GM

CRYPTOGRAPHY INDUSTRY STANDARD

OF THE PEOPLE'S REPUBLIC OF CHINA

ICS 35.040
L 80
File No.: 38314-2013

GM/T 0016-2012

Smart token cryptography application interface specification

智能密码钥匙密码应用接口规范

Issued on: November 22, 2012 **Implemented on: November 22, 2012**

Issued by: State Cryptography Administration

Table of Contents

Foreword

This Standard was drafted in accordance with the rules given in GB/T 1.1-2009.

Attention is drawn to the possibility that some of the elements of this Standard may be the subject of patent rights. The issuing authority shall not be held responsible for identifying any or all such patent rights.

This Standard was proposed by and shall be under the jurisdiction of Code Industry Standardization Technical Committee.

Appendix A of this standard is normative appendix.

Main drafting organizations of this Standard: Beijing Haitai Fangyuan Science and Technology Co., Ltd., Beijing Woqi Smart Technology Co., Ltd., Beijing Daming Wuzhou Technology Co., Ltd., Hublot Co., Ltd., Shenzhen Minghua Aohan Technology Co., Ltd., Wuhan Tianyu Information Industry Co., Ltd., Beijing Feitian Chengxin Technology Co., Ltd., Huaxiangteng Digital Technology Co., Ltd.

Main drafters of this Standard: Liu Ping, Guo Baoan, Shi Yuping, Liu Zengshou, Hu Junyi, Guan Yanjun, Xiang Li, Lei Jiye, Hu Peng, Zhao Zaixing, Duan Xiaoyi, Liu Yufeng, Liu Weifeng, Chen Ji, He Yongfu, Li Gaofeng, Huang Dongjie, Wang Jiancheng, Wang Xuelin, Zhao Liming.

This standard involves cryptographic algorithms related content, which is implemented in accordance with the relevant state laws and regulations.

Smart token cryptography application interface specification

1 Scope

This standard specifies the PKI cryptosystem-based smart token cryptographic application interface, describes the function, the data type, the definition of parameters and equipment security requirements of the cryptographic application interface.

This standard applies to the development, use and testing of smart token products.

2 Normative references

The following documents are essential to the application of this document. For the dated documents, only the versions with the dates indicated are applicable to this document; for the undated documents, only the latest version (including all the amendments) are applicable to this standard.

GM/T 0006 Cryptographic application identifier criterion specification

GM/T 0009 SM2 cryptography algorithm application specification

3 Terms and definitions

The following terms and definitions apply to this document.

3.1

Application

A structure including containers, device certification key and file, which has independent permission management.

3.2

Container

The unique storage space used to store the key in the cryptographic device.

3.3

Device

In this standard the smart token is collectively referred to as device.

3.4

Device authentication

Authentication of the application by the smart token.

3.5

Device authentication key

Key used for device authentication.

3.6

Label

Another name of device, which can be set by the user and stored inside the device.

3.7

Message authentication code; MAC

Message authentication algorithm output.

3.8

Administrator PIN

Administrator password, which is an ASCII string.

3.9

User PIN

The user's password, which is an ASCII string.

4 Abbreviations

The following abbreviations apply to this specification

API: Application Programming Interface

PKI: Public Key Infrastructure

Function prototype	ULONG DEVAPI SKF_Encrypt (HANDLE hKey, BYTE * pbData, ULONG ulDataLen, BYTE * pbEncryptedData, ULONG * pulEncryptedLen)	
Functional description	Single-group data encryption operation. It uses the specified encryption key to encrypt the specified data, the encrypted data only contains one group, the encrypted ciphertext is stored in the specified buffer area. SKF_Encrypt only encrypt a single-group data, the SKF_Encrypt must be called to initialize the encryption operation before calling the SKF_Encrypt. SKF_Encrypt is equivalent to calling SKF_EncryptUpdate first and then SKF_EncryptFinal.	
Parameter	hKey	[IN] Encryption key handle.
	pbData	[IN] Data to be encrypted.
	ulDataLen	[IN] Length of data to be encrypted.
	pbEncryptedData	[OUT] The encrypted data buffer pointer, which can be NULL, is used to obtain the encrypted data length.
	pulEncryptedLen	[IN, OUT] Input indicates the buffer length of the result data, output indicates the result data actual length.
Return value	SAR_OK	Success.
	Others	Error code.

