ICS 35.030
CCS L 80

GM

中华人民共和国密码行业标准

GM/T 0033—2023
代替 GM/T 0033—2014

时间戳接口规范

Interface specifications of time stamp

2023-12-04 发布　　2024-06-01 实施

国家密码管理局　　发布

目　　次

前　言

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。

本文件代替 GM/T 0033—2014《时间戳接口规范》，与 GM/T 0033—2014 相比，除结构调整和编辑性改动外，主要技术变化如下：

a) 增加了规范性引用文件 GM/T 0028（见 7.1）、GM/T 0081（见 7.2）、GM/Z 4001（见第 3 章）、RFC 5035（见 7.1）、RFC 6211（见 7.1），删除了规范性引用文件 GM/T 0006（见 2014 年版的 5.1）、RFC 3066（见 2014 年版的 7.2），将规范性引用文件 GM/T 0010 更改为 GB/T 35275（见 7.1，2014 年版的 7.2），将规范性引用文件 RFC 3369 更改为 GB/T 31503（见 7.2，2014 年版的 7.2）；

b) 删除了术语“证书认证机构”（见 2014 年版的 3.1）、“密码杂凑算法”（见 2014 年版的 3.2）、“数字签名”（见 2014 年版的 3.3）、“SM2 算法”（见 2014 年版的 3.4）、“时间戳”（见 2014 年版的 3.5）、“时间戳系统”（见 2014 年版的 3.6）；

c) 增加了缩略语“TSA”“HTTPS”（见第 4 章，2014 年版的第 4 章）；

d) 更改了图 1（见 6.1，2014 年版的 6.1）；

e) 更改了时间戳请求格式中有关杂凑算法标识的定义，增加了时间戳响应中对算法保护域的要求（见 7.1、7.2，2014 年版的 7.1、7.2）；

f) 更改了 8.4 的名称，由“HTTP 方式”改为“HTTPS/HTTP 方式”（见 8.4，2014 年版的 8.4）；

g) 增加了“获取时间戳响应状态信息”函数 STF_GetTSStatus（见 9.6）；

h) 更改了 STF_CreateTSRequest、STF_CreateTSResponse 和 STF_VerifyTSValidity 的参数，用以支持更多的时间戳请求的属性（见 9.4、9.5、9.7，2014 年版的 9.4、9.5、9.6）；

i) 增加了时间戳接口错误代码 STF_TS_ITEM_NOT_EXIST 的定义（见附录 A）。

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。

本文件由密码行业标准化技术委员会提出并归口。

本文件起草单位：上海市数字证书认证中心有限公司、北京数字认证股份有限公司、格尔软件股份有限公司、长春吉大正元信息技术股份有限公司、北京海泰方圆科技股份有限公司、无锡江南信息安全工程技术中心、中电科网络安全科技股份有限公司、兴唐通信科技有限公司、上海颐东网络信息有限公司、万达信息股份有限公司、飞天诚信科技股份有限公司、北京华大智宝电子系统有限公司、北京握奇智能科技有限公司、山东得安信息技术有限公司、国家信息安全工程技术研究中心、国家密码管理局商用密码检测中心、三未信安科技股份有限公司。

本文件主要起草人：刘平、刘承、崔久强、李述胜、谭武征、赵丽丽、柳增寿、徐强、李元正、王妮娜、夏东山、李海杰、于华章、陈跃、胡俊义、孔凡玉、袁峰、李志伟、冯晔、王玉林、掌晓愚、李沁芸、郝波、许永欣、王腾飞、胡伯良、李国友、顾伟平。

本文件及其所代替文件的历次版本发布情况为：

——2014 年首次发布为 GM/T 0033—2014；

——本次为第一次修订。

时间戳接口规范

1 范围

本文件规定了时间戳服务接口的请求和响应消息的格式、传输方式和时间戳服务接口函数。

本文件适用于基于公钥密码应用技术体系框架内的时间戳服务相关产品的研制。

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 20520 信息安全技术 公钥基础设施 时间戳规范

GB/T 31503 信息安全技术 电子文档加密与签名消息语法

GB/T 35275 信息安全技术 SM2密码算法加密签名消息语法规范

GM/T 0019 通用密码服务接口规范

GM/T 0028 密码模块安全技术要求

GM/T 0081 SM9 密码算法加密签名消息语法规范

GM/Z 4001 密码术语

RFC 3161 Internet X.509 公钥基础设施时间戳协议(TSP)[Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)]

RFC 5035 增强安全服务(ESS)更新：增加 CertID 算法灵活性[Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility]

RFC 6211 加密消息语法(CMS)算法标识符保护属性[Cryptographic Message Syntax (CMS) Algorithm Identifier Protection Attribute]

3 术语和定义

GM/Z 4001 界定的以及下列术语和定义适用于本文件。

3.1

时间戳服务 time stamp service

时间戳系统给用户提供的颁发时间戳的服务。

注：由用户提供文件，时间戳系统给此文件签发时间戳。

4 缩略语

下列缩略语适用于本文件。

DER 可区分编码规则(Distinguished Encoding Rules)

HTTP 超文本传输协议(HyperText Transfer Protocol)

HTTPS 基于安全套接层的超文本传输协议(HyperText Transfer Protocol over Secure Sockets

Layer)

