cipher_config_handle` 完成。
- 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。
 - 单包加密--`xmedia_cipher_encrypt`
 - 单包解密--`xmedia_cipher_decrypt`
- 步骤 5 销毁 CIPHER 句柄。调用接口 `xmedia_cipher_destroy_handle` 完成。
- 步骤 6 关闭 CIPHER 设备。调用接口 `xmedia_cipher_exit` 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 对称加解密算法。
- 支持 ECB/CBC/CFB/OFB/CTR 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。
- CIPHER 内部采用 DMA 方式传输数据，所以调用 `xmedia_cipher_encrypt` 或 `xmedia_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）：

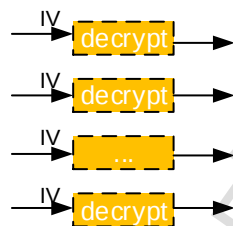
【场景 1】

每次调用 CIPHER 时都需要更新 IV，此时请设置 `iv_usage = CIPHER_IV_CHANGE_ALL_PKG`，并正确配置 IV 值。

函数调用顺序请参考：

```
xmedia_cipher_config_handle()    //should set iv_usage = 2 and update iv
xmedia_cipher_decrypt()
xmedia_cipher_config_handle()    //should set iv_usage = 2 and update iv
xmedia_cipher_decrypt()
....
xmedia_cipher_config_handle()    //should set iv_usage = 2 and update iv
xmedia_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 `iv_usage = CIPHER_IV_CHANGE_ONE_PKG`，且配置 IV 值。

函数调用顺序请参考：

```
xmedia_cipher_config_handle()    //should set iv_usage = 1 and update iv
xmedia_cipher_decrypt()
xmedia_cipher_decrypt()
....
xmedia_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 向量会随加解密流程更新。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 向量从第一个数据块开始。
- 如果结构体 `xmedia_cipher_ctrl` 的成员 `key_by_ca` 设置为 `XMEDIA_FALSE` 时，这是普通的使用模式，表示使用明文 key 进行数据的加解密，例如：

```
memcpy(cipher_ctrl.aes_ctrl.key, key_buf, 32);
```

如果 `key_by_ca` 设置为 `XMEDIA_TRUE`，此时需配置密文 key（使用 AES256 ECB 加密明文 key），表示使用 KLAD 进行 AES256 ECB 解密后送明文 key 给芯片进行数据的加解密，这样对 AES 运算实际使用的 key 进行了保护，加解密更安全。KLAD 解密使用的密钥来自 OTP。

示例

具体示例请参见发布包 sample：sample_cipher.c, sample_cipher_efuse.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 AES 加解密的过程如下：

- 步骤 1 CIPHER 设备初始化。调用接口 `xmedia_cipher_init` 完成。
- 步骤 2 创建一路 CIPHER，并获取 CIPHER 句柄。调用接口 `xmedia_cipher_create_handle` 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 `xmedia_cipher_config_handle` 完成。
- 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。
 - 多包加密--`xmedia_cipher_encrypt_multi`

- 多包解密--[xmedia_cipher_decrypt_multi](#)

步骤 5 销毁 CIPHER 句柄。调用接口 [xmedia_cipher_destroy_handle](#) 完成。

步骤 6 关闭 CIPHER 设备。调用接口 [xmedia_cipher_exit](#) 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，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（此时的 `iv_usage` 设置为 [CIPHER_IV_CHANGE_ONE_PKG](#)），或者每个包 IV 都是独立运算的（每个包使用的 IV 值都是 [xmedia_cipher_config_handle](#) 接口中的配置值，此时的 `iv_usage` 设置为 [CIPHER_IV_CHANGE_ALL_PKG](#)）。
- 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。

工作流程

步骤 1 CIPHER 设备初始化。调用接口 [xmedia_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 [xmedia_cipher_hash_init](#) 完成。

步骤 3 输入数据，逐个数据块依次计算 HASH 值。调用接口 [xmedia_cipher_hash_update](#) 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 [xmedia_cipher_hash_final](#) 完成。

步骤 6 关闭 CIPHER 设备。调用接口 `xmedia_cipher_exit` 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。

工作流程

HMAC 运算开发操作步骤如下：

- 步骤 1 调用 `xmedia_cipher_init` 初始化 CIPHER 模块。
- 步骤 2 调用 `xmedia_cipher_hash_init` 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 `xmedia_cipher_hash_update` 输入数据，可多次调用，直到输入全部数据。
- 步骤 4 调用 `xmedia_cipher_hash_final` 结束输入，并输出 HMAC 值。
- 步骤 5 调用 `xmedia_cipher_exit` 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 `xmedia_cipher_init` 完成。

步骤 2 获取 32bits 的随机数。调用接口 `xmedia_cipher_get_random_number` 完成。

步骤 3 关闭 CIPHER 设备。调用接口 `xmedia_cipher_exit` 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 `xmedia_cipher_init` 完成。

步骤 2 对数据进行加解密。根据使用的密钥不同，分为 4 个接口，用户可以调用以下接口进行加解密。

- 公钥加密--`xmedia_cipher_rsa_public_encrypt`
- 私钥解密--`xmedia_cipher_rsa_private_decrypt`
- 私钥加密--`xmedia_cipher_rsa_private_encrypt`
- 公钥解密--`xmedia_cipher_rsa_public_decrypt`

步骤 3 关闭 CIPHER 设备。调用接口 `xmedia_cipher_exit` 完成。

----结束

注意事项

RSA 密钥位宽可选 2048 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 [xmedia_cipher_init](#) 完成。

步骤 2 对数据进行签名验证，调用以下接口。

- 私钥签名--[xmedia_cipher_rsa_sign](#)
- 公钥验证--[xmedia_cipher_rsa_verify](#)

步骤 3 关闭 CIPHER 设备。调用接口 [xmedia_cipher_exit](#) 完成。

----结束

注意事项

RSA 密钥位宽可选 2048 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。

2 API 参考

CIPHER 提供以下 API:

