CIPHER

API Reference

Issue 02

Date 2015-07-28

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

About This Document

Purpose

As a module of the HiSilicon digital media processing platform (MPP), the CIPHER module encrypts or decrypts audio and video streams complying with the data encryption standard (DES), triple data encryption standard (3DES), or advanced encryption standard (AES) algorithm. This document describes the application programming interfaces (APIs), data types, and error codes related to the CIPHER module.

NOTE

- This document uses the Hi3516A as an example. Unless otherwise specified, the contents of the Hi3516A are the same as those of the Hi3516D, Hi3521A, and Hi3531A.
- Unless otherwise specified, the contents of Hi3521A are the same as those of Hi3520D V300.

Related Version

The following table lists the product version related to this document.

Product Name	Version
Hi3516A	V100
Hi3516D	V100
Hi3521A	V100
Hi3520D	V300
Hi3531A	V100

Intended Audience

This document is intended for:

- Technical support personnel
- Software development engineers

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Issue 02 (2015-07-28)

This issue is the second official release.

The contents related to the Hi3521A, Hi3520D V300 and Hi3531A are added.

Issue 01 (2014-12-20)

This issue is the first official release.

The contents related to the Hi3516D are added.

Issue 00B01 (2014-09-14)

This issue is the first draft release.

Contents

Tables

1 Overview

1.1 Introduction

As a module of the HiSilicon MPP, the CIPHER module encrypts or decrypts audio and video streams complying with the DES, 3DES, or AES algorithm.

1.2 Encrypting or Decrypting Data

To encrypt or decrypt data, perform the following steps:

Step 1 Start a CIPHER device by calling HI_UNF_CIPHER_Open.

Step 2 Create a CIPHER channel and obtain the CIPHER handle by calling HI_UNF_CIPHER_CreateHandle.

Step 3 Configure CIPHER control information, including the key, initialization vector, encryption algorithm, and working mode by calling HI_UNF_CIPHER_ConfigHandle.

Step 4 Encrypt data by calling HI_UNF_CIPHER_Encrypt or decrypt data by calling HI_UNF_CIPHER_Decrypt.

Step 5 Destroy the CIPHER handle by calling HI_UNF_CIPHER_DestroyHandle.

Step 6 Stop the CIPHER device by calling HI_UNF_CIPHER_Close.

----End

1.3 Notes

Note the following points:

- The data length must be less than 1 MB during each encryption or decryption. If the data length is greater than or equal to 1 MB, the data must be split and then encrypted or decrypted.
- The CIPHER module encrypts blocks by using the HI_UNF_CIPHER_BIT_WIDTH_E value configured by users.

- After a CIPHER channel is created and its attributes are configured (assume that the initialization vectors (IVs) are required for using the configured working mode), the IVs are used in sequence when the encryption or decryption APIs are called.

  For example, you need to encrypt data 0 and data 1 in sequence and the IVs are a, b, c, and d. After data 0 is encrypted, the last block of data 0 uses b. When data 1 is encrypted, the first block of data 1 is encrypted by using c. If the APIs are called continuously, the IVs are used in the sequence of d, a, b, c, d, ….

  Ensure that the IVs are used in the same sequence during encryption and decryption. If the CIPHER control information is reconfigured, the first IV is used.

- You are advised to reconfigure the CIPHER control information before encryption and decryption. In this way, encryption and decryption are performed from the start position of IVs.

2 API Reference

The CIPHER module provides the following APIs:

- HI_UNF_CIPHER_Open: starts a CIPHER device.
- HI_UNF_CIPHER_Close: stops a CIPHER device.
- HI_UNF_CIPHER_CreateHandle: creates a CIPHER channel and obtains the CIPHER handle.
- HI_UNF_CIPHER_DestroyHandle: destroys a CIPHER channel.
- HI_UNF_CIPHER_ConfigHandle: configures the cipher control information.
- HI_UNF_CIPHER_Encrypt: encrypts data.
- HI_UNF_CIPHER_Decrypt: decrypts data.

HI_UNF_CIPHER_Open

[Description]

Starts a CIPHER device.

[Syntax]

```
HI_S32 HI_UNF_CIPHER_Open(HI_VOID);
```

[Parameter]

No parameter is provided.

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see Chapter 4 "Error Codes."

