INTERNATIONAL STANDARD

ISO/IEC 18014-3

Second edition
2009-12-15

Information technology — Security techniques — Time-stamping services —

Part 3: Mechanisms producing linked tokens

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

Partie 3: Mécanismes produisant des jetons liés

Reference number
ISO/IEC 18014-3:2009(E)

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

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member 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 shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 18014-3 was prepared by Technical Committee ISO/TC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This second edition replaces and cancels the first edition (ISO/IEC 18014-3:2004), which has been technically revised. New message formats and data types are defined to support a protocol for extending an existing linked token with data items referring to a published value issued by the TSA. The data type clauses have been expanded and re-ordered, and the ASN.1 definitions in Annex A have been updated and reordered in line with the contents of the clauses in the main body of this International Standard. Annexes B and C have been updated.

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*

Introduction

ISO/IEC 18014-1 provides a general framework for the provision of time-stamping services. This part of ISO/IEC 18014 specifies mechanisms producing linked tokens, that is, time-stamp tokens that are related, or "linked", to other time-stamp tokens produced by the methods and processes described in this document. A time stamping authority (TSA) can utilise the methods and processes described within this document to provide a secure, verifiable cryptographic binding between a certain point in time and data values, in a way that enhances the security of the resulting token.

Information technology — Security techniques — Time-stamping services —

Part 3:
Mechanisms producing linked tokens

1 Scope

This part of ISO/IEC 18014

— describes a general model for time-stamping services producing linked tokens,

— describes the basic components used to construct a time-stamping service producing linked tokens,

— defines the data structures used to interact with a time-stamping service producing linked tokens,

— describes specific instances of time-stamping services producing linked tokens, and

— defines a protocol to be utilized by time-stamping services producing linked tokens for the purpose of extending linked tokens to published values.

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 10118 (all parts), *Information technology — Security techniques — Hash-functions*

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
aggregation
process of generating a proxy data item for a group of data items that are linked together, producing a verifiable cryptographic link between each data item and the rest of the group

3.2
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

[ISO/IEC 10118-1:2000, definition 3.2]

3.3
data items' representation
data item or its respective hash value

[ISO/IEC 18014-1: 2008]

3.4
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

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

3.5
hash value
string of bits which is the output of a hash-function

NOTE See ISO/IEC 10118-1:2000, 3.4.

3.6
link
data item attesting to the existence of at least two other data items through the use of collision-resistant hash functions

3.7
time-stamping authority
TSA
trusted third party trusted to provide a time-stamping service

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

3.8
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]

3.9
time-stamp token
TST
data structure containing a verifiable cryptographic binding between a data items' 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]

3.10
trusted third party
TTP
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 General discussion

This part of ISO/IEC 18014 describes methods and processes to form time-stamp tokens that are related, or "linked", to other time-stamp tokens produced using methods and processes described within this document. A time-stamping authority (TSA) uses these methods and processes to provide a secure, verifiable cryptographic binding between a certain point in time and data values, and to enhance the security of the resulting tokens by reducing the level of assurance required in TSA operations. The trustworthiness of the time-stamp tokens computed using these methods and processes depends on the integrity of the TSA-maintained repository of previously computed results of linking operations. The integrity of the repository of links and of the linking operations performed by the TSA can be verified cryptographically, and does not depend on the trustworthiness of a Public Key Infrastructure (PKI) or require trusting that a particular private key has not been compromised.

This part of ISO/IEC 18014 additionally:

- describes methods and processes that a TSA producing linked tokens may utilise to generate values derived from its linking operations with the intent of making them widely available in online or offline media; and

- defines a request-response protocol that allows for time-stamp tokens issued by the TSA to be extended with data items that refer to the values published by the TSA.

Time-stamping services conforming to this part of ISO/IEC 18014 may interoperate through the use of standard tokens, operations, and data formats. In general, a TSA producing linked tokens uses the cryptographic properties of hash-functions to "link" a time-stamp token to other time-stamp tokens previously generated by the TSA. A recommended means of computing a value linking an event with a previously generated link is to concatenate a data representation of the event with the link and use the resulting concatenation as input to a hash-function. The resulting hash value provides a verifiable cryptographic link between the event and the previously generated link. Collision-resistant hash-functions defined in ISO/IEC 10118 are suitable for use in forming linked tokens: pre-image resistance helps to conceal the content of the original data item from the TSA, while collision resistance ensures a false token cannot be inserted into an existing collection of linked tokens.

Note: For a complete treatment of time-stamping with linked tokens, see [HS91], [BHS93], [HS97], [BLLV98], and [BLS00]. For a proof of security of the linking technique, see [HS97] Section 3.1, Theorem 1.

5 Operations specific to TSAs producing linked tokens

5.1 Linking operation

The linking operation involves forming a verifiable binding between the time-stamp token and links previously produced by the TSA through the use of collision-resistant hash-functions. The value of the most recently produced such link provides a cryptographic summary of all time-stamp tokens that have ever been involved in the linking process.

A conformant TSA producing linked tokens shall perform linking operations using one of the following methods:

- linear chain linking;

- anti-monotone binary linking;

- threaded tree linking.

Linear chain linking involves creating links in the following way: the value of the most recently generated link shall be concatenated with the hash-value of a representation of the most recent event, and a collision-resistant hash-function shall then be applied.

Anti-monotone binary linking involves creating links in the following way: the value of two previously generated links shall be concatenated with the hash-value of a representation of the most recent event, and a collision-resistant hash-function shall then be applied. One of the two input links shall be the most recently generated link, and the precise identity of the other link will be determined by a data structure, maintained by the TSA, that shall take the form of an acyclic directed graph (see Annex B.2.3).

Threaded tree linking involves creating links in the following way: the value of some number of previously generated links shall be concatenated with the hash-value of a representation of the most recent event, and a collision-resistant hash-function shall then be applied. One of the input links shall be the most recently generated link, and the precise identity of the other links will be determined by a data structure, maintained by the TSA, that shall take the form of a threaded tree (see Annex B.2.4).

In every case, other data items may be concatenated to the input to the collision-resistant hash function prior to generating the new link (for example, time-variant parameters). Additionally, multiple collision-resistant hash-functions may be used concurrently on the same input for the purpose of generating a new link, and, in such a case, the new link is generated by concatenating the resulting hash values.

The values of the previous links used as input to the collision resistant hash-function(s) shall be included in the time-stamp token(s) associated with the most recent event and returned to the requesters within the 'links' field of the 'BindingInfo' structure (see Clause 7.4).

If hash-functions are used in linking operations, those specified in ISO/IEC 10118 shall be used.

Annex B.2 contains additional discussion regarding the algorithms that may be used in linking operations.

5.2 Aggregation operation

The aggregation operation involves forming a verifiable binding amongst a group of time-stamp tokens that are to be assigned the same time value through the use of a cryptographic function that is both pre-image resistant and collision-resistant, such as a collision-resistant hash-function. An aggregation scheme takes a group of time-stamp tokens as inputs and produces a single aggregate value, as well as verifiable cryptographic data linking together each time-stamp token with the rest of the group. The aggregate value associated with the aggregation operation is further used as an input to a linking operation by the TSA, in a way similar to the case of the linking operation for a single time-stamp token. A conformant TSA may thus perform linking operations for groups of time-stamp tokens, rather than for each time-stamp token individually, in order to improve computational efficiency or achieve a higher degree of service scalability.

