Draft EN 319 132-1 V0.0.4 (2013-11)

Electronic Signatures and Infrastructures (ESI);

XML Advanced Electronic Signatures (XAdES);

Part 1: Core Specification

STABLE DRAFT FOR PUBLIC REVIEW UNTIL 15 JANUARY 2014

Download the template for comments:

http://docbox.etsi.org/ESI/Open/Latest_Drafts/Template-for-comments.doc

Send comments to E-SIGNATURES_COMMENTS@LIST.ETSI.ORG

0

1

2

Reference
DEN/ESI-0019132-1

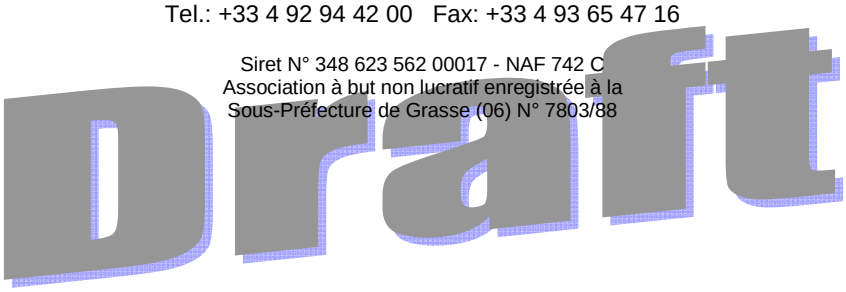
Keywords
<keywords>

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88



Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This draft European Standard (EN) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI) and is now submitted for getting public comments from stakeholders.

The present document is part 1 of a multi-part deliverable covering the XML Advanced Electronic Signatures (XAdES), as identified below:

Part 1: Core Specification

Part 2: XAdES Baseline Profile

Proposed national transposition dates	
Date of latest announcement of this EN (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

Introduction

Electronic commerce has emerged as a frequent way of doing business between companies across local, wide area and global networks. Trust in this way of doing business is essential for the success and continued development of electronic commerce. It is therefore important that companies using this electronic means of doing business have suitable security controls and mechanisms in place to protect their transactions and to ensure trust and confidence with their business partners. In this respect the electronic signature is an important security component that can be used to protect information and provide trust in electronic business.

The European Directive on a community framework for Electronic Signatures (also denoted as "the Directive" or the "European Directive" in the rest of the present document) defines an electronic signature as: "data in electronic form which is attached to or logically associated with other electronic data and which serves as a method of authentication".

The present document is intended to cover electronic signatures for various types of transactions, including business transactions (e.g. purchase requisition, contract and invoice applications). Thus the present document can be used for any transaction between an individual and a company, between two companies, between an individual and a governmental body, etc. The present document is independent of any environment. It can be applied to any environment e.g. smart cards, GSM SIM cards, special programs for electronic signatures, etc.

The present document:

- specifies XML schema [4] definitions for new XML types that can be used to generate properties that further qualify XMLDSIG signatures with information able to fulfil a number of common requirements such as the long term validity of the signature by usage of time-stamps, etc.;
- defines mechanisms for incorporating the aforementioned qualifying information;
- specifies formats for XML advanced electronic signatures that, by using the specified new XML types, remain valid over long periods and incorporate additional useful information in common use cases. These signatures will be built on XMLDSIG by encapsulating these properties (and/or references to them) within one `ds:Object` XML element defined in [2]. Here, as for the rest of the document, `ds` has been used as the prefix denoting the namespace whose URI value is “http://www.w3.org/2000/09/xmldsig#”;
- defines a set of conformance requirements to claim endorsement to the present document.

The present document specifies two main types of properties: signed properties and unsigned properties. The first ones are additional data objects that are also secured by the signature produced by the signer on the `ds:SignedInfo` element, which implies that the signer gets these data objects, computes a hash for all of them and generates the corresponding `ds:Reference` element. The unsigned properties are data objects added by the signer, by the verifier or by other parties after the production of the signature. They are not secured by the signature in the `ds:Signature` element (the one computed by the signer); however they can be actually signed by other parties (time-stamps, countersignatures, certificates and CRLs are also signed data objects).

EDITOR NOTE: a number of editor notes like this one appears througouht the document. These notes intend to attract readers’ attention and/or kindly request their feedback on certain specific issues.

EDITOR NOTE: ESI is assessing whether it is suitable or not to incorporate, in a normative annex, part of the material present in ETSI TS 101 903 v1.4.2, annex G: "Details on XAdES signatures validation". ESI has issued ETSI TS 102 853: "Signature verification procedures and policies", and will issue, based on this document, the future ETSI EN 319 102: "Procedures for signature creation and validation". It is the intention that this last document presents a validation procedures without entering in issues specific to the formats. In consequence, ESI is assessing if relevant validation issues specific to XAdES format would be missed if this annex is completely suppressed. Feedback from stakeholders regarding the suitability of keeping parts of such an annex would be highly appreciated.

1 Scope

The present document defines XML [6] formats for advanced electronic signatures that remain valid over long periods, are compliant with the European Directive and incorporate additional useful information in common uses cases. This includes evidence as to its validity even if the signer or verifying party later attempts to deny (repudiates) the validity of the signature.

The present document is based on the use of public key cryptography to produce digital signatures, supported by public key certificates.

The present document uses a signature policy, implicitly or explicitly referenced by the signer, as one possible basis for establishing the validity of an electronic signature.

The present document uses time-stamps or trusted records (e.g. time-marks) to prove the validity of a signature long after the normal lifetime of critical elements of an electronic signature and to support non-repudiation. It also specifies the optional use of additional time-stamps to provide very long-term protection against key compromise or weakened algorithms.

The present document then, specifies the use of the corresponding trusted service providers (e.g. time-stamping authorities), and the data that needs to be archived (e.g. cross certificates and revocation lists).

An advanced electronic signature aligned with the present document can, in consequence, be used for arbitration in case of a dispute between the signer and verifier, which may occur at some later time, even years later.

The present document:

- shows a taxonomy of the qualifying information (properties) whose presence in an electronic signature allows it to remain valid over long periods, to satisfy common use cases requirements, and to be compliant with the European Directive;
- specifies XML schema definitions for new elements able to carry or to refer to the aforementioned properties;
- specifies two ways for incorporating the qualifying information to XMLDSIG, namely either by direct incorporation of the qualifying information or using references to such information. Both ways make use of mechanisms defined in XMLDSIG.

Clause 4 gives an overview of some of the various types of advanced electronic signatures defined in the present document.

Clause 5 defines the namespaces used in the XML schema definitions appearing in the present document. It also defines the types for the containers of the qualifying properties, and specifies the mechanisms for incorporating them to the electronic signature.

Clause 6 defines the qualifying properties except those properties that contain references to validation data, time-stamps on these properties and a former container for archive time-stamps, nowadays superseded but kept for the sake of legacy XAdES signatures .

Clause 7 defines different conformance levels that may be claimed against the present document.

Normative Annex A defines qualifying properties that encapsulate references to validation data, properties that encapsulate time-stamp tokens for these references, and properties that have been obsoleted by new properties (for preserving management of legacy electronic signatures).

Normative Annex B defines XAdES forms that incorporate properties encapsulating references to validation data and properties encapsulating time-stamp tokens for these references.

Normative Annex C specifies conformance levels for XAdES signatures that incorporate properties encapsulating references to validation data and properties encapsulating time-stamp tokens for these references.

Informative Annex D provides rationale for the properties and mechanisms specified by this document.

Informative Annex E provides some information of the main changes introduced in the present document with regards to ETSI TS 101 903 [i.2].

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references,only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 101 903: "Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)".

[2] W3C Recommendation: "XML-Signature Syntax and Processing. Version 1.1".

[3] W3C Recommendation Part 1 (28 October 2004): "XML Schema Part 1: Structures".

[4] W3C Recommendation Part 2 (28 October 2004): "XML Schema Part 2: Datatypes".

[5] ITU-T Recommendation X.509: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".

[6] W3C Recommendation (26 November 2008): "Extensible Markup Language (XML) 1.0".

[7] IETF RFC 2560: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

[8] IETF RFC 3161: "Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP)".

[9] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[10] IETF RFC 3061: “A URN Namespace of Object Identifiers”.

[11] ETSI EN 319 102: “Electronic Signatures and Infrastructures (ESI); Procedures for Signature Creation and Validation”.

[12] ETSI EN 319 172: “Electronic Signatures and Infrastructures (ESI); Signature Policies”

[13] IETF RFC 6931: “Additional XML Security Uniform Resource Identifiers (URIs)”

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area

[i.1] ETSI EN 319 411-4: "Electronic Signatures and Infrastructures (ESI); Policy and security requirements for Trust Service Providers issuing certificates; Part 4: Policy requirements for certification authorities issuing attribute certificates".

[i.2] ETSI TS 101 903 v1.4.2: “Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signature (XAdES)”

3 Definitions, symbols and abbreviations

3.1 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AARL	Attribute Authority Revocation List
AC	Attribute Certificate
ASN.1	Abstract Syntax Notation 1
BER	Basic Encoding Rules
CA	Certification Authority
CER	Canonical Encoding Rules
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules
DTD	Document Type Definition
ES	Electronic Signature
HTTP	Hyper Text Transfer Protocol
OCSP	Online Certificate Status Protocol
OID	Object IDentifier
PER	Packed Encoding Rules
PKC	Public Key Certificate
TSA	Time-Stamping Authorities
TSP	Trusted Service Providers
TSU	Time Stamping Unit
URI	Uniform Resource Identifier
URN	Uniform Resource Name
XAdES	XML Advanced Electronic Signature
XAdES-A	XAdES Archiving validation data
XAdES-BES	XAdES Basic Electronic Signature
XAdES-C	XAdES Complete validation data
XAdES-EPES	XAdES Explicit Policy based Electronic Signature
XAdES-T	XAdES with Time-stamp
XAdES-X	XAdES eXtended validation data
XER	XML Encoding Rules
XML	eXtensible Markup Language
XMLDSIG	eXtensible Markup Language Digital SIGnature

XSLT eXtensible Stylesheet Language Transformations

4 Overview

The present document defines a set of signature properties that may be combined to obtain electronic signature forms providing satisfaction of different requirements. Below follows a short overview of the properties:

- `SigningCertificate` and `xadesenv111:SigningCertificate` . These properties contain an unambiguous reference to the signer's certificate, formed by its identifier and the digest value of the certificate. Its usage is particularly important when a signer holds a number of different certificates containing the same public key, to avoid claims by a verifier that the signature implies another certificate with different semantics. This is also important when the signer holds different certificates containing different public keys in order to provide the verifier with the correct signature validation data. Finally, it is also important in case the issuing key of the CA providing the certificate would be compromised. Details on these properties can be found in clause 6.2.2.

- `SignaturePolicyIdentifier`. This property contains information being an **unambiguous way for identifying the signature policy** under which the electronic signature has been produced. This will ensure that the verifier will be able to use the same signature policy during the validation process. A signature policy is useful to clarify the precise role and commitments that the signer intends to assume with respect to the signed data object, and to avoid claims by the verifier that the signer implied a different signature policy. Details on this property can be found in clause 6.2.9.

- Validation data properties. The present document defines a number of XML types able to contain both validation data (certificate chains, CRLs, OCSP responses, etc.) and references to them (identifiers of certificates, CRLs, OCSP responses, etc.). Properties of these types allow to incorporate all material used for validation into the signature. They can be jointly used with time-stamp properties to provide long-term validity. Some of them contain validation data:

  - `CertificateValues`. It contains the values of certificates used to validate the signature. Details on this property can be found in clause 6.4.1.

  - `RevocationValues`. It contains revocation information used for the validation of the electronic signature. Details on this property can be found in clause 6.4.2.

  - `AttrAuthoritiesCertValues`. It contains values of the Attribute Authorities certificates that have been used to validate the attribute certificate when present in the signature. It may also contain CA certificates in the certification path of the Attribute Authorities certificates. It may also contain certificates from entities issuing signed assertions on the signer, and CA certificates within their certification paths. Details on this property can be found in clause 6.4.3.

  - `AttributeRevocationValues`. It contains the full set of revocation data that have been used to validate the attribute certificate when present in the signature. It may also contain the full set of revocation data that have been used to validate signed assertions on the signer. Details on this property can be found in clause 6.4.4.

  Other contain references to validation data:

  - `CompleteCertificateRefs` and `xadesenv111:CompleteCertificateRefs`. References to the CA certificates used to validate the signature. Details on these properties can be found in normative annex A clause A.1.1.

  - `CompleteRevocationRefs`. It contains references to the full set of revocation information used for the validation of the electronic signature. Details on this property can be found in normative annex A clause A.1.2.

  - `AttributeCertficateRefs` and `xadesenv111:AttributeCertficateRefs`. They contain references to the full set of Attribute Authorities certificates that have been used to validate the attribute certificate. It may also contain references to CA certificates in the certification path of the Attribute Authorities certificates. It may also contain references to certificates from entities issuing signed assertions on the signer, and references to CA certificates within their certification paths. Details on these properties can be found in normative annex A clause A.1.3.

- `AttributeRevocationRefs`. It contains references to the full set of references to the revocation data that have been used in the validation of the attribute certificate(s) present in the signature. It may also contain the full set of references to revocation data that have been used to validate signed assertions on the signer. Details on this property can be found in normative annex A clause A.1.4.

