INTERNATIONAL STANDARD

ISO 14533-4

First edition
2019-08

Processes, data elements and documents in commerce, industry and administration — Long term signature profiles —

Part 4: Attributes pointing to (external) proof of existence objects used in long term signature formats (PoEAttributes)

Processus, éléments d'informations et documents dans le commerce, l'industrie et l'administration — Profils de signature à long terme —

Partie 4: Attributs pointant vers des objets externes de la Preuve de l'existence utilisés dans les formats de la signature à long terme

Reference number
ISO 14533-4:2019(E)

Contents

Page

Foreword .... **iv**

Introduction .... **v**

1 Scope **1**

2 Normative references **1**

3 Terms and definitions **2**

4 PoE attributes **4**

4.1 General concept of PoE 4

4.2 Abstract attribute PoEAttribute 5

4.3 LTI ***PoEAttribute*** instance based on IETF RFC 3161 timestamp or IETF RFC 4998/IETF RFC 6283 evidence record 8

4.4 ERS ***PoEAttribute*** instance based on IETF RFC 4998/IETF RFC 6283 evidence record 12

4.5 TStOCSP ***PoEAttribute*** instance 12

4.6 Attribute PoEHashIndex 13

4.7 Attribute preservation-integrity-list 14

5 Types of PoE objects with their essential fields **16**

5.1 General 16

5.2 PoE object of status at ***thisUpdate*** time value based on ***CertHash*** OCSP ***SingleResponse*** extension 17

5.3 PoE object supported by LTI PoEAttribute or ERS PoEAttribute 18

Annex A (normative) **ASN.1 module** **19**

Annex B (normative) **Definition of the *CertHash* OCSP *SingleResponse* extension** **20**

Annex C (normative) **Signature timestamp as a timestamp through OCSP** **21**

Annex D (normative) **Syntax of the ASN.1 object location in ZIP, PDF container or in DER encoded ASN.1 object** **23**

Annex E (normative) **Use of the PoE objects** **26**

Annex F (informative) **Location of DTId in the digital signature** **32**

Annex G (informative) **Media type registrations** **33**

Annex H (informative) **Evidence record syntax object** **34**

Bibliography **36**

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.

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 ISO documents 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).

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. 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).

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.

This document was prepared by Technical Committee ISO/TC 154, *Processes, data elements and documents in commerce, industry and administration*.

A list of all parts in the ISO 14533 series can be found on the ISO website.

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.

Introduction

This document provides detailed information associated with the analysis, selection and implementation of procedures associated with long term signatures. The development of this document is a result of organizational requests to receive information of already existing objects defined in technology standards, technical reports, and industry best practices for electronic signatures verifiable for a long term.

The purpose of this document is to ensure the interoperability of implementations with respect to long term signatures that make electronic signatures verifiable for a long term. This document clarifies conditions used in the validation procedure to provide a complete and unalterable result.

Processes, data elements and documents in commerce, industry and administration — Long term signature profiles —

Part 4: Attributes pointing to (external) proof of existence objects used in long term signature formats (PoEAttributes)

IMPORTANT — The electronic file of this document contains colours which are considered to be useful for the correct understanding of the document. Users should therefore consider printing this document using a colour printer.

1 Scope

This document specifies the elements defined in the international standards of ISO/ITU-T, ETSI and IETF RFC that enable at least a proof of existence of data objects and digital signatures and the preservation of the validity status of digital signatures over a long period of time used in validation.

It provides the definitions of the proof of existence (PoE) attributes and clarification of the usage of (external) PoE objects, with digital signatures and trusted time values, which have already existed and can be used by the PoE attributes pointing to (external) PoE objects used in long term signature validation or preservation.

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 8825-1[1], *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) — Part 1*

ISO/IEC 9594-8[2], *Information technology — Open Systems Interconnection — The Directory — Part 8: Public-key and attribute certificate frameworks*

ISO 32000-2, *Document management — Portable document format — Part 2: PDF 2.0*

ETSI EN 319 122-1, V1.1.1:2016-04, *Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures*

IETF RFC 3161[3], *Timestamp Protocol (TSP)*

IETF RFC 6960[4], *Online Certificate Status Protocol (OCSP)*

IETF RFC 4648[5], *The Base16, Base32, and Base64 Data Encodings*

1) Also known as ITU-T Recommendation X.690.

2) Also known as ITU-T Recommendation X.509.

3) Available at https://tools.ietf.org/html/3161.

4) Available at https://tools.ietf.org/html/6960.

5) Available at https://tools.ietf.org/html/4648.

IETF RFC 4998[6)], *Evidence Record Syntax (ERS)*

IETF RFC 6283[7)], *Extensible Markup Language Evidence Record Syntax (XMLERS)*

IETF RFC 5652[8)], *Cryptographic Message Syntax (CMS)*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9594-8, ISO 32000-2, ISO/IEC 8825-1, IETF RFC 3161, IETF RFC 6960 and the following 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
digital signature
data appended to, or cryptographic transformation of, a data string that proves the origin and the integrity of the data string and protects against forgery, e.g. by the recipient of the data string

Note 1 to entry: Digital signatures in the present document cover also electronic signatures, advanced electronic signatures, qualified electronic signatures, electronic seals, advanced electronic seals, qualified electronic seals, electronic time stamps and qualified electronic time stamps as per Regulation (EU) No 910/2014[6] and ISO/IEC 9594-8 certificate, CRL (see ISO/IEC 9594-8) or OCSP response and signatures defined in profiles ISO 14533-1, ISO 14533-2 or ISO 14533-3.

[SOURCE: ISO/IEC 7816-4:2013, 3.21, modified — Note 1 to entry has been added.]

3.2
document identifier
DId
indirect identifier for machine processing in the form of DER encoded ASN.1 type `MessageImprint` defined in IETF RFC 3161 covering the electronic document

3.3
document signature identifier
DSId
indirect identifier of signature of a document for machine processing in the form of DER encoded ASN.1 type `MessageImprint` defined in IETF RFC 3161 covering the DER encoded result of the asymmetric signature algorithm

EXAMPLE ECDSA or RSA result included in the *digital signature* (3.1) of the signed electronic document.

Note 1 to entry: DSId is mainly used for indirect machine processing identification of the electronic document which is electronically signed, e.g. DSId is used for PDF document identification if PDF file contains many versions of PDF document created by including incremental updates (see ISO 32000-2:2017, 7.5.6 for details) after more than one PDF document timestamps (see ISO 32000-2:2017, 12.8.5) or by including incremental updates after PDF signature or after PDF document timestamp.

Note 2 to entry: Indirect identifier means a relatively unique changeable hash value changing according to the used hash algorithm. A hash collision is when two different input strings of a hash function produce the same hash result.

6) Available at https://tools.ietf.org/html/4998.

7) Available at https://tools.ietf.org/html/6283.

8) Available at https://tools.ietf.org/html/5652.

3.4
document type identifier
DTId
sequence of the characters associated with the electronic document used for determining the format and interpretation of the electronic document

Note 1 to entry: DTId is crucial for correct interpretation of the content of electronic document protected by *digital signature* (3.1). DTId is implemented as the file name extension or the value of the content type (see IETF RFC 2231 or IETF RFC 2045), whose value is included in the fields protected by the digital signature (see Annex F).

3.5
end-of-line marker
EOL marker
sequence of one or two characters marking the end of a line, consisting of a CARRIAGE RETURN character (0Dh) or a LINE FEED character (0Ah) or a CARRIAGE RETURN followed immediately by a LINE FEED

3.6
evidence record
ER
collection of evidence created for one or more given data objects over time, which can be used to prove the integrity and existence of a data object or a data object group at a certain time

Note 1 to entry: See IETF RFC 4998, IETF RFC 6283 and ETSI SR 019 510.

3.7
long term
period of time long enough for there to be concern about the impacts of changing technologies, including support for new media and data formats, and of a changing user community, on the information being held in a repository, which may extend into the indefinite future

Note 1 to entry: Cryptographic algorithms could become weak.

3.8
long-term integrity preservation
long-term preservation
LTI
extension of the validity status of a *digital signature* (3.1) over long periods of time and/or of provision of proofs of existence of data over long periods of time, in spite of the obsolescence of cryptographic technology such as crypto algorithms, key sizes or hash functions, key compromises or of the loss of the ability to check the validity status of public key certificates

3.9
object identifier as a hash
ObjectId
hash reference of the *PoE object* (3.12), *PoE attribute* (3.11) or data object, consisting of object identifier like *DId* (3.2) or *DSId* (3.3)

3.10
proof of existence
PoE
evidence that proves that an object existed at a specific date/time

Note 1 to entry: See ETSI SR 019 510.

3.11
proof of existence attribute
PoE attribute
reference to the *PoE object* (3.12) containing also PoE object type, optional PoE object location, optional storage for the PoE object and optional data object references as additional clarification of data object(s), protected by PoE object, thus unambiguously specifying their semantics

Note 1 to entry: PoE attribute can be a signed or unsigned object of the *digital signature* (3.1) or the file containing the *ObjectId* (3.9), e.g. *DId* (3.2) or *DSId* (3.3), of PoE object. See 4.1 or Annex G, where the type of PoE object is the file name extension like, e.g. "timestampedFile.EXT.TST.DSId" containing PoE object. The file is stored in "timestampedFile.EXT". The timestamp is stored in timestamp file "timestampedFile.EXT.TST".

3.12
proof of existence object
PoE object
property that represents information about a protected data object like type, status or integrity, a trustworthy information of date and time and a *digital signature* (3.1), possibly as part of a timestamp, which proves the integrity of the PoE object and optionally also the origin of the PoE object

Note 1 to entry: See *DId* (3.2) or *DSId* (3.3).

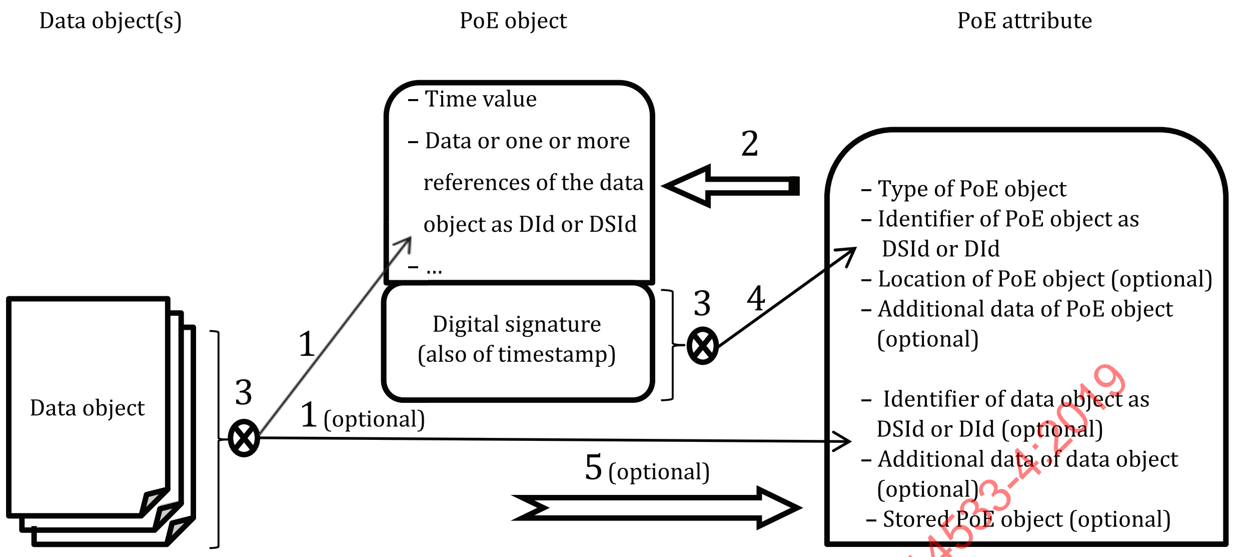
3.13
trusted list
TL
predefined list of items signed by a trusted entity where all the items are authenticated and approved by a trusted signing entity

Note 1 to entry: The primary use of TLs is to verify signed objects, using the TL as a source of trust anchors (see ISO/IEC 9594-8) — trusted root certificates. For more information about the TL see the documentation of operating systems, e.g. Windows, iOS, or the Regulation (EU) No 910/2014[6] (eIDAS Regulation).

4 PoE attributes

4.1 General concept of PoE

This document specifies the PoE attribute as an object which provides a reference between optional data object(s) and the trusted evidence (PoE object) of at least one property of the data object or a set of data objects, especially the integrity, status, type of data object or type of interpretation at a specific time interval or at the date and time, see Figure 1. The PoE attribute is defined in this document as an optional object included in signed attributes or in unsigned attributes as an extension of formats profiled in ISO 14533-1, ISO 14533-3 or marginally in ISO 14533-2 by using elements with similar fields. The PoE attribute is defined also in the form of the file containing ObjectId (e.g. DId or DSId) of the PoE object where the file name of the PoE attribute is the file name of the file containing the PoE object concatenated with a file name extension defined in Annex G, e.g. the data object file is "reports.PNG", the PoE object — file with timestamp is "reports.PNG.TST" and the PoE attribute file is "reports.PNG.TST.DSId".


