Cryptography API

Moway 1

Version 1.0.6.0

Copyright & Trademarks

Disclaimer

All attempts have been made to make the information in this document complete and accurate. But we cannot guarantee everything is perfect, we will correct it in next version released in case some error has been found. Senseshield is not responsible for any direct or indirect damages or loss of business resulted from inaccuracies or omissions.

The specifications contained in this document are subject to change without notice.

Documentation Improvement

Any suggestion to this manual from you are welcome, we are glad to hear any feedback from you which will help us to continuously improve the documents quality and support and serve the developer to protect software products more efficiently.

Contact

Company: Beijing Senseshield Technology Co., Ltd

Address: Suite 510, Block C, Internet Innovation Center, Building 5, No.10, Xibeiwang East Road, Haidian District, Beijing China

Tel: +86-10-56730936

Fax: +86-10-56730936-8007

Sales: info@senselock.com;

Website: https://lm-global.virbox.com/

Virbox Developer Center Website: https://developer.lm-global.virbox.com/

About this Manual

Objective

This manual is designed to help software developer to learn about the principle, protection scheme of Moway 1 and how to use Moway 1 and associated tools to design the software protection scheme and protect their software.

Notes

Caution: The content with this mark in the manual indicates that you need to pay highly attention, otherwise may cause serious consequences.

Mark: The content with this mark in the manual indicates that you need to pay attention to read.

Table of Contents

1 Constant Definition

1.1 Key type ID

Item	Macro	Value	Description
1	SLC_KEY_SYMMETRIC	0x01	Key for symmetric encryption algorithm, support: AES, TDES and DES.
2	SLC_KEY_PUBLIC	0x02	Public key of public key algorithm, include: the public key of RSA, ECC algorithm.
3	SLC_KEY_PRIVATE	0x03	Private key of public key algorithm, include: the private key of RSA, ECC algorithm.
4	SLC_KEY_HMAC	0x04	Key for HMAC calculation, includes: MD5, SHA1, SHA256

Note: The type of ID to the SLC-KEY will be used to the function: SlcCheckKeyType

1.2 Cryptography Algorithm ID

Item	Macro	Value	Description
1	SLC_CIPHER_ALGO_AES	0x00	Key for AES algorithm
2	SLC_CIPHER_ALGO_DES	0x01	Key for DES algorithm
3	SLC_CIPHER_ALGO_TDES	0x02	Key for TDES algorithm
4	SLC_CIPHER_ALGO_ECC	0x10	Key for ECC algorithm
5	SLC_CIPHER_ALGO_RSA	0x11	Key for RSA algorithm
6	SLC_CIPHER_ALGO_HMAC_MD5	0x41	Key for HMAC-MD5 Calculation
7	SLC_CIPHER_ALGO_HMAC_SHA1	0x42	Key for HMAC-SHA1 Calculation
8	SLC_CIPHER_ALGO_HMAC_SHA256	0x43	Key for HMAC-SHA256 calculation

Note: The algorithm type of SLC_KEY, which used to function: SlcCheckKeyType

1.3 Cryptography mode of DES/TDES/AES algorithm

Item	Macro	Value	Description
1	SLC_MODE_ECB	0x00	Symmetric algorithm ECB mode

Item	Macro	Value	Description
2	SLC_MODE_CBC	0x01	Symmetric algorithm CBC mode

1.4 Hash algorithm ID

Item	Macro	Value	Description
1	SLC_HASH_ALGO_SHA1	0x01	SHA1 Hash digest algorithm
2	SLC_HASH_ALGO_SHA256	0x02	SHA256 Hash digest algorithm
3	SLC_HASH_ALGO_MD5	0x03	MD5 Hash digest algorithm

1.5 RSA algorithm padding mode

Item	Macro	Value	Description
1	SLC_PAD_MODE_NONE	0x00	RSA algorithm, no padding.
2	SLC_PAD_MODE_PKCS_1_V1_5	0x01	RSA algorithm, PKCS1 V1.5 padding mode

1.6 Key length definition

Item	Macro	Value	Description
1	SLC_DES_KEY_BIT_LEN	56	The key length of DES algorithm
2	SLC_TDES_KEY_BIT_LEN	112	The key length of TDES algorithm
3	SLC_AES_KEY_BIT_128	128	The key length to AES algorithm is 128 bit
4	SLC_AES_KEY_BIT_192	192	The key length to AES algorithm is 192 bit
5	SLC_AES_KEY_BIT_256	256	The key length to AES algorithm is 256 bit
6	SLC_RSA_KEY_BIT_1024	1024	The key length to RSA algorithm is 1024 bits
7	SLC_RSA_KEY_BIT_2048	2048	The key length to RSA algorithm is 2048 bits
8	SLC_RSA_KEY_BIT_4096	4096	The key length to RSA algorithm is 4096 bits
9	SLC_ECC_KEY_BIT_192	192	The Key length to ECC algorithm is 192 bits
10	SLC_ECC_KEY_BIT_256	256	The Key length to ECC algorithm is 256 bits

1.7 Data block size definition to DES/TDES/AES algorithm

Item	Macro	Value	Description
1	SLC_DES_BLOCK_SIZE	8	Symmetric algorithm, DES data block size (length)
2	SLC_TDES_BLOCK_SIZE	8	Symmetric algorithm, TDES data block size (length)
3	SLC_AES_BLOCK_SIZE	16	Symmetric algorithm, AES data block size (length)

1.8 HASH algorithm digest length definition (in bytes length)

Item	Macro	Value	Description
1	SLC_MD5_DIGEST_LEN	16	MD5 digest length
2	SLC_SHA1_DIGEST_LEN	20	SHA1 digest length
3	SLC_SHA256_DIGEST_LEN	32	SHA256 digest length

1.9 The length of signed in Signature algorithm (in bytes length)

Item	Macro	Value	Description
1	SLC_RSA1024_SIG_LEN	128	The signature byte length: RSA1024 algorithm
2	SLC_RSA2048_SIG_LEN	256	The signature byte length: RSA2048 algorithm
3	SLC_RSA4096_SIG_LEN	512	The signature byte length: RSA4096 algorithm
4	SLC_ECC192_SIG_MAX_LEN	128	The signature byte length: ECC 192 algorithm
5	SLC_ECC256_SIG_MAX_LEN		The signature byte length: ECC 256 algorithm

2. Return value

Item	Macro Returned	Value	Description
1	SLC_SUCCESS	0x0000	Success execution
2	SLC_ERROR_NO_MEMORY	0x0001	Failed to allocate memory
3	SLC_ERROR_INVALID_PARAMETER	0x0002	Invalid parameter, usually refers NULL pointer used)
4	SLC_ERROR_INVALID_HASH_ALGO	0x0003	Invalid HASH algorithm ID
5	SLC_ERROR_INVALID_CIPHER_ALGO	0x0004	Invalid cryptography algorithm
6	SLC_ERROR_INVALID_MODE	0x0005	Invalid Cryptography mode or padding mode
7	SLC_ERROR_INPUTDATA_LENGTH	0x0006	The input data length error
8	SLC_ERROR_INSUFFICIENT_BUFFER	0x0007	Insufficient buffer size to output data
9	SLC_ERROR_INVALID_KEY_SIZE	0x0008	Key length error
10	SLC_ERROR_MAKE_KEY	0x0009	Failed to generate the key
11	SLC_ERROR_SIGN	0x000A	Failed to sign
12	SLC_ERROR_VERIFY	0x000B	Failed to verify signature (data doesn't consistent with signature)
13	SLC_ERROR_INVALID_INPUT_DATA	0x000C	Invalid input data
14	SLC_ERROR_BAD_KEY_FORMAT	0x000D	Key format error
15	SLC_ERROR_BAD_KEY_VERSION	0x000E	Key version incorrect.
16	SLC_ERROR_BAD_KEY_ALGORITHM	0x000F	Key algorithm error
17	SLC_ERROR_BAD_KEY_TYPE	0x0010	Key type error
18	SLC_ERROR_BAD_PADDING	0x0011	Padding mode error
19	SLC_ERROR_UNKNOW	0x0012	Unexpected error (API internal error, may not happened)

3. Type definition & Description

```
#ifndef SLC_BYTE
typedef unsigned char      SLC_BYTE;
#endif

#ifndef SLC_ULONG
typedef unsigned long      SLC_ULONG;
#endif

typedef void *             SLC_HASH_CTX;
typedef void *             SLC_KEY;
```

4. Function description

4.1 SlcAesEncRaw

AES encryption function, which use the key with non format in length of 16 bytes, 24 bytes, 32 bytes

```
SLC_ULONG SLCAPI SlcAesEncRaw(
                IN      SLC_BYTE            byMode,
                IN      const SLC_BYTE      * pbKey,
                IN      SLC_ULONG           ulKeyBitLen,
                IN      const SLC_BYTE      * pbIv,
                IN      const SLC_BYTE      * pbData,
                IN      SLC_ULONG           ulDataLen,
                OUT     SLC_BYTE            * pbOutBuf,
                IN      SLC_ULONG           ulOutBufLen,
                OUT     SLC_ULONG           * pulOutDataLen
                );
```

Parameter description

byMode	Cryptography mode Value: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
pbKey	AES key buffer size, can be 16, 24, 32 bytes, but not leave with NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulKeyBitLen	The bit length of key, in value at 128,192, 256 which correspondence with the key in 16, 24, 32 bytes respectively Error then return with: SLC_ERROR_INVALID_KEY_SIZE
pbIv	Initialization vector When: the `byMode` option value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not

	set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` option value at: SLC_MODE_CBC, this parameter need to specify the buffer with 16 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data (Plaintext) to be encrypted, can NOT be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The data bytes to be encrypted, the length must be the multiple of 16, and also cannot be 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer for encrypted data, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of output buffer, which length is greater than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the number of bytes of output data, When the output buffer is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the number of bytes of output data does not need to be returned.

