INTERNATIONAL STANDARD

ISO/IEC 18014-2

Second edition
2009-12-15

Information technology — Security techniques — Time-stamping services —

Part 2: Mechanisms producing independent tokens

Technologies de l'information — Techniques de sécurité — Services d'horodatage —

Partie 2: Mécanismes produisant des jetons indépendants

Reference number
ISO/IEC 18014-2:2009(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 18014-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 18014-2:2002). The text has been revised to clarify the presentation, and a new time-stamping mechanism has been added.

ISO/IEC 18014 consists of the following parts, under the general title *Information technology — Security techniques — Time-stamping services*:

— *Part 1: Framework*

— *Part 2: Mechanisms producing independent tokens*

— *Part 3: Mechanisms producing linked tokens*

Information technology — Security techniques — Time-stamping services —

Part 2: Mechanisms producing independent tokens

1 Scope

This part of ISO/IEC 18014 presents a general framework for the provision of time-stamping services.

Time-stamping services may generate, renew and verify time-stamp tokens.

Time-stamp tokens are associations between data and points in time, and are created in a way that aims to provide evidence that the data existed at the associated date and time. In addition, the evidence may be used by non-repudiation services.

This part of ISO/IEC 18014 specifies mechanisms that generate independent time-stamps: in order to verify an independent time-stamp token, verifiers do not need access to any other time-stamp tokens. That is, time-stamp tokens are not linked, as is the case for the token types defined in ISO/IEC 18014-3.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 8824-1:1998, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 9594-8:2005, *Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 18014-1:2008, *Information technology — Security techniques — Time-stamping services — Part 1: Framework*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
asymmetric key pair
pair of related keys where the private key defines the private transformation and the public key defines the public transformation

NOTE Adapted from ISO/IEC 9798-1.

3.2
asymmetric signature system
system based on asymmetric techniques whose private transformation is used for signing and whose public transformation is used for verification

NOTE Adapted from ISO/IEC 9798-1.

3.3
authentication
provision of assurance of the claimed identity of an entity

NOTE Adapted from ISO/IEC 18028-4.

3.4
certification authority
CA
authority trusted by one or more users to create and assign public-key certificates

NOTE Optionally, the certification authority can create the users' keys.

[ISO/IEC 9594-8:2005, definition 3.3.16]

3.5
cryptography
discipline that embodies principles, means, and mechanisms for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorized use

[ISO/IEC 7498-2:1989, definition 3.3.20]

3.6
data integrity
property that data has not been altered or destroyed in an unauthorized manner

[ISO/IEC 7498-2:1989, definition 3.3.21]

3.7
data origin authentication
corroboration that the source of data received is as claimed

[ISO/IEC 7498-2:1989, definition 3.3.22]

3.8
digital signature
data appended to, or a cryptographic transformation (see cryptography) of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery, e.g. by the recipient

[ISO/IEC 7498-2:1989, definition 3.3.26]

3.9
distinguished encoding rules
DER
encoding rules that may be applied to values of types defined using the ASN.1 notation

NOTE 1 As defined in the introduction to ISO/IEC 18028-4.

NOTE 2 Application of these encoding rules produces a transfer syntax for such values. It is implicit that the same rules are also to be used for decoding. The DER is more suitable if the encoded value is small enough to fit into the available memory and there is a need to rapidly skip over some nested values.

3.10
hash-function
function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

— it is computationally infeasible to find for a given output, an input which maps to this output

— it is computationally infeasible to find for a given input, a second input which maps to the same output

NOTE Computational feasibility depends on the specific security requirements and environment.

[ISO/IEC 10118-1:2000, definition 3.5]

3.11
hash-value
string of bits which is the output of a hash-function

NOTE Adapted from hash-code as defined in ISO/IEC 10118-1.

3.12
message authentication code
MAC
string of bits which is the output of a MAC algorithm

NOTE A MAC is sometimes called a cryptographic check value (see for example ISO 7498-2).

[ISO/IEC 9797-1:1999, definition 3.2.4]

3.13
message authentication code (MAC) algorithm
algorithm for computing a function which maps strings of bits and a secret key to fixed-length strings of bits, satisfying the following two properties:

— for any key and any input string the function can be computed efficiently

— for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute the function value on any new input string, even given knowledge of the set of input strings and corresponding function values, where the value of the ith input string may have been chosen after observing the value of the first $i-1$ function values

NOTE 1 A MAC algorithm is sometimes called a cryptographic check function (see for example ISO 7498-2).

NOTE 2 Computational feasibility depends on the user's specific security requirements and environment.

[ISO/IEC 9797-1:1999, definition 3.2.6]

3.14
private key
that key of an entity's asymmetric key pair which should only be used by that entity

[ISO/IEC 9798-1:1997, definition 3.3.17]

3.15
public key
that key of an entity's asymmetric key pair which can be made public

NOTE In the case of an asymmetric signature scheme the public key defines the verification transformation. In the case of an asymmetric encipherment system the public key defines the encipherment transformation. A key that is 'publicly known' is not necessarily globally available. The key may only be available to all members of a pre-specified group.

[ISO/IEC 11770-3:2008, definition 3.32]

3.16
public key certificate
public key information of an entity signed by the certification authority and thereby rendered unforgeable

[ISO/IEC 11770-3:2008, definition 3.33]

3.17
time-stamp requester
entity which possesses data it wants to be time-stamped

[ISO/IEC 18014-1:2008, definition 3.14]

3.18
time-stamp token
TST
data structure containing a verifiable cryptographic binding between a data item's representation and a time-value

NOTE A time-stamp token can also include additional data items in the binding.

[ISO/IEC 18014-1:2008, definition 3.15]

3.19
time-stamp verifier
entity which possesses data and wants to verify that it has a valid time-stamp bound to it

NOTE The verification process can be performed by the verifier itself or by a trusted third party.