- Time-stamp token container properties. The present document defines an abstract and two concrete XML types (`GenericTimeStampType`, `XAdESTimeStampType` and `OtherTimeStampType`) for allowing the inclusion of time-stamp tokens in a XMLDSIG signature. These types are defined in clause 6.1.4. The present document uses `XAdESTimeStampType` for defining several time-stamp token container properties, with capability for containing one or more time-stamp tokens covering different parts of the signature (common elements defined in XMLDSIG, validation data, qualifying properties, etc.). It also specifies a type for encapsulating validation data used for validating time-stamp tokens. Below follows the list:

  - `SignatureTimeStamp`. Each time-stamp token within this property covers the digital signature value element. Details on this property can be found in clause 6.3.

  - `AllDataObjectsTimeStamp`. Each time-stamp token within this property covers all the signed data objects. Details on this property can be found in clause 6.2.8.1.

  - `IndividualDataObjectsTimeStamp`. Each time-stamp token within this property covers selected signed data objects. Details on this property can be found in clause 6.2.8.2.

  - `xadesv141:ArchiveTimeStamp`. Each time-stamp token within this property covers signature and other properties required for achieving an Archival Electronic Signature and for providing long-term validity. Details on this property can be found in clause 6.5.2.

  - `SigAndRefsTimeStamp`. Each time-stamp token within this property covers the signature and references to validation data. Details on this property can be found in normative annex A clause A.1.5.1.

  - `RefsOnlyTimeStamp`. Each time-stamp token within this property covers only references to validation data. Details on this property can be found in annex A clause A.1.5.2.

  - `ArchiveTimeStamp`. This element is deprecated by `xadesv141:ArchiveTimeStamp` and kept within the normative annex A for managing legacy XAdES signatures. Details on this property can be found in normative annex A clause A.2.1.

  - `xadesv141:TimeStampValidationData`. This property encapsulates validation data for a time-stamp token embedded in one of the XAdES time-stamp token containers. Details on this property can be found in clause 6.5.1.

- Other properties. The present document defines a number of additional properties that can be useful in a wide range of environments, namely:

  - `SigningTime`. This property contains the time at which the signer claims to have performed the signing process. Details on this property can be found in clause 6.2.1.

  - `DataObjectFormat`. This property identifies the format of a signed data object (when electronic signatures are not exchanged in a restricted context) to enable the presentation to the verifier or the use by the verifier (text, sound or video) in exactly the same way as intended by the signer. Details on this property can be found in clause 6.2.4.

  - `CommitmentTypeIndication`. This property identifies the commitment undertaken by the signer in signing (a) signed data object(s) in the context of the selected signature policy (when an explicit commitment is being used). This will be required where a signature policy specifies more than a single commitment type, each of which might have different legal interpretations of the intent of the signature (e.g. proof of origin, proof of receipt, proof of creation, etc.). Details on this property can be found in clause 6.2.3.

  - `SignatureProductionPlace`. This property contains the indication of the purported place where the signer claims to have produced the signature. Details on this property can be found in clause 6.2.5.

- `SignerRole` and `xadesenv111:SignerRole`. These properties allow incorporating signer attributes (e.g. signer roles). `SignerRole`, specified in clause 6.2.6.1 allows to incorporate claimed or certified attributes using X509 based attribute certificates, issued by an Attribute Authority. Property `xadesenv111:SignerRole`, which is specified in clause 6.2.6.2 allows incorporating the aforementioned types of attributes and also signed attributes in XML syntax (for instance signed SAML assertions).

- `CounterSignature`. This property contains signature(s) produced on the signature. Details on this property can be found in clause 6.2.7.2.

- `xadesenv111:SignaturePolicyStore`. This property allows incorporating to the signature the signature policy document or a pointer to a local storage where this document may be stored for managing the signature in the long term. Details on this property may be found in clause 6.2.10.

- `xadesenv111:RenewedDigests`. This property contains digest values of detached data objects indirectly signed by using signed `ds:Manifest` elements. It allows to counter a threat resulting from the combination of the break of some of the digest algorithms used within the aforementioned signed `ds:Manifest` and the substitution of some detached data objects. Clause 6.5.3 specifies this property. Clause D1.14 provides the rationale for this property.

The aforementioned properties may be combined to generate different electronic signature forms. Some of them are defined in clause 4.1 of its normative part. Additional extended forms are defined in the normative annex B.

4.1 Electronic signature forms

The current clause specifies four forms of XML advanced electronic signatures, namely the **Basic Electronic Signature** (XAdES-BES), the **Explicit Policy based Electronic Signature** (XAdES-EPES), the **Electronic Signature with Time** (XAdES-T), and the **Archival Electronic Signature** (XAdES-A).

The normative annex B defines forms of XAdES signatures incorporating properties that encapsulate references to validation data and properties that encapsulate time-stamp tokens on the aforementioned references.

Readers are referred to RFC 6931 [13] where they may find additional URIs to those ones defined within XMLDSIG [2].

4.1.1 Basic electronic signature (XAdES-BES)

A Basic Electronic Signature (XAdES-BES) in accordance with the present document will build on a XMLDSIG by incorporating qualifying properties defined in the present document. They will be incorporated to XMLDSIG using one of the mechanisms specified in clause 5.3.

Some properties defined for building up this form will be covered by the signer's signature (descendant of `SignedProperties` element). Other properties will be not covered by the signer's signature (descendant of `UnsignedProperties` element).

In a XAdES-BES the signature value shall be computed in the usual way of XMLDSIG over the data object(s) to be signed and on the whole set of signed properties when present (`SignedProperties` element).

For this form it is mandatory to protect the signing certificate with the signature, in one of the two following ways:

- either incorporating one of the properties referencing the signing certificate (i.e. `SigningCertificate` or `xadesenv111:SigningCertificate` signed properties); or

- not incorporating none of the aforementioned signed properties but incorporating the signing certificate within the `ds:KeyInfo` element and signing at least the signing certificate.

A XAdES-BES signature shall, in consequence, contain at least one of the following elements with the specified contents:

- One of the `SigningCertificate` or the `xadesenv111:SigningCertificate` signed properties. The present property shall contain the reference and the digest value of the signing certificate. It may contain references and digests values of other certificates (that may form a chain up to the point of trust). In the case of ambiguities identifying the actual signer's certificate the applications should include one of these properties.

- The `ds:KeyInfo` element. If one of `SigningCertificate` or `xadesenv111:SigningCertificate` is incorporated to the signature, no restrictions apply to this element. If none of them are incorporated to the signature, then the following restrictions apply:

  - the `ds:KeyInfo` element shall include a `ds:X509Data` containing the signing certificate;

  - the `ds:KeyInfo` element also may contain other certificates forming a chain that MAY reach the point of trust;

  - the `ds:SignedInfo` element shall contain a `ds:Reference` element referencing `ds:KeyInfo`. That `ds:Reference` element shall be built in such a way that at least the signing certificate is actually signed.

NOTE 1: Readers are warned that signing the whole `ds:KeyInfo` locks the element: any addition of a certificate or validation data would make signature validation fail. Applications may, alternatively, use XPath transforms for signing at least the signing certificate, leaving the `ds:KeyInfo` element open for addition of new data after signing.

By incorporating one of these elements, XAdES-BES prevents the simple substitution of the signer's certificate (see clause D.1.2).

A XAdES-BES signature may also contain the following properties:

- the `SigningTime` signed property;

- the `DataObjectFormat` signed property;

- the `CommitmentTypeIndication` signed property;

- the `SignerRole` or `xadesenv111:SignerRole` signed property;

- the `SignatureProductionPlace` signed property;

- one or more `IndividualDataObjectsTimeStamp` or `AllDataObjectTimeStamp` signed properties;

- one or more `CounterSignature` unsigned properties.

Below follows the structure of the XAdES-BES built by direct incorporation of the qualifying information in the corresponding new XML elements to the XMLDSIG.

In the examples shown within these clauses "?" denotes zero or one occurrence; "+" denotes one or more occurrences; and "*" denotes zero or more occurrences. In the examples shown in these clauses, "`ds`" stands for the prefix corresponding to the namespace whose URI value is “http://www.w3.org/2000/09/xmldsig#”, and absence of prefix stands for those elements defined within the namespace whose URI value is ="http://uri.etsi.org/01903/v1.3.2#". Component [Ref. to signing certificate] denotes the choice between `SigningCertificate` and `xadesenv111:SigningCertificate`. [Signer Attrs] denotes the choice between `SignerRole` and `xadesenv111:SignerRole`.

Other XMLDSIG `ds:Object` elements with different contents may be added within the structure shown above to satisfy requirements other than the ones expressed in the present document. This also applies to the rest of the examples of structures of XAdES forms shown in this clause.

NOTE 2: The XAdES-BES is the minimum format for an electronic signature to be generated by the signer. On its own, it does not provide enough information for it to be verified in the longer term.

The XAdES-BES satisfies the legal requirements for electronic signatures as defined in the European Directive on electronic signatures. It provides basic authentication and integrity protection.

The XAdES-BES form is the XAdES instantiation of the AdES-BES form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause 7.1.

4.1.2 Explicit policy electronic signatures (XAdES-EPES)

An **Explicit Policy based Electronic Signature** (XAdES-EPES) form in accordance with the present document, extends the definition of an electronic signature to conform to the identified signature policy. A XAdES-EPES builds up on a XAdES-BES forms by incorporating the `SignaturePolicyIdentifier` element. This signed property indicates that a signature policy shall be used for signature validation. It provides means for explicitly identifying the signature policy. Other properties may be required by the mandated policy.

Clause 6.2.9 provides details on the specification of `SignaturePolicyIdentifier` property. Specification of the actual signature policies is outside the scope of the present document.

The structure of the XAdES-EPES (created **by direct incorporation** of the qualifying information to a XAdES-BES form) is illustrated below.

The XAdES-EPES form is the XAdES instantiation of the AdES-EPES form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause 7.2.

4.1.3 Electronic signature formats with validation data

Validation of an electronic signature in accordance with the present document requires additional data needed to validate the electronic signature. This additional data is called **validation data**; and includes:

- Public Key Certificates (PKCs) and Attributes Certificates (ACs);
- revocation status information for each PKC and AC (for instance Certificate Revocation Lists –CRLs- or OCSP responses);
- trusted time-stamps applied to the digital signature or a time-mark that shall be available in an audit log, evidencing that certain objects existed before a particular point in time;
- when appropriate, the details of a signature policy to be used to verify the electronic signature.

The present document defines properties able to encapsulate validation data. Clauses below summarize some signature forms that incorporate them and their most relevant characteristics

4.1.3.1 Electronic signature with time (XAdES-T)

XML **Advanced Electronic Signature with Time** (XAdES-T) builds on a XAdES-BES or a XAdES-EPES by incorporation of a trusted time associated to the signature. The trusted time may be provided by two different means:

- the `SignatureTimeStamp` as an unsigned property added to the electronic signature;
- a time mark of the electronic signature provided by a trusted service provider.

A time-mark provided by a Trusted Service would have similar effect to the `SignatureTimeStamp` property but in this case no property is added to the electronic signature as it is the responsibility of the TSP to provide evidence of a time mark when required to do so. The management of time marks is outside the scope of the present document.

EDITOR NOTE: Feedback is kindly requested on the suitability of including a referencing mechanism to a time-mark. We are especially interested in the opinion of stakeholders using a time-mark mechanism

Trusted time provides the initial steps towards providing long term validity. The XAdES-T trusted time indications shall be created before a certificate has been revoked or expired.

Below follows the structure of a XAdES-T form built on a XAdES-BES or a XAdES-EPES (if `SignaturePolicyIdentifier` signed property is present), by direct incorporation of a time-stamp token within the `SignatureTimeStamp` element. A XAdES-T form based on time-marks may exist without such an element.

```
                                XMLDISG
                                   |
<ds:Signature ID?>- - - - - - - - +- - - - +- - - +
  <ds:SignedInfo>                  |        |      |
    <ds:CanonicalizationMethod/>   |        |      |
    <ds:SignatureMethod/>          |        |      |
    (<ds:Reference URI? >          |        |      |
      (<ds:Transforms>)?           |        |      |
      <ds:DigestMethod/>           |        |      |
      <ds:DigestValue/>            |        |      |
    </ds:Reference>)+              |        |      |
  </ds:SignedInfo>                 |        |      |
  <ds:SignatureValue/>             |        |      |
  (<ds:KeyInfo>)? - - - - - - - - +        |      |
                                           |      |
  <ds:Object>                              |      |
                                           |      |
    <QualifyingProperties>                 |      |
                                           |      |
      <SignedProperties>?                  |      |
                                           |      |
        <SignedSignatureProperties>        |      |
          (SigningTime)?                   |      |
          ([Ref. to signing certificate])? |      |
          (SignaturePolicyIdentifier)?     |      |
          (SignatureProductionPlace)?      |      |
          ([Signer Attrs.])?               |      |
        </SignedSignatureProperties>       |      |
```

```
                                                |       |
        <SignedDataObjectProperties>            |       |
          (DataObjectFormat)*                   |       |
          (CommitmentTypeIndication)*           |       |
          (AllDataObjectsTimeStamp)*            |       |
          (IndividualDataObjectsTimeStamp)*     |       |
        </SignedDataObjectProperties>           |       |
                                                |       |
      </SignedProperties>                       |       |
                                                |       |
      <UnsignedProperties>                      |       |
                                                |       |
        <UnsignedSignatureProperties>           |       |
          (CounterSignature)*- - - - - - - - +   |       |
          (SignatureTimeStamp)+              |  |       |
        </UnsignedSignatureProperties>- - -+ |  |       |
                                           |  |  |       |
      </UnsignedProperties>                |  |  |       |
                                           |  |  |       |
    </QualifyingProperties>                |  |  |       |
                                           |  |  |       |
  </ds:Object>                             |  |  |       |
                                           |  |  |       |
</ds:Signature>- - - - - - - - - - - - - - +- - - +       |
                                           |  |  |       |
                                           |  |  |       |
                                  XAdES-BES(-EPES)  |    |
                                                    |    |
                                                 XAdES-T
```

The XAdES-T form is the XAdES instantiation of the AdES-T form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause 7.3.