- `xmedia_cipher_init`: 初始化 CIPHER 模块。
- `xmedia_cipher_exit`: 去初始化 CIPHER 模块。
- `xmedia_cipher_create_handle`: 创建一路的 CIPHER 句柄。
- `xmedia_cipher_destroy_handle`: 销毁已存在的 CIPHER 句柄。
- `xmedia_cipher_config_handle`: 配置 CIPHER 控制信息。
- `xmedia_cipher_get_handle_config`: 获取 CIPHER 通道对应的配置信息。
- `xmedia_cipher_encrypt`: 单包数据加密功能。
- `xmedia_cipher_decrypt`: 单包数据解密功能。
- `xmedia_cipher_encrypt_vir`: 对数据进行加密。
- `xmedia_cipher_decrypt_vir`: 对数据进行解密。
- `xmedia_cipher_encrypt_multi`: 多包数据加密功能。
- `xmedia_cipher_decrypt_multi`: 多包数据解密功能。
- `xmedia_cipher_hash_init`: HASH、HMAC 计算初始化功能。
- `xmedia_cipher_hash_update`: HASH、HMAC 计算数据输入功能。
- `xmedia_cipher_hash_final`: HASH、HMAC 计算最终结果输出功能。
- `xmedia_cipher_get_random_number`: 获取随机数功能。
- `xmedia_cipher_rsa_public_encrypt`: 使用公钥对明文进行加密。
- `xmedia_cipher_rsa_private_decrypt`: 使用私钥对密文进行解密。
- `xmedia_cipher_rsa_private_encrypt`: 使用私钥对明文进行加密。
- `xmedia_cipher_rsa_public_decrypt`: 使用公钥对密文进行解密。

- [xmedia_cipher_rsa_sign](#): 使用私钥对用户数据进行签名。
- [xmedia_cipher_rsa_verify](#): 使用公钥对用户数据进行合法性及完整性验证。
- [xmedia_cipher_klad_encrypt_key](#): 使用 KLAD 对透明密钥进行加密。

xmedia_cipher_init

【描述】

初始化 CIPHER 模块。

【语法】

```
xmedia_s32 xmedia_cipher_init(xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_NOT_PERMITTED	超过接口允许的调用次数。
XMEDIA_ERRCODE_OPEN_FAILED	打开设备节点文件失败。

【需求】

- 头文件: xmedia_cipher.h
- 库文件: libxmedia_cipher.a

【注意】

无。

【举例】

参考 sample_cipher.c。

xmedia_cipher_exit

【描述】

去初始化 CIPHER 模块。

【语法】

```
xmedia_s32 xmedia_cipher_exit(xmedia_void);
```

【参数】

无。

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_CLOSE_FAILED	关闭设备节点文件失败。

【需求】

- 头文件：xmedia_cipher.h
- 库文件：libxmedia_cipher.a

【注意】

无。

【举例】

参考 sample_cipher.c。

xmedia_cipher_create_handle

【描述】

创建一路的 Cipher 句柄。

【语法】

```
xmedia_s32 xmedia_cipher_create_handle(xmedia_handle* handle);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄指针。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_BUSY	通道已申请完，无可用通道。
XMEDIA_ERRCODE_NOT_READY	通道输入节点地址不可配置。

【需求】

头文件: xmedia_cipher.h

【注意】

- handle 不能为空。
- 句柄 handle 将用于数据加解密时的输入。
- 最大支持 2 路 cipher。
- 使用完句柄后，应销毁对应的句柄。

【举例】

参考 sample_cipher.c。

xmedia_cipher_destroy_handle

【描述】

销毁一路 CIPHER。

【语法】

```
xmedia_s32 xmedia_cipher_destroy_handle(xmedia_handle handle);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。

【需求】

头文件: xmedia_cipher.h

【注意】

创建与销毁通道成对使用。

【举例】

参考 sample_cipher.c。

xmedia_cipher_config_handle

【描述】

配置 CIPHER 控制信息。详细配置请参见结构体 [xmedia_cipher_ctrl](#)。

【语法】

```
xmedia_s32 xmedia_cipher_config_handle(xmedia_handle handle, xmedia_cipher_ctrl* ctrl);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄。	输入
ctrl	控制信息指针。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。

返回值	描述
XMEDIA_ERRCODE_BUSY	OTP 控制器忙/KLAD 忙。
XMEDIA_FAILURE	OTP 控制器操作未完成。

【需求】

头文件: xmedia_cipher.h

【注意】

控制信息指针不能为空。

【举例】

参考 sample_cipher.c。

xmedia_cipher_get_handle_config

【描述】

获取 CIPHER 通道对应的配置信息。

【语法】

```
xmedia_s32 xmedia_cipher_get_handle_config(xmedia_handle handle, xmedia_cipher_ctrl* ctrl);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄。	输入
ctrl	CIPHER 通道的配置信息。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。

【需求】

头文件: xmedia_cipher.h

【注意】

无。

【举例】

无。

xmedia_cipher_encrypt

【描述】

对数据进行加密。

【语法】

```
xmedia_s32 xmedia_cipher_encrypt(xmedia_handle handle, xmedia_ulong src_phy_addr,  
                                xmedia_ulong dest_phy_addr, xmedia_u32 length);
```

【参数】

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

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。

【需求】

头文件: xmedia_cipher.h

【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 模式为 CTR 时数据的长度为任意长度, 其他模式要求与 block size 对齐。

【举例】

参考 sample_cipher.c。

xmedia_cipher_decrypt

【描述】

对数据进行解密。

【语法】

```
xmedia_s32 xmedia_cipher_decrypt(xmedia_handle handle, xmedia_ulong src_phy_addr,  
xmedia_ulong dest_phy_addr, xmedia_u32 length);
```

【参数】

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

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。

【需求】

头文件: xmedia_cipher.h

【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 模式为 CTR 时数据的长度为任意长度，其他模式要求与 block size 对齐。

【举例】

参考 sample_cipher.c。

xmedia_cipher_encrypt_vir

【描述】

对数据进行加密。

【语法】

```
xmedia_s32 xmedia_cipher_encrypt_vir(xmedia_handle handle, const xmedia_u8 *src_data,  
xmedia_u8 *dest_data, xmedia_u32 length);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄。	输入
src_data	源数据（待加密的数据）的虚拟地址。	输入
dest_data	存放加密结果的虚拟地址。	输出
length	数据的长度（单位：字节）。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。

