CV180X & CV181X CIPHER API 使用手册

Version: 1.2.0

Release date: 2022-10-19

目录

修订记录

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

1 声明

法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。

本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 https://www.sophgo.com/

技术论坛 https://developer.sophgo.com/forum/index.html

2 概述

2.1 概述

CIPHER 是晶视智能数字媒体处理平台提供的安全算法模块，提供对称式加解密算法包括 AES/DES/SM4, 不对称加解密算法 RSA(Note SW), 随机数生成, 以及摘要算法包括 HASH, HMAC, 主要用于对音视频码流进行加解密保护, 认证用户合法性等场景, 各功能划分如下:

对称加密算法

- AES：支持 ECB/CBC/CTR/ CCM/GCM (Note SW) 等工作模式，其中 CCM/GCM 模式下, 加解密结束后需获取一次 TAG 值。
- DES：支持 ECB/CBC/CTR/ CFB/OFB (Note SW) 等工作模式，其中 CFB 和 OFB 模式支持的位宽可为 1/8/64
- SM4: 支持 ECB/CBC/CTR 等工作模式

以上算法除了 CTR/ CCM/GCM, 其它算法、模式的数据长度必须按块大小对齐；CCM/GCM 的 N、A 需要靠软件按标准把各个字段封装成块大小对齐的数据块；

不对称加解密算法

- RSA (Note SW)：支持密钥位宽 1024/2048/3072/4096

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：高速率获取随机数

摘要算法

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

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。

2.2 使用流程

2.2.1 单包数据加解密

场景说明

当物理内存中有一段码流数据需要进行加/解密时，获取其物理地址后，在用户层调用 CIPHER 模块实现单包数据加/解密。

工作流程

对数据进行对称的 AES/DES/SM4 加解密的过程如下：

步骤 1：CIPHER 设备初始化。调用接口 CVI_UNF_CIPHER_Init 完成。

步骤 2：获取 CIPHER 句柄。调用接口 CVI_UNF_CIPHER_CreateHandle 完成。

步骤 3：配置 CIPHER 控制信息，包含密钥、初始向量、加密算法、工作模式等信息。调用接口 CVI_UNF_CIPHER_ConfigHandle 或 CVI_UNF_CIPHER_ConfigHandleEx 完成。

步骤 4：对数据进行加/解密。调用以下任一接口进行加解密。

- 单包加密，CVI_UNF_CIPHER_Encrypt
- 单包解密，CVI_UNF_CIPHER_Decrypt

步骤 5：若使用 CCM、GCM (Note SW) 模式，调用接口 CVI_UNF_CIPHER_GetTag 获取 TAG 值。

步骤 6：销毁 CIPHER 句柄。调用接口 CVI_UNF_CIPHER_DestroyHandle 完成。

步骤 7：关闭 CIPHER 设备。调用接口 CVI_UNF_CIPHER_Deinit 完成。

注意事项

2.2.2 多包数据加解密

场景说明

当物理内存中有多段码流数据需要进行加/解密时，获取其物理地址后，在用户层调用 CIPHER 模块实现多包数据加/解密。

工作流程

对数据进行对称的 AES/DES/SM4 加解密的过程如下：

步骤 1：CIPHER 设备初始化。调用接口 CVI_UNF_CIPHER_Init 完成。

步骤 2：获取 CIPHER 句柄。调用接口 CVI_UNF_CIPHER_CreateHandle 完成。

步骤 3：配置 CIPHER 控制信息，包含密钥、初始向量、加密算法、工作模式等信息。调用接口 CVI_UNF_CIPHER_ConfigHandle 或 CVI_UNF_CIPHER_ConfigHandleEx 完成。

步骤 4：对数据进行加/解密。调用以下任一接口进行加解密。

- 多包加密，CVI_UNF_CIPHER_EncryptMulti
- 多包解密，CVI_UNF_CIPHER_DecryptMulti

步骤 5：销毁 CIPHER 句柄。调用接口 CVI_UNF_CIPHER_DestroyHandle 完成。

步骤 6：关闭 CIPHER 设备。调用接口 CVI_UNF_CIPHER_Deinit 完成。

注意事项

2.2.3 HASH 计算

场景说明

计算数据的 HASH 值, 可选择 SHA1/SHA2/ SHA512/SM3 (Note SW)