MIME　多用途网络邮件扩充协议(Multipurpose Internet Mail Extension)

PKI　公钥基础设施(Public Key Infrastructure)

SOAP　简单对象访问协议(Simple Object Access Protocol)

TSA　时间戳机构 (Time Stamp Authority)

5　标识和常量

5.1　标识定义

使用到的一般常量、标识按照 GB/T 20520 中的定义。

5.2　密码服务接口

调用的密码服务接口应符合 GM/T 0019。

5.3　时间戳服务接口常量定义

时间戳服务接口的常量定义见表 1。

表 1　时间戳服务接口的常量定义

宏描述	预定义值	说明
STF_TIME_OF_STAMP	0x00000001	签发时间
STF_CN_OF_TSSIGNER	0x00000002	签发者的通用名
STF_ORIGINAL_DATA_HASH	0x00000003	生成时间戳请求时使用的原始消息的 Hash 值
STF_CERT_OF_TSSERVER	0x00000004	时间戳服务器的证书
STF_CERTCHAIN_OF_TSSERVER	0x00000005	时间戳服务器的证书链
STF_SOURCE_OF_TIME	0x00000006	时间源的来源
STF_TIME_PRECISION	0x00000007	时间精度
STF_RESPONSE_TYPE	0x00000008	响应方式
STF_SUBJECT_COUNTRY_OF_TSSIGNER	0x00000009	签发者国家
STF_SUBJECT_ORGANIZATION_OF_TSSIGNER	0x0000000A	签发者组织
STF_SUBJECT_CITY_OF_TSSIGNER	0x0000000B	签发者城市
STF_SUBJECT_EMAIL_OF_TSSIGNER	0x0000000C	签发者联系用电子信箱
STF_POLICY_OF_TSA	0x0000000D	TSA 策略标识
STF_NONCE	0x0000000E	随机数
STF_EXTENSIONS	0x0000000F	时间戳请求/响应中的扩展域
STF_CERTSN_OF_TSSERVER	0x00000010	时间戳服务器证书序列号
	0x00000011～0x000000FF	为其他标识保留