返回值	描述
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。

【需求】

头文件: xmedia_cipher.h

【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 模式为 CTR 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。

xmedia_cipher_decrypt_vir

【描述】

对数据进行解密。

【语法】

```
xmedia_s32 xmedia_cipher_decrypt_vir(xmedia_handle handle, const xmedia_u8 *src_data,
xmedia_u8 *dest_data, xmedia_u32 length);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄。	输入
src_data	源数据（待解密的数据）的虚拟地址。	输入
dest_data	存放解密结果的虚拟地址。	输出
length	数据的长度（单位：字节）。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。

【需求】

头文件: xmedia_cipher.h

【注意】

- CIPHER 句柄必须已创建。
- 可多次调用。
- 模式为 CTR 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。

xmedia_cipher_encrypt_multi

【描述】

进行多个包数据的加密。

【语法】

```
xmedia_s32 xmedia_cipher_encrypt_multi(xmedia_handle handle, xmedia_cipher_package *pkg,
xmedia_u32 num);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄。	输入
pkg	待加密的数据包属性地址。	输入
num	待加密的数据包个数。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。

【需求】

头文件: xmedia_cipher.h

【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 [xmedia_cipher_config_handle](#) 配置的 IV 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。

【举例】

参考 sample_multiciphe.c。

xmedia_cipher_decrypt_multi

【描述】

进行多个包数据的解密。

【语法】

```
xmedia_s32 xmedia_cipher_decrypt_multi(xmedia\_handle handle, xmedia\_cipher\_package *pkg,
xmedia_u32 num);
```

【参数】

参数名称	描述	输入/输出
handle	CIPHER 句柄。	输入

参数名称	描述	输入/输出
pkg	待解密的数据包属性地址。	输入
num	待解密的数据包个数。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_INVALID_HANDLE	输入句柄无效。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。

【需求】

头文件: xmedia_cipher.h

【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 [xmedia_cipher_config_handle](#) 配置的 IV 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。

【举例】

参考 sample_multiciphe.c。

xmedia_cipher_hash_init

【描述】

初始化 HASH 模块。

【语法】

```
xmedia_s32 xmedia_cipher_hash_init(xmedia_cipher_hash_attr *attr, xmedia_handle*handle);
```

【参数】

参数名称	描述	输入/输出
attr	用于计算 hash 的结构体参数。	输入
handle	输出的 hash 句柄地址。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。
XMEDIA_ERRCODE_BUSY	所有通道忙，无法申请通道。

【需求】

头文件：xmedia_cipher.h

【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。

xmedia_cipher_hash_update

【描述】

计算 hash 值。

【语法】

```
xmedia_s32 xmedia_cipher_hash_update(xmedia_handle handle, xmedia_u8 *input, xmedia_u32 in_len);
```

【参数】

参数名称	描述	输入/输出
handle	Hash 句柄。	输入
input	输入数据缓冲。	输入
in_len	输入数据的长度，单位：byte。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_INVALID_HANDLE	hash 句柄无效。

【需求】

头文件：xmedia_cipher.h

【注意】

- 输入数据块的长度没有对齐限制。
- 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。

xmedia_cipher_hash_final

【描述】

获取 hash 值。

【语法】

```
xmedia_s32 xmedia_cipher_hash_final(xmedia_handle handle, xmedia_u8 *output);
```

【参数】

参数名称	描述	输入/输出
handle	Hash 句柄	输入
output	输出的 hash 值	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_INVALID_HANDLE	hash 句柄无效。

【需求】

头文件: xmedia_cipher.h

【注意】

无

【举例】

参考 sample_hash.c。

xmedia_cipher_get_random_number

【描述】

生成随机数。

【语法】

```
xmedia_s32 xmedia_cipher_get_random_number(xmedia_u32 *number);
```

【参数】

参数名称	描述	输入/输出
number	输出的随机数	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_TIMEOUT	获取随机数超时。
XMEDIA_FAILURE	随机数无效。

【需求】

头文件: xmedia_cipher.h:

【注意】

无。

【举例】

参考 sample_rng.c。

xmedia_cipher_rsa_public_encrypt

【描述】

使用 RSA 公钥加密一段明文。

【语法】

```
xmedia_s32 xmedia_cipher_rsa_public_encrypt(const xmedia_cipher_rsa_pub_enc *rsa_enc,  
xmedia_cipher_data *input, xmedia_cipher_data *output);
```

【参数】

参数名称	描述	输入/输出
rsa_enc	公钥加密属性结构体。	输入
input	包含待加密的数据及其长度，单位：byte。	输入
output	包含加密结果及其长度，单位：byte。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败
XMEDIA_FAILURE	加密失败。

【需求】

头文件: xmedia_cipher.h

【注意】

使用公钥加密的数据必须使用私钥进行解密。

【举例】

参考 sample_rsa_enc.c。

xmedia_cipher_rsa_private_decrypt

【描述】

使用 RSA 私钥解密一段密文。

【语法】

```
xmedia_s32 xmedia_cipher_rsa_private_decrypt(const xmedia_cipher_rsa_pri_enc *rsa_dec,
xmedia_cipher_data *input, xmedia_cipher_data *output);
```

【参数】

参数名称	描述	输入/输出
rsa_dec	私钥解密属性结构体。	输入
input	包含待解密的数据及其长度，单位：byte。	输入
output	包含解密的数据及其长度，单位：byte。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。
XMEDIA_FAILURE	解密失败。

【需求】

头文件: xmedia_cipher.h

【注意】

- 使用公钥加密的数据必须使用私钥进行解密。
- RSA 私钥加密或解密操作可选择硬件送 key, 此时需配置 `xmedia_cipher_rsa_pri_enc` 结构体成员 `ca_type` 为 `XMEDIA_CIPHER_KEY_SRC_KLAD_1` 或 `XMEDIA_CIPHER_KEY_SRC_KLAD_2` 或 `XMEDIA_CIPHER_KEY_SRC_KLAD_3`, 并传入加密后的 D 值。若不选择硬件送 key, 则配置 `ca_type` 为 `XMEDIA_CIPHER_KEY_SRC_USER` 并传入明文 D 值。
- 密文 D 值是由明文 D 值每 16B 进行大小端转换后使用 AES256 ECB 加密得到的, 可调用 `xmedia_cipher_klad_encrypt_key` 接口对明文 D 值进行加密。
- RSA 私钥加密或解密操作可选择使用 CRT (此时不能进行硬件送 key), 使用 CRT 时 D 值需要配置为空, 并且需要配置私钥结构体 `xmedia_cipher_rsa_pri_key` 除 D 值以外的全部属性; 不使用 CRT 算法时, 只需要配置 `xmedia_cipher_rsa_pri_key` 中的私钥 (n,d 值)。

