

MI CIPHER API

Version 2.05

© 2019 SigmaStar Technology Corp. All rights reserved.

SigmaStar Technology makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by SigmaStar Technology arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

SigmaStar is a trademark of SigmaStar Technology Corp. Other trademarks or names herein are only for identification purposes only and owned by their respective owners.

REVISION HISTORY

Revision No.	Description	Date
2.03	<ul style="list-style-type: none">Initial release	05/25/2018
2.04	<ul style="list-style-type: none">增加 RSA 公钥/私钥加解密 API，变客户理解	06/22/2018
2.05	<ul style="list-style-type: none">修改 Api 名称 En/DeCrypt -> En/Decrypt添加 dstByteLen 以返回加/解密的输出数据长度新增 KeySize 以支持不同长度的 Key	10/09/2019

TABLE OF CONTENTS

REVISION HISTORY	i
TABLE OF CONTENTS.....	ii
1. API 参考	1
1.1. 概述.....	1
1.2. 功能模块 API	1
1.2.1 MI_CIPHER_Init	2
1.2.2 MI_CIPHER_Uninit	2
1.2.3 MI_CIPHER_CreateHandle	3
1.2.4 MI_CIPHER_DestroyHandle	3
1.2.5 MI_CIPHER_ConfigHandle	4
1.2.6 MI_CIPHER_Encrypt.....	5
1.2.7 MI_CIPHER_Decrypt.....	5
1.2.8 MI_CIPHER_HashInit.....	6
1.2.9 MI_CIPHER_HashUnInit.....	7
1.2.10 MI_CIPHER_HashUpdate	7
1.2.11 MI_CIPHER_HashFinal.....	8
1.2.12 MI_CIPHER_RsaPublicEncrypt	9
1.2.13 MI_CIPHER_RsaPublicDecrypt.....	10
1.2.14 MI_CIPHER_RsaPrivateEncrypt	10
1.2.15 MI_CIPHER_RsaPrivateDecrypt	11
1.2.16 MI_CIPHER_RsaSign	12
1.2.17 MI_CIPHER_RsaVerify	13
2. 数据类型	14
2.1. MI_CIPHER_ALG_e	15
2.2. MI_CIPHER_HASH_ALGO_e.....	15
2.3. MI_CIPHER_RSA_ENC_SCHEME_E	16
2.4. MI_CIPHER_RSA_SIGN_SCHEME_E	16
2.5. MI_CIPHER_KeySize_e.....	17
2.6. MI_CIPHER_Config_t	17
2.7. MI_CIPHER_RSA_PUB_Key_t.....	18
2.8. MI_CIPHER_RSA_PUB_Key_t.....	19
2.9. MI_CIPHER_RSA_PRI_Key_t	19
2.10. MI_CIPHER_RSA_PUB_ENC_t.....	20
2.11. MI_CIPHER_RSA_PRI_ENC_t.....	20
2.12. MI_CIPHER_RSA_SIGN_t	21
2.13. MI_CIPHER_RSA_VERIFY_t	21
3. 错误码	23

1. API 参考

1.1. 概述

CIPHER 提供数据的加解密功能，提供包括 AES\RSA\SHA 加解密算法。

1.2. 功能模块 API

API 名	功能
<u>MI_CIPHER_Init</u>	初始化 CIPHER 模块。
<u>MI_CIPHER_Uninit</u>	析构 CIPHER 模块。
<u>MI_CIPHER_CreateHandle</u>	创建 CIPHER 的 Handle。
<u>MI_CIPHER_DestroyHandle</u>	销毁 CIPHER 的 Handle。
<u>MI_CIPHER_ConfigHandle</u>	配置 CIPHER 的参数。
<u>MI_CIPHER_Encrypt</u>	加密数据。
<u>MI_CIPHER_Decrypt</u>	解密数据。
<u>MI_CIPHER_HashInit</u>	初始化 HASH 库。
<u>MI_CIPHER_HashUnInit</u>	退出 HASH 库，释放资源。
<u>MI_CIPHER_HashUpdate</u>	计算 hash 值。
<u>MI_CIPHER_HashFinal</u>	获取 hash 值。
<u>MI_CIPHER_RsaPublicEncrypt</u>	使用 rsa 公钥加密一段明文
<u>MI_CIPHER_RsaPublicDecrypt</u>	使用 rsa 公钥解密一段密文
<u>MI_CIPHER_RsaPrivateEncrypt</u>	使用 rsa 私钥加密一段明文
<u>MI_CIPHER_RsaPrivateDecrypt</u>	使用 rsa 私钥解密一段密文
<u>MI_CIPHER_RsaSign</u>	使用 rsa 私钥签名
<u>MI_CIPHER_RsaVerify</u>	使用 rsa 公钥校验