6 时间戳服务描述

6.1 时间戳服务接口在公钥密码应用技术体系框架中的位置

本文件规定的时间戳服务接口在上层应用和典型密码应用支撑层之间及典型密码应用支撑层与基础设施安全支撑平台之间提供了时间戳请求和响应的生成、解析、验证接口的定义。

时间戳服务接口在公钥密码应用技术体系框架中的位置如图 1 所示。

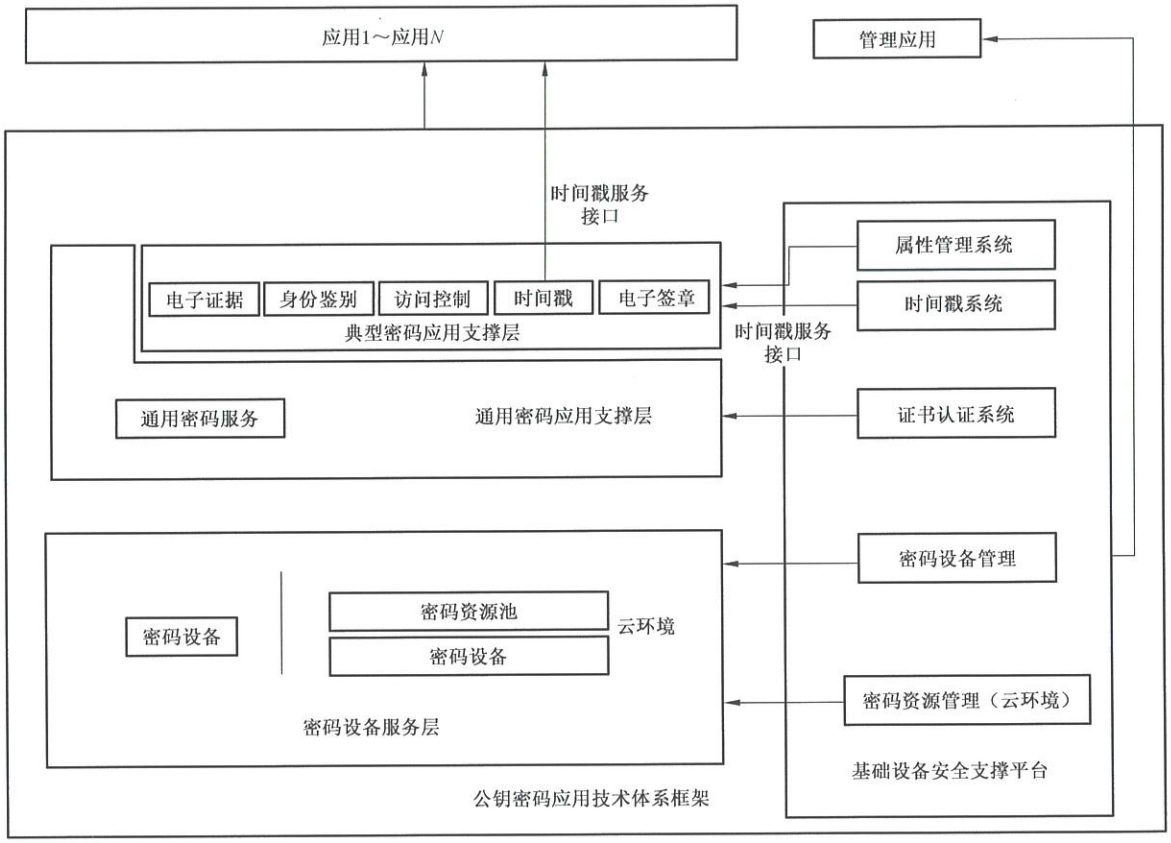

图 1 时间戳服务接口在公钥密码应用技术体系框架中的位置

6.2 时间戳服务接口的逻辑结构

时间戳服务接口位于应用层和通用密码服务接口层之间，应用通过调用时间戳服务接口获取时间戳服务；时间戳服务接口通过通用密码层提供的相应密码服务接口获取签发时间戳所需要的密钥和密码运算能力。

时间戳服务接口的逻辑结构如图 2 所示。

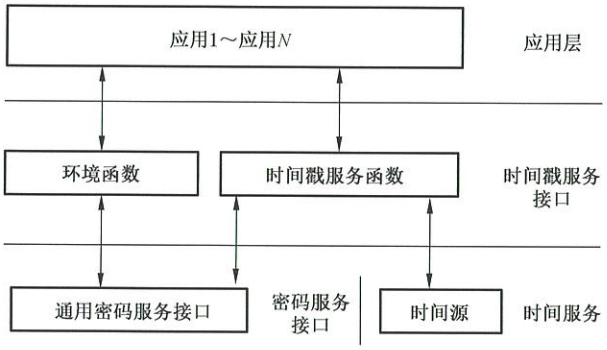

图 2　时间戳服务接口逻辑结构

7　时间戳的请求和响应格式

7.1　请求格式

定义时间戳请求的 ASN.1 数据格式如下：

```
TimeStampReq ::= SEQUENCE {
        version                 INTEGER {v2 (2)},
        messageImprint          MessageImprint,
        reqPolicy               TSAPolicyId             OPTIONAL,
        nonce                   INTEGER                 OPTIONAL,
        certReq                 BOOLEAN                 DEFAULT  FALSE,
        extensions              [0]  IMPLICIT Extension         OPTIONAL
        }
```

其中：

——version 域表示时间戳申请消息格式的版本号。

——messageImprint 域应包含需要加盖时间戳的数据的摘要值。该摘要值的类型是 OctetString，长度是具体密码杂凑算法的结果长度。具体格式为：

```
MessageImprint ::= SEQUENCE {
        hashAlgorithm           AlgorithmIdentifier,
        hashedMessage           OCTET STRING
        }
```

在 hashAlgorithm 域中表示的算法应使用 GM/T 0028 中的算法，并在返回消息中设置‘badAlg’的 pkiStatusInfo 结构。

——reqPolicy 域表示安全策略。安全策略由 TSA 提供，用户可选择需要的安全策略设置该域。reqPolicy 的类型是 TSAPolicyId，TSAPolicyId 按照 RFC 3161 中的定义。

——nonce 域是一个随机数，用于在没有可靠的本地时钟的情况下检验响应消息的合法性并防止重放攻击。

——certReq 域用于请求 TSA 公钥证书。如果为 true，则 TSA 应在响应消息中给出其公钥签名证书。SignerInfo 及 Attribute 结构按照 GB/T 35275 中的定义，在时间戳响应消息中包含 SignerInfo 结构，SignedInfo 结构中包含多个被签名的 Attribute 结构，其中应存在一项 Attribute 结构，其值类型为 CMSAlgorithmProtection，CMSAlgorithmProtection 按照 RFC 6211 中的定义。当 certReq 值为 true 时，应存在一项 Attribute 结构，其值类型为 SigningCer-

tificateV2，SigningCertificateV2 按照 RFC 5035 中的定义。TSA 的公钥签名证书应由响应消息中 SigningCertificateV2 部分包含的 ESSCertIDv2 域唯一确定，ESSCertIDv2 按照 RFC 5035 中的定义。TSA 的公钥签名证书存放在响应消息中 SignedData 部分的 Certificates 域中。

——extensions 为扩展域，用于为申请消息添加额外信息。对于其中任何一个扩展，无论其是否为关键扩展，当它在请求消息中出现且又无法被 TSA 识别时，TSA 应不为该请求生成时间戳，并且返回一个失败消息(unacceptedExtension)。

时间戳请求消息不必给出请求方的身份标识。如果 TSA 需要鉴别请求方的身份，应对请求方进行身份鉴别，采用何种身份鉴别机制不在本文件中规定。

7.2 响应格式

TSA 在收到申请消息后，应返回响应消息。

定义时间戳响应消息的 ASN.1 数据格式如下：

```
TimeStampResp ::= SEQUENCE {
    status                  PKIStatusInfo,
    timeStampToken          TimeStampToken          OPTIONAL
    }
```

其中：

```
PKIStatusInfo ::= SEQUENCE {
    status                  PKIStatus,
    statusString            PKIFreeText             OPTIONAL,
    failInfo                PKIFailureInfo          OPTIONAL
    }
```