Return value description:

Success then return: SLC_SUCCESS。

For other value returned, please refer the ”Parameter description” & “Return value” accordingly

4.2 SlcAesDecRaw

AES decryption function, which use the key with “None format” in length of 16 bytes, 24 bytes, 32 bytes

```
SLC_ULONG SLCAPI SlcAesDecRaw(
                    IN      SLC_BYTE            byMode,
                    IN      const SLC_BYTE      * pbKey,
                    IN      SLC_ULONG           ulKeyBitLen,
                    IN      const SLC_BYTE      * pbIv,
                    IN      const SLC_BYTE      * pbData,
                    IN      SLC_ULONG           ulDataLen,
                    OUT     SLC_BYTE            * pbOutBuf
```

```
    IN      SLC_ULONG           ulOutBufLen,
    OUT     SLC_ULONG           * pulOutDataLen
    );
```

Parameter description:

byMode	Encryption mode Value at： SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
pbKey	Buffer to AES key, can be 16, 24, 32 bytes, cannot be NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulKeyBitLen	The bit length of key which value at: 128, 192, 256 which correspondence with the key in 16, 24, 32 bytes respectively. Error then return with: SLC_ERROR_INVALID_KEY_SIZE
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` option value at: SLC_MODE_CBC, this parameter need to specify the buffer with 16 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data to be decrypted, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The bytes number of data to be decrypted, which length must be the multiple of 16, and also cannot be 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer of decrypted data, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulOutBufLen	The length (in bytes) of output buffer, which length is greater than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the number of bytes of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer.

	NULL can be passed in, which means that the number of bytes of output data does not need to be returned.

Return value description:

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.3 SlcAesGenerateKey

To generate the AES key with the “SLC_KEY” format

If you have “none format” key used previously, then you can use the “none format” key previously to initialize the key generated with “SLC_KEY” format, which makes the “SLC_KEY” key contain the original “none format” key accordingly.

```
SLC_ULONG SLCAPI SlcAesGenerateKey(
        IN      const SLC_BYTE          * pbInitData,
        IN      SLC_ULONG                 ulKeyBitLen,
        OUT     SLC_KEY                   * pSlcKey
        );
```

Parameter description

pbInitData	The previously used none format key, which can be 16, 24, 32 bytes, can be: NULL also, If the value is: NULL, then this API will generate a random number as the key
ulKeyBitLen	The bits length of key, which value at: 128, 192, 256. Value at 128 (SLC_AES_KEY_BIT_128), correspondence with the key in 16 bytes. Value at 192 (SLC_AES_KEY_BIT_192), correspondence with the key in 24 bytes. Value at 256 (SLC_AES_KEY_BIT_256), correspondence with the key in 32 bytes. Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pSlcKey	Pass in a “SLC_KEY” variable address, cannot be NULL This fucntion for allocating the buffer for key stored, it is require to call” SlcFreeKey” to release the key when used

	Error then return with value: SLC_ERROR_INVALID_PARAMETER.

Return value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.4 SlcAesEnc

AES Encryption function, use the AES key with “SLC_KEY” format

```
SLC_ULONG SLCAPI SlcAesEnc(
        IN      SLC_BYTE            byMode,
        IN      SLC_KEY                 SlcKey,
        IN      const SLC_BYTE      * pbIv,
        IN      const SLC_BYTE      * pbData,
        IN      SLC_ULONG             ulDataLen,
        OUT     SLC_BYTE              * pbOutBuf
                                IN      SLC_ULONG                   ulOutBufLen,
        OUT     SLC_ULONG             * pulOutDataLen
        );
```

Parameter description

byMode	Encryption mode Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
SlcKey	The AES key with “SLC_KEY” format, cannot be: NULL. When value at: NULL, error return with: SLC_ERROR_INVALID_PARAMETER. When key format error, error return with: SLC_ERROR_BAD_KEY_VERSION, SLC_ERROR_BAD_KEY_TYPE,

	SLC_ERROR_BAD_KEY_ALGORITHM, SLC_ERROR_INVALID_KEY_SIZE.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; when not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 16 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data to be encrypted (Plaintext), cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The byte length of data to be encrypted, the length must be the multiple of 16, and also cannot be 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer for data encrypted, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The number of bytes of output buffer, which length is greater than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the number of bytes of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the number of bytes of output data does not need to be returned.

Return value description:

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.5 SlcAesDec

AES decryption function, use the AES key with “SLC_KEY” format.

```
SLC_ULONG SLCAPI SlcAesDec(
        IN      SLC_BYTE            byMode,
        IN      SLC_KEY             SlcKey,
        IN      const SLC_BYTE      * pbIv,
        IN      const SLC_BYTE      * pbData,
        IN      SLC_ULONG           ulDataLen,
        OUT     SLC_BYTE            * pbOutBuf
        IN      SLC_ULONG           ulOutBufLen,
        OUT     SLC_ULONG           * pulOutDataLen
        );
```

Parameter description

byMode	Encryption Mode Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
SlcKey	The AES key with “SLC_KEY”, cannot set: NULL When value at: NULL, then error return with: SLC_ERROR_INVALID_PARAMETER. When Key format error, then error return with: SLC_ERROR_BAD_KEY_VERSION, SLC_ERROR_BAD_KEY_TYPE, SLC_ERROR_BAD_KEY_ALGORITHM, SLC_ERROR_INVALID_KEY_SIZE.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 16 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER

pbData	The data to be decrypted, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The byte length of data to be decrypted, the length must be the multiple of 16, and also not 0, Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	Output buffer of decrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The byte length of output buffer, and the length must not less than “ulDataLen” Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the number of bytes of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the number of bytes of output data does not need to be returned.

Return value description:

Success then return with: SLC_SUCCESS

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.6 SlcDesEncRaw

DES encryption function, use the key in 8 bytes with “None format”

```
SLC_ULONG SLCAPI SlcDesEncRaw(
                    IN      SLC_BYTE            byMode,
                    IN      const SLC_BYTE      * pbKey,
                    IN      const SLC_BYTE      * pbIv,
                    IN      const SLC_BYTE      * pbData,
                    IN      SLC_ULONG           ulDataLen,
                    OUT     SLC_BYTE            * pbOutBuf
                   IN      SLC_ULONG           ulOutBufLen,
```

```
    OUT     SLC_ULONG               * pulOutDataLen
);
```

Parameter description:

byMode	Encryption mode Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
pbKey	DES key buffer, 8 bytes, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data (Plaintext) to be encrypted, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The byte length of data to be encrypted, which the length must be multiple of 8, and also not 0. Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer to encrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The byte length of output buffer, the length must not less than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the number of bytes of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the number of bytes of output data does not need to be returned.

Return value description:

Success then return with: SLC_SUCCESS

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.7 SlcDesDecRaw

DES decryption function, use the key in 8 bytes with “None format”

```
SLC_ULONG SLCAPI SlcDesDecRaw(
    IN      SLC_BYTE          byMode,
    IN      const SLC_BYTE    * pbKey,
    IN      const SLC_BYTE    * pbIv,
    IN      const SLC_BYTE    * pbData,
    IN      SLC_ULONG         ulDataLen,
    OUT     SLC_BYTE          * pbOutBuf
    IN      SLC_ULONG         ulOutBufLen,
    OUT     SLC_ULONG         * pulOutDataLen
    );
```

Parameter description

byMode	Encryption mode Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
pbKey	DES KEY buffer, in 8 bytes, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL

	Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data to be decrypted, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be decrypted, the length must be multiple of 8, and also non-0, Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer for decrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The Length of output buffer (in bytes), the length must not less than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the number of bytes of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the number of bytes of output data does not need to be returned.

Return value description:

Success then return with: SLC_SUCCESS

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.8 SlcDesGenerateKey

To generate the DES KEY in “SLC_KEY” format.

If you have “none format” key used previously, then you can use the “none format” key previously to initialize the key generated with “SLC_KEY” format, which makes the “SLC_KEY” key contain the original “none format” key accordingly.

```
SLC_ULONG SLCAPI SlcDesGenerateKey(
                    IN      const SLC_BYTE          * pbInitData,
                    IN      SLC_ULONG                 ulKeyBitLen,
```

```
    OUT     SLC_KEY                 * pSlcKey
    );
```

Parameter description

pbInitData	The “None format” KEY previously used, in 8 bytes, can be: NULL If the value is: NULL, then this function will generate a random number as the KEY
ulKeyBitLen	The bits length of KEY, may value at: 56(SLC_DES_KEY_BIT_LEN) Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pSlcKey	Pass in a “SLC_KEY” variable address, cannot be NULL This function used for allocating the buffer for key stored, it is require to call” SlcFreeKey” to release the key when used Error then return with value: SLC_ERROR_INVALID_PARAMETER.