工作流程

步骤 1：CIPHER 设备初始化。调用接口 CVI_UNF_CIPHER_Init 完成。

步骤 2：获取 HASH 句柄, 选择 HASH 算法。调用接口 CVI_UNF_CIPHER_HashInit 完成。

步骤 3：输入数据, 逐个数据块依次计算 HASH 值。调用接口 CVI_UNF_CIPHER_HashUpdate 完成。

步骤 4：如果摘要未计算完成，再次执行步骤 3。

步骤 5：完成摘要计算，结束输入，获取计算结果。调用接口 CVI_UNF_CIPHER_HashFinal 完成。

步骤 6：关闭 CIPHER 设备。调用接口 CVI_UNF_CIPHER_Deinit 完成。

注意事项

2.2.4 HMAC 计算 (NOTE SW)

场景说明

基于 HASH 算法, 计算数据的 HMAC 值。

工作流程

步骤 1：CIPHER 设备初始化。调用接口 CVI_UNF_CIPHER_Init 完成。

步骤 2：获取 HASH 句柄，选择 HASH 算法并配置 HMAC 计算的密钥，调用接口 CVI_UNF_CIPHER_HashInit 完成。

步骤 3：输入数据, 逐个数据块依次计算 HMAC 值。调用接口 CVI_UNF_CIPHER_HashUpdate 完成。

步骤 4：如果摘要未计算完成，再次执行步骤 3。

步骤 5：完成摘要计算，结束输入，获取 HMAC 计算结果。调用接口 CVI_UNF_CIPHER_HashFinal 完成。

步骤 6：关闭 CIPHER 设备。调用接口 CVI_UNF_CIPHER_Deinit 完成。

注意事项

2.2.5 产生随机数

场景说明

获取硬件产生的真随机数

工作流程

步骤 1：CIPHER 设备初始化。调用接口 CVI_UNF_CIPHER_Init 完成。

步骤 2：获取 256bits 随机数, 调用接口 CVI_UNF_CIPHER_GetRandomNumber 完成。

步骤 3：关闭 CIPHER 设备。调用接口 CVI_UNF_CIPHER_Deinit 完成。

注意事项

2.2.6 RSA 加解密操作步骤 (NOTE SW)

场景说明

对数据进行 RSA 不对称算法加解密。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，必须使用公钥解密。

工作流程

步骤 1：CIPHER 设备初始化。调用接口 CVI_UNF_CIPHER_Init 完成。

步骤 2：对数据进行加解密或签名验证。根据使用的密钥不同，调用以下任一接口进行加解密、签名验证、生成密钥对等。

- 公钥加密: CVI_UNF_CIPHER_RsaPublicEncrypt
- 私钥解密: CVI_UNF_CIPHER_RsaPrivateDec
- 私钥加密: CVI_UNF_CIPHER_RsaPrivateEnc
- 公钥解密: CVI_UNF_CIPHER_RsaPublicDec
- 私钥签名: CVI_UNF_CIPHER_RsaSign
- 公钥验证: CVI_UNF_CIPHER_RsaVerify

步骤 3：关闭 CIPHER 设备。调用接口 CVI_UNF_CIPHER_Deinit 完成。

注意事项

2.2.7 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，使用进行数据验签。

工作流程

步骤 1：CIPHER 设备初始化。调用接口 CVI_UNF_CIPHER_Init 完成。

步骤 2：对数据进行签名验证。

- 私钥签名: CVI_UNF_CIPHER_RsaSign
- 公钥验证: CVI_UNF_CIPHER_RsaVerify

步骤 3：关闭 CIPHER 设备。调用接口 CVI_UNF_CIPHER_Deinit 完成。

注意事项

3 API 参考

CIPHER 提供以下 API:

- CVI_UNF_CIPHER_Init：初始化 CIPHER 模块。
- CVI_UNF_CIPHER_Deinit：去初始化 CIPHER 模块。
- CVI_UNF_CIPHER_Open：打开 CIPHER 模块。
- CVI_UNF_CIPHER_Close：关闭 CIPHER 模块。
- CVI_UNF_CIPHER_CreateHandle：创建 Cipher 句柄。
- CVI_UNF_CIPHER_DestroyHandle：销毁已存在的 CIPHER 句柄。
- CVI_UNF_CIPHER_ConfigHandle：配置 CIPHER 控制信息。
- CVI_UNF_CIPHER_ConfigHandleEx：配置 CIPHER 控制信息（扩展）。
- CVI_UNF_CIPHER_GetHandleConfig：获取 CIPHER 配置信息。
- CVI_UNF_CIPHER_Encrypt：单包数据加密功能。
- CVI_UNF_CIPHER_Decrypt：单包数据解密功能。
- CVI_UNF_CIPHER_EncryptVir：对数据进行加密。
- CVI_UNF_CIPHER_DecryptVir：对数据进行解密。
- CVI_UNF_CIPHER_EncryptMulti：多包数据加密功能。
- CVI_UNF_CIPHER_DecryptMulti：多包数据解密功能。
- CVI_UNF_CIPHER_HashInit :HASH、HMAC 计算初始化功能。
- CVI_UNF_CIPHER_HashUpdate :HASH、HMAC 计算数据输入功能。
- CVI_UNF_CIPHER_HashFinal :HASH、HMAC 计算最终结果输出功能。
- CVI_UNF_CIPHER_GetRandomNumber：获取随机数功能。
- CVI_UNF_CIPHER_GetTag：获取 TAG 值。
- CVI_UNF_CIPHER_RsaPublicEncrypt：使用公钥对明文进行加密。
- CVI_UNF_CIPHER_RsaPrivateDecrypt：使用私钥对密文进行解密。
- CVI_UNF_CIPHER_RsaPublicDecrypt：使用公钥对密文进行解密。
- CVI_UNF_CIPHER_RsaSign：使用私钥对用户数据进行签名。
- CVI_UNF_CIPHER_RsaVerify：使用公钥对用户数据进行合法性及完整性验证。

· CVI_UNF_CIPHER_KladEncryptKey：使用 KLAD 对透明密钥进行加密。

3.1 CVI_UNF_CIPHER_Init

【描述】

初始化 CIPHER 模块。

【语法】

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

【参数】

无。

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.2 CVI_UNF_CIPHER_Deinit

【描述】

去初始化 CIPHER 模块。

【语法】

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

【参数】

无。

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.3 CVI_UNF_CIPHER_Open

【描述】

打开 CIPHER 模块。

【语法】

```
#define CVI_UNF_CIPHER_Open(CVI_VOID)

CVI_UNF_CIPHER_Init(CVI_VOID);
```

【参数】

无。

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.4 CVI_UNF_CIPHER_Close

【描述】

关闭 CIPHER 模块。

【语法】

```
#define CVI_UNF_CIPHER_Close(CVI_VOID)

CVI_UNF_CIPHER_DeInit(CVI_VOID);
```

【参数】

无。

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.5 CVI_UNF_CIPHER_CreateHandle

【描述】

创建一路的 Cipher 句柄。

【语法】

```
CVI_S32 CVI_UNF_CIPHER_CreateHandle(CVI_HANDLE *phCipher, const CVI_UNF_
↪CIPHER_ATTS_S *pstCipherAttr);
```

【参数】

参数名称	描述	输入/输出
phCipher	CIPHER 句柄指针	输出
pstCipherAtt	CIPHER 属性指针	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.6 CVI_UNF_CIPHER_DestroyHandle

【描述】

销毁一路 CIPHER。

【语法】

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

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.7 CVI_UNF_CIPHER_ConfigHandle

