ICS 35.240
CCS L 70/84

T/ZISIA

团　　体　　标　　准

T/ZISIA 0102.2—2025

通用操作系统商用密码子系统功能调用接口规范 第2部分：用户态接口

General purpose operating system commercial cryptographic subsystem function calling interface specification
Part 2:User mode interface

2025－12－31 发布　　2025－12－31 实施

中关村网络安全与信息化产业联盟　　发 布

目　　次

前　言

T/ ZISIA 0102《通用操作系统商用密码子系统功能调用接口规范》分为以下2个部分：

——第 1 部分：内核态接口

——第 2 部分：用户态接口

本文件为T/ ZISIA 0102《通用操作系统商用密码子系统功能调用接口规范》的第2部分。

本文件按照 GB/T 1.1—2020《标准化工作导则 第 1 部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由中关村网络安全与信息化产业联盟提出并归口。

本文件起草单位：中关村网络安全与信息化产业联盟、蚂蚁科技集团股份有限公司、麒麟软件有限公司、中科方德软件有限公司、兴唐通信科技有限公司、北京天威诚信电子商务服务有限公司、阿里云计算有限公司、长春吉大正元信息技术股份有限公司。

本文件主要起草人：张成龙、王宇辰、杨洋、洪澄、郭智慧、昌文婷、彭晋、王震、张大朋、孟德慧、陈刚、胡昆、冯建茹、刘赢、郭海兵、魏刘虎、王辉、姚长远、王尧、李延昭、李世奇、张天佳、王克、胡金山、罗捷、刘海涛、林璟锵、何道君、申志军、王立臣。

引　言

T/ZISIA 0101—2025《通用操作系统商用密码子系统安全轮廓》要求操作系统密码子系统应具有OS内核态和用户态商密API，以及OS内核态和用户态商密资源挂接标准化接口。

本文件的主要目标是为通用操作系统密码子系统制定统一的用户态密码应用接口及密码资源挂接接口标准，为通用操作系统密码子系统产品的研制和使用，以及基于该类密码产品的应用开发提供标准依据和指导。

通用操作系统商用密码子系统
功能调用接口规范
第 2 部分：用户态接口

1 范围

本文件规定了通用操作系统密码子系统用户态密码功能调用接口框架、数据类型定义、OS用户态商密API和OS用户态商密资源挂接接口。

本文件适用于通用操作系统密码子系统产品的研制和使用，以及基于该类密码产品的应用软件以及密码模块软件的开发。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 15852.2 信息技术 安全技术 消息鉴别码 第2部分：采用专用杂凑函数的机制
GB/T 32905 信息安全技术 SM3密码杂凑算法
GB/T 32907 信息安全技术 SM4分组密码算法
GB/T 32918 (所有部分) 信息安全技术 SM2椭圆曲线公钥密码算法
GB/T 35276 信息安全技术 SM2密码算法使用规范
GB/T 36624 信息技术 安全技术 可鉴别的加密机制
GB/T 37092 信息安全技术 密码模块安全要求
GB/T 38636 信息安全技术 传输层密码协议（TLCP）
GM/Z 4001 密码术语
T/ZISIA 0101-2025 通用操作系统商用密码子系统安全轮廓

3 术语和定义

GM/Z 4001界定的术语和定义适用于本文件。

3.1
密码模块 cryptographic module
实现了安全功能的硬件、软件和/或固件的集合，并且被包含在密码边界内。
[来源：GB/T 37092，3.5]

3.2
引擎 Engine
一种用于封装操作系统底层硬件或软件设备的机制，可为操作系统商密子系统提供标准化的密码资源挂接方法，实现商密资源的动态调度、性能优化和安全控制。

3.3
上下文 Context
一个封装了密码学操作全生命周期状态的不透明对象，用于保存操作状态和算法参数、管理内部资源，其内部结构由具体实现定义。

4 符号和缩略语

下列缩略语适用于本文件。

API：应用程序接口/应用接口（Application Program Interface）
CA：证书认证机构（Certification Authority）
CBC：密文分组链接模式（Cipher Block Chaining）
CCM：带有密文分组链接消息验证码的计数器模式（Counter with CBC - MAC）
CFB：密文反馈模式（Cipher Feedback）
CRL：证书撤销列表（Certificate Revocation List）
CTR：计数器模式（Counter）
ECB：电码本模式（Electronic Code Book）
GCM：伽罗瓦/计数器模式（Galois/Counter Mode）
HMAC：基于杂凑的消息鉴别码（Hash based Message Authentication Code）
OID：对象标识符（Object Identifier）
OFB：输出反馈模式（Output Feedback）
OS：操作系统（Operation System）
OSSM：操作系统商用密码（Operation System Commercial Cryptographic）
PEM：隐私增强邮件标准规定的证书编码格式（Privacy Enhanced Mail）
XTS：带密文挪用的XEX可调分组密码工作模式（XEX tweakable block cipher with Ciphertext Stealing）

5 密码功能调用接口框架

T/ZISIA 0101-2025【要求6.4-01】要求操作系统（OS）商密子系统应具有以下密码功能调用接口方式：

——OS内核态、用户态商密API。用于内核及用户态商密应用操作调用。

——OS内核态、用户态商密资源挂接接口。用于商密子系统挂接内核态、用户态OS商密资源。

商密资源包括：软硬件密码模块（如密码算法库、密码机、密码钥匙、TPM/TCM、CPU等）及其驱动程序、中间件等。

注：用户态和核心态使用的密码资源可以是同一密码资源，也可以是不同的密码资源。

本文件规范了商密子系统OS用户态商密功能调用接口，包括OS用户态商密API（北向接口）和OS用户态商密资源挂接接口（南向接口）。OS用户态商密功能调用接口框架如图1所示。

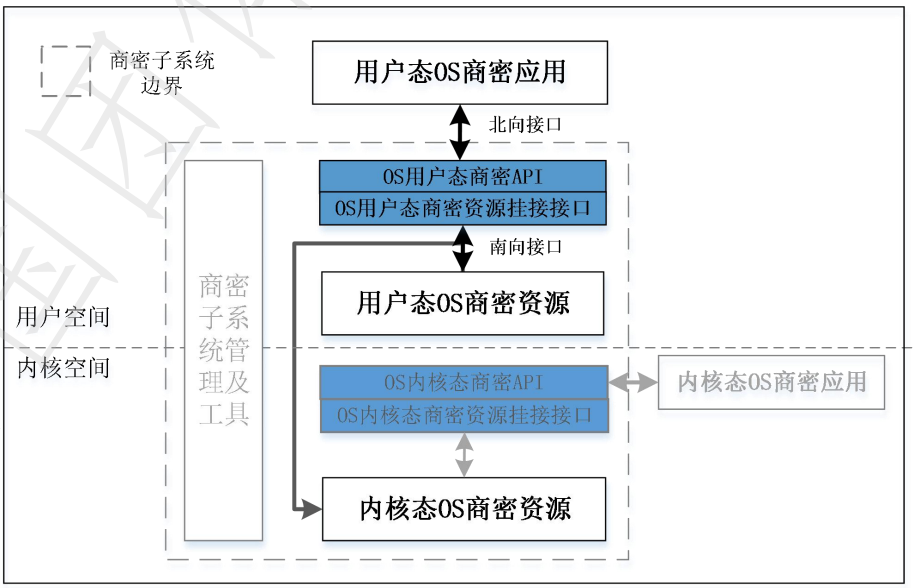

图 1 OS 用户态商密功能调用接口框架

用户态商密应用调用北向接口（第7章）实施密码操作，北向接口返回结果分两种情况：

1） 商密子系统使用缺省密码算法完成密码功能，直接返回结果；

2） 商密子系统使用挂接的软硬件密码资源完成密码功能，返回结果，条件是：

——用户态商密应用先使用引擎操作函数（7.11 章节）对密码资源进行加载和注册；

——密码资源厂商使用南向接口（第 8 章）制作密码资源中间件程序,实现与引擎操作函数（7.11 章节）接口的关联；

——当挂接硬件密码资源时，密码资源厂商还需制作底层硬件驱动程序，实现密码资源与中间件程序的连接，连接机制由厂商定义。

注：硬件驱动程序在核心态下执行。

6 数据类型定义

6.1 算法标识

对称加密算法标识：

表 1 对称加密算法标识

算法名称	描述	OID
SM4-ECB	SM4 算法，ECB 模式	1.2.156.10197.1.104.1
SM4-CBC	SM4 算法，CBC 模式	1.2.156.10197.1.104.2
SM4-CFB	SM4 算法，CFB 模式	1.2.156.10197.1.104.3
SM4-OFB	SM4 算法，OFB 模式	1.2.156.10197.1.104.4
SM4-CTR	SM4 算法，CTR 模式	1.2.156.10197.1.104.6
SM4-XTS	SM4 算法，XTS 模式	1.2.156.10197.1.104.9
SM4-GCM	SM4 算法，GCM 模式	1.2.156.10197.1.104.10
SM4-CCM	SM4 算法，CCM 模式	1.2.156.10197.1.104.11

非对称密钥算法标识：

表 2 非对称密钥算法标识

算法名称	描述	OID
SM2	SM2 算法	1.2.156.10197.1.301

杂凑算法标识：

表 3 杂凑算法标识

算法名称	描述	OID
SM3	SM3 算法	1.2.156.10197.1.401

6.2 安全通信协议标识

表 4 安全通信协议标识

协议标识	描述
TLCP1.1	传输层密码协议（TLCP）1.1版本
TLS1.3	传输层安全协议（TLS） 1.3版本

6.3 基本数据类型定义

基本数据类型均按照C语言进行定义。

表 5 基本数据类型

类型名称	说明
char	有符号字符类型
unsigned char	无符号字符类型
int	有符号整数
unsigned int	无符号整数
unsigned long	无符号长整数
size_t	无符号整数

6.4 数据结构定义

6.4.1 非对称密钥结构

原型：typedef struct ossm_pkey_st OSSM_PKEY;
描述：非对称密钥结构体，包含非对称密钥相关信息，具体内容由实现定义。

6.4.2 证书结构

原型：typedef struct ossm_x509_st OSSM_X509;
描述：证书结构体，包含证书相关信息，具体内容由实现定义。

6.4.3 证书撤销列表结构

原型：typedef struct ossm_crl_st OSSM_CRL;
描述：证书撤销列表结构体，包含证书撤销相关信息，具体内容由实现定义。

6.4.4 对称加密算法结构

原型：typedef struct ossm_cipher_st OSSM_CIPHER;
描述：对称加密算法结构体，包含对称加密算法的元数据和元方法，具体内容由实现定义。

6.4.5 非对称算法结构

原型：typedef struct ossm_pkey_meth_st OSSM_PKEY_METHOD;
描述：非对称密钥算法结构体，包含非对称算法的元数据和元方法，具体内容由实现定义。

