

ICS 35.040

L 80

备案号:



# 中华人民共和国密码行业标准

GM/T 0055- 2018

---

## 电子文件密码应用技术规范

File cryptographic technical specification

(报批稿)

2018-05-02 发布

2018-05-02 实施

---

国家密码管理局 发布

## 目次

前言	III
引言	IV
1 范围	1
2 规范性引用文件	1
3 术语和定义	1
4 符号和缩略语	2
5 标签机制	2
5.1 总体描述	2
5.2 基于标签的安全电子文件系统架构	2
5.3 基于标签的安全机制	3
5.4 中间件对安全电子文件的处理过程	4
5.5 安全电子文件的存储方式	4
5.6 标签与文件的绑定机制	5
6 密码算法与密码服务	6
6.1 密码体制	6
6.2 密码算法	6
6.3 基础密码服务	7
6.4 个性密码服务	7
6.5 密钥对象	7
7 标签	7
7.1 标签结构	7
7.2 标签属性	11
8 基础密码操作	17
8.1 概述	17
8.2 标签的完整性与绑定关系的建立	17
8.3 标签的完整性与绑定关系的验证	18
8.4 文件签名	18
8.5 添加文件签名	18
8.6 验证文件签名	18
8.7 文件加密	18
8.8 文件解密	18
9 安全电子文件密码服务接口	19
9.1 常量定义	19
9.2 结构定义	20
9.3 接口函数组成和功能说明	27
9.4 接口函数定义	28
附录 A (资料性附录) 数字水印	49
A.1 概述	49
A.2 数字水印操作过程	49
A.3 嵌入	49
A.4 提取	49
附录 B (资料性附录) 指纹识别	50

GM/T 0055-2018

B.1 概述..... 50

B.2 指纹操作..... 50

## 前 言

本标准依据GB/T1.1-2009给出的规则起草。

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这些专利的责任。

本标准由密码行业标准化技术委员会提出并归口。

本标准中的附录A、附录B为资料性附录。

本标准起草单位：上海颐东网络信息有限公司、北京海泰方圆科技有限公司、深圳奥联科技有限公司、北京书生公司、北京宇讯佳通科技有限责任公司、深圳宝嘉电子设备有限公司、北京兴唐通信科技股份有限公司、上海格尔软件股份有限公司。

本标准主要起草人：刘平、谢永泉、姜海舟、宋明华、柳增寿、刘海生、蒋健、夏东山、刘宁胜、曹学武、杨茂江、孙志辉。

## 引 言

本标准中涉及的文件为广义的文件对象，与文件的具体格式无关。

在涉及文件处理及流转的应用系统中，存在着密码协议不统一、密码接口应用混乱、密码服务处理层次不清晰等问题，从而导致文件在不同应用系统之间进行交互时，出现兼容性和安全失控等问题。

为保证文件在处理过程中的规范性、兼容性、安全性和可控性，本标准提出了一种标签与文件绑定的安全控制机制，实现了文件全生命周期的机密性、完整性、有效性和抗抵赖性等安全控制。

本标准描述了基于标签的安全电子文件系统架构和应用系统如何通过中间件，调用相应的基础密码服务和个性密码服务，实现对文件的安全操作，由中间件为应用系统提供统一、规范的密码服务，有利于应用系统、中间件和密码服务开发单位专注于自身技术的开发，促进技术的产品化。

# 电子文件密码应用技术规范

## 1 范围

本标准适用于安全电子文件密码服务中间件的开发和检测，也可用于指导使用该中间件的应用系统的开发。

本标准不规定应用系统的安全性，也不规定具体的文件类型，适用于关注文件对象自身安全的相关标准规范和应用。

## 2 规范性引用文件

下列文件对于本标准的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件，凡是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。

GM/T 0009 SM2密码算法使用规范

GM/T 0015 基于SM2密码算法的数字证书格式规范

GM/T 0017 智能密码钥匙密码应用接口数据格式规范

GM/T 0019 通用密码服务接口规范

GM/T 0031 安全电子签章密码技术规范

PKCS#1 RSA Cryptography Standard

PKCS#5 Password-based Encryption Standard

## 3 术语和定义

下列术语适用于本文件。

### 3.1

**应用系统 application system**

以文件为处理对象，对文件进行创建、修改、授权、阅读、签批、盖章、打印、添加水印、流转、存档和销毁等操作的系统。

### 3.2

**文件/电子文件 file**

以数字方式表示的、对特定的使用对象有特定意义的实体。它可以是各类公文、票据、数字作品等。

### 3.3

**标签 label**

和文件绑定的一段数字实体，用于标识文件的属性和状态，定义文件的操作对象、操作行为及访问权限，记录文件处理环节中操作者的操作行为，确保文件在创建、修改、授权、阅读、签批、盖章、打印、添加水印、流转、存档和销毁等操作中始终处于安全可控的状态，为应用系统提供追溯和审计的依据。

### 3.4

## GM/T 0055-2018

### 操作者 operator

指对文件进行创建、修改、授权、阅读、签批、盖章、打印、添加水印、流转、存档和销毁等操作的行为主体，在本标准中操作者也是对标签进行属性设置和修改的行为主体。

## 3.5

### 安全电子文件 secured file

中间件的处理对象，由文件和与之绑定的标签组成。

## 3.6

### 安全电子文件密码服务中间件 secured file cryptographic service provider

简称为中间件，为应用系统提供与文件处理和流转有关的密码服务，包括共性密码服务如文件的加密、解密、签名和验证等，个性密码服务如电子印章、数字水印、指纹识别，文件控制服务如文件流转、权限管理等，安全管理服务如行为记录等。

## 4 符号和缩略语

下列缩略语适用于本文件。

API	应用编程接口 (Application Programming Interface)
ASN.1	抽象语法标准 (Abstract Syntax Notation One)
CBC	密文分组链接 (Cipher-block Chain)
CFB	密文反馈 (Cipher Feedback)
ECB	电子码本 (Electronic Codebook)
FAR	误判率 (False Accept Rate)
OFB	输出反馈 (Output Feedback)
PKCS	公钥密码标准 (The Public-Key Cryptography Standard)
PKI	公钥基础设施 (Public Key Infrastructure)

## 5 标签机制

### 5.1 总体描述

本标准使用密码技术保证文件的机密性、完整性、有效性和抗抵赖性，把应用系统所需的密码服务功能进行了抽象和集成。在本标准中设计了标签机制，用于解决在文件生命周期全过程中的安全问题。

### 5.2 基于标签的安全电子文件系统架构

基于标签的安全电子文件系统包括应用系统、中间件、基础密码服务和个性密码服务，如图 1 示。其中基础密码服务根据使用环境分为两类，一类是遵循 GM/T 0019 的密码服务，这类密码服务是为应用层提供；另一类是遵循 GM/T 0017 的密码服务，这类密码服务是为内核层提供。

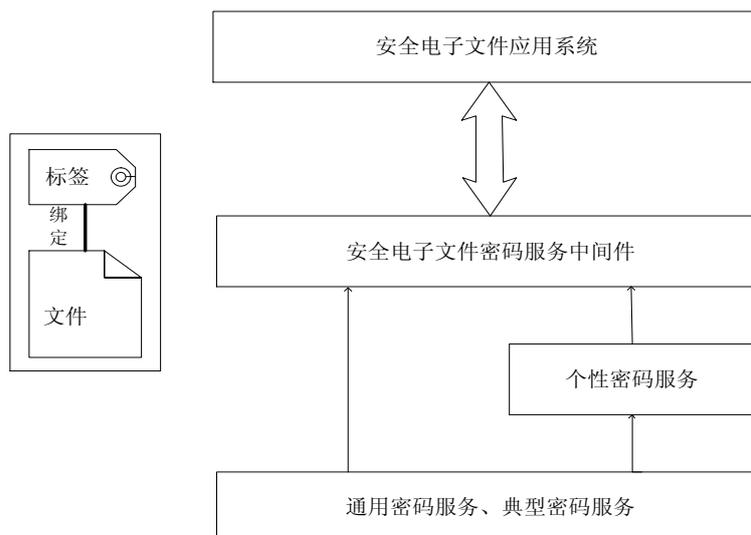


图1 安全电子文件的系统架构

应用系统在对文件进行创建、修改、授权、阅读、签批、盖章、打印、添加水印、流转、存档和销毁等处理过程中，需要提供的密码服务都通过调用中间件实现。

标签和文件是中间件的操作对象，标签与文件存在唯一绑定关系。其中，文件保存内容信息，标签提供对文件的安全控制。标签的结构定义和属性说明，具体详见第7章节。

通用密码服务和典型密码服务由密码基础设施提供，通过调用相关服务接口实现。具体详见6.3章节。

个性密码服务由电子印章、数字水印和指纹识别提供，通过调用相关服务接口实现。具体详见6.4章节。

电子文件操作者的密钥信息，由基础密码服务管理，同时基础密码服务为安全电子文件密码服务中间件提供密码操作的支持。

中间件对标签进行验证、解析和处理，根据标签属性的指示，调用相关的密码服务，完成对文件的加密、解密、签名、验证、印章处理、水印处理、指纹识别等密码操作。其中，加密、解密、签名和验证操作由基础密码服务完成；印章处理、水印处理、指纹识别等操作由个性密码服务实现。中间件提供的服务接口具体详见第9章节。

### 5.3 基于标签的安全机制

在基于标签的安全机制中，安全电子文件由文件和标签两部分组成。文件是指原始文件或对原始文件经过密码处理后的结果，标签是原始标签或对原始标签经过密码处理后的结果。在安全电子文件全生命周期中，文件和标签存在唯一绑定关系。标签只能由中间件进行处理。标签由标签头和标签体组成，标签体可加密。

标签是安全电子文件的操作依据，描述了安全电子文件的属性，主要包括签名属性、权限属性、印章属性、水印属性、指纹属性、标识属性、内容属性、扩展属性和日志属性等。

权限属性规定了安全电子文件的密码处理及操作权限。如加密、签名、添加印章、添加水印、指纹识别等密码处理方式，以及读、写、打印等操作权限。

标识属性描述了安全电子文件的编号、创建者及创建时间，当标签创建后标识属性则不可改变。

内容属性描述了安全电子文件的基本信息。如原始文件名、原始文件日期、文件类型及文件修改时间等基本信息。

扩展属性作为保留属性，用于应用系统定义自身的各种属性。

日志属性记录了操作者对安全电子文件所做的操作，如操作类型、操作者、操作时间等。

## 5.4 中间件对安全电子文件的处理过程

中间件按照请求/响应方式为应用系统提供服务，当应用系统对中间件发出操作请求之后，中间件的处理过程如下：

- a) 中间件从操作请求中获得操作者身份、标签和文件；
- b) 中间件按照绑定规则，对文件和标签进行绑定关系的验证，确认标签与文件的唯一绑定关系；
- c) 验证成功，中间件按照标签的操作权限属性，审核操作的合法性；
- d) 审核成功，中间件对标签的属性进行解析，按照标签的相关属性，调用相关的密码服务，完成对文件的具体密码操作；
- e) 操作完成，中间件在标签中修改相应的属性，添加操作日志，更新标签，并建立标签与文件新的绑定关系；
- f) 中间件向应用系统返回操作结果。

## 5.5 安全电子文件的存储方式

### 5.5.1 标签概述

标签分为内联式和外联式两种，采用 ASN.1 编码。按照标签长度分为固定标签和非固定标签两种。固定标签即在标签创建时设置一个长度，来使标签总是等于这个长度，不足的部分补零，不允许超过该长度；非固定标签即并不限制标签的长度，可以任意增长。