【举例】

参考 sample_rsa_enc.c。

`xmedia_cipher_rsa_private_encrypt`

【描述】

使用 RSA 私钥加密一段明文。

【语法】

```
xmedia_s32 xmedia_cipher_rsa_private_encrypt(const xmedia_cipher_rsa_pri_enc *rsa_enc,
xmedia_cipher_data *input, xmedia_cipher_data *output);
```

【参数】

参数名称	描述	输入/输出
rsa_enc	私钥加密属性结构体。	输入
input	包含待加密的数据及其长度，单位：byte。	输入
output	包含加密结果及其长度，单位：byte。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。
XMEDIA_FAILURE	加密失败。

【需求】

头文件：xmedia_cipher.h

【举例】

- 使用私钥加密的数据必须使用公钥进行解密。
- RSA 私钥加密或解密操作可选择硬件送 key，此时需配置 xmedia_cipher_rsa_pri_enc 结构体成员 ca_type 为 XMEDIA_CIPHER_KEY_SRC_KLAD_1 或 XMEDIA_CIPHER_KEY_SRC_KLAD_2 或 XMEDIA_CIPHER_KEY_SRC_KLAD_3，并传入加密后的 D 值。若不选择硬件送 key，则配置 ca_type 为 XMEDIA_CIPHER_KEY_SRC_USER 并传入明文 D 值。
- 密文 D 值是由明文 D 值每 16B 进行大小端转换后使用 AES256 ECB 加密得到的，可调用 xmedia_cipher_klad_encrypt_key 接口对明文 D 值进行加密。

- RSA 私钥加密或解密操作可选择使用 CRT（此时不能进行硬件送 key），使用 CRT 时 D 值需要配置为空，并且需要配置私钥结构体 `xmedia_cipher_rsa_pri_key` 除 D 值以外的全部属性；不使用 CRT 算法时，只需要配置 `xmedia_cipher_rsa_pri_key` 中的私钥（n,d 值）。

【举例】

参考 `sample_rsa_enc.c`。

`xmedia_cipher_rsa_public_decrypt`

【描述】

使用 RSA 公钥解密一段密文。

【语法】

```
xmedia_s32 xmedia_cipher_rsa_public_decrypt(const xmedia_cipher_rsa_pub_enc *rsa_dec,
xmedia_cipher_data *input, xmedia_cipher_data *output);
```

【参数】

参数名称	描述	输入/输出
rsa_dec	公钥解密属性结构体。	输入
input	待解密的数据及其长度，单位：byte。	输入
output	解密结果及其长度，单位：byte。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。
XMEDIA_FAILURE	解密失败。

【需求】

头文件: xmedia_cipher.h

【注意】

使用私钥加密的数据必须使用公钥进行解密。

【举例】

参考 sample_rsa_enc.c。

xmedia_cipher_rsa_sign

【描述】

使用 RSA 私钥签名一段文本。

【语法】

```
xmedia_s32 xmedia_cipher_rsa_sign(xmedia_cipher_rsa_sign_attr *rsa_sign_attr,  
xmedia_cipher_data *data, const xmedia_u8 *hash_data, xmedia_cipher_data *sign);
```

【参数】

参数名称	描述	输入/输出
rsa_sign_attr	签名属性结构体。	输入
data	待签名的数据及其长度, 如果 hash_data (待签名数据 hash 值)不为空, 则使用 hash_data 进行签名, 该参数将被忽略。	输入
hash_data	待签名文本的 HASH 摘要, 如果为空, 则自动计算输入数据的 HASH 摘要进行签名。	输入
sign	签名结果及其长度, 单位: byte。	输出

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。

返回值	描述
XMEDIA_FAILURE	签名失败。

【需求】

头文件: xmedia_cipher.h

【注意】

- 私钥签名公钥验签。
- RSA 私钥签名操作可选择硬件送 key, 此时需配置 `xmedia_cipher_rsa_sign_attr` 结构体成员 `ca_type` 为 `XMEDIA_CIPHER_KEY_SRC_KLAD_1` 或 `XMEDIA_CIPHER_KEY_SRC_KLAD_2` 或 `XMEDIA_CIPHER_KEY_SRC_KLAD_3`, 并传入加密后的 D 值。若不选择硬件送 key, 则配置 `ca_type` 为 `XMEDIA_CIPHER_KEY_SRC_USER` 并传入明文 D 值。
- 密文 D 值是由明文 D 值每 16B 进行大小端转换后使用 AES256 ECB 加密得到的, 可调用 `xmedia_cipher_klad_encrypt_key` 接口对明文 D 值进行加密。
- RSA 私钥签名操作可选择使用 CRT (此时不能进行硬件送 key), 使用 CRT 时 D 值需要配置为空, 并且需要配置私钥结构体 `xmedia_cipher_rsa_pri_key` 除 D 值以外的全部属性; 不使用 CRT 算法时, 只需要配置 `xmedia_cipher_rsa_pri_key` 中的私钥 (n,d 值)。

【举例】

参考 sample_rsa_sign.c。

xmedia_cipher_rsa_verify

【描述】

使用 RSA 公钥签名验证一段文本。

【语法】

```
xmedia_s32 xmedia_cipher_rsa_verify(xmedia_cipher_rsa_verify_attr *rsa_verify_attr,
xmedia_cipher_data *data, const xmedia_u8 *hash_data, xmedia_cipher_data *sign);
```

【参数】

参数名称	描述	输入/输出
rsa_verify_attr	签名验证属性结构体。	输入
data	待验证的数据及其长度（单位：byte），如果 hash_data 不为空，则使用 hash_data 进行验证，该参数将被忽略。	输入
hash_data	待验证文本的 HASH 摘要，如果为空，则自动计算待验证文本的 HASH 摘要进行验证。	输入
sign	待验证的签名数据及其长度，单位：byte。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_NOT_ENOUGH_MEMORY	申请内存失败。
XMEDIA_FAILURE	验签失败。