6.4.6 杂凑算法结构

原型：typedef struct ossm_md_st OSSM_MD;
描述：杂凑算法结构体，包含杂凑算法的元数据和元方法，具体内容由实现定义。

6.4.7 随机数算法结构

原型：typedef struct ossm_rand_meth_st OSSM_RAND_METHOD;
描述：随机数算法结构体，包含随机数算法的元数据和元方法，具体内容由实现定义。

6.4.8 引擎

原型：typedef struct ossm_engine_st OSSM_ENGINE;
描述：引擎结构体，用于表示密码资源，具体内容由实现定义。

6.4.9 引擎函数指针

定义：typedef int (*OSSM_ENGINE_INT_FUNC_PTR) (OSSM_ENGINE *e);
描述：引擎函数指针，引擎初始化、结束时和销毁时的函数，需要符合以上函数原型。
参数：e引擎对象
返回值：0表示成功，非0表示错误码。

6.4.10 引擎对称加密算法函数指针

定义：typedef int (*OSSM_ENGINE_CIPHERS_PTR) (OSSM_ENGINE *e, const OSSM_CIPHER **cipher, const int **nids, int nid);
描述：引擎加密算法函数指针。当 cipher 为 NULL 时，通过 nids 返回支持的所有对称加密算法标识；当 cipher 不为 NULL 时，根据 nid，通过 cipher 返回对应加密算法的方法对象。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
cipher	const OSSM_CIPHER **	根据 nid 输出对称加密算法方法
nids	const int **	输出支持的对称算法标识
nid	int	选择的算法标识

返回值：当 cipher 为 NULL 时，返回支持的算法个数；否则，1 表示成功，0 表示失败。

6.4.11 引擎非对称算法函数指针

定义：typedef int (*OSSM_ENGINE_PKEY_METHS_PTR) (OSSM_ENGINE *e, OSSM_PKEY_METHOD ** pmeth, const int **nids, int nid);
描述：公钥算法函数指针。当 pmeth 为 NULL 时，通过 nids 返回支持的非对称算法标识列表；当 pmeth 不为 NULL 时，根据 nid，通过 pmeth 返回对应的非对称算法方法。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
pmeth	OSSM_PKEY_METHOD **	根据 nid 输出非对称算法方法
nids	const int **	输出支持的非对称算法标识
nid	int	选择的算法标识

返回值：当pmeth为NULL，返回支持的非对称算法个数；否则，1表示成功，0表示失败。

6.4.12 引擎杂凑算法函数指针

定义：typedef int (*OSSM_ENGINE_DIGESTS_PTR) (OSSM_ENGINE *e, const OSSM_MD **digest, const int **nids,int nid);
描述：杂凑算法函数指针。当 digest 为 NULL 时，通过设置 nids，返回所有支持的杂凑算法标识；当 digest 不为 NULL 时，根据 nid，设置 digest 返回对应的杂凑算法方法。
参数：

参数名	类型	描述

e	OSSM_ENGINE *	引擎对象
digest	const OSSM_MD **	根据 nid 输出杂凑算法方法
nids	const int **	输出支持的杂凑算法标识
nid	int	选择的算法标识

返回值：当 digest 为 NULL 时，返回支持的杂凑算法个数；否则，1 表示成功，0 表示失败。

6.4.13 引擎加密密钥函数指针

定义：typedef OSSM_PKEY *(*OSSM_ENGINE_LOAD_KEY_PTR)(OSSM_ENGINE *e, const char *key_id);

描述：公钥或私钥的加载函数。

参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
key_id	const char *	密钥标识

返回值：成功返回非对称密钥对象，失败返回 NULL。

7 OS 用户态商密 API

7.1 概述

OS用户态商密API为用户态OS商密操作提供功能调用接口（北向接口），包括基础支撑接口、对称加密接口、公钥算法接口、杂凑算法接口、HMAC接口、随机数接口、密钥管理接口、证书管理接口、安全通信接口、引擎操作接口和验证调试类接口。

7.2 基础支撑接口

7.2.1 概述

基础支撑接口包括转换接口、获取接口版本信息接口和获取错误码接口，其中转换接口的转换对象为算法名称字符串和内部算法标识。

调用本文中的接口时，输入或输出的算法参数涉及字符串和整数2种表示方式，有效的算法标识应对应大于0的整数，具体的数值由实现定义。

本文中接口返回值类型主要包括2种，整数和指针。当返回值为整数类型时，如无特殊说明，一般成功时返回1，失败时返回0，当返回值为指针类型时，返回空指针（即NULL）表示失败，返回非空指针表示成功；接口在实现时，如果遇到失败的情况应该设置具体的错误码，然后接口调用者通过调用OSSM_GetLastError()来获取最近的错误码。

表 6 基础支撑接口

接口名称	说明
OSSM_Str2Nid	算法名称字符串转换成算法标识
OSSM_Nid2Str	算法标识转换成算法名称字符串
OSSM_GetLastError	获取错误码

7.2.2 字符串转换成算法标识

原型：int OSSM_Str2Nid(const char *s);

描述：将字符串转换成内部的算法标识。

参数：

参数名	类型	描述
s	const char *	字符串

返回值：成功返回有效的算法标识（大于 0），失败返回 0。

7.2.3 算法标识转换成字符串

原型：const char *OSSM_Nid2Str(int n);
描述：将内部的算法标识转换成字符串。
参数：

参数名	类型	描述
n	int	算法标识

返回值：成功返回有效的算法字符串，失败返回 NULL。

7.2.4 获取接口版本信息

原型：const char *OSSM_GetVersion();
描述：获取接口版本信息。
参数：无
返回值：成功返回对应的版本号字符串，失败返回“not available”。

7.2.5 获取错误码

原型：unsigned long OSSM_GetLastError(void);
描述：获取最近的错误码。
参数：无。
返回值：返回错误码，如果没有错误则返回 0。

7.3 对称加密接口

7.3.1 概述

对称加密接口包括上下文接口、加密操作和解密操作等接口。其中更新接口（Update）既可以用于单包，也可以用于多包。

通过OSSM_CIPHER_NewCtx()创建的上下文可以用于加密或解密，上下文保存加解密过程中的状态值，具体的数据结构由实现定义。

表 7 对称加密接口

接口名称	说明
OSSM_CIPHER_NewCtx	创建对称加密上下文
OSSM_CIPHER_FreeCtx	销毁对称加密上下文
OSSM_CIPHER_EncryptInit	对称加密初始化
OSSM_CIPHER_EncryptUpdate	对称加密更新
OSSM_CIPHER_EncryptFinal	对称加密结束
OSSM_CIPHER_GetTag	获取加密的tag
OSSM_CIPHER_SetTag	设置加密的tag
OSSM_CIPHER_DecryptInit	对称解密初始化
OSSM_CIPHER_DecryptUpdate	对称解密更新
OSSM_CIPHER_DecryptFinal	对称解密结束

7.3.2 创建对称加密上下文

原型：void *OSSM_CIPHER_NewCtx(void);
描述：创建对称加密上下文。
参数：无。

返回值：成功返回对称加密上下文对象；失败返回 NULL。

7.3.3 销毁对称加密上下文

原型：void OSSM_CIPHER_FreeCtx(void *ctx);
描述：销毁对称加密上下文。
参数：

参数名	类型	描述
ctx	void *	对称加密上下文

返回值：无。

7.3.4 对称加密初始化

原型：int OSSM_CIPHER_EncryptInit(void *ctx, const char *alg, const unsigned char *key, const unsigned char *iv);
描述：对称加密初始化。
参数：

参数名	类型	描述
ctx	void *	对称加密上下文
alg	const char *	算法字符串，参考**表 1 对称加密算法标识**
key	const unsigned char *	密钥
iv	const unsigned char *	初始化向量

返回值：成功返回 1，失败返回 0。

7.3.5 对称加密更新

原型：int OSSM_CIPHER_EncryptUpdate(void *ctx, unsigned char *out, size_t *outlen, const unsigned char *in, size_t inlen);
描述：对称加密。注意，对于 AEAD 模式，例如 GCM、CCM 等，如果存在 AAD，则需要在第一次调用 OSSM_CIPHER_EncryptUpdate()时将 AAD 作为输入数据。
参数：

参数名	类型	描述
ctx	void *	对称加密上下文
out	unsigned char *	密文输出
outlen	size_t *	密文长度
in	const unsigned char *	明文输入
inlen	size_t	明文长度

返回值：成功返回 1，失败返回 0。

7.3.6 对称加密结束

原型：int OSSM_CIPHER_EncryptFinal(void *ctx, unsigned char *out, size_t *outlen);
描述：对称加密结束，输出剩余的密文数据。
参数：

参数名	类型	描述
ctx	void *	对称加密上下文
out	unsigned char *	密文输出
outlen	size_t *	密文长度

返回值：成功返回 1，失败返回 0。

7.3.7 获取加密的 tag

原型：int OSSM_CIPHER_GetTag(void *ctx, unsigned char *tag, size_t *taglen);
描述：AEAD 模式加密时，获取加密后的 tag。
参数：

参数名	类型	描述
ctx	void *	对称加密上下文
tag	unsigned char *	tag 输出
taglen	size_t *	tag 长度

返回值：成功返回 1，失败返回 0。

7.3.8 设置加密的 tag

原型：int OSSM_CIPHER_SetTag(void *ctx, const unsigned char *tag, size_t taglen);
描述：AEAD 模式加密时，设置加密 tag。注意：需要在 OSSM_CIPHER_DecryptFinal()之前调用 OSSM_CIPHER_SetTag()，否则会导致解密失败。
参数：

参数名	类型	描述
ctx	void *	对称加密上下文
tag	const unsigned char *	tag 输入
taglen	size_t	tag 长度

返回值：成功返回 1，失败返回 0。

7.3.9 对称解密初始化

原型：int OSSM_CIPHER_DecryptInit(void *ctx, const char *alg, const unsigned char *key, const unsigned char *iv);
描述：对称解密初始化。
参数：

参数名	类型	描述
ctx	void *	对称解密上下文
alg	const char *	算法字符串，**表 1 对称加密算法标识**
key	const unsigned char *	密钥
iv	const unsigned char *	初始化向量

返回值：成功返回 1，失败返回 0。

7.3.10 对称解密更新

原型：int OSSM_CIPHER_DecryptUpdate(void *ctx, unsigned char *out, size_t *outlen, const unsigned char *in, size_t inlen);
描述：对称解密。注意，对于 AEAD 模式，例如 GCM、CCM 等，如果存在 AAD，需要在第一次调用 OSSM_CIPHER_DecryptUpdate()时将 AAD 作为输入数据。
参数：