其中：

```
PKIStatus ::= INTEGER{
    granted                 (0),
    grantedWithMods         (1),
    rejection               (2),
    waiting                 (3),
    revocationWarning       (4),
    revocationNotification  (5)
}
```

PKIStatusInfo 中的 status 值为 0 或者 1 时，响应消息中应包含 TimeStampToken，否则响应消息中应不包含 TimeStampToken。

申请失败时用 statusString 给出一个说明失败原因的字符串。

failInfo 用来说明时间戳请求被拒绝的具体原因，其值如下：

```
PKIFailureInfo ::= BIT STRING{
badAlg                      (0),   —申请使用了不支持的算法
badRequest                  (2),   —非法的申请
badDataFormat               (5),   —数据格式错误
timeNotAvailable            (14),  —TSA 的可信时间源出现问题
unacceptedPolicy            (15),  —不支持申请消息中声明的策略
unacceptedExtension         (16),  —申请消息中包括了不支持的扩展
```

```
addInfoNotAvailble          (17),  —有不理解或不可用的附加信息
systemFailure               (25)   —系统内部错误
}
```

failInfo 不能有除 PKIFailureInfo 外的其他值，请求方如果收到一个未能识别的值，必须报告错误。

TimeStampToken 是一个 ContentInfo 结构。当公钥算法为 SM2 时，ContentInfo 结构按照 GB/T 35275 中的定义；当公钥算法为 SM9 时，ContentInfo 结构按照 GM/T 0081 中的定义；当公钥算法为 RSA 时，ContentInfo 结构按照 GB/T 31503 中的定义。该结构中 content type 是一个 SignedData content type，具体值如下：

```
TimeStampToken ::= ContentInfo
    —contentType 为 id-signedData
    —content 为 SignedData
```

当公钥算法为 SM2 时，SignedData 结构按照 GB/T 35275 中的定义；当公钥算法为 SM9 时，SignedData 结构按照 GM/T 0081 中的定义；当公钥算法为 RSA 时，SignedData 结构按照 GB/T 31503 中的定义。TimeStampToken 应只含有 TSA 签名，TSA 证书的证书标识符 ESSCertIDv2 应包含在 SigningCertificateV2 属性中，SigningCertificateV2 属性应是 SignedData 中 SignerInfo 部分的一个子项。

关于 TSTInfo 的具体定义如下：

```
TSTInfo ::= SEQUENCE{
    version          INTEGER{v2(2)},
    policy           TSAPolicyId,
    messageImprint   MessageImprint,
    serialNumber     INTEGER,
    genTime          GeneralizedTime,
    accuracy         Accuracy                  OPTIONAL,
    ordering         BOOLEAN                   DEFAULT  FALSE,
    nonce            INTEGER                   OPTIONAL,
    tsa              [0]  GeneralName          OPTIONAL,
    extensions       [1]  IMPLICIT Extensions  OPTIONAL
    }
```

TSTInfo 的各项解释如下。

——version 域指明时间戳的版本号。

——policy 域指明响应消息根据 TSA 的哪个策略生成的。如果类似的域(reqPolicy)出现在 TimeStampReq 中，该域应有相同的值，否则 status 应返回 unacceptedPolicy，policy 可包含但不仅限于下列类型的信息：

- 该时间戳的使用条件；
- 时间戳日志的有效性，可用于后续增强时间戳的可信度。

——messageImprint 应同 TimeStampReq 中类似的域有相同的值，前提是杂凑值的长度与 hashAlgorithm 标记的算法预期的长度相同。

——serialNumber 域是 TSA 分配的一个整数。一个特定的 TSA 发出的每一个时间戳，serialNumber 应是唯一的(即 TSA 的名字和序列号可以确定一个时间戳标志)。应注意的是，即使经历一个可能的服务中断(例如崩溃)后，该特性也应保留。

——genTime 是 TSA 创建时间戳的时间。应用 UTC 时间表示，以减少使用本地时区造成的混乱。

——accuracy 表示时间可能出现的最大误差，genTime 加上 accuracy 的值，可以求得 TSA 创建这个时间戳的时间上限；同理，减去 accuracy 是 TSA 创建时间戳的时间下限，具体定义如下。

```
Accuracy ::= SEQUENCE{
    seconds   INTEGER                  OPTIONAL,   —s
    millis    [0]  INTEGER (1..999)    OPTIONAL,   —ms
    micros    [1]  INTEGER (1..999)    OPTIONAL    —μs
    }
```

如果 seconds、millis 或者 micros 未出现，则未出现域的值应被赋为 0。当 accuracy 这个可选项不出现时，精确度可以从其他途径得到，例如 TSAPolicyId。

——ordering 表示时间戳排序条件。若 ordering 域不出现，或者 ordering 域出现但被置为 false，则 genTime 域只表示 TSA 创建时间戳的时间。在这种情况下，当两个时间戳中第一个的 genTime 与第二个的 genTime 之差大于这两个 genTime 的精确度的和，同一个 TSA 或者不同的 TSA 签发的时间戳标志才有可能排序。如果 ordering 域出现并被置为 true，同一个 TSA 发的每一个时间戳都可以依据 genTime 排序，而不必考虑 genTime 精确度。