### 5.5.2 内联式

在内联式存储中，文件嵌入到标签，形成一个整体，在同一物理位置存放。如图 2 示。

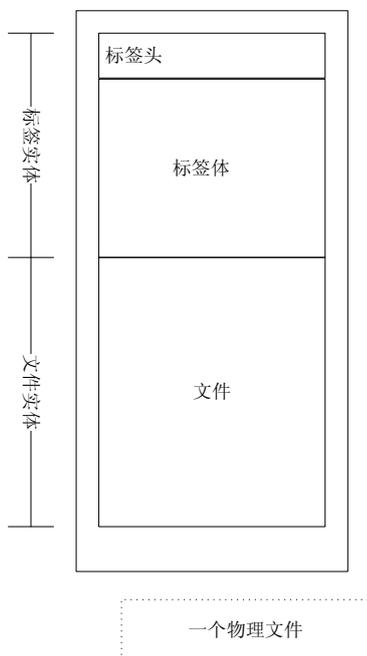


图 2 内联式的存储

### 5.5.3 外联式

在外联式存储中，标签和文件存放于两个独立文件，标签和文件实体之间的对应关系由应用系统管理。如图 3 示。

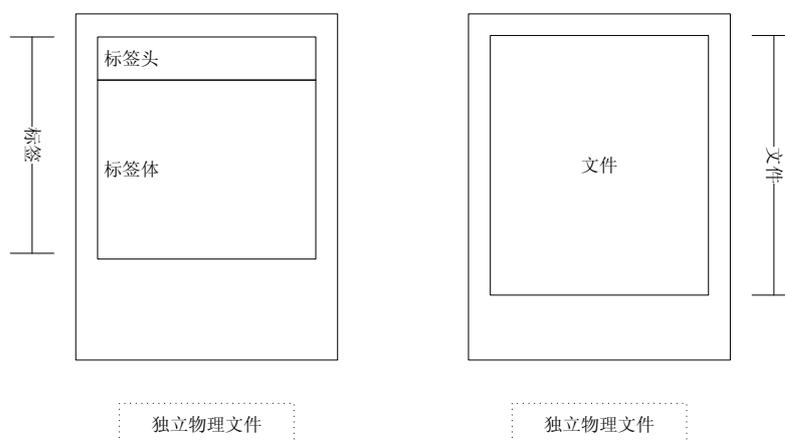


图3 外联式的存储

## 5.6 标签与文件的绑定机制

### 5.6.1 概述

标签与文件绑定关系的建立与验证由中间件进行。使用公钥密码算法和杂凑算法，其中公钥密码算法的密钥为两对，即加密密钥对和签名密钥对，这些密钥均为应用系统中操作者的密钥。

### 5.6.2 建立绑定关系

如图4所示，建立标签与文件的绑定关系流程如下：

- 对文件进行摘要计算，并签名；
- 将签名值、算法及对应的公钥证书形成签名属性填充在标签体中的签名属性集合中；
- 使用操作者签名私钥，对标签中除标签完整性签名以外的所有内容计算摘要并签名，将此签名值作为标签完整性签名置于标签头中。

#### 2 计算标签的摘要并签名

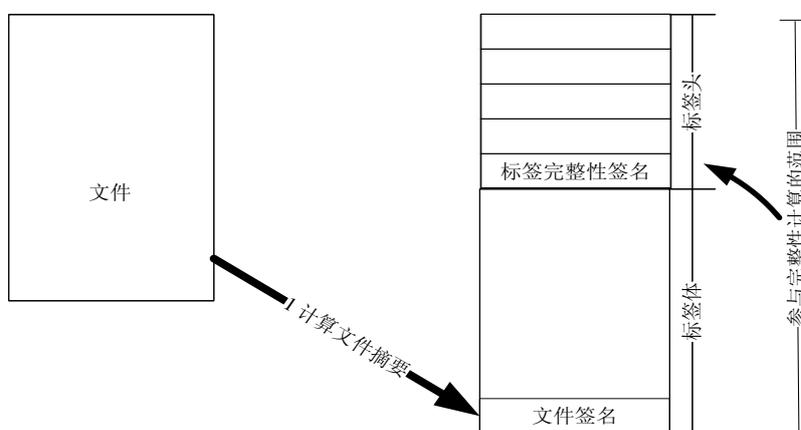


图 4 标签与文件绑定关系建立流程图

### 5.6.3 验证绑定关系

如图 5 示，验证标签与文件的绑定关系流程如下：

- 取标签头签名属性中的公钥证书，使用其对标签完整性签名进行验证。验证通过，则标签完整可信；
- 对文件进行摘要计算；
- 依次取标签体中签名属性集中的签名属性，通过其中的公钥算法对此摘要和签名值进行验证，如通过，则绑定关系验证通过。

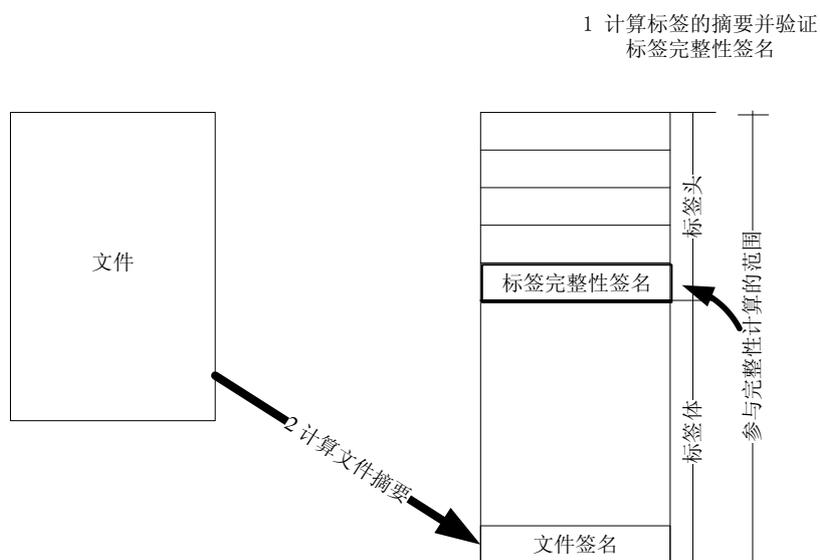


图 5 标签与文件绑定关系验证流程图

## 6 密码算法与密码服务

### 6.1 密码体制

中间件的密码机制基于 PKI 体制，使用数字证书机制对文件进行安全保护。数字证书使用双数字证书，遵循 GM/T 0015。标签的安全性保护可以采用数字证书机制或基于标识的密码机制，如 IBC。

### 6.2 密码算法

安全电子文件的密码操作需要使用对称算法、非对称算法和杂凑算法。对称算法可采用 ECB、CBC、OFB、CFB 四种模式，当使用 OFB 和 CFB 模式时，应用的反馈位数具体由应用设置，并保存在标签的数字信封中，算法要求如表 1 所示：

表 1 密码算法描述

算法类型	支持算法	用途描述
对称算法	国家密码管理主管部门许可的对称算法	用于加解密文件内容和标签内容
非对称算法	国家密码管理主管部门许可的非对称算	用于加密和解密对称算法密钥

	法	用于进行签名和签名验证
密码杂凑算法	国家密码管理主管部门许可的杂凑算法	用于完整性计算和验证

中间件对密码算法的调用是通过密码算法的标识来完成的。

### 6.3 基础密码服务

基础密码服务是由密码基础设施提供的密码服务，通过调用相关服务接口实现。

密码服务接口遵循 GM/T 0019 和 GM/T 0017。

密码服务提供基于数字证书的加密、解密、签名、签名验证等功能。其中，加密解密使用分组密码算法，密码工作模式为 ECB 和 CBC，初始化向量为零，加密数据的填充方式按照 PKCS#5 进行；签名及验证和数字信封操作使用公钥算法，当使用 RSA 算法时按照 PKCS#1 进行；使用 SM2 算法时按照 GM/T 0009。

### 6.4 个性密码服务

#### 6.4.1 概述

个性密码服务包括电子印章服务、数字水印服务和指纹识别服务。

#### 6.4.2 电子印章服务

电子印章服务包括对文件盖章、验章以及读取签章数据等。中间件通过调用电子印章服务实现印章的加盖、验证和读取等功能。电子印章遵循 GM/T 0031 技术标准。

#### 6.4.3 数字水印服务

数字水印服务包括数字水印嵌入和提取服务，以及文件隐藏服务。中间件通过调用数字水印服务实现在文件中嵌入数字水印和提取数字水印，或将信息隐藏于文件中。

#### 6.4.4 指纹识别服务

指纹识别服务实现指纹与操作权限的绑定。在操作者进行授权的操作前，中间件通过调用指纹识别服务对操作者的物理身份进行验证。

### 6.5 密钥对象

中间件涉及的密钥对象为操作者密钥。

操作者密钥包括签名密钥对和加密密钥对及对应的签名证书和加密证书。使用操作者加密证书及其私钥对文件和标签加密密钥进行加密和解密，使用操作者签名证书及其私钥对文件和标签进行签名和签名验证，以保证安全电子文件在流过程中的机密性、完整性和操作的不可抵赖性。

## 7 标签

### 7.1 标签结构

#### 7.1.1 逻辑结构

##### 7.1.1.1 概述

标签的逻辑结构如图 6 所示：



图 6 标签的逻辑结构

标签的 ASN.1 定义为:

```
SecuredFileLabel ::= SEQUENCE {
    baseHead          SFL_Head,          -- 标签头
    baseText          SFL_Body           -- 标签体
}
```

其中:

baseHead : 标签的基本信息;  
baseText : 标签的内容。

### 7.1.1.2 标签头

标签头的结构如图 7 所示:

标签标识	版本号	证书颁发者	证书序列号	创建时间	最后访问时间	自定义属性	加密属性	签名属性
------	-----	-------	-------	------	--------	-------	------	------

图 7 标签头结构

标签头的 ASN.1 定义:

```
SFL_Head ::= SEQUENCE {
    labelID          UTF8String,          -- 标志
    verID           UTF8String,          -- 版本
    issuer          RDNSSequence,       -- 标签创建者证书的颁发人
    creator         INTEGER,            -- 标签创建者的加密证书序列号
    createTime     GeneralizedTime,     -- 创建时间
    lastAccessTime GeneralizedTime,     -- 最后访问时间
    customAttr     OCTET STRING,        -- 自定义属性
    encryptionAttr EncryptionAttribute, -- 标签体的加密属性
    signAttr       SignAttribute        -- 标签的签名属性
}
```

其中:

labelID : 固定为 “@SFL”;  
verID : SFL 的格式版本, 如 “1.3”;  
customAttr : 用户自定义  
issuer : 标签创建者证书的颁发人, 遵循 GM/T 0015 中 ASN.1 的定义;  
creator : 标签创建者的加密证书序列号;  
createTime : 创建标签的时间, 为系统时间;  
lastAccessTime : 对标签最后写操作的时间, 为系统时间。

标签签名属性的 ASN.1 定义:

```
SignAttribute ::= SEQUENCE {
    signer          Certificate,         -- 签名证书
```

```

    signAlg      ObjectIdentifier,  --算法标识
    signature    BIT STRING       --签名值
}

```

其中:

signer : 签名者的签名证书, 遵循 GM/T 0015 中 ASN.1 的定义;  
 signAlg : 签名算法标识;  
 signature : 是对标签中除了签名值属性以外的所有内容的签名结果。  
 标签加密属性的 ASN.1 定义:

```

EncryptionAttribute ::= SEQUENCE {
    algorithmID      ObjectIdentifier,  --算法标识
    algMode          INTEGER,          --算法模式
    numBits          INTEGER,          --反馈位数
    decryptorList    DECRYPTLIST       --解密者列表
}