[ISO/IEC 18014-1:2008, definition 3.16]

3.20
time-stamping authority
TSA
trusted third party trusted to provide a time-stamping service

[ISO/IEC 18014-1:2008, definition 3.17]

3.21
time-stamping policy
named set of rules that indicates the applicability of a time-stamp token to a particular community or class of application with common security requirements

3.22
time-stamping service
TSS
service providing evidence that a data item existed before a certain point in time

[ISO/IEC 18014-1:2008, definition 3.18]

3.23
time-stamping unit
TSU
set of hardware and software which is managed as a unit and generates time-stamp tokens

3.24
trusted third party
security authority, or its agent, trusted by other entities with respect to security related activities

[ISO/IEC 10181-1:1996, definition 3.3.30]

4 Notation, symbols and abbreviated terms

For the purposes of this document, the following notation and abbreviated terms apply.

$\wedge$	logical conjunction, i.e. the *and* operator of Boolean algebra
ASN.1	Abstract Syntax Notation One
CA	certification authority
DER	Distinguished Encoding Rules
$H_i(D)$	hash-value calculated by applying the hash-function H_i to the data string D
HMAC	hash message authentication code
isValid (TST(t), t_v)	predicate (i.e. true or false) indicating whether or not the token TST(t) is valid at time t_v
MAC	message authentication code
OID	object identifier
PKI	public key infrastructure
TSA	time-stamping authority
TSS	time-stamping service
TST	time-stamp token
TST(t)	time-stamp token created at time t
TSU	time-stamping unit
t, t_v	points in time

Hash-functions are specified in the multi-part standard ISO/IEC 10118.

MAC functions are specified in the multi-part standard ISO/IEC 9797.

The HMAC function is specified in ISO/IEC 9797-2.

5 Time-stamping services

Time-stamping services may generate, renew, and verify time-stamp tokens.

Time-stamp tokens are associations between data and points in time, and are created in a way that aims to provide evidence that the data existed at the associated date and time. In addition, the evidence may be used by non-repudiation services.

Time-stamping services involve the following entities (defined in ISO/IEC 18014-1):

- the time-stamp requester, that has a document to time-stamp;
- the Time-Stamping Authority (TSA), that generates time-stamp tokens (TSTs);
- the time-stamp verifier, that verifies time-stamps bound to documents.

A time-stamping service (TSS) provides three specific services:

- time-stamp generation, where the requester submits data items and receives a time-stamp generated by the TSA;
- time-stamp renewal, a special case of time-stamp generation, where the requester submits an existing time-stamp and related data items and receives a new time-stamp generated by the TSA, such that the validity period of the existing time-stamp is extended by the new time-stamp;
- time-stamp verification, in which the verifier validates the time-stamp.

Time-stamping services are provided by means of two protocols, as defined in ISO/IEC 18014-1:

- time-stamp request protocol: the requester requests the TSA to time-stamp a document or renew an existing time-stamp for a document, and
- time-stamp verification protocol: the verifier submits a time-stamp token to be verified.

6 Time-stamp tokens

6.1 Contents

A time-stamp token is a data structure containing a verifiable binding between a data item's representation and a time-value. A time-stamp token may also bind additional items to the data item’s representation and the time-value.

A time-stamp token shall contain

- one or more hash-values of the data that is to be time-stamped; hash-functions are specified in the multi-part standard ISO/IEC 10118;
- a point in time (a time-value);
- a reference to the policy under which the time-stamp token is generated.

together with any additional information that may be regarded as helpful for the practical provision of the service, such as

- identification of the time-stamping service provider (to help verifiers in looking for further evidence);
- an indication of the accuracy of the time point (that is, the maximum error in the time representation);
- an indication of ordering (that is, whether the service provider guarantees the relative ordering of generated tokens);
- identification of the version of the format (foreseeing syntax changes in the future);
- a serial number (to enable reference to be made to the token);
- a reference to the user’s request[1], to help users in matching requests and responses.

[1] Often referred to as a 'nonce', a number or bit string used only once, so that there is no ambiguity about what it is referring to.

6.2 Notation

Let

$H_i(D)$

be a hash-value computed on data D using hash-function H_i.

Let

$TST(t)$

be a time-stamp token issued at the point in time t.

The time-stamp token $TST(t)$ may be further decomposed into its parts:

$TST(t) := < \{ H_i(D) \}, t , P >$ [2]

where $\{ H_i(D) \}$ is the set of one or more hash-values[3] on data D. P indicates the policy under which the token was generated.

6.3 Verification

Let t_V be the moment when the time-stamp token is verified, where t_V is measured by the entity performing the validity check.

The validity of a time-stamp token may be verified by checking that:

- the time-stamp token is syntactically well-formed;
- $t < t_V$; [4]
- the value of every component $H_i(D)$ of the time-stamp token matches the hash-value of D evaluated at t_V over the document subject to scrutiny, using the same hash-function H_i;
- at least one of the hash-functions H_i is not broken at t_V;
- the protection of the time-stamp token is technically sound when the time-stamp token is verified at time t_V; that is, the protection mechanism is not broken;
- the issuing policy P is acceptable for the verifier's purposes.

If all the previous conditions hold, we say that the time-stamp token is valid at t_V. The following notation is used for the predicate that evaluates whether a time-stamp token TST(t) is valid at $t_{V:}$

$isValid\ (TST(t), t_V) = true$

The verifier may request additional assurance that is outside the scope of this standard.

[2] The notation <a, b, c, …> denotes a tuple, that is a sequence of values called the components of the tuple.

[3] Each hash-value $H_i(D)$ shall describe both the hash value and the hash-function used to derived it, altogether with any additional information that might be needed for recreate the hash value in the future (e.g. hash-function parameters). Hash-functions are standardised in ISO/IEC 10118. Use of hash-functions chosen from amongst those specified in ISO/IEC 10118 is recommended.