A conformant TSA producing linked tokens that supports aggregation shall perform aggregation operations using one of the following methods:

- Merkle-tree aggregation;
- one-way accumulator aggregation.

All forms of aggregation involve first applying collision-resistant hash-function(s) to the encapsulated 'TSTInfo' data structures of the time-stamp tokens to be aggregated. We refer to the resulting hash-values as the 'hash-values to be aggregated'.

Merkle tree aggregation involves arranging the hash-values to be aggregated as the leaves of a tree structure (see Annex B.3.3). A value is then assigned to each non-leaf node of the tree by concatenating the hash-values assigned to its child nodes and then applying a collision-resistant hash-function, until all nodes of the tree structure have been assigned values. The value assigned to the root of the tree is the aggregate value. Multiple collision-resistant hash-functions may be used concurrently in this process, and, in such a case, the value assigned to each non-leaf node is computed by concatenating the hash values generated by each hash-function.

One-way accumulator aggregation involves computing an aggregate value in such a way that verification of the aggregate value for any of the participating hash-values may be done in constant time, regardless of the number of hash-values to be aggregated. Precisely how this is achieved is outside the scope of this part of ISO/IEC 18014; an example of a possible technique is outlined in Annex B.3.4.

If hash-functions are used in aggregation operations, those specified in ISO/IEC 10118 shall be used.

Annex B.3 contains additional discussion regarding the algorithms that may be used in aggregation operations.

Note: Common forms of aggregation include Merkle hash trees [M80], one-way accumulators [BD93], or other binary linking schemes [BLLV98].

5.3 Publishing operation

Values derived from the links produced by a conformant TSA may be published in such a way as to make the linking operations “widely-witnessed.” This can be achieved by periodic publication in widely available media, e.g., web pages or printed publications. The value published should depend on all time-stamp tokens generated by the TSA since the previous publishing event. By linking the tokens to "widely-witnessed" events, the TSA in effect generates verifiable statements of when each time-stamp token was generated by the system.

A conformant TSA producing linked tokens that supports publishing shall perform publishing operations using one of the following methods:

- single-link publishing;
- Merkle tree publishing.

Single link publishing involves the periodic publication of a single link value. This single link value enables the verification of all link values produced up to the time at which it was generated. For further details see Annex B.4.3.

Merkle tree publishing involves the periodic publication of the Merkle tree aggregation of all links generated since the last publication event. For further details see Annex B.4.4.

If hash-functions are used in publishing operations, those specified in ISO/IEC 10118 shall be used.

Annex B.4 contains additional discussion regarding the publishing operations performed by a conformant TSA.

5.4 Extend operation

If a conformant TSA makes published values available, an entity in possession of a linked token issued by this TSA may carry out a request-response protocol with the issuing TSA or other TTP that has access to the issuing TSA’s links for the purpose of extending this time-stamp token to a published value, using a data integrity and data origin authentication protected channel. This protocol should be carried out once the published value corresponding to this time-stamp token has been issued by the TSA and becomes available, and it results in the generation of a new time-stamp token, containing the same components as the original, as well as additional data items that refer to the published value and bind the linked token to it. The additional data items permit the computation of the corresponding published value without requiring access to the issuing TSA’s links. Message formats supporting the extend operation are described in Clauses 6.5 and 6.6.

6 Message formats

6.1 Time-stamp request

A time-stamp request is a message sent by a time-stamp requester to a TSA in order to request that the TSA issue a time-stamp token for the data items in the message. As defined in ISO/IEC 18014-1, the time-stamp request contains the data fields listed in Table 1.

Table 1 — Time-stamp request

Data field	Description
version	version number of this data structure
messageImprint	message imprint to which the TSA is to bind a time value
reqPolicy	service policy requested from the TSA (optional)
nonce	identifier enabling the request to be matched to the issued time-stamp token (optional)
certReq	requests the TSA to provide certificate information, if present
extensions	additional items required to properly fulfil the requested time-stamp request (optional)

The ASN.1 definition of the time-stamp request is given in Annex A as `TimeStampReq`.

6.2 Time-stamp response

A time-stamp response is a message returned by the TSA in response to a time-stamp request. As defined in ISO/IEC 18014-1, the time-stamp response contains the data fields listed in Table 2.

Table 2 — Time-stamp response

Data field	Description
status	status of the time-stamp operation
timeStampToken	issued time-stamp token, if the time-stamp operation is successful

The ASN.1 definition of the time-stamp response is given in Annex A as `TimeStampResp`.

6.3 Verify request

A verify request is a message sent by a time-stamp verifier to a TSA or other TTP that has access to the issuing TSA's links in order to request the entity receiving the message to verify the validity of the time-stamp token contained in the message. As defined in ISO/IEC 18014-1, the verify request contains the data fields listed in Table 3.

Table 3 — Verify request

Data field	Description
version	version number of this data structure
tst	time-stamp token to be verified
requestID	request identifier (optional)

The ASN.1 definition of the verify request is given in Annex A as `VerifyReq`.

6.4 Verify response

A verify response is a message returned by the TSA or other TTP that has access to the issuing TSA's links in response to a verify request. As defined in ISO/IEC 18014-1, the verify response contains the data fields listed in Table 4.

Table 4 — Verify response

Data field	Description
version	version number of this data structure
status	status of the verify operation
tst	submitted time-stamp token
requestID	response identifier, matches request identifier (optional)

The ASN.1 definition of the verify response is given in Annex A as `VerifyResp`.

6.5 Extend request

An extend request is a message sent by an entity in possession of a linked token to the issuing TSA or other TTP that has access to the issuing TSA's links in order to request the entity receiving the message to extend the time-stamp token contained in the message with additional data items that refer to values published by the issuing TSA. The extend request may be granted once the publishing event for the time-stamp token has taken place, that is, after the TSA has issued a published value for the time interval covering the time value in the time-stamp token. The extend request contains the data fields listed in Table 5.

Table 5 — Extend request

Data field	Description
version	version number of this data structure
tst	time-stamp token to be extended to a published value
requestID	request identifier (optional)

For the purposes of this standard, the 'version' field shall be set to one.

The 'requestID' field binds the extend request to the corresponding extend response.

The ASN.1 definition of the extend request is given in Annex A as `ExtendReq`.

6.6 Extend response

An extend response is a message returned by the TSA or other TTP that has access to the issuing TSA's links in response to an extend request. If the request is granted, a new time-stamp token is returned in the response with additional data items that refer to values published by the issuing TSA. The extend response contains the data fields listed in Table 6.

Table 6 — Extend response

Data field	Description
version	version number of this data structure
status	status of the extend operation
tst	updated time-stamp token if granted; original time-stamp token otherwise
requestID	response identifier, matches request identifier (optional)

For the purposes of this standard, the 'version' field shall be set to one.

The 'status' field indicates the result of the extend operation performed by the TSA or other TTP fulfilling the request to extend the submitted time-stamp token to a published value.

