```
                --a hash algorithm OID and the hash value of the data to be time-stamped
    reqPolicy           TSAPolicyId             OPTIONAL,
    nonce               INTEGER                 OPTIONAL,
    certReq             BOOLEAN                 DEFAULT FALSE,
    extensions          [0] IMPLICIT Extensions OPTIONAL  }
```

（1）version

version 字段用于区分请求包格式的版本，缺省值为 v1（0）。

（2）messageImprint

messageImprint 字段包含待签署时间戳的数据的摘要值，其格式用 ASN.1 描述如下：

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

如果 TSA 不能识别请求包中的摘要算法 messageImprint→hashAlgorithm，或者 TSA 认为请求包中的摘要算法强度不够，TSA 将拒绝提供时间戳服务，并在响应包 PKIStatusInfo 中返回状态 badAlg（表示算法错误）。

（3）reqPolicy

若 reqPolicy 字段存在，则表示响应包中 TimeStampToken 字段应遵循该策略。TSAPolicyId 用 ASN.1 描述如下：

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

（4）nonce

当客户端（请求包发起者）本地没有时钟时，通过 nonce 字段来验证响应包的时效性，表示与请求包对应的响应包中必须包含相同的 nonce，否则客户端拒绝该响应包。

nonce 是个大整数（如 64 位长整数），由客户端随机产生，尽量避免重复。

（5）certReq

如果 certReq 字段存在且设置为 true，则表示 TSA 公钥证书必须包含在响应包的 SignedData 中的 certificates 字段中。

如果 certReq 字段不存在或设置为 false，则表示响应包中 SignedData 中 certificates 字段不应该存在。

（6）extensions

extensions 字段用于请求包中的信息扩展，可包含多个扩展信息。extensions 格式同 X.509 数字证书扩展项格式。

如果 TSA 无法识别请求包中的扩展项，不管其是关键扩展项还是非关键扩展项，TSA 都不应该签发时间戳，应该在响应包 PKIStatusInfo 中返回状态 unacceptedExtension（表示扩展项未知）。

3. 响应包格式

时间戳响应包格式用 ASN.1 描述如下：

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

（1）status

status 字段表示返回状态，其格式用 ASN.1 描述如下：

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

① status：当 status 为 0 或 1 时，timeStampToken 必须存在。当 status 为其他值时，timeStampToken 不应该存在。常用 status 值用 ASN.1 描述如下：

```
PKIStatus ::= INTEGER {
    granted                 (0),    --表示成功
    grantedWithMods         (1),    --表示成功但需修改
    rejection               (2),    --表示拒绝
    waiting                 (3),    --表示等待
    revocationWarning       (4),    --表示即将作废
    revocationNotification  (5)     --表示已经作废
    }
```

② failInfo：failInfo 表示 TSA 拒绝签发时间戳的原因代码，此时 TimeStampToken 字段不存在。常用 failInfo 值用 ASN.1 描述如下：

```
PKIFailureInfo ::= BIT STRING {
    badAlg              (0),    --表示算法不支持或不识别
    badRequest          (2),    --表示请求包错误
    badDataFormat       (5),    --表示提交的数据格式错误
    timeNotAvailable    (14),   --表示 TSA 时间源不可用
    unacceptedPolicy    (15),   --表示请求包中策略不支持
    unacceptedExtension (16),   --表示请求包中扩展项不支持
    addInfoNotAvailable (17),   --表示扩展项中额外信息不支持
    systemFailure       (25)    --表示系统故障
    }
```

③ statusString：statusString 用于表示 TSA 拒绝签发时间戳的原因描述，如“messageImprint 字段格式错误”。

（2）timeStampToken

timeStampToken 表示时间戳令牌，用 ASN.1 描述如下：

```
TimeStampToken ::= ContentInfo
ContentInfo ::= SEQUENCE {
    contentType   ContentType,
    content     [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }
ContentType ::= OBJECT IDENTIFIER
```

其中，contentType 为 signedData。content 定义为 SignedData 类型。其中，SignedData→contentInfo→contentType 设置为 id-ct-TSTInfo，SignedData→contentInfo→content 定义为