1.2.1 MI_CIPHER_Init

- 功能
初始化 CIPHER 模块。
- 语法
`MI_S32 MI_CIPHER_Init(void);`
- 形参
无
- 返回值
返回值 $\begin{cases} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{cases}$
- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so
- ※ 注意
 - 无。
- 举例
无
- 相关主题
[MI_CIPHER_Uninit](#)

1.2.2 MI_CIPHER_Uninit

- 功能
析构 CIPHER，释放资源。
- 语法
`MI_S32 MI_CIPHER_Uninit (void);`
- 形参
无
- 返回值
返回值 $\begin{cases} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{cases}$
- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

- 无。

➤ 举例

无。

➤ 相关主题

无

1.2.3 MI_CIPHER_CreateHandle

➤ 功能

创建 CIPHER 的 Handle。

➤ 语法

```
MI_S32 MI_CIPHER_CreateHandle(MI_HANDLE *phandle);
```

➤ 形参

参数名称	描述	输入/输出
phandle	Cipher 的 handle 地址指针。	输出

➤ 返回值

返回值 $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 0} & \text{失败，参照[错误码](#)。} \end{array} \right.$

➤ 依赖

- 头文件：mi_cipher.h
- 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

- 无。

➤ 举例

无。

➤ 相关主题

[MI_CIPHER_DestroyHandle](#)

1.2.4 MI_CIPHER_DestroyHandle

➤ 功能

销毁已创建 CIPHER Handle

➤ 语法

```
MI_S32 MI_CIPHER_DestroyHandle(MI_HANDLE handle);
```

➤ 形参

参数名称	描述	输入/输出
handle	已经创建的 Cipher Handle。	输入

➤ 返回值

返回值 $\begin{cases} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{cases}$

➤ 依赖

- 头文件：mi_cipher.h
- 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

无

➤ 举例

无。

➤ 相关主题

无。

1.2.5 MI_CIPHER_ConfigHandle

➤ 功能

配置 cipher 的参数。

➤ 语法

MI_S32 MI_CIPHER_ConfigHandle(MI_HANDLE handle, [MI_CIPHER_Config_t](#) *pconfig);

➤ 形参

参数名称	描述	输入/输出
handle	已经创建的 cipher handle。	输入
pconfig	cipher handle 对应的配置参数	输入

➤ 返回值

返回值 $\begin{cases} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{cases}$

➤ 依赖

- 头文件：mi_cipher.h
- 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

无

- 举例
无
- 相关主题
无

1.2.6 MI_CIPHER_Encrypt

- 功能
Cipher 加密数据。
- 语法

```
MI_U32 MI_CIPHER_Encrypt(MI_HANDLE handle, void* srcAddr,
                          void* dstAddr, MI_U32 u32srcByteLen, MI_U32* pu32dstByteLen);
```

- 形参

参数名称	描述	输入/输出
handle	已经创建的 cipher handle	输入
srcAddr	需要加密的数据地址	输入
dstAddr	加密后的数据地址	输出
u32srcByteLen	加密数据的长度	输出
u32dstByteLen	输出数据的长度	输出

- 返回值

{	0 成功。 非 0 失败，参照 错误码 。
---	--

- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so

- ※ 注意
无

- 举例
无
- 相关主题
无

1.2.7 MI_CIPHER_Decrypt

- 功能
Cipher 解密数据。

➤ 语法

```
MI_CIPHER_Decrypt(MI_HANDLE handle, void* srcAddr, void* dstAddr,
MI_U32 u32srcByteLen, MI_U32* pu32dstByteLen);
```

➤ 形参

参数名称	描述	输入/输出
handle	Cipher handle	输入
srcAddr	需要解密的数据地址	输入
dstAddr	解密后的数据地址	输出
u32srcByteLen	解密数据的长度	输出
pu32dstByteLen	输出数据的长度	输出