Data object(s)
PoE object
PoE attribute
– Time value
– Data or one or more references of the data object as DId or DSId
– ...
Digital signature (also of timestamp)
2
3
4
– Type of PoE object
– Identifier of PoE object as DSId or DId
– Location of PoE object (optional)
– Additional data of PoE object (optional)
– Identifier of data object as DSId or DId (optional)
– Additional data of data object (optional)
– Stored PoE object (optional)
Data object
3
1
1 (optional)
5 (optional)

Key

1	the field where the hash value of the data object is stored
2	referenced PoE object from the PoE attribute
3	hash calculation
4	the field where the hash value of the signature of PoE object is stored
5	PoE object can be stored in the PoE attribute

Figure 1 — PoE concept

The PoE object

— can be external to the data object (two separate objects), called an external PoE object,

— can contain the data object, or

— can be a part of the data object.

NOTE 1 An example of the PoE object, which is a part of the data object, is the PDF document timestamp included in the PDF file where the document timestamp protects one PDF document, usually modified by incremental updates, of many PDF documents included in one PDF file. If the PDF document is protected by the PDF document timestamp or PDF CMS signature, then one DSId can be used for identifying one version of PDF document out of many versions of PDF document included in one PDF file.

NOTE 2 The CMS signed or unsigned attributes, defined in this document, can be used in any form of the CMS signature. When the CMS signature is included in the PDF as the PDF signature (see ISO 32000-2), CMS attributes are modified only before storing the CMS object into the PDF document. A similar functionality can be implemented in the XML digital signature using the *Reference* element, e.g. used in a *Manifest* element (see ISO 14533-2).

4.2 Abstract attribute PoEAttribute

The attribute PoEAttribute is an implementation of the PoE attribute as a DER/JER (see ISO 8825-8) encoded file or as a signed or unsigned attribute of the CMS signature (see Note 1 to entry in 3.11 for other types of implementations) and it is identified by *id-PoEAttribute* OID as specified in Annex A. The attribute PoEAttribute is defined as an abstract incomplete attribute and shall be implemented in derived attributes. It defines the common semantics of the fields which can be used or modified in derived attributes, e.g. in LTI PoE attribute, in ERS PoE attribute, in TStOCSP PoE attribute or in other derived implementations (defined in the future).

The value of *PoEAttribute.poEObjectRef.type* field determines the type of *PoEAttribute* instance and the semantics of the *poEObjectRef* fields (*objectId*, *location*, *additionalData*, *partialHash*, *contentDescription*), of the *poEObject* and of the *dataObjectRefs* fields.

The *poEObjectRef* field of the *PoEAttribute* instance contains:

— The *objectId* field is either of the *DSId* type covering a PoE digital signature (see 3.1) of the (external) PoE object or of the *DId* type covering the whole (external) PoE object.

— The optional *location* field shall be the URL location of the CMS signature, as specified in Annex D, where the PoE object is stored, e.g. location contains "#{DSId-x" where "x" is DSId for identification of one parallel signature or location of ER object storage. See EXAMPLE 2 in Annex D, IETF RFC 8089 or IETF RFC 7230, Section 2.7, where URI is used throughout HTTP as the means for identifying resources or Annex D for additional rules when ASN.1 object is stored in ZIP, PDF container or in another ASN.1 objects. If the optional field *poEObject* is included in the *PoEAttribute* instance, the field *location* should not be included.

— The optional *additionalData* field shall be a storage field for additional data used in creation, in accession or in validation of the PoE object.

— The optional *partialHash* field shall be a storage field for additional hash value used in creation, in accession or in validation of the PoE object.

— The optional *contentDescription* field shall be used for additional information about the PoE object, e.g. *contentDescription* can contain MIME header defined in IETF RFC 2045 (see Annex F).

The optional *poEObject* field of the *PoEAttribute* instance should be a storage of the PoE object in the *PoEAttribute* only if the signature format does not define a field for this type of PoE object. The *poEObject* field should be included only in one signature and only in the *PoEAttribute* instance which supports this PoE object. The *PoEAttribute* instance supports the PoE object if

— the hash algorithm in the *poEObjectRef.objectId* field is the same as used in PoE object, and

— the *PoEAttribute* instance contains the optional *dataObjectRefs* field (if needed) with references to the protected data objects. The optional *dataObjectRefs* field shall be included only in the *PoEAttribute* instance which supports the PoE object to avoid duplication of data or misusing this field.

The other signatures contain only the *PoEAttribute* instance used as a reference to the supporting *PoEAttribute* instance. One *ObjectRef* instance of many instances included in the *dataObjectRefs* field shall contain a reference to the data object with optional clarification of the data object, e.g. provides the type of data object interpretation. The data object is the object of the CMS signature where the *PoEAttribute* instance is included, the parallel CMS signature, the external CMS signature or any data, e.g. a file or signatures other than CMS (JSON signature etc.). The semantics of the *ObjectRef* fields of the *dataObjectRefs* field are as follows:

— If the value of the *poEObjectRef.type* field does not define the type of data objects the *type* field of the *ObjectRef* fields of the optional *dataObjectRefs* field shall be used for identification of the type of data object. If a CMS signature is referenced, the *type* field contains *id-signedData* OID (see IETF RFC 5652, Section 5.1). If arbitrary octet strings are referenced, the *type* field contains *id-data* OID (see IETF RFC 5652, Section 4), such as ASCII or UTF-8 text files; the interpretation is left up to the application whether it will use the value of the *contentDescription* field containing, e.g. MIME *Content-Type*.

— The *objectId* field is either of the *DSId* type covering a digital signature of data object or of the *DId* type covering the whole data object.

— The optional *location* field shall be the URL location of the data object, as specified in Annex D. See IETF RFC 8089 or IETF RFC 7230, Section 2.7, where URI is used throughout HTTP as the means for identifying resources or Annex D for additional rules when the ASN.1 object is stored in a ZIP, PDF container or in another ASN.1 object. The optional *location* field shall be not included when the *type* field contains *id-signedData* OID and the data object is the object of the CMS signature (the *SignerInfo* signature or the *SignerInfo* of parallel signatures) where the *PoEAttribute* instance supporting PoE object is included.

— The optional *additionalData* field shall be used for additional clarification of data object usage, e.g. the *additionalData* field contains the OID and an instance of the *ATSHashIndexV3* ASN.1 type defined in ETSI EN 319 122-1 V1.1.1 (2016-04), Clause 5.5.2 or contains OID of transformation method, used before hashing procedure, together with additional data used by transformation method.

— The optional *partialHash* field shall be used for additional clarification of data object usage, e.g. *partialHash* field contains the hash value as is defined for the timestamped value in the archive-timestamp-v3 attribute in ETSI EN 319 122-1 V1.1.1 (2016-04), Clause 5.5.3 or contains the hash value when the *additionalData* field is present and contains the transformation method.

— The optional *contentDescription* field shall be used for additional information about the data object interpretation, e.g. *contentDescription* can contain MIME header defined in IETF RFC 2045 (see Annex F for an example of *contentDescription* usage).

NOTE 1 As defined in 4.3, the *poEObjectRef.type* field contains the OID value *id-poe-lti-rfcts* or *id-poe-lti-ers* when the *PoEAttribute* instance is used as the long term integrity (LTI) *PoEAttribute* instance for securing signatures or data objects together with additional metadata of them by using the IETF RFC 3161 timestamp or IETF RFC 4998/IEFT RFC 6283 evidence record syntax object.

NOTE 2 As defined in 4.4, the *poEObjectRef.type* field contains the OID value *id-poe-ers*, when the attribute *PoEAttribute* instance is used for securing hash values of data objects of any type, e.g. for signature(s), where the hash values of data objects are computed and stored in the ERS *PoEAttribute* instance as defined for the LTI *PoEAttribute* instance, similar to Figure 3, with one exception: the hash value of the *dataObjectRefs* field is not the input for the Evidence Record generation procedure. Instead of that, the hash values as inputs for the Evidence Record generation procedure are handled according to IETF RFC 4998 Section 4.2 or to IETF RFC 6283 Section 3.1.1 and they are at least the following hash values stored in fields of ERS *PoEAttribute* instance:

— the *partialHash* field of *ObjectRef* included in *dataObjectRefs* (if the *additionalData* field is included), or

— the *dId* field of *ObjectRef* included in *dataObjectRefs* (if the *additionalData* field is not included).

NOTE 3 As defined in 4.5, the *poEObjectRef.type* field contains the OID value *id-TStOCSP* (timestamp through OCSP) when the TStOCSP *PoEAttribute* instance is used as an indication of the timestamp through OCSP usage.

```
id-PoEA OBJECT IDENTIFIER ::= { iso(1) standard(0)
   long-term-signature-profiles(14533) part(4) attribute(0) }
id-PoEAttribute OBJECT IDENTIFIER ::= { id-PoEA poeattribute(1) }
id-poe-lti-rfcts OBJECT IDENTIFIER ::= { id-PoEAttribute poe-lti-rfcts(1) }
id-poe-lti-ers OBJECT IDENTIFIER ::= { id-PoEAttribute poe-lti-ers(2) }
id-poe-ers OBJECT IDENTIFIER ::= { id-PoEAttribute poe-ers(3) }
URL ::= UTF8String -- see 6.2.12 of Rec. ITU-T X.520 | ISO/IEC 9594-6
DSId ::= MessageImprint
DId ::= MessageImprint
PoEAttribute ::= SEQUENCE {
      poEObjectRef ObjectRef,
      poEObject OCTET STRING OPTIONAL,
      dataObjectRefs DataObjectRefs OPTIONAL }

DataObjectRefs ::= SEQUENCE OF ObjectRef
ObjectRef ::= SEQUENCE {
      type OBJECT IDENTIFIER,
      objectId ObjectId,
      location URL OPTIONAL,
      additionalData [1] EXPLICIT AdditionalData OPTIONAL,
      partialHash [2] EXPLICIT MessageImprint OPTIONAL,
      contentDescription UTF8String OPTIONAL }
ObjectId ::= CHOICE {
      dSId DSId,
      dId [0] EXPLICIT DId }
AdditionalData ::= SEQUENCE {
      dataType OBJECT IDENTIFIER,
      otherData ANY DEFINED BY dataType }
```

4.3 LTI *PoEAttribute* instance based on IETF RFC 3161 timestamp or IETF RFC 4998/ IETF RFC 6283 evidence record

The long term integrity (LTI) *PoEAttribute* instance is derived from the *PoEAttribute* instance, see 4.2.

The LTI *PoEAttribute* instance shall be included in the CMS signature to protect the signature (this one, parallel and external ones) and any external data. The LTI *PoEAttribute* instance can be also included in the CMS signature of the IETF RFC 3161 timestamp type included e.g., in CMS with a signature timestamp.

EXAMPLE The LTI PoEAttribute instance based on timestamp or ER can be included in the ASiC document timestamp (CMS signature — see ETSI EN 319 162-1 "timestamp*.tst" timestamp token file) to protect the long-term integrity of the selected files or of all files of the ASiC container, fulfilling ETSI EN 319 162-1 or ER as defined in ETSI EN 319 162-1.

The LTI *PoEAttribute* instance shall be used as:

— an optional storage of the reference to the LTI *PoEAttribute* instance supporting the PoE object which protects the integrity of this signature;

— a storage of the reference to the PoE object if this LTI *PoEAttribute* instance supports the PoE object; only if the LTI *PoEAttribute* instance supports the PoE object, identified by the *objectId* field of the *poEObjectRef* field, shall the LTI *PoEAttribute* instance be a storage of references to the protected data objects, where each reference includes at least the hash value of the protected data object, optionally with an *additionalData* field , e.g. with the transformation method used to compute the hash value in such a way, which allows an upgrade of the protected data object, e.g. by including CRL, OCSP responses or additional information in the data object;

— an optional storage of the PoE object if this LTI *PoEAttribute* instance supports PoE object.

The LTI *PoEAttribute* instance shall be used for securing data objects (signature is one type of the data object) together with their additional optional metadata

— by using IETF RFC 3161 timestamp, in which case the *poEObjectRef.type* field shall contain the OID value *id-poe-lti-rfcts*; and if the LTI *PoEAttribute* instance supports the PoE object, the optional *poEObject* field shall contain IETF RFC 3161 timestamp which timestamps the hash value of the *dataObjectRefs* field, or

— by using evidence record (ER) IETF RFC 4998/IETF RFC 6283, in which case the *poEObjectRef.type* field shall contain the OID value *id-poe-lti-ers*; and if the LTI *PoEAttribute* instance supports the PoE object, the optional *poEObject* field may contain the ER object and the hash value of the *dataObjectRefs* field shall be an input to the procedure defined in IETF RFC 4998/IEFT RFC 6283 for ERS *PartialHashtree* usage, whereas the input to the ER procedure is not limited only to the hash values of *dataObjectRefs* fields.