[4] The notation $t_1 < t_2$ indicates that 'time t_1 is previous to time t_2' according to the clock of the validating entity. Strict precedence is not always mandatory, and a verifier may specify a tolerance or accepted error margin in time values; if such a tolerance is used, it shall be a positive number, and it shall be stated in the verifier's practice statement. In such a case, the mathematical formula becomes $t_1 - t_2 <$ tolerance.

6.4 Renewal

A time-stamp generated at t_0 is theoretically valid forever. However, in practice, a time limit should apply, for example for one of the following reasons:

- the strength of any of the underlying cryptographic primitives is under suspicion and is no longer trusted;
- the TSA's signing key is about to expire;
- the TSA is about to cease provision of a time-stamping service;
- the policy specifies a time limit that is about to expire.

In such a case, a new time-stamp token is needed to extend the validity beyond the practical limits of the original token. This new token, generated at t_1, may extend the previous bound t_0 if generated using the renewal architecture described below; that is, the new time-stamp token binds the point in time t_0 to the data, and is valid beyond t_1. In general, several time-stamp tokens may be part of a renewal chain

$$[\ TST(t_0)\ \ TST(t_1)\ \ TST(t_2)\ \ \dots\ \ TST(t_i)\ \ \dots\ ] \qquad \text{where } t_0 < t_1 < t_1 < \dots < t_i < \dots$$

that extends the validity of the binding to t_0 an unlimited number of times.

In order to achieve this objective:

- the new time-stamp token at t_i shall be generated before the previous time-stamp token expires;
- the time-stamp token $TST(t_i)$ incorporates time-stamp token $TST(t_{i-1})$ as part of the protected information;
- the time-stamp request makes explicit the previous time-stamp token so that it can be incorporated into the response;
- the time-stamp verification shall extend over the chain of renewals.

6.5 Renewal verification

Let

$$[\ TST(t_0),\ TST(t_1),\ \dots,\ TST(t_n)\ ]$$

be a renewal chain, that is, an ordered list of time-stamp tokens:

- which all refer to the same data item D;
- for which the generation time is ordered; that is, $t_0 < t_1 < \dots < t_n$.

Let t_v be the moment when the time-stamp chain is verified.

The validity extension property states that

$$isValid\ (\ [TST(t_0),\ TST(t_1)],\ t_v) = isValid\ (TST(t_0),\ \mathrm{t}_1) \wedge isValid(\ TST(t_1),\ t_v)$$

where

$$t_0 < t_1 < t_v$$

That is, the first time-stamp token shall be valid when the second one is generated, and the second one shall be valid when verification is carried out.

The validity of a time-stamp chain may be verified by iterating the previous procedure, i.e.:

$$isValid\ (\ [TST(t_0),\ \ldots,\ TST(t_n)],\ t_v) = isValid\ (TST(t_0),\ t_1) \wedge \ldots \wedge isValid(\ TST(t_n),\ t_v)$$

where

$$t_0 < t_1 < \ldots < t_n < t_v$$

If the verification is successful, then the verifier can conclude that data item D existed before time t_0.

The verifier shall also check that the issuing policy (or policies) is acceptable for its purposes.

The verifier may request additional assurance that is outside the scope of this standard.

7 Protection mechanisms

Time-stamp tokens may be protected by a variety of mechanisms, that may be chosen by the requester and/or imposed by the service provider.

TSAs conformant with this part of ISO/IEC 18014 shall employ at least one of the three possible protection mechanisms presented below. The first mechanism uses digital signatures to sign the time-stamp token. The second mechanism uses a message authentication code (MAC) to authenticate the binding in a time-stamp token, and the third mechanism is based on TSA archiving of information.

- The first mechanism involves asking the time-stamping service provider to digitally sign the binding of the time to the document so that signature verification continues to validate the evidence.

- The second mechanism involves asking the time-stamping service provider to use a MAC to protect the binding. The same secret is needed for MAC creation and for MAC verification, and this secret is kept by the TSA. Therefore, the TSA is required for verification.

- The third mechanism involves asking the time-stamping service provider to archive the token, and only publish a reference to the archive. Therefore, the TSA is required for archival and verification.

Time-stamping service users may select the mechanism to be used by means of the ExtMethod extension. If this extension is not present, the TSA uses its default time-stamping mechanism[5].

[5] The ExtMethod extension should be omitted by users requesting the first mechanism, if compatibility with IETF RFC3161-compliant TSAs is required.

8 Independent time-stamp tokens

8.1 Core structure

The following ASN.1 structures apply, as specified in ISO/IEC 18014-1:

```
TSTInfo ::= SEQUENCE {
     version Version,
     policy TSAPolicyId,
     messageImprint MessageImprint,
     serialNumber SerialNumber,
     genTime GeneralizedTime,
     accuracy Accuracy OPTIONAL,
     ordering BOOLEAN DEFAULT FALSE,
     nonce Nonce OPTIONAL,
     tsa [0] EXPLICIT GeneralName OPTIONAL,
     extensions [1] Extensions OPTIONAL
     }
```

TSTInfo is encapsulated into TimeStampTokens:

```
TimeStampToken ::= SEQUENCE {
     contentType    CONTENT.&id({Contents}),
     content        [0] EXPLICIT CONTENT.&Type({Contents}{@contentType})
     }
```

Content is built according to the protection mechanism. Possible values for the 'contentType' and 'content' fields, and related data types for the 'content' field are presented in Clause 8.3.

```
Contents CONTENT ::= {
       time-stamp-mechanism-signature
     | time-stamp-mechanism-MAC
     | time-stamp-mechanism-archival
     | time-stamp-mechanism-signerinfo
       -- Expect additional time-stamp mechanisms --
     }
```

8.2 Extensions

Extensions are additional information that may be attached to either time-stamp requests or time-stamp tokens.