➤ 返回值

返回值 $\left\{ \begin{array}{l} 0 \quad \text{成功。} \\ \text{非 } 0 \quad \text{失败，参照[错误码](#)。} \end{array} \right.$

➤ 依赖

- 头文件：Mi_cipher.h
- 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

无。

➤ 举例

无。

➤ 相关主题

无。

1.2.8 MI_CIPHER_HashInit

➤ 功能

初始化 HASH 模块。

➤ 语法

```
MI_S32 MI_CIPHER_HashInit(MI\_CIPHER\_HASH\_ALGO e eHashAlgoType,
MI_HANDLE *pHashHandle);
```

➤ 形参

参数名称	描述	输入/输出
eHashAlgoType	Hash 算法类型	输入
pHashHandle	输出的 hash 句柄	输出

➤ 返回值

返回值 $\left\{ \begin{array}{l} 0 \quad \text{成功。} \\ \text{非 } 0 \quad \text{失败，参照[错误码](#)。} \end{array} \right.$

- 依赖
 - 头文件: mi_cipher.h
 - 库文件: libmi_cipher.a/libmi_cipher.so

※ 注意
无。

➤ 举例
无。

➤ 相关主题
MI_CIPHER_HashUnInit。

1.2.9 MI_CIPHER_HashUnInit

➤ 功能
退出 hash 模块，释放资源。

➤ 语法
MI_S32 MI_CIPHER_HashUnInit(MI_HANDLE hHashHandle);

➤ 形参
无

➤ 返回值
返回值 $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{array} \right.$

- 依赖
 - 头文件: mi_cipher.h
 - 库文件: libmi_cipher.a/libmi_cipher.so

※ 注意
无。

➤ 举例
无。

➤ 相关主题
MI_CIPHER_HashInit。

1.2.10 MI_CIPHER_HashUpdate

➤ 功能
计算 hash 值。

➤ 语法

```
MI_S32 MI_CIPHER_HashUpdate(MI_HANDLE hHashHandle ,
                             MI_U8 *pu8InputData, MI_U32 u32IDataLen);
```

➤ 形参

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
u32IDataLen	输入数据长度	输入

➤ 返回值

返回值 $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{array} \right.$

➤ 依赖

- 头文件：mi_cipher.h
- 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

➤ 举例

无

➤ 相关主题

无。

1.2.11 MI_CIPHER_HashFinal

➤ 功能

获取 hash 值，在计算完所有的数据后，调用这个接口获取最终的 hash 值，该接口同时会关闭 hash 句柄。如果在计算过程中，需要中断计算，也必须调用该接口关闭 hash 句柄

➤ 语法

```
MI_S32 MI_CIPHER_HashFinal(MI_HANDLE hHashHandle,
                             MI_U8 *pu8OutputHash,
                             MI_U32 *pu32OutputHashLen);
```

➤ 形参

参数名称	描述	输入/输出
hHashHandle	Hash 句柄。	输入
pu8OutputHash	输出的 hash 值	输出
pu32OutputHashLe	输出的 Hash 长度 (byte 数目)	输出

➤ 返回值

返回值 $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{array} \right.$

- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

- 举例

无。

- 相关主题

无。

1.2.12 MI_CIPHER_RsaPublicEncrypt

- 功能

使用 RSA 公钥加解密数据。

- 语法


```
MI_S32 MI_CIPHER_RsaPublicEncrypt(MI_CIPHER_RSA_PUB_ENC t *pstRsaEncrypt,
                                   MI_U8 *pu8Input, MI_U32 u32InLen,
                                   MI_U8 *pu8Output, MI_U32 *pu32OutLen));
```

- 形参

参数名称	描述	输入/输出
pstRsaEncrypt	加解密属性结构体。	输入
pu8Input	需要加密的数据	输入
u32InLen	需要加密的数据长度	输入
pu8Output	加密后的数据	输出
pu32OutLen	加密后的数据长度	输出