7.6.21 Multi-group data encryption

Function prototype	ULONG DEVAPI SKF_EncryptUpdate (HANDLE hKey, BYTE * pbData, ULONG ulDataLen, BYTE * pbEncryptedData, ULONG * pulEncryptedLen)	
Functional description	Multi-group data encryption operation. It uses the specified encryption key to encrypt the specified data, the encrypted data contains multiple group, the encrypted ciphertext is stored in the specified buffer area. SKF_EncryptUpdate encrypts multi-group data, the SKF_EncryptInit must be called to initialize the encryption operation before calling the SKF_EncryptUpdate. SKF_EncryptFinal must be called to end the encryption operation after calling the SKF_EncryptUpdate.	
Parameter	hKey	[IN] Encryption key handle.
	pbData	[IN] Data to be encrypted.
	ulDataLen	[IN] Length of data to be encrypted.
	pbEncryptedData	[OUT] The encrypted data buffer pointer.
	pulEncryptedLen	[OUT] Return the encrypted data length.
Return value	SAR_OK	Success.
	Others	Error code.

7.6.22 End encryption

Function prototype	ULONG DEVAPI SKF_EncryptFinal (HANDLE hKey, BYTE * pbEncryptedData, ULONG * pEncryptedDataLen)	
Functional description	End the encryption of multi-group data, return the remaining encryption result. First call SKF_EncryptInit to initialize the encryption operation, then call SKF_EncryptUpdate to encrypt multi-group data, and finally call SKF_EncryptFinal to end the encryption of multi-group data.	
Parameter	hKey	[IN] Encryption key handle.
	pbEncryptedData	[OUT] Encrypted result buffer.

	SKF_DecryptUpdate decrypts multi-group data. Before calling SKF_DecryptUpdate, it must call SKF_DecryptInit to initialize the decryption operation. After calling SKF_DecryptUpdate, it must call SKF_DecryptFinal to end the decryption operation.	
Parameter	hKey	[IN] Decryption key handle.
	pbEncryptedData	[IN] Data to be decrypted.
	ulEncryptedLen	[IN] Length of data to be decrypted.
	pbData	[OUT] Pointer to the decrypted data buffer.
	pulDataLen	[IN, OUT] Input indicates the result data buffer length, output indicates the result data actual length.
Return value	SAR_OK	Success.
	Others	Error code.

7.6.26 End decryption

Function prototype	ULONG DEVAPI SKF_DecryptFinal (HANDLE hKey, BYTE * pbDecryptedData, ULONG * pulDecryptedDataLen)	
Functional description	End the decryption of multi-group data. First call SKF_DecryptInit to initialize decryption operation, and then call SKF_DecryptUpdate to decrypt multi-group data, the finally call SKF_DecryptFinal to end multi-group data decryption.	
Parameter	hKey	[IN] Decryption key handle.
	pbDecryptedData	[OUT] Point to the decrypted result buffer. If this parameter is NULL, the length of the decrypted result is returned by pulDecryptedDataLen.
	pulDecryptedDataLen	[IN, OUT] Input indicates the length of the pbDecryptedData buffer, output indicates the length of the decrypted result.
Return value	SAR_OK	Success.
	Others	Error code.

7.6.27 Cryptographic hash initialization

Function prototype	ULONG DEVAPI SKF_DigestInit (DEVHANDLE hDev, ULONG ulAlgID, ECCPUBLICKEYBLOB * pPubKey, unsigned char * pucID, ULONG ulIDLen, HANDLE * phHash)	
Functional description	Initialize the cryptographic hash calculation operation, specify the algorithm to calculate the cryptographic hash.	
Parameter	hDev	[IN] Device handle returned when the device is connected.
	ulAlgID	[IN] Cryptographic hash algorithm identifier.
	pPubKey	[IN] Signer public key, valid when ulAlgID is SGD_SM3.
	pucID	[IN] Signer ID value, valid when ulAlgID is SGD_SM3.
	ulIDLen	[IN] Signer ID length, valid when ulAlgID is SGD_SM3.
	phHash	[OUT] Cryptographic hash object handle.
Return value	SAR_OK	Success.
	Others	Error code.

Remarks: pPubKey and pucID are valid when ulAlgID is SGD_SM3 and ulIDLen is not 0, SM2 algorithm signature preprocessing 1 is executed. The calculation process follows GM/T 0009.

