CV180X & CV181X CIPHER API User Guide

Version: 1.2.0

Release date: 2022-10-19

Contents

Revision History

Revision	Date	Description
1.2.0	2022/10/19	Initial
1.2.1	2022/11/14	Review

1 Disclaimer

Terms and Conditions

The document and all information contained herein remain the CVITEK Co., Ltd' s ("CVITEK") confidential information, and should not disclose to any third party or use it in any way without CVITEK' s prior written consent. User shall be liable for any damage and loss caused by unauthority use and disclosure.

CVITEK reserves the right to make changes to information contained in this document at any time and without notice.

All information contained herein is provided in "AS IS" basis, without warranties of any kind, expressed or implied, including without limitation mercantability, non-infringement and fitness for a particular purpose. In no event shall CVITEK be liable for any third party' s software provided herein, User shall only seek remedy against such third party. CVITEK especially claims that CVITEK shall have no liable for CVITEK' s work result based on Customer' s specification or published shandard.

Contact Us

Address Building 1, Yard 9, FengHao East Road, Haidian District, Beijing, 100094, China

Building T10, UpperCoast Park, Huizhanwan, Zhancheng Community, Fuhai Street, Baoan District, Shenzhen, 518100, China

Phone +86-10-57590723 +86-10-57590724

Website https://www.sophgo.com/

Forum https://developer.sophgo.com/forum/index.html

2 Overview

2.1 Overview

CIPHER is a security algorithm module provided by CVITEK digital media processing platform. It provides symmetric encryption and decryption algorithms, including AES / DES / SM4, asymmetric encryption and decryption algorithm RSA (Note SW), random number generation, and digest algorithm, HASH and HMAC. It is mainly used for encryption and decryption protection of audio and video stream, authentication of user legitimacy and other scenarios. Each function is divided as follows:

Symmetric Encryption and Decryption Algorithm

- AES: supports ECB/CBC/CTR/CCM/GCM and other working modes. In CCM / GCM mode, the tag value needs to be obtained once after encryption and decryption.
- DES: supports ECB/CBC/CTR/CFB/OFB and other working modes. The bit width of CFB and OFB mode can be 1 / 8 / 64.
- SM4: supports ECB/CBC/CTR and other working modes.

Except CTR/CCM/GCM, the data length of the above algorithms and modes must be aligned by block size; N and A of CCM/GCM need to be encapsulated by software into block size aligned data blocks according to the standard.

Asymmetric Encryption and Decryption Algorithm

- RSA (Note SW): supports key bit width 1024/2048/3072/4096

RSA key width 1024 and below are known insecure algorithms in the industry and should not be used.

Random Number Generation

- RNG: Get random numbers at a high rate.

Digest Algorithm

- HASH : supports SHA1/SHA2/ SHA512/SM3 (Note SW);
- HMAC : support HMAC1/HMAC224/HMAC256/HMAC384/HMAC512 (Note SW);

The SHA1 algorithm is less secure and cannot be used in the scenario where it is involved in generating "digital signatures" , SHA2 (256 bits and above) algorithm is recommended.

2.2 Use Process

2.2.1 Single Packet Data Encryption and Decryption

Scene Description

When there is a piece of stream data in physical memory that needs to be encrypted/decrypted, the CIPHER module is called at the user level to encrypt/decrypt the single packet data after obtaining its physical address.

Working Process

The process of AES / DES / SM4 symmetric encryption and decryption is as follows:

Step 1 : CIPHER device initialization, call interface CVI_UNF_CIPHER_Init to complete.

Step 2 : get the CIPHER handle, call interface CVI_UNF_CIPHER_CreateHandle to complete.

Step 3 : configure the CIPHER control information, including key, initial vector, encryption algorithm, working mode and other information, call interface CVI_UNF_CIPHER_ConfigHandle or CVI_UNF_CIPHER_ConfigHandleEx to complete.

Step 4 : encrypt / decrypt the data. Call any of the following interfaces for encryption and decryption.