【描述】

配置 CIPHER 控制信息。详细配置请参见结构体 CVI_UNF_CIPHER_CTRL_S。

【语法】

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

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
pstCtrl	控制信息指针	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.8 CVI_UNF_CIPHER_ConfigHandleEx

【描述】

配置 CIPHER 控制信息。详细配置请参见结构体 CVI_UNF_CIPHER_CTRL_EX_S。

【语法】

```
CVI_S32 CVI_UNF_CIPHER_ConfigHandleEx(CVI_HANDLE hCipher, CVI_UNF_CIPHER_
↪CTRL_EX_S* pstExCtrl);
```

【参数】

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

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.9 CVI_UNF_CIPHER_GetHandleConfig

【描述】

获取 CIPHER 信道对应的配置信息。

【语法】

```
CVI_S32 CVI_UNF_CIPHER_GetHandleConfig(CVI_HANDLE hCipher, CVI_UNF_CIPHER_
↪CTRL_S* pstCtrl);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
pstCtrl	CIPHER 信道的配置信息	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.10 CVI_UNF_CIPHER_Encrypt

【描述】

对数据进行加密。

【语法】

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

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
u32SrcPhyAddr	源数据（待加密的数据）的物理地址	输入
u32DestPhyAddr	存放加密结果的物理地址	输入
u32ByteLength	数据的长度（单位: 字节）	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.11 CVI_UNF_CIPHER_Decrypt

【描述】

对数据进行解密。

【语法】

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

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
u32SrcPhyAddr	源数据（待解密的数据）的物理地址	输入
u32DestPhyAddr	存放解密结果的物理地址	输入
u32ByteLength	数据的长度（单位: 字节）	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.12 CVI_UNF_CIPHER_EncryptVir

【描述】

对数据进行加密。

【语法】

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

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
*pu8SrcData	源数据（待加密的数据）的虚拟地址。	输入
*pu8DestData	存放加密结果的虚拟地址	输出
u32ByteLength	数据的长度（单位: 字节）	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.13 CVI_UNF_CIPHER_DecryptVir

【描述】

对数据进行解密。

【语法】

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

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
*pu8SrcData	源数据（待解密的数据）的虚拟地址。	输入
*pu8DestData	存放解密结果的虚拟地址	输出
u32ByteLength	数据的长度（单位: 字节）	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.14 CVI_UNF_CIPHER_EncryptMulti

【描述】

进行多个包数据的加密。

【语法】

```
CVI_S32 CVI_UNF_CIPHER_EncryptMulti(CVI_HANDLE hCipher, CVI_UNF_CIPHER_
↪DATA_S *pstDataPkg, CVI_U32 u32DataPkgNum);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
*pstDataPkg	待加密的数据包	输入
u32DataPkgNum	待加密的数据包个数	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.15 CVI_UNF_CIPHER_DecryptMulti

【描述】

进行多个包数据的解密。

【语法】

```
CVI_S32 CVI_UNF_CIPHER_DecryptMulti(CVI_HANDLE hCipher, CVI_UNF_CIPHER_
↪DATA_S *pstDataPkg, CVI_U32 u32DataPkgNum);
```

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
*pstDataPkg	待解密的数据包	输入
u32DataPkgNum	待解密的数据包个数	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.16 CVI_UNF_CIPHER_HashInit

【描述】

初始化 HASH 模块。

【语法】

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

【参数】

参数名称	描述	输入/输出
pstHashAttr	用于计算 hash 的结构体参数	输入
pHashHandle	输出的 hash 句柄	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.17 CVI_UNF_CIPHER_HashUpdate

【描述】

计算 hash 值。

【语法】

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

【参数】

参数名称	描述	输入/输出
hHashHandl	Hash 句柄	输入
pu8InputData	输入数据缓冲	输入
u32InputDataLen	输入数据的长度，单位:byte	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.18 CVI_UNF_CIPHER_HashFinal

【描述】

获取 hash 值。

【语法】

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

【参数】

参数名称	描述	输入/输出
hHashHandl	Hash 句柄	输入
pu8OutputHash	输出的 hash	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.19 CVI_UNF_CIPHER_GetRandomNumber