参数名	类型	描述
ctx	void *	对称解密上下文
out	unsigned char *	明文输出
outlen	size_t *	明文长度

in	const unsigned char *	密文输入
inlen	size_t	密文长度

返回值：成功返回 1，失败返回 0。

7.3.11 对称解密结束

原型：int OSSM_CIPHER_DecryptFinal(void *ctx, unsigned char *out, size_t *outlen);
描述：对称解密结束，输出剩余的明文数据。
参数：

参数名	类型	描述
ctx	void *	对称解密上下文
out	unsigned char *	密文输出
outlen	size_t *	密文长度

返回值：成功返回 1，失败返回 0。

7.4 公钥算法接口

7.4.1 概述

公钥算法接口包括加密、解密、签名、验签等接口。

表 8 公钥算法接口

接口名称	描述
OSSM_PKEY_Encrypt	非对称加密
OSSM_PKEY_Decrypt	非对称解密
OSSM_PKEY_Sign	杂凑值签名
OSSM_PKEY_Verify	杂凑值验签
OSSM_PKEY_DigestSignInit	数据签名初始化
OSSM_PKEY_DigestSignUpdate	数据签名更新
OSSM_PKEY_DigestSignFinal	数据签名结束
OSSM_PKEY_DigestVerifyInit	数据验签初始化
OSSM_PKEY_DigestVerifyUpdate	数据验签更新
OSSM_PKEY_DigestVerifyFinal	数据验签结束

7.4.2 非对称加密

原型：int OSSM_PKEY_Encrypt(OSSM_PKEY *pkey, unsigned char *out, size_t *outlen, const unsigned char *in, size_t inlen);
描述：非对称加密。
参数：

参数名	类型	描述
pkey	void *	非对称密钥对象
out	unsigned char *	密文输出
outlen	size_t *	密文长度
in	const unsigned char *	明文输入
inlen	size_t	明文长度

返回值：成功返回 1，失败返回 0。

7.4.3 非对称解密

原型：int OSSM_PKEY_Decrypt(OSSM_PKEY *pkey, unsigned char *out, size_t *outlen, const

unsigned char *in, size_t inlen);

描述：非对称解密。

参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象
out	unsigned char *	明文输出
outlen	size_t *	明文长度
in	const unsigned char *	密文输入
inlen	size_t	密文长度

返回值：成功返回 1，失败返回 0。

7.4.4 杂凑值签名

原型：int OSSM_PKEY_Sign(OSSM_PKEY *pkey, unsigned char *sig, size_t *siglen, const unsigned char *md, size_t mdlen);

描述：使用私钥对杂凑值进行数字签名，在执行成功之后，sig 参数所指向的缓冲区中含有生成的签名值。当使用 SM2 算法时，md 为数据经过 SM2 签名预处理的结果，SM2 算法预处理过程应符合 GB/T 35276。

参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象
sig	unsigned char *	签名输出
siglen	size_t *	签名长度
md	const unsigned char *	待签名数据的杂凑值
mdlen	size_t	杂凑值长度

返回值：成功返回 1，失败返回 0。

7.4.5 杂凑值验签

原型：int OSSM_PKEY_Verify(OSSM_PKEY *pkey, unsigned char *sig, size_t *siglen, const unsigned char *md, size_t mdlen);

描述：使用公钥验证签名值，md 为签名数据的杂凑值。当使用 SM2 算法时，md 为数据经过 SM2 签名预处理的结果，SM2 算法预处理过程应符合 GB/T 35276。

参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象
sig	unsigned char *	签名输出
siglen	size_t *	签名长度
md	const unsigned char *	待签名数据的杂凑值
mdlen	size_t	杂凑值长度

返回值：成功返回 1，失败返回 0。

7.4.6 数据签名初始化

原型：int OSSM_PKEY_DigestSignInit(void *ctx, OSSM_PKEY *pkey, const char *md_alg);

描述：数据签名初始化。

参数：

参数名	类型	描述
ctx	void *	杂凑上下文
pkey	OSSM_PKEY *	非对称密钥对象

md_alg	const char *	杂凑算法

返回值：成功返回 1，失败返回 0。

7.4.7 数据签名更新

原型： int OSSM_PKEY_DigestSignUpdate(void *ctx, const void *data, size_t len);
描述： 数据签名更新，可以多次调用 OSSM_DigestSignUpdate()输入数据。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
data	const void *	数据输入
len	size_t	数据长度

返回值：成功返回 1，失败返回 0。

7.4.8 数据签名结束

原型：int OSSM_PKEY_DigestSignFinal(void *ctx, unsigned char *sig, size_t *siglen);
描述：数据签名结束。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
sig	unsigned char *	签名输出
siglen	size_t *	签名长度

返回值：成功返回 1，失败返回 0。

7.4.9 数据验签初始化

原型：int OSSM_PKEY_DigestVerifyInit(void *ctx, OSSM_PKEY *pkey, const char *md_alg);
描述：数据验签初始化。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
pkey	OSSM_PKEY *	非对称密钥对象
md_alg	const char *	杂凑算法

返回值：成功返回 1，失败返回 0。

7.4.10 数据验签更新

原型：int OSSM_PKEY_DigestVerifyUpdate(void *ctx, const void *data, size_t len);
描述：数据验签更新，可以多次调用 OSSM_DigestVerifyUpdate()输入数据。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
data	const void *	数据输入
len	size_t	数据长度

返回值：成功返回 1，失败返回 0。

7.4.11 数据验签结束

原型：int OSSM_PKEY_DigestVerifyFinal(void *ctx, const unsigned char *sig, size_t siglen);

描述：数据验签结束。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
sig	const unsigned char *	签名输入
siglen	size_t	签名长度

返回值：成功返回 1，失败返回 0。

7.5 杂凑算法接口

7.5.1 概述

杂凑算法接口包括上下文接口和杂凑计算接口。其中，杂凑更新数据接口既可以用于单包，也可以用于多包。

表 9 杂凑算法接口

接口名称	描述
OSSM_MD_NewCtx	创建杂凑上下文
OSSM_MD_FreeCtx	销毁杂凑上下文
OSSM_MD_Init	杂凑初始化
OSSM_MD_Update	杂凑更新数据
OSSM_MD_Final	杂凑结束

7.5.2 创建杂凑上下文

原型：void *OSSM_MD_NewCtx(void);
描述：创建杂凑上下文。
参数：无。
返回值：成功返回杂凑上下文，失败返回 NULL。

7.5.3 销毁杂凑上下文

原型：void OSSM_MD_FreeCtx(void *ctx);
描述：销毁杂凑上下文。
参数：ctx 杂凑上下文

参数名	类型	描述
ctx	void *	杂凑上下文

返回值：无。

7.5.4 杂凑初始化

原型：int OSSM_MD_Init(void *ctx, const char *md_alg);
描述：数据签名初始化。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
md_alg	const char *	杂凑算法，参考**表 3 杂凑算法标识**

返回值：成功返回 1，失败返回 0。

7.5.5 杂凑更新数据

原型：int OSSM_MD_Update(void *ctx, const void *data, size_t len);
描述：杂凑更新数据，可以多次调用 OSSM_MD_Update()输入数据。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
data	const void *	数据输入
len	size_t	数据长度

返回值：成功返回 1，失败返回 0。

7.5.6 杂凑结束

原型：int OSSM_MD_Final(void *ctx, unsigned char *md, size_t *mdlen);
描述：杂凑结束，计算的杂凑结果存储在 md 缓冲区中，杂凑长度保存在 mdlen 中。
参数：

参数名	类型	描述
ctx	void *	杂凑上下文
md	unsigned char *	杂凑输出
mdlen	size_t *	杂凑长度

返回值：成功返回 1，失败返回 0。

7.6 HMAC 接口

7.6.1 概述

HMAC接口包括上下文接口和HMAC计算接口。其中，HMAC更新数据接口既可以用于单包，也可以用于多包。

表 10 HMAC 接口

接口名称	描述
OSSM_HMAC_NewCtx	创建HMAC上下文
OSSM_HMAC_CTX_free	销毁HMAC上下文
OSSM_HMAC_Init	HMAC初始化
OSSM_HMAC_Update	HMAC更新数据
OSSM_HMAC_Final	HMAC结束

7.6.2 创建 HMAC 上下文

原型：void *OSSM_HMAC_NewCtx(void);
描述：创建 HMAC 上下文。
参数：无。
返回值：成功返回 HMAC 上下文，失败返回 NULL。

7.6.3 销毁 HMAC 上下文

原型：void OSSM_HMAC_CTX_free(void *ctx);
描述：销毁 HMAC 上下文。
参数：

参数名	类型	描述
ctx	void *	HMAC 上下文

返回值：无。

7.6.4 HMAC 初始化

原型：int OSSM_HMAC_Init(void *ctx, const unsigned char *key, size_t keylen, const char *md);
描述：HMAC 初始化，设置密钥和杂凑算法。
参数：

参数名	类型	描述
ctx	void *	HMAC 上下文
key	const unsigned char *	密钥
keylen	size_t	密钥长度
md_alg	const char *	杂凑算法，**表 3 杂凑算法标识**

返回值：成功返回 1，失败返回 0。

7.6.5 HMAC 更新数据

原型：int OSSM_HMAC_Update(void *ctx, const unsigned char *data, size_t len);
描述：HMAC 更新数据，可以多次调用 OSSM_HMAC_update()输入数据。
参数：

参数名	类型	描述
ctx	void *	HMAC 上下文
data	const unsigned char *	数据输入
len	size_t	数据长度

返回值：成功返回 1，失败返回 0。

7.6.6 HMAC 结束

原型：int OSSM_HMAC_Final(void *ctx, unsigned char *out, size_t *outlen);
描述：HMAC 结束，计算的结果存储在 out 缓冲区中，实际输出长度保存在 outlen 中。
参数：

参数名	类型	描述
ctx	void *	HMAC 上下文
out	unsigned char *	HMAC 输出
outlen	size_t *	HMAC 长度

返回值：成功返回 1，失败返回 0。

7.7 随机数接口

7.7.1 概述

随机数接口包括生成随机数。

表 11 随机数接口

接口名称	描述
OSSM_RAND_Generate	生成随机数

7.7.2 生成随机数

原型：int OSSM_RAND_Generate(unsigned char *buf, size_t num);
描述：生成 num 字节随机数，存储在 buf 缓冲区中。
参数：

参数名	类型	描述
buf	unsigned char *	随机数输出
num	size_t	随机数长度

返回值：成功返回 1，失败返回 0。

7.8 密钥管理接口

7.8.1 概述

密钥管理接口包括密钥的生成、销毁、导出、导入、密钥协商等接口。

表 12 密钥管理接口

接口名称	描述
OSSM_PKEY_GenKey	生成非对称密钥
OSSM_PKEY_Free	销毁非对称密钥
OSSM_PKEY_WritePublicKey	导出非对称密钥的公钥
OSSM_PKEY_ReadPublicKey	导入非对称密钥的公钥
OSSM_SM2_Derive	SM2密钥协商