The 'tst' field contains a time-stamp token that corresponds to the time-stamp token submitted in the corresponding extend request. If the extend request is not granted, the time-stamp token in the 'tst' field is identical to the one submitted in the extend request; if the extend request is granted, the time-stamp token in the 'tst' field matches the one in the extend request in all components with the possible exception of any data items within an existing publication extension component of the time-stamp token's 'BindingInfo' structure, and additionally carries data items referring to at least one published value. The additional data items are carried within one or more 'PublicationInfo' structures, as described in Clause 7.8. In the case of a time-stamp token using the 'DigestedData' encapsulation, as described in Clause 8.2, the 'PublicationInfo' structures are carried within a publication extension inside the 'extensions' field of the time-stamp token's 'BindingInfo' structure.

The 'requestID' field shall be present if a request identifier is present in the corresponding extend request; in this case, it shall have the same value as the request identifier in the corresponding extend request.

The ASN.1 definition of the extend response is given in Annex A as `ExtendResp`.

7 Data types

7.1 Object identifiers

A root object identifier arc is defined to support allocation of further object identifiers within this standard. The ASN.1 definition of this object identifier is given in Annex A as `tsp-lt` and is reproduced below:

```
tsp-lt ::= OBJECT IDENTIFIER { iso(1) standard(0) time-stamp(18014) lt(3) }
```

Subsequent object identifiers defined in this document originate from this root identifier and are listed in Annex A. Additional object identifiers defined elsewhere and used in this document are also presented in Annex A.

7.2 TSTInfo

The 'TSTInfo' data type represents the object generated by the TSA in the process of issuing a time-stamp token. As defined in ISO/IEC 18014-1, the 'TSTInfo' data type contains the data fields listed in Table 7.

Table 7 — TSTInfo

Data field	Description
version	version number of this data structure
policy	TSA's service policy
messageImprint	message imprint to which the TSA is to bind a time value
serialNumber	integer assigned by the TSA
genTime	time assigned by the TSA
accuracy	accuracy of 'genTime' field compared to UTC (optional)
ordering	Boolean, false by default; if set to true, allows ordering of time-stamp tokens issued by this TSA based on 'genTime' field alone (regardless of 'accuracy' field)
nonce	identifier, matches 'nonce' field in time-stamp request (optional)
tsa	TSA name (optional)
extensions	additional data items (optional)

For the purposes of this standard, the 'version' field shall be set to one.

The 'serialNumber' field may be set to zero; if it is not set to zero, its value shall be an integer assigned by the TSA to each time-stamp token, and shall be unique for each time-stamp token issued by a given TSA in its lifetime of operation.

The 'extensions' field contains data items that represent additional information related to the data items in the corresponding time-stamp request. Clause 7.9.1 defines data types that may be used in the 'extensions' field.

The ASN.1 definition of the 'TSTInfo' data type is given in Annex A as `TSTInfo`.

7.3 TimeStampToken

The ' TimeStampToken' data type represents the time-stamp token issued by a TSA. As defined in ISO/IEC 18014-1, the TSA generates a time-stamp token by:

- populating a 'TSTInfo' structure according to the data items in the time-stamp request,
- performing actions according to the time-stamping mechanism employed by the TSA, and
- encapsulating the resulting 'TSTInfo' structure and other data items according to a content type encapsulation appropriate for the time-stamping mechanism employed by the TSA.

For the purposes of this part of ISO/IEC 18014, only the ‘DigestedData’ and the ‘SignedData’ content type encapsulations are supported.

The ASN.1 definition of the 'TimeStampToken' data type is given in Annex A as `TimeStampToken`.

7.4 BindingInfo

The ‘BindingInfo’ data type represents the data items generated by a TSA as a result of linking the encapsulated 'TSTInfo' structure of a time-stamp token to previously generated time-stamp tokens. The ‘BindingInfo’ data type contains the data fields listed in Table 8.

Table 8 — BindingInfo

Data field	Description
version	version number of this data structure
msgImprints	message digest or digests computed over the encapsulated TSTInfo structure of the time-stamp token
aggregate	data authenticating the aggregation of the 'msgImprints' field with other members of the aggregation (optional)
links	data item or items that combine the result of the 'msgImprints' and 'aggregate' fields with prior results of linking operations
publish	data authenticating the result of this linking operation against a published value (optional, deprecated)
extensions	additional data items (optional)

For the purposes of this standard, the 'version' field shall be set to one.

The ‘msgImprints’ field is computed over the encapsulated 'TSTInfo' structure of the time-stamp token, as described in Clause 8.1.

The ‘aggregate’ field contains one or more instances of type 'Chain', and it authenticates the participation of the ‘msgImprints’ field in the aggregation operation as described in Clause 8.1, if present. An instance of type 'Chain' contains data authenticating participation in a single aggregation operation. The ‘aggregate’ field may contain multiple instances of type 'Chain' when multi-tiered aggregation schemes are supported, and the values of the instances of type 'Chain' are to be computed in order.

The ‘links’ field contains one or more instances of type 'Link'. It represents the linking operation for the time value assigned in the encapsulated ‘TSTInfo’ of the corresponding time-stamp token, and includes within its components the results of linking operations from prior time values that were used as input to the current one. It shall always include within its components the result of the linking operation for the immediately preceding time value, which represents the running summary of the cumulative linking operations so far. If multiple instances of type 'Link' are present, the values of the instances of type 'Link' are to be computed in order.

For a given ‘BindingInfo’ structure within a time-stamp token, the result of the linking operation for the time value in the encapsulated ‘TSTInfo’ structure shall be computed as follows: if no ‘aggregate’ field exists, the value of the ‘links’ field is computed taking into account the contents of the 'msgImprints' field; otherwise, first the value of the ‘aggregate’ field is computed taking into account the contents of the 'msgImprints' field, then the value of the 'links' filed is computed taking into account the previously computed result of the ‘aggregate’ field.

Use of the ‘publish’ field is deprecated; see Clauses 7.9.2.3 and 7.8 for additional information related to published values and data types that may refer to them within the 'extensions' field of the 'BindingInfo' structure.

The 'extensions' field contains data items that represent additional information relating to the aggregation, linking, and publishing operations for the time value assigned in the encapsulated ‘TSTInfo’ structure of the time-stamp token. Clause 7.9.2 defines data types that may be used in the 'extensions' field.

The ASN.1 definition of the 'BindingInfo' data type is given in Annex A as `BindingInfo`.

7.5 Chain

The ‘Chain’ data type represents a sequence of operations representing an aggregation operation or a publishing operation. The ‘Chain' data type contains the data fields listed in Table 9.

Table 9 — Chain

Data field	Description
algorithm	object identifier of algorithm used to compute the value of the chain
links	sequence of data items of type 'Link'

The value of an instance of type 'Chain' is computed by executing the algorithm specified in the 'algorithm' field over the sequence of instances of type 'Link' contained in the 'links' field. For example, an instance of type 'Chain' may be constructed to represent the result of a Merkle hash tree [M80] calculation; in this case, the sequence of instances of type 'Link' represents the computational path from a specific leaf node to the common root. Annex B.3 contains additional discussion regarding the algorithms that may be used in aggregation operations.

The ASN.1 definition of the 'Chain' data type is given in Annex A as `Chain`.

7.6 Link

The ‘Link’ data type represents a single linking operation or a single step of an aggregation or publishing operation. The ‘Link' data type contains the data fields listed in Table 10.

