MI CIPHER API

Version 2.05

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	05/25/2018
2.04	• Added RSA public key/private key encryption and decryption API for better understanding	06/22/2018
2.05	• Renamed Api En/DeCrypt to En/Decrypt • Added dstByteLen to specify Output data of En/Decrypt • Added Keysize to support different length of key	10/09/2019

TABLE OF CONTENTS

REVISION HISTORY i
TABLE OF CONTENTS ii
1. API REFERENCE 1
1.1. Description 1
1.2. API List 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 8
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 11
1.2.15 MI_CIPHER_RsaPrivateDecrypt 11
1.2.16 MI_CIPHER_RsaSign 12
1.2.17 MI_CIPHER_RsaVerify 13
2. DATA TYPE 15
2.1. MI_CIPHER_ALG_e 16
2.2. MI_CIPHER_HASH_ALGO_e 16
2.3. MI_CIPHER_RSA_ENC_SCHEME_E 17
2.4. MI_CIPHER_RSA_SIGN_SCHEME_E 17
2.5. MI_CIPHER_KeySize_e 18
2.6. MI_CIPHER_Config_t 19
2.7. MI_CIPHER_RSA_PUB_Key_t 19
2.8. MI_CIPHER_RSA_PRI_Key_t 20
2.9. MI_CIPHER_RSA_PUB_ENC_t 21
2.10. MI_CIPHER_RSA_PRI_ENC_t 21
2.11. MI_CIPHER_RSA_SIGN_t 22
2.12. MI_CIPHER_RSA_VERIFY_t 22
3. ERROR CODE 23

1. API REFERENCE

1.1. Description

The CIPHER module provides data encryption/decryption function, including AES/RSA/SHA encryption and decryption algorithms.

1.2. API List

Name of API	Function
MI_CIPHER_Init	Initialize CIPHER module
MI_CIPHER_UnInit	Deinitialize CIPHER module
MI_CIPHER_CreateHandle	Create a CIPHER Handle
MI_CIPHER_DestroyHandle	Destroy a CIPHER Handle
MI_CIPHER_ConfigHandle	Configure the CIPHER parameter
MI_CIPHER_Encrypt	Encrypt data
MI_CIPHER_Decrypt	Decrypt data
MI_CIPHER_HashInit	Initialize HASH library
MI_CIPHER_HashUnInit	Deinitialize HASH library and release resource
MI_CIPHER_HashUpdate	Update hash value
MI_CIPHER_HashFinal	Get hash value
MI_CIPHER_RsaPublicEncrypt	Use RSA public key to encrypt a plaintext
MI_CIPHER_RsaPublicDecrypt	Use RSA public key to decrypt a ciphertext
MI_CIPHER_RsaPrivateEncrypt	Use RSA private key to encrypt a plaintext
MI_CIPHER_RsaPrivateDecrypt	Use RSA private key to decrypt a ciphertext
MI_CIPHER_RsaSign	Use RSA private key to sign data
MI_CIPHER_RsaVerify	Use RSA public key to verify data

1.2.1 MI_CIPHER_Init

- Function

  Initialize CIPHER module

- Syntax

  MI_S32 MI_CIPHER_Init(void);

- Parameter

  N/A.

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  MI_CIPHER_Uninit

1.2.2 MI_CIPHER_Uninit

- Function

  Deinitialize CIPHER module to release resource

- Syntax

  MI_S32 MI_CIPHER_Uninit (void);

- Parameter

  N/A.

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.3 MI_CIPHER_CreateHandle

- Function

  Create a CIPHER Handle

- Syntax

  MI_S32 MI_CIPHER_CreateHandle(MI_HANDLE *phandle);

- Parameter

Parameter	Description	Input/Output
phandle	Pointer to Cipher handle address	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  MI_CIPHER_DestroyHandle

1.2.4 MI_CIPHER_DestroyHandle

- Function

  Destroy a CIPHER Handle

- Syntax

  MI_S32 MI_CIPHER_DestroyHandle(MI_HANDLE handle);

- Parameter

Parameter	Description	Input/Output
handle	The created Cipher Handle.	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.5 MI_CIPHER_ConfigHandle

- Function

  Configure the Cipher parameter.

