ICS 35.040
L 80
备案号:44634—2014

GM

中华人民共和国密码行业标准

GM/T 0033—2014

时间戳接口规范

Interface specifications of time stamp

2014-02-13 发布　　2014-02-13 实施

国家密码管理局　　发布

中华人民共和国密码
行业标准
时间戳接口规范
GM/T 0033—2014

*

中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100029)
北京市西城区三里河北街16号(100045)

网址 www.spc.net.cn
总编室:(010)64275323 发行中心:(010)51780235
读者服务部:(010)68523946

中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销

*

开本 880×1230 1/16 印张 1.25 字数 32 千字
2014年5月第一版 2014年5月第一次印刷

*

书号:155066·2-27011 定价 24.00 元

GM/T 0033-2014

目　　次

前　言

本标准按照 GB/T 1.1—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。

本标准起草单位：上海市数字证书认证中心有限公司、北京市数字证书认证中心有限公司、上海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、北京海泰方圆科技有限公司、无锡江南信息安全工程技术中心、成都卫士通信息产业股份有限公司、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。

本标准起草人：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。

时间戳接口规范

1 范围

本标准规定了面向应用系统和时间戳系统的时间戳服务接口，包括时间戳请求和响应消息的格式、传输方式和时间戳服务接口函数。

本标准适用于规范基于公钥密码基础设施应用技术体系框架内的时间戳服务相关产品，以及时间戳服务的集成和应用。

2 规范性引用文件

下列文件对于本文件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。

GB/T 20520 信息安全技术 公钥基础设施 时间戳规范

GM/T 0006 密码应用标识规范

GM/T 0010 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

RFC 3066 Tags for the Identification of Languages

RFC 3161 Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)

RFC 3369 Cryptographic Message Syntax (CMS)

3 术语和定义

下列术语和定义适用于本文件。

3.1

证书认证机构 certification authority;CA

对数字证书进行全生命周期管理的实体。也称为电子认证服务机构。

3.2

密码杂凑算法 cryptographic hash algorithm

又称杂凑算法、密码散列算法或哈希算法。该算法将一个任意长的比特串映射到一个固定长的比特串，且满足下列三个特性：

(1)为一个给定的输出找出能映射到该输出的一个输入是计算上困难的；

(2)为一个给定的输入找出能映射到同一个输出的另一个输入是计算上困难的。

(3)要发现不同的输入映射到同一输出是计算上困难的。

3.3

数字签名 digital signature

签名者使用私钥对待签名数据的杂凑值做密码运算得到的结果，该结果只能用签名者的公钥进行验证，用于确认待签名数据的完整性、签名者身份的真实性和签名行为的抗抵赖性。

3.4

SM2 算法　SM2 algorithm

一种椭圆曲线公钥密码算法，其密钥长度为 256 比特。

3.5

时间戳　time stamp；TS

对时间和其他待签名数据进行签名得到的数据，用于表明数据的时间属性。

3.6

时间戳系统　time stamp authority system

用来产生和管理时间戳的管理系统。

3.7

时间戳服务　time stamp service

时间戳系统给用户提供的颁发时间戳服务，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(Hyper Text Transfer Protocol)

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)

5　标识和数据结构

5.1　标识定义

本标准中使用到的一般常量、标识、证书解析项等的定义参见 GM/T 0006。

5.2　密码服务接口

本标准中调用的密码服务接口遵循 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
# define STF_TIME_OF_STAMP	0x00000001	签发时间
# define STF_CN_OF_TSSIGNER	0x00000002	签发者的通用名
# define STF_ORINGINAL_DATA	0x00000003	时间戳请求的原始信息
# define STF_CERT_OF_TSSERVER	0x00000004	时间戳服务器的证书
# define STF_CERTCHAIN_OF_TSSERVER	0x00000005	时间戳服务器的证书链
# define STF_SOURCE_OF_TIME	0x00000006	时间源的来源

表 1（续）

宏　描　述	预定义值	说　　明
# define STF_TIME_PRECISION	0x00000007	时间精度
# define STF_RESPONSE_TYPE	0x00000008	响应方式
# define STF_SUBJECT_COUNTRY_OF_TSSIGNER	0x00000009	签发者国家
# define STF_SUBJECT_ORGNIZATION_OF_TSSIGNER	0x0000000A	签发者组织
# define STF_SUBJECT_CITY_OF_TSSIGNER	0x0000000B	签发者城市
# define STF_SUBJECT_EMAIL_OF_TSSIGNER	0x0000000C	签发者联系用电子信箱
	0x0000000D～0x000000FF	为其他标识保留

6　时间戳服务描述

6.1　时间戳服务在公钥密码基础设施应用技术体系框架中的位置

时间戳服务接口规范依托于 GM/T 0019 和 GB/T 20520。本标准规定的时间戳服务接口向上层应用和典型密码服务层其他组成部分提供与时间戳系统无关的时间戳加盖、验证等时间认证服务。