【需求】

头文件：xmedia_cipher.h

【注意】

私钥签名公钥验签。

【举例】

参考 sample_rsa_sign.c。

xmedia_cipher_klad_encrypt_key

【描述】

使用 KLAD 对透明密钥进行加密。

【语法】

```
xmedia_s32 xmedia_cipher_klad_encrypt_key(xmedia_cipher_ca_type type,
xmedia_cipher_klad_target target, xmedia_u8 *clean_key,
xmedia_u8* encrypt_key, xmedia_u32 key_len);
```

【参数】

参数名称	描述	输入/输出
type	KLAD 根密钥选择，只能选择 OTP Key。	输入
target	使用该密钥的模块。	输入
clean_key	透明密钥。	输入
encrypt_key	加密密钥。	输出
key_len	密钥的长度，必须是 16 整数倍。	输入

【返回值】

返回值	描述
XMEDIA_SUCCESS	成功。
XMEDIA_ERRCODE_INVALID_PARAM	参数错误。
XMEDIA_ERRCODE_BUSY	OTP 控制器忙/KLAD 忙。
XMEDIA_FAILURE	OTP 控制器操作未完成。

【需求】

头文件: xmedia_cipher.h

【注意】

- 使用的加密算法为 AES256 ECB。
- target 为 XMEDIA_CIPHER_KLAD_TARGET_RSA 时，会对输入数据每 16B 进行大小端转换，然后对转换后的数据进行加密；target 为 XMEDIA_CIPHER_KLAD_TARGET_AES 时，不做额外处理。

【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：

- `xmedia_cipher_handle`: 定义 CIPHER 的句柄类型。
- `xmedia_cipher_work_mode`: 定义 CIPHER 工作模式。
- `xmedia_cipher_alg`: 定义 CIPHER 加密算法。
- `xmedia_cipher_key_length`: 定义 CIPHER 密钥长度。
- `xmedia_cipher_bit_width`: 定义 CIPHER 加密位宽。
- `xmedia_cipher_ca_type`: 定义 CIPHER key 的来源。
- `xmedia_cipher_klad_target`: 定义 Klad 产生的 Key 送达的目标选择。
- `xmedia_cipher_ctrl`: 定义 CIPHER 控制信息结构体。
- `xmedia_cipher_ctrl_aes`: AES 加密控制信息结构。
- `xmedia_cipher_package`: CIPHER 多包加解密数据包结构。
- `xmedia_cipher_data`: 定义 RSA 运算输入输出数据结构体。
- `xmedia_cipher_hash_type`: 定义 CIPHER 哈希算法类型。
- `xmedia_cipher_hash_attr`: 定义 CIPHER 哈希算法初始化输入结构体。
- `xmedia_cipher_rsa_enc_scheme`: 定义 RSA 算法数据加密填充方式。
- `xmedia_cipher_rsa_sign_scheme`: 定义 RSA 数据签名策略。
- `xmedia_cipher_rsa_pub_key`: 定义 RSA 公钥结构体。
- `xmedia_cipher_rsa_pri_key`: 定义 RSA 私钥结构体。
- `xmedia_cipher_rsa_pub_enc`: 定义 RSA 公钥加解密算法参数结构体。
- `xmedia_cipher_rsa_pri_enc`: 定义 RSA 私钥解密算法参数结构体。
- `xmedia_cipher_rsa_sign_attr`: 定义 RSA 签名算法参数输入结构体。
- `xmedia_cipher_rsa_verify_attr`: 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。

xmedia_handle

【说明】

定义 CIPHER 的句柄类型。

【定义】

```
typedef xmedia_u32 xmedia_handle;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

AES 算法、HASH 算法相关接口。

xmedia_cipher_work_mode

【说明】

定义 CIPHER 工作模式。

【定义】

```
typedef enum {  
    XMEDIA_CIPHER_WORK_MODE_ECB,  
    XMEDIA_CIPHER_WORK_MODE_CBC,  
    XMEDIA_CIPHER_WORK_MODE_CFB,  
    XMEDIA_CIPHER_WORK_MODE_OFB,  
    XMEDIA_CIPHER_WORK_MODE_CTR,  
    XMEDIA_CIPHER_WORK_MODE_MAX,  
} xmedia_cipher_work_mode;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_WORK_MODE_ECB	ECB (Electronic CodeBook) 模式。

成员名称	描述
XMEDIA_CIPHER_WORK_MODE_CBC	CBC (Cipher Block Chaining) 模式。
XMEDIA_CIPHER_WORK_MODE_CFB	CFB (Cipher FeedBack) 模式。
XMEDIA_CIPHER_WORK_MODE_OFB	OFB (Output FeedBack) 模式。
XMEDIA_CIPHER_WORK_MODE_CTR	CTR (Counter) 模式。
XMEDIA_CIPHER_WORK_MODE_MAX	无效模式。

【相关数据类型及接口】

[xmedia_cipher_ctrl](#)

xmedia_cipher_alg

【说明】

定义 CIPHER 加密算法。

【定义】

```
typedef enum {
    XMEDIA_CIPHER_ALG_AES        = 0x0,
    XMEDIA_CIPHER_ALG_DMA        = 0x1,
    XMEDIA_CIPHER_ALG_MAX        = 0x2,
} xmedia_cipher_alg;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_ALG_AES	AES 算法。
XMEDIA_CIPHER_ALG_DMA	DMA 直接拷贝，不做加解密运算。
XMEDIA_CIPHER_ALG_MAX	无效算法。

【相关数据类型及接口】

[xmedia_cipher_ctrl](#)

xmedia_cipher_key_length

【说明】

定义 CIPHER 密钥长度。

【定义】