The *poEObjectRef.objectId* field of *PoEAttribute* shall be the DSId value of the timestamp stored

— in the *poEObject* field if the *poEObjectRef.type* value is *id-poe-lti-rfcts* OID, or

— in the first *ArchiveTimeStamp* in the first *ArchiveTimeStampChain* of *archiveTimeStampSequence* of the ERS object stored in the *poEObject* field of the *PoEAttribute* instance if the *poEObjectRef.type* value is *id-poe-lti-ers* OID.

The hash algorithm identifier, with parameters, stored in the field *poEObjectRef.dSId* of the *PoEAttribute* instance which supports the PoE object, shall be the same as is used in the IETF RFC 3161 timestamp or is used in particular ERS *PartialHashtree* of the evidence record.

NOTE 1 If the *PoEAttribute* instance contains the field *dataObjectRefs,* then it supports the PoE object defined in 5.3.

NOTE 2 If the element *dataObjectRefs.ObjectRef.location* is hashed as part of the *dataObjectRefs* field, then there can be a reduction of the negotiability.

If the LTI *PoEAttribute* instance supports the PoE object, then the following fields of the *dataObjectRefs* field are included:

— If *ObjectRef* included in *dataObjectRefs* is a reference to the CMS signature (the CMS *SignerInfo* instance in the signature), the *type* field (*dataObjectRefs.ObjectRef.type*) shall contain *id-signedData* OID (see IETF RFC 5652, Section 5.1); the hash value stored in *dSId* is used for identification of one (parallel) CMS *SignerInfo* signature whose hash values are stored in the fields *additionalData* and *partialHash*:

  — The *additionalData* field contains the OID and instance of *ATSHashIndexV3* ASN.1 type defined in ETSI EN 319 122-1 V1.1.1 (2016-04), Clause 5.5.2.

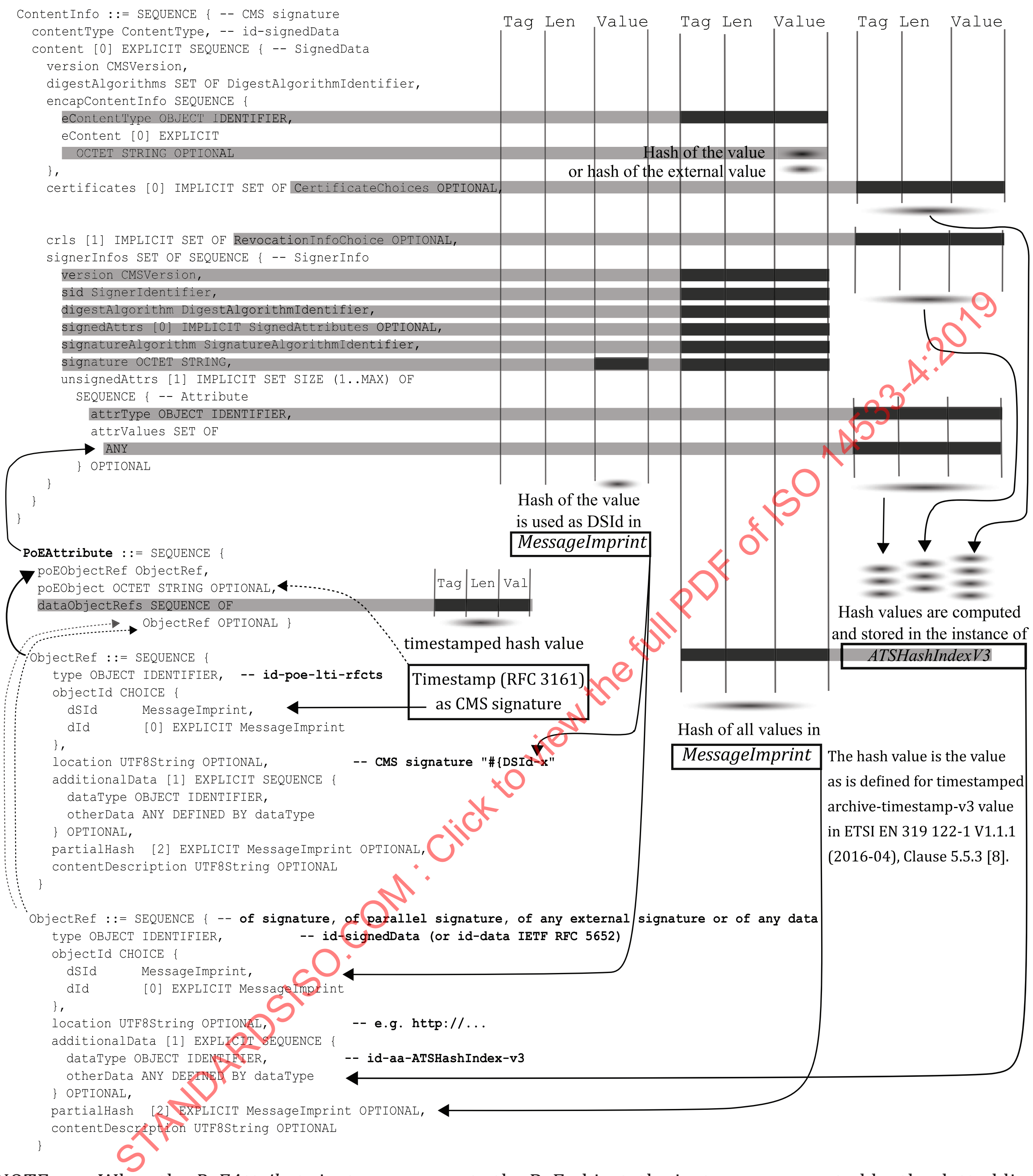
  — The field *partialHash* shall contain the hash value as is defined for the timestamped value in the archive-timestamp-v3 attribute in ETSI EN 319 122-1 V1.1.1 (2016-04), Clause 5.5.3.

— When *ObjectRef* included in *dataObjectRefs* is not a reference to the CMS *SignerInfo* signature, the *type* field (*dataObjectRefs.ObjectRef.type*) shall contain *id-data* OID (see IETF RFC 5652, Section 4). In this case the hash value of the referenced data object

  — is stored in the *partialHash* field of *ObjectRef* included in *dataObjectRefs* if referenced data are before using a hashing procedure transformed by the method identified by OID in the *additionalData* field, where the *otherData* field shall contain additional data used by transformation method if needed and *dSId* field of *dataObjectRefs.ObjectRef* is used for identification of data object in the searching procedure, e.g. of the JSON signature, or

  — is stored in the *dId* field of *ObjectRef* included in *dataObjectRefs* if the *additionalData* field is not present.

If the *poEObjectRef.type* field contains the OID value *id-poe-lti-rfcts* and any algorithms in the LTI *PoEAttribute* instance supporting the PoE object is going to be less secure or the validity of the timestamp certificate included in the PoE object is going to expire, then a new LTI *PoEAttribute* instance supporting the PoE object shall be included and a new timestamp of the PoE object shall be supported. This new LTI *PoEAttribute* instance supporting the PoE object should be referenced from all new LTI *PoEAttribute* instances (e.g. included in the signatures identified by *ObjectRef* of *dataObjectRefs* using the *location* field). Figure 2 gives an overview of the main elements of LTI *PoEAttribute* based on IETF RFC 3161 timestamp and Figure 3 gives an overview of the main elements of LTI *PoEAttribute* based on IETF RFC 4998 or IETF RFC 6283 evidence record.



```
ContentInfo ::= SEQUENCE { -- CMS signature
  contentType ContentType, -- id-signedData
  content [0] EXPLICIT SEQUENCE { -- SignedData
    version CMSVersion,
    digestAlgorithms SET OF DigestAlgorithmIdentifier,
    encapContentInfo SEQUENCE {
      eContentType OBJECT IDENTIFIER,
      eContent [0] EXPLICIT
        OCTET STRING OPTIONAL
    },
    certificates [0] IMPLICIT SET OF CertificateChoices OPTIONAL,

    crls [1] IMPLICIT SET OF RevocationInfoChoice OPTIONAL,
    signerInfos SET OF SEQUENCE { -- SignerInfo
      version CMSVersion,
      sid SignerIdentifier,
      digestAlgorithm DigestAlgorithmIdentifier,
      signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
      signatureAlgorithm SignatureAlgorithmIdentifier,
      signature OCTET STRING,
      unsignedAttrs [1] IMPLICIT SET SIZE (1..MAX) OF
        SEQUENCE { -- Attribute
          attrType OBJECT IDENTIFIER,
          attrValues SET OF
            ANY
        } OPTIONAL
      }
    }
}

PoEAttribute ::= SEQUENCE {
  poEObjectRef ObjectRef,
  poEObject OCTET STRING OPTIONAL,
  dataObjectRefs SEQUENCE OF
                ObjectRef OPTIONAL }

ObjectRef ::= SEQUENCE {
    type OBJECT IDENTIFIER,   -- id-poe-lti-rfcts
    objectId CHOICE {
      dSId      MessageImprint,
      dId       [0] EXPLICIT MessageImprint
    },
    location UTF8String OPTIONAL,          -- CMS signature "#{DSId-x"
    additionalData [1] EXPLICIT SEQUENCE {
      dataType OBJECT IDENTIFIER,
      otherData ANY DEFINED BY dataType
    } OPTIONAL,
    partialHash   [2] EXPLICIT MessageImprint OPTIONAL,
    contentDescription UTF8String OPTIONAL
  }

ObjectRef ::= SEQUENCE { -- of signature, of parallel signature, of any external signature or of any data
    type OBJECT IDENTIFIER,           -- id-signedData (or id-data IETF RFC 5652)
    objectId CHOICE {
      dSId      MessageImprint,
      dId       [0] EXPLICIT MessageImprint
    },
    location UTF8String OPTIONAL,          -- e.g. http://...
    additionalData [1] EXPLICIT SEQUENCE {
      dataType OBJECT IDENTIFIER,          -- id-aa-ATSHashIndex-v3
      otherData ANY DEFINED BY dataType
    } OPTIONAL,
    partialHash   [2] EXPLICIT MessageImprint OPTIONAL,
    contentDescription UTF8String OPTIONAL
  }
```

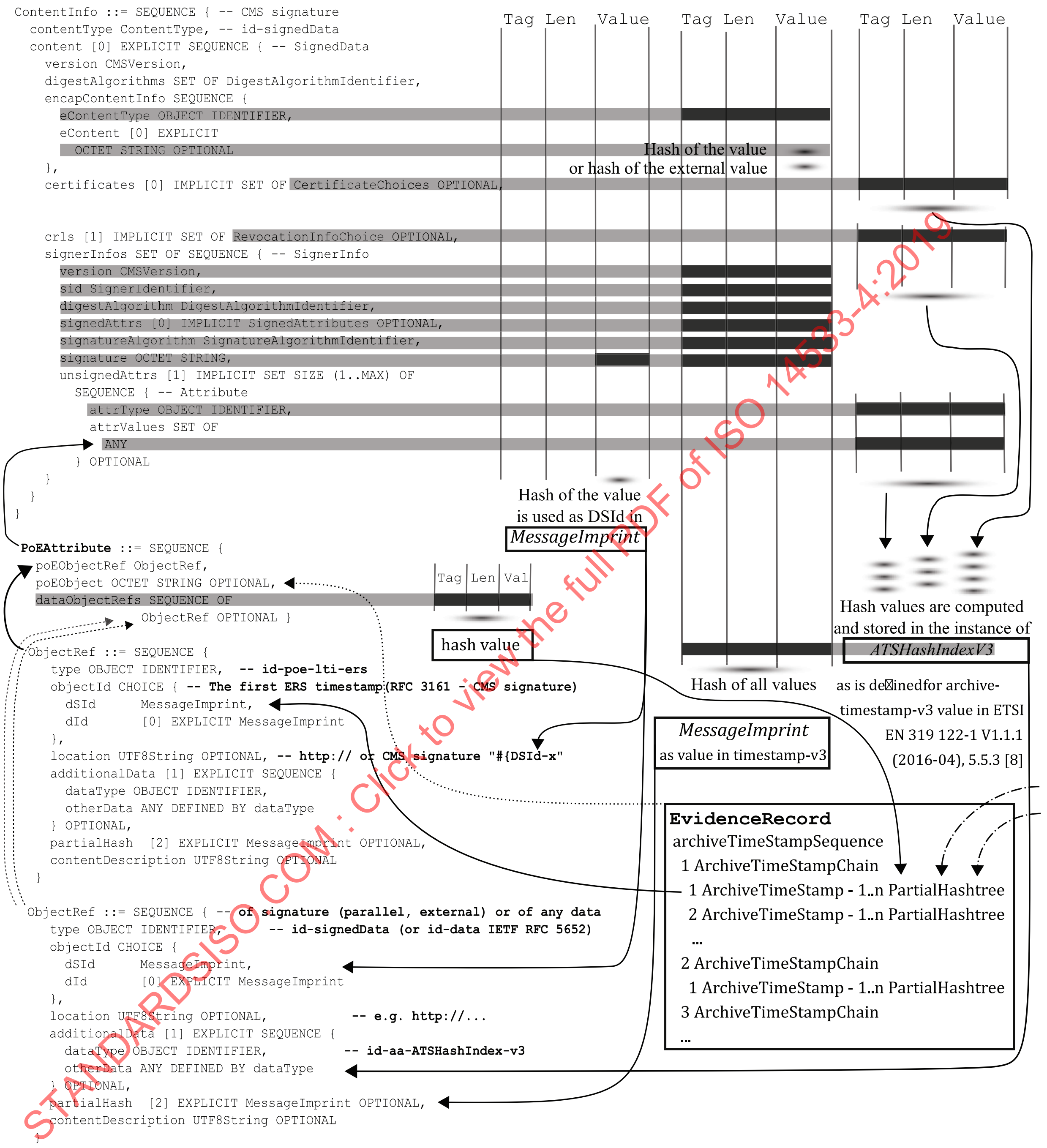
NOTE When the *PoEAttribute* instance supports the PoE object, the instances connected by the dotted line are included.