时间戳服务在公钥密码基础设施应用技术体系框架中的位置如图 1 所示。

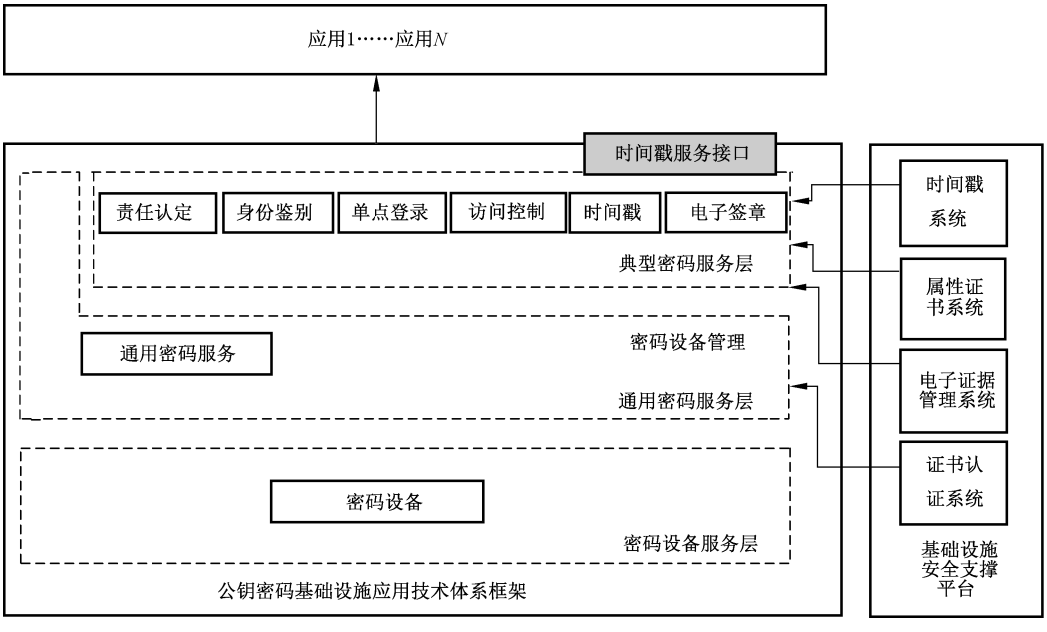

图 1　时间戳服务在公钥密码基础设施应用技术体系中的位置

6.2　时间戳服务接口的逻辑结构

时间戳服务接口作为业务应用和时间戳系统之间的中间件，位于上层应用和通用密码服务层接口之间，上层应用通过调用时间戳服务接口实现具体时间戳应用；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实现具体的密码运算和密钥使用。

时间戳服务接口的逻辑结构如图 2 所示。

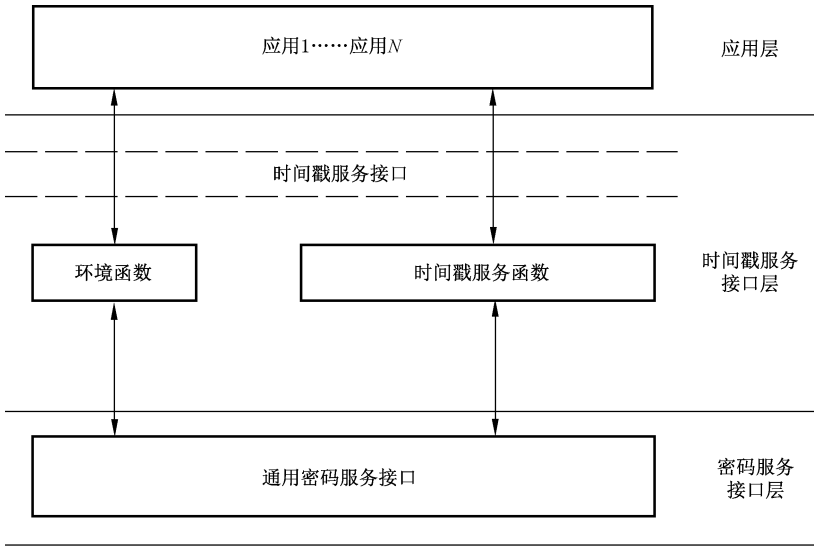

图 2　时间戳服务接口逻辑结构

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 {v1(1)},
    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 域应该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 域中表示的算法是国家密码管理部门批准的密码杂凑算法。如果 TSA 不识别给出的密码杂凑算法或这个密码杂凑算法不符合国家密码管理部门的相关规定，那么 TSA 应拒绝提供时间戳服务，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 的定义为：

```
TSAPolicyId ::= OBJECT IDENTIFIER
```