Extensions are encoded into ASN.1 structures, and require identifying OIDs.

```
tss OBJECT IDENTIFIER ::= { iso(1) standard(0) 18014 }

tss-ext OBJECT IDENTIFIER ::= { tss 1 }
```

The following extensions apply, as specified in ISO/IEC 18014-1:

- ExtHash identified by
  tss-ext-hash OBJECT IDENTIFIER ::= { tss-ext 1 }

- ExtMethod identified by
  tss-ext-method OBJECT IDENTIFIER ::= { tss-ext 2 }

- ExtRenewal identified by
  tss-ext-renewal OBJECT IDENTIFIER ::= { tss-ext 3 }

This standard does not define new extensions.

8.3 Protection mechanisms

8.3.1 Digital signatures using SignedData

The requester of the service may specify this mechanism using the ExtMethod extension that is specified in the extensions field of the time-stamp request message, using the following object identifier:

```
tss-itm-ds OBJECT IDENTIFIER::= { tss-itm 1 }
```

A requester of this type of service may omit the ExtMethod extension if the TSA's default time-stamping mechanism is known to be the digital signature mechanism using SignedData[6].

In this mechanism the TSA has an asymmetric signature key pair, and uses the private key to digitally sign the time-stamp token. Signature verification will use the public key.

A TSA may manage multiple TSUs for the purpose of issuing timestamp tokens under different levels of service or for load balancing purposes. For example, different TSUs managed by a single TSA may have distinct signing keys, distinct time sources, or distinct time-stamping policies.

The TSA shall sign all time-stamp tokens with a key reserved specifically for that purpose. If a PKI is being used that employs X.509 v3 certificates [ISO/IEC 9594-8: 2005], the corresponding certificate for the TSA shall contain only one instance of the extended key usage field extension with KeyPurposeID having value id-kp-timeStamping[7]. This extension shall be critical.

```
id-kp-timeStamping OBJECT IDENTIFIER::= {
     iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) kp (3) timestamping (8)
     }
```

The TimeStampToken encapsulates an instance of type SignedData structure as described in ISO/IEC 18014-1 Annex B.

```
time-stamp-mechanism-signature CONTENT ::=
     { SignedData IDENTIFIED BY id-signedData }

id-signedData OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2
     }
```

The input to the signing operation is the value of the encapContentInfo octet string that contains the DER-encoded value of a TSTInfo structure. See Clause B.1 in Annex B for additional information.

8.3.2 Message authentication codes using AuthenticatedData

The requester of the service may specify this mechanism using the ExtMethod extension that is specified in the extensions field of the time-stamp request message, using the following object identifier:

```
tss-itm-mac OBJECT IDENTIFIER::= { tss-itm 2 }
```

In this mechanism the TSA uses a secret key to digitally bind the point in time to the data being time-stamped. The time-stamp token is authenticated using a Message Authentication Code (MAC).

[6] The service requester should omit the method extension if compatibility with IETF RFC3161-compliant TSAs is required.

[7] This definition is compatible with [RFC 3280]. See [16].

See Annex B for information on formats to protect data using message authentication codes.

When using this mechanism, the TSA is needed to carry out the verification, and the TSA has to be trusted by all parties who wish to rely on the timestamps it generates, since there is no external evidence that might detect fraud. The secret key used to evaluate the MAC is kept secret. The secret key used for each token shall be available for later verification. The secret key used to seal a given token may be specific to that token, or common for a range of tokens.

Every exchange of information between the actors (requester, verifier, and TSA) requires data integrity and data origin authentication protection. This protection may be provided by any means, for example by transmission over a secured channel, or by signing exchanged data using public keys that do not need to last longer than the lifetime of the transaction.

The TimeStampToken encapsulates an AuthenticatedData structure as described below.

```
time-stamp-mechanism-MAC CONTENT ::=
     { AuthenticatedData IDENTIFIED BY id-ct-authData }

id-ct-authData OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
     smime(16) ct(1) 2
     }

AuthenticatedData ::= SEQUENCE {
     version CMSVersion,
     originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
     recipientInfos RecipientInfos,
     macAlgorithm MessageAuthenticationCodeAlgorithm,
     digestAlgorithm [1] DigestAlgorithmIdentifier OPTIONAL,
     encapContentInfo EncapsulatedContentInfo,
     authAttrs [2] IMPLICIT AuthAttributes OPTIONAL,
     mac MessageAuthenticationCode,
     unauthAttrs [3] IMPLICIT UnauthAttributes OPTIONAL
     }

MessageAuthenticationCode::= OCTET STRING
```

The recipientInfos component is present in a value of type AuthenticatedData, but it contains an empty set of values of type RecipientInfo:

```
RecipientInfos ::= SET SIZE(0) OF RecipientInfo
```

8.3.2.1 MAC generation

The MAC calculation process involves computing a message authentication code (MAC) on the content being authenticated. The input to the MAC calculation process is the value of the encapContentInfo eContent OCTET STRING, that contains the DER-encoded value of a TSTInfo structure. Only the octets comprising the value of the eContent are input to the MAC algorithm; the tag and the length are omitted.

The input to the MAC calculation process includes the MAC input data defined above, and an authentication key that is secret. The details of the MAC calculation depend on the MAC algorithm employed by the TSA. The object identifier, along with any parameters, that specifies the MAC algorithm employed is carried in the macAlgorithm field:

```
MessageAuthenticationCodeAlgorithm::= MACAlgorithmIdentifier

MACAlgorithmIdentifier ::= AlgorithmIdentifier {{ MACAlgorithms }}

MACAlgorithms ALGORITHM ::= {
     -- Expect additional MAC algorithms --
     }
```

8.3.2.2 MAC verification

The input to the MAC verification process includes the input data (as described in the previous clause) and the secret used by the TSA to authenticate the time-stamp token. The details of the MAC verification process depend on the MAC algorithm employed.