```

其中:

algorithmID : 算法标识。

```

DECRYPTLIST ::= SEQUENCE {
    DecryptorSet
}

```

```

DecryptorSet ::= SET OF Decryptor

```

解密者属性的 ASN.1 定义:

```

Decryptor ::= SEQUENCE {
    issuerName      RDNSSequence,      --颁发者名称
    serialnumber    INTEGER,          --加密者证书序列号
    alg             ObjectIdentifier,   --打包算法
    sessionKey      OCTSTRING         --被加密过的密钥
}

```

其中:

issuerName : 颁发者名字, 遵循 GM/T 0015 中 ASN.1 的定义;  
 serialnumber : 加密者证书序列号;  
 alg : 打包算法;  
 sessionKey : 被加密过的密钥。

### 7.1.1.3 标签体

标签体结构图 8 所示:

签名属性集合	权限属性	印章属性	水印属性	指纹属性	标识属性	内容属性	对齐属性	扩展属性	日志属性
--------	------	------	------	------	------	------	------	------	------

图 8 标签体结构

标签体的 ASN.1 定义:

```

SFL_Body ::= SEQUENCE {
    mSAttribute      SignAttributeCollection,
}

```

## GM/T 0055-2018

```
    priv          PrivAttr,  
    stampAttr    explicit [0] StampAttr,  
    waterMark    explicit [1] WaterMarkAttr,  
    fingerPrint  explicit [2] FingerAttr,  
    identify     IdentifyAttr,  
    bFileAttr    ContentAttr,  
    align        AlignAttr,  
    extend       explicit [0] ExtendAttr optional,  
    log          explicit [1] LogAttr optional  
}
```

签名属性集合的 ASN.1 定义:

SignAttributeCollection ::= SEQUENCE OF SignAttribute

其中:

mSAttribute	: 签名属性集合;
privAttr	: 详见本标准“7.2.3 权限属性”;
stampAttr	: 详见本标准“7.2.4 印章属性”;
waterMarkAttr	: 详见本标准“7.2.5 水印属性”;
fingerAttr	: 详见本标准“7.2.6 指纹属性”;
contentAttr	: 详见本标准“7.2.7 内容属性”;
identifyAttr	: 详见本标准“7.2.8 标识属性”;
extendAttr	: 详见本标准“7.2.9 扩展属性”;
align	: 详见本标准“7.2.10 对齐属性”;
logAttr	: 详见本标准“7.2.11 日志属性”。

### 7.1.2 存储结构

标签的存储结构如图 9 所示:

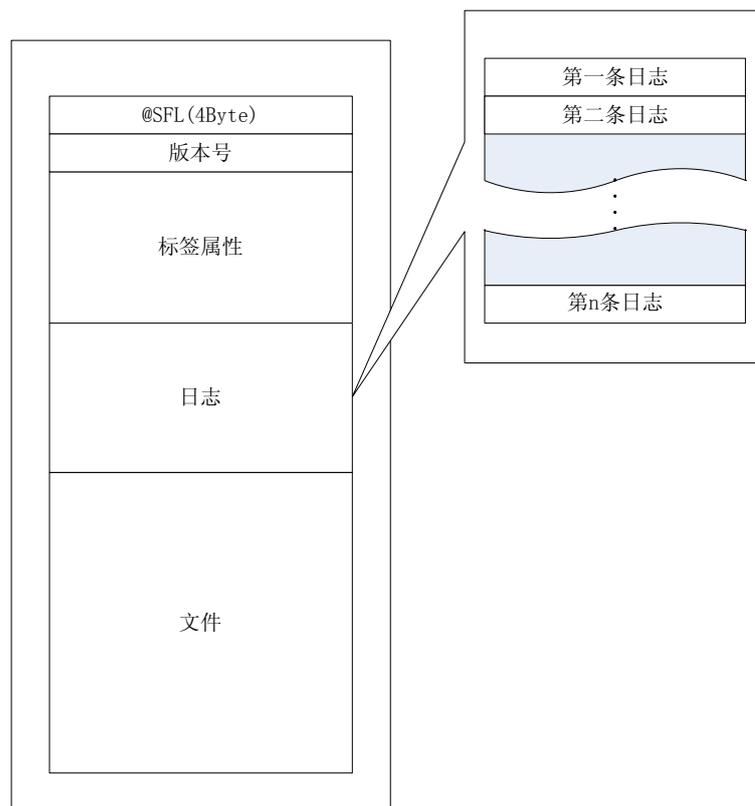


图9 标签的存储格式

其中：

如果文件实体不在标签中，表示文件是独立保存的。

## 7.2 标签属性

### 7.2.1 概述

标签属性标识了与安全电子文件相关的密码操作以及操作日志。包括：签名属性及集合、权限属性、印章属性、水印属性、指纹属性、内容属性、标识属性、扩展属性、对齐属性和日志属性。

### 7.2.2 签名属性及集合

详见本标准“7.1.1.2 标签头”中的签名属性及“7.1.1.3 标签体”中的签名属性集合的 ASN.1 描述。

### 7.2.3 权限属性

权限属性定义对文件的的操作权限，权限属性的 ASN.1 定义：

```
PrivAttr ::= SEQUENCE {
    OPERATORLIST      -- 操作者权限列表
}
```

OPERATORLIST ::= SET OF OperatorAttribute

操作者权限属性 ASN.1 定义如下：

```
OperatorAttribute ::= SEQUENCE {
```

## GM/T 0055-2018

```
operator      Decryptor,  
privilege     SEQUENCE  
}
```

其中:

operator : 结构同本标准的“7.1.1.2”中 Decryptor 的 ASN.1 定义。

操作者权限的 ASN.1 定义:

```
privilege ::= SEQUENCE {  
    cert          Certificate,          --此权限的用户证书  
    read          BOOLEAN,             --读权限  
    totalRead     INTEGER,             --可读次数  
    alreadyRead   INTEGER,             --已读次数  
    write         BOOLEAN,             --写权限  
    delete       BOOLEAN,             --删除权限  
    print         BOOLEAN,             --打印权限  
    totalPrint    INTEGER,             --可打印份数  
    alreadyPrint  INTEGER,             --已打印份数  
    expri        EXPRIVILEGE optional --扩展权限  
}
```

```
EXPRIVILEGE ::= SEQUENCE {  
    EXPRIVILEGESet  
}
```

```
EXPRIVILEGESet ::= SET OF ExPrivilege
```

扩展权限的 ASN.1 定义:

```
ExPrivilege ::= SEQUENCE {  
    priID         INTEGER,             --权限 ID  
    reserve       INTEGER,             --保留字段  
}
```

priID : 用户自定义的唯一标识;

reserve : 用户自定义标识所对应的属性。

### 7.2.4 印章属性

印章属性定义了对文件进行盖章、验章操作。印章属性的具体结构参见 GM/T 0031。

### 7.2.5 水印属性

水印属性定义了对文件嵌入/提取水印的操作，水印属性结构如图 10 所示:

水印载体	水印信息	水印算法	水印添加者	添加时间
------	------	------	-------	------

图 10 水印属性结构

水印属性的 ASN.1 定义:

```

WatermarkAttribute ::= SEQUENCE {
    embedFile          UTF8String,          --水印载体
    watermarkInfo      UTF8String,          --水印信息
    watermarkAlg       AlgorithmAttr,       --水印算法
    watermarkCreator   UTF8String,         --水印添加者
    watermarkCreateTime GeneralizedTime     --添加时间
}

```

其中:

embedFile : 包含水印的载体文件;  
watermarkInfo : 希望隐藏在载体文件中的信息;  
watermarkCreator : 应用系统指定的添加者标识;  
watermarkCreateTime : 本次添加水印的时间, 为系统时间。

水印算法的 ASN.1 定义:

```

AlgorithmAttr ::= SEQUENCE {
    embedAlgID          INTEGER,            --嵌入算法标识
    embedMode           INTEGER            --嵌入模式
}

```

## 7.2.6 指纹属性

指纹属性定义了对文件进行指纹识别的操作, 指纹属性的结构如图 11 所示:

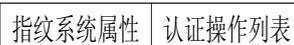


图 11 指纹属性结构

指纹属性的 ASN.1 定义:

```

FingerPrintAttribute ::= SEQUENCE {
    bioSystemAttr      BioSystemAttribute, --系统属性
    Fpopper            FPOPER              --操作权限
}

```

```

FPOPER ::= SEQUENCE {
    FPOperSet
}

```

```

FPOperSet ::= SET OF OPERPRIVILEGE

```

```

OPERPRIVILEGE ::= SEQUENCE {
    userAttr           UTF8String,         --被授权者
    actionRead         BOOLEAN,           --文件读操作
    actionWrite        BOOLEAN,           --文件写操作
    actionPrint        BOOLEAN,           --文件打印操作
    actionEncrypt      BOOLEAN,           --文件加密操作
    actionDecrypt      BOOLEAN,           --文件解密操作
}

```

GM/T 0055-2018

```

    actionSign          BOOLEAN,          --文件签名操作
    actionWatermark     BOOLEAN,          --文件水印操作
    actionStamp         BOOLEAN,          --文件签章操作
    actionLogSign       BOOLEAN          --日志签名操作
}

```

其中:

```

bioSystemAttr      : 指纹识别的系统属性;
userAttr           : 要求进行指纹认证的被授权者签名证书;
actionRead         : 是否对“文件读操作”进行指纹身份识别;
actionWrite        : 是否对“文件写操作”进行指纹身份识别;
actionPrint        : 是否对“文件打印操作”进行指纹身份识别;
actionEncrypt      : 是否对“文件加密操作”进行指纹身份识别;
actionDecrypt      : 是否对“文件解密操作”进行指纹身份识别;
actionSign         : 是否对“文件签名操作”进行指纹身份识别;
actionWatermark    : 是否对“文件水印操作”进行指纹身份识别;
actionStamp        : 是否对“文件签章操作”进行指纹身份识别;
actionLogSign      : 是否对“文件日志签名操作”进行指纹身份识别。

```

指纹识别的系统属性的 ASN.1 定义:

```

BioSystemAttribute ::= SEQUENCE {
    templateDBAttr     UTF8String,        --指纹存储载体描述
    templateFormatAttr UTF8String,        --指纹模板格式描述
    algorithmAttr      UTF8String,        --指纹算法描述
    bioSecurityLevel   UTF8String        --指纹认证安全级别
}

```

其中:

```

templateDBAttr      : 指纹模板信息存储的位置, 访问方式, 访问口令等;
templateFormatAttr  : 指纹模板的格式定义;
algorithmAttr        : 对指纹匹配算法的要求;
bioSecurityLevel    : 对 FAR (False Accept Rate) 的要求。

```

### 7.2.7 内容属性

内容属性定义了与文件内容相关的信息, 内容属性结构如图 12 所示:

文件类型	文件密级	文件大小	文件名字	文件标题	修改日期	失效日期	废止日期	销毁日期
------	------	------	------	------	------	------	------	------

图 12 内容属性结构

内容属性的 ASN.1 定义:

```

ContentAttribute ::= SEQUENCE {
    fileType           INTEGER,          --文件类型
    fileLevel          INTEGER,          --文件级别
    fileSize           INTEGER,          --文件大小
    fileName           UTF8String,       --文件名
}

```

fileTitle	UTF8String,	--文件标题
fileDate	GeneralizedTime,	--文件最后修改日期
expiredDate	GeneralizedTime,	--失效日期
desuetudeDate	GeneralizedTime,	--废止日期
destroyData	GeneralizedTime	--销毁日期