【描述】

生成随机数。

【语法】

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

【参数】

参数名称	描述	输入/输出
pu32RandomNumber	输出的随机数	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.20 CVI_UNF_CIPHER_GetTag

【描述】

CCM/GCM 模式加解密后获取 TAG 值。

【语法】

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

【参数】

参数名称	描述	输入/输出
hCipher	CIPHER 句柄	输入
pstTag	TAG 值	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.21 CVI_UNF_CIPHER_RsaPublicEncrypt

【描述】

使用 RSA 公钥加密一段明文。

【语法】

```
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);
```

【参数】

参数名称	描述	输入/输出
pstRsaEnc	公钥加密属性结构体	输入
pu8Input	待加密的数据	输入
u32InLen	待加密的数据长度，单位:byte	输入
pu8Output	加密结果数据	输出
pu32OutLen	加密结果数据长度，单位:byte	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.22 CVI_UNF_CIPHER_RsaPrivateDecrypt

【描述】

使用 RSA 私钥解密一段密文。

【语法】

```
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);
```

【参数】

参数名称	描述	输入/输出
pstRsaEnc	私钥加密属性结构体	输入
pu8Input	待加密的数据	输入
u32InLen	待加密的数据长度，单位:byte	输入
pu8Output	加密结果数据	输出
pu32OutLen	加密结果数据长度，单位:byte	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.23 CVI_UNF_CIPHER_RsaPublicDecrypt

【描述】

使用 RSA 公钥解密一段密文。

【语法】

```
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);
```

【参数】

参数名称	描述	输入/输出
pstRsaDec	公钥解密属性结构体	输入
pu8Input	待解密的数据	输入
u32InLen	待解密的数据长度，单位:byte	输入
pu8Output	解密结果数据	输出
pu32OutLen	解密结果数据长度，单位:byte	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.24 CVI_UNF_CIPHER_RsaSign

【描述】

使用 RSA 私钥签名一段文本。

【语法】

```
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);
```

【参数】

参数名称	描述	输入/输出
pstRsaSign	签名属性结构体	输入
pu8InData	待签名的数据，如果 pu8HashData 不为空，则使用 pu8HashData 进行签名，该参数将被忽略。	输入
u32InDataLen	待签名的数据长度，单位:byte	输入
pu8HashData	待签名文本的 HASH 摘要，如果为空，则自动计算 pu8InData 的 HASH 摘要进行签名	输入
pu8OutSign	签名结果数据	输出
pu32OutSignLen	签名结果数据长度，单位:byte	输出

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.25 CVI_UNF_CIPHER_RsaVerify

【描述】

使用 RSA 公钥签名验证一段文本。

【语法】

```
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);
```

【参数】

参数名称	描述	输入/输出
pstRsaVerify	签名验证属性结构体	输入
pu8InData	待验证的数据，如果 pu8HashData 不为空，则使用 pu8HashData 进行验证，该参数将被忽略	输入
u32InDataLen	待验证的数据长度，单位:byte	输入
pu8HashData	待验证文本的的 HASH 摘要，如果为空，则自动计算 pu8InData 的 HASH 摘要进行验证	输入
pu8InSign	待验证的签名数据	输入
u32InSignLen	待验证的签名数据长度，单位:byte	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

3.26 CVI_UNF_CIPHER_KladEncryptKey

【描述】

使用 KLAD 对透明密钥进行加密。

【语法】

```
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);
```

【参数】

参数名称	描述	输入/输出
enRootKey	KLAD 根密钥选择，只能选择 EFUSE Key	输入
enTarget	使用该密钥的模块	输入
pu8CleanKey	透明密钥	输入
pu8EcnryptKey	加密密钥	输出
u32KeyLen	密钥的长度，必须是 16 整数倍	输入