TSTInfo 类型，TSA 公钥整数必须包含在 SignedData→certificates 中。用 ASN.1 描述如下：

```
SignedData ::= SEQUENCE {
    version Version,
    digestAlgorithms DigestAlgorithmIdentifiers,
    contentInfo ContentInfo,
    certificates   [0] IMPLICIT ExtendedCertificatesAndCertificates   OPTIONAL,
    crls   [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos SignerInfos }
id-ct-TSTInfo   OBJECT IDENTIFIER ::= { iso（1） member-body（2） us（840） rsadsi（113549）
pkcs(1) pkcs-9(9) smime(16) ct(1) 4}
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   }
```

① version 字段用于区分时间戳令牌的版本，缺省值为 v1（0）。

② policy 字段表示产生时间戳令牌时应遵循的策略，必须与时间戳请求包 reqPolicy 字段相同。如无法遵循 reqPolicy 策略要求，则在响应包 PKIStatusInfo 中返回错误状态 unacceptedPolicy（表示策略无法接受）。常用 policy 策略包括时间戳令牌产生的条件、是否保留时间戳令牌日志（用于事后验证）等。policy 格式同请求包中的 reqPolicy 字段。

③ messageImprint 字段包含待签署时间戳的数据的摘要值，必须与时间戳请求包 messageImprint 字段相同。messageImprint 格式同请求包中的 messageImprint 字段。

④ serialNumber 字段表示时间戳令牌的唯一序列号，由 TSA 统一分配，通过 TSA 名称和时间戳令牌序列号可以唯一标识时间戳令牌。serialNumber 应该支持大整数，长度可达 160 位。

⑤ genTime 字段表示时间戳令牌的产生时间。genTime 可采用两种格式：UTCTime 和 GeneralizedTime。UTCTime 用 2 位数字表示年份，GeneralizedTime 用 4 位数字表示年份。2049 年以前的时间可采用 UTCTime 格式，但 2050 年及以后的时间必须采用 GeneralizedTime 格式。当采用 UTCTime 格式时，如 2 位年份数字 YY 小于 50 时，则年份应该解释为 20YY 年；当 YY 大于或等于 50 时，年份应该解释为 19YY 年。

⑥ accuracy 字段表示 TSA 的时间精度。将 genTime 字段与该精度相加，即可获得时间戳令牌产生的最大时间；将 genTime 字段与该精度相减，即可获得时间戳令牌产生的最小时间。当 accuracy 字段不存在时，可通过其他方式获得 TSA 的时间精度，如 TSAPolicyId。

accuracy 用 ASN.1 描述如下：

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

其中，millis 表示毫秒，micros 表示微秒。

⑦ ordering 字段表示时间戳令牌产生时间的排序状态。如果 ordering 设置为 true，则表示同一个 TSA 签发的所有时间戳令牌，其 genTime 的时间顺序就表示产生顺序。即两个时间戳令牌 A 和 B，如果 A→genTime 大于 B→enTime，则表示 B 先于 A 产生。如果 ordering 不存在或者设置为 false，则 genTime 的时间顺序不一定表示产生顺序；只有两个时间戳令牌的产生时间 genTime 之差大于其精度之和时，genTime 的时间顺序才能表示产生顺序。即两个时间戳令牌 A 和 B，如果 A→enTime 与 B→enTime 之差大于 A→ccuracy 与 B→accuracy 之和，此时才表示 B 先于 A 产生，否则无法判断 A 和 B 的产生顺序。

⑧ nonce 字段用于客户端（请求包发起者）验证响应包的时效性。必须与时间戳请求包 nonce 字段相同。

⑨ tsa 字段表示 TSA 名称，可以是 TSA 证书中的 subject 字段，也可以是 subjectAltName 扩展项。其实，TSA 证书也包含在 SignedData→signerInfos 中。

⑩ extensions 字段用于请求包中的信息扩展，可包含多个扩展信息。extensions 格式同 X.509 数字证书扩展项格式。

20.5　SET

SET 是 Secure Electronic Transaction（安全电子协议）的缩写。它是 VISA 和 MastarCard 公司于 1997 年 5 月开发的一套规范，主要目的是保证互联网在线交易时信用卡支付的安全性，得到了 IBM、HP、Microsoft、Netscape、Verifone、GTE、Verisign 等公司的支持，已成为事实上的工业标准，并且获得了 IETF 组织的认可。