Table 10 — Link

Data field	Description
algorithm	object identifier of algorithm used to compute the value of the link (optional)
identifier	local link identifier, when link is referred to by other links (optional)
members	sequence of data items of type 'Node' to be linked

The value of an instance of type 'Link' is computed by executing the algorithm specified in the 'algorithm' field over the sequence of instances of type 'Node' contained in the 'members' field. If the results of this computation are to be used as input to other operations, they are to be identified by the local identifier stored in the 'identifier' field on instance of type 'Link'. If the 'algorithm' field is not present, the algorithm used to compute the value of the instance of type 'Link' is specified by the context. In the case of an instance of type 'Chain' containing a sequence of instances of type ‘Link’, the algorithm used to compute the value of any such instance of type ‘Link’ for which the ‘algorithm’ field is not present shall be the algorithm specified in the ‘algorithm’ field of the containing instance of type ‘Chain’. Annex B.2 contains additional discussion regarding the algorithms that may be used in linking operations.

The ASN.1 definition of the 'Link' data type is given in Annex A as `Link`.

7.7 Node

The ‘Node’ data type represents a single input element to a linking operation or to a step of an aggregation or publishing operation. The ‘Node data type is defined as containing either (but not both) of the data fields listed in Table 11.

Table 11 — Node

Data field	Description
imprints	input data items
reference	local identifier of an instance of type 'Link' from which to obtain input data items

If the 'imprints' field is present, this instance of type 'Node' contains the actual data items of interest, and their value shall be used as part of further operations involving this instance of type 'Node'.

If the 'reference' field is present and its value is non-zero, this instance of type 'Node' identifies an instance of type ‘Link’ in context. In the case of an instance of type 'Chain' containing a sequence of instances of type ‘Link’, and for a specific such instance of type 'Link' containing a sequence of instances of type 'Node', the presence of the 'reference' field with a non-zero value within such an instance of type 'Node' binds this instance of type 'Node' to the referred instance of type 'Link' identified by the same integer value in its 'identifier' field within the containing instance of type 'Chain'. The value of such an instance of type 'Node' shall be the value of the referred instance of type 'Link'.

If the 'reference' field is present and its value is zero, the value of this instance of type 'Node' is obtained from an external source whose identity depends on the context. In the case of a sequence of instances of type 'Chain' where each instance of type 'Chain' consists of a sequence of instances of type 'Link', and for a specific such instance of type 'Link' containing a sequence of instances of type 'Node', the value of such an instance of type 'Node' containing a 'reference' field with a zero value shall be the value of the instance of type 'Chain' immediately preceding the instance of type 'Chain' that contains the aforementioned instance of type 'Node'.

The ASN.1 definition of the 'Node' data type is given in Annex A as `Node`.

7.8 PublicationInfo

The ‘PublicationInfo’ data type represents the information authenticating the result of a linking operation performed by the TSA against a published value issued by the TSA, and it provides the data items required for identifying and computing this published value. The ‘PublicationInfo' data type contains the data fields listed in Table 12.

Table 12 — PublicationInfo

Data field	Description
pubTime	time value associated with the publishing event (optional)
pubId	identifier of the published value (optional)
pubChains	data authenticating the participation of the result of a linking operation in the publishing event (optional)
sourceId	identifier of the source data participating in the generation of the published value (optional)

The 'pubTime' field, if present, specifies a time value associated with the publishing event for the published value of interest. For example, in the case where the 'pubId' field identifies a print media serial publication, the 'pubTime' field may specify the date of publication for the serial issue containing the published value of interest.

The 'pubId' field, if present, identifies the location of the published value of interest. For example, the 'pubId' field may be a directory name, or a URI resolving to the location of the published value of interest.

The 'pubChains' field, if present, is a sequence of instances of type ‘Chain’. For a given linking operation performed by the TSA, this field is defined according to the publishing operation performed by the TSA for the time interval properly containing the time value of the aforementioned linking operation.

The 'sourceId' field, if present, identifies the source data for the published value, that is, the results of linking operations generated and maintained by the TSA that were used to generate the published value, as well as additional intrinsic properties of the publishing event (for example, frequency of publishing, time interval covered in the repository of links maintained by the TSA, and so on).

The ASN.1 definition of the 'PublicationInfo' data type is given in Annex A as `PublicationInfo`.

7.9 Extensions

7.9.1 Time-stamp request extensions

7.9.1.1 Hash extension

A time-stamp requester may wish to submit for time-stamping more than one hash value derived from a single data item. To enable the submission of multiple hash values the hash extension is defined. This extension is carried in the 'extensions' field of both the time-stamp request sent by a requester to a TSA and in the 'extensions' field of the resulting 'TSTInfo' structure formed by the TSA and returned in a time-stamp token to the requester.

If this extension is present in a time-stamp request and the TSA is able to process it, then the TSA shall cryptographically bind both the hash value in the time-stamp request message specified in the 'messageImprints' field of the time-stamp request and those included in this extension to the time value it assigns to the 'TSTInfo' structure in the resulting time-stamp token, and shall reproduce the contents of this extension unmodified in the aforementioned 'TSTInfo' structure.

The ASN.1 definition of the hash extension is given in Annex A as `extHash`.

7.9.1.2 Method extension

A time-stamp requester may wish to indicate to a specific TSA which time-stamping method to use when forming the eventual time-stamp token. To enable a requester to indicate to a specific TSA which time-stamping method to use in forming the resulting time-stamp token, the method extension is defined. This extension is carried in the 'extensions' field of both the time-stamp request sent by a requester to a TSA and in the 'extensions' field of the resulting 'TSTInfo' structure formed by the TSA and returned in a time-stamp token to the requester.

If this extension is present in a time-stamp request and the TSA is able to process it, then the TSA shall attempt to fulfil the request for the specified method, or return an error indicating the method is not available. If the requester specifies more than one possible method, the TSA may select one of the suggested methods for use in forming the time-stamp token. If this extension is not present, then the TSA shall use its default time-stamping mechanism.

The ASN.1 definition of the method extension is given in Annex A as `extMethod`.

7.9.1.3 Renewal extension

A time-stamp requester may wish to indicate to the TSA that the current time-stamp request is a renewal of a time-stamp on data that were already time-stamped in the past, so that the validity period of the existing time-stamp token is preserved into the future (for example, when a hash-function referred to within the time-stamp token is close to being broken by new attacks or available computing power). To enable the submission of renewal time-stamp requests that include an existing time-stamp token, the renewal extension is defined. This extension is carried in the 'extensions' field of both the time-stamp request sent by a requester to a TSA and in the 'extensions' field of the resulting TSTInfo structure formed by the TSA and returned to the requester.

If this extension is present in a time-stamp request and the TSA is able to process it, then the TSA shall reproduce the contents of this extension unmodified in the 'TSTInfo' structure in the resulting time-stamp token.

The ASN.1 definition of the renewal extension is given in Annex A as `extRenewal`.

7.9.2 BindingInfo extensions

7.9.2.1 Name extension

A TSA producing linked tokens may identify each step of the overall time-stamping process by a distinct name, for auditing or record-keeping purposes. To enable the identification of these steps within the 'BindingInfo' structure, the name extension is defined. For example, a TSA may include this extension when the overall time-stamping process is implemented by a co-operating set of distinct processes.