【返回值】

返回值	描述
0	成功
非 0	参考错误码 。

4 数据类型

相关数据类型、数据结构定义如下：

- CVI_HANDLE：定义 CIPHER 的句柄类型。
- CVI_UNF_CIPHER_WORK_MODE_E：定义 CIPHER 工作模式。
- CVI_UNF_CIPHER_ALG_E：定义 CIPHER 加密算法。
- CVI_UNF_CIPHER_KEY_LENGTH_E：定义 CIPHER 密钥长度。
- CVI_UNF_CIPHER_BIT_WIDTH_E：定义 CIPHER 加密位宽。
- CVI_UNF_CIPHER_CTRL_CHANGE_FLAG_S：定义 CIPHER CCM 模式的信息结构体。
- CVI_UNF_CIPHER_CA_TYPE_E：定义 CIPHER key 的来源。
- CVI_UNF_CIPHER_KLAD_TARGET_E：定义 Klad 产生的 Key 送达的目标选择。
- CVI_UNF_CIPHER_TYPE_E：定义 CIPHER 加解密类型选择。
- CVI_UNF_CIPHER_ATTS_S：定义 CIPHER 加解密类型结构。
- CVI_UNF_CIPHER_CTRL_S：定义 CIPHER 控制信息结构体。
- CVI_UNF_CIPHER_CTRL_AES_S：AES 加密控制信息结构扩展。
- CVI_UNF_CIPHER_CTRL_AES_CCM_GCM_S：AES-CCM、AES-GCM 加密控制信息结构。
- CVI_UNF_CIPHER_CTRL_DES_S：DES 加密控制信息结构扩展。
- CVI_UNF_CIPHER_CTRL_3DES_S：3DES 加密控制信息结构。
- CVI_UNF_CIPHER_CTRL_EX_S：加密控制信息扩展结构作为算法的专用参数。
- CVI_UNF_CIPHER_DATA_S：定义 CIPHER 加解密数据。
- CVI_UNF_CIPHER_HASH_TYPE_E：定义 CIPHER 哈希算法类型。
- CVI_UNF_CIPHER_HASH_ATTS_S：定义 CIPHER 哈希算法初始化输入结构体。
- CVI_UNF_CIPHER_RSA_ENC_SCHEME_E：定义 RSA 算法数据加密填充方式。
- CVI_UNF_CIPHER_RSA_SIGN_SCHEME_E：定义 RSA 数据签名策略。
- CVI_UNF_CIPHER_RSA_PUB_KEY_S：定义 RSA 公钥结构体。
- CVI_UNF_CIPHER_RSA_PRI_KEY_S：定义 RSA 私钥结构体。

- CVI_UNF_CIPHER_RSA_PUB_ENC_S：定义 RSA 公钥加解密算法参数结体。
- CVI_UNF_CIPHER_RSA_PRI_ENC_S：定义 RSA 私钥解密算法参数结构体。
- CVI_UNF_CIPHER_RSA_SIGN_S：定义 RSA 签名算法参数输入结构体。
- CVI_UNF_CIPHER_RSA_VERIFY_S：定义 RSA 签名验证算法参数输入结构体。
- CIPHER_IV_CHANGE_ONE_PKG：CIPHER 为数据包设置向量时，仅更新一个数据包的 IV。
- CIPHER_IV_CHANGE_ALL_PKG：CIPHER 为数据包设置向量时，更新所有数据包的 IV。