SET 协议的基本设计目标是保证持卡人、商家以及收单行之间在互联网上能够安全地进行支付交易。

SET 协议建立在以下几个商业要求的基础之上：

① 为支付信息提供机密性，保证与支付信息同时传输的订购信息的机密性。

② 保证所有传输数据的完整性。

③ 为持卡人提供认证，保证一个持卡人是一个支付账户的合法用户。

④ 为商家提供认证，保证商家通过一个收单行可以接受该品牌的支付卡交易。

⑤ 保证使用最好的安全技术和系统设计来保护所有电子商务交易的合法参与者。

⑥ 创建一个不依赖于传输安全机制的协议。

⑦ 鼓励网络和软件提供商支付互操作性。

SET 规范主要包括 SET 证书管理、SET 支付系统、SET 协议的外部接口指引 3 个方面的内容。

SET 协议中定义的参与者包括持卡人、商家、发卡行、收单行、支付网关和证书授权机构，具体描述如下。

① 持卡人（Cardholder）：在电子商务环境中，是指信用卡和数字证书的持有者，通过相应软件，可以借助支付卡和数字证书与商家完成支付交易。

② 商家（Merchant）：是指能够为持卡人提供服务或商品的实体。SET 协议中定义的商家能够与持卡人进行安全的电子交易，并且一个商家必须与相关的收单行达成协议，保证可以接受支付卡付款。

③ 发卡行（Issuer）：是指金融机构为持卡人建立一个账户并发放信用卡，保证对经过授权的交易进行付款。此外，发卡行还负责为持卡人颁发数字证书，数字证书用来鉴别持卡人的身份，证书中包括关于标识持卡人所持有的信用卡信息。

④ 收单行（Acquirer）：是指为商家建立一个账户并处理信用卡授权和支付的金融机构。

⑤ 支付网关（Paymeng Gateway）：是一个由收单行或指定的第三方操作的设备，位于商家和收单行之间，连接 SET 和现有的银行支付网络，用于处理信用卡授权和支付。因此，通常商家是与发卡行的支付网关进行交互，而不是与发卡行直接进行交互。

⑥ 认证机构（Certificate Authority）：是负责为持卡人、商家和支付网关签发和管理数字证书，让持卡人、商家和支付网关之间可以通过数字证书进行认证的一个机构。

SET 协议的主要功能包括：

① SET 协议位于应用层，对网络上其他各层也有涉及。它规范了整个商务活动的流程，从持卡人到商家、支付网关、认证中心以及信用卡结算中心之间的信息流走向和必须采用的加密、认证都制定了严格的标准，从而最大限度地保证了商务性、服务性、协调性和集成性。

② SET 是一个非常复杂的协议，它详尽而准确地反映了参与交易的各方之间存在的各种关系，还定义了加密信息的格式和完成一笔支付交易过程中各方传输信息的规则。

③ SET 不只是一个技术方面的协议，它还说明了各方所持有的数字证书的合法含义和希望得到数字证书以及响应信息的各方应有的动作，以及与一笔交易紧密相关的责任。

④ 就付款方式的实现而言，在 SET 协议中，商家在收到客户的信用卡及订单后，信用卡的信息通过支付网关自动转到传统的金融网络，在得到发卡机构的核准后银行即可进行付款。

SET 协议规定的工作流程分以下 3 个阶段：

① 在购买请求阶段，持卡人选购商品，确定支付方式，向商家发送购货单和一份经过签名、加密的信托书。信托书的信用卡号是经过加密的，商家无法获得。

② 在支付确认阶段，商家把信托书传送到收单银行，收单银行可以解密信用卡号，并通过认证验证签名。收单银行向发卡银行查问，确认持卡人的信用卡是否属实。属实则发卡银行认可并确认该笔交易，收单银行认可商家并确认此交易，最后商家向客户传送货物和收据。

③ 在收款阶段，交易成功，商家向收单银行出示所有交易的细节索款，收单银行按合同将货款划给商家。发卡银行向用户定期寄去信用卡消费账单。

SET 协议在一般使用环境下的工作步骤如下：

① 持卡人利用电子商务平台选择物品，并提交订单。