```
typedef enum {  
    XMEDIA_CIPHER_KEY_AES_128BIT    = 0x0,  
    XMEDIA_CIPHER_KEY_AES_192BIT    = 0x1,  
    XMEDIA_CIPHER_KEY_AES_256BIT    = 0x2,  
    XMEDIA_CIPHER_KEY_AES_MAX        = 0x3,  
} xmedia_cipher_key_length;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_KEY_AES_128BIT	AES 运算方式下采用 128bit 密钥长度。
XMEDIA_CIPHER_KEY_AES_192BIT	AES 运算方式下采用 192bit 密钥长度。
XMEDIA_CIPHER_KEY_AES_256BIT	AES 运算方式下采用 256bit 密钥长度。
XMEDIA_CIPHER_KEY_AES_MAX	无效值。

【注意事项】

AES 的密钥长度可以为 128bit, 192bit 或 256bit。

【相关数据类型及接口】

[xmedia_cipher_ctrl_aes](#)

xmedia_cipher_bit_width

【说明】

定义 CIPHER 加密位宽。

【定义】

```
typedef enum {  
    XMEDIA_CIPHER_BIT_WIDTH_8BIT    = 0x0,  
    XMEDIA_CIPHER_BIT_WIDTH_1BIT    = 0x1,  
    XMEDIA_CIPHER_BIT_WIDTH_128BIT  = 0x2,  
    XMEDIA_CIPHER_BIT_WIDTH_MAX      = 0x3,  
} xmedia_cipher_bit_width;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_BIT_WIDTH_8BIT	8bit 位宽。
XMEDIA_CIPHER_BIT_WIDTH_1BIT	1bit 位宽。
XMEDIA_CIPHER_BIT_WIDTH_128BIT	128bit 位宽。
XMEDIA_CIPHER_BIT_WIDTH_MAX	非法值。

【注意事项】

无。

【相关数据类型及接口】

[xmedia_cipher_ctrl_aes](#)

xmedia_cipher_ca_type

【说明】

定义 CIPHER key 的来源。

【定义】

```
typedef enum {
    XMEDIA_CIPHER_KEY_SRC_USER      = 0x0,
    XMEDIA_CIPHER_KEY_SRC_KLAD_1,
    XMEDIA_CIPHER_KEY_SRC_KLAD_2,
    XMEDIA_CIPHER_KEY_SRC_KLAD_3,
    XMEDIA_CIPHER_KEY_SRC_MAX,
} xmedia_cipher_ca_type;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_KEY_SRC_USER	用户配置的 Key。
XMEDIA_CIPHER_KEY_SRC_KLAD_1	KLAD 的第 1 组 Key, 其 Root Key 为 OTP 的第 1 组 Key。
XMEDIA_CIPHER_KEY_SRC_KLAD_2	KLAD 的第 2 组 Key, 其 Root Key 为 OTP 的第 2 组 Key。

成员名称	描述
XMEDIA_CIPHER_KEY_SRC_KLAD_3	KLAD 的第 3 组 Key, 其 Root Key 为 OTP 的第 3 组 Key。
XMEDIA_CIPHER_KEY_SRC_MAX	无效类型。

【注意事项】

无。

【相关数据类型及接口】

[xmedia_cipher_ctrl](#)

[xmedia_cipher_rsa_pri_enc](#)

[xmedia_cipher_rsa_sign_attr](#)

[xmedia_cipher_klad_encrypt_key](#)

[xmedia_cipher_klad_target](#)

【说明】

定义 Klad 产生的 Key 送达的目标选择。

【定义】

```
typedef struct {
    XMEDIA_CIPHER_KLAD_TARGET_AES        = 0x0,
    XMEDIA_CIPHER_KLAD_TARGET_RSA,
    XMEDIA_CIPHER_KLAD_TARGET_MAX,
} xmedia_cipher_klad_target;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_KLAD_TARGET_AES	Klad 产生的 Key 送到 AES。
XMEDIA_CIPHER_KLAD_TARGET_RSA	Klad 产生的 Key 送到 RSA。
XMEDIA_CIPHER_KLAD_TARGET_MAX	无效参数。

【注意事项】

无。

【相关数据类型及接口】

[xmedia_cipher_klad_encrypt_key](#)

xmedia_cipher_ctrl

【说明】

定义 CIPHER 控制信息结构体。

【定义】

```
typedef struct {  
    xmedia_cipher_alg alg;  
    xmedia_cipher_work_mode mode;  
    xmedia_bool key_by_ca;  
    xmedia_cipher_ca_type ca_type;  
    xmedia_cipher_ctrl_aes aes_ctrl;  
} xmedia_cipher_ctrl;
```

【成员】

成员名称	描述
alg	算法类型。
mode	工作模式。
key_by_ca	是否使用 CA key 进行加解密。
ca_type	CA 类型。
aes_ctrl	AES 算法相关参数。

【注意事项】

ECB 模式下不需要初始向量。

【相关数据类型及接口】

[xmedia_cipher_config_handle](#)

[xmedia_cipher_get_handle_config](#)

xmedia_cipher_ctrl_aes

【说明】

AES 加密控制信息结构体。

【定义】

```
typedef struct {  
    xmedia_u32 key[8];  
    xmedia_u32 iv[4];  
    xmedia_u32 iv_usage;  
    xmedia_cipher_bit_width bit_width;  
    xmedia_cipher_key_length key_len;  
} xmedia_cipher_ctrl_aes;
```

【成员】

成员名称	描述
key	AES 运算使用的 key。
iv	向量。
iv_usage	向量变更：1 只为第一个变更；2 为每个包变更。
bit_width	加密位宽。
key_len	加密 key 长度。

【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】

[xmedia_cipher_ctrl](#)

xmedia_cipher_package

【说明】

定义 CIPHER 多包加解密数据结构体。

【定义】

```
typedef struct {
    xmedia_ulong src_phy_addr;
    xmedia_ulong dest_phy_addr;
    xmedia_u32 byte_length;
} xmedia_cipher_package;
```

【成员】

成员名称	描述
src_phy_addr	源数据物理地址。
dest_phy_addr	目的数据物理地址。
byte_length	加解密数据长度。

【注意事项】

无。

【相关数据类型及接口】

[xmedia_cipher_encrypt_multi](#)

[xmedia_cipher_decrypt_multi](#)

xmedia_cipher_data

【说明】

定义 RSA 运算输入输出数据结构体。

【定义】