[Requirement]

- Header files: hi_error_ecs.h, hi_type.h, hi_unf_ecs.h, priv_cipher.h
- Library file: libhi_cipher.a

[Note]

No note is provided.

[Example]

No example is provided.

HI_UNF_CIPHER_Close

[Description]

Stops a CIPHER device.

[Syntax]

```
HI_S32 HI_UNF_CIPHER_Close(HI_VOID);
```

[Parameter]

No parameter is provided.

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see Chapter 4 "Error Codes."

[Requirement]

- Header files: hi_error_ecs.h, hi_type.h, hi_unf_ecs.h, priv_cipher.h
- Library file: libhi_cipher.a

[Note]

No note is provided.

[Example]

No example is provided.

HI_UNF_CIPHER_CreateHandle

[Description]

Creates a CIPHER channel and obtains the CIPHER handle.

[Syntax]

```
HI_S32 HI_UNF_CIPHER_CreateHandle(HI_HANDLE* phCipher);
```

[Parameter]

Parameter	Description	Input/Output
phCipher	Pointer to the CIPHER handle.	Output

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see Chapter 4 "Error Codes."

[Requirement]

- Header files: hi_error_ecs.h, hi_type.h, hi_unf_ecs.h, priv_cipher.h
- Library file: libhi_cipher.a

[Note]

- The **phCipher** parameter cannot be null.
- The handle is the input when data is encrypted or decrypted.
- A maximum of seven CIPHER channels are supported.
- Single-block encryption or decryption is not supported.

[Example]

No example is provided.

HI_UNF_CIPHER_DestroyHandle

[Description]

Destroys a CIPHER channel.

[Syntax]

```
HI_S32 HI_UNF_CIPHER_DestroyHandle(HI_HANDLE hCipher);
```

[Parameter]

Parameter	Description	Input/Output
hCipher	CIPHER handle.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see Chapter 4 "Error Codes."

[Requirement]

- Header files: hi_error_ecs.h, hi_type.h, hi_unf_ecs.h, priv_cipher.h
- Library file: libhi_cipher.a

[Note]

A CIPHER handle can be destroyed repeatedly.

[Example]

No example is provided.

HI_UNF_CIPHER_ConfigHandle

[Description]

Configures the cipher control information. For details, see the data type HI_UNF_CIPHER_CTRL_S.

[Syntax]

```
HI_S32 HI_UNF_CIPHER_ConfigHandle(HI_HANDLE hCipher, HI_UNF_CIPHER_CTRL_S*
pstCtrl);
```

[Parameter]

Parameter	Description	Input/Output
hCipher	CIPHER handle.	Input
pstCtrl	Pointer to the control information.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see Chapter 4 "Error Codes."

[Requirement]

- Header files: hi_error_ecs.h, hi_type.h, hi_unf_ecs.h, priv_cipher.h
- Library file: libhi_cipher.a

[Note]

The pointer to the control information cannot be null.

[Example]

No example is provided.

HI_UNF_CIPHER_Encrypt

[Description]

Encrypts data.

[Syntax]

```
HI_S32 HI_UNF_CIPHER_Encrypt(HI_HANDLE hCipher, HI_U32 u32SrcPhyAddr, HI_U32
u32DestPhyAddr, HI_U32 u32ByteLength);
```

[Parameter]

Parameter	Description	Input/Output
hCipher	CIPHER handle.	Input
u32SrcPhyAddr	Physical address of the source data (data to be encrypted).	Input
u32DestPhyAddr	Physical address of the encrypted data.	Output
u32ByteLength	Data length, in bytes.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see Chapter 4 "Error Codes."

[Requirement]

- Header files: hi_error_ecs.h, hi_type.h, hi_unf_ecs.h, priv_cipher.h
- Library file: libhi_cipher.a

[Note]

- A CIPHER handle must be created before you call this API.
- This API can be called repeatedly.
- The data length must be greater than or equal to 16 bytes, but less than 1024x1024 bytes.

[Example]

No example is provided.

HI_UNF_CIPHER_Decrypt

[Description]

Decrypts data.