——nonce 域若在 TimeStampReq 中出现，则此处也应出现，其值应等于 TimeStampReq 中的值。

——tsa 域是为鉴别 TSA 名字提供的一个线索。若出现应与验证时间戳的证书里的 subject names 中的一个相同。

——extensions(扩展)域是为将来增加额外的信息而采用的一种通常的做法。特殊的扩展类型可以由组织或者团体自行定义并声明注册。

8 时间戳服务接口的通信方式及格式

8.1 电子邮件方式

采用电子邮件方式时，用户使用电子邮件向一个 TSA 指定的电子邮件地址发送时间戳申请，TSA 将颁发的时间戳通过电子邮件返回给用户。申请与颁发应使用以下 MIME 对象。

a) 申请消息

Content-Type:application/timestamp-query

Content-Transfer-Encoding:base64

申请消息的 DER 编码，再用 base64 编码。

b) 响应消息

Content-Type:application/timestamp-reply

Content-Transfer-Encoding:base64

响应消息的 DER 编码，再用 base64 编码。

8.2 文件方式

采用文件方式时，用户将申请消息存储在一个扩展名为.tsq 的文件中传送给 TSA，TSA 将产生的响应消息保存在一个扩展名为.tsr 的文件中返回给用户。请求文件和响应文件应只包含消息的 DER 编码。

8.3 Socket 方式

采用 Socket 方式时，用户与 TSA 应通过安全的 socket 传输申请消息和响应消息。通信的消息格式应符合 RFC 3161 3.3 中的格式和定义。

8.4 HTTPS/HTTP 方式

采用 HTTPS/HTTP 方式时，用户通过 HTTPS/HTTP 协议将申请消息发送给 TSA，TSA 通过 HTTPS/HTTP 协议返回响应消息。

使用 HTTPS/HTTP 协议传输时间戳申请和响应消息使用的 MIME 对象如下。

a) 申请消息

Content-Type：application/timestamp-query

申请消息的 DER 编码。

b) 响应消息

Content-Type：application/timestamp-reply

响应消息的 DER 编码。

8.5 SOAP 方式

采用 SOAP 方式时，TSA 应以同样的格式返回响应消息。

SOAP 规范传输格式的内容如下。

a) 申请消息

〈soap：Body〉

〈timestamp-query〉

申请消息的 DER 编码，再用 base64 编码。

〈/timestamp-query〉

〈/soap：Body〉

b) 响应消息

〈soap：Body〉

〈timestamp-reply〉

响应消息的 DER 编码，再用 base64 编码。

〈/timestamp-reply〉

〈/soap：Body〉

9 时间戳服务接口组成和功能说明

9.1 接口组成

规定的时间戳服务接口共提供 8 个与时间戳服务有关的函数：

——环境函数(用于创建和释放应用于时间戳接口服务的上下文的特定属性值)

初始化环境	STF_InitEnvironment
清除环境	STF_ClearEnvironment

——时间戳服务函数

生成时间戳请求	STF_CreateTSRequest
生成时间戳应答	STF_CreateTSResponse
获取时间戳响应的状态信息	STF_GetTSStatus
验证时间戳有效性	STF_VerifyTSValidity
获取时间戳主要信息	STF_GetTSInfo
解析时间戳详细信息	STF_GetTSDetail

上述函数的返回值应符合附录 A 中的要求。

上述函数的调用示例见附录 B。

对于时间戳请求及响应使用的通信方式基于第 8 章，通信接口的定义不在本文件的范围内。

9.2 初始化环境函数

原型： SGD_UINT32 STF_InitEnvironment(void * * phTSHandle)

描述： 建立时间戳环境句柄。

参数： phTSHandle[OUT]：时间戳环境句柄指针

返回值： STF_TS_OK：成功

其他：失败

9.3 清除环境函数

原型： SGD_UINT32 STF_ ClearEnvironment(void * hTSHandle)

描述： 清除时间戳环境句柄。

参数： hTSHandle[IN]：时间戳环境句柄

返回值： STF_TS_OK：成功

其他：失败

9.4 生成时间戳请求

```
原型： SGD_UINT32 STF_CreateTSRequest(void *          hTSHandle,
                                      SGD_UINT8    *pucInData,
                                      SGD_UINT32   uiInDataLength,
                                      SGD_UINT32   uiReqType,
                                      SGD_UINT8    *pucTSExt,
                                      SGD_UINT32   uiTSExtLength,
                                      SGD_UINT8    *pucNonce,
                                      SGD_UINT32   uiNonceLength,
                                      SGD_UINT32   uiHashAlgID,
                                      SGD_UINT8    *pucReqPolicy,
                                      SGD_UINT32   uiReqPolicyLength,
                                      SGD_UINT8    *pucTSRequest,
                                      SGD_UINT32   *puiTSRequestLength);
```