Furthermore, the verifier shall carry out a verification protocol exchange with the issuing TSA (see ISO/IEC 18014-1) using a data integrity and data origin authentication protected channel.

The TSA must be available for audit by third parties, under prior agreement, to verify that appropriate security measures are applied to protect activity logs and secret keys.

8.3.3 Archival

The requester of the service may specify this mechanism using the ExtMethod extension that is specified in the extensions field of the time-stamp request message, using the following object identifier:

```
tss-itm-archive OBJECT IDENTIFIER::= { tss-itm 3 }
```

In this mechanism the TSA returns a time-stamp token that references information to bind the time-stamp to the data in the time-stamp token. The TSA archives locally enough information to verify that the time-stamp is correct.

The TSA archives may consist of time-stamp tokens using any mechanism, e.g. activity logs, transmission logs, etc.

When using this mechanism, the TSA is needed to carry out the verification, and the TSA has to be thoroughly trusted since there is no external evidence that might detect fraud.

NOTE: In this scenario the TSA is playing the role of an Electronic Notary. Guidance on the use and management of this kind of Trusted Third Party may be found in ISO/IEC TR 14516.

Every exchange of information between the actors (requester, verifier, and TSA) requires data integrity and data origin authentication protection. This protection may be provided by any means, for example by transmission over a secured channel, or using private/public key pairs that do not need to last longer than the lifetime of the transaction.

The TimeStampToken encapsulates an instance of type "octet string" containing TSTInfo encoded using DER.

```
time-stamp-mechanism-archival CONTENT ::=
     { ETSTInfo IDENTIFIED BY id-data }

id-data OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1
     }

ETSTInfo ::=
     OCTET STRING (CONTAINING TSTInfo ENCODED BY der)
```

8.3.4 Digital signatures using SignerInfo

The requester of the service may specify this mechanism using the ExtMethod extension that is specified in the extensions field of the time-stamp request message, using the following object identifier:

```
tss-itm-signerinfo OBJECT IDENTIFIER::= { tss-itm 4 }
```

In this mechanism, the time-stamp token is the result of a signature operation, as described further below. The TSA has an asymmetric signature key pair, and uses the private key to digitally sign the time-stamp token. Signature verification will use the public key.

A TSA may manage multiple TSUs for the purpose of issuing timestamp tokens under different levels of service or for load balancing purposes. For example, different TSUs managed by a single TSA may have distinct signing keys, distinct time sources, or distinct time-stamping policies.

The TSA shall sign all time-stamp tokens with a key reserved specifically for that purpose. If a PKI is being used that employs X.509 v3 certificates [ISO/IEC 9594-8:2005], the corresponding certificate for the TSA shall contain only one instance of the extended key usage field extension with KeyPurposeID having value id-kp-timeStamping[8]. This extension shall be critical.

```
id-kp-timeStamping OBJECT IDENTIFIER::= {
     iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) kp (3) timestamping (8)
     }
```

The TimeStampToken encapsulates an instance of type SignerInfo structure as defined below.

```
time-stamp-mechanism-signerinfo CONTENT ::=
     { SignerInfo IDENTIFIED BY id-signerinfo }

id-signerInfo OBJECT IDENTIFIER ::= { tss-itm-signerinfo 1 }
```

This format is constructed as a digital signature in such a way that:

1) the "content" field of the time-stamp token may be annexed into a signed data encapsulation of the object being time-stamped, along with other signatures over the same object,

2) the signature may be parsed as a standard signature, and

3) the signer is the TSA.

The aim is to provide one more signature for a document, where this additional signature is of special relevance with respect to the time of generation. Several time-stamping signatures may be accumulated to enhance the level of trust in the time-stamp (e.g. time-stamp tokens from different providers, and/or under different policies).

SignerInfo is defined as[9]:

```
SignerInfo ::= SEQUENCE {
     version CMSVersion,
     sid SignerIdentifier,
     digestAlgorithm DigestAlgorithmIdentifier,
     signedAttrs [0] IMPLICIT SignedAttributes[10],
     signatureAlgorithm SignatureAlgorithmIdentifier,
     signature SignatureValue,
     unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL
     }

SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

Attribute ::= SEQUENCE {
     attrType OBJECT IDENTIFIER,
     attrValues SET OF AttributeValue }

AttributeValue ::= ANY
```

[8] This definition is compatible with [RFC 3280]. See [16].

[9] This data structure is compatible with SignerInfo in [RFC 3852].

[10] This field is OPTIONAL in [RFC 3852] but here it is required to contain two attributes: the signing time, and the time-stamp token information.

There shall be two signed attributes, at least:

- a time-stamp attribute
- a signing time attribute

The attribute type of the time-stamp attribute shall be tss-attribute:

```
tss-attribute OBJECT IDENTIFIER ::= { tss-itm-signerinfo 2 }
```

The attribute value is the DER encoding of the TSTInfo structure described in clause 8.1.

The attribute type of the signing-time attribute shall be:

```
id-signingTime OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) 5
     }
```

Signing-time attribute values have ASN.1 type GeneralizedTime.

The same point in time T shall be used for the signing-time attribute value in this clause, and for the genTime field in TSTInfo.

8.4 Protocols

The following protocols apply as specified in ISO/IEC 18014-1:2008:

- time-stamp request (ISO/IEC 18014-1:2008, 7.1);
- time-stamp response (ISO/IEC 18014-1:2008, 7.2);
- verification request (ISO/IEC 18014-1:2008, 7.3);
- verification response (ISO/IEC 18014-1:2008, 7.3);

This part of ISO/IEC 18014 does not define new protocols.

Annex A
(normative)

ASN.1 Module for Time-Stamping

This annex provides the ASN.1 definitions described in the body. The syntax is according to ISO/IEC 8824-1:1998.

Table 1 – Summary of object identifier branches introduced or extended by the module below.

