GM

CRYPTOGRAPHY INDUSTRY STANDARD

OF THE PEOPLE'S REPUBLIC OF CHINA

ICS 35.040

L 80

File No.: 62991-2018

GM/T 0056-2018

Specification of cryptography application interface with multi-applications equipment

多应用载体密码应用接口规范

Issued on: May 02, 2018 **Implemented on: May 02, 2018**

Issued by: State Cryptography Administration

Table of Contents

Specification of cryptography application interface with multi-applications equipment

1 Scope

This Standard specifies the cryptography application interface of SM2/3/4 series of algorithms in multi-applications equipment, including:

- defines the identity of SM2/SM3/SM4 algorithms in multi-applications equipment.

- defines the cryptography application interface specifications of SM2/SM3/SM4 algorithms.

This Standard is applicable to the development of various multi-applications equipment, and can also be used to guide the cryptography application test of multi-applications equipment.

2 Normative references

The following documents are indispensable for the application of this document. For dated references, only the dated version applies to this document. For undated references, the latest edition (including all amendments) applies to this document.

GB/T 32905-2016 Information security techniques - SM3 cryptographic hash algorithm

GB/T 32907-2016 Information security technology - SM4 block cipher algorithm

GB/T 32918 Information security technology - Public key cryptographic algorithm SM2 based on elliptic curves

ISO 9797 Information technology - Security techniques - Message authentication codes (MACs)

RFC 2898 Specification of PKCS # 5

SM4 algorithm

An algorithm defined by GB/T 32907-2016.

3.8

SM9 algorithm

A public key cryptographic algorithm based on elliptic curves, using bilinear pairings.

4 Abbreviations

For the purpose this document, the following abbreviations apply.

AID: Application Identifier

AKEY: Auxiliary Key

API : Application Programming Interface

CBC: Cipher-block chaining

COS: Chip Operating System

ECB: Electronic Codebook

ISO: International Organization for Standardization

MAC: Message Authentication Code

MKEY: Message Key

OFB: Output Feedback

5 Multi-applications equipment system framework

Multi-applications equipment is composed of hardware layer, driver layer, OS layer and application layer, as shown in Figure 1.

algorithmic capability of the equipment.

c) Call the cryptography application interface: If the equipment supports the cryptographic algorithm, the upper computer software calls the user application on the equipment. After the user application receives the call command on the equipment, it processes according to the command.

d) Call the cryptography application interface: If the processing of the command requires the use of the cryptography function, the call is made through the cryptography application interface of the equipment's operating system.

e) Return the cryptography application interface call result: After the cryptography application API is called, process by the algorithm and the processing result is returned to the user application on the equipment.

f) Return the application call result: The user application on the equipment returns the call result to the upper computer software. The process ends.

6.2 Cryptographic algorithm capability identity

Multi-applications equipment shall identify the supported cryptographic algorithm capability so that the outer-equipment entity can know the algorithm supported by the equipment.

Since the multi-applications equipment can be implemented by using different technical solutions, such as Java technical solution, C technical solution, etc., the algorithm capability identity of the corresponding technical solutions is described in Clause 7.

6.3 Cryptography application interface specification

The interface specification that the cryptography shall call shall be defined in the multi-applications equipment to facilitate user application calls on the equipment.

Since the security equipment uses different technical solutions in the implementation, such as Java technical solution, C technical solution, etc., the call interface specification of the corresponding technical solutions is described in Clause 7.

7 Cryptography application interfaces of Java technology solution

7.1 Introduction

This clause describes the capability identity and application interface specification definition of the SM2/3/4 series of algorithms in multi-applications equipment using the Java technology solution. Applications using the SM2/3/4 series of algorithms can call these application interfaces to use cryptographic functions.

The key object in the cryptographic algorithm needs to create a key instance by the GMKeyBuilder.buildKey method, and then set the key value used by the key object by means of setXXX, etc., and the signature algorithm and encryption and decryption algorithm will use these key objects. Symmetric key objects include: SM4Key; asymmetric key objects include: SM2PrivateKey, SM2PublicKey.

The signature and verification algorithm is implemented by the GMSignature class. Before using signature and verification algorithm, it is required to get the instance object of the corresponding algorithm by the GMSignature.getInstance method, and then the corresponding key object is used to achieve the purpose of generating signature data and verifying the signature data.

The data encryption and decryption algorithm is implemented by the GMCipher class. Before using the data encryption and decryption algorithm, it is required to get the instance object of the corresponding algorithm by the GMCipher.getInstance method, and then the corresponding key object is used to achieve the purpose of encrypting or decrypting the data.