- Single packet encryption, CVI_UNF_CIPHER_Encrypt
- Single packet decryption, CVI_UNF_CIPHER_Decrypt

Step 5 : if CCM and GCM (Note SW) modes are used, Call interface CVI_UNF_CIPHER_GetTag to get the TAG value.

Step 6 : destroy the CIPHER handle, call interface CVI_UNF_CIPHER_DestroyHandle to complete.

Step 7 : shut down the CIPHER device, call interface CVI_UNF_CIPHER_Deinit to complete.

Note

2.2.2 Multi Packet Data Encryption and Decryption

Scene Description

When there are multiple streams of data in physical memory that need to be encrypted/decrypted, the CIPHER module is called at the user level to encrypt/decrypt multiple packets of data after obtaining their physical addresses.

Working Process

The process of AES / DES / SM4 symmetric encryption and decryption is as follows:

Step 1 : CIPHER device initialization, call interface CVI_UNF_CIPHER_Init to complete.

Step 2 : get the CIPHER handle, call interface CVI_UNF_CIPHER_CreateHandle to complete.

Step 3 : configure the CIPHER control information, including key, initial vector, encryption algorithm, working mode and other information, call interface CVI_UNF_CIPHER_ConfigHandle or CVI_UNF_CIPHER_ConfigHandleEx to complete.

Step 4 : encrypt / decrypt the data. Call any of the following interfaces for encryption and decryption.

- Multi packet encryption, CVI_UNF_CIPHER_EncryptMulti
- Multi packet decryption, CVI_UNF_CIPHER_DecryptMulti

Step 5 : destroy the CIPHER handle, call interface CVI_UNF_CIPHER_DestroyHandle to complete.

Step 6 : shut down CIPHER device, call interface CVI_UNF_CIPHER_Deinit to complete.

Note

2.2.3 HASH Calculation

Scene Description

The HASH value of calculated data can be SHA1/SHA2/ SHA512/SM3 (Note SW)

Working Process

Step 1 : CIPHER device initialization, call interface CVI_UNF_CIPHER_Init to complete.

Step 2 : get the HASH handle, select HASH calculation, and call interface CVI_UNF_CIPHER_HashInit to complete.

Step 3 : input data and calculate HASH value one by one. Call interface CVI_UNF_CIPHER_HashUpdate to complete.

Step 4 : if the summary is not calculated, perform step 3 again.

Step 5 : complete the summary calculation, end the input, and obtain the calculation results. Call interface CVI_UNF_CIPHER_HashFinal to complete.

Step 6 : shut down the CIPHER device, call interface CVI_UNF_CIPHER_Deinit to complete.

Note

2.2.4 HMAC Calculation (NOTE SW)

Scene Description

Based on HASH algorithm, HMAC value of data is calculated.

Working Process

Step 1 : CIPHER device initialization, call interface CVI_UNF_CIPHER_Init to complete.

Step 2 : get the HASH handle, select HASH calculation, configure the key of HMAC calculation , and call interface CVI_UNF_CIPHER_HashInit to complete.

Step 3 : input data and calculate HASH value one by one. Call interface CVI_UNF_CIPHER_HashUpdate to complete.

Step 4 : if the summary is not calculated, perform step 3 again.

Step 5 : complete the summary calculation, end the input, and obtain the calculation results. Call interface CVI_UNF_CIPHER_HashFinal to complete.

Step 6 : shut down the CIPHER device, call interface CVI_UNF_CIPHER_Deinit to complete.

Note

2.2.5 Generating Random Numbers

Scene Description

Get the true random number generated by the hardware.

Working Process

Step 1 : CIPHER device initialization, call interface CVI_UNF_CIPHER_Init to complete.

Step 2 : get 256bits random numbers, call interface CVI_UNF_CIPHER_GetRandomNumber to complete.

Step 3 : shut down the CIPHER device, call interface CVI_UNF_CIPHER_Deinit to complete.

Note

2.2.6 RSA Encryption and Decryption Operation Steps (NOTE SW)

Scene Description

The data is encrypted and decrypted by RSA asymmetric algorithm. Data encrypted by public key must be decrypted by private key. On the contrary, data encrypted by private key must be decrypted by public key.