7.8.2 生成非对称密钥

原型：OSSM_PKEY *OSSM_PKEY_GenKey(const char *algorithm);
描述：根据算法 algorithm 生成非对称密钥对。
参数：

参数名	类型	描述
algorithm	const char *	非对称算法字符串，参考**表 2 非对称密钥算法标识**

返回值：成功返回非对称密钥对象，失败返回 NULL。

7.8.3 销毁非对称密钥

原型：void OSSM_PKEY_Free(OSSM_PKEY *pkey);
描述：安全销毁非对称密钥对象。
参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象

返回值：无。

7.8.4 以字符串形式导出非对称密钥的公钥

原型：int OSSM_PKEY_WritePublicKey(OSSM_PKEY *pkey, unsigned char *pub, size_t *len);
描述：从非对称密钥对象中导出公钥；导出格式为 PEM 格式，SM2 公钥编码参考 GM/T 0010。
参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象
pub	unsigned char *	公钥输出
len	size_t *	公钥长度

返回值：成功返回 1，失败返回 0。

7.8.5 从字符串导入非对称密钥的公钥

原型：OSSM_PKEY *OSSM_PKEY_ReadPublicKey(const unsigned char *pub, size_t len);
描述：加载非对称密钥的公钥，返回非对称密钥对象。公钥格式为 PEM 格式，SM2 公钥编码参考 GM/T 0010。
参数：

参数名	类型	描述
pub	const unsigned char *	公钥输入
len	size_t	公钥长度

返回值：成功返回非对称密钥对象，失败返回 NULL。

7.8.6 以文件形式导出非对称密钥的公钥

原型：int OSSM_PKEY_WritePublicKeyFile(OSSM_PKEY *pkey, char * file_path);
描述：从非对称密钥对象中导出公钥并写入文件；导出格式为 PEM 格式，SM2 公钥编码参考 GM/T 0010。
参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象
file_path	char *	文件路径

返回值：成功返回 1，失败返回 0。

7.8.7 从文件导入非对称密钥的公钥

原型：OSSM_PKEY *OSSM_PKEY_ReadPublicKeyFile(char * file_path);
描述：从文件中加载非对称密钥的公钥，返回非对称密钥对象。公钥格式为 PEM 格式，SM2 公钥编码参考 GM/T 0010。
参数：

参数名	类型	描述
file_path	char *	文件路径

返回值：成功返回非对称密钥对象，失败返回 NULL。

7.8.8 以字符串形式导出非对称密钥的私钥

原型：int OSSM_PKEY_WritePrivateKey(OSSM_PKEY *pkey, unsigned char *pri, size_t *len);
描述：从非对称密钥对象中导出私钥；导出格式为 PEM 格式，SM2 私钥编码参考 GM/T 0010。
参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象
pri	unsigned char *	私钥输出
len	size_t *	私钥长度

返回值：成功返回 1，失败返回 0。

7.8.9 从字符串导入非对称密钥的私钥

原型：OSSM_PKEY *OSSM_PKEY_ReadPrivateKey(const unsigned char *pri, size_t len);
描述：导入非对称密钥的私钥，返回非对称密钥对象；私钥格式为 PEM 格式，SM2 私钥编码参考 GM/T 0010。
参数：

参数名	类型	描述

pri	const unsigned char *	私钥输入
len	size_t	私钥长度

返回值：成功返回非对称密钥对象，失败返回 NULL。

7.8.10 以文件形式导出非对称密钥的私钥

原型：int OSSM_PKEY_WritePrivateKeyFile(OSSM_PKEY *pkey, char * file_path);

描述：从非对称密钥对象中导出私钥并写入文件；导出格式为 PEM 格式，SM2 私钥编码参考 GM/T 0010。

参数：

参数名	类型	描述
pkey	OSSM_PKEY *	非对称密钥对象
file_path	char *	文件路径

返回值：成功返回 1，失败返回 0。

7.8.11 从文件导入非对称密钥的私钥

原型：OSSM_PKEY *OSSM_PKEY_ReadPrivateKeyFile(char * file_path);

描述：从文件中导入非对称密钥的私钥，返回非对称密钥对象；私钥格式为 PEM 格式，SM2 私钥编码参考 GM/T 0010。

参数：

参数名	类型	描述
file_path	char *	文件路径

返回值：成功返回非对称密钥对象，失败返回 NULL。

7.8.12 SM2 密钥协商

原型：int OSSM_SM2_Derive(int initiator, const unsigned char *self_id, size_t self_id_len, const unsigned char *peer_id, size_t peer_id_len, OSSM_PKEY *self_key, OSSM_PKEY *self_tmp_key, OSSM_PKEY *peer_key, OSSM_PKEY *peer_tmp_key, size_t key_len, unsigned char *key);

描述：SM2 密钥推导，根据自己和对方的非对称密钥推导出临时的会话密钥，计算过程应符合 GB/T 35276。

参数：

参数名	类型	描述
initiator	int	1 表示发起者， 0 表示响应者
self_id	const unsigned char *	我方 ID 值
self_id_len	size_t	我方 ID 值长度
peer_id	const unsigned char *	对方 ID 值
peer_id_len	size_t	对方 ID 值长度
self_key	OSSM_PKEY *	我方的私钥
self_tmp_key	OSSM_PKEY *	我方的临时私钥
peer_key	OSSM_PKEY *	对方的公钥
peer_tmp_key	OSSM_PKEY *	对方的临时公钥
key_len	size_t	输出的会话密钥长度
key	unsigned char *	输出的临时的会话密钥

返回值：成功返回1，失败返回0。

7.9 证书管理接口

7.9.1 概述

证书管理接口包括加载和导出证书、CA证书和CRL操作等接口。

表 13 证书管理接口

接口名称	描述
OSSM_X509_ReadCertificate	加载证书
OSSM_X509_WriteCertificate	导出证书
OSSM_X509_AddCaCertificate	添加CA证书
OSSM_X509_GetCaCertificateCount	获取CA证书个数
OSSM_X509_GetCaCertificate	获取CA证书
OSSM_X509_DeleteCaCertificate	删除CA证书
OSSM_X509_ReadCRL	加载CRL
OSSM_X509_WriteCRL	导出CRL
OSSM_X509_AddCRL	添加CRL
OSSM_X509_VerifyCertificate	验证用户证书

7.9.2 加载证书

原型：OSSM_CERT *OSSM_X509_ReadCertificate(const unsigned char *pem, size_t len);
描述：载入 PEM 格式证书，返回证书对象。
参数：

参数名	类型	描述
pem	const unsigned char *	输入的 PEM 证书
len	size_t	证书长度

返回值：成功返回证书对象，失败返回 NULL。

7.9.3 导出证书

原型：int OSSM_X509_WriteCertificate(OSSM_CERT *cert, unsigned char *pem, size_t *len);
描述：根据证书对象，导出 PEM 证书。
参数：

参数名	类型	描述
cert	OSSM_CERT *	证书对象
pem	unsigned char *	PEM 证书输出
len	size_t *	证书长度

返回值：成功返回 1，失败返回 0。

7.9.4 添加 CA 证书

原型：int OSSM_X509_AddCaCertificate(OSSM_CERT *cert, unsigned int *index);
描述：将证书添加到可信 CA 证书列表中，并将对应的索引存储到 index 中。
参数：

参数名	类型	描述
cert	OSSM_CERT *	证书对象
index	unsigned int *	证书索引

返回值：成功返回 1，失败返回 0。

7.9.5 获取 CA 证书个数

原型：int OSSM_X509_GetCaCertificateCount(unsigned int *cnt);
描述：根据 CA 证书的索引获取证书对象。
参数：

参数名	类型	描述
cnt	unsigned int *	证书个数

返回值：成功返回 1，失败返回 0。

7.9.6 获取 CA 证书

原型：OSSM_CERT *OSSM_X509_GetCaCertificate(unsigned int index);
描述：根据 CA 证书的索引获取证书对象。
参数：

参数名	类型	描述
index	unsigned int	证书索引

返回值：成功返回证书对象，失败返回 NULL。

7.9.7 删除 CA 证书

原型：int OSSM_X509_DeleteCaCertificate(unsigned int index);
描述：删除索引位置的 CA 证书。
参数：

参数名	类型	描述
index	unsigned int	证书索引

返回值：成功返回 1，失败返回 0。

7.9.8 加载 CRL

原型：OSSM_CRL *OSSM_X509_ReadCRL(const unsigned char *crl, size_t len);
描述：载入 PEM 格式证书，返回证书对象。
参数：

参数名	类型	描述
crl	const unsigned char *	输入的 CRL，PEM 格式
len	size_t	CRL 长度

返回值：成功返回 CRL 对象，失败返回 NULL。

7.9.9 导出 CRL

原型：int OSSM_X509_WriteCRL(OSSM_CRL *crl，unsigned char *pem, size_t *len);
描述：将 crl 对象导出成 PEM 格式。
参数：

参数名	类型	描述
crl	OSSM_CRL *	CRL 对象
pem	unsigned char *	PEM 格式输出
len	size_t *	输出的 PEM 长度

返回值：成功返回 1，失败返回 0。

7.9.10 添加 CRL

原型：int OSSM_X509_AddCRL(OSSM_CRL *crl);
描述：添加 CRL。
参数：

参数名	类型	描述
crl	OSSM_CRL *	CRL 对象

返回值：成功返回 1，失败返回 0。

7.9.11 验证用户证书

原型：int OSSM_X509_VerifyCertificate(OSSM_CERT *cert);
描述：验证用户证书的有效性，包括验证有效期、证书信任列表、吊销状态。
参数：

参数名	类型	描述
cert	OSSM_CERT *	证书对象

返回值：成功返回 1，失败返回 0。

7.10 安全通信接口

7.10.1 概述

安全通信接口包括上下文接口和收发数据等接口。

表 14 安全通信接口

接口名称	描述
OSSM_GMTP_NewCtx	创建安全通信上下文
OSSM_GMTP_FreeCtx	销毁安全通信上下文
OSSM_GMTP_CTX_SetCertificate	设置证书
OSSM_GMTP_CTX_SetPrivateKey	设置私钥
OSSM_GMTP_CTX_SetEncCertificate	设置签名证书
OSSM_GMTP_CTX_SetSignPrivateKey	设置签名私钥
OSSM_GMTP_CTX_SetSignCertificate	设置加密证书
OSSM_GMTP_CTX_SetEncPrivateKey	设置加密私钥
OSSM_GMTP_CTX_AddExtraChainCert	设置证书链CA证书
OSSM_GMTP_CTX_SetCipherSuites	设置加密套件
OSSM_GMTP_New	创建安全通信对象
OSSM_GMTP_Free	销毁安全通信对象
OSSM_GMTP_Connect	客户端发起连接
OSSM_GMTP_Accept	服务端等待连接
OSSM_GMTP_Read	安全通信接收数据
OSSM_GMTP_Write	安全通信发送数据
OSSM_GMTP_Shutdown	关闭安全通信连接