其中:

fileType	: 文件的类型, 具体含义由应用定义;
fileLevel	: 文件的级别, 具体含义由应用定义;
fileSize	: 文件明文的字节数;
fileName	: 文件名;
fileTitle	: 文件标题;
fileDate	: 文件最后修改日期;
expiredDate	: 文件的失效日期, 超过该日期文件即失效, 无法修改, 只能读取;
desuetudeDate	: 在文件有效期内, 让文件失效的日期;
destroyData	: 文件的销毁日期, 过期后, 文件无法读取。

### 7.2.8 标识属性

标识属性是文件的唯一标识, 该标识在安全电子文件创建时确定, 并在安全电子文件全生命周期中保持不变, 标识属性结构如图 13 所示:

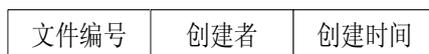


图 13 标识属性结构

标识属性的 ASN.1 定义:

```
IdentifyAttr ::= SEQUENCE {
    fileID          UTF8string,          --文件编号
    creator         UTF8string,         --文件创建者
    createTime     GeneralizedTime     --文件创建时间
}
```

其中:

fileID	: 应用系统确定的文件编号;
creator	: 应用系统指定的创建者;
createTime	: 文件创建时间, 该时间为系统时间。

### 7.2.9 扩展属性

扩展属性是由应用系统自定义的属性, 由应用系统定义结构及各要素的含义, 扩展属性的结构如图 14 所示:



图 14 扩展属性结构

## GM/T 0055-2018

扩展属性的 ASN.1 定义:

```
ExtendAttribute ::= SEQUENCE {
    Extend          EXTEND
}
EXTEND ::= SET OF SEQUENCE {
    exAttrID        INTEGER,           --扩展属性标识
    exAttrContent   OCTET STRING      --扩展属性内容
}
```

### 7.2.10 对齐属性

对齐属性定义了标签文件两部分的对齐大小，分别为标签对齐和文件对齐（属于应用写入部分），如图 15 所示：

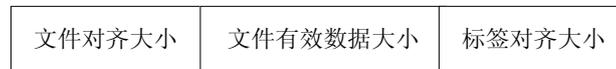


图 15 对齐属性结构

对齐属性的 ASN.1 定义:

```
AlignAttr ::= SEQUENCE {
    fileAlignSize    INTEGER,
    fileEffectSize   INTEGER,
    labelAlignSize   INTEGER
}
```

其中:

fileAlignSize : 标签文件中应用需要的文件数据部分对齐大小;

fileEffectSize : 应用写入的数据大小;

labelAlignSize : ASN.1 编码部分的对齐大小。

### 7.2.11 日志属性

日志属性定义了对文件操作过程中的日志信息，日志属性结构如图 16 所示：

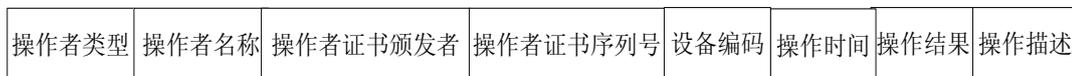


图 16 日志属性结构

日志属性的 ASN.1 定义:

```
LogAttribute ::= SEQUENCE {
    LogContent       LOGCONTENT
}
LOGCONTENT ::= SET OF SEQUENCE {
    actionType       INTEGER,           --操作类型
    operatorName     UTF8String,       --操作者名称
}
```

```

issuerName          RDNSSequence,          --操作者所持有证书的颁发人
operatorCert        INTEGER,                --操作者证书序列号
deviceNo            INTEGER,                --设备编码
actionTime          GeneralizedTime,        --操作时间
actionResult        INTEGER,                --操作结果
operateDesc         UTF8String              --操作描述
}

```

其中：

actionType : 对文件操作行为的编码，必填，如表 2 所示；

operatorName: 操作者名称，必填；

issuerName : 操作者所持有证书的颁发人，遵循 GM/T 0015 中 ASN.1 的定义；

operatorCert: 操作者的签名证书序列号，必填；

deviceNo : 设备编号，用户自定义格式，可选；

actionTime : 操作时间，为系统时间，必填；

actionResult: 操作结果；

operateDesc : 操作描述，必填。

表 2 操作行为编码

编码	操作行为
0	读文件
1	打印文件
2	修改文件
3	删除文件
4	印章操作
5	水印操作
6	指纹操作

## 8 基础密码操作

### 8.1 概述

基础密码操作是指中间件对标签和文件实施的各种共性密码操作。中间件密码操作时使用的数字证书由应用系统确保有效性，然后才可以进行密码操作。

### 8.2 标签的完整性与绑定关系的建立

标签建立和更新时需要进行此操作，过程如下：

- 从标签头的签名属性 (SFL\_Head:signAttr) 中获取算法标识 (algorithmID)；
- 用算法标识中指定的杂凑算法对除标签头中的签名值外的所有标签内容计算摘要；
- 用算法标识中指定的公钥算法及操作者签名私钥对步骤 b) 生成的摘要进行数字签名；
- 将签名值填充在标签头的签名属性 (SFL\_Head:signAttr) 中；
- 如果标签头的加密属性 (SFL\_Head:encryptionAttr) 不为空，则使用算法标识中指定的分组密码算法对整个标签体进行加密；加密密钥随机产生，使用算法标识中指定的公钥算法及操作者的加密公钥生成数字信封，存放在标签头的加密属性的解密者列表 (SFL\_Head:encryptionAttr: decryptorList) 中。

### 8.3 标签的完整性与绑定关系的验证

中间件在对安全电子文件进行操作前均应进行此验证操作，过程如下：

- a) 从标签头的签名属性 (SFL\_Head:signAttr) 中获取算法标识 (algorithmID)；
- b) 如果标签头的加密属性 (SFL\_Head:encryptionAttr) 不为空，则使用算法标识中指定的分组密码算法对整个标签体内容进行解密；
- c) 用算法标识中指定的杂凑算法对除标签头中的签名值属性外的所有标签内容计算摘要并验证签名的正确性；
- d) 验证标签体的内容属性中的失效日期 (SFL\_Body:contentAttr:expiredDate) 和废止时间 (SFL\_Body:contentAttr:desuetudeDate)，确定文件的有效性。如文件已经失效，仅可进行读操作。

### 8.4 文件签名

文件签名是对应用系统提交的文件进行签名操作。过程如下：

- a) 获取之前设置的签名算法标识 (algorithmID)；
- b) 用算法标识中指定的杂凑算法对文件计算摘要；
- c) 使用操作者签名私钥对摘要值进行数字签名；
- d) 将签名值、算法及证书填充到标签体的签名属性集合 (SFL\_Body: mSAttribute) 中。

### 8.5 添加文件签名

添加文件签名是指除创建该文件的用户之外，其它用户对此文件追加文件签名的方法，过程如下：

- a) 获取之前设置的签名算法标识 (algorithmID)；
- b) 用算法标识中指定的杂凑算法对文件计算摘要；
- c) 使用操作者签名私钥对摘要值进行数字签名；
- d) 将签名值及证书增加到标签体的签名属性集合 (SFL\_Body: mSAttribute) 中。

### 8.6 验证文件签名

验证文件签名是指对应用系统指定的文件进行验证签名操作，过程如下：

- a) 获取之前设置的签名算法标识 (algorithmID)；
- b) 用算法标识中指定的杂凑算法对文件计算摘要；
- c) 依次验证标签体中保存的所有文件签名值。

### 8.7 文件加密

文件加密是指对应用系统指定的文件进行加密操作，过程如下：

- a) 从标签头的加密属性 (SFL\_Head:encryptionAttr) 中获取算法标识 (algorithmID)；
- b) 随机生成加密密钥，使用算法标识中的分组密码算法对文件进行加密；
- c) 使用算法标识中指定的公钥算法及解密者加密公钥，对加密密钥进行加密，存放在标签体的操作者列表中 (SFL\_Body:PrivAttr: operatorList) 中。

### 8.8 文件解密

文件解密是指对应用系统指定的文件进行解密操作，过程如下：

- a) 从标签体的权限属性 (SFL\_Body: PrivAttr) 中获取算法标识 (algorithmID)；
- b) 从标签体权限属性的操作者列表 (SFL\_Body: PrivAttr: operatorList) 中获取数字信封，使用操作者加密私钥解密数字信封得到加密密钥；
- c) 使用算法标识中指定的分组算法及加密密钥解密文件。

## 9 安全电子文件密码服务接口

## 9.1 常量定义

常量定义见表 3

表 3 常量定义

数据类型	预定义值	描述
typedef unsigned char	BYTE	8 位无符号型
typedef unsigned int	UINT32	32 位无符号整数
typedef int	INT32	32 位有符号整数
typedef unsigned int	BOOL	布尔类型，非零表示真
typedef _int64/uint64_t	TIME64	64 位时间
操作类型		
#define LOG_READ	1	读文件
#define LOG_WRITE	2	写文件
#define LOG_PRINT	3	打印文件
#define LOG_STAMP	6	印章操作
#define LOG_WATERMARK	7	水印操作
#define LOG_FINGERPRINT	11	指纹操作
#define LOG_REMOVE	12	删除日志
算法模式		
#define MODE_ECB	1	ECB 模式
#define MODE_CBC	2	CBC 模式
#define MODE_OFB	3	OFB 模式
#define MODE_CFB	4	CFB 模式
错误码		
#define LR_SUCCESS	0x00000000	成功
#define LR_UNKNOWN_ERROR	0x09000001	未知错误
#define LR_INVALID_PARAM	0x09000002	非法参数
#define LR_LABEL_ABOLISHED	0x09000003	该标签已经废止
#define LR_LABEL_EXPIRED	0x09000004	标签已经失效
#define LR_NO_PRIVILEGE	0x09000005	没有权限
#define LR_SIGN_KEY_NOT_MATCH	0x09000006	签名公私钥不匹配
#define LR_EX_ATTR_EXISTENT	0x09000007	扩展属性 ID 已经存在
#define LR_EX_ATTR_INEXISTENT	0x09000008	没找到相应的属性
#define LR_COUNT_READ_INSUFFICIENCY	0x09000009	剩余读份数不足
#define LR_INVALID_HANDLE	0x0900000a	句柄无效
#define LR_NO_SET_SIGNALG	0x0900000b	没有设置签名算法
#define LR_NO_SET_PRIVILEGE	0x0900000c	没有添加权限
#define LR_NO_SET_CRYPTALG	0x0900000d	没有设置加密算法
#define LR_NOT_RECOGNIZE_CRYPTALG	0x0900000e	不可识别的加密算法