[Syntax]

```
HI_S32 HI_UNF_CIPHER_Decrypt(HI_HANDLE hCipher, HI_U32 u32SrcPhyAddr, HI_U32
u32DestPhyAddr, HI_U32 u32ByteLength);
```

[Parameter]

Parameter	Description	Input/Output
hCipher	CIPHER handle.	Input
u32SrcPhyAddr	Physical address of the source data (data to be decrypted).	Input
u32DestPhyAddr	Physical address of the decrypted data.	Output
u32ByteLength	Data length, in bytes.	Input

[Return Value]

Return Value	Description
0	Success.
Other values	For details, see Chapter 4 "Error Codes."

[Requirement]

- Header files: hi_error_ecs.h, hi_type.h, hi_unf_ecs.h, priv_cipher.h
- Library file: libhi_cipher.a

[Note]

- A CIPHER handle must be created before you call this API.
- This API can be called repeatedly.
- The data length must be greater than or equal to 16 bytes, but less than 1024x1024 bytes.

[Example]

No example is provided.

3 Data Types

The CIPHER data types are as follows:

- HI_HANDLE: defines the handle type of the CIPHER module.
- HI_UNF_CIPHER_WORK_MODE_E: defines the working mode of the CIPHER module.
- HI_UNF_CIPHER_ALG_E: defines the encryption and decryption algorithms of the CIPHER module.
- HI_UNF_CIPHER_KEY_LENGTH_E: defines the key length of the CIPHER module.
- HI_UNF_CIPHER_BIT_WIDTH_E: defines the encryption bit width of the CIPHER module.
- HI_UNF_CIPHER_CTRL_S: defines the structure of the control information.

HI_HANDLE

[Description]

Defines the handle type of the CIPHER module.

[Syntax]

```
typedef HI_U32 HI_HANDLE;
```

[Member]

No parameter member is provided.

[Note]

No note is provided.

[See Also]

No related data type is provided.

HI_UNF_CIPHER_WORK_MODE_E

[Description]

Defines the working mode of the CIPHER module.

[Syntax]

```
typedef enum hiHI_UNF_CIPHER_WORK_MODE_E
{
    HI_UNF_CIPHER_WORK_MODE_ECB = 0x0,
    HI_UNF_CIPHER_WORK_MODE_CBC = 0x1,
    HI_UNF_CIPHER_WORK_MODE_CFB = 0x2,
    HI_UNF_CIPHER_WORK_MODE_OFB = 0x3,
    HI_UNF_CIPHER_WORK_MODE_CTR = 0x4,
    HI_UNF_CIPHER_WORK_MODE_BUTT = 0x5
};
```

[Member]

Member	Description
HI_UNF_CIPHER_WORK_MODE_ECB	Electronic codebook (ECB) mode
HI_UNF_CIPHER_WORK_MODE_CBC	Cipher block chaining (CBC) mode
HI_UNF_CIPHER_WORK_MODE_CFB	Cipher feedback (CFB) mode
HI_UNF_CIPHER_WORK_MODE_OFB	Output feedback (OFB) mode
HI_UNF_CIPHER_WORK_MODE_CTR	Counter (CTR) mode

[Note]

No note is provided.

[See Also]

No related data type is provided.

HI_UNF_CIPHER_ALG_E

[Description]

Defines the encryption and decryption algorithms of the CIPHER module.

[Syntax]

```
typedef enum hiHI_UNF_CIPHER_ALG_E
{
    HI_UNF_CIPHER_ALG_DES  = 0x0,
    HI_UNF_CIPHER_ALG_3DES = 0x1,
    HI_UNF_CIPHER_ALG_AES  = 0x2,
    HI_UNF_CIPHER_ALG_BUTT = 0x3
}HI_UNF_CIPHER_ALG_E;
```

[Member]

Member	Description
HI_UNF_CIPHER_ALG_DES	DES algorithm

Member	Description
HI_UNF_CIPHER_ALG_3DES	3DES algorithm
HI_UNF_CIPHER_ALG_AES	AES algorithm

[Note]

No note is provided.

[See Also]

No related data type is provided.

HI_UNF_CIPHER_KEY_LENGTH_E

[Description]

Defines the key length of the CIPHER module.