NOTE : As a minimum, the signer will provide the XAdES-BES or when indicating that the signature conforms to an explicit signing policy the XAdES-EPES. To reduce the risk of repudiating signature creation, the trusted time indication needs to be as close as possible to the time the signature was created. The signer or a TSP could provide the XAdES-T. If the signer did not provide it, the verifier SHOULD create the XAdES-T on first receipt of an electronic signature, because the XAdES-T provides independent evidence of the existence of the signature prior to the trusted time indication.

4.1.3.2 Archival electronic signatures (XAdES-A)

Archival signatures in accordance with the present document incorporate `CertificateValues` unless the `ds:KeyInfo` element does contain the full set of certificates used to validate the electronic signature. They also incorporate `RevocationValues` unless the `ds:KeyInfo` element contains the revocation information that has to be shipped with the electronic signature. Archival signatures also incorporate one or more `xadesv141:ArchiveTimeStamp` unsigned properties. They may contain other properties. Each `xadesv141:ArchiveTimeStamp` element contains time-stamp tokens covering among other elements, those ones that contain validation data. These forms are used for archival of signatures. Successive archive time-stamps protect the whole material against vulnerable hashing algorithms or the breaking of the cryptographic material or algorithms and the expiration of the time-stamp token certificate.

Below follows the structure of a XAdES-A built on a XAdES-T by incorporation of at least one `xadesv141:ArchiveTimeStamp` element. In the figure below, the prefix "xadesv141" prefix corresponds to XML Namespace whose URI value is "http://uri.etsi.org/01903/v1.4.1#"

```
                                  XMLDISG
                                    |
<ds:Signature ID?>- - - - - - - - +- - - - - - - - - +
  <ds:SignedInfo>                 |                 |
    <ds:CanonicalizationMethod/>  |                 |
    <ds:SignatureMethod/>         |                 |
    (<ds:Reference URI? >         |                 |
      (<ds:Transforms/>)?         |                 |
      <ds:DigestMethod/>          |                 |
      <ds:DigestValue/>           |                 |
    </ds:Reference>)+             |                 |
  </ds:SignedInfo>                |                 |
```

The XAdES-A form is the XAdES instantiation of the AdES-A form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause 7.4.

5 General Syntax

The present clause defines the namespaces used in the XML schema definitions appearing in the present document. It also defines the types for the containers of the qualifying properties, and specifies the mechanisms for incorporating them to the electronic signature.

5.1 XML Namespaces for the present document

The present document uses the following URI namespaces:

- http://uri.etsi.org/01903/v1.3.2#

- http://uri.etsi.org/01903/v1.4.1#
- http://uri.etsi.org/19132/v1.1.1#
- http://www.w3.org/2000/09/xmldsig#
- http://www.w3.org/2001/XMLSchema

The table below shows the correspondence between the namespaces' URIs and the prefixes used throughout the present document:

Table 5.1.1: Namespaces and prefixes

XML Namespace URI	Prefix
http://uri.etsi.org/01903/v1.3.2#	xades
http://uri.etsi.org/01903/v1.4.1#	xadesv141
http://uri.etsi.org/19132/v1.1.1#	xadesenv111
http://www.w3.org/2000/09/xmldsig#	ds
http://www.w3.org/2001/XMLSchema	xsd

NOTE 1: The http://uri.etsi.org/01903/v1.3.2# URI was defined by ETSI TS 101 903 v1.3.2. Most of the XML elements and types used by XAdES signatures were defined in this namespace. Additionally, ETSI TS 101 903 v.1.4.1 defined http://uri.etsi.org/01903/v1.4.1# URI, where new types and elements were defined. Finally, the present document will also define new types and elements, which will be defined within the namespace whose URI is http://uri.etsi.org/19132/v1.1.1# .

NOTE 2: This specification is accompanied by three xml schema files, namely: (TO BE PROVIDED)

EDITOR NOTE: The xml schema will be provided with the final version in separated files.

5.2 The `QualifyingProperties` element

The `QualifyingProperties` element acts as a container element for all the qualifying information that is added to an XML signature. The element has the following structure.

```
<xsd:element name="QualifyingProperties" type="QualifyingPropertiesType"/>
<xsd:complexType name="QualifyingPropertiesType">
  <xsd:sequence>
    <xsd:element name="SignedProperties" type="SignedPropertiesType"
      minOccurs="0"/>
    <xsd:element name="UnsignedProperties"
      type="UnsignedPropertiesType"
      minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Target" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The qualifying properties are split into properties that are cryptographically bound to (i.e. signed by) the XML signature (`SignedProperties`), and properties that are not cryptographically bound to the XML signature (`UnsignedProperties`). The `SignedProperties` shall be covered by a `ds:Reference` element of the XML signature.

The mandatory `Target` attribute shall refer to the `Id` attribute of the corresponding `ds:Signature`. Its value shall be an URI with a bare-name XPointer fragment. When this element is enveloped by the XAdES signature, its not-fragment part shall be empty. Otherwise, its not-fragment part may not be empty.

The optional `Id` attribute can be used to make a reference to the `QualifyingProperties` container.

It is strongly recommended not to include empty `xades:SignedProperties` or empty `xades:UnsignedProperties` elements within the signature. Applications verifying XAdES signatures shall ignore empty `xades:SignedProperties` and empty `xades:UnsignedProperties` elements.

5.2.1 The SignedProperties element

The SignedProperties element contains a number of properties that are collectively signed by the XML signature.

Below follows the schema definition for SignedProperties element.

```
<xsd:element name="SignedProperties" type="SignedPropertiesType" />

<xsd:complexType name="SignedPropertiesType">
  <xsd:sequence>
    <xsd:element name="SignedSignatureProperties"
      type="SignedSignaturePropertiesType" minOccurs="0"/>
    <xsd:element name="SignedDataObjectProperties"
      type="SignedDataObjectPropertiesType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The SignedProperties element may contain properties that qualify the XML signature itself or the signer. If present, they are included as content of the SignedSignatureProperties element, specified in clause 5.2.3.

NOTE: If the ds:KeyInfo element is built according to what is specified in clause 4.1.1, it could happen that no signed signature property is required, and no SignedSignatureProperties element would be needed in the XAdES signature.

The SignedProperties element may also contain properties that qualify some of the signed data objects. These properties are included as content of the SignedDataObjectProperties element, specified in clause 5.2.4.

The optional Id attribute can be used to make a reference to the SignedProperties element.

5.2.2 The UnsignedProperties element

The UnsignedProperties element contains a number of properties that are not signed by the XML signature. Below follows the schema definition for this element.

```
<xsd:element name="UnsignedProperties" type="UnsignedPropertiesType" />

<xsd:complexType name="UnsignedPropertiesType">
  <xsd:sequence>
    <xsd:element name="UnsignedSignatureProperties"
      type="UnsignedSignaturePropertiesType" minOccurs="0"/>
    <xsd:element name="UnsignedDataObjectProperties"
      type="UnsignedDataObjectPropertiesType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The UnsignedProperties element may contain properties that qualify the XML signature itself or the signer. These properties are included as content of the UnsignedSignatureProperties element (see clause 5.2.5).

The UnsignedProperties element may also contain properties that qualify some of the signed data objects. These properties are included as content of the UnsignedDataObjectProperties element (see clause 5.2.6).

The optional Id attribute can be used to make a reference to the UnsignedProperties element.

5.2.3 The SignedSignatureProperties element

This element contains properties that qualify the XML signature that has been specified with the Target attribute of the QualifyingProperties container element.

```
<xsd:element name="SignedSignatureProperties"
  type="SignedSignaturePropertiesType" />

<xsd:complexType name="SignedSignaturePropertiesType">
  <xsd:sequence>
    <xsd:element name="SigningTime" type="xsd:dateTime"
     minOccurs="0"/>
    <xsd:element name="SigningCertificate" type="CertIDListType"
```

```
      minOccurs="0"/>
    <xsd:element name="SignaturePolicyIdentifer"
     type="SignaturePolicyIdentifierType" minOccurs="0"/>
    <xsd:element name="SignatureProductionPlace"
      type="SignatureProductionPlaceType"
      minOccurs="0"/>
    <xsd:element name="SignerRole" type="SignerRoleType"
     minOccurs="0"/>
    <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The optional `Id` attribute can be used to make a reference to the `SignedSignatureProperties` element.

The `xsd:any` element shall be used only for ensuring compatibility among different versions of XAdES. By using this element, applications may add any signed signature property defined in any other version of XAdES. It shall not contain elements whose types and contents are defined outside of a XAdES specification.

The qualifying property `SigningTime` is described in detail in clause 6.2.1, `SigningCertificate` in clause 6.2.2.1, `SignaturePolicyIdentifier` in clause 6.2.9, `SignatureProductionPlace` in clause 6.2.5, `SignerRole` in clause 6.2.6.1. Finally, the present specification defines `xadesenv111:SigningCertificate` in clause 6.2.2.2, as an alternative to `SigingCertificate`, and `xadesenv111:SignerRole` in clause 6.2.7, as an alternative to `SignerRole`. These two last qualifying properties are defined within the namespace whose URI is http://uri.etsi.org/19132/v1.1.1#.

5.2.4 The `SignedDataObjectProperties` element

This element contains properties that qualify some of the signed data objects.


```
<xsd:element name="SignedDataObjectProperties"
  type="SignedDataObjectPropertiesType"/>

<xsd:complexType name="SignedDataObjectPropertiesType">
  <xsd:sequence>
    <xsd:element name="DataObjectFormat" type="DataObjectFormatType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="CommitmentTypeIndication"
      type="CommitmentTypeIndicationType" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="AllDataObjectsTimeStamp" type="XAdESTimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="IndividualDataObjectsTimeStamp"
      type="XAdESTimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>

  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The optional `Id` attribute can be used to make a reference to the `SignedDataObjectProperties` element.

The `xsd:any` element shall be used only for ensuring compatibility among different versions of XAdES. By using this element, applications may add any signed data object property defined in any other version of XAdES. It shall not contain elements whose types and contents are defined outside of a XAdES specification.

The qualifying property `AllDataObjectsTimeStamp` is described in detail in clause 6.2.8.1, `IndividualDataObjectsTimeStamp` in clause 6.2.8.2, `DataObjectFormat` in clause 6.2.4, and `CommitmentTypeIndication` in clause 6.2.3.

5.2.5 The `UnsignedSignatureProperties` element

This element contains properties that qualify the XML signature that has been specified with the `Target` attribute of the `QualifyingProperties` container element. The content of this element is not covered by the XML signature.

```
<xsd:element name="UnsignedSignatureProperties"
  type="UnsignedSignaturePropertiesType"/>
```

```
<xsd:complexType name="UnsignedSignaturePropertiesType">
  <xsd:choice maxOccurs="unbounded">
    <xsd:element name="CounterSignature" type="CounterSignatureType" />
    <xsd:element name="SignatureTimeStamp" type="XAdESTimeStampType/>
    <xsd:element name="CompleteCertificateRefs"
      type="CompleteCertificateRefsType"/>
    <xsd:element name="CompleteRevocationRefs"
      type="CompleteRevocationRefsType"/>
    <xsd:element name="AttributeCertificateRefs"
      type="CompleteCertificateRefsType"/>
    <xsd:element name="AttributeRevocationRefs"
      type="CompleteRevocationRefsType"/>
    <xsd:element name="SigAndRefsTimeStamp" type="XAdESTimeStampType"/>
    <xsd:element name="RefsOnlyTimeStamp" type="XAdESTimeStampType"/>
    <xsd:element name="CertificateValues" type="CertificateValuesType"/>
    <xsd:element name="RevocationValues" type="RevocationValuesType"/>
    <xsd:element name="AttrAuthoritiesCertValues"
      type="CertificateValuesType"/>
    <xsd:element name="AttributeRevocationValues"
      type="RevocationValuesType"/>
    <xsd:element name="ArchiveTimeStamp" type="XAdESTimeStampType"/>
    <xsd:any namespace="##other" />
  </xsd:choice>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The optional `Id` attribute can be used to make a reference to the `UnsignedSignatureProperties` element.

The `xsd:any` element shall be used only for ensuring compatibility among different versions of XAdES. By using this element, applications MAY add any unsigned signature property defined in any other version of XAdES. It shall not contain elements whose types and contents are defined outside of a XAdES specification..

Some of these qualifying properties are defined within the normative body of the present document. These properties are: `CounterSignature`, which is defined in clause 6.2.7.2, `SignatureTimeStamp` in clause 6.3, `CertificateValues` in clause 6.4.1, `RevocationValues` in clause 6.4.2, `AttrAuthoritiesCertValues` in clause 6.4.3, and `AttributeRevocationValues` in clause 6.4.5.

Some of the qualifying properties are defined within the normative Annex A. These properties are: `CompleteCertificateRefs`, which is defined in clause A.1.1.1 of the aforementioned Annex, `CompleteRevocationRefs` in clause A.1.2, `AttributeCertificateRefs` in clause A1.3.1, `AttributeRevocationRefs` in clause A1.4, `SigAndRefsTimeStamp` in clause A.1.5.1, `RefsOnlyTimeStamp` in clause A.1.5.2.

