GM

CRYPTOGROPHIC INDUSTRY STANDARD

OF THE PEOPLE'S REPUBLIC OF CHINA

ICS 35.030

CCS L 80

GM/T 0019-2023

Replacing GM/T 0019-2012

Universal cryptography service interface specification

通用密码服务接口规范

Issued on: December 04, 2023 **Implemented on: June 01, 2024**

Issued by: State Cryptographic Administration

Table of Contents

Universal cryptography service interface specification

1 Scope

This document specifies the data structure, interface description, function definition requirements for the general cryptographic service interface; describes the corresponding verification methods.

This document is applicable to the development of cryptographic application services within the public key cryptography application technology system, the development and testing of cryptographic application support platforms, the development of cryptographic device application systems.

2 Normative references

The contents of the following documents, through normative references, constitute essential provisions of this document. For dated references, only the version corresponding to that date applies to this document. For undated references, the latest version (including all amendments) applies to this document. GB/T25069 Information Security Technology - Terminology

GB/T 25069 Information security techniques - Terminology

GB/T 41389 Information security technology - SM9 cryptographic algorithm application specification

GM/T 0003.1 Public key cryptographic algorithm SM2 based on elliptic curves - Part 1: General

GM/T 0006 Cryptographic application identifier criterion specification

GM/T 0009 SM2 cryptographic algorithm usage specifications

GM/T 0010 SM2 cryptography message syntax specification

GM/T 0015 Digital certificate format based on SM2 algorithm

GM/T 0016 Smart token cryptography application interface specification

GM/T 0018 Interface specifications of cryptography device application

GM/T 0094 Public key cryptographic application technology framework specification

GM/Z 4001 Cryptographic terminology

3 Terms and definitions

For the purposes of this document, the terms and definitions defined in GB/T 25069 and GM/Z 4001, as well as the terms and definitions listed below, apply.

3.1

Container

A unique storage space in a cryptographic device used to store cryptographic keys.

4 Abbreviations

The following abbreviations apply to this document.

API: Application Program Interface

CA: Certification Authority

CN: Common Name

CRL: Certificate Revocation List

DER: Distinguished Encoding Rules

DN: Distinguished Name

LDAP: Lightweight Directory Access Protocol

OCSP: Online Certificate Status Protocol

OID: Object Identifier

PKCS: Public-Key Cryptography Standard

5 Algorithm identifiers and data structures

5.1 Algorithm identifiers and constant definitions

The constant definitions, algorithm identifiers, certificate resolution identifiers used in this document shall comply with GM/T 0006.

6.2 Cryptographic service interface composition and functional description

6.2.1 Overview

The general cryptographic service interface consists of the following components:

a) Environment class functions;

b) Certificate class functions;

c) Cryptographic operation class functions;

d) Message class functions.

The relevant interfaces in this document are defined in C language. When developing interfaces in other languages, it may refer to the C language interface definitions.

6.2.2 Environment class functions

The environment class functions are responsible for creating and managing the various resources and signals required in the program space and ensuring that the program space is protected from unauthorized access during application execution, which could result in information leakage. The environment class functions are responsible for establishing a secure connection with the cryptographic device, ensuring that subsequent operations are performed within the secure program space. The environment class functions are also responsible for creating and managing application interface handles between the user and the cryptographic device. Two types of application interface handles can be created: one for ordinary users, which identifies the user as an ordinary user and can only access their own information and data in the cryptographic device; the other for administrators, which identifies the user as an administrator and can manage ordinary user application interface handles.

When using the general cryptographic service interface, an application must first call the initialization environment function (SAF_Initialize) to create and initialize a secure program space, completing the connection and initialization with the cryptographic device. Before calling any cryptographic service functions, the application must ensure a secure operating environment. This can be achieved by using a whitelist mechanism or authenticating the access device. Before performing any private key calculations, the application must first call the user login function (SAF_Login), to create an application interface handle. When the cryptographic service function is no longer called, the logout function (SAF_Logout) shall be called to deregister the security access token to prevent unauthorized access to the cryptographic device. Before terminating the application, the cleanup function (SAF_Finalize) shall be called to terminate the connection to the cryptographic device and destroy the created secure program space, to prevent security

risks caused by residual memory.