数据类型	预定义值	描述
#define LR_NOT_RECOGNIZE_SINGALG	0x0900000f	不可识别的签名算法
#define LR_FILE_DEFECTED	0x09000010	文件已经失效
#define LR_VERIFY_LABELHEAD_ERROR	0x09000011	验证标签头失败
#define LR_GET_PAD_SIZE_ERROR	0x09000012	获得补丁大小失败
#define LR_SET_PAD_ERROR	0x09000013	设置补丁失败
#define LR_SET_PAD_SIZE_ERROR	0x09000014	设置补丁大小失败
#define LR_MALLOC_MEM_ERROR	0x09000015	分配内存失败
#define LR_OPEN_FILE_ERROR	0x09000016	打开文件失败
#define LR_CALCLABELSIZE_ERROR	0x09000017	计算标签大小失败
#define LR_ENCODE_PLAINT_ERROR	0x09000018	编码明文标签失败
#define LR_DECODE_PLAINT_ERROR	0x09000019	解码明文标签失败
#define LR_ENCODE_LABEL_HEAD_ERROR	0x0900001a	编码标签头失败
#define LR_DECODE_LABEL_HEAD_ERROR	0x0900001b	解码标签头失败
#define LR_DECRYPT_LABEL_BODY_ERROR	0x0900001c	解密标签体失败
#define LR_DECRYPT_BODY_KEY_ERROR	0x0900001d	解密标签体密钥失败
#define LR_NOT_FIND_PRIVILEGE_ERROR	0x0900001e	没有查找到权限
#define LR_FORBIDDEN_READ_ERROR	0x0900001f	禁止读操作
#define LR_READ_COUNT_USED_ERROR	0x09000020	读次数已经使用完毕
#define LR_DECRYPT_CIPHER_ERROR	0x09000021	解密密文失败
#define LR_COPY_FILE_ERROR	0x09000022	复制文件失败
#define LR_VERIFY_CIPHER_INIT_ERROR	0x09000023	验证密文初始化失败
#define LR_VERIFY_CIPHER_FAILURE	0x09000024	验证数据失败
#define LR_FORBIDDEN_WRITE_ERROR	0x09000025	禁止写操作
#define LR_GET_FILE_SIZE_ERROR	0x09000026	获得文件大小失败
#define LR_SET_FILE_SIZE_ERROR	0x09000027	设置文件大小失败
#define LR_SIGN_CIPHER_INIT_ERROR	0x09000028	签名数据初始化失败
#define LR_SIGN_CIPHER_ERROR	0x09000029	签名数据失败
#define LR_ENCRYPT_CIPHER_ERROR	0x0900002a	加密数据失败
#define LR_PAD_FAILURE	0x0900002b	添加补丁失败
#define LR_DECODE_LABEL_BODY_ERROR	0x0900002c	解码标签体失败
#define LR_NOT_FIND_FILE_DIGITENVLOP_ERROR	0x0900002d	没有查找到数字信封
#define LR_GET_ASN_SIGNATURE_MESSAGE_ERROR	0x0900002e	获得标签 ASN.1 编码消息 失败
#define LR_VERIFY_LABELHEAD_INIT_ERROR	0x0900002f	验证标签头初始化失败
#define LR_DCRYPT_DIGITALENVELOP_ERROR	0x09000030	解密数字信封失败
#define LR_PASSWD_ERR	0x09000031	密码错误
#define LR_ENCODE_SIGNATTR_ERROR	0x09000032	编码多签签名集合失败

## 9.2 结构定义

### 9.2.1 标识属性

IIdentifyAttr的结构定义如表4所示:

表 4 标识属性结构

字段名称	数据类型	数据长度 (字节)	含义
szFileNo	Char [32]	32	文件编号
szFileCreator	Char [32]	32	文件创建者
tFileCreate	TIME64	8	文件创建时间

实际数据结构定义:

```
typedef struct tagIIdentifyAttr
{
    char        szFileNo[32];
    char        szFileCreator[32];
    TIME64      tFileCreate;
} IIdentifyAttr;
```

### 9.2.2 算法属性

IAlgAttr的结构定义如表5所示:

表 5 算法属性结构

字段名称	数据类型	数据长度 (字节)	含义
bCrypt	BOOL	1	是否加密
szCryptAlg	char*	'\0' 结尾	加密算法
ucCryptMode	BYTE	1	加密模式
numbits	UINT	4	反馈位数
szSignAlg	char*	'\0' 结尾	签名 (摘要) 算法

实际数据结构定义:

```
typedef struct tagAlgAttr
{
    BOOL        bCrypt;
    char        *szCryptAlg;
    BYTE        ucCryptMode;
    UINT        numbits;
    char        *szSignAlg;
} IAlgAttr;
```

### 9.2.3 文件内存

FileBuffer的结构定义如表6所示:

表 6 文件内存结构

字段名称	数据类型	数据长度 (字节)	含义
pbData	BYTE*	nLen	数据内容

字段名称	数据类型	数据长度（字节）	含义
nLen	UINT	4	数据长度

实际数据结构定义：

```
typedef struct tagBuffer
{
    BYTE *pbData;
    UINT nLen;
}FileBuffer, CertBuffer, SignBuffer, DataBuffer, CustomAttr;
```

#### 9.2.4 扩展权限属性

IexPrivilegeAttr的结构定义如表7所示：

表 7 扩展权限属性结构

字段名称	数据类型	数据长度（字节）	含义
nPriID	USHORT	2	扩展权限 ID
nReserve	UINT	4	保留字段（用户自定义含义）

实际数据结构定义：

```
typedef struct tagExPrivilegeAttr
{
    USHORT nPriID;
    UINT nReserve;
}IExPrivilegeAttr;
```

#### 9.2.5 扩展权限属性列表

IExPrivilegeAttrList的结构定义如表8所示：

表 8 扩展属性列表结构

字段名称	数据类型	数据长度（字节）	含义
nExtPriCount	UINT	4	扩展权限个数
pExtPriAttrList	IExPrivilegeAttr*	n	扩展权限

实际数据结构定义：

```
typedef struct tagExPrivilegeAttrList
{
    UINT nExtPriCount;
    IExPrivilegeAttr *pExtPriAttrList;
}IExPrivilegeAttrList;
```

#### 9.2.6 权限属性

IPrivilege的结构定义如表9所示：

表 9 权限属性结构

字段名称	数据类型	数据长度（字节）	含义
exCert	CertBuffer	n	个人加密证书
bRead	BYTE	1	是否有读权限
uTotalRead	UINT	4	可读份数
uAlread	UINT	4	已读份数
bWrite	BYTE	1	是否有写权限
bDel	BYTE	1	是否有删除权限
bPrint	BYTE	1	是否有打印权限
uPrintCount	UINT	4	可打印份数
uPrintedCount	UINT	4	已打印份数
exPriList	IExPrivilegeAttrList	n	扩展权限

实际数据结构定义：

```
typedef struct tagPrivilegeAttr
{
    CertBuffer *exCert;
    BYTE      bRead;
    UINT      uTotalRead;
    UINT      uAlread;
    BYTE      bWrite;
    BYTE      bDelete;
    BYTE      bPrint;
    UINT      uPrintCount;
    UINT      uPrintedCount;
    IExPrivilegeAttrList  exPriList optional ;
} IPrivilegeAttr;
```

### 9.2.7 扩展属性

IExtendAttr的结构定义如表10所示：

表 10 扩展属性结构

字段名称	数据类型	数据长度（字节）	含义
nAttrID	USHORT	2	扩展属性ID
pbContent	BYTE*	nLen	扩展属性内容
nLen	UINT	4	扩展属性长度

实际数据结构定义：

```
typedef struct tagExtendAttr
{
    USHORT      nAttrID;
    unsigned char *pbContent;
    UINT      nLen;
} IExtendAttr;
```

## 9.2.8 日志属性

IlogAttr的结构定义如表11所示：

表 11 日志属性结构

字段名称	数据类型	数据长度（字节）	含义
uType	UINT	4	操作类型
szName	Char*	n	操作者
pIssuer	IssuerName *	N	颁发证书者
pSerial	BYTE*	uSerial	操作者签名证书序列号
uSerial	UINT	4	签名证书序列号长度
szHardCode	Char*	n	硬件编码
operTime	TIME64	8	操作时间
uResult	UINT	4	操作结果
szDescript	Char*	N	操作描述

实际数据结构定义：

```
typedef struct tagDName
```

```
{
```

```
    Char *szType;
```

```
    BYTE *pValue;
```

```
    UINT uValen;
```

```
}DName;
```

```
typedef struct tagIssuerName
```

```
{
```

```
    DName *pName;
```

```
    UINT uCount;
```

```
}IssuerName;
```

```
typedef struct tagLogAttr
```

```
{
```

```
    UINT uType;
```

```
    Char *szName;
```

```
    IssuerName *pIssuer;
```

```
    BYTE *pSerial;
```

```
    UINT uSerial;
```

```
    Char *szHardCode;
```

```
    BYTE uHCLen;
```

```
    TIME64 operTime;
```

```
    UINT uResult;
```

```
    Char *szDescript;
```

```
}IlogAttr;
```

## 9.2.9 印章属性

印章属性定义了对文件进行盖章、验章操作。印章属性的具体结构参见GM/T 0031。

### 9.2.10 水印属性

WaterMarkAttr的结构定义如表12所示：

表 12 水印属性结构

字段名称	数据类型	数据长度（字节）	含义
embedFile	CertBuffer	n	包含水印的载体文件
waterLoadInfo	CertBuffer	n	水印信息
embedAlgID	UINT	4	算法 ID
embedMode	UINT	4	算法模式
creator	Char*	n	创建者
creatorTime	TIME64	8	创建时间

实际数据结构定义：

```
typedef struct tagWaterMarkAttr
{
    CertBuffer embedFile;
    CertBuffer waterLoadInfo;
    UINT embedAlgID;
    UINT embedMode;
    char creator[128];
    TIME64 creatorTime;
}WaterMarkAttr;
```

### 9.2.11 指纹标签属性

FingerPrintTagAttribute 的结构定义如表 13 所示：

表 13 指纹标签属性结构

字段名称	数据类型	数据长度	含义
bioVerifyActionList	BioVerifyAction	BioVerifyAction 数据结构长度	指纹识别操作列表
bioSysAttr	BioSystemAttribute	BioSystemAttribute 数据结构长度	指纹识别系统属性

实际数据结构定义：

```
typedef struct {
    BioVerifyAction bioVerifyActionList;
    BioSystemAttribute bioSysAttr;
} FingerPrintTagAttribute;
```

BioVerifyAction 的结构定义如表 14 所示：

表 14 指纹验证行为结构

字段名称	数据类型	数据长度	含义
operator	Certificate	Certificate 数 数据结构长度	用户的标识信息
bActionRead	Bool	1	是否对“文件读操作”进行指纹身份识别
bActionWrite	Bool	1	是否对“文件写操作”进行指纹身份识别
bActionPrint	Bool	1	是否对“文件打印操作”进行指纹身份识别
bActionEncrypt	Bool	1	是否对“文件加密操作”进行指纹身份识别
bActionDecrypt	Bool	1	是否对“文件解密操作”进行指纹身份识别
bActionSign	Bool	1	是否对“文件签名操作”进行指纹身份识别
bActionWatermark	Bool	1	是否对“文件水印操作”进行指纹身份识别
bActionStamp	Bool	1	是否对“文件签章操作”进行指纹身份识别

实际数据结构定义:

```
typedef struct{
    Certificate operator;
    bool bActionRead;
    bool bActionWrite;
    bool bActionPrint;
    bool bActionEncrypt;
    bool bActionDecrypt;
    bool bActionSign;
    bool bActionWatermark;
    bool bActionStamp;
} BioVerifyAction, *PBioVerifyAction;
```

BioVerifyActionList的结构定义如表15所示:

表 15 指纹识别操作列表结构

字段名称	数据类型	数据长度	含义
nOperatorCount	UINT32	4	指纹识别操作列表包含的操作者数量
bActionRead	BioVerifyAction	BioVerifyAction 数据结构长度	指纹识别操作列表

实际数据结构定义:

```
typedef struct{
    UINT32          nOperatorCount;
    BioVerifyAction* bioVerifyAction;
}BioVerifyActionList, *PBioVerifyActionList;
```

BioAttr的结构定义如表16所示:

表 16 指纹属性结构

字段名称	数据类型	数据长度	含义
szTemplateDBAttr	Char*	字符串长度	指纹存储载体描述
szTemplateAttr	Char*	字符串长度	指纹模板格式描述
szAlgorithmAttr	Char*	字符串长度	指纹算法描述
bioSecurityLevel	int	4	指纹识别安全级别

实际数据结构定义:

```
typedef struct{
    char*    szTemplateDBAttr;
    char*    szTemplateAttr;
    char*    szAlgorithmAttr;
    int      bioSecurityLevel;
} BioSystemAttribute;
```

### 9.2.12 身份标识

SToken 的结构定义如表 17 所示:

表 17 身份标识结构

字段名称	数据类型	数据长度	含义
exCert	CertBuffer		加密证书
signCert	CertBuffer		签名证书

实际数据结构定义:

```
typedef struct tagToken
{
    CertBuffer exCert;
    CertBuffer signCert;
}SToken;
```

## 9.3 接口函数组成和功能说明

### 9.3.1 概述

接口函数由以下部分组成:

- a) 初始化函数
- b) 标签和文件操作函数
- c) 属性操作函数
- d) 密码初始化函数

### 9.3.2 初始化函数

初始化函数用户系统初始参数的设定和设备连接等。

### 9.3.3 标签和文件操作函数

标签操作函数用户标签的打开、读取、修改、保存以及关闭等，同时处理文件的加密和解密操作。

### 9.3.4 属性操作函数

## GM/T 0055-2018

标签属性的添加、修改、删除、获取等操作。

### 9.3.5 密码操作函数

密码操作函数用于对分段的数据块进行加解密以及签名和验证操作。

## 9.4 接口函数定义

### 9.4.1 初始化函数

#### 9.4.1.1 概述

初始化函数包括以下具体函数：

- a) 设置密码基础服务：SFF\_SetProvider
- b) 获取密码基础服务：SFF\_GetProvider
- c) 登录密码基础服务：SFF\_Login

#### 9.4.1.2 设置密码基础服务

原型：int SFL\_API SFF\_SetProvider(IN const char \*pszCSPName);

描述：1) 检查密码基础服务是否支持；  
2) 重新初始化密码服务。

参数：IN pszCSPName : 密码服务名称。

返回值：0 成功。

非0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

#### 9.4.1.3 获取密码基础服务

原型：int SFL\_API SFF\_GetProvider(OUT char pszCSPName[128]);

描述：1) 获取当前使用的密码基础服务名称

参数：OUT pszCSPName : 密码服务名称

返回值：0 成功。

非0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

#### 9.4.1.4 登录密码基础服务设备

原型：int SFL\_API SFF\_Login(IN const char \*szPin);

描述：1) 登录密码基础服务设备。

参数：IN szPin : 密码设备口令。

返回值：0 成功。

非0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

### 9.4.2 标签和文件操作函数

#### 9.4.2.1 概述

- a) 打开安全电子文件标签内存：SFF\_OpenSFLB
- b) 读取外联式安全电子文件内存：SFF\_ExternalReadSFB
- c) 读取内联式安全电子文件内存：SFF\_InternalReadSFB
- d) 修改外联式安全电子文件内存：SFF\_ExternalWriteSFB
- e) 修改内联式安全电子文件内存：SFF\_InternalWriteSFB
- f) 添加文件签名：SFF\_AddSignAttr

- g) 保存安全电子文件内存: SFF\_SaveSFLB
- h) 打开安全电子文件标签: SFF\_OpenSFL
- i) 读取外联式安全电子文件: SFF\_ExternalReadSF
- j) 读取内联式安全电子文件: SFF\_InternalReadSF
- k) 修改外联式安全电子文件: SFF\_ExternalWriteSF
- l) 修改内联式安全电子文件: SFF\_InternalWriteSF
- m) 保存安全电子文件: SFF\_SaveSFL
- n) 关闭安全电子文件: SFF\_CloseSFL
- o) 生成电子签章: SFF\_Stamp
- p) 验证电子签章: SFF\_VerifyStamp
- q) 添加水印: SFF\_SetWaterMarkInfo
- r) 提取水印: SFF\_GetWaterMarkInfo
- s) 释放水印属性内存: SFF\_FreetWaterMarkInfo

#### 9.4.2.2 打开安全电子文件标签内存

原型: `int SFF_OpenSFLB(IN const SToken *pToken,  
                          IN const FileBuffer *pSFLBuffer,  
                          OUT HSFL *phSf1);`

描述: a) 如果是新建, 则创建标签结构;  
      b) 如果是打开, 则解析标签头;  
      c) 根据标签头中的信息, 解密标签体;  
      d) 验证标签体签名;  
      e) 解析标签体。

参数: IN pToken : 打开者, 身份标识;  
      IN pSFLBuffer : 标签数据内容;  
      OUT phSf1 : 返回标签句柄。

返回值: 0 成功。  
      非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.3 读取外联式安全电子文件内存

原型 `int SFL_API SFF_ExternalReadSFB(IN HSFL hSf1,  
  IN const FileBuffer *pSFFBuffer,  
  OUT FileBuffer *pDstFile);`

描述: a) 判断用户权限;  
      b) 解密密文数据。

参数: IN hSf1 : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
      IN pSFFBuffer : 密文数据;  
      OUT pDstFile : 解密后的明文数据, 如果 FileBuffer->pbData 传入 NULL 则返回实际需要的大小。

返回值: 0 成功。  
      非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.4 读取内联式安全电子文件内存

原型: `int SFL_API SFF_InternalReadSFB (IN HSFL hSf1,`

```
OUT FileBuffer *pDstFile);
```

描述: a) 判断用户权限;  
b) 从内联式标签中获取密文;  
c) 解密密文数据。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT pDstFile : 解密后的明文数据, 如果 FileBuffer-> pbData 传入 NULL 则返回实际需要的大小。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.5 修改外联式安全电子文件内存

```
原型: int SFL_API SFF_ExternalWriteSFB(IN HSFL hSfl,
                                       IN const FileBuffer *pSrcFile,
                                       OUT FileBuffer *pDstFile);
```

描述: a) 判断用户权限;  
b) 加密明文数据。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pszSrcFile : 新的明文数据;  
OUT pszDstFile : 加密后的密文数据, 如果 FileBuffer-> pbData 传入 NULL 则返回实际需要的大小。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.6 修改内联式安全电子文件内存

```
原型: int SFL_API SFF_InternalWriteSFB(IN HSFL hSfl,
                                       OUT const FileBuffer *pSrcFile);
```

描述: a) 判断用户权限;  
b) 保存明文数据。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pszSrcFile : 明文数据。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 调用 SFF\_SaveSFL 或者 SFF\_SaveSFLB 时才实际完成数据加密。

#### 9.4.2.7 添加文件签名

```
原型: int SFL_API SFF_AddSignAttr(IN HSFL hSfl);
```

描述: 增加文件的签名至签名集合的末尾。

参数: IN HSFL : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回。

返回值: 0 成功  
非 0 失败

#### 9.4.2.8 保存安全电子文件内存

```
原型: int SFL_API SFF_SaveSFLB(IN HSFL hSfl,
                                OUT FileBuffer *pSFLBuffer);
```

描述: a) 如果是内联式, 则加密明文数据;

b) 签名标签体;

c) 加密标签体;

d) 生成标签。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

OUT pSFLBuffer: 标签数据。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.9 打开安全电子文件标签

原型: int SFL\_API SFF\_OpenSFL(IN const SToken \*pToken,

IN const char \*pszSFL,

OUT HSFL \*phSfl);

描述: a) 如果是新建, 则创建标签结构;

b) 如果是打开, 则解析标签头;

c) 根据标签头中的信息, 解密标签体;

d) 验证标签体签名;

e) 解析标签体。

参数: IN pToken : 打开者, 身份标识;

IN pszSFL : 标签文件名;

OUT phSfl : 返回标签句柄。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.10 读取外联式安全电子文件

原型 int SFL\_API SFF\_ExternalReadSF(IN HSFL hSfl,

IN const char \*pszSFF,

IN const char \*pszDstFile);

描述: a) 判断用户权限;

b) 解密密文文件。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN pszSFF : 密文文件名;

IN pszDstFile : 解密后的明文文件名。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.11 读取内联式安全电子文件

原型: int SFL\_API SFF\_InternalReadSF(IN HSFL hSfl,

IN const char \*pszDstFile);

描述: a) 判断用户权限;

b) 从内联式标签中获取密文;

c) 解密密文文件。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN pszDstFile : 解密后的明文文件。

## GM/T 0055-2018

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.2.12 修改外联式安全电子文件

原型: `int SFL_API SFF_ExternalWriteSF(IN HSFL hSfl,  
IN const char *pszSrcFile,  
IN const char *pszDstFile);`

描述: a) 判断用户权限;  
b) 加密明文文件。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pszSrcFile : 新的明文文件;  
IN pszDstFile : 加密后的密文文件。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.2.13 修改内联式安全电子文件

原型: `int SFL_API SFF_InternalWriteSF(IN HSFL hSfl,  
IN const char *pszSrcFile);`

描述: a) 判断用户权限;  
b) 保存明文文件名。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pszSrcFile : 明文文件。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 调用 SFF\_SaveSFL 或者 SFF\_SaveSFL 时才实际完成文件加密

### 9.4.2.14 保存安全电子文件

原型: `int SFL_API SFF_SaveSFL(IN HSFL hSfl,  
OUT const char *pszSFL);`

描述: a) 如果是内联式, 则加密明文文件;  
b) 签名标签体;  
c) 加密标签体;  
d) 生成标签。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pszSFL : 标签文件名。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.2.15 关闭安全电子文件标签

原型: `int SFL_API SFF_CloseSFL(IN HSFL hSfl);`

描述: a) 释放标签结构。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.16 生成电子签章

原型: `int SFL_API SFF_Stamp(IN HSFL hSfl);`

描述: a) 从标签中获取签章数据;  
b) 对文件实体生成电子签章, 具体过程参见 GM/T 0031;  
c) 并将签章信息存入标签。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.17 验证电子签章

原型: `int SFL_API SFF_VerifyStamp (IN HSFL hSfl);`

描述: a) 从标签中获取签章数据;  
b) 验证文件实体电子签章, 具体过程参见 GM/T 0031。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.18 添加水印

原型: `int SFL_API SFF_SetWaterMarkInfo(IN HSFL hSfl, IN WaterMarkAttr *pAttr);`

描述: a) 添加水印属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pAttr : 水印属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.19 提取水印

原型: `int SFL_API SFF_GetWaterMarkInfo(IN HSFL hSfl,  
OUT WaterMarkAttr **ppAttr);`

描述: a) 提取水印属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT ppAttr : 返回水印属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.2.20 释放水印属性内存