Return value description:

Success then return with: SLC_SUCCESS

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.9 SlcDesEnc

DES encryption function, use the DES key in 8 bytes with “None format”

```
SLC_ULONG SLCAPI SlcDesEnc(
    IN      SLC_BYTE            byMode,
    IN      SLC_KEY             SlcKey,
    IN      const SLC_BYTE      * pbIv,
    IN      const SLC_BYTE      * pbData,
    IN      SLC_ULONG           ulDataLen,
    OUT     SLC_BYTE            * pbOutBuf
    IN      SLC_ULONG           ulOutBufLen,
    OUT     SLC_ULONG           * pulOutDataLen
```

);

Parameter description

byMode	Encryption mode, Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
SlcKey	The DES KEY with “SLC_KEY” format, cannot be: NULL. Value at: NULL, then Error will be return with: SLC_ERROR_INVALID_PARAMETER. When KEY format error, then return with: SLC_ERROR_BAD_KEY_VERSION, SLC_ERROR_BAD_KEY_TYPE, SLC_ERROR_BAD_KEY_ALGORITHM, SLC_ERROR_INVALID_KEY_SIZE.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data (Plaintext) to be encrypted, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The data length (in bytes) to be encrypted, the length must be the multiple of 8, and also not: 0) Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer to encrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The byte length of output buffer, which the length must not been less than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the number of bytes of output data, When the output buffer

	length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the number of bytes of output data does not need to be returned.

Return value description:

Success then return with: SLC_SUCCESS

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.10 SlcDesDec

DES Decryption function, use the DES KEY with SLC_KEY format.

```
SLC_ULONG SLCAPI SlcDesDec(
        IN      SLC_BYTE            byMode,
        IN      SLC_KEY             SlcKey,
        IN      const SLC_BYTE      * pbIv,
        IN      const SLC_BYTE      * pbData,
        IN      SLC_ULONG           ulDataLen,
        OUT     SLC_BYTE            * pbOutBuf
        IN      SLC_ULONG           ulOutBufLen,
        OUT     SLC_ULONG           * pulOutDataLen
        );
```

Parameter description

byMode	Encryption mode, Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
SlcKey	DES KEY with “SLC_KEY” format, cannot be: NULL When Value in: NULL， then Error return with: SLC_ERROR_INVALID_PARAMETER. When KEY format invalid, error, then return with: SLC_ERROR_BAD_KEY_VERSION,

	SLC_ERROR_BAD_KEY_TYPE, SLC_ERROR_BAD_KEY_ALGORITHM, SLC_ERROR_INVALID_KEY_SIZE.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data to be decrypted, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be decrypted, the length must be the multiple of 16, and also non 0. Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer to decrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The byte length of output buffer, which the length must not been less than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return value description:

Success then return with: SLC_SUCCESS

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.11 SlcTDesEncRaw

TDES encryption function, use the KEY with none format in 16 bytes

```
SLC_ULONG SLCAPI SlcTDesEncRaw(
                    IN      SLC_BYTE            byMode,
                    IN      const SLC_BYTE      * pbKey,
```

```
    IN      const SLC_BYTE      * pbIv,
    IN      const SLC_BYTE      * pbData,
    IN      SLC_ULONG             ulDataLen,
    OUT     SLC_BYTE              * pbOutBuf
    IN      SLC_ULONG             ulOutBufLen,
    OUT     SLC_ULONG             * pulOutDataLen
    );
```

Parameter description

byMode	Encryption mode, Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
pbKey	TDES KEY buffer, in 16 bytes, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data (Plaintext) to be encrypted, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be encrypted, the length must be the multiple of 8, and also none 0. Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer for encrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length of output buffer (in bytes), which length must not be less than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。

pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.12 SlcTDesDecRaw

TDES Encryption function, use the KEY with none format, in 16 bytes

```
SLC_ULONG SLCAPI SlcTDesDecRaw(
    IN      SLC_BYTE          byMode,
    IN      const SLC_BYTE    * pbKey,
    IN      const SLC_BYTE    * pbIv,
    IN      const SLC_BYTE    * pbData,
    IN      SLC_ULONG         ulDataLen,
    OUT     SLC_BYTE          * pbOutBuf
    IN      SLC_ULONG         ulOutBufLen,
    OUT     SLC_ULONG         * pulOutDataLen
    );
```

Parameter description

byMode	Encryption mode, Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
pbKey	Buffer for AES key, in 16 bytes, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.

pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data to be decrypted, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be decrypted. The length must be the multiple of 8, and also none 0. Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer to decrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length of output buffer (in bytes), the length must not less than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.13 SlcTDesGenerateKey

To generate the TDES KEY in “SLC_KEY” format

If you have “none format” key used previously, then you can use the “none format” key previously to initialize the key generated with “SLC_KEY” format, which makes the “SLC_KEY” key contain the original “none format” key accordingly.

```
SLC_ULONG SLCAPI SlcTDesGenerateKey(
                    IN      const SLC_BYTE          * pbInitData,
                    IN      SLC_ULONG               ulKeyBitLen,
                    OUT     SLC_KEY                 * pSlcKey
                    );
```

Parameter description

pbInitData	The KEY previously used, without format (None format), in 16 bytes, can be: NULL If the value is: NULL, then this API will generate a random number as the KEY
ulKeyBitLen	The bits length of KEY, which value can be: 112(SLC_TDES_KEY_BIT_LEN) Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pSlcKey	Pass in a “SLC_KEY” variable address, cannot be NULL This API used for allocating the buffer for key stored, it is require to call” SlcFreeKey” to release the key when used Error then return with value: SLC_ERROR_INVALID_PARAMETER.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.14 SlcTDesEnc

TDES encryption function, use the TDES key in “SLC_KEY” format

```
SLC_ULONG SLCAPI SlcTDesEnc(
                    IN      SLC_BYTE                byMode,
                    IN      SLC_KEY                 SlcKey,
                    IN      const SLC_BYTE          * pbIv,
                    IN      const SLC_BYTE          * pbData,
```

```
    IN      SLC_ULONG       ulDataLen,
    OUT     SLC_BYTE        * pbOutBuf
    IN      SLC_ULONG       ulOutBufLen,
    OUT     SLC_ULONG       * pulOutDataLen
    );
```

Parameter description

byMode	Encryption mode Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error then return with: SLC_ERROR_INVALID_MODE。
SlcKey	TDES KEY in “SLC_KEY” format, cannot be: NULL When value at: NULL, then error return with: SLC_ERROR_INVALID_PARAMETER. When Key format invalid/error, then error return with: SLC_ERROR_BAD_KEY_VERSION, SLC_ERROR_BAD_KEY_TYPE, SLC_ERROR_BAD_KEY_ALGORITHM, SLC_ERROR_INVALID_KEY_SIZE.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data (plaintext) to be encrypted, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be encrypted, the length must be the multiple of 8, and also none 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer to encrypted data, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of output buffer, the length must not be less than ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。

pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.15 SlcTDesDec

TDES decryption functions, use the TDES key in “SLC_KEY” format

```
SLC_ULONG SLCAPI SlcTDesDec(
        IN      SLC_BYTE            byMode,
        IN      SLC_KEY             SlcKey,
        IN      const SLC_BYTE      * pbIv,
        IN      const SLC_BYTE      * pbData,
        IN      SLC_ULONG           ulDataLen,
        OUT     SLC_BYTE            * pbOutBuf
        IN      SLC_ULONG           ulOutBufLen,
        OUT     SLC_ULONG           * pulOutDataLen
        );
```

Parameter Description

byMode	Encryption mode, Value at: SLC_MODE_ECB, SLC_MODE_CBC. Error than return with: SLC_ERROR_INVALID_MODE。
SlcKey	TDES Key in SLC_KEY format, cannot be: NULL When value at: NULL, then error return with: SLC_ERROR_INVALID_PARAMETER. When Key format invalid/error, then error return with: SLC_ERROR_BAD_KEY_VERSION

	SLC_ERROR_BAD_KEY_TYPE SLC_ERROR_BAD_KEY_ALGORITHM SLC_ERROR_INVALID_KEY_SIZE.
pbIv	Initialization Vector When: the `byMode` parameter value at: SLC_MODE_ECB, this parameter value can be set to: NULL; if not set to: NULL, it will not be valid & used, also no error returned. When: the `byMode` parameter value at: SLC_MODE_CBC, this parameter need to specify the buffer with 8 bytes as for vector initialized, and cannot be set: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
pbData	The data to be decrypted, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be decrypted, the length must be the multiple of 8, and also not be 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer to decrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (bytes) of output buffer, the length must not be less than: ulDataLen Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.16 SlcSha1Init

When you use the SHA-1 algorithm to calculate the HASH value of the data, Call this function to initialize the “CONTEXT” which required in the calculation process.

```
SLC_ULONG SLCAPI SlcSha1Init(
                    OUT     SLC_HASH_CTX            * pHashCtx
                    );
```

Parameter description

pHashCtx	Variable address to the variable type of “SLC_HASH_CTX”, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.17 SlcSha1Update

When you use SHA-1 algorithm to calculate the HASH of data, call this function to input the data to proceed

Before using this function, Call “SlcSha1Init” function to initialize the variable of “SLC_HASH_CTX”

If the data processed is long, the data can be split into several blocks and each call to this function to process one data block. No limit to the length of the data block.

```
SLC_ULONG SLCAPI SlcSha1Update(
                    IN OUT  SLC_HASH_CTX            * pHashCtx,
                    IN      SLC_BYTE                * pbData,
                    IN      SLC_ULONG               ulDataLen
                    );
```