OID	ASN.1 identifier	meaning
1	iso	
1.0	standard	
1.0.18014	tss	time-stamping services
1.0.18014.0	modules	parts of the standard
1.0.18014.0.2	TimeStampingServices-2	part 2: independent tokens
1.0.18014.1	tss-ext	extensions
1.0.18014.1.1	tss-ext-hash	hash extension
1.0.18014.1.2	tss-ext-method	mechanism extension
1.0.18014.1.3	tss-ext-renewal	time-stamp renewal
1.0.18014.2	tss-itm	independent tokens - protection mechanisms
1.0.18014.2.1	tss-itm-ds	signed data
1.0.18014.2.2	tss-itm-mac	authenticated data
1.0.18014.2.3	tss-itm-archive	archival
1.0.18014.2.4	tss-itm-signerinfo	signer info
1.0.18014.2.4.1	id-signerInfo	SignerInfo contents
1.0.18014.2.4.2	tss-attribute	time-stamp info as attribute

```
-- Information technology - Security techniques -
-- Time-stamping services - Part 2: Independent Tokens

TimeStampingServices-2 {
  iso(1) standard(0) tss(18014) modules(0) part2(2) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS All; --

IMPORTS

  -- ISO/IEC 9594-8 | ITU-T Rec. X.509 AuthenticationFramework --
  EXTENSION, Extensions
    FROM AuthenticationFramework {
      joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 4 }

  -- ISO/IEC 9594-8 | ITU-T Rec. X.509 CertificateExtensions --
  GeneralName
    FROM CertificateExtensions {
      joint-iso-itu-t ds(5) module(1) certificateExtensions(26) 4 }
  SignedData, AuthenticatedData,
  SignatureAlgorithmIdentifier, SignatureValue,
  SignerIdentifier, Attribute
    FROM CryptographicMessageSyntax {
      iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) cms-2004(24) };

-- object identifiers for time-stamping services

pkcs OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  }

pkcs-7 OBJECT IDENTIFIER ::= { pkcs pkcs7(7) }
pkcs-9 OBJECT IDENTIFIER ::= { pkcs pkcs9(9) }

id-signedData OBJECT IDENTIFIER ::= { pkcs-7 signedData(2) }
id-ct-authData OBJECT IDENTIFIER ::= { pkcs-9 smime(16) ct(1) 2 }
id-data OBJECT IDENTIFIER ::= { pkcs-7 data(1) }

id-ct-TSTInfo OBJECT IDENTIFIER ::= { pkcs-9 smime(16) ct(1) 4 }

der OBJECT IDENTIFIER ::= {
  joint-iso-itu-t asn1(1) ber-derived(2) distinguished-encoding(1)
  }

id-ripemd160 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) teletrust(36)
  algorithm(3) hashAlgorithm(2) ripemd160(1)
  }
id-sha1 OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) oiw(14) secsig(3) 2 26
  }
id-sha224 OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  csor(3) nistalgorithm(4) hashalgs(2) 4
  }
id-sha256 OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  csor(3) nistalgorithm(4) hashalgs(2) 1
  }
id-sha384 OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  csor(3) nistalgorithm(4) hashalgs(2) 2
  }
id-sha512 OBJECT IDENTIFIER ::= {
  joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
  csor(3)nistalgorithm(4) hashalgs(2) 3
  }

-- time-stamping services
tss OBJECT IDENTIFIER ::= { iso(1) standard(0) 18014 }
modules OBJECT IDENTIFIER ::= { tss modules(0) }

-- extensions
tss-ext OBJECT IDENTIFIER ::= { tss ext(1) }
tss-ext-hash OBJECT IDENTIFIER ::= { tss-ext hash(1) }
tss-ext-method OBJECT IDENTIFIER ::= { tss-ext method(2) }
tss-ext-renewal OBJECT IDENTIFIER ::= { tss-ext renewal(3) }
```

```
-- protection mechanisms
tss-itm OBJECT IDENTIFIER ::= { tss 2 }
tss-itm-ds OBJECT IDENTIFIER ::= { tss-itm 1 }
tss-itm-mac OBJECT IDENTIFIER ::= { tss-itm 2 }
tss-itm-archive OBJECT IDENTIFIER ::= { tss-itm 3 }
tss-itm-signerinfo OBJECT IDENTIFIER ::= { tss-itm 4 }

id-signerinfo OBJECT IDENTIFIER ::= { tss-itm-signerinfo 1 }
tss-attribute OBJECT IDENTIFIER ::= { tss-itm-signerinfo 2 }

OIDS ::= CLASS {
  &id OBJECT IDENTIFIER UNIQUE
  }
  WITH SYNTAX { OID &id }

-- TSTInfo

TSTInfo ::= SEQUENCE {
  version Version,
  policy TSAPolicyId,
  messageImprint MessageImprint,
  serialNumber SerialNumber OPTIONAL,
  genTime GeneralizedTime,
  accuracy Accuracy OPTIONAL,
  ordering BOOLEAN DEFAULT FALSE,
  nonce Nonce OPTIONAL,
  tsa [0] EXPLICIT GeneralName OPTIONAL,
  extensions [1] Extensions OPTIONAL
  }

Version ::= INTEGER { v1(1) }

TSAPolicyId ::= POLICY.&id({TSAPolicies})

TSAPolicies POLICY ::= {
  --
  ... -- Any supported TSA policy --
  }

  POLICY ::= OIDS -- Supported TSA policies

MessageImprint ::= SEQUENCE {
  hashAlgorithm DigestAlgorithmIdentifier,
  hashedMessage OCTET STRING
  }

DigestAlgorithmIdentifier ::= AlgorithmIdentifier {{ DigestAlgorithms }}

AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
  algorithm ALGORITHM.&id({IOSet}),