Figure 2 — LTI *PoEAttribute* based on IETF RFC 3161 timestamp

NOTE 3 The hash value, stored in the *partialHash* field, protects the hash validation result from inconsistency in the situations when some (parallel) signatures will be unavailable at later time, e.g. when subsequent timestamps are going to be included in a new LTI *PoEAttribute* instance.

NOTE 4 The LTI *PoEAttribute* instance can be stored in an external CMS *SignerInfo* signature or in any parallel CMS *SignerInfo* signature.

NOTE 5 The LTI *PoEAttribute* instance can be used for securing the integrity of signatures, e.g. for parallel signatures or external signatures or for securing the integrity of any data together with information of its interpretation.



```
ContentInfo ::= SEQUENCE { -- CMS signature
  contentType ContentType, -- id-signedData
  content [0] EXPLICIT SEQUENCE { -- SignedData
    version CMSVersion,
    digestAlgorithms SET OF DigestAlgorithmIdentifier,
    encapContentInfo SEQUENCE {
      eContentType OBJECT IDENTIFIER,
      eContent [0] EXPLICIT
        OCTET STRING OPTIONAL
    },
    certificates [0] IMPLICIT SET OF CertificateChoices OPTIONAL,


    crls [1] IMPLICIT SET OF RevocationInfoChoice OPTIONAL,
    signerInfos SET OF SEQUENCE { -- SignerInfo
      version CMSVersion,
      sid SignerIdentifier,
      digestAlgorithm DigestAlgorithmIdentifier,
      signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
      signatureAlgorithm SignatureAlgorithmIdentifier,
      signature OCTET STRING,
      unsignedAttrs [1] IMPLICIT SET SIZE (1..MAX) OF
        SEQUENCE { -- Attribute
          attrType OBJECT IDENTIFIER,
          attrValues SET OF
           ANY
        } OPTIONAL
    }
  }
}
PoEAttribute ::= SEQUENCE {
  poEObjectRef ObjectRef,
  poEObject OCTET STRING OPTIONAL,
  dataObjectRefs SEQUENCE OF
                 ObjectRef OPTIONAL }

ObjectRef ::= SEQUENCE {
   type OBJECT IDENTIFIER,  -- id-poe-lti-ers
   objectId CHOICE { -- The first ERS timestamp (RFC 3161 - CMS signature)
     dSId      MessageImprint,
     dId       [0] EXPLICIT MessageImprint
   },
   location UTF8String OPTIONAL, -- http:// or CMS signature "#{DSId-x"
   additionalData [1] EXPLICIT SEQUENCE {
     dataType OBJECT IDENTIFIER,
     otherData ANY DEFINED BY dataType
   } OPTIONAL,
   partialHash  [2] EXPLICIT MessageImprint OPTIONAL,
   contentDescription UTF8String OPTIONAL
 }

ObjectRef ::= SEQUENCE { -- of signature (parallel, external) or of any data
   type OBJECT IDENTIFIER,       -- id-signedData (or id-data IETF RFC 5652)
   objectId CHOICE {
     dSId      MessageImprint,
     dId       [0] EXPLICIT MessageImprint
   },
   location UTF8String OPTIONAL,          -- e.g. http://...
   additionalData [1] EXPLICIT SEQUENCE {
     dataType OBJECT IDENTIFIER,         -- id-aa-ATSHashIndex-v3
     otherData ANY DEFINED BY dataType
   } OPTIONAL,
   partialHash  [2] EXPLICIT MessageImprint OPTIONAL,
   contentDescription UTF8String OPTIONAL
 }
```

EvidenceRecord
archiveTimeStampSequence
1 ArchiveTimeStampChain
  1 ArchiveTimeStamp - 1..n PartialHashtree
  2 ArchiveTimeStamp - 1..n PartialHashtree
  ...
2 ArchiveTimeStampChain
  1 ArchiveTimeStamp - 1..n PartialHashtree
3 ArchiveTimeStampChain
...

Tag Len Value Tag Len Value Tag Len Value
Hash of the value or hash of the external value
Hash of the value is used as DSId in
MessageImprint
Tag Len Val
hash value
Hash values are computed and stored in the instance of
ATSHashIndexV3
Hash of all values
as is deinedfor archive-timestamp-v3 value in ETSI EN 319 122-1 V1.1.1 (2016-04), 5.5.3 [8]
MessageImprint
as value in timestamp-v3

NOTE When the *PoEAttribute* instance supports the PoE object, the instances connected by the dotted line are included. The *EvidenceRecord* object can be later stored in the *PoEAttribute* of the signature from the external (http://...) location. The dot-and-dash line represents any additional data objects included in the ER object.

Figure 3 — LTI *PoEAttribute* based on IETF RFC 4998/IETF RFC 6283 evidence record

If the *poEObjectRef.type* field contains the OID value *id-poe-lti-ers* and any algorithms in the LTI *PoEAttribute* instance or the validity of the last timestamp certificate is going to be less secure or expired, one of these renewal procedures shall be used:

— Timestamp Renewal according to IETF RFC 4998/IETF RFC 6283, or

— Hash-Tree Renewal according to IETF RFC 4998/IETF RFC 6283 as follows:

  — before starting the Hash-Tree Renewal procedure, new hash values are computed and stored in new instances of *ObjectRef* included in *dataObjectRefs* of a new LTI *PoEAttribute* instance created according to the present LTI *PoEAttribute* instance from all instances *ObjectRef* included in *dataObjectRefs*;

  — if the present LTI *PoEAttribute* instance, supporting the PoE object, contains the *poEObject* field (as a storage of the ER object) then the content of *poEObject* field shall be copied to the new LTI *PoEAttribute* instance;

  — the hash value of the *dataObjectRefs* of the new LTI *PoEAttribute* instance is input to the Hash-Tree Renewal procedure together with other hash values known to the ER object system;

  — the renewed ER object can be stored externally or included in a new LTI *PoEAttribute* instance.

4.4 ERS *PoEAttribute* instance based on IETF RFC 4998/IETF RFC 6283 evidence record

The unsigned ERS *PoEAttribute* instance is a modification of the LTI *PoEAttribute* instance with the *poEObjectRef.type* field containing the OID value *id-poe-ers*.

If the *poEObjectRef.type* field contains the OID value *id-poe-ers*, the *PoEAttribute* instance is used for securing hash values of objects referenced by *dataObjectrefs.ObjectRef*, e.g. signature (this one, parallel and external ones) and any external data.

The hash values itself are computed as defined for the LTI *PoEAttribute* instance (with type field containing *id-poe-lti-ers* OID), but instead of calculating the evidence record based on the hash value of the *dataObjectRefs* field, a binary hash tree is built with a root hash value which is then timestamped according to the points 3, 4 and 5 of Section 4.2 of IETF RFC 4998 or Section 3.1.1 of IETF RFC 6283.

That means, the processing of the hash values used as an input to the evidence record object generation process pursuant to IETF RFC 4998/IETF RFC 6283 makes the difference between *id-poe-lti-ers* and *id-poe-ers*.

All hash values stored in the following fields are input to the evidence record object fields according to rules defined in IETF RFC 4998/ IETF RFC 6283, Section 4:

— the *partialHash* field of *ObjectRef* included in *dataObjectRefs* (if the *additionalData* field is included), or

— the *dId* field of *ObjectRef* included in *dataObjectRefs* (if the *additionalData* field is not included).

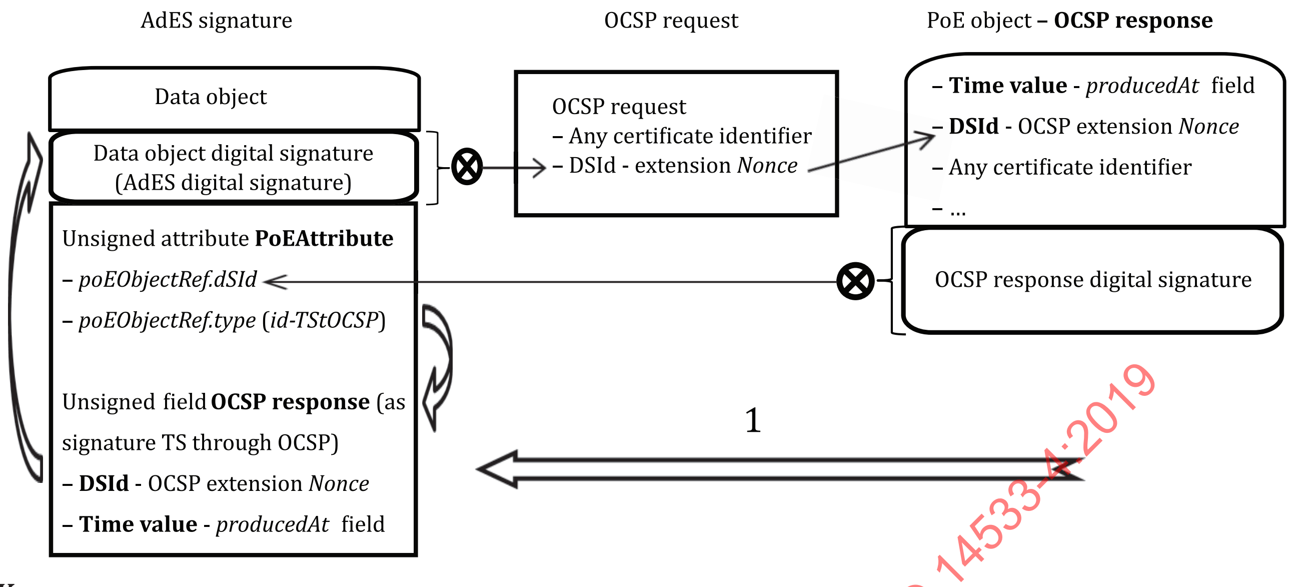
If the *PoEAttribute* instance supports the PoE object (evidence record object), the *poEObject* field shall be an optional store for the evidence record object pursuant to IETF RFC 4998/IETF RFC 6283 where it can be included anytime. See 5.3 about the PoE object supported by LTI *PoEAttribute* or ERS *PoEAttribute*.

NOTE The advantage of the usage of ERs is that they can be used in connection to any type of data objects, see Figure 3 where the dot-and-dash line represents any additional data objects included in the ER object.

4.5 TStOCSP *PoEAttribute* instance

When the signature timestamp is implemented as an electronic timestamp through OCSP, it can be indicated by the presence of the **TStOCSP *PoEAttribute*** instance derived from attribute PoEAttribute, where the *poEObjectRef.dSId* field of *PoEAttribute* contains the DSId value of the OCSP response used as the signature timestamp and the *poEObjectRef.type* field of *PoEAttribute* contains the OID value *id-TStOCSP* "1.0.14533.4.1.1" timestamp through OCSP, as specified in Annex C and used in the validation as specified in Annex E. Figure 4 gives an overview of the used elements.

```
id-TStOCSP OBJECT IDENTIFIER ::= { iso(1) standard(0)
   long-term-signature-profiles(14533) part(4) certificate-policy(1)
   timestamp-through-OCSP(1) }
```


AdES signature
OCSP request
PoE object – OCSP response
Data object
Data object digital signature (AdES digital signature)
Unsigned attribute PoEAttribute
– poEObjectRef.dSId
– poEObjectRef.type (id-TStOCSP)
Unsigned field OCSP response (as signature TS through OCSP)
– DSId - OCSP extension Nonce
– Time value - producedAt field
OCSP request
– Any certificate identifier
– DSId - extension Nonce
– Time value - producedAt field
– DSId - OCSP extension Nonce
– Any certificate identifier
– ...
OCSP response digital signature
1

Key

1 external PoE object is included in the signature

NOTE OCSP extension Nonce contains the hash value of the signature.

Figure 4 — Signature timestamp as an electronic timestamp through OCSP

4.6 Attribute PoEHashIndex

This attribute provides a hash reference which can be covered by a signature as a proof that the referenced object existed before a signature creation. When the referenced object contains a trusted time value, it is a proof that the signature was created after the creation of the referenced object.