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。 It will cause the error to access memory when use the variable of “SLC_HASH_CTX” without initialized
pbData	The buffer to store the data to be processed. Cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulDataLen	The length (in bytes) of the data to be output, no limitation to the length, can be 0, but if the total length of data is 0 also, then the output is uncertain, but no error reported.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.18 SlcSha1Final

When you use SHA-1 algorithm to calculate the HASH value of the data, call this function to close the calculation, and return with the digest of data, the length of digest is 20 bytes.

```
SLC_ULONG SLCAPI SlcSha1Final(
                IN OUT    SLC_HASH_CTX          * pHashCtx,
                OUT       SLC_BYTE              * pbOutBuf
                IN        SLC_ULONG             ulOutBufLen,
                OUT       SLC_ULONG             * pulOutDataLen
                );
```

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。

	It will cause the error to access memory when use the variable of “SLC_HASH_CTX” without initialized
pbOutBuf	The output buffer of calculation result, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (bytes) of output buffer, the length must be not less than 20 bytes (SLC_SHA1_DIGEST_LEN) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.19 SlcSha1

For the short data, call this function to complete data digest calculation of input data in one time, the data digest length is 20 bytes

```
SLC_ULONG SLCAPI SlcSha1(
                    IN      SLC_BYTE            * pbData,
                    IN      SLC_ULONG             ulDataLen,
                    OUT      SLC_BYTE             * pbOutBuf
                    IN      SLC_ULONG             ulOutBufLen,
                    OUT      SLC_ULONG            * pulOutDataLen
                    );
```

Parameter description

pbData	The buffer to store the data to be processed, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulDataLen	The length (in bytes) of the data to be output, no limitation to the length, can be 0, but if the data is 0 then the output is uncertain, no error reported also.
pbOutBuf	The output buffer to calculation result, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (bytes) of output buffer, the length must not be less than 20 bytes (SLC_SHA1_DIGEST_LEN) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.20 SlcSha256Init

When you use SHA-256 algorithm to calculate the HASH value of data, call this function to initialize the context used in the HASH calculation.

```
SLC_ULONG SLCAPI SlcSha256Init(
                    OUT     SLC_HASH_CTX            * pHashCtx
                    );
```

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.21 SlcSha256Update

When you use SHA-256 algorithm to calculate the HASH value of data, call this function to input the data to proceed

Before using this function, Call “SlcSha256Init” function to initialize the variable of “SLC_HASH_CTX”

If the data processed is long, the data can be split into several blocks and each call to this API to process one data block. No limit to the length of the data block.

```
SLC_ULONG SLCAPI SlcSha256Update(
                    IN OUT   SLC_HASH_CTX          * pHashCtx,
                    IN       SLC_BYTE              * pbData,
                    IN       SLC_ULONG               ulDataLen
                    );
```

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。 It will cause the error to access memory when use the variable of “SLC_HASH_CTX” without initialized
pbData	The buffer to store the data to be processed, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.

ulDataLen	The length (in bytes) of the data to be output, no limitation to the length, can be 0, but if the total length of data is 0 also, then the output is uncertain, no error reported also..

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.22 SlcSha256Final

When you use SHA-256 algorithm to calculate the HASH value of data, call this function to close the calculation and return with the digest of data, the digest length is 32 bytes.

```
SLC_ULONG SLCAPI SlcSha256Final(
        IN OUT   SLC_HASH_CTX          * pHashCtx,
        OUT      SLC_BYTE              * pbOutBuf
        IN       SLC_ULONG               ulOutBufLen,
        OUT      SLC_ULONG               * pulOutDataLen
        );
```

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。 It will cause the error to access memory when use the variable of “SLC_HASH_CTX” without initialized
pbOutBuf	The output buffer for HASH calculation, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of the output buffer, the length must not be less than 32 (SLC_SHA256_DIGEST_LEN) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。

pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.23 SlcSha256

For the short data, call this function to complete data digest calculation of input data in one time, the data digest length is 32 bytes

```
SLC_ULONG SLCAPI SlcSha256(
        IN      SLC_BYTE        * pbData,
        IN      SLC_ULONG        ulDataLen,
        OUT     SLC_BYTE        * pbOutBuf
        IN      SLC_ULONG        ulOutBufLen,
        OUT     SLC_ULONG       * pulOutDataLen
        );
```

Parameter description:

pbData	The buffer to store the data to be processed, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulDataLen	The length (in bytes) of the data to be output, no limitation to the length, can be 0, but if the data is 0 then the output is uncertain, no error reported also.
pbOutBuf	The output buffer to hash calculation, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。

ulOutBufLen	The length (in bytes) of output buffer, the length must not less than 32 (SLC_SHA256_DIGEST_LEN) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.24 SlcMd5Init

When you use MD5 algorithm to calculate the HASH value of data, call this function to initialize the context used in the HASH calculation.

```
SLC_ULONG SLCAPI SlcMd5Init(
                    OUT     SLC_HASH_CTX            * pHashCtx
                    );
```

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.25 SlcMd5Update

When you use MD5 algorithm to calculate the HASH value of data, call this function to input the data to proceed

Before use this function, Call “SlcShaMD5Init” function to initialize the variable of “SLC_HASH_CTX”

```
SLC_ULONG SLCAPI SlcMd5Update(
        IN OUT  SLC_HASH_CTX        * pHashCtx,
        IN      SLC_BYTE            * pbData,
        IN      SLC_ULONG           ulDataLen
        );
```

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。 It will cause the error to access memory when use the variable of “SLC_HASH_CTX” without initialized
pbData	The buffer to store the data to be processed, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulDataLen	The length (in bytes) of the data to be output, no limitation to the length, can be 0, but if the total length of data is 0 also, then the output is uncertain, no error reported also..

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.26 SlcMd5Final

When you use MD5 algorithm to calculate the HASH value of data, call this function to close the calculation and return with the digest of data, the digest length is 16 bytes.

SLC_ULONG SLCAPI SlcMd5Final(

IN OUT	SLC_HASH_CTX	* pHashCtx,
OUT	SLC_BYTE	* pbOutBuf,
IN	SLC_ULONG	ulOutBufLen,
OUT	SLC_ULONG	* pulOutDataLen

);

Parameter description

pHashCtx	The variable address of type of “SLC_HASH_CTX” variable, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。 It will cause the error to access memory when use the variable of “SLC_HASH_CTX” without initialized
pbOutBuf	The buffer to calculation of output, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of output buffer, the length must not be less than 16 (SLC_MD5_DIGEST_LEN) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.27 SlcMd5

For the short data, call this function to complete data digest calculation of input data in one time, the data digest length is 16 bytes

SLC_ULONG SLCAPI SlcMd5(

IN	SLC_BYTE	* pbData,
IN	SLC_ULONG	ulDataLen,
OUT	SLC_BYTE	* pbOutBuf
IN	SLC_ULONG	ulOutBufLen,
OUT	SLC_ULONG	* pulOutDataLen

);

Parameter description

pbData	The buffer to store the data to be processed, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulDataLen	The length (in bytes) of the data to be output, no limitation to the length, can be 0, but if the data is 0 then the output is uncertain, no error reported also.
pbOutBuf	The output buffer to calculation, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in Bytes) of output buffer, the length must not be less than 16 (SLC_MD5_DIGEST_LEN) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.28 SlcRsaGenerateKey

Generate the RSA public & Private key pair in “SLC_KEY” format

```
SLC_ULONG SLCAPI SlcRsaGenerateKey(
                    IN      SLC_ULONG           ulKeyBitLen,
                    OUT     SLC_KEY             * pSlcPubKey,
                    OUT     SLC_KEY             * pSlcPriKey
                    );
```

Parameter description

ulKeyBitLen	The bits length of key, value can be at: 1024, 2048, 4096 (SLC_RSA_KEY_BIT_1024, SLC_RSA_KEY_BIT_2048, SLC_RSA_KEY_BIT_4096 Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pSlcPubKey	Pass in the address for the variable of “SLC_KEY” which used to store the public key, cannot be NULL This function used for allocate the buffer to key, call the function: “SlcFreeKey” to release the key after used. Error then return with: SLC_ERROR_INVALID_PARAMETER.
pSlcPriKey	Pass in the address for the variable of “SLC_KEY” which used to store the private key, cannot be NULL This function used for allocate the buffer to key, call the function: “SlcFreeKey” to release the key when used. Error then return with: SLC_ERROR_INVALID_PARAMETER.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.29 SlcRsaPubEnc

Use the RSA public key in “SLC_KEY” format to encrypt data

```
SLC_ULONG SLCAPI SlcRsaPubEnc(
    IN      SLC_BYTE            byPadMode,
    IN      SLC_KEY             SlcKey,
    IN      const SLC_BYTE      * pbData,
    IN      SLC_ULONG           ulDataLen,
    OUT     SLC_BYTE            * pbOutBuf
    IN      SLC_ULONG           ulOutBufLen,
    OUT     SLC_ULONG           * pulOutDataLen
    );
```