The ASN.1 definition of the name extension is given in Annex A as `extName`.

7.9.2.2 Time extension

A TSA producing linked tokens may record the time value at which each step in the overall time-stamping process occurred, for auditing or record-keeping purposes. To enable the recording of these time values within the 'BindingInfo' structure, the time extension is defined. For example, a TSA may include this extension when the overall time-stamping process is implemented by a co-operating set of distinct processes.

The ASN.1 definition of the time extension is given in Annex A as `extTime`.

7.9.2.3 Publication extension

A requester of time-stamping services may carry out a request-response protocol with the issuing TSA or other TTP that has access to the issuing TSA's links, allowing it to extend an existing time-stamp token to a published value. To enable the recording of the data extending a time-stamp token to a published value within the 'BindingInfo' structure, the publication extension is defined.

The data items of this extension may be used to verify the validity of a time-stamp token against the published value issued by the TSA and against the respective TSA publishing event. If the value published by the TSA is available in an authenticated manner from widely witnessed sources, verification of the time-stamp token against such a published value and the time value of the respective publishing event may be conducted independently of the issuing TSA that generated the original time-stamp token. Note that the time values associated with TSA publishing events represent a time scale that is coarser than the one defined by the time values assigned to the 'genTime' field of the encapsulated 'TSTInfo' structure of the time-stamp tokens.

The publication extension allows for the presence of multiple fields of type 'PublicationInfo' to provide for published values at different intervals offering different time granularity, for example, daily or weekly.

The ASN.1 definition of the hash extension is given in Annex A as `extPublication`.

Note: The 'BindingInfo' data type contains a 'publish' field that may be used for limited support of published values by a conforming TSA generating linked tokens. The 'PublicationInfo' data type as used within a publication extension supports a significantly larger feature set and deprecates the use of the 'publish' field within the 'BindingInfo' data type.

8 Generating a time-stamp token

8.1 General

In order to acquire a time-stamp token over given data, the time-stamp requester shall submit a time-stamp request to the TSA, as described in Clause 6.1. The requester may specify the method extension with the `tsp-req-link` object identifier, as defined in Annex A in the 'extensions' field for the purpose of requesting time-stamp tokens using the 'DigestedData' encapsulation. Alternatively, the requester may specify the method extension with the `tsp-req-link-ds` object identifier, as defined in Annex A in the 'extensions' field for the purpose of requesting time-stamp tokens using the 'SignedData' encapsulation.

If a method extension is specified in the 'extensions' field of the time-stamp request, and the method extension contains one or more method identifiers corresponding to methods supported by the TSA, a TSA producing linked tokens shall fulfil the request using one of the specified supported methods. If no method extension is specified in the 'extensions' field of the time-stamp request, a TSA producing linked tokens shall use either the 'DigestedData' encapsulation or the 'SignedData' for the generated tokens, according to its service policy.

The time-stamp response returned by a conformant TSA may include a 'TimeStampToken' structure, if the request is granted. A conformant TSA produces linked time-stamp tokens according to the steps described below.

Time-stamp information is first generated by the TSA according to the 'TSTInfo' structure defined in Clause 7.2. This structure contains the message digest of the data to be time-stamped, the time value assigned by the TSA, and other relevant information. The 'tsa' field of the 'TSTInfo' structure shall be present; the 'tsa' field is required for time-stamp verification. The 'extensions' field of the 'TSTInfo' shall be present if the corresponding time-stamp request included data items in the 'extensions' field.

Subsequently, the TSA shall encode the generated 'TSTInfo' structure using DER encoding rules, and store the resulting octets in the 'eContent' fileld of an 'EncapsulatedContentInfo' structure.

The TSA shall then compute one or more hash values on this 'eContent' field using one or more hash-functions. The resulting hash value or values are inserted in the 'msgImprints' field of the 'BindingInfo' structure defined in Clause 7.4.

The TSA shall use the contents of the 'msgImprints' field in a linking operation in combination with links from prior linking operations, possibly in aggregation with 'msgImprints' fields from other concurrent time-stamp requests (that is, time-stamp requests receiving the same time value in the resulting 'TSTInfo' structure), and shall populate the 'links' and 'aggregate' fields of the 'BindingInfo' structure accordingly.

Finally, the TSA shall generate the time-stamp token by encapsulating the 'TSTInfo' structure and its respective 'BindingInfo' structure in a 'TimeStampToken' structure. The TSA may perform either of two types of encapsulation:

- 'DigestedData' encapsulation, in which case the 'contentType' field shall contain the value `id-digestedData` and the 'content' field shall contain a value of type 'DigestedData'; this encapsulation may be used when the time-stamp request includes a method extension in the 'extensions' field with value the object identifier `tsp-req-link`.

- 'SignedData' encapsulation, in which case the 'contentType' field shall contain the value `id-signedData` and the 'content' field shall contain a value of type 'SignedData'; this encapsulation may be used when the time-stamp request includes a method extension in the 'extensions' field with value the object identifier `tsp-req-link-ds`.

8.2 DigestedData encapsulation

A TSA generating time-stamp tokens using the 'DigestedData' encapsulation method shall first perform the steps described in Clause 8.1 and place the resulting 'EncapsulatedContentInfo' structure in the 'encapContentInfo' field of the 'DigestedData' structure (as defined in Annex A). Once all fields of the 'BindingInfo' structure have been populated, the TSA shall encode the 'BindingInfo' structure using DER encoding rules, and store the resulting octets in the 'digest' field of the 'DigestedData' structure.

Finally, the TSA shall insert the object identifier corresponding to this encapsulation method `tsp-digestedData` (as defined in Annex A) into the 'digestAlgorithm' field of the 'DigestedData' structure.

8.3 SignedData encapsulation

A TSA may integrate digital signatures with a time-stamping system producing linked tokens. By linking together tokens signed independently, the TSA can verifiably demonstrate the order in which the individually signed tokens were issued.

The following object identifier identifies the signed data content type:

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

The 'SignedData' structure contains one or more 'SignerInfo' structures, which may in turn contain additional signed or unsigned attributes, and is defined in ISO/IEC 18014-1 Annex A as follows:

```
SignedData    ::= SEQUENCE {
    version             CMSVersion,
    digestAlgorithms    DigestAlgorithmIdentifiers,
    encapContentInfo    EncapsulatedContentInfo,
    certificates        [0] IMPLICIT CertificateSet OPTIONAL,
    crls                [1] IMPLICIT CertificateRevocationLists OPTIONAL,
    signerInfos         SignerInfos
}
SignerInfos ::= SET OF SignerInfo
SignerInfo ::= SEQUENCE {
    version             CMSVersion,
    sid                 SignerIdentifier,
    digestAlgorithm     DigestAlgorithmIdentifier,
    signedAttrs         [0] IMPLICIT SignedAttributes OPTIONAL,
    signatureAlgorithm  SignatureAlgorithmIdentifier,
    signature           SignatureValue,
    unsignedAttrs       [1] IMPLICIT UnsignedAttributes OPTIONAL
}
SignedAttributes ::= SET SIZE (1..MAX) OF Attribute
UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute
```

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

A TSA generating time-stamp tokens using the 'SignedData' encapsulation method shall use a 'SignedData' structure to encapsulate both the signature data and linking data. The TSA shall first perform the steps described in Clause 8.1, and then place the resulting 'EncapsulatedContentInfo' structure in the 'encapContentInfo' field of the 'SignedData' structure. Once all fields of the 'BindingInfo' structure have been populated, the TSA shall encode the 'BindingInfo' structure as an octet string using DER encoding rules, and insert it as an element of the 'attrValues' field in an 'Attribute' structure. The TSA shall also insert the object identifier corresponding to this encapsulation method `tsp-signedData`, as defined in Annex A into the 'attrType' field of the 'Attribute' structure. The TSA shall then insert the resulting 'Attribute' structure in the 'signedAttrs' field of a 'SignerInfo' structure, and complete the signing process according to the signing procedures defined in [CMS].

