GM

CRYPTOGRAPHY INDUSTRY STANDARD

OF THE PEOPLE'S REPUBLIC OF CHINA

ICS 35.040

L 80

File No.: 38317-2013

GM/T 0019-2012

Universal cryptography service interface specification

通用密码服务接口规范

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.

Appendix A of this standard is normative Appendix.

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

Main drafting organizations of this Standard: Beijing Digital Certification Co., Ltd., Shanghai Geer Software Co., Ltd., Beijing Haitai Fangyuan Technology Co., Ltd., Wuxi Jiangnan Information Security Engineering Technology Center, Shanghai Digital Certificate Certification Center Co., Ltd., Guardian Information Industry Co., Ltd., Shandong De'an Information Technology Co., Ltd., National Information Security Engineering Technology Research Center.

Main drafters of this Standard: Liu Ping, Li Shusheng, Tan Wuzheng, Liu Zengshou, Xu Qiang, Liu Cheng, Li Yuanzheng, Gao Zhiquan, Kong Fanyu, Yuan Feng.

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

Universal cryptography service interface specification

1 Scope

This standard specifies a unified universal cryptography service interface.

This standard applies to the cryptography application service development under the public key application technology system, the R&D and detection of the cryptography application support platform, and to guide the development of the application system by direct use of the cryptography device.

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 0015 Digital certificate format based on SM2 algorithm

GM/T 0018 Interface specifications of cryptography device application

GM/T 0016 Smart token cryptography application interface specification

GM/T 0010 SM2 cryptography message syntax specification

GM/T 0009 SM2 Cryptography Algorithm Application Specification

PKCS #7: Cryptographic Message Syntax

3 Terms and definitions

The following terms and definitions apply to this document.

3.1

Digital certificate

Digital file signed by the authentication authority number, including public key owner information, public key, signer information, validation date, and

some extension information.

3.2

User key

An asymmetric key pair stored in the device that is used for application cryptographic operations, including a signature key pair and an encryption key pair.

3.3

Container

It is used in the cryptographic device to store the unique storage space divided by the key.

4 Symbols and abbreviations

The following abbreviations apply to this document:

API: Application Program Interface, referred to as application interface

CA: Certification Authority

CN: Common Name

CRL: Certificate Revocation List

DER: Distinguished Encoding Rules

DN: Distinguished Name

ECC: Elliptic Curve Cryptography

LDAP: Lightweight Directory Access Protocol

OID: Object IDentifier

PKCS: the Public-Key Cryptography Standard

5 Algorithm identification and data structure

5.1 Algorithm identifier and constant definition

The constant definitions used in this specification, the specific definitions of

operations are carried out in a safe and trusted program space. Environment class functions are also responsible for creating and managing the security access token between the user and the cryptographic devices. There are two types of user secure access tokens that can be created, one for normal users, this type of secure access token identifies that this user is a normal user, who can only access his/her own information and data in the cryptographic device; the other is for administrator, this type of secure access token identifies that this user is administrator, who can manage the security token of the normal user.

When the application uses the cryptography service interface, it must first call the initialization environment function (SAF_Initialize) to create and initialize the secure application space, to complete the connection and initialization with the cryptography device. Before aborting the application, it shall call the clear environment function (SAF_Finalize), to abort the connection to the cryptography device, destroy the security program space created, and prevent the security risk caused by memory residue. Application shall first call the user login function (SAF_Login) to establish the secure access token before performing any cryptography operation by calling any cryptography service function. After establishing the secure access token, it can call any cryptography service function. When no more cryptography service function is called, it shall call the logout function (SAF_Logout) to logout the secure access token, to avoid the cryptography device from illegal access.

6.2.3 Certificate class functions

Certificate class functions set various types of digital certificates to the application interface session environment to verify user certificates and get digital certificates or CRL, to provide a series of specific functions including certificate acquisition, CRL acquisition, CA root certificate setting, user certificate verification, and user certificate information acquisition. The application achieves digital certificate-based identity authentication through calling the certificate function, acquires relevant information from certificate, achieves authorization management, access control, and other security mechanism. The digital certificate formats covered in this standard shall follow GM/T 0015.

6.2.4 Cryptography operation class functions

The cryptography class function is responsible for interacting with the cryptography device to achieve a specific cryptographic operation, and returning the result of the cryptography operation back to the application, which is the foundation for the applications to achieve security mechanisms such as data confidentiality, integrity, and non-repudiation.