7.10.2 创建安全通信上下文

原型：void *OSSM_GMTP_NewCtx(const char *protocol);
描述：创建安全通信上下文，需要传入协议标识，参见**表 4 安全通信协议标识**。
参数：

参数名	类型	描述
protocol	const char *	协议字符串

返回值：成功，返回安全通信上下文；失败，返回 NULL。

7.10.3 销毁安全通信上下文

原型：void OSSM_GMTP_FreeCtx(void *ctx);
描述：销毁安全通信上下文。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文

返回值：无。

7.10.4 设置证书

原型：int OSSM_GMTP_CTX_SetCertificate(void *ctx, OSSM_X509 *cert);
描述：将签名证书保存到安全通信上下文内。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
cert	OSSM_X509 *	证书对象

返回值：成功返回 1， 失败返回 0。

7.10.5 设置私钥

原型：int OSSM_GMTP_CTX_SetPrivateKey(void *ctx, OSSM_PKEY *pkey);
描述：将私钥保存到安全通信上下文里，该私钥可用于数字签名或者非对称加密算法。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
pkey	OSSM_PKEY *	非对称密钥对象

返回值：成功返回 1， 失败返回 0。

7.10.6 设置签名证书

原型：int OSSM_GMTP_CTX_SetEncCertificate(void *ctx, OSSM_X509 *cert);
描述：将签名证书保存到安全通信上下文内。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
cert	OSSM_X509 *	证书对象

返回值：成功返回 1， 失败返回 0。

7.10.7 设置签名私钥

原型：int OSSM_GMTP_CTX_SetSignPrivateKey(void *ctx, OSSM_PKEY *pkey);
描述：将签名私钥保存到安全通信上下文里。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
pkey	OSSM_PKEY *	非对称密钥对象

返回值：成功返回 1， 失败返回 0。

7.10.8 设置加密证书

原型：int OSSM_GMTP_CTX_SetSignCertificate(void *ctx, OSSM_X509 *cert);
描述：将加密证书保存到安全通信上下文内。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
cert	OSSM_X509 *	证书对象

返回值：成功返回 1， 失败返回 0。

7.10.9 设置加密私钥

原型：int OSSM_GMTP_CTX_SetEncPrivateKey(void *ctx, OSSM_PKEY *pkey);
描述：将加密私钥保存到安全通信上下文里。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
pkey	OSSM_PKEY *	非对称密钥对象

返回值：成功返回 1， 失败返回 0。

7.10.10 设置证书链 CA 证书

原型：int OSSM_GMTP_CTX_AddExtraChainCert(void *ctx, OSSM_X509 *cert);
描述：添加证书链 CA 证书，可以先后多次调用本函数添加多个 CA 证书。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
cert	OSSM_X509 *	证书对象

返回值：成功返回 1， 失败返回 0。

7.10.11 设置加密套件

原型：int OSSM_GMTP_CTX_SetCipherSuites (void *ctx, const char *str);
描述：将加密套件设置到安全通信上下文里。
参数：

参数名	类型	描述
ctx	void *	安全通信上下文
str	const char *	加密套件字符串，具体格式由实现定义

返回值：成功返回 1， 失败返回 0。

7.10.12 创建安全通信对象

原型：void *OSSM_GMTP_New(void *ctx);
描述：根据安全通信上下文创建安全通信对象。
参数：

参数名	类型	描述

ctx	void *	安全通信上下文

返回值：成功返回安全通信对象，失败返回 NULL。

7.10.13 销毁安全通信对象

原型：void OSSM_GMTP_Free(void *gmtp);
描述：销毁安全通信对象。
参数：

参数名	类型	描述
gmtp	void *	安全通信对象

返回值：无。

7.10.14 客户端发起连接

原型：int OSSM_GMTP_Connect(void *gmtp);
描述：客户端发起连接。
参数：

参数名	类型	描述
gmtp	void *	安全通信对象

返回值：成功返回 1，失败返回 0。

7.10.15 服务端等待连接

原型：int OSSM_GMTP_Accept(void *gmtp);
描述：服务端等待客户端连接。
参数：

参数名	类型	描述
gmtp	void *	安全通信对象

返回值：成功返回 1，失败返回 0。

7.10.16 安全通信接收数据

原型：void OSSM_GMTP_Read(void *gmtp, void *buf, size_t num, size_t *readbytes);
描述：从安全通信连接上读取密文，解密后写入 buf 中，buf 长度至少为 num，实际读取的长度保存在 readbytes 中。
参数：

参数名	类型	描述
gmtp	void *	安全通信对象
buf	void *	输出缓冲区
num	size_t	要读取的长度
readbytes	size_t *	实际读取的长度

返回值：成功返回 1，失败返回 0。

7.10.17 安全通信发送数据

原型：void OSSM_GMTP_Write(void *gmtp, const void *buf, size_t num, size_t *written);
描述：将数据从安全通信连接上发出去，明文数据在 buf 缓冲区中，要发送的数据长度为 num 字节，实际发送成功的数据长度保存在 written 中。
参数：

参数名	类型	描述
gmtp	void *	安全通信对象
buf	const void *	输入缓冲区
num	size_t	要写入的数据长度
written	size_t *	成功写入的数据长度

返回值：成功返回 1，失败返回 0。

7.10.18 关闭安全通信连接

原型：int OSSM_GMTP_Shutdown(void *gmtp);
描述：关闭安全通信连接。
参数：

参数名	类型	描述
gmtp	void *	安全通信对象

返回值：成功返回1，失败返回0。

7.11 引擎操作接口

7.11.1 概述

引擎操作接口包括引擎的加载和释放、初始化和结束、下发命令，以及注册相关接口。用户态商密应用程序可以通过调用引擎操作接口来注册多个密码资源，以供商密子系统选择使用。

表 15 引擎操作接口

接口名称	描述
OSSM_ENGINE_by_id	加载引擎
OSSM_ENGINE_free	释放引擎
OSSM_ENGINE_init	引擎初始化
OSSM_ENGINE_finish	引擎结束
OSSM_ENGINE_ctrl_cmd_string	引擎下发命令
OSSM_ENGINE_set_default_ciphers	注册引擎的对称加密算法
OSSM_ENGINE_set_default_pkey_meths	注册引擎的公钥算法
OSSM_ENGINE_set_default_digests	注册引擎的杂凑算法
OSSM_ENGINE_set_default_rand	注册引擎的随机数算法

7.11.2 加载引擎

原型：OSSM_ENGINE *OSSM_ENGINE_by_id(const char *id);
描述：通过引擎ID加载引擎。
参数：

参数名	类型	描述
id	const char *	引擎标识字符串

返回值：成功返回引擎对象，失败返回NULL。

7.11.3 释放引擎

原型：int OSSM_ENGINE_free(OSSM_ENGINE *e);
描述：释放引擎。
参数：e 引擎对象

参数名	类型	描述

e	OSSM_ENGINE *	引擎对象

返回值：成功返回1，失败返回0。

7.11.4 引擎初始化

原型：int OSSM_ENGINE_init(OSSM_ENGINE *e);
描述：引擎初始化。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象

返回值：成功返回1，失败返回0。

7.11.5 引擎结束

原型：int OSSM_ENGINE_finish(OSSM_ENGINE *e);
描述：引擎结束。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象

返回值：成功返回1，失败返回0。

7.11.6 引擎下发命令

原型：int OSSM_ENGINE_ctrl_cmd_string(OSSM_ENGINE *e, const char *cmd_name, const char *arg, int cmd_optional);
描述：给引擎下发命令。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
cmd_name	const char *	命令名称
arg	const char *	命令参数
cmd_optional	int	命令是否可选

返回值：成功返回1，失败返回0。

7.11.7 注册引擎的对称加密算法

原型：int OSSM_ENGINE_set_default_ciphers(OSSM_ENGINE *e);
描述：将引擎e中支持的加密算法注册为默认的加密算法实现。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象

返回值：成功返回1，失败返回0。

7.11.8 注册引擎的公钥算法

原型：int OSSM_ENGINE_set_default_pkey_meths(OSSM_ENGINE *e);
描述：将引擎e中支持的公钥算法注册为默认的公钥算法实现。
参数：

参数名	类型	描述

e	OSSM_ENGINE *	引擎对象

返回值：成功返回1，失败返回0。

7.11.9 注册引擎的杂凑算法

原型：int OSSM_ENGINE_set_default_digests(OSSM_ENGINE *e);
描述：将引擎e中支持的杂凑算法注册为默认的加密算法实现。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象

返回值：成功返回1，失败返回0。

7.11.10 注册引擎的随机数算法

原型：int OSSM_ENGINE_set_default_rand(OSSM_ENGINE *e);
描述：将引擎e中支持的随机数算法注册为默认的随机数实现。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象

返回值：成功返回1，失败返回0。

8 OS用户态商密资源挂接接口

8.1 概述

OS商密子系统基于引擎（Engine）机制通过OS用户态商密资源挂接接口（南向接口）可以挂接新增的商密资源，扩展或替换密码子系统的密码算法。挂接的密码资源既可以是软件密码模块，也可以密码硬件，比如密码卡等。不同的密码资源通过引擎框架，即调用本文件定义的商密资源挂接接口来开发插件。

用户态应用软件可按照本文件7.11引擎操作接口加载和操作不同的密码资源。

8.2 引擎绑定和设置接口

8.2.1 概述

引擎绑定和设置接口主要包括引擎入口函数和引擎基本信息设置函数。

表 17 引擎绑定和设置接口

接口名称	描述
OSSM_ENGINE_bind	引擎绑定
OSSM_ENGINE_set_id	设置引擎ID
OSSM_ENGINE_set_name	设置引擎名字
OSSM_ENGINE_set_init_function	设置引擎初始化函数
OSSM_ENGINE_set_finish_function	设置引擎结束函数
OSSM_ENGINE_set_destroy_function	设置引擎释放函数

8.2.2 引擎绑定

原型：int OSSM_ENGINE_bind(OSSM_ENGINE *e, const char *id);
描述：该函数需要在引擎动态库中实现，在动态加载时由密码子系统自动调用，在此函数中需要对引擎 e 进行初始化。