[Syntax]

```
typedef enum hiHI_UNF_CIPHER_KEY_LENGTH_E
{
    HI_UNF_CIPHER_KEY_AES_128BIT = 0x0,
    HI_UNF_CIPHER_KEY_AES_192BIT = 0x1,
    HI_UNF_CIPHER_KEY_AES_256BIT = 0x2,
    HI_UNF_CIPHER_KEY_DES_3KEY = 0x2,
    HI_UNF_CIPHER_KEY_DES_2KEY = 0x3,
}HI_UNF_CIPHER_KEY_LENGTH_E;
```

[Member]

Member	Description
HI_UNF_CIPHER_KEY_AES_128BIT	128-bit key for the AES algorithm
HI_UNF_CIPHER_KEY_AES_192BIT	192-bit key for the AES algorithm
HI_UNF_CIPHER_KEY_AES_256BIT	256-bit key for the AES algorithm
HI_UNF_CIPHER_KEY_DES_3KEY	Three keys for the DES algorithm
HI_UNF_CIPHER_KEY_DES_2KEY	Two keys for the DES algorithm

[Note]

- The key length for the AES algorithm is 128 bits, 192 bits, or 256 bits.
- The number of keys for the 3DES algorithm is 2 or 3. Each key is 64 bits and is used for DES encryption.
- This data type is invalid for the DES algorithm.

[See Also]

No related data type is provided.

HI_UNF_CIPHER_BIT_WIDTH_E

[Description]

Defines the encryption bit width of the CIPHER module.

[Syntax]

```
typedef enum hiHI_UNF_CIPHER_BIT_WIDTH_E
{
    HI_UNF_CIPHER_BIT_WIDTH_64BIT  = 0x0,
    HI_UNF_CIPHER_BIT_WIDTH_8BIT   = 0x1,
    HI_UNF_CIPHER_BIT_WIDTH_1BIT   = 0x2,
    HI_UNF_CIPHER_BIT_WIDTH_128BIT = 0x3,
}HI_UNF_CIPHER_BIT_WIDTH_E;
```

[Member]

Member	Description
HI_UNF_CIPHER_BIT_WIDTH_64BIT	Bit width of 64 bits
HI_UNF_CIPHER_BIT_WIDTH_8BIT	Bit width of 8 bits
HI_UNF_CIPHER_BIT_WIDTH_1BIT	Bit width of 1 bit
HI_UNF_CIPHER_BIT_WIDTH_128BIT	Bit width of 128 bits

[Note]

No note is provided.

[See Also]

No related data type is provided.

HI_UNF_CIPHER_CTRL_S

[Description]

Defines the structure of the control information.

[Syntax]

```
typedef struct hiHI_UNF_CIPHER_CTRL_S
{
    HI_U32                u32Key[8];
    HI_U32                u32IV[4];
    HI_BOOL               bKeyByCA;
    HI_UNF_CIPHER_ALG_E       enAlg;
    HI_UNF_CIPHER_BIT_WIDTH_E  enBitWidth;
    HI_UNF_CIPHER_WORK_MODE_E  enWorkMode;
```

```
    HI_UNF_CIPHER_KEY_LENGTH_E enKeyLen;
} HI_UNF_CIPHER_CTRL_S;
```

[Member]

Member	Description
u32Key[8]	Key
u32IV[4]	Initialization vector
bKeyByCA	Whether to use advanced conditional access (CA) encryption or decryption keys
enAlg	Encryption algorithm
enBitWidth	Bit width for encryption or decryption
enWorkMode	Working mode
enKeyLen	Key length

[Note]

The initialization vector is not required in ECB mode or CTR mode.

[See Also]

No related data type is provided.

4 Error Codes

Table 4-1 describes the error codes for the CIPHER module.

Table 4-1 Error codes for the CIPHER module

Error Code	Macro Definition	Description
0x804D0001	HI_ERR_CIPHER_NOT_INIT	The CIPHER device is not initialized.
0x804D0002	HI_ERR_CIPHER_INVALID_HANDLE	The handle ID is invalid.
0x804D0003	HI_ERR_CIPHER_INVALID_POINT	The pointer is null.
0x804D0004	HI_ERR_CIPHER_INVALID_PARA	The parameter is invalid.
0x804D0005	HI_ERR_CIPHER_FAILED_INIT	The CIPHER module fails to be initialized.
0x804D0006	HI_ERR_CIPHER_FAILED_GETHANDLE	The handle fails to be obtained.
–1	HI_FAILURE	The operation fails.