Cryptography operation class functions provide including base64 codec,

7.2 Certificate class function

7.2.1 Overview

Certificate class functions include the following specific functions, the return value of each function is as shown in the Appendix A: Error code definition:

a) Add root CA certificate: SAF_AddTrustedRootCaCertificate

b) Get number of root CA certificates: SAF_GetRootCaCertificateCount

c) Get root CA certificate: SAF_GetRootCaCertificate

d) Remove root CA certificate: SAF_RemoveRootCaCertificate

e) Add CA Certificate: SAF_AddCaCertificate

f) Get number of CA certificates: SAF_GetCaCertificateCount

g) Get the CA certificate: SAF_GetCaCertificate

h) Remove CA certificate: SAF_RemoveCaCertificate

i) Add CRL: SAF_AddCrl

j) Verify user certificate: SAF_VerifyCertificate

k) Get user certificate logout status by CRL file: SAF_VerifyCertificateByCrl

l) Get certificate status by OCSP: SAF_GetCertificateStateByOCSP

m) Get certificate from LDAP: SAF_GetCertificateFromLdap

n) Get CRL corresponding to the certificate from LDAP: SAF_GetCrlFromLdap

o) Get certificate information: SAF_GetCertificateInfo

p) Get certificate extension information: SAF_GetExtTypeInfo

q) Enumerate user certificates: SAF_EnumCertificates

r) Enumerate user key container information: SAF_EnumKeyContainerInfo

s) Release memory of enumeration user certificates: SAF_EnumCertificatesFree

t) Release memory of enumeration key container information:

i) Generate a random number: SAF_GenRandom

j) HASH operations: SAF_Hash

k) Create a HASH object: SAF_CreateHashObj

l) Remove HASH object: SAF_DestroyHashObj

m) Multiple HASH operations through the object: SAF_HashUpdate

n) End HASH operation: SAF_HashFinal

o) Generate an RSA key pair: SAF_GenRsaKeyPair

p) Get RSA public key: SAF_GetPublicKey

q) RSA signature operation: SAF_RsaSign

r) RSA signature operation for the file: SAF_RsaSignFile

s) RSA verification signature operation: SAF_RsaVerifySign

t) RSA verification for the file and its signature: SAF_RsaVerifySignFile

u) Certificate-based RSA public key authentication: SAF_VerifySignByCert

v) Generate ECC key pair: SAF_GenEccKeyPair

w) Get ECC public key: SAF_GetEccPublicKey

x) ECC signature: SAF_EccSign

y) ECC verification: SAF_EccVerifySign

z) ECC public key encryption: SAF_EccPublicKeyEnc

aa) Certificate-based ECC public key encryption: SAF_EccPublicKeyEncByCert

bb) Certificate-based ECC public key authentication: SAF_EccVerifySignByCert

cc) Create symmetric algorithm object: SAF_CreateSymmAlgoObj

dd) Generate a session key and use an external public key to encrypt output: SAF_GenerateKeyWithEPK

ee) Import encrypted session key: SAF_ImportEncedKey

	unsigned char * pucInData,	
	unsigned int uiInDataLen,	
	unsigned char * pucOutData,	
	unsigned int * puiOutDataLen);	
Description:	Perform base64 decoding for the input data.	
Parameter:	pucInData[in]	The data before decoding
	uiInDataLen[in]	The length of the data before decoding
	pucOutData[out]	The data after decoding
	puiOutDataLen[in, out]	The length of the data after decoding
Return value:	Zero	Succeeded
	Non-zero	Failed, return error code

7.3.4 Create base64 object

Prototype:	int SAF_Base64_CreateBase64Obj(	
	void * * phBase64Obj);	
Description:	Create a base64 object for base64 encoding and decoding of arbitrary length data.	
Parameter:	phBase64Obj[out]	Pointer to the handle of the base64 object created
Return value:	Zero	Succeeded
	Non-zero	Failed, return error code
Remarks:	This function is used together with such functions as	
	SGD_Base64_DestroyBase64Obj, SGD_Base64_EncodeUpdate，	
	SGD_Base64_EneodeFinal，SGD-Base64_DecodeUpdate，SGD_Base64_	
	DecodeFinal to support the base64 encoding and decoding of arbitrary length data.	