```
typedef struct {
    xmedia_u8 *data;
    xmedia_u32 data_len;
} xmedia_cipher_data;
```

【成员】

成员名称	描述
data	数据起始地址。
data_len	数据长度。

【注意事项】

无。

【相关数据类型及接口】

RSA 加解密、签名验签接口。

xmedia_cipher_hash_type

【说明】

定义 CIPHER 哈希算法类型。

【定义】

```
typedef enum {  
    XMEDIA_CIPHER_HASH_TYPE_SHA1,  
    XMEDIA_CIPHER_HASH_TYPE_SHA224,  
    XMEDIA_CIPHER_HASH_TYPE_SHA256,  
    XMEDIA_CIPHER_HASH_TYPE_HMAC_SHA1,  
    XMEDIA_CIPHER_HASH_TYPE_HMAC_SHA224,  
    XMEDIA_CIPHER_HASH_TYPE_HMAC_SHA256,  
    XMEDIA_CIPHER_HASH_TYPE_MAX,  
} xmedia_cipher_hash_type;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_HASH_TYPE_SHA1	SHA1 哈希算法。
XMEDIA_CIPHER_HASH_TYPE_SHA224	SHA224 哈希算法。
XMEDIA_CIPHER_HASH_TYPE_SHA256	SHA256 哈希算法。
XMEDIA_CIPHER_HASH_TYPE_HMAC_SHA1	HMAC_SHA1 哈希算法。
XMEDIA_CIPHER_HASH_TYPE_HMAC_SHA224	HMAC_SHA224 哈希算法。
XMEDIA_CIPHER_HASH_TYPE_HMAC_SHA256	HMAC_SHA256 哈希算法。
XMEDIA_CIPHER_HASH_TYPE_MAX	无效算法。

【相关数据类型及接口】

[xmedia_cipher_hash_attr](#)

xmedia_cipher_hash_attr

【说明】

定义 CIPHER 哈希算法初始化输入结构体。

【定义】

```
typedef struct {
    xmedia_u8 *hmac_key;
    xmedia_u32 hmac_key_len;
    xmedia_cipher_hash_type type;
} xmedia_cipher_hash_attr;
```

【成员】

成员名称	描述
hmac_key	HAMC 密钥。
hmac_key_len	HAMC 密钥长度。
type	选择哈希算法类型。

【注意事项】

无。

【相关数据类型及接口】

[xmedia_cipher_hash_init](#)

xmedia_cipher_rsa_enc_scheme

【说明】

定义 RSA 算法数据加密填充方式。

【定义】

```
typedef enum {
    XMEDIA_CIPHER_RSA_ENC_SCHEME_NO_PADDING,
    XMEDIA_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_0,
    XMEDIA_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_1,
    XMEDIA_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_2,
    XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA1,
    XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA224,
```

```

XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256,
XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5,
XMEDIA_CIPHER_RSA_ENC_SCHEME_MAX,
} xmedia_cipher_rsa_enc_scheme;

```

【成员】

成员名称	描述
XMEDIA_CIPHER_RSA_ENC_SCHEME_NO_PADDING	不填充。
XMEDIA_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_0,	PKCS#1 的 block type 0 填充方式。
XMEDIA_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_1	PKCS#1 的 block type 1 填充方式。
XMEDIA_CIPHER_RSA_ENC_SCHEME_BLOCK_TYPE_2	PKCS#1 的 block type 2 填充方式。
XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA1	PKCS#1 的 RSAES-OAEP-SHA1 填充方式。
XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA224	PKCS#1 的 RSAES-OAEP-SHA224 填充方式。
XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256	PKCS#1 的 RSAES-OAEP-SHA256 填充方式。
XMEDIA_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5	PKCS#1 的 PKCS1_V1_5 填充方式。
XMEDIA_CIPHER_RSA_ENC_SCHEME_MAX	无效值。

【相关数据类型及接口】

[xmedia_cipher_rsa_pub_enc](#)

[xmedia_cipher_rsa_pri_enc](#)

[xmedia_cipher_rsa_sign_scheme](#)

【说明】

定义 RSA 数据签名策略。

【定义】

```
typedef enum {
    XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA1 = 0x100,
    XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA224,
    XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256,
    XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1,
    XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA224,
    XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256,
    XMEDIA_CIPHER_RSA_SIGN_SCHEME_MAX,
} xmedia_cipher_rsa_sign_scheme;
```

【成员】

成员名称	描述
XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA1	PKCS#1 RSASSA_PKCS1_V15_SHA1 签名算法。
XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA224	PKCS#1 RSASSA_PKCS1_V15_SHA224 签名算法。
XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256	PKCS#1 RSASSA_PKCS1_V15_SHA256 签名算法。
XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1	PKCS#1 RSASSA_PKCS1_PSS_SHA1 签名算法。
XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA224	PKCS#1 RSASSA_PKCS1_PSS_SHA224 签名算法。
XMEDIA_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256	PKCS#1 RSASSA_PKCS1_PSS_SHA256 签名算法。
XMEDIA_CIPHER_RSA_SIGN_SCHEME_MAX	无效算法。

【相关数据类型及接口】

[xmedia_cipher_rsa_sign_attr](#)

[xmedia_cipher_rsa_verify_attr](#)

xmedia_cipher_rsa_pub_key

【说明】

定义 RSA 公钥结构体。

【定义】

```
typedef struct {  
    xmedia_u8  *n;  
    xmedia_u8  *e;  
    xmedia_u16 n_len;  
    xmedia_u16 e_len;  
} xmedia_cipher_rsa_pub_key;
```

【成员】

成员名称	描述
n	指向 RSA 公钥 N 的指针。
e	指向 RSA 公钥 E 的指针。
n_len	RSA 公钥 N 的长度。
e_len	RSA 公钥 E 的长度。

【相关数据类型及接口】

[xmedia_cipher_rsa_pub_enc](#)

[xmedia_cipher_rsa_verify_attr](#)

xmedia_cipher_rsa_pri_key

【说明】

定义 RSA 私钥结构体。

【定义】

```
typedef struct {  
    xmedia_u8 *n;  
    xmedia_u8 *e;  
    xmedia_u8 *d;  
    xmedia_u8 *p;  
    xmedia_u8 *q;  
    xmedia_u8 *dp;
```

```

xmedia_u8 *dq;
xmedia_u8 *qp;
xmedia_u16 n_len;
xmedia_u16 e_len;
xmedia_u16 d_len;
xmedia_u16 p_len;
xmedia_u16 q_len;
xmedia_u16 dp_len;
xmedia_u16 dq_len;
xmedia_u16 qp_len;
} xmedia_cipher_rsa_pri_key;