参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
id	const char *	引擎 ID

返回值：成功返回 1，失败返回 0。

8.2.3 设置引擎 ID

原型：int OSSM_ENGINE_set_id(OSSM_ENGINE *e, const char *id)
描述：设置引擎的 ID，每个引擎的 ID 需要保持唯一。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
id	const char *	引擎 ID

返回值：成功返回 1，失败返回 0。

8.2.4 设置引擎名字

原型：int OSSM_ENGINE_set_name(OSSM_ENGINE *e, const char *name)
描述：设置引擎的名字。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
name	const char *	名字

返回值：成功返回 1，失败返回 0。

8.2.5 设置引擎初始化函数

原型：int OSSM_ENGINE_set_init_function(OSSM_ENGINE *e, OSSM_ENGINE_INT_FUNC_PTR init_f)
描述：设置引擎的初始化回调函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
init_f	OSSM_ENGINE_INT_FUNC_PTR	初始化函数

返回值：成功返回 1，失败返回 0。

8.2.6 设置引擎结束函数

原型：int OSSM_ENGINE_set_finish_function(OSSM_ENGINE *e, OSSM_ENGINE_INT_FUNC_PTR finish_f)
描述：设置引擎的结束时回调函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
finish_f	OSSM_ENGINE_INT_FUNC_PTR	结束时函数

返回值：成功返回 1，失败返回 0。

8.2.7 设置引擎释放函数

原型：int OSSM_ENGINE_set_destroy_function(OSSM_ENGINE *e,
OSSM_ENGINE_INT_FUNC_PTR destroy_f)
描述：设置引擎销毁时的回调函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
destroy_f	OSSM_ENGINE_INT_FUNC_PTR	销毁时函数

返回值：成功返回 1，失败返回 0。

8.3 引擎对称加密算法接口

8.3.1 概述

引擎对称加密算法接口包括设置引擎的对称算法函数，设置加密算法对象的元数据和元方法相关接口。

表 18 引擎对称加密算法接口

接口名称	描述
OSSM_ENGINE_set_ciphers	设置引擎的对称算法函数
OSSM_CIPHER_meth_new	创建加密算法对象
OSSM_CIPHER_meth_free	销毁加密算法对象
OSSM_CIPHER_meth_set_iv_length	设置加密算法对象的初始化向量长度
OSSM_CIPHER_meth_set_init	设置加密算法对象的初始化函数
OSSM_CIPHER_meth_set_do_cipher	设置加密算法对象的加密函数
OSSM_CIPHER_meth_set_cleanup	设置加密算法对象的清理函数
OSSM_CIPHER_meth_set_impl_ctx_size	设置加密算法对象的上下文大小
OSSM_CIPHER_meth_set_flags	设置加密算法对象的标志

8.3.2 设置引擎的加密算法函数

原型：int OSSM_ENGINE_set_ciphers(OSSM_ENGINE *e, OSSM_ENGINE_CIPHERS_PTR f);
描述：设置引擎的加密算法函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
f	OSSM_ENGINE_CIPHERS_PTR	加密算法函数指针

返回值：成功返回 1，失败返回 0。

8.3.3 创建加密算法对象

原型：OSSM_CIPHER *OSSM_CIPHER_meth_new(const char *cipher_id, int block_size, int key_len);
描述：创建加密算法对象。
参数：

参数名	类型	描述
cipher_id	const char *	加密算法标识
block_size	int	加密算法分组长度
key_len	int	加密算法密钥长度

返回值：成功返回加密算法对象，失败返回 NULL。

8.3.4 销毁加密算法对象

原型：void OSSM_CIPHER_meth_free(OSSM_CIPHER *cipher);
描述：销毁加密算法对象。
参数：

参数名	类型	描述
cipher	OSSM_CIPHER *	加密算法对象

返回值：无。

8.3.5 设置加密算法对象初始化向量长度

原型：int OSSM_CIPHER_meth_set_iv_length(OSSM_CIPHER *cipher, int iv_len);
描述：加密算法对象设置初始化向量长度。
参数：

参数名	类型	描述
cipher	OSSM_CIPHER *	加密算法对象
iv_len	int	IV 长度

返回值：成功返回 1，失败返回 0。

8.3.6 设置加密算法对象的初始化函数

原型：int OSSM_CIPHER_meth_set_init(OSSM_CIPHER *cipher,
int (*init)(void *ctx,
const unsigned char *key,
const unsigned char *iv,
int enc));
描述：设置加密算法对象的初始化函数。
参数：

参数名	类型	描述
cipher	OSSM_CIPHER *	加密算法对象
init	int (*)(void *ctx, const unsigned char *key, const unsigned char *iv, int enc)	初始化函数，传入加密算法上下文 ctx、密钥 key、初始化向量 iv，enc 为 1 时表示加密，0 表示解密，返回 1 表示成功，0 表示 失败。

返回值：成功返回 1，失败返回 0。

8.3.7 设置加密算法对象的加密函数

原型：int OSSM_CIPHER_meth_set_do_cipher(OSSM_CIPHER *cipher,
int (*do_cipher)(void *ctx,
unsigned char *out,
const unsigned char *in,
size_t inl));
描述：设置加密算法对象的加密函数。
参数：

参数名	类型	描述
cipher	OSSM_CIPHER *	加密算法对象
do_cipher	int (*)(void *ctx, unsigned char *out,	加密函数，输入加密上下

	const unsigned char *in, size_t inl)	文，输出缓冲区，输入缓冲区，输入长度，成功返回 1，失败返回 0。

返回值：成功返回 1，失败返回 0。

8.3.8 设置加密算法对象的清理函数

原型：int OSSM_CIPHER_meth_set_cleanup(OSSM_CIPHER *cipher,
int (*cleanup)(void *));
描述：设置加密算法对象的加密函数。
参数：

参数名	类型	描述
cipher	OSSM_CIPHER *	加密算法对象
cleanup	int (*)(void *)	清理函数，输入加密算法上下文，成功返回 1，失败返回 0。

返回值：成功返回 1，失败返回 0。

8.3.9 设置加密算法对象的上下文大小

原型：int OSSM_CIPHER_meth_set_impl_ctx_size(OSSM_CIPHER *cipher, int ctx_size);
描述：设置加密算法对象的上下文大小。
参数：

参数名	类型	描述
cipher	OSSM_CIPHER *	加密算法对象
ctx_size	int	上下文大小

返回值：成功返回 1，失败返回 0。

8.3.10 设置加密算法对象的标志

原型：int OSSM_CIPHER_meth_set_flags(OSSM_CIPHER *cipher, unsigned long flags);
描述：设置加密算法对象的标志，flags 的具体含义由实现定义。
参数：

参数名	类型	描述
cipher	OSSM_CIPHER *	加密算法对象
flags	unsigned long	标志

返回值：成功返回 1，失败返回 0。

8.4 引擎公钥算法接口

8.4.1 概述

引擎公钥算法接口包括设置引擎的公钥算法函数，以及设置公钥算法对象的元方法的函数。

表 19 引擎公钥算法接口

接口名称	描述
OSSM_ENGINE_set_pkey_meths	设置引擎的公钥算法函数
OSSM_PKEY_meth_new	创建公钥算法对象
OSSM_PKEY_meth_free	销毁公钥算法对象
OSSM_PKEY_meth_set_init	设置公钥算法对象的初始化函数

OSSM_PKEY_meth_set_cleanup	设置公钥算法对象的清理函数
OSSM_PKEY_meth_set_paramgen	设置公钥算法对象的参数生成函数
OSSM_PKEY_meth_set_keygen	设置公钥算法对象的密钥生成函数
OSSM_PKEY_meth_set_sign	设置公钥算法对象的签名函数
OSSM_PKEY_meth_set_verify	设置公钥算法对象的验签函数
OSSM_PKEY_meth_set_encrypt	设置公钥算法对象的加密函数
OSSM_PKEY_meth_set_decrypt	设置公钥算法对象的解密函数
OSSM_PKEY_meth_set_derive	设置公钥算法对象的密钥推导函数
OSSM_PKEY_meth_set_digestsign	设置公钥算法对象的数据签名函数
OSSM_PKEY_meth_set_digestverify	设置公钥算法对象的数据验签函数

8.4.2 设置引擎的公钥算法函数

原型：int OSSM_ENGINE_set_pkey_meths(OSSM_ENGINE *e,
OSSM_ENGINE_PKEY_METHS_PTR f);
描述：设置引擎的公钥算法函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
f	OSSM_ENGINE_PKEY_METHS_PTR	公钥算法函数指针

返回值：成功返回 1，失败返回 0。

8.4.3 创建公钥算法对象

原型：OSSM_PKEY_METHOD *OSSM_PKEY_meth_new(int id);
描述：创建公钥算法对象。
参数：id 公钥算法标识

参数名	类型	描述
id	int	公钥算法标识

返回值：成功时返回公钥算法对象，失败时返回 NULL。

8.4.4 销毁公钥算法对象

原型：void OSSM_PKEY_meth_free(OSSM_PKEY_METHOD *pmeth);
描述：销毁公钥算法对象。
参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象

返回值：无。

8.4.5 设置公钥算法对象的初始化函数

原型：void OSSM_PKEY_meth_set_init(OSSM_PKEY_METHOD *pmeth,
int (*init) (void *ctx));
描述：设置公钥算法对象的初始化函数。
参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象

init	int (*) (void *ctx)	初始化函数，参数为公钥算法上下文，成功返回 1，失败返回 0。

返回值：无。

8.4.6 设置公钥算法对象的清理函数

原型：void OSSM_PKEY_meth_set_cleanup(OSSM_PKEY_METHOD *pmeth,
void (*cleanup) (void *ctx));
描述：设置公钥算法对象的清理函数。
参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
cleanup	int (*) (void *ctx)	清理函数，参数为公钥算法上下文。

返回值：无。

8.4.7 设置公钥算法对象的参数生成函数

原型：void OSSM_PKEY_meth_set_paramgen(OSSM_PKEY_METHOD *pmeth,
int (*paramgen_init) (void *ctx),
int (*paramgen) (void *ctx,
OSSM_PKEY *pkey));
描述：设置公钥算法对象的参数生成函数。
参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
paramgen_init	int (*) (void *ctx)	参数生成初始化函数，参数为公钥算法上下文。
paramgen	int (*)(void *ctx, OSSM_PKEY *pkey)	参数生成函数，参数为公钥算法上下文和公钥对象。

返回值：无。

8.4.8 设置公钥算法对象的密钥生成函数

原型：void OSSM_PKEY_meth_set_keygen(OSSM_PKEY_METHOD *pmeth,
int (*keygen_init) (void *ctx),
int (*keygen) (void *ctx,
OSSM_PKEY *pkey));
描述：设置公钥算法对象的密钥生成函数。
参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
keygen _init	int (*) (void *ctx)	密钥生成初始化函数，参数为公钥算法上下文。
keygen	int (*)(void *ctx, OSSM_PKEY *pkey)	密钥生成函数，参数为公钥算法上下文和公钥对象。