6.2.3 Certificate class functions

Certificate class functions set various digital certificates into the application interface session environment, verify user certificates, obtain digital certificates or CRLs. They provide a series of specific functions, including certificate acquisition, CRL acquisition, CA root certificate setting, user certificate verification, user certificate information acquisition. Applications call certificate class functions to implement digital certificate-based identity authentication, obtain relevant information from the certificate, implement authorization management and access control security mechanisms. The digital certificate format mentioned in this document must comply with GM/T 0015.

6.2.4 Cryptographic operation functions

Cryptographic operation functions are responsible for interacting with cryptographic devices to perform specific cryptographic operations and returning the results to the application. They are the foundation for applications to implement data confidentiality, integrity, non-repudiation security mechanisms.

Cryptographic operation functions provide services including Base64 encoding and decoding, random number generation, cryptographic hashing, various symmetric and asymmetric cryptographic operations. Cryptographic service functions support cryptographic operations on both fixed-length and variable-length data. For fixed-length data, the corresponding functions can be directly called for processing. For variable-length data, the corresponding cryptographic operation object shall be created first; then the corresponding functions shall be called to continuously process the data. When data processing is complete, the corresponding function shall be called to indicate completion; finally, the corresponding function shall be called to destroy the corresponding cryptographic operation object.

6.2.5 Message class functions

Message class functions encapsulate data in a digital envelope format, to enable application system interoperability and information sharing. They comply with PKCS#7 when using the RSA algorithm and GM/T 0010 when using the National Cryptography Algorithm.

Message class functions provide data encoding and decoding, signature data encoding and decoding, digital envelope encoding and decoding. Data encoding and decoding formats must comply with PKCS#7 (RSA algorithm) or GM/T 0010 (SM2 algorithm); signature data encoding and decoding formats must comply with PKCS#7 (RSA algorithm) or GM/T 0010 (SM2 algorithm) and PKCS#7 (RSA algorithm); digital envelope encoding and decoding formats must comply with GM/T 0010 (SM2 algorithm). These functions facilitate application implementation of identity authentication, confidentiality, integrity, non-repudiation security measures.

The calculated digest value shall be consistent with the expected result. When pucOutData = NULL during the calculation, the actual length of the result data shall be returned through puiOutDataLen.

b) Abnormal verification:

The expected result is obtained.

8.5.2 Multi-block digest calculation

Verification purpose:

Verify that multi-block digest calculations can be performed correctly.

Verification condition:

The initialization environment function SAF_Initialize has been successfully called.

Verification process:

a) Normal verification:

Step 1: Call the SAF_CreateHashObj interface to create a HASH object;

Step 2: Call the SAF_HashUpdate interface to continue the HASH operation on multiple blocks of predetermined data;

Step 3: Call the SAF_HashFinal interface to complete the HASH operation and obtain the digest value;

Step 4: Call the SAF_DestroyHashObj interface to delete the HASH object;

b) Abnormal verification:

Calling the above interfaces with invalid parameters shall return an error code.

Pass criteria:

a) Normal verification:

When setting pucOutData = NULL in step 3, the expected result shall be obtained.

b) Abnormal verification:

8.6 Asymmetric encryption and decryption verification

8.6.1 SM2 public key encryption

Verification purpose:

Verify that the input data can be correctly encrypted using an external SM2 public key and the encrypted result is output.

Verification conditions:

Environment initialization is complete.

Verification process:

a) Normal verification:

Use the pre-defined SM2 key pair, to call the SAF_Sm2PublicKeyEnc interface to encrypt the pre-defined source text and return the encrypted result.

b) Abnormal verification:

Calling this interface with invalid parameters shall return an error code.

Pass criteria:

a) Normal verification:

Using the pre-defined private key to decrypt the encrypted ciphertext, the result is identical to the pre-defined source text. The encrypted data format shall conform to the definition of the encrypted data format in GM/T 0009.

b) Abnormal verification:

The expected result is obtained.