Working Process

Step 1: CIPHER device initialization, call interface CVI_UNF_CIPHER_Init to complete.

Step 2: encryption and decryption or signature verification of data. Depending on the key used, call any of the following interfaces to perform encryption and decryption, signature verification, generate depending on the key used.

- Public key encryption : CVI_UNF_CIPHER_RsaPublicEncrypt
- Private key decryption : CVI_UNF_CIPHER_RsaPrivateDec
- Private key encryption : CVI_UNF_CIPHER_RsaPrivateEnc
- Public key decryption : CVI_UNF_CIPHER_RsaPublicDec
- Private key signature : CVI_UNF_CIPHER_RsaSign
- Public key verification : CVI_UNF_CIPHER_RsaVerify

Step 3 : shut down the CIPHER device, call interface CVI_UNF_CIPHER_Deinit to complete.

Note

2.2.7 RSA Signature and Signature Verification Procedures

Scene Description

When RSA is used to sign and verify the data, the private key is used to sign the data, and the public key is used to verify the data.

Working Process

Step 1 : CIPHER device initialization, call interface CVI_UNF_CIPHER_Init to complete.

Step 2 : verify the signature of the data.

- Private key signature : CVI_UNF_CIPHER_RsaSign
- Public key verification : CVI_UNF_CIPHER_RsaVerify

Step 3 : shut down the CIPHER device, call interface CVI_UNF_CIPHER_Deinit to complete.

Note

3 API Reference

CIPHER provides the following API:

- *CVI_UNF_CIPHER_Init* : initialize CIPHER module.
- *CVI_UNF_CIPHER_Deinit* : de-initialize CIPHER module.
- *CVI_UNF_CIPHER_Open* : open CIPHER module.
- *CVI_UNF_CIPHER_Close* : close CIPHER module.
- *CVI_UNF_CIPHER_CreateHandle* : create CIPHER handle.
- *CVI_UNF_CIPHER_DestroyHandle* : destroy existing CIPHER handle.
- *CVI_UNF_CIPHER_ConfigHandle*: configure CIPHER control information.
- *CVI_UNF_CIPHER_ConfigHandleEx* : configure CIPHER control information (extended).
- *CVI_UNF_CIPHER_GetHandleConfig* : get CIPHER configuration information.
- *CVI_UNF_CIPHER_Encrypt* : single packet data encryption function.
- *CVI_UNF_CIPHER_Decrypt* : single packet data decryption function.
- *CVI_UNF_CIPHER_EncryptVir* : encrypt the data.
- *CVI_UNF_CIPHER_DecryptVir* : decrypt the data.
- *CVI_UNF_CIPHER_EncryptMulti* : multi packet data encryption function.
- *CVI_UNF_CIPHER_DecryptMulti* : multi packet data decryption function.
- *CVI_UNF_CIPHER_HashInit* : HASH, HMAC calculate initialization function.
- *CVI_UNF_CIPHER_HashUpdate* : HASH, HMAC calculate data input function.
- *CVI_UNF_CIPHER_HashFinal* :HASH, HMAC calculate the final result output function.
- *CVI_UNF_CIPHER_GetRandomNumber* : get random number function.
- *CVI_UNF_CIPHER_GetTag* : get TAG value.
- *CVI_UNF_CIPHER_RsaPublicEncrypt* : use public key to encrypt the plaintext.
- *CVI_UNF_CIPHER_RsaPrivateDecrypt* : use private key decrypt the ciphertext.
- *CVI_UNF_CIPHER_RsaPublicDecrypt* : use public key decrypt the ciphertext.
- *CVI_UNF_CIPHER_RsaSign* : use private key to sign the user data.

- *CVI_UNF_CIPHER_RsaVerify*：use public key to verify the validity and integrity of user data.
- *CVI_UNF_CIPHER_KladEncryptKey*： use KLAD to encrypt the transparent key.

3.1 CVI_UNF_CIPHER_Init

【Description】

Initialize CIPHER module.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_Init(void);
```

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.2 CVI_UNF_CIPHER_Deinit

【Description】

De-initialize CIPHER module.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_DeInit(void);
```

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.3 CVI_UNF_CIPHER_Open

【Description】

Open CIPHER module.