7.6.28 Single-group data cryptographic hash

Function prototype	ULONG DEVAPI SKF_Digest (HANDLE hHash, BYTE * pbData, ULONG ulDataLen, BYTE * pbHashData, ULONG * pulHashLen)	
Functional description	Cryptographic hash calculation for a single-group messages. Before calling SKF_Digest, it must call SKF_DigestInit to initialize the cryptographic hash calculation. SKF_Digest is equivalent to calling SKF_DigestFinal after multiple calling SKF_DigestUpdate.	
Parameter	hHash	[IN] Cryptographic hash object handle.
	pbData	[IN] Point to message data buffer.
	ulDataLen	[IN] The length of the message data.
	pbHashData	[OUT] Cryptographic hash data buffer pointer, when this parameter is NULL, the length of the cryptographic hash result is returned by pulHashLen.
	pulHashLen	[IN, OUT] Input indicates the length of the result data buffer. Output indicates the actual length of the result data.
Return value	SAR_OK	Success.
	Others	Error code.

7.6.29 Multi-group data cryptographic hash

Function prototype	ULONG DEVAPI SKF_DigestUpdate (HANDLE hHash, BYTE * pbData, ULONG ulDataLen)	
Functional description	Perform cryptographic hash calculation for multi-group messages. Before calling SKF_DigestUpdate, it must call SKF_DigestInit to initialize the cryptographic hash calculation; after calling SKF_DigestUpdate, it must call SKF_DigestFinal to end the cryptographic hash calculation.	
Parameter	hHash	[IN] Cryptographic hash object handle.
	pbData	[IN] Point to message data buffer.
	ulDataLen	[IN] The length of the message data.
Return value	SAR_OK	Success.
	Others	Error code.

7.6.30 End cryptographic hash

Function prototype	ULONG DEVAPI SKF_DigestFinal (HANDLE hHash, BYTE * pHashData, ULONG * pulHashLen)	
Functional description	End the cryptographic hash calculation of multi-group messages, save the cryptographic hash result to the specified buffer.	
Parameter	hHash	[IN] Cryptographic hash object handle.
	pHashData	[OUT] The returned cryptographic hash result buffer pointer, if this parameter is NULL, the length of the hash result is returned by pulHashLen.
	pulHashLen	[IN, OUT] Input indicates hash results buffer length, output indicates the length of the cryptographic hash result.

Parameter	hMac	[IN] Message authentication code handle.
	pbData	[IN] Point to the buffer of data to be calculated.
	ulDataLen	[IN] The length of data to be calculated.
Return value	SAR_OK	Success.
	Others	Error code.

Remarks: Before calling SKF_MacUpdate, SKF_MacInit must be called to initialize message authentication code calculation; after calling SKF_MacUpdate, SKF_MacFinal must be called to end the message authentication code operation of multi-group data.

7.6.34 End message authentication code operation

Function prototype	ULONG DEVAPI SKF_MacFinal (HANDLE hMac, BYTE * pbMacData, ULONG * pulMacDataLen)	
Functional description	End the message authentication code calculation operation of multi-group data.	
Parameter	hMac	[IN] Message authentication code handle.
	pbMacData	[OUT] Point to the message authentication code buffer, when this parameter is NULL, the length of message authentication code is returned by pulMacDataLen.
	pulMacDataLen	[OUT] When it is called, it indicates the maximum length of the message authentication code buffer, returns the length of the message authentication code.
Return value	SAR_OK	Success.
	Others	Error code.

Remarks: SKF_MacFinal must be used after SKF_MacUpdate.

7.6.35 Close cryptographic object handle

Function prototype	ULONG DEVAPI SKF_CloseHandle (HANDLE hHandle)	
Functional description	Close session key, cryptographic hash object, message authentication code object, ECC key negotiation, and other handles.	
Parameter	hHandle	[IN] Object handle to be closed.
Return value	SAR_OK	Success.
	Others	Error code.

8 Device security requirements

8.1 Device use phase

The use of device is divided into two phases:

a) Exit-factory phase: When the device is exit-factory, a device authentication key is preset. During this phase, other operations are forbidden except to modify the device authentication key and create an application operation.

b) Application phase: The device that has created the application enters the

application phase. At this stage, it can perform any operations.

8.2 Permission management

8.2.1 Permission classification

Permissions are divided into device permissions, user permissions, and administrator permissions.

a) Device permission: Get device permission after passing device authentication.

b) User permission: Get user permission after passing user PIN verification, user permission applies only to the application at which it is located.

c) Administrator permissions: Get administrator permission after passing the administrator PIN authentication, administrator permission only applies to the application at which it is located.