Parameter description

byPadMode	Padding mode to encrypt data. Value can be selected: SLC_PAD_MODE_NONE, SLC_PAD_MODE_PKCS_1_V1_5.
SlcKey	The RSA public key in “SLC_KEK” format, Cannot be: NULL. Pass in: NULL, then error and return with: SLC_ERROR_INVALID_PARAMETER; Pass in: non RSA public Key (ECC private key, symmetric key), then error and return with: SLC_ERROR_BAD_KEY_TYPE; Pass in ECC public key, then error & return with: SLC_ERROR_BAD_KEY_ALGORITHM;
pbData	The data to be encrypted, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be encrypted, must be greater than 0 For padding mode: if the “byPadMode” value at: SLC_PAD_MODE_PKCS_1_V1_5, then the length (bytes) must be less or equal to (length in bit of key/8- 11); if the “byPadMode” value at: SLC_PAD_MODE_NONE, then the length (bytes) must be equal to (length in bit of key), and the highest bit of the first byte is 0. Error then return with: SLC_ERROR_INPUTDATA_LENGTH， SLC_ERROR_INVALID_INPUT_DATA.

pbOutBuf	The output buffer to encrypted data, cannot be: NULL Error than return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of output buffer, the length must not be less than (the length of bits of Key)/8 Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.30 SlcRsaPriDec

Use RSA private key (in SLC_KEY format) to decrypt the cipher text

```
SLC_ULONG SLCAPI SlcRsaPriDec(
                    IN      SLC_BYTE            byPadMode,
                    IN      SLC_KEY             SlcKey,
                    IN      const SLC_BYTE      * pbData,
                    IN      SLC_ULONG           ulDataLen,
                    OUT     SLC_BYTE            * pbOutBuf
                    IN      SLC_ULONG           ulOutBufLen,
                    OUT     SLC_ULONG           * pulOutDataLen
                    );
```

Parameter description

byPadMode	Padding mode

	2 Value can be selected: SLC_PAD_MODE_NONE, SLC_PAD_MODE_PKCS_1_V1_5.
SlcKey	The Public key in “SLC_KEY” format, cannot be “NULL” Pass in “NULL”, then error return with: SLC_ERROR_INVALID_PARAMETER Pass in Non RSA Private Key (ECC Public key, Symmetric key), then error return with: SLC_ERROR_BAD_KEY_TYPE； Pass in ECC private key, then error return with: SLC_ERROR_BAD_KEY_ALGORITHM;
pbData	The data to be decrypted, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The data length (in bytes) to be decrypted, which length must be equal to (bits length of key/8). Error then return with: SLC_ERROR_INPUTDATA_LENGTH.
pbOutBuf	The output buffer to data decrypted, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in Byte) of output buffer, if the buffer length is insufficient, then error will return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.31 SlcRsaPriEnc

Use the RSA private in SLC_KEY format to encrypt data

```
SLC_ULONG SLCAPI SlcRsaPriEnc(
                    IN      SLC_BYTE            byPadMode,
                    IN      SLC_KEY             SlcKey,
                    IN      const SLC_BYTE      * pbData,
                    IN      SLC_ULONG           ulDataLen,
                    OUT     SLC_BYTE            * pbOutBuf
                   IN      SLC_ULONG           ulOutBufLen,
                    OUT     SLC_ULONG           * pulOutDataLen
                    );
```

Parameter description

byPadMode	Padding mode Value can be selected: SLC_PAD_MODE_NONE SLC_PAD_MODE_PKCS_1_V1_5.
SlcKey	RSA private key with “SLC_KEY” format, cannot be : NULL. Pass in: NULL, then error return with: SLC_ERROR_INVALID_PARAMETER； Pass in None RSA private key (ECC public key, symmetric key), then error return with: SLC_ERROR_BAD_KEY_TYPE； Pass in ECC private key, then error return with: SLC_ERROR_BAD_KEY_ALGORITHM;
pbData	The data to be encrypted, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be encrypted, the length must be greater than 0 For padding mode: if the “byPadMode” value at: SLC_PAD_MODE_PKCS_1_V1_5, then the length (bytes) must be less or equal to (length in bit of key/8- 11); if the “byPadMode” value at: SLC_PAD_MODE_NONE, then the length (bytes) must be equal to (length in bit of key), and the highest bit of the first byte is 0. Error then return with: SLC_ERROR_INPUTDATA_LENGTH， SLC_ERROR_INVALID_INPUT_DATA.

pbOutBuf	The output buffer of Encrypted data, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in Byte) of output buffer, the length must not less than (length of key (in bit)/8) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.32 SlcRsaPubDec

Use RSA Public key with “SLC_KEY” format to decrypt the cipher text

```
SLC_ULONG SLCAPI SlcRsaPubDec(
                IN      SLC_BYTE            byPadMode,
                IN      SLC_KEY             SlcKey,
                IN      const SLC_BYTE      * pbData,
                IN      SLC_ULONG           ulDataLen,
                OUT     SLC_BYTE            * pbOutBuf
                IN      SLC_ULONG           ulOutBufLen,
                OUT     SLC_ULONG           * pulOutDataLen
                );
```

Parameter description

byPadMode	Padding mode Value can be selected: SLC_PAD_MODE_NONE, SLC_PAD_MODE_PKCS_1_V1_5.
SlcKey	The Public key in “SLC_KEY” format, cannot be: NULL. Pass in: NULL, then error return with: SLC_ERROR_INVALID_PARAMETER; Pass in Non RSA public key (ECC private key, symmetric key), then error return with: SLC_ERROR_BAD_KEY_TYPE; Pass in ECC Public key, then error return with: SLC_ERROR_BAD_KEY_ALGORITHM;
pbData	The data to be decrypted, cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in byte) of data to be decrypted, the length must be equal to (the length (in bits) of key/8) Error then return with: SLC_ERROR_INPUTDATA_LENGTH.
pbOutBuf	The output buffer to decrypted data, cannot be: NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of output buffer, if the buffer length is insufficient, then error will return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer ”Parameter description” & “Return value” accordingly

4.33 SlcRsaSign

Use RSA private key to sign the data

```
SLC_ULONG SLCAPI SlcRsaSign(
        IN      SLC_BYTE            byPadMode,
        IN      SLC_ULONG           ulHashAlgo,
        IN      SLC_KEY             SlcPriKey,
        IN      const SLC_BYTE      * pbMessage,
        IN      SLC_ULONG           ulMessageLen,
        OUT     SLC_BYTE            * pbOutBuf
        IN      SLC_ULONG           ulSignatureBufLen,
        OUT     SLC_ULONG           * pulSignatureLen
        );
```

Parameter description

byPadMode	Padding mode, Value at: SLC_PAD_MODE_PKCS_1_V1_5. Error then return with: SLC_ERROR_INVALID_MODE。
byHashAlgo	HASH 算法 ID. Hash Algorithm ID, which value can be selected: SLC_HASH_ALGO_SHA1, SLC_HASH_ALGO_SHA256, SLC_HASH_ALGO_MD5。 Error then return with: SLC_ERROR_INVALID_HASH_ALGO。
SlcPriKey	RSA Private key in “SLC_KEY” format, cannot be : NULL. When Pass in NULL, then error will return: SLC_ERROR_INVALID_PARAMETER； When Pass in None RSA private key (ECC public key, symmetric key), then error return with: SLC_ERROR_BAD_KEY_TYPE； When pass in ECC private key, then error return with : SLC_ERROR_BAD_KEY_ALGORITHM;
pbMessage	The data (message) to be signed, cannot be: NULL。 Error then return: SLC_ERROR_INVALID_PARAMETER

ulMessageLen	The length (in bytes) of data (message) to be signed, cannot be with 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH.
pbSignatureBuf	The output buffer to signed data, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulSignatureBufLen	The length (in bytes) of output buffer, the length must not less than (bits of key/8) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulSignatureLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.34 SlcRsaVerify

Use RSA public key to verify the data signed

```
SLC_ULONG SLCAPI SlcRsaVerify(
                IN      SLC_BYTE            byPadMode,
                IN      SLC_ULONG           ulHashAlgo,
                IN      SLC_KEY             SlcPubKey,
                IN      const SLC_BYTE      * pbMessage,
                IN      SLC_ULONG             ulMessageLen,
                IN      const SLC_BYTE      * pbSignature,
                IN      SLC_ULONG             ulSignatureLen
                );
```

Parameter description

byPadMode	Padding mode, Value at: SLC_PAD_MODE_PKCS_1_V1_5. Error then return with: SLC_ERROR_INVALID_MODE。
byHashAlgo	HASH Algorithm ID, value can be selected with: SLC_HASH_ALGO_SHA1, SLC_HASH_ALGO_SHA256, SLC_HASH_ALGO_MD5. Error then return with: SLC_ERROR_INVALID_HASH_ALGO。
SlcPubKey	RSA Public key in SLC_KEY format, cannot be: NULL. When pass in NULL, then error return with: SLC_ERROR_INVALID_PARAMETER; When pass in none RSA public Key (ECC private key, symmetric key), then error return with: SLC_ERROR_BAD_KEY_TYPE; When pass in ECC public key, then return with: SLC_ERROR_BAD_KEY_ALGORITHM;
pbMessage	The data (message) correspondence with signature, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulMessageLen	The Length (in byte) of data (message) correspondence with signature.
pbSignature	The buffer to store digital signature. Cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulSignatureLen	The length (in bytes) of signature, the length is equal to (the length (in bits) of key/8) Error then return with: SLC_ERROR_INPUTDATA_LENGTH。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.35 SlcRsaPubKeyToSlcKey

Convert the RSA public (none format) to the “SLC_KEY” format

```
SLC_ULONG SLCAPI SlcRsaPubKeyToSlcKey(
        IN      SLC_ULONG           ulKeyBitLen,
        IN      const SLC_BYTE      * pbRsaPubKey,
        IN      SLC_ULONG           ulPubKeyLen,
        OUT     SLC_KEY             * pSlcPubKey
        );
```

