INTERNATIONAL STANDARD

ISO/IEC 18014-2

Third edition
2021-09

Information security — Time-stamping services —

Part 2: Mechanisms producing independent tokens

Sécurité de l'information — Services d'horodatage —

Partie 2: Mécanismes produisant des jetons indépendants

Reference number
ISO/IEC 18014-2:2021(E)

Contents

Page

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

This third edition cancels and replaces the second edition (ISO/IEC 18014-2:2009), which has been technically revised.

The main changes compared to the previous edition are as follows:

— updated the definition of a hash function to a collision-resistant hash-function;

— application of style and editorial changes.

A list of all parts in the ISO/IEC 18014 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Information security — Time-stamping services —

Part 2: Mechanisms producing independent tokens

1 Scope

This document specifies mechanisms that generate, renew, and verify independent time-stamps. In order to verify an independent time-stamp token, time-stamp verifiers do not need access to any other time-stamp tokens. That is, such time-stamp tokens are not linked.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 18014-1, *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.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
time-stamp token
TST
data structure containing a verifiable binding between a data items' representation and a time-value

[SOURCE: ISO/IEC 18014-1:2008, 3.15, modified – Note to entry has been removed.]

3.2
time-stamping service
TSS
service providing evidence that a data item existed before a certain point in time

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

3.3
time-stamping policy
set of rules that indicates the applicability of a *time-stamp token* (3.1) to a particular community and/or class of application with common security requirements

[SOURCE: ISO/IEC 18014-1:2008, 3.23]

3.4
time-stamp requester
entity which possesses data it wants to be time-stamped

[SOURCE: ISO/IEC 18014-1:2008, 3.14, modified – Note to entry has been removed.]

3.5
time-stamp verifier
entity which possesses data and wants to verify that it has a valid time-stamp bound to it

Note 1 to entry: The verification process can be performed by the verifier itself or by a trusted third party.

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

3.6
time-stamping authority
TSA
trusted third party trusted to provide a *time-stamping service* (3.2)

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

3.7
time-stamping unit
TSU
set of hardware and software which is managed as a unit and generates *time-stamp tokens* (3.1)

3.8
data origin authentication
corroboration that the source of data received is as claimed

Note 1 to entry: Data origin is sometimes called data source.

[SOURCE: ISO 7498-2:1989, 3.3.22, modified — In the definition, the initial article "the" has been removed. Note 1 to entry has been added.]

3.9
data integrity
property that data has not been altered or destroyed in an unauthorized manner

[SOURCE: ISO/IEC 9797-1:2011, 3.4]

3.10
asymmetric key pair
pair of related keys where the *private key* (3.11) defines the private transformation and the *public key* (3.12) defines the public transformation

[SOURCE: ISO/IEC 9798-1:2010, 3.3]

3.11
private key
key of an entity's *asymmetric key pair* (3.10) that is kept private

Note 1 to entry: The security of an *asymmetric signature system* (3.15) depends on the privacy of this key.

[SOURCE: ISO/IEC 11770-1:2010, 2.35, modified — The definition was restricted to asymmetric signature system.]

3.12
public key
key of an entity's *asymmetric key pair* (3.10) which can usually be made public without compromising security

[SOURCE: ISO/IEC 11770-1:2010, 2.36]

3.13
public key certificate
public key (3.12) information of an entity signed by the certification authority

[SOURCE: ISO/IEC 11770-1:2010, 2.37]

3.14
public-key infrastructure
PKI
infrastructure able to support the management of public keys able to support authentication, encryption, integrity or non-repudiation services

[SOURCE: ISO/IEC 9594-8:2020, 3.5.60, modified — In the definition, the initial article "the" has been removed.]

3.15
asymmetric signature system
system based on asymmetric cryptographic techniques whose private transformation is used for signing and whose public transformation is used for verification

[SOURCE: ISO/IEC 9798-1:2010, 3.4]

3.16
digital signature
data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the *source* (3.8) and *integrity* (3.9) of the data unit and protect against forgery, e.g. by the recipient

[SOURCE: ISO/IEC 9798-1:2010, 3.11]

3.17
collision-resistant hash-function
hash-function satisfying the following property: it is computationally infeasible to find any two distinct inputs which map to the same output

[SOURCE: ISO/IEC 10118-1:2016, 3.1, modified — Note 1 to entry has been removed.]