Finally, the present document also defines unsigned properties not included in the namespace whose URI is http://uri.etsi.org/01903/v1.3.2#. Two of them are defined within the XML namespace whose URI is http://uri.etsi.org/01903/v1.4.1#, namely: `xadesv141:ArchiveTimeStamp` specified in clause 6.5.1, and `xadesv141:TimeStampValidationData` specified in clause 6.6. Four of them are defined within the XML XML namespace whose URI is http://uri.etsi.org/19132/v1.1.1#, namely: `xadesenv111:SignaturePolicyStore` in clause 6.2.10, `xadesenv111:RenewedDigests` in clause 6.5.3, `xadesenv111:CompleteCertificateRefs` in clause A1.1.1.2, and `xadesenv111:AttributeCertificateRefs` in clause A1.1.3.2.

5.2.6 The `UnsignedDataObjectProperties` element

This element contains properties that qualify some of the signed data objects. The signature generated by the signer does not cover the content of this element.

```
<xsd:element name="UnsignedDataObjectProperties"
  type="UnsignedDataObjectPropertiesType" />

<xsd:complexType name="UnsignedDataObjectPropertiesType">
  <xsd:sequence>
    <xsd:element name="UnsignedDataObjectProperty" type="AnyType"
       maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The optional `Id` attribute can be used to make a reference to the `UnsignedDataObjectProperties` element.

The present document does not specify the usage of any unsigned property qualifying the signed data object. It, however, defines this element for the sake of completeness and to cope with potential future needs for inclusion of such kind of properties. The schema definition leaves open the definition of the contents of this type. The type `AnyType` is defined in clause 6.1.1.

5.3 Incorporating qualifying properties into an XML signature

The present document utilizes the `ds:Object` auxiliary element from XMLDSIG [2]. It shall be used to incorporate the qualifying properties into the XMLDSIG signature. In principle, two different means are provided for this incorporation:

- direct incorporation means that a `QualifyingProperties` element is put as a child of the `ds:Object`;
- indirect incorporation means that one or more `QualifyingPropertiesReference` elements appear as children of the `ds:Object`. Each one contains information about one `QualifyingProperties` element that is stored in a place different from the signature (see clause 6.3.2).

However, the following restrictions apply for using `ds:Object`, `QualifyingProperties` and `QualifyingPropertiesReference`:

- all instances of the `QualifyingProperties` and the `QualifyingPropertiesReference` elements shall occur within a single `ds:Object` element;
- at most one instance of the `QualifyingProperties` element may occur within this `ds:Object` element;
- all signed properties shall occur within a single `QualifyingProperties` element. This element can either be a child of this `ds:Object` element (direct incorporation), or it can be referenced by a `QualifyingPropertiesReference` element. See clause 5.3.1 for information how to sign properties;
- zero or more instances of the `QualifyingPropertiesReference` element may occur within this `ds:Object` element.

No restrictions apply to the relative position of the `ds:Object` containing the `QualifyingProperties` or `QualifyingPropertiesReference` with respect to others `ds:Object` elements present within `ds:Signature`.

It is out of the scope of the present document to specify the mechanisms required to guarantee the correct storage of the distributed `QualifyingProperties` elements (i.e. that the properties are stored by the entity that has to store them and that they are not undetectably modified)

5.3.1 Signing properties

As has already been stated, all the properties that should be protected by the signature have to be collected in a single instance of the `QualifyingProperties` element. Actually these properties are children of the `SignedProperties` child of this element.

In order to protect the properties with the signature, a `ds:Reference` element shall be added to the XML signature. This `ds:Reference` element shall be composed in such a way that it uses the `SignedProperties` element mentioned above as the input for computing its corresponding digest.

Additionally, applications claiming conformance with the present document shall include the `Type` attribute to this particular `ds:Reference` element, with its value set to:

- http://uri.etsi.org/01903#SignedProperties.

This value indicates that the data used for hash computation is a `SignedProperties` element and therefore helps a verifying application to detect the signed properties of a signature conforming to the present document.

5.3.2 The QualifyingPropertiesReference element

This element contains information about a QualifyingProperties element that is stored in a place different from the signature, for instance in another XML document.

```
<xsd:element name="QualifyingPropertiesReference"
  type="QualifyingPropertiesReferenceType"/>

<xsd:complexType name="QualifyingPropertiesReferenceType">
  <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The mandatory URI attribute contains a bare-name XPointer fragment and references an external QualifyingProperties element. Its not-fragment part identifies the enclosing document and its bare-name XPointer fragment identifies the aforementioned element.

The optional Id attribute can be used to make a reference to the QualifyingPropertiesReference element.

5.4 Managing canonicalization of XML nodesets

A number of qualifying properties specified in the present document incorporate optional means for identifying a canonicalization algorithm for computing the canonical form of a certain XML node set.

When dealing with legacy XAdES signatures, i.e., signatures created before the publication of the present document, applications claiming conformance to the present document shall assume that absence of the canonicalization indication means that the actual canonicalization algorithm is Canonical XML 1.0 without comments, identified by the URI:

http://www.w3.org/TR/2001/REC-xml-c14n-20010315

When generating new XAdES signatures, applications claiming conformance to the present document shall explicitly indicate the canonicalization algorithm in all the XAdES qualifying properties by including a container for canonicalization algorithm identifier.

When upgrading a legacy XAdES signature to an upper form by the generation and incorporation of a certain XAdES qualifying property whose XML Schema definition includes an optional identifier of a canonicalization algorithm, applications claiming conformance to the present document shall include the aforementioned canonicalization algorithm identifier within the qualifying property.

Readers are warned that Canonical XML 1.0 does not properly process the inheritance of attributes in the XML namespace (xml:id and xml:base) when canonicalizing document sub-trees. Canonical XML version 1.1 (whose version omitting comments is identified by the URI http://www.w3.org/2006/12/xml-c14n11), specifies a variant of the former canonicalization algorithm that properly addresses these issues.

Readers are also warned that Canonical XML 1.0 when applied to an XML sub-tree, includes the subree's ancestor context including all of the namespace declarations and attributes in the "xml:" namespace. The exclusive XML Canonicalization algorithm (whose version omitting comments is identified by the URI http://www.w3.org/2001/10/xml-exc-c14n#) completely excludes this ancestor context from the canonicalized sub-tree.

6 Qualifying properties syntax

This clause specifies some of the qualifying properties defined in the present document, as well as a number of auxiliary types.

Clause 6.1 defines the auxiliary types.

Clause 6.2 specifies the qualifying properties that may appear in XAdES-BES and XAdES-EPES electronic signatures forms as described in clause 4.

Clause 6.3 specifies the `SignatureTimeStamp` qualifying properties for XAdES-T electronic signatures forms as described in clause 4.

Clause 6.4 specifies the qualifying properties that can contain validation data values.

Clause 6.5 specifies qualifying properties for XAdES-A electronic signatures forms as described in clause 4.

6.1 Auxiliary syntax

The next sub-clauses below specify certain auxiliary XML structures, utilized in several cases throughout the present document.

6.1.1 The `AnyType` data type

The `AnyType` Schema data type has a content model that allows a sequence of arbitrary XML elements that (mixed with text) is of unrestricted length. It also allows for text content only. Additionally, an element of this data type can bear an unrestricted number of arbitrary attributes. It is used throughout the remaining parts of the present document wherever the content of an XML element has been left open.

```
<xsd:complexType name="AnyType" mixed="true">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:any namespace="##any" processContents="lax"/>
  </xsd:sequence>
  <xsd:anyAttribute namespace="##any"/>
</xsd:complexType>
```

6.1.2 The `ObjectIdentifierType` data type

The `ObjectIdentifierType` data type can be used to identify a particular data object.

It allows the specification of a unique and permanent identifier of an object. In addition, it may also contain, a textual description of the nature of the data object, and a number of references to documents where additional information about the nature of the data object can be found.

```
<xsd:complexType name="ObjectIdentifierType">
  <xsd:sequence>
    <xsd:element name="Identifier" type="IdentifierType"/>
    <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
    <xsd:element name="DocumentationReferences"
     type="DocumentationReferencesType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

The `Identifier` element contains a permanent identifier. Once the identifier is assigned, it can never be re-assigned again. It supports both the mechanism that is used to identify objects in ASN.1 and the mechanism that is usually used to identify objects in an XML environment:

- in an XML environment objects are typically identified by means of a Uniform Resource Identifier, URI. In this case, the content of `Identifier` consists of the identifying URI, and the optional `Qualifier` attribute does not appear;
- in ASN.1 an Object IDentifier (OID) is used to identify an object. To support an OID, the content of Identifier consists of an OID, either encoded as Uniform Resource Name (URN) or as Uniform Resource Identifier (URI). The optional `Qualifier` attribute can be used to provide a hint about the applied encoding (values "OIDAsURN" or "OIDAsURI").

Applications claiming conformance to the present document and needing to identify an object by an OID, should encode it as an URN as specified by the RFC 3061 [10], and set Qualifier attribute to "OIDAsURN" value.

- Should an OID and an URI exist identifying the same object, the present document encourages the use of the `URI` as explained in the first bullet above.

```
<xsd:complexType name="IdentifierType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:anyURI">
      <xsd:attribute name="Qualifier" type="QualifierType"
        use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="QualifierType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="OIDAsURI"/>
    <xsd:enumeration value="OIDAsURN"/>
  </xsd:restriction>
</xsd:simpleType>
```

The optional `Description` element contains an informal text describing the object identifier.

The optional `DocumentationReferences` element consists of an arbitrary number of references pointing to further explanatory documentation of the object identifier.

```
<xsd:complexType name="DocumentationReferencesType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element name="DocumentationReference" type="xsd:anyURI"/>
  </xsd:sequence>
</xsd:complexType>
```

6.1.3 The `EncapsulatedPKIDataType` data type

The `EncapsulatedPKIDataType` is used to incorporate non XML pieces of PKI data into an XML structure. Examples of such PKI data that are widely used at the time being include X.509 certificates and revocation lists, OCSP responses, attribute certificates and time-stamp tokens.


```
<xsd:complexType name="EncapsulatedPKIDataType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:base64Binary">
      <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
      <xsd:attribute name="Encoding" type="xsd:anyURI"
        use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

The content of this data type is the piece of PKI data, base-64 encoded as defined in [2].

The `Encoding` attribute is an URI identifying the encoding used in the original PKI data. So far, the following URIs have been identified:

- http://uri.etsi.org/01903/v1.2.2#DER for denoting that the original PKI data were ASN.1 data encoded in DER.
- http://uri.etsi.org/01903/v1.2.2#BER for denoting that the original PKI data were ASN.1 data encoded in BER.
- http://uri.etsi.org/01903/v1.2.2#CER for denoting that the original PKI data were ASN.1 data encoded in CER.
- http://uri.etsi.org/01903/v1.2.2#PER for denoting that the original PKI data were ASN.1 data encoded in PER.
- http://uri.etsi.org/01903/v1.2.2#XER for denoting that the original PKI data were ASN.1 data encoded in XER.

If the `Encoding` attribute is not present, then it is assumed that the PKI data is ASN.1 data encoded in DER.

NOTE: In some clauses of the present document, specific XAdES properties related to these data restrict the encoding options to only one certain type of the aforementioned PKI data.

The optional `ID` attribute can be used to make a reference to an element of this data type

6.1.4 Types for time-stamp tokens management

XAdES uses time-stamp tokens in a number of use cases. The present document defines:

- A XML schema definition of an abstract base type and two concrete derived types used as containers for time-stamp tokens.
- A number of properties of one of the aforementioned concrete types. Time-stamp tokens included in these properties will cover a specific set of elements and properties of XAdES signature forms and will satisfy, in this way, different requirements.

A time-stamp token is obtained by sending the digest value of the given data (message imprint henceforth) to the Time-Stamp Authority (TSA). The returned time-stamp token is a signed data that contains the digest value, the identity of the TSA, and the time of stamping. This proves that the given data existed *before* the time of stamping.

NOTE: readers should note that the term time-stamp token used throughout the present document does NOT refer to the TSA's response to a requesting client, but the token generated by the TSA, which is present within this response. In the case of RFC 3161 [8] protocol, the time-stamp token term is referring to the `timeStampToken` field within the `TimeStampResp` element (the TSA's response returned to the requesting client).

XAdES time-stamp tokens container properties contain time-stamp tokens computed on both, elements defined in XMLDSIG [2] and properties defined in the present document. The present document uses the term time-stamped data objects for indistinctly denoting any of them.

6.1.4.1 Time-stamp properties in XAdES

Below follows the list of the properties containing time-stamps that are defined by the present document:

- Properties that contain time-stamp tokens proving that some or all the data objects to be signed have been created before some time: `AllDataObjectsTimeStamp` and `IndividualDataObjectsTimeStamp`.
- `SignatureTimeStamp`: it is a container for a time-stamp token over the `SignatureValue` element to protect against repudiation in case of a key compromise.
- To provide for long term validity of an XML signature, the signature and validation data values are time-stamped. `xadesv141:ArchiveTimeStamp` and `xadesenv111:ArchiveTimeStamp` are defined for this purpose. More than one instance of these properties can be added as time goes on to the archived electronic signature
- Annex A specifies two properties that contain time-stamp tokens on properties defined in that annex.