【Syntax】

```
#define CVI_UNF_CIPHER_Open(CVI_VOID)

CVI_UNF_CIPHER_Init(CVI_VOID);
```

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.4 CVI_UNF_CIPHER_Close

【Description】

Close CIPHER module.

【Syntax】

```
#define CVI_UNF_CIPHER_Close(CVI_VOID)

CVI_UNF_CIPHER_DeInit(CVI_VOID);
```

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.5 CVI_UNF_CIPHER_CreateHandle

【Description】

Create single-way Cipher handle.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_CreateHandle(CVI_HANDLE * phCipher, const CVI_UNF_CIPHER_
↪ATTS_S * pstCipherAttr);
```

【Parameter】

Parameter	Description	Input/Output
phCipher	CIPHER handle pointer	Output
pstCipherAtt	CIPHER property pointer	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.6 CVI_UNF_CIPHER_DestroyHandle

【Description】

Destroy single-way CIPHER.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_DestroyHandle(CVI_HANDLE hCipher);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.7 CVI_UNF_CIPHER_ConfigHandle

【Description】

Configure CIPHER control information. Please refer to structure CVI_UNF_CIPHER_CTRL_S.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_ConfigHandle(CVI_HANDLE hCipher, CVI_UNF_CIPHER_CTRL_S*␣
↪pstCtrl);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
pstCtrl	Control information pointer	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.8 CVI_UNF_CIPHER_ConfigHandleEx

【Description】

Configure CIPHER control information. Please refer to structure CVI_UNF_CIPHER_CTRL_EX_S.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_ConfigHandleEx(CVI_HANDLE hCipher, CVI_UNF_CIPHER_CTRL_
↪EX_S* pstExCtrl);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
pstExCtrl	Control extended information pointer	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.9 CVI_UNF_CIPHER_GetHandleConfig

【Description】

Get channel CIPHER corresponding configuration information.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_GetHandleConfig(CVI_HANDLE hCipher, CVI_UNF_CIPHER_CTRL_
↪S* pstCtrl);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
pstCtrl	CIPHER channel configuration information	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.10 CVI_UNF_CIPHER_Encrypt

【Description】

Encrypt the data.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_Encrypt(CVI_HANDLE hCipher, CVI_U32 u32SrcPhyAddr, CVI_
↪U32 u32DestPhyAddr, CVI_U32 u32ByteLength);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
u32SrcPhyAddr	The physical address of the source data (the data to be encrypted)	Input
u32DestPhyAddr	The physical address where the encryption results are stored	Input
u32ByteLength	The length of the data (in bytes)	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.11 CVI_UNF_CIPHER_Decrypt

【Description】

Decrypt the data.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_Decrypt(CVI_HANDLE hCipher, CVI_U32 u32SrcPhyAddr, CVI_
↪U32 u32DestPhyAddr, CVI_U32 u32ByteLength);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
u32SrcPhyAddr	The physical address of the source data (the data to be decrypted)	Input
u32DestPhyAddr	The physical address where the decryption results are stored	Input
u32ByteLength	The length of the data (in bytes)	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.12 CVI_UNF_CIPHER_EncryptVir

【Description】

Encrypt the data.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_EncryptVir(CVI_HANDLE hCipher, const CVI_U8 * pu8SrcData,
↪ CVI_U8 * pu8DestData, CVI_U32 u32ByteLength);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
*pu8SrcData	The virtual address of the source data (the data to be encrypted)	Input
*pu8DestData	The virtual address where the encryption results are stored	Output
u32ByteLength	The length of the data (in bytes)	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.13 CVI_UNF_CIPHER_DecryptVir

【Description】

Decrypt the data.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_DecryptVir(CVI_HANDLE hCipher, const CVI_U8 * pu8SrcData,
↪ CVI_U8 * pu8DestData, CVI_U32 u32ByteLength);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
*pu8SrcData	The virtual address of the source data (the data to be decrypted)	Input
*pu8DestData	The virtual address where the decryption results are stored	Output
u32ByteLength	The length of the data (in bytes)	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.14 CVI_UNF_CIPHER_EncryptMulti

【Description】

Encrypt multiple packet data.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_EncryptMulti(CVI_HANDLE hCipher, CVI_UNF_CIPHER_DATA_S *␣
↪pstDataPkg, CVI_U32 u32DataPkgNum);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
*pstDataPkg	Packets to be encrypted	Input
u32DataPkgNum	The number of packets to be encrypted	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.15 CVI_UNF_CIPHER_DecryptMulti

【Description】

Decrypt multiple packet data.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_DecryptMulti(CVI_HANDLE hCipher, CVI_UNF_CIPHER_DATA_S *␣
↪pstDataPkg, CVI_U32 u32DataPkgNum);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
*pstDataPkg	Packets to be decrypted	Input
u32DataPkgNum	The number of packets to be decrypted	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.16 CVI_UNF_CIPHER_HashInit

【Description】

Initialize HASH module.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_HashInit(CVI_UNF_CIPHER_HASH_ATTS_S * pstHashAttr, CVI_
↪HANDLE * pHashHandle);
```