- 返回值

返回值 $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{array} \right.$

- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

无。

- 举例

无。

- 相关主题

无。

1.2.13 MI_CIPHER_RsaPublicDecrypt

➤ 功能

使用 RSA 公钥加解密数据。

➤ 语法

```
MI_S32 MI_CIPHER_RsaPublicDecrypt(MI_CIPHER_RSA_PUB_ENC_t *pstRsaDecrypt,
MI_U8 *pu8Input, MI_U32 u32InLen,
MI_U8 *pu8Output, MI_U32 *pu32OutLen);
```

➤ 形参

参数名称	描述	输入/输出
pstRsaDecrypt	解密属性结构体。	输入
pu8Input	需要解密的数据	输入
u32InLen	需要解密的数据长度	输入
pu8Output	解密后的数据	输出
pu32OutLen	解密后的数据长度	输出

➤ 返回值

返回值 $\begin{cases} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{cases}$

➤ 依赖

- 头文件：mi_cipher.h
- 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

无。

➤ 举例

无。

1.2.14 MI_CIPHER_RsaPrivateEncrypt

➤ 功能

使用私钥加解密数据。

➤ 语法

```
MI_S32 MI_CIPHER_RsaPrivateEncrypt(MI_CIPHER_RSA_PRI_ENC_t *pstRsaEncrypt,
MI_U8 *pu8Input, MI_U32 u32InLen,
MI_U8 *pu8Output, MI_U32 *pu32OutLen);
```

➤ 形参

参数名称	描述	输入/输出
pstRsaEncrypt	加密属性结构体。	输入
pu8Input	需要加密的数据	输入
u32InLen	需要加密的数据长度	输入
pu8Output	加密后的数据	输出
pu32OutLen	加密后的数据长度	输出

- 返回值

返回值	{	0	成功。
	}	非 0	失败，参照 错误码 。
- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so
- ※ 注意

无。
- 举例

无。
- 相关主题

无。

1.2.15 MI_CIPHER_RsaPrivateDecrypt

- 功能

使用私钥加解密数据。
- 语法


```
MI_S32 MI_CIPHER_RsaPrivateDecrypt(MI_CIPHER_RSA_PRI_ENC_t *pstRsaDecrypt ,
                                     MI_U8 *pu8Input, MI_U32 u32InLen,
                                     MI_U8 *pu8Output, MI_U32 *pu32OutLen));
```

- 形参

参数名称	描述	输入/输出
pstRsaDecrypt	解密属性结构体。	输入
pu8Input	需要解密的数据	输入
u32InLen	需要解密的数据长度	输入
pu8Output	解密后的数据	输出
pu32OutLen	解密后的数据长度	输出

- 返回值

返回值	{	0	成功。
	}	非 0	失败，参照 错误码 。
- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so

- ※ 注意

无。

- 举例
无。
- 相关主题
无。

1.2.16 MI_CIPHER_RsaSign

- 功能
使用 RSA 私钥签名数据。
- 语法

```
MI_S32 MI_CIPHER_RsaSign(MI\_CIPHER\_RSA\_SIGN\_t *pstRsaSign,
                          MI_U8 *pu8InHashData,
                          MI_U32 u32HashDataLen,
                          MI_U8 *pu8OutSign,
                          MI_U32 *pu32OutSignLen);
```

- 形参

参数名称	描述	输入/输出
pstRsaSign	签名属性结构体。	输入
pu8InHashData	待签名文本的 HASH 摘要。	输入
u32HashDataLen	输入的 hash 摘要的长度	输入
pu8OutSign	签名信息	输出
pu32OutSignLen	签名信息的长度	输出

- 返回值

返回值
 $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{array} \right.$
- 依赖
 - 头文件：mi_cipher.h
 - 库文件：libmi_cipher.a/libmi_cipher.so

- ※ 注意
无。

- 举例
无。
- 相关主题
无。

1.2.17 MI_CIPHER_RsaVerify

➤ 功能

使用 RSA 公钥验证。

➤ 语法

```
MI_S32 MI_CIPHER_RsaVerify(MI\_CIPHER\_RSA\_VERIFY\_t *pstRsaVerify,
                           MI_U8 *pu8InHashData,
                           MI_U32 u32HashDataLen,
                           MI_U8 *pu8InSign,
                           MI_U32 u32InSignLen);
```

➤ 形参

参数名称	描述	输入/输出
pstRsaVerify	校验属性结构体。	输入
pu8InHashData	待签名文本的 HASH 摘要，	输入
u32HashDataLen	输入的 hash 摘要的长度	输入
pu8InSign	签名信息	输出
u32InSignLen	签名信息的长度	输出