Parameter Description

ulKeyBitLen	The length (in bits) of key, which value can be selected: 1024, 2048, 4096 (SLC_RSA_KEY_BIT_1024, SLC_RSA_KEY_BIT_2048, SLC_RSA_KEY_BIT_4096) Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pbRsaPubKey	Pass in none format RSA Public key. Which cannot be: NULL。 When pass in: NULL， then error return with: SLC_ERROR_INVALID_PARAMETER；
ulPubKeyLen	The length of RSA public key RSA , cannot be 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH。
pSlcPubKey	Output buffer, used to store the converted Public key in SLC_KEY format, which cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.36 SlcRsaPriKeyToSlcKey

Convert the none format RSA private key to the key with SLC_KEY

```
SLC_ULONG SLCAPI SlcRsaPriKeyToSlcKey(
                    IN      SLC_ULONG           ulKeyBitLen,
                    IN      const SLC_BYTE    * pbRsaPriKey,
                    IN      SLC_ULONG           ulPriKeyLen,
                    OUT      SLC_KEY            * pSlcPriKey
                    );
```

Parameter description

ulKeyBitLen	The bits length of key, which value can be selected: 1024, 2048, 4096 (SLC_RSA_KEY_BIT_1024, SLC_RSA_KEY_BIT_2048, SLC_RSA_KEY_BIT_4096) Error then return: SLC_ERROR_INVALID_KEY_SIZE.
pbRsaPriKey	Pass in the none formatted RSA private key, cannot be: NULL。 When pass in: NULL， then Error return with: SLC_ERROR_INVALID_PARAMETER；
ulPriKeyLen	The length of RSA private key, cannot be 0。 Error then return with: SLC_ERROR_INPUTDATA_LENGTH。
pSlcPriKey	Output buffer, which used to store the converted private key in SLC_KEY format, cannot be NULL Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.37 SlcEccGenerateKey

To generate the ECC Public & Private key pair (in SLC_KEY format)

SLC_ULONG SLCAPI SlcEccGenerateKey(

IN	SLC_ULONG	ulKeyBitLen,
OUT	SLC_KEY	* pSlcPubKey,
OUT	SLC_KEY	* pSlcPriKey

);

Parameter description

ulKeyBitLen	The bits length of key, which value can be selected: 192, 256, 384 (SLC_ECC_KEY_BIT_192, SLC_ECC_KEY_BIT_256, SLC_ECC_KEY_BIT_384) Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pSlcPubKey	Pass in the address for the variable of “SLC_KEY” which used to store the public key, cannot be NULL This function used for allocate the buffer to key, call the function: “SlcFreeKey” to release the key after used. Error then return with: SLC_ERROR_INVALID_PARAMETER.
pSlcPriKey	Pass in the address for the variable of “SLC_KEY” which used to store the private key, cannot be NULL This function used for allocate the buffer to the key, call the function: “SlcFreeKey” to release the key after used. Error then return with: SLC_ERROR_INVALID_PARAMETER.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.38 SlcEccSign

Use ECC private key to sign the signature to message

SLC_ULONG SLCAPI SlcEccSign(

IN	SLC_ULONG	ulHashAlgo
IN	SLC_KEY	SlcPriKey,
IN	const SLC_BYTE	* pbMessage,
IN	SLC_ULONG	ulMessageLen,
OUT	SLC_BYTE	* pbSignatureBuf

```
    IN      SLC_ULONG           ulSignatureBufLen,
    OUT     SLC_ULONG           * pulSignatureLen
);
```

Parameter description

byHashAlgo	HASH algorithm ID. Value can be selected: SLC_HASH_ALGO_SHA1, SLC_HASH_ALGO_SHA256, SLC_HASH_ALGO_MD5。 Error then return with: SLC_ERROR_INVALID_HASH_ALGO。
SlcPriKey	The ECC private key in "SLC_KEY" format, cannot be: NULL. When pass in: NULL， then Error return with: SLC_ERROR_INVALID_PARAMETER； When pass in none ECC Private key(RSA Public key, Symmetric key), then error will return with: SLC_ERROR_BAD_KEY_TYPE； When pass in RSA Private key, then error will return with: SLC_ERROR_BAD_KEY_ALGORITHM;
pbMessage	The data to be signed, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulMessageLen	The length (in bytes) of message to be signed, cannot be: 0. Error then return with: SLC_ERROR_INPUTDATA_LENGTH.
pbSignatureBuf	The output buffer for signed message, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulSignatureBufLen	The length (bytes) of output buffer(signed message), usually the message signed by ECC signature is 56 bytes, and not greater than 128 bytes, if you want to get signed message in one time calling, it is recommend to set the length of output buffer no less than 128 bytes. Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulSignatureLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.39 SlcEccVerify

Use ECC public key to verify the message signed

```
SLC_ULONG SLCAPI SlcEccVerify(
        IN      SLC_ULONG           ulHashAlgo,
        IN      SLC_KEY             SlcKey,
        IN      const SLC_BYTE    * pbMessage,
        IN      SLC_ULONG             ulMessageLen,
        IN      const SLC_BYTE    * pbSig,
        IN      SLC_ULONG           ulSigLen,
        );
```

Parameter description

byHashAlgo	The HASH algorithm Type ID, which value can be set: SLC_HASH_ALGO_SHA1, SLC_HASH_ALGO_SHA256, SLC_HASH_ALGO_MD5. Error then return with: SLC_ERROR_INVALID_HASH_ALGO。
SlcPubKey	The ECC public key in “SLC_KEY” format, cannot be: NULL. When pass in: NULL, then error will return with: SLC_ERROR_INVALID_PARAMETER； When pass in: none ECC Public key (RSA private key, symmetric key), then error will return with: SLC_ERROR_BAD_KEY_TYPE； When pass in RSA public key, then error will return with: SLC_ERROR_BAD_KEY_ALGORITHM;
pbMessage	The data correspondence with signature, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulMessageLen	The Length (in bytes) of data (message) correspondence with signature.
pbSignature	The buffer to store the signature, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulSignatureLen	The length of signature (in bytes)

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.40 SlcEccPubKeyToSlcKey

Convert the “none format” ECC public key to the key with “SLC_KEY” format.

```
SLC_ULONG SLCAPI SlcEccPubKeyToSlcKey(
    IN      SLC_ULONG       ulKeyBitLen,
    IN      SLC_BYTE      * pbECCPubKey,
    IN      SLC_ULONG       ulECCPubKeyLen,
    OUT     SLC_KEY         * pSlcPubKey
    );
```

Parameter description

ulKeyBitLen	The bits length of key, which value can be selected: 192, 256, 384 (SLC_ECC_KEY_BIT_192, SLC_ECC_KEY_BIT_256, SLC_ECC_KEY_BIT_384) Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pbECCPubKey	Pass in none format ECC public key, cannot be: NULL。 When pass in NULL, then error will return with: SLC_ERROR_INVALID_PARAMETER；
ulECCPubKeyLen	The length of ECC public key, cannot be with 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH。
pSlcPubKey	Output buffer, used to store the converted public key in “SLC_KEY” format, cannot be with NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.41 SlcEccPriKeyToSlcKey

Convert “None format” ECC private key to “SLC_KEY” format

```
SLC_ULONG SLCAPI SlcEccPriKeyToSlcKey(
                    IN      SLC_ULONG           ulKeyBitLen,
                    IN      SLC_BYTE          * pbECCPriKey,
                    IN      SLC_ULONG           ulECCPriKeyLen,
                    OUT      SLC_KEY            * pSlcPriKey
                    );
```

Parameter description

ulKeyBitLen	The bits length of key, which value can be selected:192, 256, 384 (SLC_ECC_KEY_BIT_192, SLC_ECC_KEY_BIT_256, SLC_ECC_KEY_BIT_384) Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pbECCPriKey	Pass in none format ECC private key, cannot be: NULL。 When pass in NULL, then error returned with: SLC_ERROR_INVALID_PARAMETER；
ulECCPriKeyLen	The length of ECC private key, cannot be 0。 Error then return with:SLC_ERROR_INPUTDATA_LENGTH。
pSlcPriKey	Output buffer, used to store the converted private key in “SLC_KEY” format, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.42 SlcEccKeyToSlcKey

Convert the none format ECC public & private key pair to the key pair with “SLC_KEY”

```
SLC_ULONG SLCAPI SlcEccKeyToSlcKey(
                    IN      SLC_ULONG           ulKeyBitLen,
                    IN      SLC_BYTE          * pbECCKey,
```

```
    OUT     SLC_KEY             * pSlcPubKey,
    OUT     SLC_KEY             * pSlcPriKey
    );
```

Parameter description

ulKeyBitLen	The length (Bits) of key, which value can be selected: 192, 256, 384 (SLC_ECC_KEY_BIT_192, SLC_ECC_KEY_BIT_256, SLC_ECC_KEY_BIT_384) Error then return with: SLC_ERROR_INVALID_KEY_SIZE.
pbECCKey	ECC key pair, cannot be NULL Pass in NULL, then error return with: SLC_ERROR_INVALID_PARAMETER；
pSlcPubKey	Output buffer, used to store the converted public key in “SLC_KEY” format, cannot be NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER。
pSlcPriKey	Output buffer, used to store the converted private key in “SLC_KEY” format, cannot be NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER。