原型: `int SFL_API SFF_FreetWaterMarkInfo(IN WaterMarkAttr *pAttr);`

描述: a) 释放水印属性内存。

参数: IN pAttr : 水印属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3 属性操作接口

#### 9.4.3.1 概述

GM/T 0055-2018

- a) 设置算法属性: SFF\_SetAlgAttr
- b) 获取算法属性: SFF\_GetAlgAttr
- c) 添加权限属性: SFF\_AddPrivilegeAttr
- d) 获取权限属性: SFF\_GetPrivilegeAttr
- e) 获取权限个数: SFF\_GetPrivilegeCount
- f) 根据序号获取权限: SFF\_GetPrivilege
- g) 释放权限结构内存: SFF\_FreePrivilegeAttr
- h) 设置标志属性: SFF\_SetIdentifyAttr
- i) 获取标识属性: SFF\_GetIdentifyAttr
- j) 设置自定义属性: SFF\_SetCustomAttr
- k) 获取自定义属性: SFF\_GetCustomAttr
- l) 设置文件类型: SFF\_SetFileType
- m) 获取文件类型: SFF\_GetFileType
- n) 设置文件级别: SFF\_SetFileLevel
- o) 获取文件级别: SFF\_GetFileLevel
- p) 设置明文文件大小: SFF\_SetFileSize
- q) 获取明文文件大小: SFF\_GetFileSize
- r) 设置明文文件名: SFF\_SetFileName
- s) 获取明文文件名: SFF\_GetFileName
- t) 设置文件标题: SFF\_SetFileTitle
- u) 获取文件标题: SFF\_GetFileTitle
- v) 设置明文文件创建者: SFF\_SetFileCreator
- w) 获取明文文件创建者: SFF\_GetFileCreator
- x) 设置明文文件最后修改时间: SFF\_SetFileModifyTime
- y) 获取明文文件最后修改时间: SFF\_GetFileModifyTime
- z) 设置安全电子文件失效时间: SFF\_SetExpired
- aa) 获取安全电子文件失效时间: SFF\_GetExpired
- bb) 设置安全电子文件销毁时间: SFF\_SetDestroyTime
- cc) 获取安全电子文件销毁时间: SFF\_GetDestroyTime
- dd) 废止安全电子文件: SFF\_AbolishSF
- ee) 获取废止时间: SFF\_GetAbolishTime
- ff) 添加扩展属性: SFF\_AddExtendAttr
- gg) 获取扩展属性: SFF\_GetExtendAttr
- hh) 获取扩展属性个数: SFF\_GetExtendCount
- ii) 根据 ID 获取扩展属性: SFF\_GetExtend
- jj) 删除扩展属性: SFF\_DelExtendAttr
- kk) 释放扩展属性结构内存: SFF\_FreeExAttr
- ll) 添加日志属性: SFF\_AddLogAttr
- mm) 获取日志个数: SFF\_GetLogCount
- nn) 获取日志: SFF\_GetLogAttr
- oo) 删除日志: SFF\_DelAllLogAttr
- pp) 释放日志属性结构内存: SFF\_FreeLogAttr
- qq) 设置水印属性: SFF\_SetWaterMarkInfo
- rr) 获取水印属性: SFF\_GetWaterMarkInfo



);

描述: a) 根据证书获取用户权限;  
b) 返回用户权限。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pCert : 证书;  
IN uCertLen : 证书长度。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 仅内核版需要该接口。

#### 9.4.3.6 获取权限个数

原型: int SFL\_API SFF\_GetPrivilegeCount(IN HSFL hSfl,  
OUT UINT \*pCount);

描述: a) 从标签中获取权限个数。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pCount : 返回权限个数。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.7 根据序号获取权限

原型: int SFL\_API SFF\_GetPrivilege(IN HSFL hSfl,  
IN UINT uIndex,  
OUT IPrivilegeAttr \*\*ppAttr);

描述: a) 判断序号是否合法;  
b) 根据序号获取权限。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN uIndex : 权限序号, 小于等于 SFF\_GetPrivilegeCount 中返回的权限个数;  
OUT ppAttr : 返回权限结构。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.8 释放权限属性结构内存

原型: void SFL\_API SFF\_FreePrivilegeAttr(IN IPrivilegeAttr \*pAttr);

描述: 1) 释放权限内存。

参数: IN pAttr : 要释放的权限结构。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.9 设置标志属性

原型: int SFL\_API SFF\_SetIdentifyAttr(IN HSFL hSfl,  
IN const IIdentifyAttr \*pAttr);

描述: a) 设置标签标识属性

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN pAttr : 标识属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.10 获取标识属性

原型: int SFL\_API SFF\_GetIdentifyAttr(IN HSFL hSfl,  
OUT IIdentifyAttr \*pAttr);

描述: a) 从标签中获取标识属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

OUT pAttr : 返回标识属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.11 设置用户自定义属性

原型: int SFL\_API SFF\_SetCustomAttr(IN HSFL hSfl,  
IN const CustomAttr \*pAttr);

描述: a) 设置标签自定义属性

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN pAttr : 用户自定义属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.12 获取用户自定义属性

原型: int SFL\_API SFF\_GetCustomAttr (IN HSFL hSfl,  
OUT const CustomAttr \*pAttr);

描述: a) 从标签中获取用户自定义属性

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

OUT pAttr : 用户自定义属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.13 设置文件类型

原型: int SFL\_API SFF\_SetFileType(IN HSFL hSfl, IN UINT nFileType);

描述: a) 设置文件类型。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN nFileType : 文件类型, 含义由使用者自定义。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.14 获取文件类型

原型: int SFL\_API SFF\_GetFileType(IN HSFL hSfl, OUT UINT \*pFileType);

描述: a) 设置文件类型。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

OUT pFileType : 返回文件类型, 含义由使用者自定义。

## GM/T 0055-2018

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.15 设置文件级别

原型: `int SFL_API SFF_SetFileLevel(IN HSFL hSfl, IN UINT nLevel);`  
描述: a) 设置文件级别。  
参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN nLevel : 文件级别。  
返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.16 获取文件级别

原型: `int SFL_API SFF_GetFileLevel(IN HSFL hSfl, OUT UINT *pLevel);`  
描述: a) 获取文件级别。  
参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT nLevel: 返回文件级别。  
返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.17 设置明文文件的大小

原型: `int SFL_API SFF_SetFileSize(IN HSFL hSfl, IN INT64 nFileSize);`  
描述: a) 设置明文文件的大小。  
参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN nFileSize : 文件大小。  
返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。  
注: 该接口仅在使用内存形式分段加密的情况下使用。

### 9.4.3.18 获取明文文件的大小

原型: `int SFL_API SFF_GetFileSize(IN HSFL hSfl, OUT INT64 *pFileSize);`  
描述: a) 获取明文文件的大小。  
参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT pFileSize : 返回文件大小。  
返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.19 设置明文文件名

原型: `int SFL_API SFF_SetFileName(IN HSFL hSfl, IN char *szFileName);`  
描述: a) 设置明文文件名。  
参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN szFileName : 明文文件名。  
返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。  
注: 该接口仅在使用内存形式加密的情况下使用。

**9.4.3.20 获取明文文件名**

原型: `int SFL_API SFF_GetFileName(IN HSFL hSfl, OUT char szFileName[MAX_PATH]);`

描述: a) 获取文件名。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

OUT szFileName : 返回文件名。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

**9.4.3.21 设置文件标题**

原型: `int SFL_API SFF_SetFileTitle(IN HSFL hSfl, IN char *szFileTitle);`

描述: a) 设置文件标题。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN szFileTitle : 文件标题。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

**9.4.3.22 获取文件标题**

原型: `int SFL_API SFF_GetFileTitle(IN HSFL hSfl, OUT char szFileTitle[256]);`

描述: a) 获取文件标题。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

OUT szFileTitle : 返回文件标题。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

**9.4.3.23 设置明文文件创建者**

原型: `int SFL_API SFF_SetFileCreator(IN HSFL hSfl, IN const char szFileCreator[32]);`

描述: a) 设置明文文件创建者。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN szFileCreator : 明文文件创建者。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 该接口仅在使用内存形式加密的情况下使用。

**9.4.3.24 获取明文文件创建者**

原型: `int SFL_API SFF_GetFileCreator(IN HSFL hSfl, OUT char szFileCreator[32]);`

描述: a) 获取明文文件创建者。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN szFileCreator : 返回明文文件创建者。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

**9.4.3.25 设置明文文件最后修改时间**

## GM/T 0055-2018

原型: `int SFL_API SFF_SetFileModifyTime(IN HSFL hSfl, IN TIME64 tModify);`

描述: a) 设置明文文件最后修改时间。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN tModify : 明文文件修改时间。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 该接口仅在使用内存形式加密的情况下使用, 且时间单位为秒。

### 9.4.3.26 获取明文文件最后修改时间

原型: `int SFL_API SFF_GetFileModifyTime(IN HSFL hSfl, OUT TIME64 *ptModify);`

描述: a) 获取明文文件最后修改时间。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN tModify : 返回明文文件修改时间。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.27 设置安全电子文件失效时间

原型: `int SFL_API SFF_SetExpired(IN HSFL hSfl, IN TIME64 tFileExpired);`

描述: a) 设置安全电子文件失效时间。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN tFileExpired : 安全电子文件失效时间。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 失效后, 文件不能修改, 只能读取。

### 9.4.3.28 获取安全电子文件失效时间

原型: `int SFL_API SFF_GetExpired(IN HSFL hSfl, OUT TIME64 *ptFileExpired);`

描述: a) 获取安全电子文件失效时间。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT ptFileExpired: 返回安全电子文件失效时间。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.29 设置安全电子文件销毁时间

原型: `int SFL_API SFF_SetDestroyTime(IN HSFL hSfl, IN TIME64 tFileDestroy);`

描述: a) 设置安全电子文件销毁时间。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN tFileDestroy: 销毁时间, 单位为秒。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 过了销毁时间后, 文件不能修改, 只能读, 具体销毁操作由应用完成。

### 9.4.3.30 获取安全电子文件销毁时间

原型: `int SFL_API SFF_GetDestroyTime(IN HSFL hSfl, OUT TIME64 *tFileDestroy);`

描述: a) 获取安全电子文件销毁时间。

参数: IN hSf1 : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT tFileDestroy: 返回安全电子文件销毁时间。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.31 废止安全电子文件

原型: int SFL\_API SFF\_AbolishSF(IN HSFL hSf1);

描述: a) 废止安全电子文件。

参数: IN hSf1 : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 废止相当于提前失效, 废止时间为当前时间。

#### 9.4.3.32 获取废止时间

原型: int SFL\_API SFF\_GetAbolishTime(IN HSFL hSf1, OUT TIME64 \*tFileAbolish);

描述: a) 获取废止时间。

参数: IN hSf1 : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT tFileAbolish : 返回废止时间。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.33 添加扩展属性

原型: int SFL\_API SFF\_AddExtendAttr(IN HSFL hSf1, IN const IExtendAttr \*pAttr);

描述: a) 判断扩展属性 ID 是否存在;

b) 添加扩展属性。

参数: IN hSf1 : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pAttr : 扩展属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.34 根据 ID 获取扩展属性

原型: int SFL\_API SFF\_GetExtendAttr(IN HSFL hSf1,  
IN USHORT usAttrId,  
OUT IExtendAttr \*\*ppExAttr);

描述: a) 根据 ID 获取扩展属性。

参数: IN hSf1 : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN usAttrId : 扩展属性 ID;  
OUT ppExAttr : 返回扩展属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.35 获取扩展属性个数

原型: int SFL\_API SFF\_GetExtendCount(IN HSFL hSf1, OUT UINT \*pCount);

## GM/T 0055-2018

描述: a) 获取扩展属性个数。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN pCount : 返回扩展属性个数。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.36 根据序号获取扩展属性

原型: int SFL\_API SFF\_GetExtend(IN HSFL hSfl,  
IN int nIndex,  
OUT IExtendAttr \*\*ppExAttr);

描述: a) 判断序号是否合法;

b) 根据序号获取扩展属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN nIndex : 扩展属性序号, 小于等于 SFF\_GetExtendCount 获得的个数;

OUT ppExAttr : 返回扩展属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.37 删除扩展属性

原型: int SFL\_API SFF\_DelExtendAttr(IN HSFL hSfl, IN USHORT usAttrId);

描述: a) 判断扩展属性 ID 是否合法;

b) 删除扩展属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN usAttrId : 扩展属性 ID。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.38 释放扩展属性内存