➤ 返回值

返回值 $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 } 0 & \text{失败，参照[错误码](#)。} \end{array} \right.$

➤ 依赖

- 头文件：mi_cipher.h
- 库文件：libmi_cipher.a/libmi_cipher.so

※ 注意

无。

➤ 举例

无。

➤ 相关主题

无。

2. 数据类型

相关数据类型、数据结构、联合体定义如下：

<u>MI_CIPHER_ALG_e</u>	定义 AES 加解密算法枚举类型
<u>MI_CIPHER_HASH_ALGO_e;</u>	定义哈希算法枚举类型
<u>MI_CIPHER_RSA_ENC_SCHEME_E</u>	定义 RSA 加密方案枚举类型
<u>MI_CIPHER_RSA_SIGN_SCHEME_E</u>	定义 RSA 签名方案枚举类型
<u>MI_CIPHER_KeySize_e</u>	定义 AES 加密密钥长度枚举类型
<u>MI_CIPHER_Config_t</u>	定义 Cipher 配置结构体
<u>MI_CIPHER_RSA_PUB_Key_t</u>	定义公钥 key 数据结构类型
<u>MI_CIPHER_RSA_PRI_Key_t</u>	定义私钥 key 数据结构类型
<u>MI_CIPHER_RSA_PUB_ENC_t</u>	定义 MD 结果的结构体
<u>MI_CIPHER_RSA_PRI_ENC_t</u>	定义私钥加解密结构体类型
<u>MI_CIPHER_RSA_SIGN_t</u>	定义签名结构体类型
<u>MI_CIPHER_RSA_VERIFY_t;</u>	定义校验结构体类型

2.1. MI_CIPHER_ALG_e

➤ 说明

定义 AES 加解密算法枚举值

➤ 定义

```
typedef enum
{
    MI_CIPHER_ALG_AES_CBC,
    MI_CIPHER_ALG_AES_CTR,
    MI_CIPHER_ALG_AES_ECB,
} MI_CIPHER_ALG_e;
```

➤ 成员

成员名称	描述
MI_CIPHER_ALG_AES_CBC	CBC（Cipher Block Chaining）模式 AEC 算法
MI_CIPHER_ALG_AES_CTR	CTR（Counter）模式 AEC 算法
MI_CIPHER_ALG_AES_ECB	ECB（Electronic CodeBook）模式 AEC 算法

※ 注意事项

无。

➤ 相关数据接口及类型

无。

2.2. MI_CIPHER_HASH_ALGO_e

➤ 说明

哈希算法类型。

➤ 定义

```
typedef enum
{
    MI_CIPHER_HASH_ALG_SHA1;
    MI_CIPHER_HASH_ALG_SHA256;
} MI_CIPHER_HASH_ALGO_e;
```

➤ 成员

成员名称	描述
MI_CIPHER_HASH_ALG_SHA1	SHA1 哈希算法
MI_CIPHER_HASH_ALG_SHA256	SHA256 哈希算法

※ 注意事项

无。

➤ 相关数据类型及接口

无。

2.3. MI_CIPHER_RSA_ENC_SCHEME_E

➤ 说明

定义 RSA 加密方案枚举类型。

➤ 定义

```
typedef enum
{
    MI_CIPHER_RSA_ENC_SCHEME_NO_PADDING,
    MI_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA1,
    MI_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256,
    MI_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5,
    MI_CIPHER_RSA_ENC_SCHEME_BUTT,
}MI_CIPHER_RSA_ENC_SCHEME_E;
```

➤ 成员

成员名称	描述
MI_CIPHER_RSA_ENC_SCHEME_NO_PADDING	不填充
MI_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA1	OAEP 填充, SHA1 运算
MI_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256	OAEP 填充, SHA256 运算
MI_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5	PKCS1v15 填充
MI_CIPHER_RSA_ENC_SCHEME_BUTT	错误参数