Inserting the 'BindingInfo' structure as a signed attribute within the time-stamp token preserves the validity of the encapsulated 'TSTInfo' structure beyond the lifetime of the digital certificate used to form the digital signature. Additionally, this time-stamp token may be easily transformed into a time-stamp token using 'DigestedData' encapsulation and containing 'TSTInfo' and 'BindingInfo' structures identical to those in the original time-stamp token. This newly formed time-stamp token contains the same linking data as the original time-stamp token and omits the digital signature parts.

8.4 Security considerations

Conformant TSAs producing linked tokens shall employ strong cryptographic primitives in operations related to the generation of time-stamp tokens, links, and related data items during aggregation, linking, and publishing operations, and shall replace and upgrade these cryptographic primitives as needed before they become insufficiently strong to provide the security guarantees required of the service.

Conformant TSAs producing linked tokens should employ multiple hash-functions concurrently in the calculation of 'msgImprints' field of the 'BindingInfo' structure of a time-stamp token, in the aggregation process, and in the linking process so that the resulting time-stamp token and the link generated by the TSA are insulated from the cryptographic failure of any single hash-function.

9 Verifying a time-stamp token

9.1 General

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

- the time-stamp token is syntactically well-formed,
- the value of the 'messageImprint' field of the encapsulated 'TSTInfo' structure matches the value of a message imprint evaluated over the document subject to scrutiny,
- the value of every 'messageImprint' field inside a hash extension in the 'extensions' field of the encapsulated 'TSTInfo' structure, if present, matches the value of a message imprint evaluated over the document subject to scrutiny,
- the policy under which the token is issued is acceptable for the intended usage,

and by carrying out additional verification steps appropriate to the encapsulation method specified in the time-stamp token.

9.2 DigestedData encapsulation

In the case of a time-stamp token using the ‘DigestedData’ encapsulation, the verifier shall submit a verify request to the issuing TSA or other TTP that has access to the issuing TSA’s links, as described in Clause 6.3, using a data integrity and data origin authentication protected channel. If there is no link in the TSA’s or TTP’s repository of links associated with the same time value as the encapsulated ‘TSTInfo’ structure of the submitted time-stamp token, verification fails. Otherwise, the issuing TSA or other TTP fulfilling the request shall proceed to verify that the 'msgImprints' field in the 'BindingInfo' structure of the time-stamp token matches the hash values computed over the encapsulated 'TSTInfo' structure. If not, verification fails; otherwise, if the ‘aggregate’ field is present in the ‘BindingInfo’ structure, the issuing TSA or TTP fulfilling the request shall carry out the computation of the value of the ‘aggregate’ field (assuming that a zero ‘reference’ value within an instance of type 'Node' field refers to the ‘msgImprints’ field of the ‘BindingInfo’ structure), and apply the linking algorithm on this result and the ‘links’ field in order to verify whether the result matches the value of the archived link for the time value in the time-stamp token’s encapsulated ‘TSTInfo’ structure; if the ‘aggregate’ field is missing, the issuing TSA or TTP fulfilling the request shall apply the linking algorithm on the ‘msgImprints’ and ‘links’ fields of the ‘BindingInfo’ structure in the same manner, and carry out the same verification step on the resulting value. In any event, an appropriate 'status' field is included in the verify response and the submitted time-stamp token is reproduced unmodified, as described in Clause 6.4.

9.3 SignedData encapsulation

In the case of a time-stamp token using the ‘SignedData’ encapsulation, the time-stamp verifier shall either:

- carry out the same additional verification steps as described for a time-stamp token created using the 'DigestedData' encapsulation, namely, submit a verify request to the issuing TSA or other TTP that has access to the issuing TSA’s links as described in Clause 6.3, using a data integrity and data origin authentication protected channel, and in such a case, the TSA or TTP fulfilling the request shall access the same components of the time-stamp token and carry out the same verification steps as described for a time-stamp token using the 'DigestedData' encapsulation in Clause 9.2;

or:

- treat the time-stamp token as an independent token and perform additional verification steps in the same manner as for time-stamp tokens with digital signatures described in ISO/IEC 18014-2.

9.4 Security considerations

For the purpose of supporting verification requests against the links maintained by the TSA or other TTP, the TSA or other TTP fulfilling the request should fix in time the links and related data necessary for time-stamp verification and for auditing purposes, before the cryptographic primitives used in linking, aggregation and publishing operations that were used to generate the links and related data are considered weak. For example, this may be accomplished by time-stamping the values of the links and related data of interest, and by renewing the resulting time-stamp tokens as needed, and as described in Clause 11.

10 Extending a time-stamp token

For the purpose of extending a time-stamp token to a published value, the entity in possession of the time-stamp token may submit an extend request to the issuing TSA or other TTP that has access to the issuing TSA’s links, as described in Clause 6.5, using a data integrity and data origin authentication protected channel. If the TSA or other TTP fulfilling the request supports the extend operation, it shall carry out the verification steps on the submitted time-stamp token according to the 'DigestedData' encapsulation as described in Clause 9.2. If the validity of the token is verified, the TSA or TTP fulfilling the request shall proceed to examine whether there exists a published value that may be referred to by the submitted time-stamp token; in that case, the TSA or TTP fulfilling the request shall populate a 'PublicationInfo’ structure with data items sufficient to authenticate the time-stamp token against the published value according to Clause 7.9.2.3, and will return an updated time-stamp token to the submitter within an extend response as described in Clause 6.6. If the extend operation is not available, if the submitted time-stamp token fails the verification steps, or if a published value is not yet available for the submitted time-stamp token, then the extend request is not granted, an appropriate 'status' field is included in the extend response and the submitted time-stamp token is reproduced unmodified as described in Clause 6.6.

11 Renewing a time-stamp token

11.1 General

As defined in ISO/IEC 18014-1, renewal is a variant of the basic time-stamp operation where an existing time-stamp token over previously time-stamped data items is explicitly incorporated as data to be bound to a newly generated time-stamp token over the same data items with a new (current) time value. By incorporating the pre-existing time-stamp token in the generation of the new time-stamp token, and assuming that security considerations are satisfied appropriately, the validity period of the earlier time-stamp token over the time-stamped data items is extended by the coverage of the new time-stamp token.

In the case of linked tokens, renewal may be necessary when the cryptographic function used to bind the time value to the data is still trusted, but there is strong evidence that it will become vulnerable in the near future (e.g. when a hash-function is close to being broken by new attacks or available computing power). Renewal has to be performed before such a condition makes the original time-stamp token void.