6.1.4.2 The `GenericTimeStampType` data type

The abstract base container type for time-stamp tokens specified by the present document does have the following features:

- It may contain encapsulated RFC 3161 [8] time-stamp tokens as well as XML time-stamp tokens.
- It may contain more than one time-stamp token generated for the same XAdES data objects (each one issued by different TSAs, for instance).
- It provides means for managing time-stamp tokens computed on XAdES data objects (for instance XAdES properties) or time-stamp tokens computed on external data.
- It may use specific elements for identifying what is time-stamped and how to generate the input data for the computation of the digest value to be sent to the TSA. For certain XAdES data objects under certain circumstances this information may be implicit.

Below follows the schema definition for the data type.

```
<xsd:element name="Include" type="IncludeType"/>
```

```
<xsd:complexType name="IncludeType">
  <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="referencedData" type="xsd:boolean"
    use="optional"/>
</xsd:complexType>

<xsd:element name="ReferenceInfo" type="ReferenceInfoType"/>
<xsd:complexType name="ReferenceInfoType">
  <xsd:sequence>
    <xsd:element ref="ds:DigestMethod"/>
    <xsd:element ref="ds:DigestValue"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
  <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>

<xsd:complexType name="GenericTimeStampType" abstract="true">
  <xsd:sequence>
    <xsd:choice minOccurs="0">
      <xsd:element ref="Include" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ReferenceInfo" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:element ref="ds:CanonicalizationMethod" minOccurs="0"/>
    <xsd:choice maxOccurs="unbounded">
      <xsd:element name="EncapsulatedTimeStamp"
        type="EncapsulatedPKIDataType"/>
      <xsd:element name="XMLTimeStamp" type="AnyType"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The optional `ds:CanonicalizationMethod` element indicates the canonicalization algorithm used for canonicalizing XML node sets resulting after retrieving (and processing when required) the data objects covered by the time-stamp token(s). Clause 5.4 of the present document specifies the requirements that applications claiming conformance to the present document shall satisfy when dealing with this element.

The time-stamp token generated by the TSA can be either an ASN.1 data object (as defined in [8], use `EncapsulatedTimeStamp`), or it can be encoded as XML (use `XMLTimeStamp`).

Details on the different elements and supporting types are given in the clauses that define the two concrete types: `XAdESTimeStampType` and `OtherTimeStampType`.

6.1.4.3 The `XAdESTimeStampType` data type

This concrete derived type is provided for containing time-stamp tokens computed on data objects of XAdES signatures. Applications claiming alignment with the present document shall implement it because all the properties listed in clause 6.1.4.1 are elements of this type.

Below follows the schema definition for the data type.

```
<xsd:element name="XAdESTimeStamp" type="XAdESTimeStampType"/>

<xsd:complexType name="XAdESTimeStampType">
  <xsd:complexContent>
    <xsd:restriction base="GenericTimeStampType">
      <xsd:sequence>
        <xsd:element ref="Include" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element ref="ds:CanonicalizationMethod" minOccurs="0"/>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="EncapsulatedTimeStamp"
            type="EncapsulatedPKIDataType"/>
          <xsd:element name="XMLTimeStamp" type="AnyType"/>
        </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

This type provides two mechanisms for identifying data objects that are covered by the time-stamp token present in the container, and for specifying how to compute the time-stamp token's message imprint:

- Explicit. This mechanism uses the `Include` element for referencing specific data objects and for indicating their contribution to the input of the message imprint's computation.

- Implicit. For certain time-stamp container properties under certain circumstances, applications do not require any additional indication for knowing that certain data objects are covered by the time-stamp tokens and how they contribute to the input of the message imprint's computation. The present document specifies, in the clauses defining such properties (clauses 6.2.8.1, 6.3, 6.5), how applications shall act in these cases without explicit indications.

Clause 6.1.4.3.1 shows the principles that govern the explicit indication mechanism.

6.1.4.3.1 Include mechanism

`Include` elements explicitly identify data objects that are time-stamped. Their order of appearance indicates how the data objects contribute in the generation of the input to the time-stamp's message imprint computation.

The `IndividualDataObjectsTimeStamp` time-stamp token container property uses this mechanism. Each `Include` element shall contain an URI referencing one of the `ds:Reference` elements in the XAdES signature.

Two unsigned properties specified in Annex A clauses A.1.5.1 and A.1.5.2 also use this mechanism under certain circumstances.

The `URI` attribute in `Include` element identifies one time-stamped data object. Its value shall follow the rules indicated below:

- It shall have an empty not-fragment part and a bare-name XPointer fragment when the `Include` and the time-stamped data object are in the same document.

- It shall have a not empty not-fragment part and a bare-name XPointer fragment when the `Include` and the time-stamped data object are not in the same document.

- When not empty, its not-fragment part shall be equal to:

  - the not-fragment part of the `Target` attribute of the `QualifyingProperties` enclosing the `Include` element if the time-stamped data object is enveloped by the XAdES signature; or

  - the not-fragment part of the `URI` attribute of the `QualifyingPropertiesReference` element referencing the `QualifyingProperties` element enveloping the time-stamped data object if this `QualifyingProperties` element is not enveloped by the XAdES signature.

Applications aligned with the present document shall parse the retrieved resource, and then process the bare-name XPointer as explained below to get a XPath node-set suitable for being processed according to the selected canonicalization algorithm. For processing the bare-name XPointer, applications shall use as XPointer evaluation context the root node of the XML document that contains the element referenced by the not-fragment part of URI. Applications shall derive an XPath node-set from the resultant location-set as indicated below:

1) Replace the element node E retrieved by the bare-name XPointer with E plus all descendants of E (text, comments, PIs, elements) and all namespace and attribute nodes of E and its descendant elements.

2) Delete all the comment nodes.

In time-stamps that cover `ds:Reference` elements, the attribute `referencedData` may be present. If present with value set to `"true"`, the time-stamp shall be computed on the result of processing the corresponding `ds:Reference` element according to the XMLDSIG processing model. If the attribute is not present or is present with value `"false"`, the time-stamp shall be computed on the `ds:Reference` element itself. When appearing in a time-stamp container property, each `Include` element shall be processed in order as detailed below:

1) Retrieve the data object referenced in the `URI` attribute following the referencing mechanism indicated above.

2) If the retrieved data is a `ds:Reference` element and the `referencedData` attribute is set to the value `"true"`, take the result of processing the retrieved `ds:Reference` element according to the reference processing model of XMLDSIG; otherwise take the `ds:Reference` element itself.

3) If the resulting data is an XML node set, canonicalize it as specified in clause 5.4.

4) Concatenate the resulting octets to those resulting from previous processing as indicated in the corresponding time-stamp container property.

6.1.4.4 The `OtherTimeStampType` data type

This concrete derived type is provided for containing time-stamp tokens computed on a collection of data objects that are not present in the XAdES signature.

Below follows the schema definition for the data type.

```
<xsd:element name="OtherTimeStamp" type="OtherTimeStampType"/>

<xsd:complexType name="OtherTimeStampType">
    <xsd:complexContent>
        <xsd:restriction base="GenericTimeStampType">
            <xsd:sequence>
                <xsd:element ref="ReferenceInfo"  maxOccurs="unbounded"/>
                <xsd:element ref="ds:CanonicalizationMethod" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="EncapsulatedTimeStamp"
                type="EncapsulatedPKIDataType"/>
                    <xsd:element name="XMLTimeStamp" type="AnyType"/>
            </xsd:choice>
            </xsd:sequence>
            <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
        </xsd:restriction>
    </xsd:complexContent>
</xsd:complexType>
```

Each `ReferenceInfo` element contains the digest of one external data object. Attribute `URI` identifies the data object. As in XMLDSIG, if it is omitted, the application is expected to know the identity of the referenced object. Attribute `Id` permits this element to be referenced from elsewhere. Element `ds:DigestMethod` identifies the digest algorithm applied to the external data object. Element `ds:DigestValue` contains the base-64 encoded value of the digest of the referenced data object.

Attribute `Id` and elements `ds:CannonicalizationMethod`, `EncapsulatedTimeStamp` and `XMLTimeStamp` will be used exactly as in `XAdESTimeStampType`.

For this type the actual input to the computation of the message imprint that will be sent to the TSA is the concatenation of the present `ReferenceInfo` elements, canonicalized as specified in clause 5.4.

The implementation of such a type is not mandatory for applications that claim conformance to the present document, as it does not define any property of this type

6.2 Properties for XAdES-BES and XAdES-EPES forms

This clause describes in detail the qualifying properties that can appear in XAdES-BES and XAdES-EPES forms as described in clauses 4.1.1 and 4.1.2.

6.2.1 The `SigningTime` element

The `SigningTime` property is an optional signed property that qualifies the whole signature. There shall be at most one occurrence of this property in the signature.

The `SigningTime` property specifies the time at which the signer (purportedly) performed the signing process.

Below follows the Schema definition for this element.

```
<xsd:element name="SigningTime" type="xsd:dateTime"/>
```

6.2.2 References to the signing certificate

Sub-clauses below specify two signed properties qualifying the signature used as containers of a reference and the digest value of the signing certificate (and optionally to other certificates in its certification path), namely: `SigningCertificate` and `xadesenv111:SigningCertificate`.

Only one of these two properties may be incorporated to a XAdES signature: if one of these properties is incorporated, then the other one shall not be incorporated.

EDITOR NOTE: feedback from stakeholders is requested on the solution proposed: define the new `xadesenv111:SigningCertificate` for acknowledging deprecation of `ds:X509IssuerSerial`, but keep `xades:SigningCertificate` for keeping backwards compatibility.

6.2.2.1 The `SigningCertificate` element

The `SigningCertificate` property is a signed property that qualifies the signature. At most one `SigningCertificate` element may be present in the signature.

The `SigningCertificate` property contains references to certificates and digest values computed on their DER-encodings.

Below follows the Schema definition.

```
<xsd:element name="SigningCertificate" type="CertIDListType"/>

<xsd:complexType name="CertIDListType">
    <xsd:sequence>
        <xsd:element name="Cert" type="CertIDType"
       maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CertIDType">
    <xsd:sequence>
        <xsd:element name="CertDigest" type="DigestAlgAndValueType"/>
        <xsd:element name="IssuerSerial" type="ds:X509IssuerSerialType"/>
    </xsd:sequence>
    <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>

<xsd:complexType name="DigestAlgAndValueType">
    <xsd:sequence>
        <xsd:element ref="ds:DigestMethod"/>
        <xsd:element ref="ds:DigestValue"/>
    </xsd:sequence>
</xsd:complexType>
```

The `SigningCertificate` element contains the aforementioned sequence of certificate identifiers and digests computed on the certificates (`Cert` elements).

The element `IssuerSerial` contains the identifier of the referenced certificate. Should the `ds:X509IssuerSerial` element appear in the signature to denote the same certificate, its value shall be consistent with the corresponding `IssuerSerial` element.

The element `CertDigest` contains the digest of the referenced certificate. It contains two elements: `ds:DigestMethod` indicates the digest algorithm and `ds:DigestValue` contains the base-64 encoded value of the digest computed on the DER-encoded certificate.

The optional `URI` attribute provides an indication of where the referenced certificate may be found. It is intended that this attribute is be used as a hint, as implementations may have alternative ways for retrieving the referenced certificate if it is not found at the referenced place.

EDITOR NOTE: clarification of the usage of this attribute. If the certificate is not found anymore at the place referenced by the URI, implementations may still be able to retrieve it.

The certificate used to verify the signature shall be present in this property. Other certificates may also be present in the sequence, which may include all the certificates up to the point of trust.

6.2.2.2 The xadesenv111:SigningCertificate element

The xadesenv111:SigningCertificate property is a signed property that qualifies the signature. At most one xadesenv111:SigningCertificate element may be present in the signature.

The xadesenv111:SigningCertificate property contains references to certificates and digest values computed on their DER-encodings.

Below follows the Schema definition.

```
<!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->

<xsd:element name="SigningCertificate" type="xadesenv111:CertIDListType"/>

<xsd:complexType name="CertIDListType">
    <xsd:sequence>
        <xsd:element name="Cert" type="xadesenv111:CertIDType" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CertIDType">
    <xsd:sequence>
        <xsd:element name="CertDigest" type="xades:DigestAlgAndValueType"/>
        <xsd:element name="IssuerSerial" type="xadesenv111:IssuerSerialType"/>
    </xsd:sequence>
    <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>

<xsd:complexType name="IssuerSerialType">
  <xsd:sequence>
    <xsd:element name="X509IssuerName" type="xsd:string"/>
    <xsd:element name="X509SerialNumber" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```


The semantics of the elements and types defined above are identical to the elements and types defined in clause 6.2.2.1, with the only exception of the new element xadesenv111:IssuerSerial element, of type xadesenv111:IssuerSerialType. Child xadesenv111:X509IssuerName has the same syntax and semantics than ds:X509IssuerName. The content of child xadesenv111:X509SerialNumber shall be a string containing the textual representation of the serial number field of the referenced certificate in base 10 without leading zeroes.

NOTE: The reason for defining the new xadesenv111:SigningCertificate signed property is that XMLDSig in its version 1.1 [2], has deprecated the usage of ds:X509IssuerSerial element. The rationale for its deprecation is that some XML Schema validation tools do not deal with integer values that have more than 18 decimal digits. It is not uncommon that the randomly generated serial numbers need more than 18 digits.

EDITOR NOTE: The ds:X509IssuerSerial element is a mandatory element of SigningCertificate element for keeping semantic alignment with CAdES and because it provides readable information to human beings on the referenced certificate. These are the reasons why the xadesenv111:Issuerserial element is specified as mandatory in the new xadesenv111:CertIDType. Feedback is kindly requested to stakeholders on the suitability of defining the new xadesenv111:SigningCertificate and on its specification itself.

6.2.3 The CommitmentTypeIndication element

The CommitmentTypeIndication property is an optional signed property that qualifies signed data object(s). A XAdES signature aligned with the present document may contain more than one CommitmentTypeIndication elements.

The CommitmentTypeIndication property contains an explicit indication of the type of the commitment made by the signatory when signing a certain data object.

Below follows the schema definition for this element.