```

【成员】

成员名称	描述
n	指向 RSA 私钥 N 的指针。
e	指向 RSA 私钥 E 的指针。
d	指向 RSA 私钥 D 的指针。
p	指向 RSA 私钥 P 的指针。
q	指向 RSA 私钥 Q 的指针。
dp	指向 RSA 私钥 DP 的指针。
dq	指向 RSA 私钥 DQ 的指针。
qp	指向 RSA 私钥 QP 的指针。
n_len	RSA 私钥 N 的长度。
e_len	RSA 私钥 E 的长度。
d_len	RSA 私钥 D 的长度。
p_len	RSA 私钥 P 的长度。
q_len	RSA 私钥 Q 的长度。
dp_len	RSA 私钥 DP 的长度。
dq_len	RSA 私钥 DQ 的长度。
qp_len	RSA 私钥 QP 的长度。

【相关数据类型及接口】

[xmedia_cipher_rsa_pri_enc](#)

[xmedia_cipher_rsa_sign_attr](#)

[xmedia_cipher_rsa_pub_enc](#)

【说明】

定义 RSA 公钥加解密算法参数结构体。

【定义】

```
typedef struct {  
    xmedia\_cipher\_rsa\_enc\_scheme scheme;  
    xmedia\_cipher\_rsa\_pub\_key pub_key;  
} xmedia_cipher_rsa_pub_enc;
```

【成员】

成员名称	描述
scheme	RSA 数据加解密算法策略。
pub_key	RSA 公钥结构体。

【相关数据类型及接口】

[xmedia_cipher_rsa_public_encrypt](#)

[xmedia_cipher_rsa_public_decrypt](#)

[xmedia_cipher_rsa_pri_enc](#)

【说明】

定义 RSA 私钥解密算法参数结构体。

【定义】

```
typedef struct {  
    xmedia\_cipher\_rsa\_enc\_scheme scheme;  
    xmedia\_cipher\_rsa\_pri\_key pri_key;  
    xmedia\_cipher\_ca\_type ca_type;  
} xmedia_cipher_rsa_pri_enc;
```

【成员】

成员名称	描述
scheme	RSA 数据加解密算法策略。
pri_key	RSA 私钥结构体。
ca_type	RSA 使用的私钥来源选择。

【注意事项】

ca_type 可选择 CPU Key 或 Klad Key。

【相关数据类型及接口】

[xmedia_cipher_rsa_private_decrypt](#)

[xmedia_cipher_rsa_private_encrypt](#)

xmedia_cipher_rsa_sign_attr

【说明】

定义 RSA 签名算法参数输入结构体。

【定义】

```
typedef struct {  
    xmedia_cipher_rsa_sign_scheme scheme;  
    xmedia_cipher_rsa_pri_key pri_key;  
    xmedia_cipher_ca_type ca_type;  
} xmedia_cipher_rsa_sign_attr;
```

【成员】

成员名称	描述
scheme	RSA 签名算法策略。
pri_key	RSA 私钥结构体。
ca_type	RSA 使用的私钥来源选择。

【注意事项】

ca_type 可选择 CPU Key 或 Klad Key。

【相关数据类型及接口】

[xmedia_cipher_rsa_sign](#)

xmedia_cipher_rsa_verify_attr

【说明】

定义 RSA 签名验证算法参数输入结构体。

【定义】

```
typedef struct {  
    xmedia_cipher_rsa_sign_scheme scheme;  
    xmedia_cipher_rsa_pub_key pub_key;  
} xmedia_cipher_rsa_verify_attr;
```

【成员】

成员名称	描述
scheme	RSA 数据加解密算法策略。
pub_key	RSA 公钥结构体。

【注意事项】

无。

【相关数据类型及接口】

[xmedia_cipher_rsa_verify](#)

CIPHER_IV_CHANGE_ONE_PKG

【说明】

单包场景下，请参考图 1-2 说明的应用场景。

多包场景下，CIPHER 为数据包设置初始向量时，仅更新第一个数据包的 IV。

【定义】

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

【注意事项】

多包加解密下 iv_usage 配置为 CIPHER_IV_CHANGE_ONE_PKG 时，前一个包的向量运算结果会作为下一个包的 IV，此时可以进行多包拼接，即将多个包拼接成一个包进行加解密运算。

【相关数据类型及接口】

[xmedia_cipher_ctrl_aes](#)

CIPHER_IV_CHANGE_ALL_PKG

【说明】

单包场景下，请参考图 1-1 说明的应用场景。

多包场景下，CIPHER 为数据包设置初始向量时，更新所有数据包的 IV。

【定义】

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

【注意事项】

多包加解密下 iv_usage 配置为 CIPHER_IV_CHANGE_ALL_PKG 时，每个包使用的 IV 值都是 [xmedia_cipher_config_handle](#) 接口中的配置值。

【相关数据类型及接口】

[xmedia_cipher_ctrl_aes](#)

4 Proc 调试信息

4.1 CIPHER 状态

【调试信息】

```
-----CIPHER STATUS-----
Chnid  Status  Decrypt  Alg    Mode  KeyLen  Addr in/out
1      close   1       AES    ECB   016     42062030/42061030
2      close   1       AES    ECB   016     42065030/42064030

KeyFrom  INT-RAW in/out  INT-EN in/out  IVOUT
SW       1/0           0/1           00000000000000000000000000000000
SW       1/0           0/1           00000000000000000000000000000000
```

【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(0)/解密(1)
	Alg	算法, AES/DMA
	Mode	模式, ECB/CBC/CFB/OFB/CTR
	KeyLen	密钥长度, 16/24/32Byte

参数		描述
	Addr in/out	输入/输出物理地址
	KeyFrom	密钥来源, CPU 或 OTP
	INT-RAW	是否有原始中断
	INT-EN	是否中断使能
	IVOUT	AES 算法向量输出