3.18
hash-code
string of bits which is the output of a *collision-resistant hash-function* (3.17)

Note 1 to entry: The definition in ISO/IEC 10118-1 does not require collision-resistance. In this document, all hash functions are collision-resistant hash functions.

[SOURCE: ISO/IEC 10118-1:2016, 3.3, modified — In the definition, "collision-resistant" has been added. Note 1 to entry has been replaced.]

3.19
message authentication code algorithm
MAC algorithm
algorithm for computing a function which maps strings of bits and a *MAC algorithm key* (3.20) to fixed-length strings of bits, satisfying the following two properties:

— for any *MAC algorithm key* (3.20) and any input string, the function can be computed efficiently;

— for any fixed *MAC algorithm key* (3.20), and given no prior knowledge of the *MAC algorithm key* (3.20), it is computationally infeasible to compute the function value on any new input string, even given knowledge of a set of input strings and corresponding function values, where the value of the i-th input string might have been chosen after observing the value of the first $i - 1$ function values

[SOURCE: ISO/IEC 9797-1:2011, 3.10, modified — In the definition, "secret key" has been replaced with "MAC algorithm" and "key" has been replaced with "MAC algorithm key". At the end, the parentheses have been removed.]

3.20
MAC algorithm key
key that controls the operation of a *MAC algorithm* (3.19)

[SOURCE: ISO/IEC 9797-1:2011, 3.8]

3.21
message authentication code
MAC
string of bits which is the output of a *MAC algorithm* (3.19)

[SOURCE: ISO/IEC 9797-1:2011, 3.9, modified — Note 1 to entry has been removed.]

3.22
distinguished encoding rules
DER
encoding rules that may be applied to values of types defined using the ASN.1 notation

Note 1 to entry: 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.

4 Notation, symbols and abbreviated terms

ASN.1	abstract syntax notation one Annex A provides the ASN.1 definitions described in this document, which shall be used to identify the OIDs of ASN.1 module for time-stamping.
$\mathrm{H_i}$	collision-resistant hash-function
$\mathrm{H_i}(X)$	hash-code computed on data X
isValid(TST(t), t_v)	predicate (i.e. True or False) indicating whether or not the token TST(t) is valid at point in time t_v
OID	object identifier
TST(t)	time-stamp token created at point in time t
t, t_v	points in time
$t_1 < t_2$	The time-stamp verifier considers t_1 to be an earlier point in time than t_2. Strict precedence is not always mandatory, and a time-stamp verifier may specify a tolerance or accepted error margin in time units. When such tolerance is permitted, the allowed value ε shall be a positive number, and it shall be stated in the time-stamp verifier's practice statement. In such a case, the time-stamp verifier shall accept t_1 as being earlier than t_2 as long as no more than ε time units have elapsed from t_2 to t_1.
$<a, b, c>$	a triplet, that is a sequence of values called the components of the triplet
$\wedge$	logical conjunction, i.e. the *and* operator of Boolean algebra

5 Time-stamp tokens

5.1 Contents

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

A time-stamp token shall contain:

— one or more hash-codes of the data that is to be time-stamped;

— a point in time;

— a reference to the time-stamping policy under which the time-stamp token is generated;

together with any additional information that can be regarded as helpful for the practical provision of the time-stamping service, such as:

— identification of the time-stamping authority (to help time-stamp verifiers in looking for further evidence);

— an indication of the accuracy of the point in time (that is, the maximum error in the representation of the point in time);

— an indication of ordering (that is, whether the time-stamping authority 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.

5.2 Generation

Let $\mathrm{H_i}(D)$ be a hash-code computed on data D using a collision-resistant hash-function $\mathrm{H_i}$.

The time-stamp token TST(t) consists of the triplet

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

where $\{ \mathrm{H_i}(D) \}$ is a set of one or more hash-codes on data D. P indicates the time-stamping policy under which the time-stamping token was generated. Each hash-code $\mathrm{H_i}(D)$ shall describe both the hash-code and the collision-resistant hash-function used to derive it, together with any additional information that is needed to recreate the hash-code in the future (e.g. collision-resistant hash-function parameters).

NOTE Collision-resistant hash-functions are standardized in the ISO/IEC 10118 series.