For the purpose of referencing the PoE object with a trusted time value of its creation, e.g. OCSP response, in the context of this document, the ats-hash-index-v3 attribute shall be also **a signed** attribute of the CMS signature as an additional usage of the ats-hash-index-v3 attribute defined in Clause 5.5.2 of ETSI EN 319 122-1 V1.1.1 (2016-04), Clause 5.5.2 and the alias name of this attribute is the PoEHashIndex attribute identified by *id-poe-hash-index* OID.

```
id-poe-hash-index OBJECT IDENTIFIER ::= {
   iso(1) standard(0) long-term-signature-profiles(14533) part(4) attribute(0)
      poe-hash-index(2) }
PoEHashIndex ::= ATSHashIndexV3
```

The CMS signature signs in the *PoEHashIndex* value of the signed attribute also the hash value of the CRL or OCSP responses, included in the CMS signature, as a proof that the CMS signature was created **after** the time value stored in the *thisUpdate* field of the CRL or in the *producedAt* field of the OCSP response. For more information see Figures E.1 and E.2.

The type of PoE object is CRL or OCSP response included in the *crls* field of *SignedData*. The identifier of the PoE object is the value of *crlshashIndex* of the PoEHashIndex attribute. Figure 5 gives an overview of the used elements.

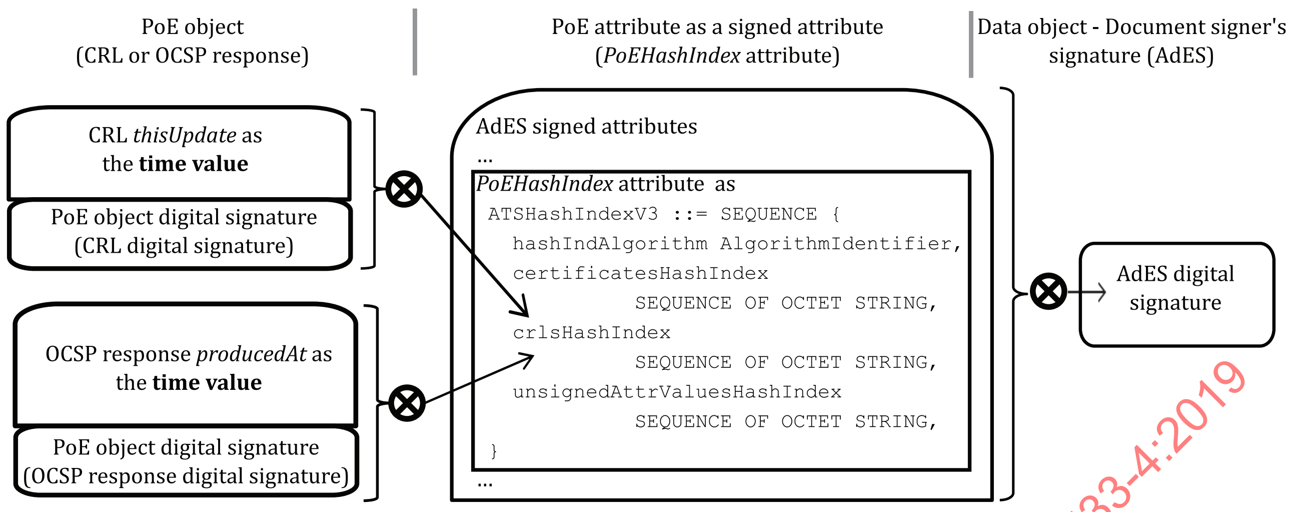

PoE object
(CRL or OCSP response)
PoE attribute as a signed attribute
(*PoEHashIndex* attribute)
Data object - Document signer's signature (AdES)
CRL *thisUpdate* as the **time value**
PoE object digital signature (CRL digital signature)
OCSP response *producedAt* as the **time value**
PoE object digital signature (OCSP response digital signature)
AdES signed attributes
...
PoEHashIndex attribute as
ATSHashIndexV3 ::= SEQUENCE {
hashIndAlgorithm AlgorithmIdentifier,
certificatesHashIndex
SEQUENCE OF OCTET STRING,
crlsHashIndex
SEQUENCE OF OCTET STRING,
unsignedAttrValuesHashIndex
SEQUENCE OF OCTET STRING,
}
...
AdES digital signature

Figure 5 — Signature created after the *thisUpdate* time of CRL or *producedAt* of OCSP response

4.7 Attribute preservation-integrity-list

The preservation-integrity-list attribute is a signed or unsigned attribute of the CMS signature identified by *id-preservation-integrity-list* OID. The value *PreservationIntegrityList* of the attribute is UTF8 encoded text with two or more lines. Figures 6 and 7 give an overview of the usage of elements.

NOTE 1 The UTF8 encoded text with two or more lines is used for the value of the attribute to be independent of the syntax type of the signature, e.g. ASN.1 CMS, XML digital signature or PDF signature, where it can be included or exported and used at the application level after transforming to the preferred encoding which can be changed later due to concerns about the impact of changing technologies over a long period of time, e.g. XML to JSON. The text format of the preservation-integrity-list attribute can be also used as a neutral format when references to the objects included in the *dataObjectRefs* of attribute PoEAttribute are exported for the application level, especially when human-readable formats are to be used, as well.

```
id-preservation-integrity-list OBJECT IDENTIFIER ::= {
   iso(1) standard(0) long-term-signature-profiles(14533) part(4) attribute(0)
      preservation-integrity-list(3) }
PreservationIntegrityList ::= UTF8String
```

The text of the *PreservationIntegrityList* shall consist of 3 types of subsequent text lines: mandatory "FILE=x", mandatory "HASH=x" and optional "NOTICE=x" followed by a single EOL marker, where ‘x’ shall be as follows:

— If the line is "FILE=x", the x shall be a URL of the object. See EXAMPLE 2 in Annex D or IETF RFC 7230, Section 2.7, where URI is used throughout HTTP as the means for identifying resources.

— If the line is "HASH=x", the x shall be a Base64 (IETF RFC 4648) encoded DId of the object whose location is in the previous line "FILE=x". The subsequent line may be repeated more than once with the line "HASH=x", e.g. with a different hash algorithm and hash value for the same object referenced by URL in previous line "FILE=x". The hash algorithm in the first line "HASH=x" shall be the same as is the algorithm in the signed attribute *messageDigest*.

— If the line is "NOTICE=x", the x shall be any additional information of the object whose location and hash are in the previous lines "FILE=x" and "HASH=x".

The subsequent text lines "FILE=x", "HASH=x" and optional "NOTICE=x" may be repeated more than once, e.g. with a different object referenced by URL in the "FILE=x" line. The hash of the signed document shall be checked. However, in this case the "HASH=x" checking is defined at the application level, namely if any of the "HASH=x" are actually checked against the objects referenced in "FILE=x" and what to do if the object is inaccessible or the "HASH=x" comparison fails.

The x of "NOTICE=x" line may contain MIME *Content-Type* defined in IETF RFC 2045, Section 5 (see Annex E).

The preservation-integrity-list signed attribute can be used by the CMS signature to sign the hash value of the referenced data identified in the "FILE=x" and "HASH=x" lines as a proof that such data were created before the time value stored in the signature timestamp or another timestamp covering that signature.

When the hash value of the referenced data identified in the "FILE=x" and "HASH=x" lines cover object with a trusted time (PoE object), e.g. timestamp, in this case it is proof of existence of the signature creation after the time value of the PoE object (see Figures E.1 and E.2). The same functionality can be implemented in the XML digital signature using the *Manifest* element (see ISO 14533-2).

In the first part of Figure 6, according to the concept of 4.1, PoE object type is determined by FILE content of the PoE Object. The PoE object DId is the HASH content of the PoE Object and the data object is the signature (created after the time of the PoE object).

In the second part of Figure 6, according to the concept of 4.1, the PoE object is the signature timestamp, where the type of the PoE object in the PoE attribute and the identifier of the PoE object are implicit (of timestamped signature, signing this attribute) and data objects are objects specified in the FILE, HASH and NOTICE fields.

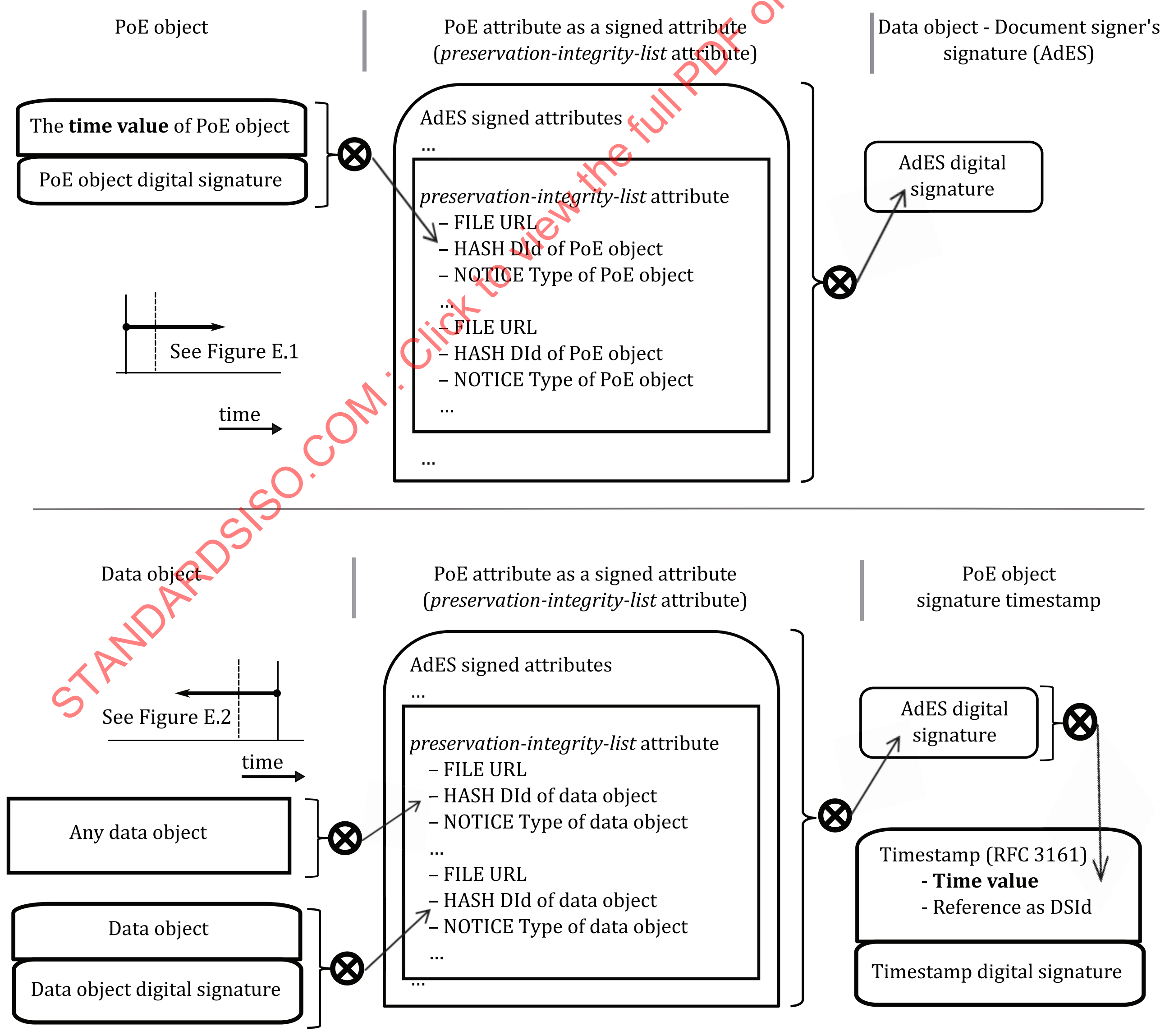

PoE object
PoE attribute as a signed attribute (*preservation-integrity-list* attribute)
Data object - Document signer's signature (AdES)
The **time value** of PoE object
PoE object digital signature
AdES signed attributes
...
preservation-integrity-list attribute
– FILE URL
– HASH DId of PoE object
– NOTICE Type of PoE object
...
– FILE URL
– HASH DId of PoE object
– NOTICE Type of PoE object
...
...
AdES digital signature
See Figure E.1
time
Data object
PoE attribute as a signed attribute (*preservation-integrity-list* attribute)
PoE object signature timestamp
AdES signed attributes
...
preservation-integrity-list attribute
– FILE URL
– HASH DId of data object
– NOTICE Type of data object
...
– FILE URL
– HASH DId of data object
– NOTICE Type of data object
...
...
See Figure E.2
time
Any data object
Data object
Data object digital signature
AdES digital signature
Timestamp (RFC 3161)
- **Time value**
- Reference as DSId
Timestamp digital signature

Figure 6 — Example of the preservation-integrity-list attribute as a signed attribute