  parameters ALGORITHM.&Type({IOSet}{@algorithm}) OPTIONAL
  }

ALGORITHM ::= CLASS {
  &id OBJECT IDENTIFIER UNIQUE,
  &Type OPTIONAL
  }
  WITH SYNTAX { OID &id [PARMS &Type] }
```

```
DigestAlgorithms ALGORITHM ::= {
  { OID id-ripemd160 PARMS NULL } |
  { OID id-sha1 PARMS NULL } |
  { OID id-sha224 PARMS NULL } |
  { OID id-sha256 PARMS NULL } |
  { OID id-sha384 PARMS NULL } |
  { OID id-sha512 PARMS NULL },
  --
  ... -- Expect additional digest algorithms --
  }

SerialNumber ::= INTEGER -- Expect large values

Accuracy ::= SEQUENCE {
  seconds INTEGER OPTIONAL,
  millis[0] INTEGER(1..999) OPTIONAL,
  micros [1] INTEGER(1..999) OPTIONAL
  }
  (ALL EXCEPT({ -- no components present -- }))

Nonce ::= INTEGER -- Expect large values

-- Time-stamping extensions --

TSExtensions EXTENSION ::= {
  extHash |
  extMethod |
  extRenewal,
  --
  ... -- Expect additional extensions --
  }

extHash EXTENSION ::= {SYNTAX ExtHash IDENTIFIED BY tss-ext-hash}
ExtHash ::= SEQUENCE SIZE(1..MAX) OF MessageImprint

extMethod EXTENSION ::= {SYNTAX ExtMethod IDENTIFIED BY tss-ext-meth}
ExtMethod ::= SEQUENCE SIZE(1..MAX) OF Method

Method ::= METHOD.&id({Methods})

Methods METHOD ::= {
  --
  ... -- Any time-stamping method --
  }

METHOD ::= OIDS -- TSA Methods

extRenewal EXTENSION ::= { SYNTAX ExtRenewal IDENTIFIED BY tss-ext-renewal }
ExtRenewal ::= TimeStampToken

-- time-stamp tokens

TimeStampToken ::= SEQUENCE {
  contentType CONTENT.&id({Contents}),
  content [0] EXPLICIT CONTENT.&Type({Contents}{@contentType})
  }

CONTENT ::= TYPE-IDENTIFIER-- ISO/IEC 8824-2, Annex A
```

```
Contents CONTENT ::= {
  { SignedData IDENTIFIED BY id-signedData } |
    -- time-stamp mechanism signature
  { AuthenticatedData IDENTIFIED BY id-ct-authData } |
    -- time-stamp mechanism MAC
  { ETSTInfo IDENTIFIED BY id-data } |
    -- time-stamp mechanisn archival
  { SignerInfo IDENTIFIED BY id-signerinfo },
    -- time-stamp mechanism signer-info
  --
  ... -- Expect additional time-stamp mechanisms --
  }

-- additional structures for protection tokens

ETSTInfo ::=
  OCTET STRING (CONTAINING TSTInfo ENCODED BY der)

SignerInfo ::= SEQUENCE {
  version INTEGER,
  sid SignerIdentifier,
  digestAlgorithm DigestAlgorithmIdentifier,
  signedAttrs [0] IMPLICIT SignedAttributes,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature SignatureValue,
  unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL
  }


SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute

-- time-stamp generation
TimeStampReq ::= SEQUENCE {
  version Version,
  messageImprint MessageImprint,
  reqPolicy TSAPolicyId OPTIONAL,
  nonce Nonce OPTIONAL,
  certReq BOOLEAN DEFAULT FALSE,
  extensions[0] Extensions OPTIONAL
  }

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

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

PKIStatus ::= INTEGER {
  granted (0),
    -- the request is completely granted
  grantWithMods (1),
    -- modifications were necessary,
    -- the requester is responsible for asserting the differences
  rejection (2),
      -- the request could not be fulfilled,
      -- the failure code delivers additional information
    waiting (3),
      -- the request is not processed
    revocationWarning (4),
      -- a revocation is imminent
    revocationNotification (5)
      -- notification that a revocation occurred
    }

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

PKIFailureInfo ::= BIT STRING {
    badAlg (0),
      -- unrecognized or unsupported algorithm Identifier
    badRequest (2),
      -- transaction not permitted or supported
    badDataFormat (5),
      -- data submitted has the wrong format
    timeNotAvailable (14),
      -- the TSAs service is not available
    unacceptedPolicy (15),
      -- the requested TSA policy is not supported
    unacceptedExtension (16),
      -- the requested TSA extension is not supported,
    addInfoNotAvailable (17),
      -- the requested additional information is not available,
    systemNotAvailable (24),
      -- system is not available
    systemFailure (25),
      -- System Failure
    verificationFailure (27)
      -- verification of time stamp has failed
    }

-- time-stamp verification
VerifyReq ::= SEQUENCE {
    version Version,
    tst TimeStampToken,
    requestID [0] OCTET STRING OPTIONAL
    }

VerifyResp ::= SEQUENCE {
    version Version,
    status PKIStatusInfo,
    tst TimeStampToken,
    requestID [0] OCTET STRING OPTIONAL
    }

END
```