5.3 Verification

Let t_v be the point in time when the time-stamp token is verified, where t_v is measured by the time-stamp verifier.

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

— the time-stamp token t is syntactically well-formed;

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

— $t<t_v$;

— the value of every component in { $H_i(D)$ } of the time-stamp token matches the hash-code of D evaluated at t_v over the data subject to scrutiny, using the same collision-resistant hash-function H_i together with any additional information that was used to generate the hash-code;

— the issuing time-stamping policy P is acceptable for the time-stamp verifier's purposes.

If all these conditions hold, it is said that the time-stamp token is valid at t_v. Otherwise, the time-stamp token is said to be invalid. 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   if the time-stamp token is valid;

isValid (TST(t), t_v) = False otherwise.

The time-stamp verifier may request additional assurances that are outside the scope of this document.

5.4 Renewal

A time-stamp generated at t_0 is theoretically valid indefinitely. However, in practice, a time limit (i.e. a point in time after which a token generated at t_0 is no longer valid) should apply, for example for one of the following reasons:

— the strength of any of the underlying cryptographic primitives is no longer trusted;

— the TSA's secret key is about to expire;

— the TSA is about to cease provision of a time-stamping service;

— the time-stamping policy specifies a point in time limit after which the time-stamp expires.

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

[TST(t_0), TST(t_1), TST(t_2), ..., TST(t_i), ...]

where $t_0 < t_1 < t_2 < ... < t_i < ...$

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 request shall make explicit the previous time-stamp token so that it can be incorporated into the response;

— the time-stamp token TST(t_i) shall incorporate the time-stamp token TST(t_{i-1}) as part of the protected information.

5.5 Renewal verification

Let [TST(t_0), TST(t_1), ..., TST(t_n)] be a renewal chain, i.e. 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, $t_0 < t_1 < ... < t_n$.

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

The validity extension property states that

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

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 shall be verified by iterating the previous procedure, i.e.:

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

The time-stamp verifier shall also check that the issuing time-stamping policy (or time-stamping policies) is acceptable for its purposes. If the verification is successful, the time-stamp verifier can conclude that data item D existed before t_0. The time-stamp verifier may request additional assurances that are outside the scope of this document.

6 Protection mechanisms

Time-stamp tokens can be protected by a variety of mechanisms that may be chosen by the time-stamp requester and/or imposed by the time-stamping authority.

TSAs conformant with this document shall employ at least one of the four possible protection mechanisms presented in 7.3. The first and fourth mechanisms use an asymmetric signature system to sign the time-stamp token. The second mechanism uses a MAC algorithm to authenticate the binding in a time-stamp token, and the third mechanism is based on TSA archiving of information.

— The first and fourth mechanisms involve asking the time-stamping authority to digitally sign the binding of the point in time to the document so that digital signature verification continues to validate the evidence.

— The second mechanism involves asking the time-stamping authority to use a MAC algorithm 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 authority 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 shall use its default time-stamping mechanism. The *ExtMethod* extension should be omitted by users requesting the first mechanism, if compatibility with IETF RFC 5816-compliant TSAs is required (see RFC 5816).

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

Every exchange of information between the actors (time-stamp requester, time-stamp verifier, and TSA) requires data integrity and data origin authentication protection. This protection can be provided by any appropriate means, for example by transmission over a secured channel, or by signing exchanged data using an asymmetric key pair that does not need to last longer than the lifetime of the transaction.

7 Independent time-stamp tokens

7.1 Core structure

The following ASN.1 structures shall 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 *TimeStampToken*:

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

The *content* field 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 7.3.

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

7.2 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 shall 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 document does not define new extensions.

7.3 Protection mechanisms

7.3.1 Digital signatures using *SignedData*

The time-stamp requester 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 time-stamp requester may omit the *ExtMethod* extension if the TSA's default time-stamping mechanism is known to be the asymmetric signature system using *SignedData.* The time-stamp requester should omit the method extension if compatibility with IETF RFC 5816-compliant TSAs is required (see RFC 5816).

In this mechanism the TSA has an asymmetric key pair, and uses the private key to digitally sign the time-stamp token. Verification shall use the corresponding public key.