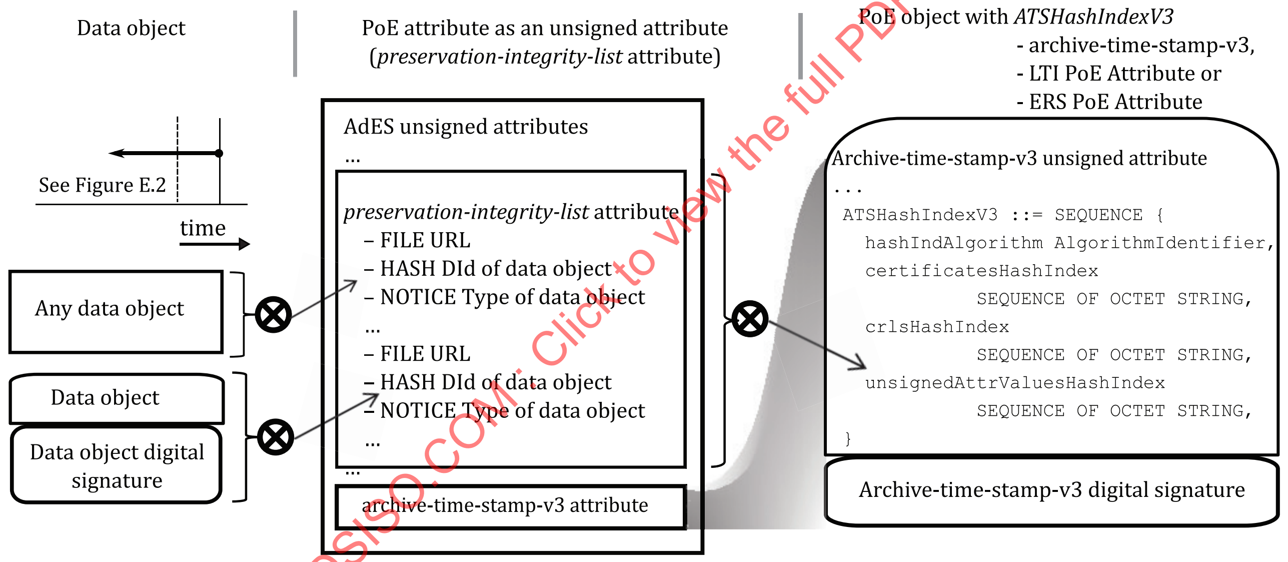
NOTE 2 The content of the preservation-integrity-list attribute is expected to be used for protection of the integrity of additional referenced data from the signed document when using the CMS, PDF or XML signatures. Additional referenced data are unprotected referenced data from a signed document which can change. Any change in referenced (unprotected) data can further change, e.g. the visualization of the signed document, and thus the signature guarantee that such external referenced data have not been changed and will not be changed.

NOTE 3 The preservation service can use the signed document for the declaration of the preserved objects referenced in the preservation-integrity-list attribute.

NOTE 4 The value of the attribute as UTF8 encoded text can be exported from the signature and provided as a TXT document for users of the signed document as additional data for correct usage and interpretation of the signed document (TXT document contains relevant objects located outside of the signed document).

When the preservation-integrity-list attribute is used as an unsigned attribute to protect the referenced data (additional data not protected by the AdES signature used, e.g. for correct interpretation of the signed document), its content may be protected by the *archive-timestamp-v3* attribute as defined in ETSI EN 319 122-1 V1.1.1 (2016-04), Clause 5.5.3 or by the LTI PoE Attribute or the ERS PoE Attribute. The LTI *PoEAttribute* instance or ERS *PoEAttribute* instance may coexist with the archive-time-stamp-v3 attribute in the CMS signature according to ETSI EN 319 122-1 V1.1.1 (2016-04), Clause 5.5.3.

According to the concept in 4.1 the type and identifier of the PoE object are implicit (the PoE object covering the PoE attribute), e.g. the archive-time-stamp-v3 attribute (see ETSI EN 319 122-1 V1.1.1 (2016-04)). Data objects are objects covered by fields FILE, HASH and NOTICE.


Data object
PoE attribute as an unsigned attribute
(*preservation-integrity-list* attribute)
PoE object with *ATSHashIndexV3*
- archive-time-stamp-v3,
- LTI PoE Attribute or
- ERS PoE Attribute
See Figure E.2
time
Any data object
Data object
Data object digital signature
AdES unsigned attributes
...
preservation-integrity-list attribute
– FILE URL
– HASH DId of data object
– NOTICE Type of data object
...
– FILE URL
– HASH DId of data object
– NOTICE Type of data object
...
...
archive-time-stamp-v3 attribute
Archive-time-stamp-v3 unsigned attribute
Archive-time-stamp-v3 digital signature

```
. . .
ATSHashIndexV3 ::= SEQUENCE {
  hashIndAlgorithm AlgorithmIdentifier,
  certificatesHashIndex
                 SEQUENCE OF OCTET STRING,
  crlsHashIndex
                 SEQUENCE OF OCTET STRING,
  unsignedAttrValuesHashIndex
                 SEQUENCE OF OCTET STRING,
}
```

Figure 7 — Example of the preservation-integrity-list attribute as an unsigned attribute

5 Types of PoE objects with their essential fields

5.1 General

This document specifies additional PoE objects, other than already defined PoE objects like signature timestamp (see IETF RFC 3161), long term availability and integrity (LTA) objects (see ETSI EN 319 122-1 or ETSI EN 319 132-1), long term integrity (LTI) objects (see 4.3 — Attribute LTI PoEAttribute), PDF document timestamp (DTS) object (see ISO 32000-2), ASiC document timestamp (see ETSI EN 319 162-1 "timestamp*.tst" timestamp token file), Evidence Record (see RFC4998, RFC6283, and ETSI EN 319 162-1 "evidencerecord.ers" or "evidencerecord.xml" and ETSI EN 319 122-3).

5.2 PoE object of status at *thisUpdate* time value based on *CertHash* OCSP *SingleResponse* extension

The OCSP responder may include this extension in OCSP response to send the hash of the requested certificate to the requestor. The hash serves as an evidence that the certificate is known to the responder (i.e. it is available in the queried directory — the certificate database) and will be used as means of providing a positive statement (proof) of existence and the known certificate validity status at the time value stored in the *thisUpdate* OCSP response field.

The added value of the positive statement, based on information from the certificate database, is also in the past validation, where

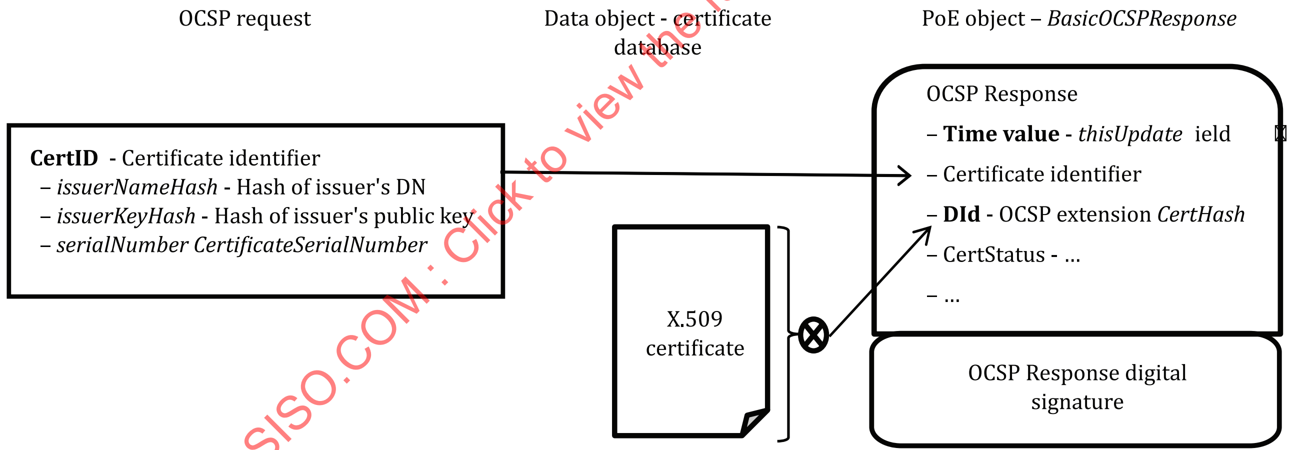
— the OCSP response signature is verified with a presently valid certificate of the OCSP response, and

— the presently more secure hash algorithm is used to compute the hash value of the requested certificate stored in the *CertHash* OCSP *SingleResponse* extension, which protects the content of the requested certificate against considering the certificate as invalid, when

  — the hash algorithm or the private signing key of the CA, used as secure in the past for issuing the certificate, is presently assumed compromised, or

  — the associated CA public key certificate is expired and therefore the certificate signature is assumed invalid.

The PoE object in this case is the proof of existence of certificate (*certHash*) at *thisUpdate* date and time value at which the validity status is provided. Figure 8 gives an overview of the used elements.


OCSP request
Data object - certificate database
PoE object – *BasicOCSPResponse*
CertID - Certificate identifier
– *issuerNameHash* - Hash of issuer's DN
– *issuerKeyHash* - Hash of issuer's public key
– *serialNumber CertificateSerialNumber*
X.509 certificate
OCSP Response
– **Time value** - *thisUpdate* ield
– Certificate identifier
– **DId** - OCSP extension *CertHash*
– CertStatus - ...
– ...
OCSP Response digital signature

NOTE OCSP request contains only the certificate serial number.

Figure 8 — *CertHash* positive statement, based on information from the certificate database

The *CertHash* OCSP *SingleResponse* extension is defined in the Annex B.

According to IETF RFC 6960, *thisUpdate* is the most recent time at which the status being indicated is known by the responder to have been correct.

The content of *thisUpdate* field is the PoE of the validity status of the certificate, valid at the date and time value stored in the *thisUpdate* field when

— the certificate identifier is not included in the CRL, or

— the OCSP response contains a positive statement (e.g. extension *CertHash*) of the certificate whose identifier is included in the OCSP response and the value of the *CertStatus* OCSP field is set to ***good***.

It means the certificate is valid at *thisUpdate* time value whereas a revocation of the certificate may occur only after *thisUpdate* time value or the certificate can later expire.

If the OCSP response contains the *CertHash* OCSP *SingleResponse* extension, see Figure 8, the OCSP response provides to any relying party information on the validity of certificates, known at the time value stored in the *thisUpdate* field, details are given in Annex B. For more information see Figures E.3, E.4 and E.5. Figures E.3 and E.4 describe the usage of the *thisUpdate* field in validation conditions. E.5 and Figure E.5 provides the time consequences of the *thisUpdate* field usage.

5.3 PoE object supported by LTI PoEAttribute or ERS PoEAttribute

If the PoE object is supported by the LTI *PoEAttribute* instance or the ERS *PoEAttribute* instance, then the *PoEAttribute* instance shall contain the *dataObjectRefs* field and the *poEObject* field shall be an optional storage for the PoE object

— IETF RFC 3161 timestamp, the field *poEObjectRef.type* shall contain the OID value *id-poe-lti-rfcts* (see 4.3 for more details), or

— IETF RFC 4998/IETF RFC 6283 evidence record object, the field *poEObjectRef.type* shall contain the OID value *id-poe-lti-ers* or *id-poe-ers* (see 4.3 and 4.4 for more details).

The evidence record object is itself a PoE attribute where one or more PoE objects are included as, e.g. IETF RFC 3161 timestamp of structures where *PartialHashtree* is used, see Annex H.