```
<xsd:element name="CommitmentTypeIndication" type="CommitmentTypeIndicationType"/>

<xsd:complexType name="CommitmentTypeIndicationType">
    <xsd:sequence>
        <xsd:element name="CommitmentTypeId"
       type="ObjectIdentifierType"/>
        <xsd:choice>
            <xsd:element name="ObjectReference" type="xsd:anyURI"
         maxOccurs="unbounded"/>
            <xsd:element name="AllSignedDataObjects"/>
        </xsd:choice>
        <xsd:element name="CommitmentTypeQualifiers"
      type="CommitmentTypeQualifiersListType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CommitmentTypeQualifiersListType">
    <xsd:sequence>
        <xsd:element name="CommitmentTypeQualifier"
      type="AnyType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

The `CommitmentTypeId` element univocally identifies the type of commitment made by the signer. Below follows a list of commitment types and their corresponding URIs:

- **Proof of origin** indicates that the signer recognizes to have created, approved and sent the signed data object. The URI for this commitment is http://uri.etsi.org/01903/v1.2.2#ProofOfOrigin.
- **Proof of receipt** indicates that signer recognizes to have received the content of the signed data object. The URI for this commitment is http://uri.etsi.org/01903/v1.2.2#ProofOfReceipt.
- **Proof of delivery** indicates that the TSP providing that indication has delivered a signed data object in a local store accessible to the recipient of the signed data object. The URI for this commitment is http://uri.etsi.org/01903/v1.2.2#ProofOfDelivery.
- **Proof of sender** indicates that the entity providing that indication has sent the signed data object (but not necessarily created it). The URI for this commitment is http://uri.etsi.org/01903/v1.2.2#ProofOfSender.
- **Proof of approval** indicates that the signer has approved the content of the signed data object. The URI for this commitment is http://uri.etsi.org/01903/v1.2.2#ProofOfApproval.
- **Proof of creation** indicates that the signer has created the signed data object (but not necessarily approved, nor sent it). The URI for this commitment is http://uri.etsi.org/01903/v1.2.2#ProofOfCreation

One `ObjectReference` element refers to one `ds:Reference` element of the `ds:SignedInfo` or a signed `ds:Manifest`, corresponding with one data object qualified by this property. If some but not all the signed data objects share the same commitment, one `ObjectReference` element shall appear for each one of them. However, if all the signed data objects share the same commitment, the `AllSignedDataObjects` empty element shall be present.

The `CommitmentTypeQualifiers` element provides means to include additional qualifying information on the commitment made by the signatory.

6.2.4 The `DataObjectFormat` element

The `DataObjectFormat` property is an optional signed property that qualifies one specific signed data object. In consequence, a XAdES signature aligned with the present document may contain more than one `DataObjectFormat` elements, each one qualifying one signed data object.

The `DataObjectFormat` element provides information that describes the format of the signed data object. This element should be present when the signed data is to be presented to human users on validation and the presentation format is not implicit within the data that has been signed.

Below follows the schema definition for this element.

```
<xsd:element name="DataObjectFormat" type="DataObjectFormatType"/>

<xsd:complexType name="DataObjectFormatType">
    <xsd:sequence>
        <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
        <xsd:element name="ObjectIdentifier" type="ObjectIdentifierType"
      minOccurs="0"/>
        <xsd:element name="MimeType" type="xsd:string" minOccurs="0"/>
        <xsd:element name="Encoding" type="xsd:anyURI" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ObjectReference" type="xsd:anyURI"
   use="required"/>
</xsd:complexType>
```

The mandatory `xades:ObjectReference` attribute shall reference the `ds:Reference` child of the `ds:SignedInfo` or a signed `ds:Manifest` element referencing the signed data object qualified by this property.

This element can convey:

- textual information related to the signed data object in element `Description`;
- an identifier indicating the type of the signed data object in element `ObjectIdentifier`;
- an indication of the MIME type of the signed data object in element `MimeType`;
- an indication of the encoding format of the signed data object in element `Encoding`.

At least one element of `Description`, `ObjectIdentifier` and `MimeType` shall be present within the property.

If the `DataObjectFormat` property references a `ds:Reference` that in turn references a `ds:Object` within the XAdES signature, and if this `ds:Object` element has the `MimeType` or(and) the `Encoding` attribute(s), then `DataObjectFormat`'s children `MimeType` and `Encoding` shall have exactly the same values when present.

6.2.5 The `SignatureProductionPlace` element

The `SignatureProductionPlace` property is an optional signed property that qualifies the signer. There shall be at most one occurrence of this property in the signature.

The `SignatureProductionPlace` element specifies an address associated with the signer at a particular geographical (e.g. city) location.

Below follows the schema definition for this element.

```
<xsd:element name="SignatureProductionPlace" type="SignatureProductionPlaceType"/>

<xsd:complexType name="SignatureProductionPlaceType">
    <xsd:sequence>
        <xsd:element name="City" type="xsd:string" minOccurs="0"/>
        <xsd:element name="StateOrProvince" type="xsd:string"
       minOccurs="0"/>
        <xsd:element name="PostalCode" type="xsd:string" minOccurs="0"/>
        <xsd:element name="CountryName" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

6.2.6 Elements for incorporating signer attributes

XAdES signatures allow the incorporation of optional signed properties encapsulating signer attributes (e.g. role). The present document differentiates two types of attributes:

- attributes claimed by the signer;
- attributes certified by a trusted authority.

Sub-clauses below specify two ways of incorporating signer attributes in a XAdES signature.

6.2.6.1 The SignerRole element

The SignerRole property is an optional unsigned property that qualifies the signer. There shall be at most one occurence of this property within a XAdES signature.

Below follows the Schema definition for this element:

```
<xsd:element name="SignerRole" type="SignerRoleType"/>

<xsd:complexType name="SignerRoleType">
  <xsd:sequence>
    <xsd:element name="ClaimedRoles" type="ClaimedRolesListType"
      minOccurs="0"/>
    <xsd:element name="CertifiedRoles" type="CertifiedRolesListType"
      minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ClaimedRolesListType">
  <xsd:sequence>
    <xsd:element name="ClaimedRole" type="AnyType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CertifiedRolesListType">
  <xsd:sequence>
    <xsd:element name="CertifiedRole" type="EncapsulatedPKIDataType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

This property contains a sequence of roles that the signer can play (element SignerRole). At least one of the two elements ClaimedRoles or CertifiedRoles shall be present.

The ClaimedRoles element contains a sequence of roles claimed by the signer but not certified. Additional contents types may be defined on a domain application basis and be part of this element. The namespaces given to the corresponding XML schemas will allow their unambiguous identification in the case these attributes are expressed in XML syntax (e.g. SAML assertions of different versions).

The CertifiedRoles element contains the base-64 encoding of one or more DER-encoded attribute certificates as specified within ITU-T X.509 [5], for the signer.

6.2.6.2 The xadesenv111:SignerRole element

The xadesenv111:SignerRole property is an optional unsigned property that qualifies the signer. There shall be at most one occurence of this property within a XAdES signature.

Below follows the Schema definition for this element. Note that the elements are defined within the xadesenv111 namespace, whose URI is http://uri.etsi.org/19132/v1.1.1#.

```
<!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->

<xsd:element name="SignerRole" type="xadesenv111:SignerRoleType"/>

<xsd:complexType name="SignerRoleType">
    <xsd:sequence>
        <xsd:element name="ClaimedRoles" type="xades:ClaimedRolesListType" minOccurs="0"/>
        <xsd:element name="CertifiedRoles" type="xadesenv111:CertifiedRolesListType" minOccurs="0"/>
        <xsd:element name="SignedAssertions" type="xadesenv111:SignedAssertionsListType"
minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

</xsd:complexType>
<xsd:complexType name="CertifiedRolesListType">
    <xsd:sequence>
        <xsd:element name="CertifiedRole" type="xadesenv111:CertifiedRoleType"
maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="SignedAssertionsListType">
    <xsd:sequence>
        <xsd:element name="SignedAssertion" type="xades:AnyType" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CertifiedRoleType">
    <xsd:choice>
        <xsd:element name="x509AttributeCertificate" type="xades:EncapsulatedPKIDataType"/>
        <xsd:element name="otherAttributeCertificate" type="xades:AnyType"/>
    </xsd:choice>
</xsd:complexType>
```

The `ClaimedRoles` element has the same semantics and syntax as in `xades:SignerRole`.

The `CertifiedRoles` element may contain:

- the base-64 encoding of one or more DER-encoded X509 attribute certificates for the signer within the `x509AttributeCertificate` element or
- attribute certificates (issued, in consequence, by Attribute Authorities) in different syntax than the one used for X509 attribute certificates, within the `otherAttributeCertificate` element. The definition of specific `otherAttributeCertificates` is outside of the scope of the present document.

The `SignedAssertions` element contains signed assertions (e.g. signed SAML assertions). These assertions are signed by entities that do not satisfy all the requirements specified for being considered Attribute Authorities. The definition of specific `signerAssertions` is outside of the scope of the present document.

NOTE 1: EN 319 411-4 [i.1], annex A specifies general requirements for attribute certificates

6.2.7 Countersignatures

This clause defines two standard mechanisms for managing countersignatures. Details are given in clauses below.

6.2.7.1 Countersignature identifier in `Type` attribute of `ds:Reference`

The present document defines the following URI value:

- http://uri.etsi.org/01903#CountersignedSignature.

A XAdES signature containing a `ds:Reference` element whose `Type` attribute has this value will indicate that it is, in fact, a countersignature of the signature referenced by this element. The `ds:Reference` element shall be built so that the countersignature actually signs the `ds:SignatureValue` element of the countersigned signature. All the XMLDSIG rules apply in the processing of the aforementioned `ds:Reference` element. The only purpose of this definition is to serve as an easy identification of a signature as actually being a countersignature.

6.2.7.2 Enveloped countersignatures: the `CounterSignature` element

The `CounterSignature` property is an optional unsigned property that qualifies the signature. A XAdES signature may have more than one `CounterSignature` properties.

The `CounterSignature` property contains one countersignature of the qualified signature.

Below follows the schema definition for this element.

```
<xsd:element name="CounterSignature" type="CounterSignatureType" />

<xsd:complexType name="CounterSignatureType">
    <xsd:sequence>
        <xsd:element ref="ds:Signature"/>
    </xsd:sequence>
</xsd:complexType>
```

The content of this property is a XMLDSIG or XAdES signature whose `ds:SignedInfo` shall contain one `ds:Reference` element referencing the `ds:SignatureValue` element of the embedding and countersigned XAdES signature.

The content of the `ds:DigestValue` in the aforementioned `ds:Reference` element of the countersignature shall be the base-64 encoded digest of the complete (and canonicalized) `ds:SignatureValue` element (i.e. including the starting and closing tags) of the embedding and countersigned XAdES signature. Applications shall build this `ds:Reference` accordingly, using any of the mechanisms specified by XMLDSIG for achieving this objective. By doing this the countersignature actually signs the `ds:SignatureValue` element of the embedding XAdES signature.

Applications may add other `ds:Reference` elements referencing the `ds:SignatureValue` elements of previously existent `CounterSignature` elements. This allows for building arbitrarily long chains of explicit countersignatures.

A countersignature may itself be qualified by a `CounterSignature` property, which will have a `ds:Reference` element referencing the `ds:SignatureValue` of the first countersignature, built as described above. This is an alternative way of constructing arbitrarily long series of countersignatures, each one signing the `ds:SignatureValue` element of the one where it is directly embedded.

If the countersignature is a XAdES signature, its production shall follow the rules dictated by the present document.

Below follows the schema definition for this element.