7.3.5 Destroy base64 object

Prototype:	int SAF_Base64_DestroyBase64Obj(	
	void * hBase64Obj);	
Description:	Delete base64 object.	
Parameter:	phBase64Obj[out]	Base64 object handle to be removed
Return value:	Zero	Succeeded
	Non-zero	Failed, return error code

7.3.6 Multi-block encoding via base64 object

Prototype:	int SAF_Base64_EncodeUpdate(	
	void * hBa5e64Obj,	
	unsigned char * pucInData,	
	unsigned int uiInDataLen,	
	unsigned char * pucOutData,	
	unsigned int * puiOutDataLen);	
Description:	Encode multi-block base64 of data via base64 object.	
Parameter:	hBase64Obj[in]	base64 object
	pucInData[in]	Data before encoding
	uiInDataLen[in]	The length of the data before encoding
	pucOutData[out]	Data after encoding
	puiOutDataLen[in, out]	Return the length of data after encoding

unsigned int uiDerSignedAndEnvelopedDataLen,
unsigned char * pucData,
unsigned int * puiDataLen,
unsigned char * pucSignerCertificate,
unsigned int * puiSignerCertificateLen,
unsigned int * puiDigestAlgorithms);

Description: Decode signed digital envelope data based on the SM2 algorithm.

Parameter:

hAppHandle[in]	Application interface handle
pucDecContainerName[in]	Container name of the decryption private key
uiDecContainerNameLen[in]	Container name length of the decryption private key
uiDecKeyUsage[in]	Usage of the decryption private key
pucDerSignedAndEnvelopedData[in]	Der-encoded SignedAndEnvelopedData digital envelope data, the SignedAndEnvelopedData data format is as defined in GM/T 0010
puiDerSignedAndEnvelopedDataLen[in]	Encoded signed digital envelope data length
pucData[out]	Original data after decoding
puiDataLen[in, out]	Original data length
pucSignerCertificate[out]	Signer certificate
puiSignerCertificateLen[in, out]	Signer certificate length
puiDigestAlgorithms[out]	HASH algorithm identifier

Return value:

Zero	Succeeded
Non-zero	Failed, return error code

7.4.12 Encode signature data based on SM2 algorithm

Prototype: int SAF_SM2_EncodeSignedData(
void * hAppHandle,
unsigned char * pucSignContainerName,
unsigned int uiSignContainerNameLen,
unsigned int uiSignKeyUsage,
unsigned char * pucSignerCertificate,
unsigned int uiSignerCertificateLen,
unsigned int uiDigestAlgorithms,
unsigned char * pucData,
unsigned int uiDataLen,
unsigned char * pucDerSignedData,
unsigned int * puiDerSignedDataLen);

Description: Encode signature data based on the SM2 algorithm.

Parameter:

hAppHandle[in]	Application interface handle
pucSignContainerName[in]	The container name of the signature private key
uiSignContainerNameLen[in]	The container name length of the signature private key

	unsigned char * pucEncCertificate,	
	unsigned int uiEncCertificateLen,	
	unsigned int uiSymmAlgorithm,	
	unsigned char * pucDerEnvelopedData,	
	unsigned int * puiDerEnvelopedDataLen);	
Description:	Encode digital envelope data based on the SM2 algorithm.	
Parameter:	hAppHandle[in]	Application interface handle
	pucData[in]	Data to be digitally enveloped
	uiDataLen[in]	Length of data to be digitally enveloped
	pucEncCertificate[in]	Der-encoded receiver certificate
	uiEncCertificateLen[in]	Receiver certificate length
	uiSymmAlgorithm[in]	Symmetric algorithm identifier
	pucDerEnvelopedData[out]	Der-encoded EnvelopedData digital envelope data, EnvelopedData data format definition follows GM/T 0010
	puiDerEnvelopedDataLen[in, out]	Encoded digital envelope data length
Return value:	Zero	Succeeded
	Non-zero	Failed, return error code