Annex A
(normative)

ASN.1 module

This annex lists the ASN.1 module assigned to the attributes pointing to external proof of existence objects used in long term signature formats specified in this document.

```
LongtermSignatureProfilesPoEAttributes {
   iso(1) standard(0) long-term-signature-profiles(14533) part(4)
    asn1-module(1) poe-attributes(0) }
DEFINITIONS IMPLICIT TAGS ::= BEGIN
-- EXPORTS All; --
IMPORTS
-- Imports from IETF RFC 3161
   MessageImprint
    FROM PKIXTSP {
     iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-tsp(13)}
-- Imports from ETSI EN 319 122-1 V1.1.1 (2016-04)
ATSHashIndexV3
   FROM ETSI-CAdES-ExplicitSyntax88 { itu-t(0) identified-organization(4)
      etsi(0) cades(19122) id-mod(0) cades-explicit88(0)}
id-PoEA OBJECT IDENTIFIER ::= { iso(1) standard(0)
    long-term-signature-profiles(14533) part(4) attribute(0) }
id-PoEAttribute OBJECT IDENTIFIER ::= { id-PoEA poeattribute(1) }
id-poe-lti-rfcts OBJECT IDENTIFIER ::= { id-PoEAttribute poe-lti-rfcts(1) }
id-poe-lti-ers OBJECT IDENTIFIER ::= { id-PoEAttribute poe-lti-ers(2) }
id-poe-ers OBJECT IDENTIFIER ::= { id-PoEAttribute poe-ers(3) }
URL ::= UTF8String -- see 6.2.12 of Rec. ITU-T X.520 | ISO/IEC 9594-6
DSId ::= MessageImprint
DId ::= MessageImprint
PoEAttribute ::= SEQUENCE {
       poEObjectRef ObjectRef,
       poEObject OCTET STRING OPTIONAL,
       dataObjectRefs DataObjectRefs OPTIONAL }
DataObjectRefs ::= SEQUENCE OF ObjectRef
ObjectRef ::= SEQUENCE {
       type OBJECT IDENTIFIER,
       objectId ObjectId,
       location URL OPTIONAL,
       additionalData [1] EXPLICIT AdditionalData OPTIONAL,
       partialHash [2] EXPLICIT MessageImprint OPTIONAL,
       contentDescription UTF8String OPTIONAL }
ObjectId ::= CHOICE {
       dSId DSId,
       dId [0] EXPLICIT DId }
AdditionalData ::= SEQUENCE {
       dataType OBJECT IDENTIFIER,
       otherData ANY DEFINED BY dataType }
id-poe-hash-index OBJECT IDENTIFIER ::= {
   id-PoEA poe-hash-index(2) }
PoEHashIndex ::= ATSHashIndexV3
id-preservation-integrity-list OBJECT IDENTIFIER ::= {
   id-PoEA preservation-integrity-list(3) }
PreservationIntegrityList ::= UTF8String
id-TStOCSP OBJECT IDENTIFIER ::= { iso(1) standard(0)
   long-term-signature-profiles(14533) part(4) certificate-policy(1)
   timestamp-through-OCSP(1) }
END -- LongtermSignatureProfilesPoEAttributes –
```

Annex B
(normative)

Definition of the *CertHash* OCSP *SingleResponse* extension

This annex defines the *CertHash* OCSP *SingleResponse* extension for common use. The following definition is a revision of the Part 4, Chapter 3.12, Table 15 of the Common PKI specification 2.0[5], where this extension was defined as Common PKI Private OCSP Extension.

```
CertHash (Positive Statement) single OCSP extension:
id-commonpki-at-certHash OBJECT IDENTIFIER ::= {1 3 36 8 3 13}
CertHash ::= SEQUENCE {
   hashAlgorithm  AlgorithmIdentifier,
   certificateHash OCTET STRING }
   -- A hash over the DER-encoding of the entire certificate
```

According to IETF RFC 6960, the "good" status indicates a positive response to the status inquiry. At minimum, this positive response indicates that no certificate with the requested certificate serial number, currently within its validity interval, has been revoked. This status does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval. Response extensions may be used to convey additional information on assertions made by the responder regarding the status of the certificate, such as a positive statement about issuance, validity, etc.

The responder may include the Common PKI Profile extension in response to send the hash of the requested certificate to the responder. This hash is cryptographically bound to the certificate and serves as an evidence that the certificate is known to the responder (i.e. it has been issued and is present in the directory — the certificate database). Hence, this extension is a means to provide a positive statement of availability. Clients may rely on this information to be able to validate signatures after the expiry of the corresponding certificate. Hence, clients shall support this extension. The OCSP *ArchiveCutoff* extension with an appropriate content should be present and independent of whether *CertHash* is present or not.

If a positive statement of availability is to be delivered, this extension syntax and OID shall be used. A further note on security is that including the hash of the queried certificate in the response prevents impersonation attacks of the following scenario:

Mallory manages to get the private key of a CA. The corresponding CA certificate is immediately revoked. Using the stolen CA key, Mallory creates a fake certificate with the identical serial number as an existing (original) one but with a new public key. Using the corresponding private key, Mallory signs a message and sends it, along with the fake certificate, to Alice. Alice succeeds in mathematically verifying the signature and wants to check the status of the received certificate by sending its serial number to the OCSP server. The server returns the answer good, if the original certificate has not been revoked. Having received the response good, Alice thinks that the (actually fake) certificate is OK and accepts the signature. She is unable to detect that the response corresponds to a different certificate than the one she was asking about.

This threat is apparently not handled by PKIX documents. The security gap can be closed by including either the certificate or a fingerprint of it in the response, respectively in the positive statement as proposed here. It is crucial that the signature of the responder can be reliably verified. Hence, departing from the practice proposed by IETF RFC 6960, the certificate of the responder should be issued by some independent CA, i.e. not by the CA the certificates of which the responder provides information about.

Annex C
(normative)

Signature timestamp as a timestamp through OCSP

The signature timestamp implemented as a timestamp through OCSP is based on the field *MessageImprint* defined in IETF RFC 3161, used in this document as DSId.

NOTE 1 The signed attributes of the signature can contain the signature policy attribute containing OID of the signature policy declaring the OCSP usage as an electronic timestamp, e.g. 1.3.6.1.4.1.10015.1000.3.2.3 OID of the signature policy in a human readable PDF document or 1.3.158.36061701.1.2.2 OID of the signature policy in an ASN.1 DER (see ISO/IEC 8825-1) encoded document for machine processing. The use of the signature policy OID or the certificate policy OID 1.3.158.36061701.1.3.2 included in the OCSP response signer certificate in the certificate policy extension is mentioned only for backward compatibility.

The formats of the OCSP request and the OCSP response are defined in IETF RFC 6960.

The OCSP *producedAt* field contains the time value used as the timestamp time.

The input to the hash calculation function is independent from the type of the electronic signature format. The input to the hash calculation function, being timestamped, shall be the DER encoded digital signature:

— for CMS signature, the value of *OCTET STRING* of *SignatureValue* type without *OCTET STRING TAG* and *LEN* of the signature field within *SignerInfo* of the *signedData* is taken;

— for XML signature, the content of the <*SignatureValue*> element (i.e. the value without XML tags) decoded from Base64 encoding is taken.

EXAMPLE The input to the hash calculation function is DER encoded result of the asymmetric function:

```
SEQUENCE {
 INTEGER 2D792927A77E87125091439D3A35990EDA4B00E49931356593A8DC7EC6EC0D01
 INTEGER 26E51D1E0C71555635O3DDBE6A9AF6DD4233A0B10869018DEE41C491246951D0
}
```

where

`SEQUENCE`	is the ASN.1 type of DER encoded object;
`INTEGER`	is e.g. value of the elliptic curve. In this case the result of the elliptic curve consists of two INTEGER values.

The value of the result of the hash calculation function is stored in the *MessageImprint* field defined in IETF RFC 3161. The *MessageImprint* field is included in the OCSP extension Nonce defined in IETF RFC 6960. The OCSP extension Nonce is included in the OCSP request and the OCSP responder has used the extension Nonce from the OCSP request and includes the same value in the OCSP extension Nonce of the OCSP response.

NOTE 2 According to IETF RFC 6960, Section 4.4.1, the *nonce* cryptographically binds a request and a response to prevent replay attacks. The instance of the *MessageImprint* field included in the OCSP extension *Nonce*, as implementation of the electronic timestamp through OCSP, is a correct value of the *Nonce*, because the *Nonce* value is unique for each timestamped signature.

The signature validation procedure performs a test, which confirms

— the possibility of decoding the DER value of the OCSP extension *Nonce* as the ASN.1 structure of the *MessageImprint* type, and

— after a successful comparison of the hash value from the *MessageImprint* with a hash value of the digital signature, the signature validation procedure can attest the expected security level by the following identification of the service of the signature timestamp through OCSP:

  — the OCSP responder certificate can be included in the trusted list (TL) as a trusted service, and

  — the type of service "signature timestamp through OCSP" can be identified by the certificate policy OID "1.0.14533.4.1.1" — *id-TStOCSP* included in the OCSP response signer certificate in the certificate policy extension. The certificate policy document, identified by *id-TStOCSP* OID, shall include these additional requirements:

    — the time shall be included in the field *producedAt* of OCSP response defined in IETF RFC 6960 with an accuracy of 1 s as a minimum (the OCSP response shall not be pre-generated);

    — *Nonce* extension is supported as defined in IETF RFC 6960 (Online Certificate Status Protocol).

Annex D
(normative)

Syntax of the ASN.1 object location in ZIP, PDF container or in DER encoded ASN.1 object

The following rules are used to parse the location mainly included in *ObjectRef.location* of *PoEAttribute* pointing to the DER encoded field of an ASN.1 object optionally stored in a ZIP or PDF container, accessible locally or through network, by defining the additional *fragment* usage where the location containing the *fragment* is defined in IETF RFC 7230, Section 2.7 as URI.

This Annex defines the structure of the *fragment* defined in IETF RFC 3986, Section 4.2 "Relative Reference" (see an example of the link with the fragment) where the sign "#" separator is followed by a fragment identifier (Fragment, see IETF RFC 3986, Section 3.5).

The syntax of the fragment identifier, extended according to rules defined in this document, shall follow the separator sign "{". The separator sign "{" by itself can create a hierarchy and can be repeated. The closing separator sign "}" of the pair should not be used.

The text of the fragment identifier shall consist of these types "{URL-x", "{DSId-x", "{DId-x" or "{ASN-x", where sign "-" is minus and ‘x’ shall be as follows:

— If the text is "{URL-x", the x shall be a URL of the object, as defined in this Annex.

— If the text is "{DSId-x", the x shall be a "base64url" (IETF RFC 4648) encoded DSId of the data object whose hash of the digital signature is equal to this value, e.g. signed PDF document of many subsequent PDF signatures in one PDF file. The recognition of the type of the signed object can be e.g. based on the content of text preceding the text "{DSId-x", where the file name extension can be used (see Example 1) or other methods for detection of the signed object type which can be based on the context in which the location is used.

— If the text is "{DId-x", the x shall be a "base64url" (IETF RFC 4648) encoded DId of the object whose hash value is equal to this value, e.g. file in directory (see Example 3).

— If the text is "{ASN-x", the x shall be a sequence of searching values separated by the sign "|" of the following syntax, which moves the byte index from zero (first byte) in the ASN.1 DER encoded object. The index value "-1" means, the search is unsuccessful. The final result of search "{ASN-x" is the index of the searched DER object within the browsed DER object. The searching values are as follows:

  — "X**n**Y" means the index is moved to the position of "X" type (uppercase hex ASN.1 DER type) according to the next "Y" occurrences of "X" type on the same level of the ASN.1 hierarchy (sibling).

  — "X**i**Y" means the index is moved to the first inner object (child) of e.g. "SET OF" or "SEQUENCE OF" and then the index is moved as in "X**n**Y".

  — "SZOIDx.x.x" means the index is moved according to the search of object on the same level of the ASN.1 hierarchy (sibling). The search of object is based on "ZOIDx.x.x", where "Z" is a sequence of one or more "X**n**Y" or "X**i**Y" separated by the sign "-", representing temporally moved index of ASN.1 DER objects to the inner OID object which is to be compared with the OID "x.x.x" value in dot notation.

  — "DSId-x" means the index is moved to the object on the same level of the ASN.1 hierarchy (sibling) which contains the expected signature identified by DSId. The x shall be a "base64url" (IETF RFC 4648) encoded DSId of the data object whose hash of the digital signature is equal to this value. The type of signed object is based e.g. on the previous sequence of one or more "XnY" or "XiY" in the ASN.1 hierarchy or other methods for detection of the signed object type which can be based on the context in which the search is used.

— "DId-x" means the index is moved to the object on the same level of the ASN.1 hierarchy (sibling), which represents the expected object identified by DId. The x shall be a "base64url" (IETF RFC 4648) encoded DId of the object whose hash value is equal to this value.

The list of some searching values if the text is "{ASN-x" and the ASN.1 object is the CMS signature is:

— `"A0i0|30i0|31i1|"` if the result is *SET OF SignerInfo*;

— `"A0i0|30i0|A0i0|"` if the result is *certificates [0] IMPLICIT CertificateSet OPTIONAL*;

— `"A0i0|30i0|A1i0|30i0|"` if the result is the first object from *SET OF RevocationInfoChoice* (CRL — SEQUENCE);

— `"A0i0|30i0|A1i0|A1i0|S06i0OID1.3.6.1.5.5.7.48.1.1"` if the result is the *OtherRevocationInfoFormat* object with the first basic OCSP response from *SET OF RevocationInfoChoice* (OCSP — *other [1] IMPLICIT OtherRevocationInfoFormat*).

EXAMPLE 1 The result is a PDF document signed by the CMS signature stored in PDF, where the one of the many subsequent PDF signed documents in PDF file has the CMS signature value equal to DSId:

```
ENISAReport.pdf#{DSId-MDEwDQYJYIZIAWUDBAIBBQAEIIc5x25oH5AJI7kAyd8O91z0IdOcq7VGUMS5rRm2p22F
```

where

`ENISAReport.pdf`	is the PDF file;
`DSId-MDEwDQYJYIZIAWUDBAIBBQAEIIc5x25oH5AJI7kAyd8O91z0IdOcq7VGUMS5rRm2p22F`	is the identifier of the signature of the document as "base64url" encoded DER of DSId.