原型: int SFL\_API SFF\_FreeExAttr(IN IExtendAttr \*pExAttr);

描述: a) 释放扩展属性内存。

参数: IN pExAttr : 扩展属性。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.39 添加文件操作日志

原型: int SFL\_API SFF\_AddLogAttr(IN HSFL hSfl, IN const ILogAttr \*pLog);

描述: a) 添加文件操作日志。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;

IN pLog : 文件操作日志。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.40 获取日志条数

原型: int SFL\_API SFF\_GetLogCount(IN HSFL hSfl, OUT UINT \*pCount);

描述: a) 获取日志条数。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pCount : 日志条数。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.41 获取日志

原型: int SFL\_API SFF\_GetLogAttr(IN HSFL hSfl,  
IN int nIndex,  
OUT ILogAttr \*\*ppAttr);

描述: a) 判断序号是否合法;  
b) 获取日志。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN nIndex : 日志序号, 小于等于 SFF\_GetLogCount 获取的日志条数;  
IN ppAttr : 返回日志。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.42 删除所有日志

原型: int SFL\_API SFF\_DelAllLogAttr(IN HSFL hSfl);

描述: a) 删除所有日志;  
b) 记录删除日志的操作。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

注: 记录日志删除的日志, 不会被删除

#### 9.4.3.43 释放日志内存

原型: void SFF\_FreeLogAttr(IN ILogAttr \*pAttr);

描述: a) 释放日志内存。

参数: IN pAttr : 日志属性。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.44 设置印章属性

原型: int SFL\_API SFF\_SetStampInfo(IN HSFL hSfl, IN SESeal \*pAttr);

描述: a) 设置印章属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pAttr : 印章属性。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.3.45 获取印章属性

原型: int SFL\_API SFF\_GetStampInfo(IN HSFL hSfl, IN SESeal \*\*ppAttr);

## GM/T 0055-2018

描述: a) 获取印章属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
OUT ppAttr: 返回印章属性。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.46 释放印章属性内存

原型: int SFL\_API SFF\_FreeStampInfo(IN SESeal \*pAttr);

描述: a) 释放印章属性内存。

参数: IN pAttr : 印章属性。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.47 设置指纹属性

原型: int SFL\_API SFF\_SetFingerPrintInfo(IN HSFL hSfl, IN FingerPrintAttr \*pAttr);

描述: a) 设置指纹属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN pAttr : 指纹属性。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.48 获取指纹属性

原型: int SFL\_API SFF\_GetFingerPrintInfo(IN HSFL hSfl, IN FingerPrintAttr \*\*ppAttr);

描述: a) 获取指纹属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN ppAttr : 获取指纹属性。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.49 释放指纹属性内存

原型: int SFL\_API SFF\_FreeFingerPrintInfo(IN FingerPrintAttr \*pAttr);

描述: a) 释放指纹属性内存。

参数: IN pAttr : 指纹属性。

返回值: 0 成功。  
非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

### 9.4.3.50 根据 ID 获取标签属性

原型: int SFF\_GetAttribute(IN HSFL hSfl, IN int nAttrID, OUT void \*\*pAttr);

描述: a) 判断属性 ID 是否合法;  
b) 根据 ID 获取标签属性。

参数: IN hSfl : 标签句柄, 由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回;  
IN nAttrID : 属性 ID;  
OUT pAttr : 返回属性值。

返回值: 0 成功。

非 0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

注：根据不同的 ID 返回不同的属性

#### 9.4.3.51 根据 ID 设置标签属性

原型：int SFF\_SetAttribute(IN HSFL hSfl, IN int nAttrID, OUT void \*pAttr);

描述：a) 判断属性 ID 是否合法；

b) 根据 ID 设置标签属性。

参数：IN hSfl : 标签句柄，由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回；

IN nAttrID : 属性 ID；

IN pAttr : 属性值。

返回值：0 成功。

非 0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

注：根据不同的 ID 设置不同的属性

#### 9.4.3.52 释放属性内存

原型：int SFF\_FreeAttribute(IN int nAttrID, void \*pAttr);

描述：a) 释放属性内存。

参数：IN nAttrID: 属性 ID；

IN pAttr : 属性内存。

返回值：0 成功。

非 0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

#### 9.4.3.53 设置标签大小

原型：int SFL\_API SFF\_SetLabelSize(IN HSFL hSfl, IN UINT nSize);

描述：a) 判断大小是否合适；

b) 设置标签大小。

参数：IN hSfl : 标签句柄，由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回；

IN nSize : 标签大小。

返回值：0 成功。

非 0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

注：如果未调用此接口设置标签大小，则标签大小以实际生成的为准。

#### 9.4.3.54 获取标签大小

原型：int SFL\_API SFF\_GetLabelSize(IN HSFL hSfl, OUT UINT \*pSize);

描述：a) 获取标签大小。

参数：IN hSfl : 标签句柄，由 SFF\_OpenSFL 或者 SFF\_OpenSFLB 返回；

OUT pSize : 返回之前设置的标签大小。

返回值：0 成功。

非 0 失败，返回错误码，错误码定义详见 9.1 章节表 3。

### 9.4.4 密码操作函数

#### 9.4.4.1 概述

应用系统有时不能一次性获得全部文件内容，而仅能得到部分文件内容，此时如果需要进行密码运算，则使用以下函数。

## GM/T 0055-2018

密码函数包括以下具体函数：

- a) 对称加密数据：SFF\_SymEncrypt
- b) 对称解密数据：SFF\_SymDecrypt
- c) 文件签名初始化：SFF\_SignFileInit
- d) 文件签名更新：SFF\_SignFileUpdate
- e) 文件签名结束：SFF\_SignFileFinal
- f) 文件签名验证初始化：SFF\_VerifyFileInit
- g) 文件签名验证更新：SFF\_VerifyFileUpdate
- h) 文件签名验证结束：SFF\_VerifyFileFinal

### 9.4.4.2 对称加密数据

原型：int SFL\_API SFF\_SymEncrypt(IN const HSFL hSfl,  
                                  IN BOOL bFinal,  
                                  IN const BYTE \*pSrcData,  
                                  IN UINT nSrcLen,  
                                  OUT BYTE \*pDstData,  
                                  OUT UINT \*pDstLen);

描述：a) 判断密钥句柄是否有效；  
      b) 如果是最后一段数据，则进行补码；  
      c) 加密数据。

参数：IN hSfl      : 标签句柄；  
      IN bFinal   : 是否为最后一段数据；  
      IN pSrcData  : 源数据；  
      IN nSrcLen   : 源数据长度；  
      OUT pDstData : 密文数据；  
      OUT pDstLen  : 密文数据长度。

返回值：0      成功。  
      非 0      失败，返回错误码，错误码定义详见 9.1 章节表 3。

### 9.4.4.3 对称解密数据

原型：int SFL\_API SFF\_SymDecrypt(IN const HSFL hSfl,  
                                  IN BOOL bFinal,  
                                  IN const BYTE \*pSrcData,  
                                  IN UINT nSrcLen,  
                                  OUT BYTE \*pDstData,  
                                  OUT UINT \*pDstLen);

描述：a) 解密数据；  
      b) 如果是最后一段数据，则去掉补码。

参数：IN hSfl      : 标签句柄；  
      IN bFinal   : 是否最后一段数据；  
      IN pSrcData  : 密文数据；  
      IN nSrcLen   : 密文数据长度；  
      OUT pDstData : 明文数据；

OUT pDstLen : 明文数据长度。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.4.4 签名文件初始化

原型: int SFF\_SignFileInit(IN HSFL hSfl);

描述: a) 签名文件初始化。

参数: IN hSfl : 标签句柄。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.4.5 签名文件更新

原型: int SFF\_SignFileUpdate(IN HSFL hSfl, IN BYTE \*pData, IN UINT nLen);

描述: a) 签名文件更新;

参数: IN hSfl : 标签句柄。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.4.6 签名文件结束

原型: int SFF\_SignFileFinal(IN HSFL hSfl);

描述: a) 签名文件;

b) 保存签名值到标签中。

参数: IN hSfl : 标签句柄。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.4.7 文件签名验证初始化

原型: int SFF\_VerifyFileInit(IN HSFL hSfl);

描述: a) 文件签名验证初始化。

参数: IN hSfl : 标签句柄。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.4.8 文件签名验证更新

原型: int SFF\_VerifyFileUpdate(IN HSFL hSfl, IN BYTE \*pData, IN UINT nLen);

描述: a) 文件签名验证更新。

参数: IN hSfl : 标签句柄。

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

#### 9.4.4.9 文件签名验证结束

原型: int SFL\_API SFF\_VerifyFileFinal(IN HSFL hSfl);

描述: a) 根据标签体中的签名值, 验证文件签名。

参数: IN hSfl : 标签句柄。

GM/T 0055-2018

返回值: 0 成功。

非 0 失败, 返回错误码, 错误码定义详见 9.1 章节表 3。

## 附录A (资料性附录) 数字水印

### A.1 概述

数字水印是通过在原始数据中嵌入秘密信息来证实该数据的所有权，嵌入的秘密信息称为水印，可以是一段文字、标识、序列号等。水印通常不可见或不可擦除，与原始数据（如图象、音频、视频数据等）紧密结合且隐藏其中。

数字水印可以和密码技术相结合，通过向文件嵌入已加密的数字水印，以保证文件的机密性、完整性、有效和不可抵赖性。

### A.2 数字水印操作过程

#### A.2.1 嵌入

水印信息嵌入是指将水印信息嵌入到水印载体中，如图 17 所示：

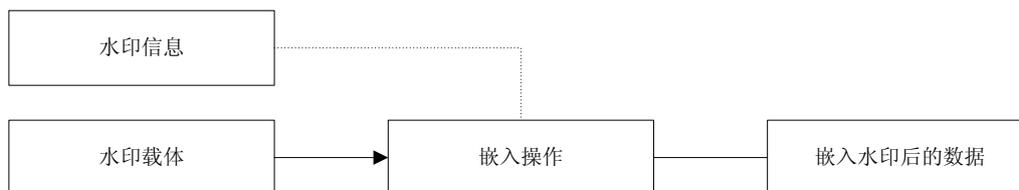


图 17 水印信息嵌入模型

图中，水印信息是指要嵌入到载体中的信息，水印载体是嵌入水印的载体。

#### A.2.2 提取

水印信息提取是指从含有水印的数据中将水印信息提取出来，如图 18 所示：

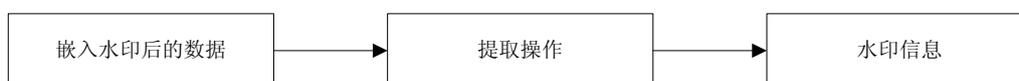


图 18 水印信息提取模型

**附录B**  
**(资料性附录)**  
**指纹识别**

**B.1 概述**

指纹识别是指利用指纹信息进行个人身份验证,在安全电子文件中可用来对文件进行操作的授权。指纹识别分为指纹录入和指纹验证两个过程:

指纹录入是从指纹传感器中采集指纹图像信息,通过图像处理和指纹识别算法处理后,提取指纹特征,制作和保存指纹特征模板。

指纹验证是从指纹传感器中采集指纹图像信息,通过图像处理和指纹识别算法处理后,提取指纹特征,与存储的指纹模板进行比对,以确认操作者的合法身份。

**B.2 指纹操作**

指纹操作包括注册、验证或识别。

注册是指制作指纹模板的过程。

验证是指提取的指纹特征与一个指定的指纹特征模板进行匹配的过程。

识别是指提取的指纹特征在注册的指纹特征模板库中找出最相似的模板过程。

---