【Parameter】

Parameter	Description	Input/Output
pstHashAttr	Structure parameters for hash calculation	Input
pHashHandle	The hash handle of the output	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.17 CVI_UNF_CIPHER_HashUpdate

【Description】

Calculate hash value.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_HashUpdate(CVI_HANDLE hHashHandle, CVI_U8 * pu8InputData,
↪ CVI_U32 u32InputDataLen);
```

【Parameter】

Parameter	Description	Input/Output
hHashHandl	Hash handle	Input
pu8InputData	Input data buffer	Input
u32InputDataLen	The length of the input data (in bytes)	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.18 CVI_UNF_CIPHER_HashFinal

【Description】

Get hash value.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_HashFinal(CVI_HANDLE hHashHandle, CVI_U8 *␣
↪pu8OutputHash);
```

【Parameter】

Parameter	Description	Input/Output
hHashHandl	Hash handle	Input
pu8OutputHash	Output 的 hash	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.19 CVI_UNF_CIPHER_GetRandomNumber

【Description】

Generate random numbers.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_GetRandomNumber(CVI_U32 * pu32RandomNumber);
```

【Parameter】

Parameter	Description	Input/Output
pu32RandomNumber	Random number of output	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.20 CVI_UNF_CIPHER_GetTag

【Description】

The TAG value is obtained after encryption and decryption in CCM / GCM mode.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_GetTag(CVI_HANDLE hCipher, CVI_U8 * pstTag);
```

【Parameter】

Parameter	Description	Input/Output
hCipher	CIPHER handle	Input
pstTag	TAG value	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.21 CVI_UNF_CIPHER_RsaPublicEncrypt

【Description】

Use RSA public key to encrypt a piece of plaintext.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_RsaPublicEncrypt(CVI_UNF_CIPHER_RSA_PUB_ENC_S *␣
↪pstRsaEnc, CVI_U8 * pu8Input, CVI_U32 u32InLen, CVI_U8 * pu8Output, CVI_U32␣
↪CIPHER * pu32OutLen);
```

【Parameter】

Parameter	Description	Input/Output
pstRsaEnc	Public key encryption attribute structure	Input
pu8Input	Data to be encrypted	Input
u32InLen	The length of the data to be encrypted (in bytes)	Input
pu8Output	Encrypted results data	Output
pu32OutLen	The length of the encrypted results data (in bytes)	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.22 CVI_UNF_CIPHER_RsaPrivateDecrypt

【Description】

Use RSA private key to decrypt a ciphertext.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_RsaPrivateDecrypt(CVI_UNF_CIPHER_RSA_PRI_ENC_S *␣
↪pstRsaDec, CVI_U8 * pu8Input, CVI_U32 u32InLen, CVI_U8 * pu8Output, CVI_U32 *␣
↪pu32OutLen);
```