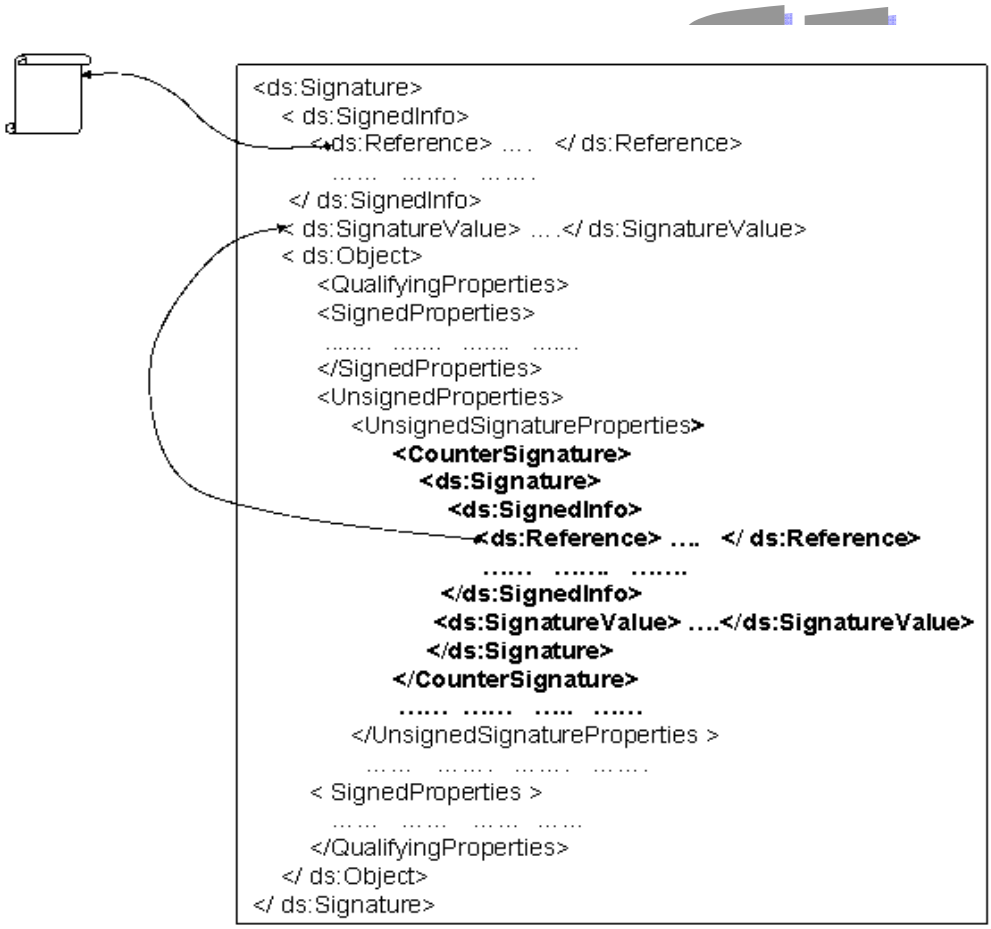


Figure 1: Use of CounterSignature element

6.2.8 Time-stamps on signed data objects

6.2.8.1 The `AllDataObjectsTimeStamp` element

The `AllDataObjectsTimeStamp` element is an optional signed property. Several instances of this property from different TSAs may occur within one XAdES signature.

The `AllDataObjectsTimeStamp` element contains the time-stamp computed before the signature production, over the sequence formed by ALL the `ds:Reference` elements within the `ds:SignedInfo` in their order of appearance referencing, whatever the signer wants to sign except the `SignedProperties` element.

Below follows the schema definition for this element.

```
<xsd:element name="AllDataObjectsTimeStamp" type="XAdESTimeStampType"/>
```

This property uses the Implicit mechanism. The input to the computation of the message imprint shall be the result of processing the aforementioned suitable `ds:Reference` elements in their order of appearance within `ds:SignedInfo` as follows:

1) Process the retrieved `ds:Reference` element according to the reference processing model of XMLDSIG.

2) If the result is a XML node set, canonicalize it as specified in clause 5.4.

3) Concatenate the resulting octets to those resulting from previously processed `ds:Reference` elements in `ds:SignedInfo`.

6.2.8.2 The `IndividualDataObjectsTimeStamp` element

The `IndividualDataObjectsTimeStamp` element is an optional signed property that qualifies the signed data object(s). Several instances of this property can occur within the one XAdES signature.

The `IndividualDataObjectsTimeStamp` element contains the time-stamp computed before the signature production, over a sequence formed by SOME `ds:Reference` elements within the `ds:SignedInfo` and any signed `ds:Manifest`. Note that this sequence cannot contain a `ds:Reference` computed on the `SignedProperties` element.

Below follows the schema definition for this element.

```
<xsd:element name="IndividualDataObjectsTimeStamp"
  type="XAdESTimeStampType"/>
```

This property uses the explicit (`Include`) mechanism. Generating applications shall compose the `Include` elements to refer to those `ds:Reference` elements that are to be time-stamped. Their corresponding `referencedData` attribute shall be present and set to `"true"`.

The message imprint computation input shall be the result of processing the selected `ds:Reference` within `ds:SignedInfo` as follows:

1) Process the retrieved `ds:Reference` element according to the reference processing model of XMLDSIG.

2) If the result is a XML node set, canonicalize it as specified in clause 5.4.

3) Concatenate the resulting octets to those resulting from previously processed `ds:Reference` elements in `ds:SignedInfo`.

6.2.9 The `SignaturePolicyIdentifier` element (XAdES-EPES)

The `SignaturePolicyIdentifier` property is a signed property qualifying the signature. At most one `SignaturePolicyIdentifier` element may be present in the signature.

Below follows the Schema definition for this type.

```
<xsd:element name="SignaturePolicyIdentifier" type="SignaturePolicyIdentifierType"/>
```

```
<xsd:complexType name="SignaturePolicyIdentifierType">
  <xsd:choice>
    <xsd:element name="SignaturePolicyId" type="SignaturePolicyIdType"/>
    <xsd:element name="SignaturePolicyImplied"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="SignaturePolicyIdType">
  <xsd:sequence>
    <xsd:element name="SigPolicyId" type="ObjectIdentifierType"/>
    <xsd:element ref="ds:Transforms" minOccurs="0"/>
    <xsd:element name="SigPolicyHash" type="DigestAlgAndValueType"/>
    <xsd:element name="SigPolicyQualifiers"
      type="SigPolicyQualifiersListType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SigPolicyQualifiersListType">
  <xsd:sequence>
    <xsd:element name="SigPolicyQualifier" type="AnyType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The `SignaturePolicyId` element will appear when the signature policy is identified using the first alternative. The `SigPolicyId` element contains an identifier that uniquely identifies a specific version of the signature policy.

The `SigPolicyHash` element contains the identifier of the hash algorithm and the hash value of the signature policy.

The optional `ds:Transforms` element may contain the transformations performed on the signature policy document before computing its hash. The processing model for these transformations is described in [2].

The present document defines a new Transform, which will be identified by setting the `ds:Transform`'s `Algorithm` attribute's value to:

http://uri.etsi.org/19132/v1.1.1/SignaturePolicy/SPDocDigestAsInSpecification

If used, this transform notifies that the hash value of the signature policy document has been computed as specified in a certain technical specification. If this transform is used, then the `SignaturePolicyIdentifier` shall be qualified at least by the `SPDocSpecification` qualifier, specified in (6.2.9.1), which identifies the aforementioned technical specification.

This transform may be used when the technical specifcation defines a mechanism for computing the hash value of the signature policy document that is not easily implementable using widely used XML technologies (e.g. XPath), as might occur, for instance, when the signature policy document is DER-encoded ASN.1.

This transform shall not be used in elements different than `SignaturePolicyId` element.

EDITOR NOTE: feedback from stakeholders is kindly requested, as the rationalized framework of electronic signatures anticipates that signature policy documents may be written in XML, ASN.1 or written in human readable way, and in past interoperability test events, raised an interoperability problem when a XAdES signature was pointing to a ASN.1 defined signature policy document.

The `SigPolicyQualifier` element may contain additional information qualifying the signature policy identifier.

Alternatively, the `SignaturePolicyImplied` empty element indicates that the data object(s) being signed and other external data imply the signature policy. It appears when the signature policy can be unambiguously derived from the semantics of the type of data object(s) being signed, and some other information.

6.2.9.1 Signature policy qualifier

Three qualifiers for the signature policy have been identified so far:

- a URL where a copy of the signature policy may be obtained (`SPURI` element);
- a user notice that should be displayed when the signature is verified (`SPUserNotice` element).
- An identifier of the technical specification that defines the syntax used for producing the signature policy document (`SPDocSpecification` element).

Below follows the Schema definition for `SPURI` and `SPUserNotice` elements.

```
<xsd:element name="SPURI" type="xsd:anyURI"/>
<xsd:element name="SPUserNotice" type="SPUserNoticeType"/>

<xsd:complexType name="SPUserNoticeType">
  <xsd:sequence>
    <xsd:element name="NoticeRef" type="NoticeReferenceType"
      minOccurs="0"/>
    <xsd:element name="ExplicitText" type="xsd:string"
      minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="NoticeReferenceType">
  <xsd:sequence>
    <xsd:element name="Organization" type="xsd:string"/>
    <xsd:element name="NoticeNumbers" type="IntegerListType"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="IntegerListType">
  <xsd:sequence>
    <xsd:element name="int" type="xsd:integer" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```


The `SPUserNotice` element is intended for being displayed whenever the signature is validated. The `ExplicitText` element contains the text of the notice to be displayed. Other notices could come from the organization issuing the signature policy. The `NoticeRef` element names an organization and identifies by numbers (`NoticeNumbers` element) a group of textual statements prepared by that organization, so that the application could get the explicit notices from a notices file.

Below follows the Schema definition for `SPDocSpecification` element:

```
<!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
<xsd:element name="SPDocSpecification" type="xades:ObjectIdentifierType"/>
```

If the technical specification is identified using an OID, then the `xades:Identifier` child shall contain a URN encoding this OID as specified in RFC 3061 [10], and its `QualifierType` attribute shall be present with its value set to `"OIDAsURN"`. If the technical specification is identified using a URI, then the `xades:Identifier` child shall contain this URI and its `QualifierType` attribute shall not be present.

EDITOR NOTE: This new qualifier will allow to identify whether the signature policy document is human readable, XML encoded, or ASN.1 encoded, by identifying the specific Technical Specifications where these formats will be defined.

6.2.10 The `xadesenv111:SignaturePolicyStore` element

The `xadesenv111:SignaturePolicyStore` is an optional unsigned property qualifying the signature.

The `xadesenv111:SignaturePolicyStore` property may be used to store the signature policy document which is referenced in the `SignaturePolicyIdentifier` attribute so that it can be used for offline and long-term validation.

Below follows the schema definition for this element:

```
<!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->

<xsd:element name="SignaturePolicyStore" type="xadesenv111:SignaturePolicyStoreType"/>
<xsd:complexType name="SignaturePolicyStoreType">
    <xsd:sequence>
        <xsd:element ref="xadesenv111:SPDocSpecification"/>
        <xsd:choice>
            <xsd:element name="SignaturePolicyDocument" type="xsd:base64Binary"/>
            <xsd:element name="SigPolDocLocalURI" type="xsd:anyURI"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

The `xadesenv111:SignaturePolicyStore` element may contain the base-64 encoded signature policy document as content of the `xadesenv111:SignaturePolicyDocument` element, or an URI to a local store where this document may be retrieved, as `xadesenv111:SigPolDocLocalURI` element's value.

NOTE 1: The URI value within `xadesenv111:SigPolDocLocalURI` element may be different than the `SPURI` qualifier's value.

The `xadesenv111:SPDocSpecification` element shall identify the technical specification that defines the syntax used for producing the signature policy document.

NOTE 2: It is the responsibility of the entity adding the signature policy into the signature-policy-store to make sure that the correct document is stored

6.3 The `SignatureTimeStamp` element (XAdES-T)

The `SignatureTimeSamp` property is an optional unsigned property qualifying the signature. A XAdES-T form signature may contain several `SignatureTimeSamp` elements, encapsulating time-stamp tokens obtained from different TSAs

The `SignatureTimeStamp` element encapsulates encapsulates the time-stamp token over the `ds:SignatureValue` element.

Below follows the schema definition for this element.

```
<xsd:element name="SignatureTimeStamp" type="XAdESTimeStampType"/>
```

This property uses the implicit mechanism as the time-stamped data object is always the same. For building the input to the message imprint computation, applications shall:

1) Take the `ds:SignatureValue` element and its contents.

2) Canonicalize it as specified in clause 5.4.

6.4 Properties for validation data values

This clause describes in detail those properties that allow the incorporation of validation data values to the electronic signature.

6.4.1 The `CertificateValues` Property element

The `CertificateValues` property is an optional unsigned property qualifying the signature. There shall be at most one occurrence of this property in the signature.

The `CertificateValues` element contains the full set of certificates that have been used to validate the electronic signature, including the signer's certificate, except those ones already present in the `ds:KeyInfo` element of the signature.

Below follows the schema definition for this element.

```
<xsd:element name="CertificateValues" type="CertificateValuesType"/>

<xsd:complexType name="CertificateValuesType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="EncapsulatedX509Certificate"
          type="EncapsulatedPKIDataType"/>
        <xsd:element name="OtherCertificate" type="AnyType"/>
    </xsd:choice>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The `EncapsulatedX509Certificate` element is able to contain the base-64 encoding of a DER-encoded X.509 certificate. The `OtherCertificate` element is a placeholder for potential future new formats of certificates.

Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain the certification chain for any TSUs providing such time-stamp tokens, if these certificates are not already present in the time-stamp tokens themselves as part of the TSUs' signatures. In this case, an element of this type could be added as an unsigned property to the XML time-stamp token using the incorporation mechanisms defined in the present document.

6.4.2 The `RevocationValues` property element

The `RevocationValues` property is an optional unsigned property that qualifies the signature. There shall be at most one occurence of this property in the signature.

The `RevocationValues` property element is used to hold the values of the revocation information that are to be shipped with the electronic signature.

Below follows the Schema definition for this element.

```
<xsd:element name="RevocationValues" type="RevocationValuesType"/>

<xsd:complexType name="RevocationValuesType">
  <xsd:sequence>
    <xsd:element name="CRLValues" type="CRLValuesType"
      minOccurs="0"/>
    <xsd:element name="OCSPValues" type="OCSPValuesType"
      minOccurs="0"/>
    <xsd:element name="OtherValues" type="OtherCertStatusValuesType"
      minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID"  use="optional"/>
</xsd:complexType>
```

Revocation information can include Certificate Revocation Lists (`CRLValues`) or responses from an online certificate status server (`OCSPValues`). Additionally a placeholder for other revocation information (`OtherValues`) is provided for future use.

```
<xsd:complexType name="CRLValuesType">
  <xsd:sequence>
     <xsd:element name="EncapsulatedCRLValue"
       type="EncapsulatedPKIDataType"
       maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Certificate Revocation Lists (`CRLValues`) shall consist of a sequence of at least one Certificate Revocation List. Each `EncapsulatedCRLValue` shall contain the base-64 encoding of a DER-encoded X.509 CRL. Should the validation data contain one or more Delta CRLs, this property shall include the set of CRLs required to provide complete revocation lists.