The data hash algorithm is implemented by the GMMessageDigest class. Before using the data hash algorithm, it is required to get the instance object of the SM3 algorithm by the GMMessageDigest.getInstance method, and then the data can be hashed.

The GMKeyPair class is defined in the cryptographic algorithm API, for generating an asymmetric key pair (SM2) in the cryptographic algorithm within the equipment.

7.2 Cryptographic algorithm capability identity

The definition of multi-applications equipment's cryptographic algorithm capability identity is shown in Table 1.

7.5.2.2 getA

7.5.2.2.1 Declaration

```
public short getA(
   byte buffer[],
   short offset
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.2.2.2 Description

The method is used to get the data of curve parameter A. The output data length is 32 bytes. The data format is big endian and right aligned, i.e. the least significant bit is the least significant bit of the last byte.

7.5.2.2.3 Parameters

buffer - This byte array holds the output data.

offset - The starting position of the data in the byte array.

7.5.2.2.4 Return value

The data length of curve parameter A, in bytes.

7.5.2.2.5 Throw exception

CryptoException.UNINITIALIZED_KEY - This exception is thrown when the key is in an uninitialized state.

NullPointerException - This exception is thrown when the keyData parameter is null.

ArrayIndexOutOfBoundsException - This exception is thrown when the offset parameter is negative or exceeds the buffer array, or when the length of offset plus the curve parameter A exceeds the length of the buffer data.

7.5.2.3 getB

7.5.2.3.1 Declaration

```
public short getB(
   byte buffer[],
   short offset
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.3.2.2 Description

The SM2 private key data got by this method. The output private key reference data length is 32 bytes. The data format is big endian and right aligned, i.e. the least significant bit is the least significant bit of the last byte.

7.5.3.2.3 Parameters

buffer - This byte array holds the returned private key data.

offset - The starting position of the data in the byte array.

7.5.3.2.4 Return value

The length of the private key data in bytes is fixed to 32 in this Specification.

7.5.3.2.5 Throw exception

CryptoException.UNINITIALIZED_KEY - This exception is thrown when the key is in an uninitialized state.

NullPointerException - This exception is thrown when the buffer parameter is null.

ArrayIndexOutOfBoundsException - This exception is thrown when the offset parameter is negative or exceeds the buffer array, or when the length of offset plus the key data exceeds the length of the buffer data.

7.5.3.3 setS

7.5.3.3.1 Declaration

```
public void setS(
   byte[] buffer,
   short offset
   short length
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.3.3.2 Description

This method is used to set the SM2 private key data. The input private key reference data length is 32 bytes. The data format is big endian and right aligned, i.e. the least significant bit is the least significant bit of the last byte.

7.5.3.3.3 Parameters

buffer - This byte array holds the private key data to be set.

the last 32 bytes is the y coordinate. The data format is big endian and right aligned, i.e. the least significant bit is the least significant bit of the last byte.

7.5.4.2.3 Parameters

buffer - This byte array holds the returned public key data.

offset - The starting position of the data in the byte array.

7.5.4.2.4 Return value

The length of the public key data is 64 bytes. The public key data format is x coordinate and y coordinate content, and does not contain identity information in front.

7.5.4.2.5 Throw exception

CryptoException.UNINITIALIZED_KEY - This exception is thrown when the key is in an uninitialized state.

NullPointerException - This exception is thrown when the buffer parameter is null.

ArrayIndexOutOfBoundsException - This exception is thrown when the offset parameter is negative or exceeds the buffer array, or when the length of offset plus the key data exceeds the length of the buffer data.

7.5.4.3 setW

7.5.4.3.1 Declaration

```
public void setW(
   byte[] buffer,
   short offset
   short length
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.4.3.2 Description

This method is used to set the SM2 public key data. The input public key reference data length is 64 bytes. The first 32 bytes is the x coordinate of the public key, and the last 32 bytes is the y coordinate. The data format is big endian and right aligned, i.e. the least significant bit is the least significant bit of the last byte.

7.5.4.3.3 Parameters

7.5.7.2 GMMessageDigest construction method

7.5.7.2.1 Declaration

Protected GMMessageDigest ();

7.5.7.2.2 Description

The constructor of the GMMessageDigest class.

7.5.7.2.3 return value

Return an instance of the GMMessageDigest class.

7.5.7.3 getInstance

7.5.7.3.1 Declaration

```
public static final MessageDigest getInstance(
   byte algorithm,
   boolean externalAccess
) throws CryptoException;
```

7.5.7.3.2 Description

Create a corresponding cryptographic algorithm message hash algorithm.