【Parameter】

Parameter	Description	Input/Output
pstRsaEnc	Private key encryption attribute structure	Input
pu8Input	Data to be encrypted	Input
u32InLen	The length of the data to be encrypted (in bytes)	Input
pu8Output	Encrypted results data	Output
pu32OutLen	The length of the encrypted results data (in bytes)	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.23 CVI_UNF_CIPHER_RsaPublicDecrypt

【Description】

Use RSA public key to decrypt a ciphertext.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_RsaPrivateDecrypt(CVI_UNF_CIPHER_RSA_PUB_ENC_S *␣
↪pstRsaDec, CVI_U8 * pu8Input, CVI_U32 u32InLen, CVI_U8 * pu8Output, CVI_U32 *␣
↪pu32OutLen);
```

【Parameter】

Parameter	Description	Input/Output
pstRsaDec	Public key decryption attribute structure	Input
pu8Input	Data to be decrypted	Input
u32InLen	The length of the data to be decrypted (in bytes)	Input
pu8Output	Decrypted results data	Output
pu32OutLen	The length of the decrypted results data (in bytes)	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.24 CVI_UNF_CIPHER_RsaSign

【Description】

Sign a piece of text with RSA private key.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_RsaSign(CVI_UNF_CIPHER_RSA_SIGN_S * pstRsaSign, CVI_U8 *␣
↪pu8InData, CVI_U32 u32InDataLen, CVI_U8 * pu8HashData,CVI_U8 * pu8OutSign,␣
↪CVI_U32 * pu32OutSignLen);
```

【Parameter】

Parameter	Description	Input/Output
pstRsaSign	Signature attribute structure	Input
pu8InData	The data to be signed, if pu8HashData is not empty, pu8HashData will be used for signing and this parameter will be ignored	Input
u32InDataLen	The length of the data to be signed (in bytes)	Input
pu8HashData	HASH digest of the text to be signed, if it is empty, the HASH digest of pu8InData is automatically calculated for signing	Input
pu8OutSign	Signature results data	Output
pu32OutSignLen	The length of the signature results data (in bytes)	Output

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.25 CVI_UNF_CIPHER_RsaVerify

【Description】

Use RSA public key signature to verify a piece of text.

【Syntax】

```
CVI_S32 CVI_UNF_CIPHER_RsaVerify(CVI_UNF_CIPHER_RSA_VERIFY_S * pstRsaVerify,CVI_
↪U8 * pu8InData, CVI_U32 u32InDataLen, CVI_U8 * pu8HashData,CVI_U8 * pu8InSign,
↪ CVI_U32 u32InSignLen);
```

【Parameter】

Parameter	Description	Input/Output
pstRsaVerify	Signature verification attribute structure	Input
pu8InData	Data to be validated, if pu8HashData is not empty, then use pu8HashData for validation, this parameter will be ignored	Input
u32InDataLen	The length of the data to be verified (in bytes)	Input
pu8HashData	HASH digest of the text to be validated, if empty, the HASH digest of pu8InData is automatically calculated for validation	Input
pu8InSign	Signature data to be verified	Input
u32InSignLen	The length of the signature data to be verified (in bytes)	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

3.26 CVI_UNF_CIPHER_KladEncryptKey

【Description】

Use KLAD to encrypt transparent keys.

【Syntax】

```
CV_S32 CVI_UNF_CIPHER_KladEncryptKey(CVI_UNF_CIPHER_CA_TYPE_E enRootKey, CVI_
↪UNF_CIPHER_KLAD_TARGET_E enTarget, CVI_U8 * pu8CleanKey, CVI_U8 *␣
↪pu8EcnryptKey, CVI_U32 u32KeyLen);
```

【Parameter】

Parameter	Description	Input/Output
enRootKey	KLAD root key selection, only EFUSE Key can be selected	Input
enTarget	Modules using this key	Input
pu8CleanKey	Transparent Key	Input
pu8EcnryptKey	Encryption Key	Output
u32KeyLen	The length of the key, which must be an integer multiple of 16	Input

【Return Value】

Return Value	Description
0	Success
Not 0	Refer to *Error Codes*.