8.6.2 Certificate-based SM2 public key encryption

Verification purpose:

Verify whether the SM2 public key from an external certificate can be used to correctly encrypt input data and output the encrypted result.

Verification conditions:

Complete environment initialization and apply for a certificate using the pre-defined SM2 private key.

Verification process:

a) Normal verification:

Use the SAF_Sm2PublicKeyEncByCert interface within the pre-defined

container, to encrypt the pre-defined source text and return the encrypted result.

b) Abnormal verification:

Calling this interface with invalid parameters shall return an error code.

Pass criteria:

a) Normal verification:

Use the pre-defined private key to decrypt the encrypted ciphertext; the result is identical to the pre-defined source text. The encrypted data format shall conform to the definition of the encrypted data format in GM/T 0009.

b) Abnormal verification:

Expected result is obtained.

8.6.3 Certificate-based SM2 decryption

Verification purpose:

Verify whether the private key corresponding to the certificate identifier in the container can be correctly used to decrypt and output the original text.

Verification condition:

Environment initialization is complete; the private key corresponding to the certificate identifier exists.

Verification process:

a) Normal verification:

Use the SAF_Sm2PublicKeyDecByCert interface within the specified container, to decrypt the specified ciphertext and return the plaintext data.

b) Abnormal verification:

Calling this interface with invalid parameters shall return an error code; if the decryption key does not exist, an error code shall be returned.

Pass criteria:

a) Normal verification:

The returned plaintext data is consistent with the specified plaintext.

b) Abnormal verification:

The expected result is obtained.

8.7 Symmetric encryption and decryption verification

8.7.1 Single-block encryption

Verification purpose:

Verify whether the encryption operation of a single block of data can be performed correctly.

Verification conditions:

The environment initialization function SAF_Initialize, the user login function SAF_Login, the symmetric algorithm object SAF_CreateSymmAlgoObj have been successfully called; the session key has been generated, imported, or negotiated.

Verification process:

a) Normal verification:

Call the SAF_SymmEncrypt interface to calculate the ciphertext value for the predetermined data.

b) Abnormal verification:

Calling the SAF_SymmEncrypt interface with invalid parameters shall return an error code.

Pass criteria:

a) Normal verification:

The ciphertext value obtained during the calculation shall be consistent with the predetermined result. When pucOutData = NULL is set during the calculation, the actual length of the result data shall be returned through puiOutDataLen.

b) Abnormal verification:

The expected result is obtained.

8.7.2 Single-block decryption

Verification purpose:

Verify whether encryption operations for multiple blocks of data can be performed correctly.

Verification conditions:

The environment initialization function SAF_Initialize, the user login function SAF_Login, the symmetric algorithm object creation function SAF_CreateSymmAlgoObj have been successfully called; the session key has been generated, imported, or negotiated.

Verification process:

a) Normal verification:

Step 1: Call the SAF_SymmEncryptUpdate interface to encrypt multiple predetermined plaintext data blocks and return the ciphertext data.

Step 2: Call the SAF_SymmEncryptFinal interface to complete the encryption and obtain the ciphertext data.

b) Abnormal verification:

Calling the above interface with invalid parameters shall return an error code.

Skipping step 1 and proceeding directly to step 2 shall return an error code.

Pass criteria:

a) Normal verification:

The encrypted ciphertext returned by steps 1 and 2 is consistent with the predetermined ciphertext. In steps 1 and 2, if puiOutData = NULL, the actual length of the result data shall be returned via puiOutDataLen;

b) Abnormal verification:

Expected result shall be obtained.

8.7.3 Multi-block encryption

Verification purpose:

Verify that single-block data decryption can be performed correctly.

Verification conditions:

The environment initialization function SAF_Initialize, the user login function SAF_Login, the symmetric algorithm object creation function SAF_CreateSymmAlgoObj have been successfully called; the session key has been generated, imported, or negotiated.

Verification process:

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

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