The TSA shall sign all time-stamp tokens with a private key reserved specifically for that purpose. If a PKI is being used that employs X.509 v3 public key certificates (see ISO/IEC 9594-8), the corresponding public key certificate for the TSA shall contain only one instance of the extended key usage field extension with *KeyPurposeID* having value *id-kp-timeStamping*.[2] 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* field encapsulates an instance of type *SignedData* structure as described in ISO/IEC 18014-1.

```
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.2 for additional information).

7.3.2 Message authentication codes using *AuthenticatedData*

7.3.2.1 General

The time-stamp requester 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 MAC algorithm key to digitally bind the point in time to the data being time-stamped. The time-stamp token is authenticated using a MAC algorithm.

See Annex B for information on formats to protect data using this mechanism.

When using this mechanism, the TSA is required to carry out the verification, and the TSA shall be trusted by all parties who wish to rely on the time-stamp tokens it generates since there is no external evidence that can be used to detect fraud. The MAC Algorithm key shall be kept secret by the TSA. The MAC algorithm key used for each token shall be available for later verification. The MAC algorithm key used to authenticate a given time-stamp token may be specific to that token, or common for a range of tokens.

The *TimeStampToken* encapsulates an *AuthenticatedData* structure as described below.

2) This definition is compatible with RFC 5280.[12]

```
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
```

7.3.2.2 MAC generation

The MAC algorithm 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.

NOTE MAC algorithms are standardized in the ISO/IEC 9797 series.

The input to the MAC calculation process includes the MAC input data defined above, and a MAC algorithm key. 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 --
   }
```

7.3.2.3 MAC verification

The input to the MAC verification process includes the input data as described in 7.3.2.2 and the MAC algorithm key used by the TSA to authenticate the time-stamp token. The details of the MAC verification process depend on the MAC algorithm employed. The time-stamp verifier shall carry out a verification protocol exchange with the issuing TSA using a data integrity and data origin authentication protected channel.

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

7.3.3 Archival

The time-stamp requester 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 enough information locally to verify that the time-stamp is correct. The structure and format of the TSA archives is outside the scope of this document.

When using this mechanism, the TSA is required to carry out the verification, and the TSA shall be trusted by all parties who wish to rely on the time-stamp tokens it generates since there is no external evidence that can be used to 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 a trusted third party appears in ISO/IEC TR 14516.

The *TimeStampToken* encapsulates an 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)
```

7.3.4 Digital signatures using *SignerInfo*

The time-stamp requester 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 an asymmetric signature system, as described further below. The TSA has an asymmetric key pair, and uses the private key to digitally sign the time-stamp token. Digital signature verification shall use the corresponding public key.

The TSA shall sign all time-stamp tokens with a private key reserved specifically for that purpose. If a PKI is being used that employs X.509 v3 certificates (see ISO/IEC 9594-8), the corresponding public key certificate for the TSA shall contain only one instance of the extended key usage field extension with *KeyPurposeID* having value *id-kp-timeStamping*.3) 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:

a) 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 digital signatures over the same object;

b) the digital signature can be parsed as such; and

c) the signer is the TSA.

The aim is to provide an additional digital signature for a data item, where this additional digital signature is of special relevance with respect to the point in time when it was generated.

NOTE Several time-stamping digital signatures can be accumulated to provide a higher level of assurance.

3) This definition is compatible with RFC 5280.[12]

SignerInfo is defined as:[4)]

```
SignerInfo ::= SEQUENCE {
   version CMSVersion,
   sid SignerIdentifier,
   digestAlgorithm DigestAlgorithmIdentifier,
   signedAttrs [0] IMPLICIT SignedAttributes5),
   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
```

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 7.1.

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 7.3.4, and for the *genTime* field in *TSTInfo*.

7.4 Protocols

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

— time-stamp request;

— time-stamp response;

— verification request;

— verification response.

This document does not define new protocols.

4) This data structure is compatible with *SignerInfo* in RFC 5652.[13]

5) This field is OPTIONAL in RFC 5652, but here it is required to contain two attributes: the signing point in time, and the time-stamp token information.

Annex A
(normative)

ASN.1 Module for time-stamping

This annex provides the ASN.1 definitions described in this document. The syntax is according to ISO/IEC 8824-1. Table A.1 presents the meaning of OIDs and ASN.1 identifiers.

Table A.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
```