4 Data Type

Relevant data types and data structures are defined as follows:

- CVI_HANDLE: define the handle type of the CIPHER.
- CVI_UNF_CIPHER_WORK_MODE_E: define the working mode of the CIPHER.
- CVI_UNF_CIPHER_ALG_E: define the encryption algorithm of the CIPHER.
- CVI_UNF_CIPHER_KEY_LENGTH_E: define the CIPHER key length.
- CVI_UNF_CIPHER_BIT_WIDTH_E: define the CIPHER bit width.
- CVI_UNF_CIPHER_CTRL_CHANGE_FLAG_S: define the information structure of CIPHER CCM pattern.
- CVI_UNF_CIPHER_CA_TYPE_E: define the source of the CIPHER key.
- CVI_UNF_CIPHER_KLAD_TARGET_E: define the target selection for the delivery of the Key generated by Klad.
- CVI_UNF_CIPHER_TYPE_E: define CIPHER encryption and decryption type selection.
- CVI_UNF_CIPHER_ATTS_S: define CIPHER encryption and decryption type structure.
- CVI_UNF_CIPHER_CTRL_S: define CIPHER control information structure.
- CVI_UNF_CIPHER_CTRL_AES_S: AES encryption control information structure expansion.
- CVI_UNF_CIPHER_CTRL_AES_CCM_GCM_S: AES-CCM, AES-GCM encryption control information structure.
- CVI_UNF_CIPHER_CTRL_DES_S: DES encryption control message structure extension.
- CVI_UNF_CIPHER_CTRL_3DES_S: 3DES encryption control information structure.
- CVI_UNF_CIPHER_CTRL_EX_S: The extended structure of encrypted control information is used as the special parameter of the algorithm.
- CVI_UNF_CIPHER_DATA_S: define CIPHER encrypted and decrypted data.
- CVI_UNF_CIPHER_HASH_TYPE_E: define the type of CIPHER hash algorithm.
- CVI_UNF_CIPHER_HASH_ATTS_S: define the CIPHER hash algorithm to initialize the input structure.
- CVI_UNF_CIPHER_RSA_ENC_SCHEME_E: define RSA algorithm data encryption filling method.

- CVI_UNF_CIPHER_RSA_SIGN_SCHEME_E: define RSA data signature policy.
- CVI_UNF_CIPHER_RSA_PUB_KEY_S: define RSA public key structure.
- CVI_UNF_CIPHER_RSA_PRI_KEY_S: define RSA private key structure.
- CVI_UNF_CIPHER_RSA_PUB_ENC_S: define the parameter structure of RSA public key encryption and decryption algorithm.
- CVI_UNF_CIPHER_RSA_PRI_ENC_S: define the parameter structure of RSA private key encryption and decryption algorithm.
- CVI_UNF_CIPHER_RSA_SIGN_S: define RSA signature algorithm parameter input structure.
- CVI_UNF_CIPHER_RSA_VERIFY_S: define the parameter input structure of RSA signature verification algorithm.
- CIPHER_IV_CHANGE_ONE_PKG: when CIPHER sets vectors for packets, only the IV of one packet is updated
- CIPHER_IV_CHANGE_ALL_PKG: when CIPHER sets vectors for packets, the IV of all packets are updated.