——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证书，该证书由响应消息中 SigningCertificate 属性 ESSCertID 指出，证书存放在响应消息中 SignedData 结构的 Certificates 域中。

——extension 为扩展域，用于给申请消息添加额外信息。对于一个扩展，无论其是否为关键扩展，只要它在请求消息中出现，且又无法被 TSA 识别，则 TSA 应该不生成时间戳且返回一个失败消息(unacceptedExtension)。

时间戳请求消息不需要给出请求方的身份标识。如果 TSA 需要鉴别请求方的身份，应另外进行双向身份鉴别，双向身份鉴别的实现不在本标准中规定。

7.2 响应格式

TSA 在收到申请消息后，无论申请成功还是失败，都要给请求方返回一个响应消息。该响应消息或者是正确的时间戳，或者是包含了失败信息的时间戳。

定义时间戳响应消息的 ASN.1 数据格式如下：

```
TimeStampResq ::= 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，响应消息中的 TimeStampToken 就应出现，否则 TimeStampToken就不能出现。

status 不能有除 PKIStatus 外的其他值，请求方如果收到一个不识别的值，必须报告错误。

申请失败时用 statusString 给出一个说明失败原因的字符串。statusString 的类型是 PKIFreeText，PKIFreeText 的定义为：

```
PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
```

每一个 UTF-8 String 应包含一个语言标签，该标签按照 RFC 3066 定义，用来指示 UTF-8 String 的语言。

failInfo 用来说明时间戳请求被拒绝的具体原因，具体值如下：

```
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不识别的值，必须报告错误。

TimeStampToken 是一个 ContentInfo 结构。当公钥算法为 RSA 时，ContentInfo 结构定义参见 RFC 3369；当公钥算法为 SM2 时，ContentInfo 结构定义参见 GM/T 0010。该结构中 content type 是一个 signed data content type，具体值如下：

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

TimeStampToken 不能含有除 TSA 签名以外的任何签名。而 TSA 证书的证书标识符(ESSCertID)应作为 SignerInfo 的属性包含在 SigningCertficate 属性里。

关于 TSTInfo 的具体定义如下：

```
TSTInfo ::= SEQUENCE{
  version                 INTEGER{v1(1)},
  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
  }
```

对于 TSTInfo 的各项解释如下：

——version 域说明了时间戳的版本号。

——policy 域应指明响应消息是根据 TSA 的哪个策略生成的。如果类似的域出现在 Time Stamp Req 中，这里应有相同的值，否则应返回错误(unacceptedPolicy)。这个 policy 可以包含但不仅限于下列类型的信息：

- 这个时间戳在什么条件下使用；
- 时间戳日志的有效性，以便以后能够证实时间戳是可信的。

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

——serialNumber 域是 TSA 分配的一个整数。对一个给定的 TSA 发出的每一个时间戳，serialNumber 都应是唯一的(即 TSA 的名字和序列号可以确定一个时间戳标志)。应该注意的是，

即使经历一个可能的服务中断(例如崩溃)后,这个特性也应保留。

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

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

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

如果 seconds、millis 或者 micros 没出现,则不出现的这些域的值应被赋为 0。当 accuracy 这个可选项不出现时,精确度可以从别的途径得到,例如 TSAPolicyId。

——ordering 表示时间戳排序条件。如果 ordering 域不出现,或者 ordering 域出现但被置为 false,那么 genTime 域只表示 TSA 创建时间戳的时间。在这种情况下,只有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 域如果在 TimeStampReq 中出现,在这里也应出现,值也应该等于 TimeStampReq 中的值。

——tsa 域的目的是为鉴别 TSA 的名字提供一个线索。如果出现,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 时间戳服务与时间戳系统的通信方式

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,而 TSA 也将颁发的时间戳通过电子邮件返回给用户。申请与颁发使用以下 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 同样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只包含消息的 DER 编码,文件的传送应采用可信赖的方法。

8.3 Socket 方式

采用 Socket 方式时，用户与 TSA 服务建立一个安全的 socket 连接后，通过该安全连接传输申请消息和响应消息。

通讯的消息格式为：

length (32-bits)，flag (8-bits)，value

其中 flag 所代表的消息定义如表 2 所示。

表 2 flag 信息定义表

flag	value	备注(消息名称)
'00'H	DER 编码	tsaMsg
'01'H	polling reference (32 bits)， time-to-check-back (32 bits)	pollRep
'02'H	polling reference (32 bits)	pollReq
'03'H	'00'H	negPollRep
'04'H	next polling reference (32 bits)， time-to-check-back (32 bits)， DER 编码	partialMsgRep
'05'H	DER 编码	finalMsgRep
'06'H	错误信息	errorMsgRep