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.43 SlcHmacGenerateKey

Generate the HMAC key in SLC_KEY format

If you have “none format” key used previously, then you can use the “none format” key previously to initialize the key generated in “SLC_KEY” format, which make the “SLC_KEY” format key to contain the previously key in “None format”

```
SLC_ULONG SLCAPI SlcHmacGenerateKey(
    IN      const SLC_BYTE      * pbInitData,
    IN      SLC_ULONG           ulKeyBitLen,
    IN      SLC_ULONG           ulAlgorithm,
    OUT     SLC_KEY             * pSlcKey
    );
```

Parameter Description

pbInitData	The none format key used previously, the key length can be: 1~255 bytes, can be NULL When key value is NULL, then this function will generate a random number to be the key.
ulKeyBitLen	The bits length of key, the length of key generated can be the bytes of key (1~255)*8
ulAlgorithm	The type of HMAC key, please refer the macro definition: The type of definition of HMAC key
pSlcKey	Pass in a “SLC_KEY” variable address, cannot be NULL This function used for allocating the buffer for key stored, it is require to call ”SlcFreeKey” to release the key when used Error then return with value: SLC_ERROR_INVALID_PARAMETER.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.44 SlcHmacRaw

HMAC Function， use the none format Key in 1~256 bytes

```
SLC_ULONG SLCAPI SlcHmacRaw(
                    IN      SLC_BYTE            byAlgoHmac,
                    IN      const SLC_BYTE      * pbKey,
                    IN      SLC_ULONG           ulKeyLen,
                    IN      const SLC_BYTE      * pbData,
                    IN      SLC_ULONG           ulDataLen,
                    OUT     SLC_BYTE            * pbOutBuf
                   IN      SLC_ULONG           ulOutBufLen,
                    OUT     SLC_ULONG           * pulOutDataLen
                    );
```

Parameter description

bAlgoHmac	HASH calculation algorithm Value can be selected:

	SLC_CIPHER_ALGO_HMAC_MD5, SLC_CIPHER_ALGO_HMAC_SHA1, SLC_CIPHER_ALGO_HMAC_SHA256, Error then return with: SLC_ERROR_INVALID_MODE。
pbKey	The buffer to HMAC key, the length varies from 1~256 bytes, cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER.
ulKeyLen	The length (in bytes) of key “pbKey”, varies from 1~256 bytes. Error then return with: SLC_ERROR_INVALID_KEY_SIZE
pbData	The data to be calculated, cannot be: NULL。 Error then return: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length of data to be calculated (in bytes) Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer of data calculation, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of output buffer Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.45 SlcHmac

HMAC calculation function, which using the HMAC key in “SLC_KEY” format

SLC_ULONG SLCAPI SlcHmac(IN SLC_KEY SlcKey,

```
    IN      const SLC_BYTE      * pbData,
    IN      SLC_ULONG           ulDataLen,
    OUT     SLC_BYTE            * pbOutBuf
    IN      SLC_ULONG           ulOutBufLen,
    OUT     SLC_ULONG           * pulOutDataLen
    );
```

Parameter description

SlcKey	The HMAC key in SLC_KEY format, cannot be: NULL. When set to NULL, error will return with: SLC_ERROR_INVALID_PARAMETER. When key format incorrect, then error will return with: SLC_ERROR_BAD_KEY_VERSION SLC_ERROR_BAD_KEY_TYPE SLC_ERROR_BAD_KEY_ALGORITHM SLC_ERROR_INVALID_KEY_SIZE.
pbData	The data to be calculated , cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER
ulDataLen	The length (in bytes) of data to be calculated, the length must be the multiple of 16, and also not be 0 Error then return with: SLC_ERROR_INPUTDATA_LENGTH
pbOutBuf	The output buffer to calculated data. Cannot be with: NULL. Error then with: SLC_ERROR_INVALID_PARAMETER。
ulOutBufLen	The length (in bytes) of output buffer, the length must not be less than “ulDataLen” Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.46 SlcKeyToAesKey

Convert the key in “SLC_KEY” format to the AES key in “None Format”

```
SLC_ULONG SLCAPI SlcKeyToAesKey(
    IN      SLC_KEY         SlcKey,
    OUT     SLC_BYTE        * pbKeyBuf,
    IN      SLC_ULONG       ulKeyBufLen,
    OUT     SLC_ULONG       * pulKeyByteLen
    );
```

Parameter description

SlcKey	AES Key in “SLC_KEY” format, cannot be NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER. If the key is not symmetric key, then error will return with: SLC_ERROR_BAD_KEY_TYPE； If the key is the symmetric key, but not AES key, then error will return with : SLC_ERROR_BAD_KEY_ALGORITHM。
pbKeyBuf	The output buffer to store the converted key, in none format, which cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulKeyBufLen	The length (in bytes) of output buffer, value can set: 16, 24 & 32 (which correspondence with the key which the bits of key in 128, 192 & 256 respectively.) Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulKeyByteLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

4.47 SlcKeyToDesKey

Convert the Key in “SLC_KEY” format to DES key in “None format”

```
SLC_ULONG SLCAPI SlcKeyToDesKey(
    IN      SLC_KEY         SlcKey,
    OUT     SLC_BYTE        * pbKeyBuf,
```

```
    IN      SLC_ULONG       ulKeyBufLen,
    OUT     SLC_ULONG       * pulOutDataLen
    );
```

Parameter description

SlcKey	The DES key in “SLC_KEY” format, cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER. If the key is not symmetric key, then error return with: SLC_ERROR_BAD_KEY_TYPE； If the key is the symmetric key but not the DES Key, then error return with: SLC_ERROR_BAD_KEY_ALGORITHM。
pbKeyBuf	Output buffer to store the converted DES key in “none format”, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulKeyBufLen	The length (in bytes) of output buffer, not less than 8 bytes Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.48 SlcKeyToTDesKey

Convert the key in “SLC_KEY” format to the TDES key in “None format”

```
SLC_ULONG SLCAPI SlcKeyToTDesKey(
    IN      SLC_KEY         SlcKey,
    OUT     SLC_BYTE        * pbKeyBuf,
    IN      SLC_ULONG       ulKeyBufLen,
    OUT     SLC_ULONG       * pulOutDataLen
    );
```

Parameter description

SlcKey	The TDES key in “SLC_KEY:” format, cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER. If the key is not symmetric key, then error return with: SLC_ERROR_BAD_KEY_TYPE； If the key is symmetric key but not TDES key, then error return with: SLC_ERROR_BAD_KEY_ALGORITHM。
pbKeyBuf	The output buffer used to store the converted key in none format, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulKeyBufLen	The length (in bytes) of output buffer, the length is not less than 16 bytes. Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulOutDataLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

4.49 SlcKeyToHmacKey

Convert the key in “SLC_KEY” format to the HMAC key in “none format”

```
SLC_ULONG SLCAPI SlcKeyToHmacKey(
    IN      SLC_KEY     SlcKey,
    OUT     SLC_BYTE    * pbKeyBuf,
    IN      SLC_ULONG   ulKeyBufLen,
    OUT     SLC_ULONG   * pulOutDataLen，
    OUT     SLC_BYTE    * sbAlgorithm
    );
```

Parameter description

SlcKey	The HMAC Key in “SLC_KEY” format, cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER. If the key is not symmetric key, then error return with: SLC_ERROR_BAD_KEY_TYPE； If the key is symmetric key, but not the HMAC key, then error return with: SLC_ERROR_BAD_KEY_ALGORITHM。
pbKeyBuf	Output buffer, used to store the converted HMAC key in “None format”, cannot be: NULL

	Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulKeyBufLen	The length (in bytes) of output buffer. Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulKeyByteLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.
sbAlgorithm	The HASH algorithm which used by the Key.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.50 SlcKeyToRSAKey

Convert the Key in “SLC_KEY” format to the RSA key in “None format”

```
SLC_ULONG SLCAPI SlcKeyToRSAKey(
        IN      SLC_KEY         SlcKey,
        OUT     SLC_BYTE        * pbKeyBuf,
        IN      SLC_ULONG       ulKeyBufLen,
        OUT     SLC_ULONG       * pulKeyByteLen,
        OUT     SLC_ULONG       * pulKeyBitLen,
        OUT     SLC_BYTE        * pbKeyType
        );
```

Parameter description

SlcKey	The RSA key in “SLC_KEY” format, which cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER. If the key is not asymmetric key, then error return with: SLC_ERROR_BAD_KEY_TYPE； If the key is asymmetric key, but it is not RSA key, then error will return: SLC_ERROR_BAD_KEY_ALGORITHM。
pbKeyBuf	Output buffer used to store the converted key in “none format”, which cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。

ulKeyBufLen	The length (in bytes) of output buffer Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulKeyByteLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.
pulKeyBitLen	The bit length of key in output buffer Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pbKeyType	The type of key output, the possible return value can be: SLC_KEY_SYMMETRIC (symmetric key) SLC_KEY_PUBLIC (public key) SLC_KEY_PRIVATE (private key)

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.51 SlcKeyToECCKey

Convert the key in “SLC_KEY” format to ECC Key in none format

```
SLC_ULONG SLCAPI SlcKeyToECCKey(
        IN      SLC_KEY         SlcKey,
        OUT     SLC_BYTE        * pbKeyBuf,
        IN      SLC_ULONG       ulKeyBufLen,
        OUT     SLC_ULONG       * pulKeyByteLen,
        OUT     SLC_ULONG       * pulKeyBitLen,
        OUT     SLC_BYTE        * pbKeyType
        );
```

Parameter Description

SlcKey	The ECC key in “SLC_KEY” format, which cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER.