- Syntax

  MI_S32 MI_CIPHER_ConfigHandle(MI_HANDLE handle, MI_CIPHER_Config_t *pconfig);

- Parameter

Parameter	Description	Input/Output
handle	The created Cipher Handle.	Input
pconfig	Configuration parameter corresponding to the cipher handle	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.6 MI_CIPHER_Encrypt

- Function

  Encrypt data.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
handle	The created Cipher Handle	Input
srcAddr	Address of data to be encrypted	Input
dstAddr	Address of data after encryption	Output
u32srcByteLen	Length of encrypted data	Output
pu32dstByteLen	Length of Output data for encryption	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.7 MI_CIPHER_Decrypt

- Function

  Decrypt data.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
handle	The created Cipher Handle	Input
srcAddr	Address of data to be decrypted	Input
dstAddr	Address of data after decryption	Output
u32srcByteLen	Length of decrypted data	Output
pu32dstByteLen	Length of Output data for decryption	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: Mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.8 MI_CIPHER_HashInit

- Function

  Initialize HASH module.

- Syntax

  MI_S32 MI_CIPHER_HashInit(MI_CIPHER_HASH_ALGO_e eHashAlgoType,
  MI_HANDLE *pHashHandle);

- Parameter

Parameter	Description	Input/Output
eHashAlgoType	Hash algorithm type	Input
pHashHandle	Output hash handle	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details
- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a
- Note

  N/A.
- Example

  N/A.
- Related API

  MI_CIPHER_HashUnInit.

1.2.9 MI_CIPHER_HashUnInit

- Function

  Deinitialize hash module to release resource.
- Syntax

  MI_S32 MI_CIPHER_HashUnInit(MI_HANDLE hHashHandle);
- Parameter

  N/A.
- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details
- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a
- Note

  N/A.
- Example

  N/A.
- Related API

  MI_CIPHER_HashInit.

1.2.10 MI_CIPHER_HashUpdate

- Function

  Update hash value.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
hHashHandle	Hash handle	Input
pu8InputData	Input data buffer	Input
u32IDataLen	Input data length	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.11 MI_CIPHER_HashFinal

- Function

  Get hash value. After calculating all the data, you can call this interface to get the final hash value. This interface, when called, will close the hash handle. If the calculation is to be suspended halfway, this interface must be called as well, to close the hash handle.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
hHashHandle	Hash handle	Input
pu8OutputHash	Output hash value	Output
pu32OutputHashLe	Output hash length (byte count)	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.12 MI_CIPHER_RsaPublicEncrypt

- Function

  Use RSA public key to encrypt data.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
pstRsaEncrypt	Encrypt attribute structure	Input
pu8Input	Data to be encrypted	Input
u32InLen	Length of data to be encrypted	Input
pu8Output	Data after encryption	Output
pu32OutLen	Length of data after encryption	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.13 MI_CIPHER_RsaPublicDecrypt

- Function

  Use RSA public key to decrypt data.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
pstRsaDecrypt	Decrypt attribute structure	Input
pu8Input	Data to be decrypted	Input
u32InLen	Length of data to be decrypted	Input
pu8Output	Data after decryption	Output
pu32OutLen	Length of data after decryption	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

1.2.14 MI_CIPHER_RsaPrivateEncrypt

- Function

  Use private key to encrypt data.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
pstRsaEncrypt	Encrypt attribute structure	Input
pu8Input	Data to be encrypted	Input
u32InLen	Length of data to be encrypted	Input
pu8Output	Data after encryption	Output
pu32OutLen	Length of data after encryption	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.15 MI_CIPHER_RsaPrivateDecrypt

- Function

  Use private key to decrypt data.

- Syntax

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

- Parameter

Parameter	Description	Input/Output
pstRsaDecrypt	Decrypt attribute structure	Input
pu8Input	Data to be decrypted	Input
u32InLen	Length of data to be decrypted	Input
pu8Output	Data after decryption	Output
pu32OutLen	Length of data after decryption	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.16 MI_CIPHER_RsaSign

- Function

  Use RSA private key to sign data.

- Syntax

```
MI_S32   MI_CIPHER_RsaSign(MI_CIPHER_RSA_SIGN_t *pstRsaSign,
                           MI_U8 *pu8InHashData,
                           MI_U32 u32HashDataLen,
                           MI_U8 *pu8OutSign,
                           MI_U32 *pu32OutSignLen);
```

- Parameter

Parameter	Description	Input/Output
pstRsaSign	Signature attribute structure	Input
pu8InHashData	Hash summary of the text to be signed	Input
u32HashDataLen	Input hash data length	Input
pu8OutSign	Signature information	Output
pu32OutSignLen	Length of signature information	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

1.2.17 MI_CIPHER_RsaVerify

- Function

  Use RSA public key to verify data.