7.4.15 Decode digital envelope based on SM2 algorithm

Prototype:	int SAF_SM2_DecodeEnvelopedData(	
	void * hAppHandle,	
	unsigned char * pucDecContainerName,	
	unsigned int uiDecContainerNameLen,	
	unsigned int uiDecKeyUsage,	
	unsigned char * pucDerEnvelopedData,	
	unsigned int uiDerEnvelopedDataLen,	
	unsigned char * pucData,	
	unsigned int * puiDataLen);	
Description:	Decode digital envelope data based on the SM2 algorithm.	
Parameter:	hAppHandle[in]	Application interface handle
	pucDecContainerName[in]	Container name of decryption private key
	uiDecContainerNameLen[in]	Container name length of decryption private key
	uiDecKeyUsage[in]	Usage of decryption private key
	pucDerEnvelopedData[in]	Der-encoded EnvelopedData digital envelope data, EnvelopedData data format definition follows GM/T 0010
	uiDerEnvelopedDataLen[in]	Digital envelope data length
	pucData[out]	Data original text obtained after decoding
	puiDataLen[in, out]	Data original text length obtained after decoding
Return value:	Zero	Succeeded
	Non-zero	Failed, return error code

References

[1] GB/T 17964-2008 Information technology - Security techniques - Modes of operation for a block cipher

[2] GB/T 17903.1-2008 Information technology - Security techniques - Non-repudiation - Part 1: General

[3] GB/T 17903.2-2008 Information technology - Security techniques - Non-repudiation - Part 2: Mechanisms using symmetric techniques

[4] GB/T 17903.3-2008 Information technology - Security techniques - Non-repudiation - Part 3: Mechanisms using asymmetric techniques

[5] GB/T 18238.1-2000 Information technology - Security techniques - Hash function - Part 1: General

[6] GB/T 18238.2-2002 Information technology - Security techniques - Hash functions - Part 2: Hash functions using an n-bit block cipher

[7] GB/T 18238.3-2002 Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions

[8] GB/T 19713-2005 Information technology - Security techniques - Public key infrastructure - Online certificate status protocol

[9] GB/T 19771-2005 Information technology - Security technology - Public key infrastructure - Minimum interoperability specification for PKI components

[10] GB 15851 Information technology - Security techniques - Digital signature scheme giving message recovery

[11] RFC 2560 X.509 Internet public key infrastructure online certificate status protocol - OCSP

[12] RFC 2459 X.509 Internet public key infrastructure certificate and CRL profile

[13] RSA Security: Public-Key Cryptography Standards (PKCS)

[14] Pkcs # 11 Cryptographic Token Interface Standard

[15] IETF Rfc2459, Internet X.509 Public Key Infrastructure Certificate and CRL Profile

[16] IETF Rfc2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol

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

Full-copy PDF can be purchased from 1 of 3 websites:

1. https://www.ChineseStandard.us

- SEARCH the standard ID, such as GB 4943.1-2022.
- Select your country (currency), for example: USA (USD); Germany (Euro).
- Full-copy of PDF (text-editable, true-PDF) can be downloaded in 9 seconds.
- Tax invoice can be downloaded in 9 seconds.
- Receiving emails in 9 seconds (with download links).

2. https://www.ChineseStandard.net

- SEARCH the standard ID, such as GB 4943.1-2022.
- Add to cart. Only accept USD (other currencies - https://www.ChineseStandard.us).
- Full-copy of PDF (text-editable, true-PDF) can be downloaded in 9 seconds.
- Receiving emails in 9 seconds (with PDFs attached, invoice and download links).

3. https://www.google.com/search?tbm=bks&q=ChineseStandard.net

- SEARCH the standard ID, such as GB 4943.1-2022.
- Google Books -- Select your currency.
- Processed by Google (delivery, tax invoice etc.). Delivered in 9 seconds by Google.
- Tips: Download an unprotected True-PDF (text-editable) from Google-Books:
  1. https://play.google.com/books → 2. Sign in → Google account
  3. Find the BOOK you bought → 4. Click "3-dots" → Export
  5. Save as "*.pdf" (Save True-PDF to your local computer for offline reading/printing)

Translated by: Field Test Asia Pte. Ltd. (Incorporated & taxed in Singapore. Tax ID: 201302277C)

Accountable person and shareholder: Wayne Zheng

About Us (Goodwill, Policies, Fair Trading…): https://www.chinesestandard.net/AboutUs.aspx

Contact: Wayne Zheng, Sales@ChineseStandard.net

Linkin: https://www.linkedin.com/in/waynezhengwenrui/

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