返回值：无。

8.4.9 设置公钥算法对象的签名函数

原型：void OSSM_PKEY_meth_set_sign(OSSM_PKEY_METHOD *pmeth,
int (*sign_init) (void *ctx),
int (*sign) (void *ctx,
unsigned char *sig, size_t *siglen,
const unsigned char *tbs,
size_t tbslen));

描述：设置公钥算法对象的签名函数。

参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
sign _init	int (*) (void *ctx)	签名初始化函数，参数为公钥算法上下文。
sign	int (*) (void *ctx, unsigned char *sig, size_t *siglen, const unsigned char *tbs, size_t tbslen)	签名函数，参数为公钥算法上下文，签名输出，签名长度，待签名杂凑，待签名杂凑的长度，成功返回 1，失败返回 0。

返回值：无。

8.4.10 设置公钥算法对象的验签函数

原型：void OSSM_PKEY_meth_set_verify(OSSM_PKEY_METHOD *pmeth,
int (*verify_init) (void *ctx),
int (*verify) (void *ctx,
const unsigned char *sig,
size_t siglen,
const unsigned char *tbs,
size_t tbslen));

描述：设置公钥算法对象的验签函数。

参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
verify_init	int (*) (void *ctx)	验签初始化函数，参数为公钥算法上下文。
verify	int (*) (void *ctx, const unsigned char *sig, size_t siglen, const unsigned char *tbs, size_t tbslen)	验签函数，参数为公钥算法上下文，签名，签名长度，待签名杂凑，待签名杂凑的长度，成功返回 1，失败返回 0。

返回值：无。

8.4.11 设置公钥算法对象的加密函数

原型：void OSSM_PKEY_meth_set_encrypt(OSSM_PKEY_METHOD *pmeth,
int (*encrypt_init) (void *ctx),
int (*encryptfn) (void *ctx,
unsigned char *out,
size_t *outlen,
const unsigned char *in,

size_t inlen));

描述：设置公钥算法对象的加密函数。

参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
encrypt_init	int (*) (void *ctx)	加密初始化函数，参数为公钥算法上下文。
encryptfn	int (*) (void *ctx, unsigned char *out, size_t *outlen, const unsigned char *in, size_t inlen)	加密函数，参数为公钥算法上下文，密文输出，密文长度，明文输入，明文长度，成功返回 1，失败返回 0

返回值：无。

8.4.12 设置公钥算法对象的解密函数

原型：void OSSM_PKEY_meth_set_decrypt(OSSM_PKEY_METHOD *pmeth,
int (*decrypt_init) (void *ctx),
int (*decryptfn) (void *ctx,
unsigned char *out,
size_t *outlen,
const unsigned char *in,
size_t inlen));

描述：设置公钥算法对象的解密函数。

参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
decrypt_init	int (*) (void *ctx)	解密初始化函数，参数为公钥算法上下文。
decryptfn	int (*) (void *ctx, unsigned char *out, size_t *outlen, const unsigned char *in, size_t inlen)	解密函数，参数为公钥算法上下文，明文输出，明文长度，密文输入，密文长度，成功返回 1，失败返回 0。

返回值：无。

8.4.13 设置公钥算法对象的密钥推导函数

原型：void OSSM_PKEY_meth_set_derive(OSSM_PKEY_METHOD *pmeth,
int (*derive_init) (void *ctx),
int (*derive) (void *ctx,
unsigned char *key,
size_t *keylen));

描述：设置公钥算法对象的密钥推导函数，密钥推导函数的作用为扩展公钥算法中用于加密数据的密钥至给定长度。

参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
derive_init	int (*) (void *ctx)	密钥推导初始化函数，参数为公钥算法上下文。
derive	int (*) (void *ctx, unsigned	密钥推导函数，参数为公钥

	char *key, size_t *keylen)	算法上下文，密钥输出，密钥长度，成功返回 1，失败返回 0。

返回值：无。

8.4.14 设置公钥算法对象的数据签名函数

原型：void OSSM_PKEY_meth_set_digestsign(OSSM_PKEY_METHOD *pmeth,
int (*digestsign) (void *ctx, unsigned char *sig, size_t *siglen,
const unsigned char *tbs, size_t tbslen));
描述：设置公钥算法对象的数据签名函数。
参数：

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
digestsign	int (*) (void *ctx, unsigned char *sig, size_t *siglen, const unsigned char *tbs, size_t tbslen)	签名函数，参数为杂凑上下文，签名输出，签名长度，待签名数据，待签名数据的长度，成功返回 1，失败返回 0。

返回值：无。

8.4.15 设置公钥算法对象的数据验签函数

原型：void OSSM_PKEY_meth_set_digestverify(OSSM_PKEY_METHOD *pmeth,
int (*digestverify) (void *ctx, const unsigned char *sig,
size_t siglen, const unsigned char *tbs,
size_t tbslen));
描述：设置公钥算法对象的数据验签函数。
参数：pmeth 公钥算法对象
digestverify 签名函数，参数为杂凑上下文，签名，签名长度，待签名数据，待签名数据的长度，成功返回 1，失败返回 0。

参数名	类型	描述
pmeth	OSSM_PKEY_METHOD *	公钥算法对象
digestverify	int (*) (void *ctx, const unsigned char *sig, size_t siglen, const unsigned char *tbs, size_t tbslen)	验签函数，参数为杂凑上下文，签名输入，签名长度，待签名数据，待签名数据的长度，成功返回 1，失败返回 0。

返回值：无。

8.5 引擎杂凑算法接口

8.5.1 概述

引擎杂凑算法包括设置引擎的杂凑算法函数和设置杂凑算法对象的元数据和元方法，包括分组大小、结果长度、初始化函数、更新函数、结束函数等。

表 20 引擎杂凑算法接口

接口名称	描述
OSSM_ENGINE_set_digests	设置引擎的杂凑算法函数

OSSM_MD_meth_new	创建杂凑算法对象
OSSM_MD_meth_free	销毁杂凑算法对象
OSSM_MD_meth_set_input_blocksize	设置杂凑算法对象的输入分组大小
OSSM_MD_meth_set_result_size	设置杂凑算法对象的结果长度
OSSM_MD_meth_set_app_datasize	设置杂凑算法对象的应用数据长度
OSSM_MD_meth_set_flags	设置杂凑算法对象的标志
OSSM_MD_meth_set_init	设置杂凑算法对象的初始化函数
OSSM_MD_meth_set_update	设置杂凑算法对象的更新函数
OSSM_MD_meth_set_final	设置杂凑算法对象的结束函数
OSSM_MD_meth_set_cleanup	设置杂凑算法对象的清理函数

8.5.2 设置引擎的杂凑算法函数

原型：int OSSM_ENGINE_set_digests(OSSM_ENGINE *e, OSSM_ENGINE_DIGESTS_PTR f);
描述：设置引擎的杂凑算法函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
f	OSSM_ENGINE_DIGESTS_PTR	杂凑算法函数指针

返回值：成功返回 1，失败返回 0。

8.5.3 创建杂凑算法对象

原型：OSSM_MD *OSSM_MD_meth_new(int md_type, int pkey_type);
描述：创建杂凑算法对象。
参数：

参数名	类型	描述
md_type	int	杂凑算法 ID
pkey_type	int	公钥签名算法 ID

返回值：成功杂凑算法对象，失败返回 NULL。

8.5.4 销毁杂凑算法对象

原型：void OSSM_MD_meth_free(OSSM_MD *md);
描述：销毁杂凑算法对象。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象

返回值：无。

8.5.5 设置杂凑算法对象的输入分组大小

原型：int OSSM_MD_meth_set_input_blocksize(OSSM_MD *md, int blocksize);
描述：设置杂凑算法对象的输入分组大小。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
blocksize	int	分组大小

返回值：成功返回 1，失败返回 0。

8.5.6 设置杂凑算法对象的结果长度

原型：int OSSM_MD_meth_set_result_size(OSSM_MD *md, int resultsize);
描述：设置杂凑算法对象的结果长度。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
resultsize	int	杂凑结果的长度

返回值：成功返回 1，失败返回 0。

8.5.7 设置杂凑算法对象的应用数据长度

原型：int OSSM_MD_meth_set_app_datasize(OSSM_MD *md, int datasize);
描述：设置杂凑算法对象的应用数据长度。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
datasize	int	应用数据的长度

返回值：成功返回 1，失败返回 0。

8.5.8 设置杂凑算法对象的标志

原型：int OSSM_MD_meth_set_flags(OSSM_MD *md, unsigned long flags);
描述：设置杂凑算法对象的标志。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
flags	unsigned long	标志，具体的标志位由实线定义

返回值：成功返回 1，失败返回 0。

8.5.9 设置杂凑算法对象的初始化函数

原型：int OSSM_MD_meth_set_init(OSSM_MD *md, int (*init)(void *ctx));
描述：设置杂凑算法对象的初始化函数。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
init	int (*)(void *ctx)	初始化函数，输入杂凑算法上下文，成功返回 1，失败返回 0

返回值：成功返回 1，失败返回 0。

8.5.10 设置杂凑算法的更新函数

原型：int OSSM_MD_meth_set_update(OSSM_MD *md, int (*update)(void *ctx,
const void *data,

size_t count));
描述：设置杂凑算法对象的更新函数。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
update	int (*)(void *ctx, const void *data, size_t count)	更新函数，输入杂凑算法上下文，数据和数据长度，成功返回 1，失败返回 0

返回值：成功返回 1，失败返回 0。

8.5.11 设置杂凑算法结构的结束函数

原型：int OSSM_MD_meth_set_final(OSSM_MD *md, int (*final)(void *ctx,
unsigned char *md));
描述：设置杂凑算法对象的结束时函数。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
final	int (*)(void *ctx, unsigned char *md)	计算杂凑结束时调用的函数，输入杂凑算法上下文，杂凑输出缓冲区，成功返回 1，失败返回 0

返回值：成功返回 1，失败返回 0。

8.5.12 设置杂凑算法对象的清理函数

原型：int OSSM_MD_meth_set_cleanup(OSSM_MD *md, int (*cleanup)(void *ctx));
描述：设置杂凑算法对象的清理函数。
参数：

参数名	类型	描述
md	OSSM_MD *	杂凑算法对象
cleanup	int (*)(void *ctx)	清理函数，输入杂凑算法上下文，成功返回 1，失败返回 0