描述： 用指定算法对时间戳请求信息 indata 进行密码杂凑运算，生成时间戳请求包。

参数： hTSHandle [IN]：时间戳环境句柄

pucInData [IN]：需要加盖时间戳的用户信息

uiInDataLength [IN]：用户信息的长度，以字节为单位

uiReqType [IN]：请求的时间戳服务类型

pucTSExt [IN]：时间戳请求包的其他扩展，DER 编码格式

uiTSExtLength[IN]：时间戳请求包扩展的长度，以字节为单位

uiHashAlgID [IN]：密码杂凑算法标识

pucNonce [IN]：随机数 Nonce

uiNonceLength [IN]：随机数 Nonce 的长度

pucReqPolicy [IN]：TSA 安全策略标识，DER 编码格式

uiReqPolicyLength [IN]：TSA 安全策略标识长度，以字节为单位

pucTSRequest [OUT]:时间戳请求

puiTSRequestLength [IN/OUT]:时间戳请求的长度,以字节为单位

返回值: STF_TS_OK:成功

其他:失败

备注: uiReqType: 0 代表时间戳响应应包含时间戳服务器的证书,1 代表时间戳响应不包含时间戳服务器的证书。

pucNonce:在时间戳请求 TimeStampReq 中 Nonce 是可选的,当时间戳请求中不包含 Nonce 时,该变量取值为空指针 NULL,uiNonceLength 取值为 0。

pucReqPolicy :在时间戳请求 TimeStampReq 中 reqPolicy 是可选的,当时间戳请求中不包含 reqPolicy 时,该变量取值为空指针 NULL,uiReqPolicyLength 取值为 0。

puiTSRequestLength[IN/OUT]: 入口值为指定的用于存放时间戳请求的字符数组的最大长度,出口值为时间戳请求的实际长度。

9.5 生成时间戳响应

原型:
```
SGD_UINT32 STF_CreateTSResponse(void *          hTSHandle,
                                SGD_UINT8       * pucTSRequest,
                                SGD_UINT32      uiTSRequestLength,
                                SGD_UINT8       * pucSerialNumber,
                                SGD_UINT32      uiSerialNumberLength,
                                SGD_UINT32      uiSignatureAlgID,
                                SGD_UINT8       * pucTSCert,
                                SGD_UINT32      uiTSCertLength,
                                SGD_UINT8       * pucTSResponse,
                                SGD_UINT32      * puiTSResponseLength);
```

描述: 根据时间戳请求包生成时间戳响应包。

参数: hTSHandle [IN]:时间戳环境句柄

pucTSRequest [IN]:时间戳请求

uiTSRequestLength [IN]:时间戳请求的长度

pucSerialNumber[IN]: TSA 为生成的时间戳响应分配的序列号

uiSerialNumberLength[IN]: TSA 为生成的时间戳响应分配的序列号的长度

uiSignatureAlgID [IN]:签名算法标识

pucTSCert [IN]:TSA 的签名证书

uiTSCertLength [IN]:TSA 签名证书的长度

pucTSResponse [OUT]:时间戳响应

puiTSResponseLength [IN/OUT]:时间戳响应的长度

返回值: STF_TS_OK:成功

其他:失败

备注: pucTSCert:表示 TSA 的签名证书,DER 编码格式。在时间戳请求中可能会要求 TSA 在响应中包含 TSA 的签名证书,此外在计算时间戳响应中包含的 ESSCertIDv2 时,也要用到 TSA 的证书。

puiTSResponseLength [IN/OUT]:入口值为指定的用于存放时间戳的字符数组的最大长度,出口值为时间戳的实际长度。

9.6 获取时间戳响应的状态信息

原型：SGD_UINT32 STF_GetTSStatus(void * hTSHandle,
SGD_UINT8 * pucTSResponse,
SGD_UINT32 uiTSResponseLength,
SGD_UINT8 * pucPKIStatus);

描述：获取时间戳响应中包含的状态代码的值。

参数：hTSHandle [IN]:时间戳环境句柄
pucTSResponse [IN]:获取的时间戳响应
uiTSResponseLength [IN]:时间戳响应的长度
pucPKIStatus [OUT]:时间戳响应中包含的状态信息代码的值

返回值：STF_TS_OK:成功
其他:失败

备注：时间戳响应 TimeStampResp 中包含状态信息代码 PKIStatus,该变量可能的取值范围为 0～5。当该变量取值为 0 或 1 时,响应消息中才会出现 TimeStampToken,这时才能从时间戳响应中提取出时间,才有必要验证时间戳响应是否有效。当 PKIStatus 取其他值时,表示时间戳服务器未能正常响应收到的时间戳请求,时间戳响应中不包含 TimeStampToken,此时时间戳响应中不包含有效时间。

9.7 验证时间戳有效性

原型：SGD_UINT32 STF_VerifyTSValidity(void * hTSHandle,
SGD_UINT8 * pucTSResponse,
SGD_UINT32 uiTSResponseLength,
SGD_UINT8 * pucInDataHash,
SGD_UINT32 uiInDataHashLength,
SGD_UINT32 uiHashAlgID,
SGD_UINT8 * pucTSCert,
SGD_UINT32 uiTSCertLength);

描述：验证时间戳响应是否有效。

参数：hTSHandle [IN]:时间戳环境句柄
pucTSResponse [IN]:获取的时间戳响应
uiTSResponseLength [IN]:时间戳响应的长度
pucInDataHash [IN]:生成时间戳请求时使用的原始数据的杂凑值
uiInDataHashLength [IN]:生成时间戳请求时使用的原始数据的杂凑值长度
uiHashAlgID [IN]:生成时间戳请求时使用的密码杂凑算法
pucTSCert [IN]:TSA 的证书,DER 编码格式
uiTSCertLength [IN]:TSA 证书的长度