- Syntax

```
MI_S32   MI_CIPHER_RsaVerify(MI_CIPHER_RSA_VERIFY_t *pstRsaVerify,
                             MI_U8 *pu8InHashData,
                             MI_U32 u32HashDataLen,
                             MI_U8 *pu8InSign,
                             MI_U32 u32InSignLen);
```

- Parameter

Parameter	Description	Input/Output
pstRsaVerify	Verification attribute structure	Input
pu8InHashData	Hash summary of the text to be verified	Input
u32HashDataLen	Input hash data length	Input
pu8InSign	Signature information	Output
u32InSignLen	Length of signature information	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_common.h, mi_sys.h
  - Library: libmi.a

- Note

  N/A.

- Example

  N/A.

- Related API

  N/A.

2. DATA TYPE

The Cipher related data type, data structure, and complex are defined in the following table:

MI_CIPHER_ALG_e	Define AES encryption/decryption algorithm enumeration type
MI_CIPHER_HASH_ALGO_e	Define hash algorithm enumeration type
MI_CIPHER_RSA_ENC_SCHEME_E	Define encryption scheme enumeration type
MI_CIPHER_RSA_SIGN_SCHEME_E	Define sign scheme enumeration type
MI_CIPHER_KeySize_e	Define AES key size enumeration type
MI_CIPHER_Config_t	Define Cipher configuration structure
MI_CIPHER_RSA_PUB_Key_t	Define public key data structure
MI_CIPHER_RSA_PRI_Key_t	Define private key data structure
MI_CIPHER_RSA_PUB_ENC_t	Define Public Key Encrypt structure
MI_CIPHER_RSA_PRI_ENC_t	Define private key encryption/decryption structure
MI_CIPHER_RSA_SIGN_t	Define signature structure
MI_CIPHER_RSA_VERIRY_t	Define verification structure

2.1. MI_CIPHER_ALG_e

- Description

  Define AES Encryption/Decryption algorithm enumeration value

- Definition

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

- Member

Member	Description
MI_CIPHER_ALG_AES_CBC	CBC (Cipher Block Chaining) mode AEC algorithm
MI_CIPHER_ALG_AES_CTR	CTR (Counter) mode AEC algorithm
MI_CIPHER_ALG_AES_ECB	ECB (Electronic CodeBook) mode AEC algorithm

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.2. MI_CIPHER_HASH_ALGO_e

- Description

  Hash algorithm type

- Definition

```
typedef enum
{
    MI_CIPHER_HASH_ ALG_SHA1 ;
    MI_CIPHER_HASH_ ALG_SHA256 ;

} MI_CIPHER_HASH_ALGO_e;
```

- Member

Member	Description
MI_CIPHER_HASH_ ALG_SHA1	SHA1 Hash Algorithm
MI_CIPHER_HASH_ ALG_SHA256	SHA256 Hash Algorithm

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.3. MI_CIPHER_RSA_ENC_SCHEME_E

- Description

  Define RSA encrypt scheme.

- Definition

```
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;
```

- Member

Member	Description
MI_CIPHER_RSA_ENC_SCHEME_NO_PADDING	Without padding
MI_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA1	PKCS#1 RSAES-OAEP-SHA1 padding
MI_CIPHER_RSA_ENC_SCHEME_RSAES_OAEP_SHA256	PKCS#1 RSAES-OAEP-SHA256 padding
MI_CIPHER_RSA_ENC_SCHEME_RSAES_PKCS1_V1_5	PKCS#1 RSAES-PKCS1_V1_5 padding
MI_CIPHER_RSA_ENC_SCHEME_BUTT	Unknown

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.4. MI_CIPHER_RSA_SIGN_SCHEME_E

- Description

  Define RSA sign scheme.

- Definition

```
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;
```

- Member

Member	Description
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA1	PKCS#1 RSASSA_PKCS1_V15_SHA1 signature
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_V15_SHA256	PKCS#1 RSASSA_PKCS1_V15_SHA256 signature
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA1	PKCS#1 RSASSA_PKCS1_PSS_SHA1 signature
MI_CIPHER_RSA_SIGN_SCHEME_RSASSA_PKCS1_PSS_SHA256	PKCS#1 RSASSA_PKCS1_PSS_SHA256 signature
MI_CIPHER_RSA_SIGN_SCHEME_BUTT	Unknown

- Note

N/A.