7.5.7.3.3 Parameters

algorithm - The message hash algorithm value.

externalAccess - Whether the specified message hash instance can be accessed by multiple application instances. true means that the message hash instance can be accessed by multiple application instances, and false means that it cannot be accessed by multiple application instances.

7.5.7.3.4 Return value

Return a message hash instance of the specified algorithm.

7.57.3.5 Throw exception

CryptoException.NO_SUCH_ALGORITHM - This exception is thrown when the specified algorithm is not supported.

7.5.8 GMKeyBuilder

7.5.8.1 General

Asymmetric key algorithms use a public key (encryption) or a private key (signature) for encryption. In addition, they use a private key (decryption) or a public key (authentication) for decryption.

The object of the GMCipher class is reset to the state when the init() method is called after a card pull or reset event occurs. When the key corresponding to the object of the GMCipher class is invalid [when the clear event related to the key object is encountered], the object of the GMCipher class also returns to the uninitialized state.

For the intermediate results of the calculation of GMCipher class instances, no transaction management is required.

The GMCipher class definition is shown in Table 11.

Table 11 -- GMCipher

Type	Definition	Description
public static final byte	ALG_SM4_CBC_NOPAD = 0x89	The encryption algorithm uses the SM4 algorithm CBC mode, without filing
public static final byte	ALG_SM4_CBC_ISO9797_M1 = 0x8A	The encryption algorithm uses the SM4 algorithm CBC mode, using the filling method of ISO 9797 method 1
public static final byte	ALG_SM4_CBC_ISO9797_M2 = 0x8B	The encryption algorithm uses the SM4 algorithm CBC mode, using the filling method of ISO 9797 method 2
public static final byte	ALG_SM4_CBC_PKCS5 = 0x8C	The encryption algorithm uses the SM4 algorithm CBC mode, using the filling method of PKCS5
public static final byte	ALG_SM4_ ECB_NOPAD = 0x8D	The encryption algorithm uses the SM4 algorithm ECB mode, without filing
public static final byte	ALG_SM4_ECB_ISO9797_M1 = 0x8E	The encryption algorithm uses the SM4 algorithm ECB mode, using the filling method of ISO 9797 method 1
public static final byte	ALG_SM4_ECB_ISO9797_M2 = 0x8F	The encryption algorithm uses the SM4 algorithm ECB mode, using the filling method of ISO 9797 method 2
public static final byte	ALG_SM4_ECB_PKCS5 = 0x90	The encryption algorithm uses the SM4 algorithm ECB mode, using the filling method of PKCS5
public static final byte	ALG_SM2_WITH_SM3_NOPAD = 0xA1	The encryption algorithm uses SM3 as the digest algorithm and uses SM2 for encryption and decryption, without filling the data

Return a GMCipher class object of the specified algorithm.

7.5.9.3.5 Throw exception

CryptoException.NO_SUCH_ALGORITHM - This exception is thrown when the specified algorithm is not supported.

7.5.10 GMSM2KeyExchange

7.5.10.1 General

GMSM2KeyExChange class is the SM2 key exchange algorithm class in cryptographic algorithms. It implements the relevant contents of the SM2 key exchange algorithm in cryptographic algorithms.

The definition of GMSM2KeyExchange class is shown in Table 12.

Table 12 -- Definition of GMSM2KeyExchange class

Type	Definition	Description
public static final byte	SEND_MODE = 1	The key exchange process is the initiator
public static final byte	RECIVE_MODE = 0	The key exchange process is the responder
public static final byte	KEY_LEN_8 = 8	Exchange 8-byte keys
public static final byte	KEY_LEN_16 = 16	Exchange 16-byte keys
public static final byte	PARAM_THIS_ID = 1	Set the parameter to one's own ID
public static final byte	PARAM_OTHER_ID = 2	Set the parameter to the other party's ID
public static final byte	PARAM_THIS_FIX_PUBLICKEY = 3	Set the parameter to one's own fixed public key
public static final byte	PARAM_THIS_FIX_PRIVATEKEY = 4	Set the parameter to one's own fixed private key
public static final byte	PARAM_THIS_TEMP_PUBLICKEY_X = 7	Set the parameter to X part of one's own temporary public key
public static final byte	PARAM_THIS_TEMP_PRIVATEKEY = 5	Set the parameter to one's own temporary private key
public static final byte	PARAM_OTHER_FIX_PUBLICKEY = 6	Set the parameter to the other party's fixed public key
public static final byte	PARAM_OTHER_ TEMP_PUBLICKEY = 8	Set the parameter to the other party's temporary

null.