※ 注意事项
无。

➤ 相关数据类型及接口
无。

2.4. MI_CIPHER_RSA_SIGN_SCHEME_E

➤ 说明

定义 RSA 签名方案类型。

➤ 定义

```
typedef enum
{
    MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA1 = 0x100,
    MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256,
    MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1,
    MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256,
    MI_CIPHER_RSA_SIGN_SCHEME_BUTT,
}MI_CIPHER_RSA_SIGN_SCHEME_E;
```

➤ 成员

成员名称	描述
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA1	PKCS#1 RSASSA_PKCS1_V15_SHA1
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256	PKCS#1 RSASSA_PKCS1_V15_SHA256
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1	PKCS#1 RSASSA_PKCS1_PSS_SHA1
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256	PKCS#1 RSASSA_PKCS1_PSS_SHA256
MI_CIPHER_RSA_SIGN_SCHEME_BUTT	错误参数

※ 注意事项
无。

➤ 相关数据类型及接口
无。

2.5. MI_CIPHER_KeySize_e

➤ 说明

定义 Aes 加密密钥长度枚举类型。

➤ 定义

```
typedef enum {
    E_MI_CIPHER_KEY_SIZE_128 = 0,
    E_MI_CIPHER_KEY_SIZE_192,
    E_MI_CIPHER_KEY_SIZE_256,
} MI_CIPHER_KeySize_e;
```

➤ 成员

成员名称	描述
E_MI_CIPHER_KEY_SIZE_128	密钥长度为 128 位
E_MI_CIPHER_KEY_SIZE_192	密钥长度为 192 位
E_MI_CIPHER_KEY_SIZE_256	密钥长度为 256 位

※ 注意事项
无。

➤ 相关数据类型及接口
无。

2.6. MI_CIPHER_Config_t

➤ 说明

定义 Cipher 配置结构体类型。

➤ 定义

```
typedef struct MI_CIPHER_Config_s
{
    MI_CIPHER_KeySize_e eKeySize;
    MI_U8 key[MI_CIPHER_KEY_SIZE_MAX];
    MI_U8 iv[AES_BLOCK_SIZE];
    MI_CIPHER_ALG_e eAlg;
} MI_CIPHER_Config_t;
```

➤ 成员

成员名称	描述
eKeySize	指定加密密钥长度的枚举类型
key	存储加密密钥的数组， MI_CIPHER_KEY_SIZE_MAX 等于 32
iv	初始化向量，AES_BLOCK_SIZE 等于 16
eAlg	加解密算法类型

※ 注意事项

无。

➤ 相关数据接口及类型

无。

2.7. MI_CIPHER_RSA_PUB_Key_t

➤ 说明

定义公钥 key 的数据结构类型。

➤ 定义

```
typedef struct MI_CIPHER_RSA_PUB_Key_s
{
    MI_U8* pu8ExpE;
    MI_U8* pu8ModN;
    MI_U32 expSize;
    MI_U32 modSize;
} MI_CIPHER_RSA_PUB_Key_t;
```

➤ 成员

成员名称	描述
pu8ExpE;	公钥指数数据指针
pu8ModN	公钥模数数据指针
expSize	公钥指数数据长度
modSize	公钥模数数据长度

※ 注意事项

无。

➤ 相关数据类型及接口

无。

2.8. MI_CIPHER_RSA_PUB_Key_t

➤ 说明

定义公钥 **key** 的数据结构类型。

➤ 定义

```
typedef struct MI_CIPHER_RSA_PUB_Key_s
{
    MI_U8*    pu8ExpE;
    MI_U8*    pu8ModN;
    MI_U32    expSize;
    MI_U32    modSize;
} MI_CIPHER_RSA_PUB_Key_t;
```

➤ 成员

成员名称	描述
pu8ExpE;	公钥指数数据指针
pu8ModN	公钥模数数据指针
expSize	公钥指数数据长度
modSize	公钥模数数据长度

※ 注意事项

无。

➤ 相关数据类型及接口

无。

2.9. MI_CIPHER_RSA_PRI_Key_t

➤ 说明

定义私钥 **key** 的数据结构类型。

➤ 定义

```
typedef struct MI_CIPHER_RSA_PRI_Key_s
{
    MI_U8*    pu8ExpD;
    MI_U8*    pu8ModN;
    MI_U32    expSize;
    MI_U32    modSize;
} MI_CIPHER_RSA_PRI_Key_t;
```