- Related Data Type and Interface

N/A.

2.5. MI_CIPHER_KeySize_e

- Description

The Enum to specify Key Size

- Definition

```
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;
```

- Member

Member	Description
E_MI_CIPHER_KEY_SIZE_128	key size is 128-bit
E_MI_CIPHER_KEY_SIZE_192	key size is 192-bit
E_MI_CIPHER_KEY_SIZE_256	key size is 256-bit

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.6. MI_CIPHER_Config_t

- Description

  Define Cipher configuration structure

- Definition

```
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;
```

- Member

Member	Description
eKeySize	The Enum to specify the size of the key
key	The Array to store the key
iv	Initialization vector, AES_BLOCK_SIZE is 16
eAlg	Encryption/Decryption algorithm type

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.7. MI_CIPHER_RSA_PUB_Key_t

- Description

  Define public key data structure

- Definition

```
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;
```

- Member

Member	Description
pu8ExpE;	Pointer to store key Exponent data
pu8ModN	Pointer to store key Module data
expSize	Size of exponent data
modSize	Size of module data

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.8. MI_CIPHER_RSA_PRI_Key_t

- Description

  Define private key data structure

- Definition

```
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;
```

- Member

Member	Description
pu8ExpD	Pointer to store key Exponent data
pu8ModN	Pointer to store key Module data
expSize	Size of exponent data
modSize	Size of module data

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.9. MI_CIPHER_RSA_PUB_ENC_t

- Description

  Define public key encryption/decryption algorithm parameter structure

- Definition

```
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;
```

- Member

Member	Description
eRsaAlgoType	Encryption/Decryption algorithm type
stPubKey	Key data

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.10. MI_CIPHER_RSA_PRI_ENC_t

- Description

  Define public key encryption/decryption parameter structure

- Definition

```
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;
```

- Member

Member	Description
eRsaAlgoType	Encryption/Decryption algorithm type
stPriKey	Private key data

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.11. MI_CIPHER_RSA_SIGN_t

- Description

  Define RSA signature structure

- Definition

```
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;
```

- Member

Member	Description
eRsaAlgoType	Encryption/Decryption algorithm type
stPriKey	Private key data, for signature

- Note

  N/A.

- Related Data Type and Interface

  N/A.

2.12. MI_CIPHER_RSA_VERIFY_t

- Description

  Define public key encryption/decryption algorithm parameter structure

- Definition

```
typedef struct MI_CIPHER_RSA_Veriry_s
{
    MI_CIPHER_RSA_SIGN_SCHEME_E eRsaAlgoType;
    MI_CIPHER_RSA_PUB_Key_t stPubKey;
} MI_CIPHER_RSA_VERIRY_t;
```

- Member

Member	Description
eRsaAlgoType	Encryption/Decryption algorithm type
stPubKey	Public key data, for verification

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3. ERROR CODE

Table 1: CIPHER API error codes

Macro Definition	Description
MI_CIPHER_ERR_INVALID_DEVID	Invalid device number
MI_CIPHER_ERR_ILLEGAL_PARAM	Invalid parameter setting
MI_CIPHER_ERR_NOT_ENABLED	Device is not enabled
MI_CIPHER_ERR_NOT_DISABLED	Device is not disabled
MI_CIPHER_ERR_NULL_PTR	Using a NULL point
MI_CIPHER_ERR_INVALID_CHNID	Invalid Channel ID
MI_CIPHER_ERR_NOT_CONFIG	Device is not configured
MI_CIPHER_ERR_NOT_SUPPORT	No supported operation
MI_CIPHER_ERR_NOT_PERM	Operation is not permitted
MI_CIPHER_ERR_NOMEM	The Device lacks of memory
MI_CIPHER_ERR_NOBUF	Insufficient buffer
MI_CIPHER_ERR_BUF_EMPTY	Buffer is empty
MI_CIPHER_ERR_BUF_FULL	Buffer is full
MI_CIPHER_ERR_SYS_NOTREADY	System is not initialized
MI_CIPHER_ERR_BUSY	System is busy
MI_CIPHER_ERR_MOD_NOTINIT	Module not initialized before use
MI_CIPHER_ERR_MOD_INITED	Module already initialized
MI_CIPHER_ERR_FAILED	Unexpected error