	If the key is not asymmetric key, then error will return with: SLC_ERROR_BAD_KEY_TYPE； If the key is asymmetric key, but it is not ECC key, then error will return with: SLC_ERROR_BAD_KEY_ALGORITHM。
pbKeyBuf	Output buffer, used to store the key converted in none format, which cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulKeyBufLen	The length (in bytes) of output buffer Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulKeyByteLen	Variable address which used to store the length (in bytes) of output data, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.
pulKeyBitLen	The bits length of key of the output buffer. Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pbKeyType	The type of key output, the return value can be: SLC_KEY_SYMMETRIC (symmetric key) SLC_KEY_PUBLIC (public key) SLC_KEY_PRIVATE (private key)

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.52 SlcKeyToAlgoKey

Convert the key in “SLC_KEY” format to correspondence key in “none format”

```
SLC_ULONG SLCAPI SlcKeyToAlgoKey(
                    IN      SLC_KEY         SlcKey,
                    OUT     SLC_BYTE        * pbKeyBuf,
                    IN      SLC_ULONG       ulKeyBufLen,
                    OUT     SLC_ULONG       * pulKeyByteLen,
                    OUT     SLC_BYTE        * pbAlgorithm,
                    OUT     SLC_ULONG       * pulKeyBitLen,
```

OUT SLC_BYTE * pbKeyType
);

Parameter description

SlcKey	The ECC key in “SLC_KEY” format, which cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER. If the key is not asymmetric key, error will return with: SLC_ERROR_BAD_KEY_TYPE； If the key is asymmetric key, but it is not the ECC key, then error will return with: SLC_ERROR_BAD_KEY_ALGORITHM。
pbKeyBuf	Output buffer, used to store the key converted in none format, which cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulKeyBufLen	The length (in bytes) of output buffer Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulKeyByteLen	Variable address which used to store the length (in bytes) of key in none format, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.
pbAlgorithm	The type of algorithm output Return value could be: SLC_CIPHER_ALGO_ECC, SLC_CIPHER_ALGO_RSA, SLC_CIPHER_ALGO_AES, SLC_CIPHER_ALGO_DES, SLC_CIPHER_ALGO_TDES, SLC_CIPHER_ALGO_HMAC_MD5, SLC_CIPHER_ALGO_HMAC_SHA1, SLC_CIPHER_ALGO_HMAC_SHA256,
pulKeyBitLen	The bit length of key of output buffer. Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pbKeyType	The type of key output, the value return could be: SLC_KEY_SYMMETRIC (symmetric key) SLC_KEY_PUBLIC (public key) SLC_KEY_PRIVATE (private key)

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.53 SlcCheckKeyType

To check & verify if the key type & algorithm in “SLC_KEY” format are consistent with the specified key type & algorithm, if they are consistent, then return with: SLC_SUCCESS.

Which used to import the key and to justify the key import is consistent with usage later, for tool applications.

```
SLC_ULONG SLCAPI SlcCheckKeyType (
                    IN      SLC_KEY         SlcKey,
                    IN      SLC_BYTE        byKeyType,
                    IN      SLC_BYTE        byAlgorithm
                    );
```

Parameter description

SlcKey	The key in “SLC_KEY” format, which cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER.
byKeyType	The key type of ID, which value can be selected: SLC_KEY_SYMMETRIC (symmetric key) SLC_KEY_PUBLIC (public key) SLC_KEY_PRIVATE (private key) For other value pass in, it will not be considered an error.
byAlgorithm	Key algorithm ID, Value can be selected: SLC_CIPHER_ALGO_AES, SLC_CIPHER_ALGO_DES, SLC_CIPHER_ALGO_TDES, SLC_CIPHER_ALGO_RSA, SLC_CIPHER_ALGO_ECC. For other value pass in, it will not be considered an error.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.54 SlcGetKeyType

Get The type of key in “SLC_KEY” format

```
SLC_ULONG SLCAPI SlcGetKeyType(
                    IN      SLC_KEY         SlcKey,
                    OUT     SLC_BYTE        * pbAlgorithm,
                    OUT     SLC_BYTE        * pbKeyType
                    );
```

Parameter description

SlcKey	The key in “SLC_KEY” format, cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER.
pbAlgorithm	The output key algorithm ID, which cannot be: NULL. The possible value output: SLC_CIPHER_ALGO_AES, SLC_CIPHER_ALGO_DES, SLC_CIPHER_ALGO_TDES, SLC_CIPHER_ALGO_RSA, SLC_CIPHER_ALGO_ECC.
pbKeyType	The key type of ID output, which cannot be: NULL. The possible value output: SLC_KEY_SYMMETRIC (Symmetric key) SLC_KEY_PUBLIC (Public key) SLC_KEY_PRIVATE (Private key)

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.55 SlcGetKeySize

Get the length (in bytes) of the key, which used for export the key, for tool applications.

```
SLC_ULONG SLCAPI SlcGetKeySize(
```

```
                    IN      SLC_KEY             SlcKey,
                    );
```

Parameter description

SlcKey	The key in “SLC_KEY” format, which cannot be: NULL. Error then return with: 0

Return Value description

Success then return with none zero value, if returns with 0, that means the key pass in is incorrect.

Note: This function returns 0 to indicate an error. (The value of SLC_SUCCESS is also 0, don’t be confused.)

4.56 SlcGetKeyBitLength

Get the bits length of key in “SLC_KEY” format, which used for tool applications.

```
SLC_ULONG SLCAPI SlcGetKeyBitLength(
                    IN      SLC_KEY             SlcKey
                    );
```

Parameter description

SlcKey	The key in “SLC_KEY” format, which cannot be: NULL. Error then return with: 0.

Return Value description

Success then return with none zero value, if returns with 0, that means the key pass in is incorrect.

Note: This function returns 0 to indicate an error. (The value of SLC_SUCCESS is also 0, don’t be confused.)

4.57 SlcExportKey

Export the key in SLC_KEY format to the buffer. Used for export key. For tool applications

```
SLC_ULONG SLCAPI SlcExportKey(
                    IN      SLC_KEY             SlcKey,
                    OUT     SLC_BYTE            * pbOutBuf
                    IN      SLC_ULONG           ulOutBufLen,
                    OUT     SLC_ULONG           * pulOutDataLen
                    );
```

Parameter description

SlcKey	The key in “SLC_KEY” format, cannot be: NULL. Error then return with: SLC_ERROR_INVALID_PARAMETER.
pbKeyBuf	The output buffer which used to store the key in “SLC_KEY” format, cannot be: NULL。 Error then return with: SLC_ERROR_INVALID_PARAMETER。
ulKeyBufLen	The length (bytes) of output buffer, the length of buffer required which can be got via “SlcGetKeySize” in advance. Error then return with: SLC_ERROR_INSUFFICIENT_BUFFER。
pulKeyByteLen	Variable address which used to store the length (in bytes) of the key in SLC_KEY format, When the output buffer length is insufficient, this variable returns the number of bytes required in the buffer. NULL can be passed in, which means that the length (in byte) of output data does not need to be returned.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & “Return value” accordingly

4.58 SlcImportKey

Import the key in the buffer and save with key pointer in “SLC_KEY” format, this function used to allocate the memory occupied by the key, the key import must be released after used.

Used to import key, for tools application.

```
SLC_ULONG SLCAPI SlcImportKey(
                    OUT     SLC_KEY             * pSlcKey,
```

```
    IN      const SLC_BYTE      * pbInputBuf,
    IN      SLC_ULONG             ulInputDataLen
    );
```

Parameter description

pSlcKey	The address of the variable in SLC_KEY format, cannot be: NULL. This function responsible to allocate the storage space to the key, after used the key imported, call the function: “SlcFreeKey” to release. Error then return with: SLC_ERROR_INVALID_PARAMETER.
pbInputBuf	The buffer to store the key data, cannot be: NULL
ulInputDataLen	The length (in bytes) of valid data in the buffer which store the key.

Return Value description

Success then return with: SLC_SUCCESS。

For other value returned, please refer: ”Parameter description” & following description attached in below

This function will check the length, format of data, it may return with following value:

SLC_ERROR_INPUTDATA_LENGTH,（(Data length error)

SLC_ERROR_BAD_KEY_VERSION, (Key version information error)

SLC_ERROR_BAD_KEY_TYPE, (Key type of ID error)

SLC_ERROR_BAD_KEY_FORMAT, (The data of the fields within the key do not match, such as the key type ID field and the key length field do not match)

SLC_ERROR_BAD_KEY_ALGORITHM, (Key algorithm ID error)

4.59 SlcFreeKey

Release all of capacity which stored & occupied by the key in “SLC_KEY” format.

Applicable for:

The key which used the series function of “SlcXXXGenerateKey” generated, Use this function to release capacity for all the key imported by use of “SlcImportKey” function when used.

```
SLC_ULONG SLCAPI SlcFreeKey(
    IN OUT  SLC_KEY             * pSlcKey
    );
```

Parameter description

pSlcKey	The address of variable in “SLC_KEY” format, can be NULL. And executed successful.

Return Value description

This function will only return with: SLC_SUCCESS

When the variable address pass in is not NULL, and the original value of Variable is not NULL, then the variable will be set to be: NULL when returned.

When the variable address pass in is not NULL, and the original value of this variable is NULL, then the variable Value still kept: NULL