5 Error Code

Errore code	Macro definition	Description
0x804D0001	CVI_ERR_CIPHER_NOT_INIT	Device not initialized
0x804D0002	CVI_ERR_CIPHER_INVALID_HANDLE	Invalid handle number
0x804D0003	CVI_ERR_CIPHER_INVALID_POINT	Null pointer in parameter
0x804D0004	CVI_ERR_CIPHER_INVALID_PARA	Invalid parameter
0x804D0005	CVI_ERR_CIPHER_FAILED_INIT	Initialization failed
0x804D0006	CVI_ERR_CIPHER_FAILED_GETHANDLE	Failed to get handle
0x804D0007	CVI_ERR_CIPHER_FAILED_RELEASEHANDLE	Release handle failed
0x804D0008	CVI_ERR_CIPHER_FAILED_CONFIGAES	Invalid AES configuration
0x804D0009	CVI_ERR_CIPHER_FAILED_CONFIGDES	Invalid DES configuration
0x804D000A	CVI_ERR_CIPHER_FAILED_ENCRYPT	Encryption failed
0x804D000B	CVI_ERR_CIPHER_FAILED_DECRYPT	Decryption failed
0x804D000C	CVI_ERR_CIPHER_BUSY	Busy status
0x804D000D	CVI_ERR_CIPHER_NO_AVAILABLE_RNG	No random numbers available
0x804D000E	CVI_ERR_CIPHER_FAILED_MEM	Memory request error
0x804D000F	CVI_ERR_CIPHER_UNAVAILABLE	Unavailable
0x804D0010	CVI_ERR_CIPHER_OVERFLOW	Data overflow
0x804D0011	CVI_ERR_CIPHER_HARD_STATUS	Hardware status error
0x804D0012	CVI_ERR_CIPHER_TIMEOUT	Waiting timeout
0x804D0013	CVI_ERR_CIPHER_UNSUPPORTED	Unsupported configuration
0x804D0014	CVI_ERR_CIPHER_REGISTER_IRQ	Invalid interrupt number
0x804D0015	CVI_ERR_CIPHER_ILLEGAL_UUID	Illegal UUID
0x804D0016	CVI_ERR_CIPHER_ILLEGAL_KEY	Illegal key
0x804D0017	CVI_ERR_CIPHER_INVALID_ADDR	Invalid address
0x804D0018	CVI_ERR_CIPHER_INVALID_LENGTH	Invalid length
0x804D0019	CVI_ERR_CIPHER_ILLEGAL_DATA	Invalid data
0x804D001A	CVI_ERR_CIPHER_RSA_SIGN	RSA signature failed
0x804D001B	CVI_ERR_CIPHER_RSA_VERIFY	RSA verification failed
0x804D001E	CVI_ERR_CIPHER_RSA_CRYPT_FAILED	RSA encryption and decryption failed
-1	CVI_FAILURE	Operation failed
0x004D0001	CVI_LOG_ERR_MEM	Memory operation failed
0x004D0002	CVI_LOG_ERR_SEM	Semaphore operation failed
0x004D0003	CVI_LOG_ERR_FILE	File operation failed
0x004D0004	CVI_LOG_ERR_LOCK	Lock operation failed
0x004D0005	CVI_LOG_ERR_PARAM	Invalid parameter
0x004D0006	CVI_LOG_ERR_TIMER	Timer error

continues on next pag

Table 5.1 – continued from previous page

Errore code	Macro definition	Description
0x004D0007	CVI_LOG_ERR_THREAD	Thread failure
0x004D0008	CVI_LOG_ERR_TIMEOUT	Timeout
0x004D0009	CVI_LOG_ERR_DEVICE	Device operation failure
0x004D0010	CVI_LOG_ERR_STATUS	Status error
0x004D0011	CVI_LOG_ERR_IOCTRL	IO operation failure
0x004D0012	CVI_LOG_ERR_INUSE	Resource in use
0x004D0013	CVI_LOG_ERR_EXIST	Exist failed
0x004D0014	CVI_LOG_ERR_NOEXIST	Resource not exited
0x004D0015	CVI_LOG_ERR_UNSUPPORTED	Unsupported
0x004D0016	CVI_LOG_ERR_UNAVAILABLE	Unavailable
0x004D0017	CVI_LOG_ERR_UNINITED	Uninited
0x004D0018	CVI_LOG_ERR_DATABASE	Database error
0x004D0019	CVI_LOG_ERR_OVERFLOW	Overflow
0x004D0020	CVI_LOG_ERR_EXTERNAL	External error
0x004D0021	CVI_LOG_ERR_UNKNOWNED	Location error
0x004D0022	CVI_LOG_ERR_FLASH	Flash operation error
0x004D0023	CVI_LOG_ERR_ILLEGAL_IMAGE	Illegal Mirror
0x004D0024	CVI_LOG_ERR_ILLEGAL_UUID	Illegal UUID
0x004D0025	CVI_LOG_ERR_NOPERMISSION	Operation not permitted