每个消息名称所代表的含义见 RFC 3161 的“TSA”。

8.4 HTTP 方式

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

使用 HTTP 协议传输时间戳申请和响应消息需要用到 MIME 对象，可以使用的对象如下：

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，用户将符合 SOAP 规范传输格式的申请消息发送给 TSA，TSA 也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 概述

本标准规定的接口共提供了 7 个与时间戳服务有关的函数，涵盖了获取时间戳服务的全部功能，包括：

——环境函数

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
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

上述函数的返回值见附录 A。

本标准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标准范围内。

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： 0：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： 0：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_UINT32   uiHashAlgID,
                                        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

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

uiHashAlgID [IN]：密码杂凑算法标识

pucTSRequest [OUT]：时间戳请求

puiTSRequestLength [IN/OUT]：时间戳请求的长度

返回值：　0：成功；

其他：失败。

备注：　uiReqType：0 代表时间戳响应应该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。

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_ CreateTSReponse (void*   hTSHandle,
                                        SGD_UINT8   *pucTSRequest,
                                        SGD_UINT32   uiTSRequestLength,
                                        SGD_UINT32   uiSignatureAlgID,
                                        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

uiSignatureAlgID [IN]：签名算法标识

pucTSResponse [OUT]：时间戳响应

puiTSResponseLength [IN/OUT]：时间戳响应的长度

返回值：　0：成功；

其他：失败。

备注：　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 验证时间戳有效性

原型：
```
SGD_UINT32 STF_VerifyTSValidity(void*  hTSHandle,
                                SGD_UINT8  *pucTSResponse,
                                SGD_UINT32  uiTSResponseLength,
                                SGD_UINT32  uiHashAlgID,
                                SGD_UINT32  uiSignatureAlgID,
                                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
uiHashAlgID [IN]：密码杂凑算法标识
uiSignatureAlgID [IN]：签名算法标识
pucTSCert [IN]：TSA 的证书，DER 编码格式
uiTSCertLength [IN]：TSA 证书的长度

返回值：0：成功；
其他：失败。

备注：该函数验证时间戳响应是否有效。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。

9.7 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：0：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。

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。

9.8 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]:指定获取时间戳详细信息的项目编号

pucItemValue [OUT]:解析得到的时间戳相关信息

puiItemValueLength [IN/OUT]:时间戳相关信息的对应长度

返回值:0:成功;

其他:失败。

备注:该函数解析时间戳的详细信息,uiItemnumber 定义:

STF_TIME_OF_STAMP:签发时间

STF_CN_OF_TSSIGNER:签发者的通用名

STF_ORINGINAL_DATA:时间戳请求的原始信息

STF_CERT_OF_TSSERVER:时间戳服务器的证书

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_ORGNIZATION_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。

具体信息项的宏定义请参见 5.3。

当时间戳响应不包含时间戳服务器证书,调用该函数解析时间戳服务器的证书和证书链时,返回的是对应的证书序列号。此时,无法解析签发者的国家、组织、城市和电子信箱信息;但可以正常解析签发时间、签发者通用名和时间戳请求的原始信息。

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
（规范性附录）
时间戳接口错误代码定义和说明

宏 描 述	预定义值	说 明
STF_TS_OK	0	正常返回
STF_TS_ERROR_BASE	0x04000000	
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	输入项目编号无效
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_AVAILBLE	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
	0x04000010～0x040000FF	预留

附 录 B
（资料性附录）
时间戳接口应用示例

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。

a） 时间戳请求方主要程序代码如下：

```
# define SGD_SM3 0x1;
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;
SGD_UINT8 tscert[4096];
SGD_UINT32 tscertlen = 4096;
SGD_UINT8 itemvalue[64];
SGD_UINT32 itemvaluelength = 64;

retcode = STF_InitEnvironment(&handle);
if(retcode! = 0){
    /*错误处理*/
    return;
}
/*生成时间戳请求*/
retcode = STF_CreateTSRequest(handle, message, strlen(message), 1,"", 0, SGD_SM3,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*/
/*验证时间戳*/
/*时间戳服务器证书读到 tscert 里*/
retcode = STF_VerifyTSValidity(tsresponse, tsresponselength, 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, SGD_TIME_OF_STAMP, itemvalue, &itemvaluelength);
if (retcode ! = 0) {
```

```
    /* 错误处理 */
    STF_ClearEnvironment(handle);
    return;
}
STF_ClearEnvironment(handle);
```

b) 时间戳服务方主要程序代码如下：

```
void* handle;
SGD_UINT32 retcode;
SGD_UINT8 tsrequest[5120], tsresponse[5120];
SGD_UINT32 tsrequestlength = 5120, tsresponselength = 5120;

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, rsrequest, rsrequestlength, 1, 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);
```