➤ 成员

成员名称	描述
pu8ExpD	私钥指数数据指针
pu8ModN	私钥模数数据指针
expSize	私钥指数数据长度
modSize	私钥模数数据长度

※ 注意事项

无。

- 相关数据类型及接口
无。

2.10. MI_CIPHER_RSA_PUB_ENC_t

- 说明
定义公钥加解密算法参数结构体。

- 定义

```
typedef struct MI_CIPHER_RSA_PUB_ENC_s
{
    MI\_CIPHER\_RSA\_ENC\_SCHEME\_E eRsaAlgoType;
    MI\_CIPHER\_RSA\_PUB\_Key\_t stPubKey;
} MI_CIPHER_RSA_PUB_ENC_t;
```

- 成员

成员名称	描述
eRsaAlgoType	加解密算法类型
stPubKey	Key 数据

- ※ 注意事项
无。

- 相关数据类型及接口
无。

2.11. MI_CIPHER_RSA_PRI_ENC_t

- 说明
定义公钥加解密参数结构体。

- 定义

```
typedef struct MI_CIPHER_RSA_PRI_ENC_s
{
    MI\_CIPHER\_RSA\_ENC\_SCHEME\_E eRsaAlgoType;
    MI\_CIPHER\_RSA\_PRI\_Key\_t stPriKey;
} MI_CIPHER_RSA_PRI_ENC_t;
```

- 成员

成员名称	描述
eRsaAlgoType	加解密算法类型
stPriKey	私钥 Key 数据

- ※ 注意事项
无。

- 相关数据类型及接口
无。

2.12. MI_CIPHER_RSA_SIGN_t

- 说明
定义 RSA 签名的结构体。

- 定义

```
typedef struct MI_CIPHER_RSA_SIGN_s
{
    MI\_CIPHER\_RSA\_SIGN\_SCHEME\_E eRsaAlgoType;
    MI\_CIPHER\_RSA\_PRI\_Key\_t stPriKey;
} MI_CIPHER_RSA_SIGN_t;
```

- 成员

成员名称	描述
成员名称	描述
eRsaAlgoType	加解密算法类型
stPriKey	私钥 Key 数据，用于签名

- ※ 注意事项
无。

- 相关数据类型及接口
无。

2.13. MI_CIPHER_RSA_VERIFY_t

- 说明
定义公钥加解密算法参数结构体。

- 定义

```
typedef struct MI_CIPHER_RSA_Veriry_s
{
    MI\_CIPHER\_RSA\_SIGN\_SCHEME\_E eRsaAlgoType;
    MI\_CIPHER\_RSA\_PUB\_Key\_t stPubKey;
} MI_CIPHER_RSA_VERIFY_t;
```

- 成员

成员名称	描述
eRsaAlgoType	加解密算法填充类型
stPubKey	公钥 Key 数据，用于校验

- ※ 注意事项
 无。
- 相关数据类型及接口
 无。

3. 错误码

AI API 错误码如表 3-1 所示：

表 3-1 AI API 错误码

宏定义	描述
MI_CIPHER_ERR_INVALID_DEVID	设备号无效
MI_CIPHER_ERR_ILLEGAL_PARAM	参数设置无效
MI_CIPHER_ERR_NOT_ENABLED	设备没有使能
MI_CIPHER_ERR_NOT_DISABLED	设备没有关闭
MI_CIPHER_ERR_NULL_PTR	使用空指针
MI_CIPHER_ERR_INVALID_CHNID	通道号无效
MI_CIPHER_ERR_NOT_CONFIG	设备没有配置
MI_CIPHER_ERR_NOT_SUPPORT	操作不支持
MI_CIPHER_ERR_NOT_PERM	操作不允许
MI_CIPHER_ERR_NOMEM	分配内存失败
MI_CIPHER_ERR_NOBUF	缓存不足
MI_CIPHER_ERR_BUF_EMPTY	缓存为空
MI_CIPHER_ERR_BUF_FULL	缓存为满
MI_CIPHER_ERR_SYS_NOTREADY	系统为初始化
MI_CIPHER_ERR_BUSY	系统繁忙
MI_CIPHER_ERR_MOD_NOTINIT	模块未初始化
MI_CIPHER_ERR_MOD_INITED	模块已初始化
MI_CIPHER_ERR_FAILED	非预期错误