返回值：STF_TS_OK:成功
其他:失败

备注：该函数验证时间戳响应是否有效。调用本函数的前提是时间戳响应中包含 TimeStampToken。对于不包含时间戳服务器证书的响应,需要指定时间戳服务器的证书才能进行验证;对于包含时间戳服务器证书的响应,可以把入口的证书参数置为空,使用响应中自带的证书进行验证,否则将使用指定的证书进行验证,即指定的证书优先于自带的

证书。

验证时间戳时，需要计算原始数据的杂凑值，然后与时间戳响应中 TSTInfo 部分 messageImprint 里包含的杂凑值相比较，如果二者不一致则验证失败；还需比较生成时间戳时使用的密码杂凑算法与时间戳响应中 TSTInfo 部分 messageImprint 里包含的密码杂凑算法标识是否一致。

9.8 获取时间戳主要信息

原型：
```
SGD_UINT32 STF_GetTSInfo(void *          hTSHandle,
                         SGD_UINT8   * pucTSResponse,
                         SGD_UINT32  uiTSResponseLength,
                         SGD_UINT8   * pucIssuerName,
                         SGD_UINT32  * puiIssuerNameLength,
                         SGD_UINT8   * pucTime,
                         SGD_UINT32  * puiTimeLength);
```

描述：获取时间戳的主要信息。

参数：hTSHandle [IN]：时间戳环境句柄

pucTSResponse [IN]：获取的时间戳响应

uiTSResponseLength [IN]：时间戳响应的长度

pucIssuerName [OUT]：TSA 的通用名

puiIssuerNameLength [IN/OUT]：TSA 通用名的长度

pucTime [OUT]：时间戳标注的时间值

puiTimeLength [IN/OUT]：时间戳标注的时间值长度

返回值：STF_TS_OK：成功

其他：失败

备注：该函数解析时间戳的主要信息，包括 TSA 的通用名和时间戳的签发时间。调用本函数的前提是时间戳响应中包含 TimeStampToken。

puiIssuerNameLength [IN/OUT]：入口值为指定的用于存放签发者名称的字符数组的最大长度，出口值为签发者名称的实际长度。

puiTimeLength [IN/OUT]：入口值为指定的用于存放时间戳标注时间值的字符数组的最大长度，出口值为时间戳标注的时间值的实际长度。

9.9 解析时间戳详细信息

原型：
```
SGD_UINT32 STF_GetTSDetail(void *          hTSHandle,
                           SGD_UINT8     * pucTSResponse,
                           SGD_UINT32    uiTSResponseLength,
                           SGD_UINT32    uiItemNumber,
                           SGD_UINT8     * pucItemValue,
                           SGD_UINT32    * puiItemValueLength);
```

描述：解析时间戳的详细信息。

参数：hTSHandle[IN]：时间戳环境句柄

pucTSResponse [IN]：获取的时间戳响应

uiTSResponseLength [IN]：时间戳响应的长度

uiItemNumber [IN]：指定获取时间戳响应中特定域的编号

pucItemValue [OUT]:解析得到的时间戳响应中特定域的相关信息

puiItemValueLength [IN/OUT]:时间戳响应中特定域的对应长度

返回值： STF_TS_OK:成功

其他:失败

备注： 该函数解析时间戳的详细信息,调用本函数的前提是时间戳响应中包含TimeStampToken。uiItemNumber 定义如下:

STF_TIME_OF_STAMP:签发时间;

STF_CN_OF_TSSIGNER:签发者的通用名;

STF_ORIGINAL_DATA_HASH:生成时间戳请求时使用的原始消息的杂凑值;

STF_CERT_OF_TSSERVER:时间戳服务器证书;

STF_CERTSN_OF_TSSERVER:时间戳服务器证书序列号;

STF_CERTCHAIN_OF_TSSERVER:时间戳服务器的证书链;

STF_SOURCE_OF_TIME:时间源的来源;

STF_TIME_PRECISION:时间精度;

STF_RESPONSE_TYPE:响应方式;

STF_SUBJECT_COUNTRY_OF_TSSIGNER:签发者国家;

STF_SUBJECT_ORGANIZATION_OF_TSSIGNER:签发者组织;

STF_SUBJECT_CITY_OF_TSSIGNER:签发者城市;

STF_SUBJECT_EMAIL_OF_TSSIGNER:签发者联系用电子信箱;

STF_NONCE:nonce 值;

STF_POLICY_OF_TSA:TSA 策略标识;