Annex B
(informative)

Cryptographic Syntax

ASN.1 cryptographic syntax is standardised in RFC 3852 [17] that provides:

- a format for signing data in section "5.4 Signed data", including signature production and verification; this format is adequate for the protection mechanism described in clause 8.3.1 of this standard;
- a format for protecting data using message authentication codes in section "5.6 Authenticated data", including the production and validation processes; this format is adequate for the protection mechanism described in clause 8.3.2 of this standard;
- a format for transporting opaque information; this format is adequate as protection mechanism described in clause 8.3.3 of this standard;
- a format for transporting signers' information in section "5.4.2 Signer information"; this format is adequate for the protection mechanism described in clause 8.3.4 of this standard.

The format for time-stamp tokens is specified in Annex A.

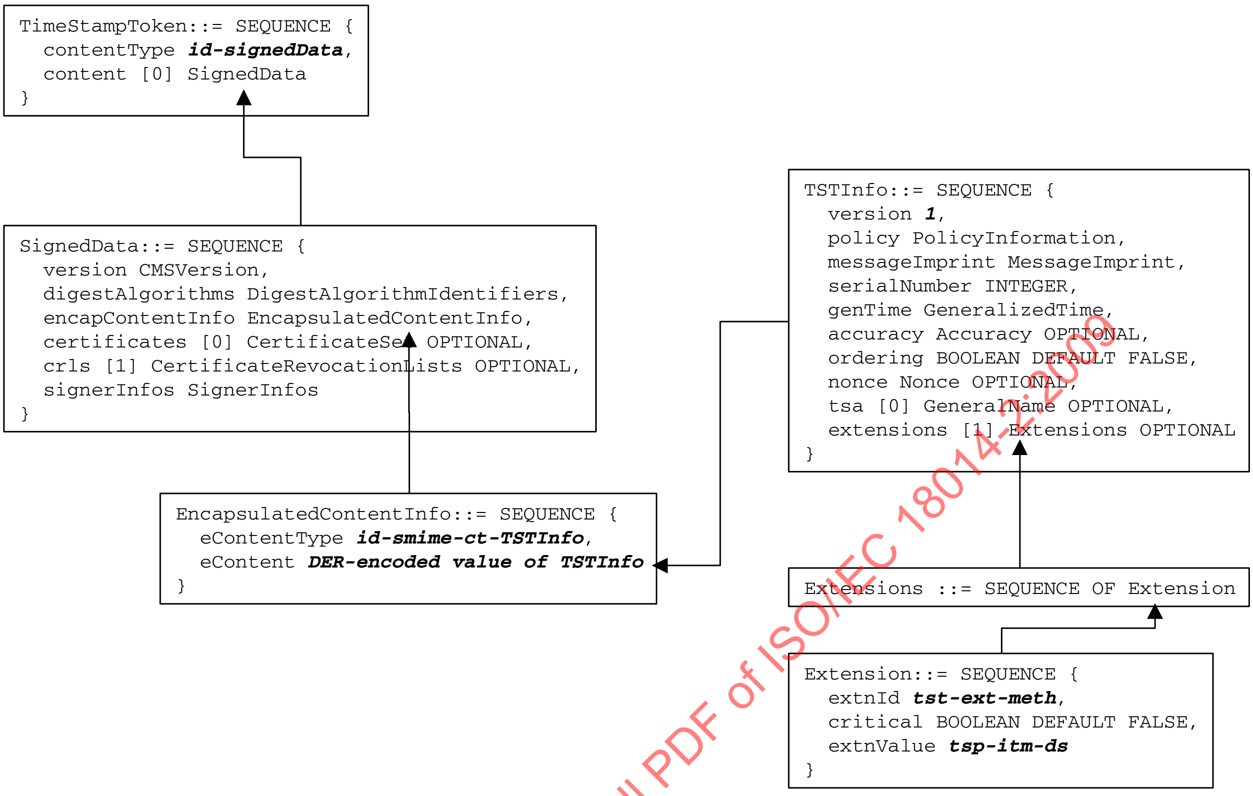
Further detail is provided in clauses B.1 to B.4 below.

[ISO/IEC 18014-1] describes ASN.1 formats that are a subset of those described in this standard. It only defines the format covered in clause 8.3.1 of this part of ISO/IEC 18014.

RFC 3161 [15] describes ASN.1 formats that are compatible with the ones described in this standard:

- RFC 3161 only defines part of the formats covered in clause 8 of this standard.
- This standard makes extensive use of extensions, allowed but not detailed in RFC 3161. The format is backwards compatible; that is, the default values for this standard are those in RFC 3161.
- RFC 3161 compliant service providers may not understand the extensions presented in clause 8.2 of this standard; therefore, they are expected to apply the default.
- This standard extends the status codes with respect to those defined in RFC 3161.

B.1 Signed data



```
TimeStampToken::= SEQUENCE {
  contentType id-signedData,
  content [0] SignedData
}
```

```
SignedData::= SEQUENCE {
  version CMSVersion,
  digestAlgorithms DigestAlgorithmIdentifiers,
  encapContentInfo EncapsulatedContentInfo,
  certificates [0] CertificateSet OPTIONAL,
  crls [1] CertificateRevocationLists OPTIONAL,
  signerInfos SignerInfos
}
```

```
EncapsulatedContentInfo::= SEQUENCE {
  eContentType id-smime-ct-TSTInfo,
  eContent DER-encoded value of TSTInfo
}
```

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

```
Extensions ::= SEQUENCE OF Extension
```

```
Extension::= SEQUENCE {
  extnId tst-ext-meth,
  critical BOOLEAN DEFAULT FALSE,
  extnValue tsp-itm-ds
}
```

Figure B.1 — Time-stamp tokens using SignedData

The structure is built as a SignedData construct, as described in ISO/IEC 18014-1 and in RFC 3852 [17].

contentType is:

```
id-signedData OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

content is:

```
SignedData ::= SEQUENCE {
     version CMSVersion,
     digestAlgorithms DigestAlgorithmIdentifiers,
     encapContentInfo EncapsulatedContentInfo,
     certificates [0] IMPLICIT CertificateSet OPTIONAL,
     crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
     signerInfos SignerInfos }

EncapsulatedContentInfo::= SEQUENCE {
     eContentType CONTENT.&id({EContents}),
     eContent [0] EXPLICIT CONTENT.&Type({EContents}{@eContentType}) }

EContents CONTENT ::= {
     { ETSTInfo IDENTIFIED BY id-ct-TSTInfo },
     --
     ... -- expect additional content types -- }

ETSTInfo ::= OCTET STRING (CONTAINING TSTInfo ENCODED BY der)

ContentType ::= OBJECT IDENTIFIER
```