返回值：成功返回 1，失败返回 0。

8.6 引擎随机数接口

8.6.1 概述

引擎随机数接口包括随机数生成、增加熵、获取状态等函数。

表 21 引擎随机数接口

接口名称	描述
OSSM_ENGINE_set_RAND	设置引擎的随机数函数
OSSM_RAND_meth_new	创建随机数算法对象
OSSM_RAND_meth_free	销毁随机数算法对象
OSSM_RAND_meth_set_generate	设置随机数算法对象的随机数生成函数
OSSM_RAND_meth_set_add	设置随机数算法对象的增加熵函数
OSSM_RAND_meth_set_status	设置随机数算法对象的获取状态函数

8.6.2 设置引擎的随机数函数

原型：int OSSM_ENGINE_set_RAND(OSSM_ENGINE *e, const OSSM_RAND_METHOD *rand_meth);
描述：设置引擎的随机数函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
rand_meth	const OSSM_RAND_METHOD *	随机数算法对象

返回值：成功返回 1，失败返回 0。

8.6.3 创建随机数算法对象

原型：OSSM_RAND_METHOD *OSSM_RAND_meth_new(void);
描述：创建随机数算法对象。
参数：无。
返回值：成功时返回随机数算法对象，失败返回 NULL。

8.6.4 销毁随机数算法对象

原型：void OSSM_RAND_meth_free(OSSM_RAND_METHOD *meth);
描述：销毁随机数算法对象。
参数：

参数名	类型	描述
meth	OSSM_RAND_METHOD *	随机数算法对象

返回值：无。

8.6.5 设置随机数算法对象的随机数生成函数

原型：int OSSM_RAND_meth_set_generate(OSSM_RAND_METHOD *meth, int (*bytes) (unsigned char *buf, int num));
描述：设置随机数算法对象的随机数函数。
参数：

参数名	类型	描述
meth	OSSM_RAND_METHOD *	随机数算法对象
bytes	int (*) (unsigned char *buf, int num)	随机数生成函数，参数为输出缓冲区和期望获得的随机数长度（字节），返回 1 表示成功，返回 0 表示失败。

返回值：成功返回 1，失败返回 0。

8.6.6 设置随机数算法对象的熵源函数

原型：int OSSM_RAND_meth_set_entropy_source (OSSM_RAND_METH *meth, int (*entropy_source) (const void *buf, int num, double randomness));
描述：设置随机数算法对象的熵源函数。
参数：

参数名	类型	描述
meth	OSSM_RAND_METHOD *	随机数算法对象

entropy_source	int (*)(const void *buf, int num, double randomness)	熵源函数，参数为输入缓冲区、缓冲区长度和缓冲区数据具有的熵值，返回 1 表示成功，0 表示失败。

返回值：成功返回 1，失败返回 0。

8.6.7 设置随机数算法对象的获取状态函数

原型：int OSSM_RAND_meth_set_status(OSSM_RAND_METHOD *meth, int (*status) (void));
描述：设置随机数算法对象的增加熵函数。
参数：

参数名	类型	描述
meth	OSSM_RAND_METHOD *	随机数算法对象
status	int (*) (void)	获取随机数状态的函数，返回 1 表示随机数已经使用足够的数据进行播种，否则返回 0。

返回值：成功返回 1，失败返回 0。

8.7 引擎密钥管理接口

8.7.1 概述

引擎密钥管理接口包括设置引擎的公钥加载函数和私钥加载函数。

表 22 引擎密钥管理接口

接口名称	描述
OSSM_ENGINE_set_load_pubkey_function	设置引擎的公钥加载函数
OSSM_ENGINE_set_load_privkey_function	设置引擎的私钥加载函数

8.7.2 设置引擎的公钥加载函数

原型：int OSSM_ENGINE_set_load_pubkey_function(OSSM_ENGINE *e,
OSSM_ENGINE_LOAD_KEY_PTR loadpub_f);
描述：设置引擎的公钥加载函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象
loadpub_f	OSSM_ENGINE_LOAD_KEY_PTR	公钥加载函数

返回值：成功返回 1，失败返回 0。

8.7.3 设置引擎的私钥加载函数

原型：int OSSM_ENGINE_set_load_privkey_function(OSSM_ENGINE *e,
OSSM_ENGINE_LOAD_KEY_PTR loadpriv_f);
描述： 设置引擎的私钥加载函数。
参数：

参数名	类型	描述
e	OSSM_ENGINE *	引擎对象

loadpriv_f	OSSM_ENGINE_LOAD_KEY_PTR	私钥加载函数

返回值：成功返回 1，失败返回 0。

附录 A
（规范性）
错误码定义

表A.1给出OSSM_GetLastError()接口返回的错误码定义。

表 A.1 OSSM_GetLastError()接口返回的错误码定义

错误码名称	预定义值	说明
OSSM_ERR_INVAL	1	参数无效或非法
OSSM_ERR_NOMEM	2	内存分配失败
OSSM_ERR_AGAIN	3	需要重试，
OSSM_ERR_BUSY	4	资源被占用
OSSM_ERR_NOKEY	5	不存在该密钥
OSSM_ERR_BADMSG	6	数据完整性校验失败
OSSM_ERR_IO	7	底层硬件或传输错误
OSSM_ERR_NOSYS	8	请求的操作未实现
OSSM_ERR_PROTO	9	协议错误
OSSM_ERR_NODEV	10	设备不存在
OSSM_ERR_FAULT	11	指针非法
OSSM_ERR_NOENT	12	文件或目录不存在
OSSM_ERR_TIMEDOUT	13	超时
OSSM_ERR_ALREADY	14	操作进行中
OSSM_ERR_INTERNAL_ERROR	15	内部错误

附录 B
（资料性）
引擎开发和使用示例

以下是一个密码资源调用OS用户态商密资源挂接接口开发引擎的示例。示例代码实现了一个名为demo的引擎，该引擎提供SM4算法CBC模式的密码功能。

引擎开发示例代码

```
// 定义加解密算法的上下文结构体，用于存储计算过程中的相关数据
typedef struct {
   // something
} demo_ctx;

// 定义引擎的ID
static const char *engine_demo_id = "demo";
// 定义引擎的名字，用于唯一标识该引擎
static const char *engine_demo_name = "demo engine";
static const OSSM_CIPHER *sm4_cipher = NULL;

// 定义支持的对称加密算法，这里只支持SM4-CBC
static int afalg_cipher_nids[] = {
      NID_sm4_cbc,
};

// 引擎初始化函数，在引擎启动时调用，执行初始化等操作
static int demo_init(OSSM_ENGINE *e)
{
      // init
      return 1;
}

// 引擎结束函数，在引擎退出时调用，执行资源清理等操作
static int demo_finish(OSSM_ENGINE *e)
{
      // finish
      return 1;
}

// 引擎销毁函数，引擎销毁时调用，执行资源释放等操作
static int demo_destroy(ENGINE *e)
{
      // destory
      return 1;
}

// 对称加解密初始化函数，用于初始化加解密上下文，设置密钥和初始化向量
static int demo_cipher_init(void *ctx, const unsigned char *key,
                                    const unsigned char *iv, int enc)
{
      // do init
```

```
    return 1;
}

// 对称加解密函数，执行实际的加解密操作
static int demo_do_cipher(void *ctx, unsigned char *out,
                          const unsigned char *in, size_t inl)
{
    // do cipher
    return 1;
}

// 对称加解密清理函数，重置加解密上下文的时候调用，清理上下文
static int demo_cipher_cleanup(void *ctx)
{
     // do cleanup
     return 1;
}

// 创建并设置加密算法对象，包括设置IV长度、标志位、初始化函数、加解密函数、清理函数和
// 上下文大小，其中FLAGS_CIPH_CBC_MODE为实现自定义的标志位，表示支持CBC模式
static const OSSM_CIPHER *demo_sm4_cbc(int nid)
{
    if (sm4_cipher == NULL
        && ((sm4_cipher =
          OSSM_CIPHER_meth_new(nid, 16, 16)) == NULL
        || !OSSM_CIPHER_meth_set_iv_length(sm4_cipher, 16)
        || !OSSM_CIPHER_meth_set_flags(sm4_cipher, FLAGS_CIPH_CBC_MODE)
        || !OSSM_CIPHER_meth_set_init(sm4_cipher, demo_cipher_init)
        || !OSSM_CIPHER_meth_set_do_cipher(sm4_cipher, demo_do_cipher)
        || !OSSM_CIPHER_meth_set_cleanup(sm4_cipher, demo_cipher_cleanup)
        || !OSSM_CIPHER_meth_set_impl_ctx_size(sm4_cipher, sizeof(demo_ctx)))) {
        OSSM_CIPHER_meth_free(sm4_cipher);
        sm4_cipher = NULL;
    }
    return sm4_cipher;
}

// 定义引擎的对称算法函数，根据传入的nid返回相应的加密算法对象，如果cipher为NULL，
// 则返回支持的算法个数
static int demo_ciphers(OSSM_ENGINE *e, const OSSM_CIPHER **cipher,
                        const int **nids, int nid)
{
    int r = 1;

    if (cipher == NULL) {
        *nids = demo_cipher_nids;
        return (sizeof(demo_cipher_nids) / sizeof(demo_cipher_nids[0]));
    }

    switch (nid) {
    case NID_sm4_cbc:
        *cipher = demo_sm4_cbc();
        break;
    default:
```

```
            *cipher = NULL;
            r = 0;
        }
        return r;
}

// 引擎绑定函数，设置引擎ID、名字、初始化函数、结束函数、销毁函数和对称加密函数
static int OSSM_ENGINE_bind(OSSM_ENGINE *e, const char *id)
{
        if (id && (strcmp(id, engine_afalg_id) != 0))
            return 0;

        if (!OSSM_ENGINE_set_id(e, engine_demo_id)
            || !OSSM_ENGINE_set_name(e, engine_demo_name)
            || !OSSM_ENGINE_set_destroy_function(e, demo_destroy)
            || !OSSM_ENGINE_set_init_function(e, demo_init)
            || !OSSM_ENGINE_set_finish_function(e, demo_finish)
            || !OSSM_ENGINE_set_ciphers(e, demo_ciphers))
            return 0;

        return 1;
}
```

引擎使用示例代码

```
ENGINE *e;
int ret;

e = OSSM_ENGINE_by_id("demo");
if (e == NULL) {
        // 错误处理
}

ret = OSSM_ENGINE_init(e);
if (!ret) {
        // 错误处理
}

ret = OSSM_ENGINE_set_default_ciphers(e);
if (!ret) {
        // 错误处理
}

// 密码运算

OSSM_ENGINE_finish(e);
OSSM_ENGINE_free(e);
```

参 考 文 献

[1] GB/T 17964-2021 信息安全技术 分组密码算法的工作模式
[2] GM/T 0016-2023 智能密码钥匙密码应用接口规范
[3] GM/T 0018-2023 密码设备应用接口规范
[4] GM/T 0019-2023 通用密码服务接口规范
[5] RFC2104 HMAC：Keyed-Hashing for Message Authentication