EXAMPLE 2 The result is the first DER encoded object *SignerInfo* of the CMS signature of the document stored in ZIP in a signature.p7s file, where the one of the parallel signatures has the "base64url" hash value of the digital signature (*SignerInfos ::= SET OF SignerInfo*):

```
ENISAReport.asc#{URL-/META-INF/signature.p7s{ASN-A0i0|30i0|31i1|30i0|DSId-MDEwDQYJYIZIAWUD
BAIBBQAEIIc5x25oH5AJI7kAyd8O91z0IdOcq7VGUMS5rRm2p22F
```

where

`ENISAReport.asc`	is the ASiC file;
`#{URL-/META-INF/signature.p7s`	is the URL of the DER signature file in ASiC;
`{ASN-A0i0|30i0|31i1|30i0`	is the: (content [0] EXPLICIT ANY DEFINED BY contentType, }), (SignedData ::= SEQUENCE {), (SignerInfos ::= SET OF SignerInfo), (SignerInfo);
`DSId-MDEwDQYJYIZIAWUDBAIBBQAEIIc5x25oH5AJI7kAyd8O91z0IdOcq7VGUMS5rRm2p22F`	is the signature identifier — hash of the signature value (without TAG and LEN of OCTET STRING) of the signature (SignatureValue ::= OCTET STRING) of the document compared with the hash value stored in "base64url" encoded DER of the DSId instance.

EXAMPLE 3 The result is a document, where one of the many documents in directory has the hash value equal to the hash value stored in the DId instance:

```
ENISAReport.zip#{URL-/ENISA/dat?.*{DId-MDEwDQYJYIZIAWUDBAIBBQAEIIc5x25oH5AJI7kAyd8O91z0IdO
cq7VGUMS5rRm2p22F
```

where

`ENISAReport.zip`	is the ZIP container;
`#{URL-/ENISA/dat?.*`	is the URL of the files in ZIP directory;

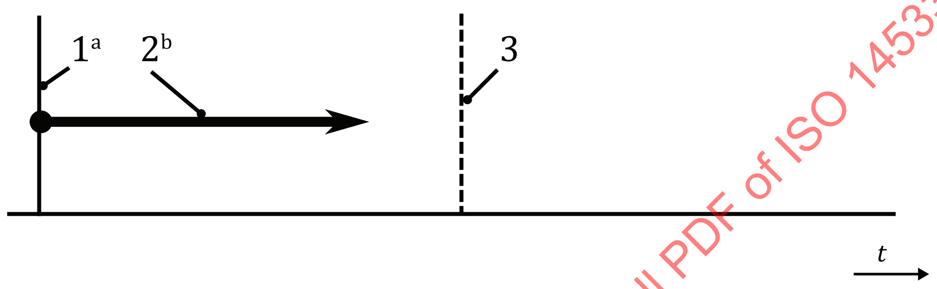
`{DId-MDEwDQYJYIZIAWUDBAIBBQAEIIc5x25oH5AJI7kAyd8O91z0IdOcq7VGUMS5rRm2p22F`	is the document identifier (the hash of the file).

Annex E
(normative)

Use of the PoE objects

E.1 Signature created after the time value stored in the PoE object

The *PoEHashIndex* signed CMS attribute is a PoE attribute that the CMS signature was created after the time value stored in the *thisUpdate* field of the CRL or in the *producedAt* field of the OCSP response included in the CMS signature.



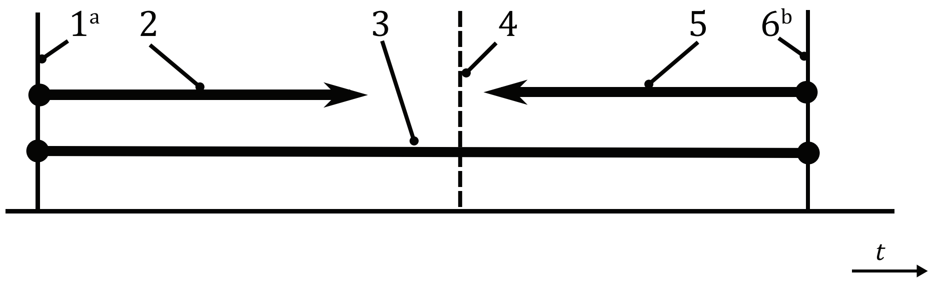
Key

t	time
1	PoE
2	the half-closed interval in which the signature was created
3	the factual time of the signature creation of data (electronic document)
a	The signature was created after the time value stored in *thisUpdate* field of the CRL or in *producedAt* field of the OCSP response covered by the PoEHashIndex signed attribute or covered by the preservation-integrity-list signed attribute or covered by the PoEAttribute signed attribute. The signature was created after the time value stored in the signed attribute like content timestamp.
b	The signature covers the objects listed in Key 1 covered by the hash value.

Figure E.1 — PoE of the half-closed interval

E.2 PoE objects used to specify the interval of the signature creation

The validation procedure can use many kinds of PoE objects to specify the closed interval in which the signature was created at a given time, see ISO/IEC 9594-8:2017, 3.5.58. The objects used as the PoE can be internal or external objects of the signature. Internal objects are used in the Figure E.2. External objects can be, e.g. the objects included in signatures which cover the content to the digital signature value, like PDF (ISO 32000-2) subsequent signatures, PDF document timestamps or external record like evidence record defined in IETF RFC 4998 or IETF RFC 6283.



Key

t	time
1	PoE
2	interval in which the signature covers the objects listed in Key 1 covered by the hash value
3	the closed interval in which the signature was created (given-time)
4	the factual time of the signature creation of data (electronic document)
5	interval — the objects listed in Key 6 cover the value of the digital signature covered by the hash value
6	PoE

[a] The signature was created after the time value stored in

— *thisUpdate* field of the CRL or in the *producedAt* field of the OCSP response covered by the *PoEHashIndex* signed attribute or covered by the PoEAttribute signed attribute,

— the content timestamp (CTS) attribute, or

— the objects (the timestamp, in *thisUpdate* field of the CRL or in *producedAt* field of the OCSP response) of the previous signature covered by the signature or covered by the preservation-integrity-list signed attribute.

[b] The signature was created before the time value stored in

— the signature timestamp (STS) defined in IETF RFC 3161,

— *producedAt* field of the OCSP response when the OCSP *Nonce* extension contains *MessageImprint* field, defined in IETF RFC 3161, covering the value of the digital signature as the signature timestamp (STS) implemented through OCSP,

— the timestamp of the subsequent signature covering the signature value of the digital signature,

— the PDF subsequent document timestamp, or

— the hash value included in the external objects covering the signature value of the digital signature, e.g. timestamp of preservation service record or the Evidence Record defined in IETF RFC 4998 or IETF RFC 6283 or the attribute PoEAttribute with the *poEObjectRef.type* field with *id-TStOCSP* OID, *id-poe-lti-rfcts* OID, *id-poe-lti-ers* OID or *id-poe-ers* OID.

Figure E.2 — *given-time* in a PoE of the closed interval

E.3 Use of *thisUpdate* field of CRL in validation conditions as a PoE of status

Conditions of the validation procedure using CRL shall provide a complete and unalterable result:

```
1[a]  if ( certificate.notBefore < CRL.thisUpdate ) and
        ( ((CRL.expiredCertsOnCRL <= certificate.notAfter ) and ( 0 < CRL.expiredCertsOnCRL )) or
         (( CRL.thisUpdate <= certificate.notAfter ) and ( 0 = CRL.expiredCertsOnCRL ))) then
2         if certificate is not revoked in CRL then
3             if given-time <= CRL.thisUpdate then
                  VALID
4[b]          else
                  WAS VALID at [CRL.thisUpdate], INDETERMINATE
5         else
              if given-time < CRL[certificate].revocationDate then
                  VALID
6             else
                  INVALID – revoked on [ CRL[certificate].revocationDate ]
7[c]  else
          INDETERMINATE (INCOMPLETE AUTOMATIC VERIFICATION)
```

Key

1 the test of the time interval in which the CRL contains verification data

2 the certificate was not revoked; it is not in CRL

3 the certificate status in CRL is updated after *given-time*

4 the certificate was valid at the time value of CRL.*thisUpdate* field

5 the certificate was revoked after *given-time*, thus the certificate was valid at the *given-time*

6 the certificate was revoked before *given-time* on CRL[certificate].*revocationDate*

7 the time interval in which the CRL does not contain verification data

[a] CRL was updated in time of certificate validity + a period of time during which the record about the certificate revocation is listed in CRL even after the certificate expiration. The value of CRL.*expiredCertsOnCRL* is the number representing the content of the extension "Expired certificates on CRL" defined in ISO/IEC 9594-8. If "expired certificates on CRL" extension is not present in CRL the value of CRL.*expiredCertsOnCRL* shall be set to 0. The result TRUE of the test "0= CRL.*expiredCertsOnCRL*" allows testing the conditions connected by using "and".

CRL.*thisUpdate* is the time when the certificate status was updated, which means that the certificate status will not be changed to "*revoked"* before the time value *thisUpdate*. The certificate status can be changed only after the time value *thisUpdate*.

Certificate.*notBefore* is the time since when it is possible to use the certificate and its status can be included in CRL. Certificate.*notAfter* is the time after which the certificate status in CRL shall not be changed anymore but the status may be included in CRL.

[b] The later status is not confirmed.

If you need a confirmation of the later status, try to get a newer updated CRL.

CRL is not issued after *given-time*. When the status at *given-time* is necessary, the validation procedure shall wait for a new updated CRL (CRL.*thisUpdate >= given-time*).

[c] It is necessary to obtain CRL or OCSP response, which is updated at a time when the certificate has not been expired yet + a period of time in which the certificate status is still known in OCSP or CRL. Request CA for CRL that may contain the status of the certificate being verified.

CRL is updated before the certificate usage period, Certificate.*notBefore* time.

Figure E.3 — Validation using CRL

E.4 Use of *thisUpdate* field of OCSP response in validation conditions as a PoE of status

Conditions of the validation procedure using OCSP response shall provide a complete and unalterable result:

```
1[a]  if ( certificate.notBefore < OCSP[certificate].thisUpdate ) and
        ( (( OCSP.ArchiveCutoff <= certificate. notAfter ) and ( 0 < OCSP. ArchiveCutoff )) or
         (( OCSP[certificate].thisUpdate <= certificate.notAfter) and (0 = OCSP.ArchiveCutoff )) or
         (OCSP[certificate].CertHash = certificate.CertHash) ) then
2[b]        if OCSP[certificate].CertStatus = good then
3                   if given-time <=  OCSP[certificate].thisUpdate then
                            VALID
4[c]                else
                            WAS VALID at [OCSP[certificate].thisUpdate], INDETERMINATE
            else
5                   if OCSP[certificate].CertStatus = revoked then
                            if given-time <  OCSP[certificate].revocationTime then
                                    VALID
6[d]                        else
                                    INVALID - revoked at [ OCSP[certificate].revocationTime ]
                    else
7[e]                        INDETERMINATE (INCOMPLETE AUTOMATIC VERIFICATION:
                            OCSP[certificate].CertStatus = unknown )
8[f]  else
            INDETERMINATE (INCOMPLETE AUTOMATIC VERIFICATION )
```

Key

1 the test of the time interval in which the OCSP response contains verification data

2 the certificate was not revoked

3 certificate status in OCSP is updated after *given-time*

4 the certificate was valid at the time value of OCSP[certificate].*thisUpdate* field

5 the certificate was revoked after *given-time*, thus the certificate was valid at the *given-time*

6 the certificate is revoked in OCSP response before *given-time*

7 OCSP response is not able to determine a certificate status, it is necessary to try other OCSP responder or CRL

8 the time interval in which the OCSP response does not contain verification data

a OCSP was updated in time of certificate validity + a period of time during which the record about the certificate revocation for OCSP is known even after the certificate expiration. Certificate.*notBefore* is the time since when it is possible to use the certificate and the certificate status can be included in OCSP (CRL). Certificate.*notAfter* is the time after which the certificate status in CRL (OCSP) shall not be changed but the certificate status can be included in CRL (OCSP).

The value of OCSP.*ArchiveCutoff* is the number representing the content of the extension *ArchiveCutoff* defined in IETF RFC 6960. If the extension *ArchiveCutoff* is not present in OCSP response the value of OCSP.*ArchiveCutoff* shall be set to 0. The result TRUE of the test "0=OCSP.*ArchiveCutoff*" allows testing the conditions connected by using "and".

OCSP.*ArchiveCutoff* — if OCSP extension *ArchiveCutoff* is not present in the OCSP response, then OCSP. *ArchiveCutoff* value shall be 0; otherwise the *ArchiveCutoff* value shall be according to *ArchiveCutoff* extension defined in IETF RFC 6960.

OCSP[certificate].*CertHash* is the hash value of the certificate whose status is confirmed by the OCSP response. See Annex B. If this extension is found in the OCSP response, the certificate status is known to OCSP and the hash value ensures the integrity by currently secure hash algorithm. Certificate.*CertHash* is the hash value of the certificate whose status is verified.

OCSP[certificate].*thisUpdate* is the time when the certificate status was updated, which means the certificate status will not be changed to "*revoked*" with the time value before the time value *thisUpdate.* The certificate status may be changed only after the time value *thisUpdate*. The value shall be smaller or equal to OCSP. *producedAt*. OCSP.*producedAt* is the time of the OCSP response issuance.

OCSP[certificate].*nextUpdate* is the auxiliary time when the latest occurrence of the information about the status will be available.

b OCSP[certificate].*CertStatus* is the status of the certificate being verified with the values: *good*, *revoked* or *unknown*.