For the purpose of renewing a time-stamp token over data in his possession, the requester shall submit a time-stamp request to the TSA as described in Clause 6.1 and include the existing time-stamp token in a renewal extension within the 'extensions' field of the time-stamp request according to Clause 7.9.1.3. The TSA shall return a time-stamp response according to Clause 7.9.1.3.

11.2 Renewal and verify operation

As described in ISO/IEC 18014-1, verification of a renewed time-stamp token is performed in such a way that the outermost (most recently issued) time-stamp token is verified at the current time, while the enclosed (previously issued) time-stamp token is verified with respect to the time of issuance of the enclosing time-stamp token. In the event of multiple nested renewals, each nested time-stamp token is verified at the time of issuance of the subsequent (enclosing) renewal time-stamp token, until the outermost time-stamp token is verified at the current time. In the event where all time-stamp tokens are generated by TSAs conforming to this part of ISO/IEC 18014, the innermost (oldest) time-stamp token is verified for the current time if the following are true:

- the outermost and all nested time-stamp tokens are verified according to the verification steps for 'DigestedData' encapsulation according to Clause 9,
- the cryptographic functions listed in each enclosed time-stamp token are trusted at the time of the issuance of the immediately enclosing renewal time-stamp token, and
- the repositories of links maintained by the TSAs issuing renewal time-stamp tokens are trusted to have existed prior to any time when cryptographic primitives required for their algorithmic verification may have been considered weak.

11.3 Renewal and extend operation

Renewal considerations also affect time-stamp tokens that have been extended to a published value. While such time-stamp tokens may be verified against a published value independently of the issuing TSA's links, as long as the published value is available in an authenticated manner from widely witnessed sources, this verification process is subject to security considerations related to the strength of the cryptographic primitives used in such a process. Extending a time-stamp token to a published value, followed by a renewal operation and a subsequent extend operation to a new published value on the renewal time-stamp token, preserves the ability to assert validity of the original time-stamp token independently of the issuing TSAs' links. For example, a time-stamp token t_1 is extended to a published value v_1 corresponding to a TSA publishing event at time value p_1, resulting in a time-stamp token t_{1e}; subsequently time-stamp token t_{1e} is renewed resulting in a renewal time-stamp token t_2, and this time-stamp token t_2 is further extended to a published value v_2 corresponding to TSA publishing event at time value p_2, resulting in an extended time-stamp token t_{2e}; if the following hold:

- the time-stamp token t_{1e} is verified against published value v_1,

- the renewal token t_{2e} is verified against published value v_2,
- the cryptographic primitives related to the verification of t_{1e} are not considered weak at time p_2, and
- the cryptographic primitives related to the verification of t_{2e} are not considered weak at current time,

then the validity of the statement that token t_{1e} is valid at asserted time value p_1 may be extended beyond the current time, without requiring any additional reference to the links maintained by the TSAs, even if the cryptographic primitives related to the verification of t_{1e} against published value v_1 are considered weak at the current time.

Annex A
(normative)