STF_EXTENSIONS:扩展域。

具体信息项的宏定义见 5.3。

当时间戳响应不包含时间戳服务器证书,调用该函数获取时间戳服务器证书、证书链、签发者的国家、组织、城市和电子信箱信息时,均返回空值。

pulItemValueLength [IN/OUT]: 入口时指定用于存放时间戳相关信息的字符数组的最大长度,出口时为时间戳相关信息的实际长度。

附 录 A
（规范性）
时间戳服务接口错误代码定义

表 A.1 给出了时间戳服务接口错误代码定义。

表 A.1 时间戳服务接口错误代码定义

宏描述	预定义值	说明
STF_TS_OK	0x0	正常返回
STF_TS_ERROR_BASE	0x04000000	错误码基础值
STF_TS_INDATA_TOOLONG	0x04000001	输入的用户信息超出规定范围
STF_TS_NOT_ENOUGH_MEMORY	0x04000002	分配给 tsrequest 的内存空间不够
STF_TS_SERVER_ERROR	0x04000003	找不到服务器或超时响应
STF_TS_MALFORMAT	0x04000004	时间戳格式错误
STF_TS_INVALID_ITEM	0x04000005	输入域编号无效
STF_TS_INVALID_SIGNATURE	0x04000006	签名无效
STF_TS_ INVALID_ALG	0x04000007	申请使用了不支持的算法
STF_TS_ INVALID_REQUEST	0x04000008	非法的申请
STF_TS_INVALID_DATAFORMAT	0x04000009	数据格式错误
STF_TS_TIME_NOT_AVAILABLE	0x0400000A	TSA 的可信时间源出现问题
STF_TS_UNACCEPTED_POLICY	0x0400000B	不支持申请消息中声明的策略
STF_TS_UNACCEPTED_EXTENSION	0x0400000C	申请消息中包括了不支持的扩展
STF_TS_ADDINFO_NOT_AVAILABLE	0x0400000D	有不理解或不可用的附加信息
STF_TS_SYSTEM_FAILURE	0x0400000E	系统内部错误
STF_TS_ITEM_NOT_EXIST	0x0400000F	输入域的值不存在
……	0x04000010～0x040000FF	预留

附 录 B
（资料性）
时间戳服务接口应用示例

以下例子说明函数的一般使用方法和次序。假设使用 STF 的时间戳服务，对一段信息加盖时间戳；并对获取到的时间戳信息进行解析和验证。

时间戳请求方主要程序代码如下：

```
#define SGD_SM3 (1)
void* handle;
SGD_UINT32 retcode;
SGD_UINT8 message[128]="hiShecaLcIl5uurgooddone";
//要加盖时间戳的原始信息
SGD_UINT8 tsrequest[5120], tsresponse[5120];
SGD_UINT32 tsrequestlength=5120, tsresponselength=5120, tsresponsestatus;
SGD_UINT8 tscert[4096];
SGD_UINT32 tscertlen=4096;
SGD_UINT8 itemvalue[64];
SGD_UINT32 itemvaluelength=64;
SGD_UINT8 datahash[32];
SGD_UINT32 datahashlength=32;

retcode = STF_InitEnvironment(&handle);
if(retcode! = 0){
    /*错误处理*/
    return;
}
/*生成时间戳请求*/
retcode = STF_CreateTSRequest(handele, message, strlen(message), 1, NULL, 0,
SGD_SM3,"12345678",8, NULL, 0, tsrequest, &tsrequestlength);
if(retcode! = 0){
    /*错误处理*/
    STF_ClearEnvironment(handle);
    return;
}
/*采用某种通信方式将 tsrequest 发送到时间戳服务器端，并接收时间戳服务器的响应，保存到 tsresponse*/
/*获取并检查时间戳响应的状态信息*/
retcode = STF_GetTSStatus(handle, tsresponse, tsresponselength, &tsresponsestatus);
if(retcode! = 0){
    /*错误处理*/
    STF_ClearEnvironment(handle);
return;
```

```
    }
    if (tsresponsestatus ! = 0 && tsresponsestatus ! = 1) {
        /* 错误处理 */
        STF_ClearEnvironment(handle);
        return;
    }
    /* 验证时间戳 */
    /* 待验证 HASH 值读到 datahash 里 */
    /* 时间戳服务器证书读到 tscert 里 */
    retcode = STF_VerifyTSValidity(handle, tsresponse, tsresponselength, datahash, datahash-
length, SGD_SM3, tscert, tscertlen);
    if (retcode ! = 0) {
        /* 错误处理 */
        STF_ClearEnvironment(handle);
        return;
    }
    /* 显示时间戳信息 */
    retcode = STF_GetTSDetail(handle, tsresponse, tsresponselength, STF_TIME_OF_STAMP, item-
value, &itemvaluelength);
    if (retcode ! = 0) {
        /* 错误处理 */
        STF_ClearEnvironment(handle);
        return;
    }
      STF_ClearEnvironment(handle);
```

时间戳服务方主要程序代码如下：

```
#define SGD_SM2_1 (0x00020200)
void * handle;
SGD_UINT32 retcode;
SGD_UINT8 tsrequest[5120], tsresponse[5120], serialnumber[8];
SGD_UINT32 tsrequestlength=5120, tsresponselength=5120;
SGD_UINT8 tscert[4096];
SGD_UINT32 tscertlen=4096;

retcode = STF_InitEnvironment(&handle);
if(retcode! = 0){
    /* 错误处理 */
    return;
}
/* 采用某种通信方式接受客户端的时间戳请求,保存到 tsrequest */
/* 生成时间戳响应 */
```

```
retcode = STF_CreateTSResponse(handle, tsrequest, tsrequestlength, serialnumber, sizeof(serialnumber), SGD_SM2_1, NULL,0, tsresponse, &tsresponselength);
if(retcode ! = 0){
    /*错误处理*/
    STF_ClearEnvironment(handle);
    return;
}
/*采用某种通信方式将时间戳响应发送到客户端*/
STF_ClearEnvironment(handle);
```