5 错误码

错误代码	宏定义	描述
0x804D0001	CVI_ERR_CIPHER_NOT_INIT	设备未初始化
0x804D0002	CVI_ERR_CIPHER_INVALID_HANDLE	Handle 号无效
0x804D0003	CVI_ERR_CIPHER_INVALID_POINT	参数中有空指针
0x804D0004	CVI_ERR_CIPHER_INVALID_PARA	无效参数
0x804D0005	CVI_ERR_CIPHER_FAILED_INIT	初始化失败
0x804D0006	CVI_ERR_CIPHER_FAILED_GETHANDLE	获取 handle 失败
0x804D0007	CVI_ERR_CIPHER_FAILED_RELEASEHANDLE	释放 handle 失败
0x804D0008	CVI_ERR_CIPHER_FAILED_CONFIGAES	AES 配置无效
0x804D0009	CVI_ERR_CIPHER_FAILED_CONFIGDES	DES 配置无效
0x804D000A	CVI_ERR_CIPHER_FAILED_ENCRYPT	加密失败
0x804D000B	CVI_ERR_CIPHER_FAILED_DECRYPT	解密失败
0x804D000C	CVI_ERR_CIPHER_BUSY	忙状态
0x804D000D	CVI_ERR_CIPHER_NO_AVAILABLE_RNG	没有可用的随机数
0x804D000E	CVI_ERR_CIPHER_FAILED_MEM	内存申请错误
0x804D000F	CVI_ERR_CIPHER_UNAVAILABLE	不可用
0x804D0010	CVI_ERR_CIPHER_OVERFLOW	数据溢出
0x804D0011	CVI_ERR_CIPHER_HARD_STATUS	硬件状态错误
0x804D0012	CVI_ERR_CIPHER_TIMEOUT	等待超时
0x804D0013	CVI_ERR_CIPHER_UNSUPPORTED	不支持的配置
0x804D0014	CVI_ERR_CIPHER_REGISTER_IRQ	中断号无效
0x804D0015	CVI_ERR_CIPHER_ILLEGAL_UUID	非法 UUID
0x804D0016	CVI_ERR_CIPHER_ILLEGAL_KEY	非法 key
0x804D0017	CVI_ERR_CIPHER_INVALID_ADDR	无效地址
0x804D0018	CVI_ERR_CIPHER_INVALID_LENGTH	无效长度
0x804D0019	CVI_ERR_CIPHER_ILLEGAL_DATA	无效数据
0x804D001A	CVI_ERR_CIPHER_RSA_SIGN	RSA 签名失败
0x804D001B	CVI_ERR_CIPHER_RSA_VERIFY	RSA 校验失败
0x804D001E	CVI_ERR_CIPHER_RSA_CRYPT_FAILED	RSA 加解密失败
-1	CVI_FAILURE	操作失败
0x004D0001	CVI_LOG_ERR_MEM	内存操作失败
0x004D0002	CVI_LOG_ERR_SEM	Semaphore 操作失败
0x004D0003	CVI_LOG_ERR_FILE	文件操作失败
0x004D0004	CVI_LOG_ERR_LOCK	锁操作失败
0x004D0005	CVI_LOG_ERR_PARAM	参数无效
0x004D0006	CVI_LOG_ERR_TIMER	计时器错误

下页继续

表 5.1 – 续上页

错误代码	宏定义	描述
0x004D0007	CVI_LOG_ERR_THREAD	线程失败
0x004D0008	CVI_LOG_ERR_TIMEOUT	超时
0x004D0009	CVI_LOG_ERR_DEVICE	Device 操作失败
0x004D0010	CVI_LOG_ERR_STATUS	状态出错
0x004D0011	CVI_LOG_ERR_IOCTRL	IO 操作失败
0x004D0012	CVI_LOG_ERR_INUSE	资源使用中
0x004D0013	CVI_LOG_ERR_EXIST	退出失败
0x004D0014	CVI_LOG_ERR_NOEXIST	资源未退出
0x004D0015	CVI_LOG_ERR_UNSUPPORTED	不支持
0x004D0016	CVI_LOG_ERR_UNAVAILABLE	不可用
0x004D0017	CVI_LOG_ERR_UNINITED	未初始化
0x004D0018	CVI_LOG_ERR_DATABASE	数据库出错
0x004D0019	CVI_LOG_ERR_OVERFLOW	溢出
0x004D0020	CVI_LOG_ERR_EXTERNAL	外部出错
0x004D0021	CVI_LOG_ERR_UNKNOWNED	位置错误
0x004D0022	CVI_LOG_ERR_FLASH	Flash 操作失败
0x004D0023	CVI_LOG_ERR_ILLEGAL_IMAGE	非法镜像
0x004D0024	CVI_LOG_ERR_ILLEGAL_UUID	非法 UUID
0x004D0025	CVI_LOG_ERR_NOPERMISSION	操作不允许