8.2.2 User of permission

The use of permissions follows the following requirements:

a) Device permissions are used only for creating applications, deleting applications and changing device authentication keys.

b) User permissions are required to create and delete containers.

c) The permission to create the file is specified when creating the application.

d) File read and write permissions are specified when creating the file.

e) The use of the private key in the container requires user permission.

f) Both the user PIN and administrator PIN have the maximum number of retries, which is set when creating the application. When the number of verification PIN error reaches the maximum number of retries, the PIN code is locked.

g) User PIN unlock requires administrator permission.

h) Change of user PIN requires user permission. Change of administrator PIN requires administrator permission.

8.2.3 Device authentication

Creation and deletion of application in device can only be performed after passing the device authentication.

Device authentication uses a block cipher algorithm and device authentication key. The certification process is as follows:

a) The authenticated party calls SKF_GenRandom function to obtain 8 bytes random number RND from the device, uses the 0x00 to fill it to the block length of the cryptographic algorithm, to form the data block D0;

b) The authenticated party encrypts D0 to obtain the encrypted result D1, and calls SKF_DevAuth() to send D1 to the device;

c) After the device receives D1, verify whether D1 is correct. If it is correct, it passes the device authentication; otherwise the device authentication fails.

8.2.4 PIN code security requirements

a) PIN length is not less than 6 bytes;

b) The PIN code shall be protected during the transmission between the device and this interface, to prevent the PIN code from being leaked;

c) The PIN code shall be securely stored in the device AND cannot be exported from the device.

8.3 Key security requirements

The key shall follow the following security requirements:

a) Random numbers generated within the device shall be true random numbers, and shall comply with the requirements of randomness detection;

b) The session key generated within the device shall use a random number;

c) The prime numbers used in the device to generate asymmetric keys shall satisfy the requirements of the prime;

d) The keys in the device shall have effective key protection mechanism to prevent dissection, detection and reading;

e) The key in the device shall be used in accordance with the permission requirements;

f) Keys other than the public key cannot appear outside the device in plaintext;

g) Signature private key must be generated in the device;

h) The private key in the container cannot be exported in any way;

Appendix A

(Normative)

Error code definition and description

Macros description	Predefined value	Descriptions
SAR_OK	0x00000000	Succeeded
SAR_FAIL	0x0A000001	Failed
SAR_UNKNOWNERR	0x0A000002	Exceptional error
SAR_NOTSUPPORTYETERR	0x0A000003	Unsupported service
SAR_FILEERR	0x0A000004	File operation error
SAR_INVALIDHANDLEERR	0x0A000005	Invalid handle
SAR_INVALIDPARAMERR	0x0A000006	Invalid parameter
SAR_READFILEERR	0x0A000007	Read file error
SAR_WRITEFILEERR	0x0A000008	Write file error
SAR_NAMELENERR	0x0A000009	Name length error
SAR_KEYUSAGEERR	0x0A00000A	Key usage error
SAR_MODULUSLENERR	0x0A00000B	Modulus length error
SAR_NOTINITIALIZEERR	0x0A00000C	Not initialized
SAR_OBJERR	0x0A00000D	Object error
SAR_MEMORYERR	0x0A00000E	Memory error
SAR_TIMEOUTERR	0x0A00000F	Overtime
SAR_INDATALENERR	0x00000010	Input data length error
SAR_INDATAERR	0x0A000011	Input data error
SAR_GENRANDERR	0x0A000012	Generated random number error
SAR_HASHOBJERR	0x0A000013	HASH object error
SAR_HASHERR	0x0A000014	HASH operation error
SAR_GENRSAKEYERR	0x0A000015	Generated RSA key error
SAR_RSAMODULUSLENERR	0x0A000016	RSA key modulus length error
SAR_CSPIMPRTPUBKEYERR	0x0A000017	CSP service import public key error
SAR_RSAENCERR	0x0A000018	RSA encryption error
SAR_RSADECERR	0x0A000019	RSA decryption error
SAR_HASHNOTEQUALERR	0x0A00001A	HASH value not equivalent
SAR_KEYNOTFOUNTERR	0x0A00001B	Key not found
SAR_CERTNOTFOUNTERR	0x0A00001C	Certificate not found
SAR_NOTEXPORTERR	0x0A00001D	Object not exported
SAR_DECRYPTPADERR	0x0A00001E	Patch error when decrypted

This is an excerpt of the PDF (Some pages are marked off intentionally)

------ The End ------