ASN.1 Module for time-stamping

This module contains ASN.1 based on the current ASN.1 standards.

```
TimeStampProtocol-3 {
    iso(1) standard(0) time-stamp(18014) modules(0) part3(3)}
    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
              FROM CryptographicMessageSyntax {
                   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                   smime(16) modules(0) cms(1) };
-- Supporting Definitions --
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] }
CONTENT ::= TYPE-IDENTIFIER      -- ISO/IEC 8824-2, Annex A
OIDS ::= CLASS {
    &id             OBJECT IDENTIFIER  UNIQUE
}
WITH SYNTAX { OID &id }
POLICY ::= OIDS  -- Supported TSA policies
METHOD ::= OIDS  -- TSA Methods
```

```
-- Time-stamp Request --
TimeStampReq ::=  SEQUENCE {
    version          Version,
    messageImprint   MessageImprint,
    reqPolicy        TSAPolicyId OPTIONAL,
    nonce            Nonce OPTIONAL,
    certReq          BOOLEAN DEFAULT FALSE,
    extensions       [0] Extensions OPTIONAL
}
Version ::= INTEGER { v1(1) }
MessageImprint ::= SEQUENCE  {
    hashAlgorithm    DigestAlgorithmIdentifier,
    hashedMessage    OCTET STRING
}
MessageImprints ::= SEQUENCE SIZE (1..MAX) OF MessageImprint
DigestAlgorithmIdentifier ::= AlgorithmIdentifier {{ DigestAlgorithms }}
DigestAlgorithms ALGORITHM ::= {
    { OID id-ripemd160  PARMS NULL } |
    { OID id-sha1 PARMS NULL } |
    { OID id-sha256 PARMS NULL } |
    { OID id-sha224 PARMS NULL } |
    { OID id-sha384 PARMS NULL } |
    { OID id-sha512 PARMS NULL } ,
    --
    ... -- Expect additional digest algorithms --
}
TSAPolicyId ::= POLICY.&id({TSAPolicies})
TSAPolicies POLICY ::= {
      --
    ...  -- Any supported TSA policy --
}
Nonce ::= INTEGER  -- Expect large values
-- Time-stamp Response --
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, the requester
                               -- receives a receipt that the request
                              -- has been received
    revocationWarning          (4),   -- a revocation is imminent
    revocationNotification     (5)    -- notification that a revocation
                                      -- has 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 additional information requested
                               -- could not be understood or is not available
    systemNotAvailable  (24),  -- system is not available
    systemFailure       (25),  -- system failure
    verificationFailure (27)   -- verification of time stamp has failed
}
-- Verify Request --
VerifyReq ::=     SEQUENCE {
    version       Version,
    tst           TimeStampToken,
    requestID     [0] OCTET STRING OPTIONAL
}
-- Verify Response --
VerifyResp ::=      SEQUENCE {
    version       Version,
    status        PKIStatusInfo,
    tst           TimeStampToken,
    requestID     [0] OCTET STRING OPTIONAL
```

```
}
-- Extend Request --
ExtendReq ::=      SEQUENCE {
    version        Version,
    tst            TimeStampToken,
    requestID      [0] OCTET STRING OPTIONAL
}
-- Extend Response --
ExtendResp ::=     SEQUENCE {
    version        Version,
    status         PKIStatusInfo,
    tst            TimeStampToken,
    requestID      [0] OCTET STRING OPTIONAL
}
-- Object Identifiers ---
   -- 18014-3 arc --
tsp-lt  OBJECT IDENTIFIER ::= { iso(1) standard(0) time-stamp(18014) lt(3) }
tsp-req-link OBJECT IDENTIFIER ::= {tsp-lt link(1)}
tsp-req-link-ds OBJECT IDENTIFIER ::= {tsp-lt link-ds(2)}
tsp-ext-name OBJECT IDENTIFIER ::= { tsp-lt name(5) }
tsp-ext-time OBJECT IDENTIFIER ::= { tsp-lt time(6) }
tsp-ext-publication OBJECT IDENTIFIER ::= {tsp-lt publication(7) }
tsp-digestedData OBJECT IDENTIFIER ::= { tsp-lt digestedData(8) }
tsp-signedData OBJECT IDENTIFIER ::= { tsp-lt signedData(9) }
   -- 18014-1 arc --
tsp-ext OBJECT IDENTIFIER ::= { iso(1) standard(0) time-stamp(18014) ext(1) }
tsp-ext-hash OBJECT IDENTIFIER ::= { tsp-ext hash(1) }
tsp-ext-meth OBJECT IDENTIFIER ::= { tsp-ext meth(2) }
tsp-ext-renewal OBJECT IDENTIFIER ::= { tsp-ext renewal(3) }
   -- other --
der OBJECT IDENTIFIER ::= {
    joint-iso-itu-t asn1(1) ber-derived(2) distinguished-encoding(1) }
id-ct-TSTInfo OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) ct(1) 4 }
id-digestedData OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 5 }
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-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 }
id-signedData OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) signedData(2) }
-- TSTInfo --
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
}
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 -- }))
-- Time-stamp Token --
TimeStampToken ::= SEQUENCE {
    contentType         CONTENT.&id({Contents}),
    content          [0] EXPLICIT CONTENT.&Type({Contents}{@contentType})
}
```

```
Contents CONTENT ::= {
{ DigestedData IDENTIFIED BY id-digestedData } |
{ SignedData IDENTIFIED BY id-signedData },
--
... -- expect additional time-stamp encapsulations --
}
DigestedData ::= SEQUENCE {
    version             CMSVersion,
    digestAlgorithm     DigestAlgorithmIdentifier,
    encapContentInfo    EncapsulatedContentInfo,
    digest              Digest
}
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)
Digest ::= OCTET STRING -- (CONTAINING BindingInfo ENCODED BY der)
-- Binding Info --
BindingInfo ::= SEQUENCE {
    version             Version,
    msgImprints         MessageImprints,
    aggregate           [0] Chains OPTIONAL,
    links               Links,
    publish             [1] Chains OPTIONAL,
    extensions          [2] Extensions OPTIONAL
}
-- Chain --
Chain ::= SEQUENCE {
    algorithm ChainAlgorithmIdentifier,
    links     Links
}
Chains ::= SEQUENCE SIZE (1..MAX) OF Chain
ChainAlgorithmIdentifier ::= AlgorithmIdentifier {{ ChainAlgorithms }}
```

```
ChainAlgorithms ALGORITHM ::= {
    --
    ... -- expect additional chain algorithms --
}
-- Link --
Link ::= SEQUENCE {
    algorithm       [0] LinkAlgorithmIdentifier OPTIONAL,
    identifier      [1] INTEGER OPTIONAL,
    members         Nodes
}
Links ::= SEQUENCE SIZE (1..MAX) OF Link
LinkAlgorithmIdentifier ::= AlgorithmIdentifier {{ LinkAlgorithms }}
LinkAlgorithms ALGORITHM ::= {
    --
    ... -- expect additional link algorithms --
}
-- Node --
Node ::= CHOICE {
    imprints  [0] Imprints,
    reference [1] INTEGER
}
Nodes ::= SEQUENCE SIZE (1..MAX) OF Node
Imprint ::= OCTET STRING
Imprints ::=  SEQUENCE SIZE (1..MAX) OF Imprint
-- Publication Info --
PublicationInfo ::= SEQUENCE {
    pubTime    GeneralizedTime OPTIONAL,
    pubId      [0] GeneralName OPTIONAL,
    pubChains  [1] Chains      OPTIONAL,
    sourceId   [2] GeneralName OPTIONAL
}
-- Time-stamp Request Extensions --
TSExtensions EXTENSION ::= {
    extHash    |
    extMethod  |
    extRenewal,
     --
    ...  -- expect additional extensions --
}
```

```
extHash EXTENSION ::= {SYNTAX ExtHash IDENTIFIED BY tsp-ext-hash}
ExtHash ::= SEQUENCE SIZE(1..MAX) OF MessageImprint
extMethod EXTENSION ::= {SYNTAX ExtMethod IDENTIFIED BY tsp-ext-meth}
ExtMethod ::= SEQUENCE SIZE(1..MAX) OF Method
Method ::= METHOD.&id({Methods})
Methods METHOD ::= {
    tsp-digestedData |
    tsp-signedData,
    --
    ...  -- any time-stamping method  --
}
extRenewal EXTENSION ::= {SYNTAX ExtRenewal IDENTIFIED BY tsp-ext-renewal}
ExtRenewal ::= TimeStampToken
-- Binding Info Extensions --
BExtensions EXTENSION ::= {
    extName    |
    extTime  |
    extPublication,
     --
    ...  -- expect additional extensions --
}
extName EXTENSION ::= { SYNTAX ExtName IDENTIFIED BY tsp-ext-name }
ExtName ::= GeneralName
extTime EXTENSION ::= { SYNTAX ExtTime IDENTIFIED BY tsp-ext-time }
ExtTime ::= GeneralizedTime
extPublication EXTENSION ::= {
    SYNTAX ExtPublication IDENTIFIED BY tsp-ext-publication
}
ExtPublication ::= SEQUENCE SIZE (1..MAX) OF PublicationInfo

END  -- TimeStampProtocol 3 --
```

Annex B
(informative)

Additional discussion

B.1 Introduction

This annex provides additional explanatory text for use by conformant TSAs when choosing specific processes and techniques for the linking, aggregation, and publishing operations.

B.2 Linking

B.2.1 General

Linking is the means whereby a hash value is cryptographically linked to other hash value representations, in order to both express the causal ordering of time-stamping events and to serve as a witness to all previously-linked events. Linking functions must be cryptographically strong, in the sense that once a new hash value is computed by a linking operation, it cannot be changed or deleted without detection. Linking successive hash values together demonstrates the sequence in which they were generated, as a means for verifying the correct application of time values. The linking process also serves to make each new hash value witness to all preceding hash values, in the sense that previous hash values cannot be altered without successive calculations being affected. Each hash value identifies an event which, when widely witnessed, can be used to verify the correctness of all previously submitted events.

In practice, the TSA generates a hash value representing either a single encapsulated 'TSTInfo' structure or an aggregation of multiple encapsulated 'TSTInfo' structures containing the same time value, and performs a linking operation together with data items representing the previously linked hash values, such that the newly generated hash value by the linking operation in effect forms a running summary of all previously linked hash values. The TSA maintains a repository of such values produced by the linking process, referred to as links, in order to support future verify and extend requests, as well as publishing and auditing operations.

It is recommended that multiple hash-functions be used concurrently in the linking process so that the calculation of links is insulated from the cryptographic failure of any single hash-function. For example, a TSA may perform a linking operation by first computing the SHA256 hash value over the concatenated input data of the linking operation, then computing the RIPEMD hash value over the same input data, and finally concatenating the two computed hash values together to generate the value of the link that is the result of the linking operation. In subsequent clauses, it is assumed that whenever a single hash value representing some data item is mentioned, a TSA may instead utilize multiple hash values representing the same data item, computed using distinct hash-functions.

B.2.2 Linear chain linking

Using this method of linking, each link in the linear chain is formed by concatenating the value of the most recently generated link in the linear chain with the current hash value to be linked, and using the resulting bitstring as input to a hash-function. The output of the hash-function is the value of the new link that becomes the running summary of all linking operations so far, and it is stored in the repository of links maintained by the TSA; the value of the previous link that was used as input to the linking operation is returned to the requester within the 'links' field of the respective 'BindingInfo' structure.