ArrayIndexOutOfBoundsException- This exception is thrown when the offset parameter is negative or exceeds the buffer array, or when the length of offset plus the parameter exceeds the length of the buffer data.

CryptoException.ILLEGAL_VALUE - This exception is thrown when the input parameter is incorrect.

7.5.10.7 getParam

7.5.10.7.1 Declaration

```
Public short getParam(byte[] buffer, short offset, byte type);
```

7.5.10.7.2 Description

Get the parameter that has been written and used for the key exchange.

7.5.10.7.3 Parameters

buffer - The array where the read data is.

offset - The read data position offset.

type - The read data type, e.g. PARAM_THIS_ID means to read one's own ID parameter.

7.5.10.7.4 Return value

Read data length.

7.5.10.7.5 Throw exception

NullPointerException - This exception is thrown when the buffer parameter is null.

ArrayIndexOutOfBoundsException- This exception is thrown when the offset parameter is negative or exceeds the buffer array, or when the length of offset plus the parameter exceeds the length of the buffer data.

CryptoException.UNINITIALIZED_KEY - This exception is thrown when the corresponding parameter is in an uninitialized state.

CryptoException.ILLEGAL_VALUE - This exception is thrown when the input parameter is incorrect.

IDLength - The ID value length.

PubKey - The public key object.

destBuffer - The array where the Za value is stored.

destOffset - The Za value offset.

type - The curve parameter type, currently only supports one more, i.e. PARAM_FP_256.

7.5.11.4.4 Return value

Za length.

7.5.11.4.5 Throw exception

CryptoException.ILLEGAL_VALUE - This exception is thrown when the input parameter is incorrect.

7.5.11.5 generateSM2Pubkey

7.5.11.5.1 Declaration

```
public static short generateSM2Pubkey (SM2PublicKey public_key, SM2PrivateKey private_key);
```

7.5.11.5.2 Description

Output the corresponding public key information according to the known private key object.

7.5.11.5.3 Parameters

public_key - The output object of the public key data.

private_key - The known private key object.

7.5.11.5.4 Return value

The length of the public key.

7.5.11.5.5 Throw exception

CryptoException.INVALID_INIT - This exception is thrown when the private key is not initialized.

NullPointerException - This exception is thrown when the public key object is null.

Annex A

(Informative)

Cryptography application requirements for multi-application security management

A.1 Introduction

This annex describes the cryptography use requirements for multi-application security management in multi-applications equipment. Multi-application management adopts the technical solution of security domain. Each security domain is divided into a separate logical security zone, which can be configured with an independent cryptography. Cryptography use in the secure domain includes:

- acquisition of cryptographic algorithm identity in security domains;
- secure channel management of out-of-equipment entities and security domains;
- signature and verification of CAP packages;
- authorization token for downloading, installing, migrating and deleting of applications;
- entrusted management receipts.

The equipment's multi-application security management technology framework is shown in Figure A.1.

phase. After the personalization is completed, modification is not allowed during the lifecycle of the equipment.

A.3 Cryptography application rules for secure channel

A.3.1 SCP02 secure channel

SCP02 secure channel shall be capable of supporting the SM4 symmetric cryptographic algorithm in this Specification. The out-of-equipment entity may first get the SCP02's cryptographic algorithm identity by getdata to confirm the used algorithm, or confirm the used algorithm by default convention.

When the SM4 algorithm is selected, the three initialized keys, i.e. encryption, MAC and sensitive data, of security domain are written into the security domain by the master security domain through the PUT KEY command. The Check value of the PUT KEY command takes the left three bytes. When the out-of-equipment entity opens the SCP02 secure channel, the session key generation rules comply with the requirements of the SCP02 secure channel.

When the SM4 algorithm is selected, sensitive data encryption uses ECB (128-bit) mode, encryption and MAC uses CBCX (128-bit) mode. The specific operation process of data elements for encryption and MAC calculation is as follows:

a) Process key calculation

Calculate according to the calculation process of SCP02, using SM4 algorithm.

b) Card cryptogram calculation

The implementation of the card cryptogram requires connection to an 8-byte Host Challenge and a 2-byte sequence counter generated outside the card, and a 6-byte Card Challenge generated in the card, to generate a 16-byte data block, which is not filled.

The signature method uses the S-ENC session key and ICV that are all binary 0 to act on the 16-byte data block. After SM4 encryption, the last 8 bytes are the authentication cryptography of the card.

c) Host cryptogram

The implementation of the host cryptogram requires connection to a 2-byte sequence counter, and a 6-byte Card Challenge generated in the card and an 8-byte Host Challenge generated outside the card, to generate a 16-byte data block.

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

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