```
<xsd:complexType name="OCSPValuesType">
  <xsd:sequence>
      <xsd:element name="EncapsulatedOCSPValue"
        type="EncapsulatedPKIDataType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

OCSP Responses (`OCSPValues`) consist of a sequence of at least one OCSP Response. The `EncapsulatedOCSPValue` element shall contain the base-64 encoding of a DER-encoded `OCSPResponse` defined in RFC 2560 [7].

```
<xsd:complexType name="OtherCertStatusValuesType">
  <xsd:sequence>
    <xsd:element name="OtherValue" type="AnyType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The `OtherValues` element provides a placeholder for other revocation information that can be used in the future.

Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain the values of revocation data including CRLs and OCSP responses for any TSUs providing such time-stamp tokens, if they are not already present in the time-stamp tokens themselves as part of the TSUs' signatures. In this case, an element of this type could be added as an unsigned property to the XML time-stamp token using the incorporation mechanisms defined in the present document.

6.4.3 The `AttrAuthoritiesCertValues` element

The `AttrAuhtoritiesCertValues` property is an optional unsigned property that qualifies the signature. There shall be at most one occurrence of this property in the signature.

This property contains the certificate values of the Attribute Authorities that have been used to validate the attribute certificate when present in the signature. It may also contain values of the CA certificates within the AA certificate's certification path if they are not present elsewhere in the XAdES signature.

EDITOR NOTE: previous versions of XAdES did not make this issue clear. Feedback is requested from stakeholders regarding the suitability of this amendment.

It may also contain the certificates of signers of signed assertions present within the `xadesenv111:SignedAssertions` element, and CA certificates within their certification paths if they are not present elsewhere in the XAdES signature.

EDITOR NOTE: this may also be the place for the material related to the signed assertions, even if the entities that issue such signed assertions are not considered AttributeAuthorities.

Below follows the Schema definition for this element.

```
<xsd:element name="AttrAuthoritiesCertValues" type="CertificateValuesType"/>
```

Any certificate present within `CertificateValues` property, which has been used for validating the attribute certificate, does not need to appear within the `AttrAuthoritiesCertValues`.

6.4.4 The `AttributeRevocationValues` Property element

The `AttributeRevocationValues` property is an optional unsigned property that qualifies the signature. There shall be at most one occurrence of this property in the signature.

This property contains the set of revocation data that have been used to validate the attribute certificate when present in the signature, if not present anywhere else within the XAdES signature. It may also contain the set of revocation data that have been used to validate the signatures of signed assertions within `xadesenv111:SignedAssertions` element, if not present elsewhere within the XAdES signature.

Below follows the Schema definition for this element.

```
<xsd:element name="AttributeRevocationValues" type="RevocationValuesType"/>
```

Any revocation data present within `RevocationValues` property, which has been used for validate the attribute certificate or the signed assertions, does not need to appear within the `AttributeRevocationValues`.

EDITOR NOTE: this may also be the place for the revocation material related to the signed assertions, even if the entities that issue such signed assertions are not considered AttributeAuthorities

Should the validation data contain one or more Delta CRLs, this property shall include the set of CRLs required to provide complete revocation lists

6.5 Properties for XAdES-A form

6.5.1 The `xadesv141:TimeStampValidationData` element

The `TimeStampValidationData` element is an optional unsigned property qualifying the signature. Several occurrences of this element may be present within a XAdES signature.

This element is specified to serve as an optional container for validation data required for carrying a full verification of time-stamp tokens embedded within any of the different time-stamp containers defined in the present document.

Below follows the schema definition for this element.

```
<!-- targetNamespace="http://uri.etsi.org/101903/v1.4.1#" -->

<xsd:element name="TimeStampValidationData" type=""xadesv141:ValidationDataType"/>

<xsd:complexType name="ValidationDataType">
    <xsd:sequence>
        <xsd:element ref="xades:CertificateValues" minOccurs="0" />
        <xsd:element ref="xades:RevocationValues" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
    <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>
```


The structure of `xades:CertificateValues` child is defined in clause 6.41. When present, it shall contain certificates used in the full verification of time-stamp tokens embedded in one XAdES time-stamp container. This element may contain all the certificates required for a full verification of the time-stamp tokens, but it may also contain only a part of them if the rest are present in other place of the XAdES signature (like within the time-stamp token itself, or even in other `xadesv141:TimeStampValidationData` created for other time-stamp tokens).

The structure of `xades:RevocationValues` child is defined in clause 6.4.2. When present, it shall contain the revocation information used in the full verification of time-stamp tokens embedded in one XAdES time-stamp container. This element may contain all the revocation information pieces (for instance CRLs or OCSP responses) required for a full verification of the time-stamp tokens, but it may also contain only a part of them if the rest are present in other place of the XAdES signature (like within the time-stamp token itself, or even in other `xadesv141:TimeStampValidationData` created for other time-stamp tokens).

Optional `Id` attribute allows referencing this element.

Optional `URI` attribute, when present, is used for referencing the time-stamp container of the time-stamp token whose validation data is contained within this element.

6.5.1.1 Use of `URI` attribute

When a XAdES signature requires to include all the validation data required for a full verification of a time-stamp token embedded in any of the following containers: `SignatureTimeStamp`, `RefsOnlyTimeStamp`, `SigAndRefsTimeStamp`, or `ArchiveTimeStamp`, and that validation data is not present in other parts of the signature, a new `xadesv141:TimeStampValidationData` element shall be created containing the missing validation data information and it shall be added as a child of `UnsignedSignatureProperties` elements immediately after the respective time-stamp token container element. Under these circumstances there is no need to use `URI` attribute as the identification of the related time-stamp token container is implicit in the relative position of both elements, the container and the `xadesv141:TimeStampValidationData` element.

When a XAdES requires to include all the validation data required for a full verification of a time-stamp token embedded in any of the following containers: `IndividualDataObjectsTimeStamp` or `AllDataObjectTimeStamp`, the treatment is different because first, there may be more than one signed time-stamp tokens containers, and second they are signed properties whereas the corresponding `xadesv141:TimeStampValidationData` elements are unsigned and they appear as children of different parents. Under these circumstances the `URI` attribute within the corresponding `xadesv141:TimeStampValidationData` element shall be present and shall be used to reference the specific signed container encapsulating time-stamp tokens whose validation data that element actually contains.

6.5.2 The `xadesv141:ArchiveTimeStamp` element

The `xadesv141:ArchiveTimeStamp` element is an optional unsigned property qualifying the signature. Several occurrences of this element may be present within a XAdES signature.

Below follows the schema definition for this element.

```
<xsd:element name="ArchiveTimeStamp" type="XAdESTimeStampType"/>
```

Should the XAdES signature incorporate a `CounterSignature` unsigned property, implementers should ensure that all the required material for conducting the validation of the counter-signature is incorporated to the XAdES signature before generating the first `xadesv141:ArchiveTimeStamp` property. This may be done within the counter-signature itself or within the containers available within the counter-signed XAdES signature.

Should a `CounterSignature` unsigned property be time-stamped by the `xadesv141:ArchiveTimeStamp`, any ulterior change of their contents (by addition of unsigned properties if the counter-signature is a XAdES signature, for instance) would make the validation of the `xadesv141:ArchiveTimeStamp`, and in consequence of the countersigned XAdES signature, fail. Implementers should, in consequence, not change the contents of the `CounterSignature` property once it has been time-stamped by the `xadesv141:ArchiveTimeStamp`. Implementors may, under these circumstances, to make use of the detached counter-signature mechanism specified in clause 6.2.7.1.

In addition it has to be noted that the present document allows to counter-sign a previously time-stamped countersignature with another `CounterSignature` property added to the embedding XAdES signature after the time-stamp container.

NOTE 1: Readers are warned that once an `xadesv141:ArchiveTimeStamp` property is added to the signature, any ulterior addition of a `ds:Object` to the signature would make the verification of such time-stamp fail.

Depending whether all the unsigned properties covered by the time-stamp token and the `xadesv141:ArchiveTimeStamp` property itself have the same parent or not, its contents may be different. Details are given in clauses below.

6.5.2.1 Not distributed case

When `xadesv141:ArchiveTimeStamp` and all the unsigned properties covered by its time-stamp token have the same parent, this property uses the Implicit mechanism for all the time-stamped data objects. The input to the computation of the digest value shall be built as follows:

1) Initialize the final octet stream as an empty octet stream.

2) Take all the `ds:Reference` elements in their order of appearance within `ds:SignedInfo` referencing whatever the signer wants to sign including the `SignedProperties` element. Process each one as indicated below:

   - Process the retrieved `ds:Reference` element according to the reference processing model of XMLDSIG.

   - If the result is a XML node set, canonicalize it as specified in clause 5.4.

   - Concatenate the resulting octets to the final octet stream.

3) Take the following XMLDSIG elements in the order they are listed below, canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream:

- The `ds:SignedInfo` element.
- The `ds:SignatureValue` element.
- The `ds:KeyInfo` element, if present.

4) Take the unsigned signature properties that appear before the current `xadesv141:ArchiveTimeStamp` in the order they appear within the `xades:UnsignedSignatureProperties`, canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream. While concatenating the following rules apply:

- The `xades:CertificateValues` property shall be added if it is not already present and the ds:KeyInfo element does not contain the full set of certificates used to validate the electronic signature.
- The `xades:RevocationValues` property shall be added if it is not already present and the ds:KeyInfo element does not contain the revocation information that has to be shipped with the electronic signature.
- The `xades:AttrAuthoritiesCertValues` property shall be added if not already present and the following conditions are true: there exist an attribute certificate in the signature AND a number of certificates that have been used in its validation do not appear in `CertificateValues`.
- The `xades:AttributeRevocationValues` property shall be added if not already present and there the following conditions are true: there exist an attribute certificate AND some revocation data that have been used in its validation do not appear in `RevocationValues`.

5) Take all the `ds:Object` elements except the one containing `xades:QualifyingProperties` element. Canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream.

6.5.2.2 Distributed case

When `xadesv141:ArchiveTimeStamp` and some of the unsigned properties covered by its time-stamp token DO NOT have the same parent, applications shall use the explicit (based on `xades:Include` elements) mechanism only for referencing the unsigned properties. Applications SHALL build one `xades:Include` element for each unsigned property that is covered by the time-stamp token. These `xades:Include` elements will be added in the same order as the unsigned properties are processed for contributing to the digest computation input.

No `xades:Include` elements are generated for any other XMLDSIG element present in the signature, although they are actually time-stamped as they contribute to the generation of the message imprint computation input.

Generating applications shall build digest computation input as for the Implicit case (clause 6.5.4.1) substituting step 4 by the one specified below:

4) Take the unsigned signature properties present in the signature, extract comment nodes, canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream. While concatenating, the following rules apply:

- The `xades:CertificateValues` property shall be added if it is not already present and the ds:KeyInfo element does not contain the full set of certificates used to validate the electronic signature.
- The `xades:RevocationValues` property shall be added if it is not already present and the ds:KeyInfo element does not contain the revocation information that has to be shipped with the electronic signature.
- The `xades:AttrAuthoritiesCertValues` property shall be added if not already present and the following conditions are true: there exist an attribute certificate in the signature AND a number of certificates that have been used in its validation do not appear in `CertificateValues`.

- The xades:AttributeRevocationValues property shall be added if not already present and the following conditions are true: there exist attribute certificates AND some revocation data that have been used in its validation do not appear in RevocationValues.

6.5.3 The xadesenv111:RenewedDigests element

EDITOR NOTE: Feedback on the suitability of this property and its definition is kindly requested to stakeholders.

The xadesenv111:RenewedDigests property is an optional unsigned property qualifying the signature. Several occurrences of this element may be present within a XAdES signature. This property may be used when the electronic signature contains one or more signed ds:Manifest referencing data objects that are detached from the signature.

The xadesenv111:RenewedDigests property contains the digest values of the aforementioned indirectly signed detached data objects computed with a stronger digest algorithm than the one used for computing the digest values present within the ds:Manifest's ds:Reference children.

Below follows the schema definition for this element. Note that the elements are defined within the xadesenv111 namespace, whose URI is http://uri.etsi.org/19132/v1.1.1#.

```
<!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->

<xsd:element name="RenewedDigests" type="xadesenv111:RenewedDigestsType"/>

<xsd:complexType name="RenewedDigestsType">
    <xsd:sequence>
        <xsd:element ref="ds:DigestMethod"/>
        <xsd:element name="RecomputedDigestValue" type="xadesenv111:RecomputedDigestValueType"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<xsd:complexType name="RecomputedDigestValueType">
    <xsd:simpleContent>
        <xsd:extension base="ds:DigestValueType">
            <xsd:attribute name="order" type="xsd:integer" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```


The ds:DigestMethod child indicates a digest algorithm (stronger than the algorithm used for computing the digest of the signed data objects referenced within a signed ds:Manifest element).

Each RecomputedDigestValue child contains the base-64 encoded digest value, computed with the digest algorithm indicated within the aforementioned ds:DigestMethod, of some (or all) of the signed data objects referenced within a signed ds:Manifest element.

The mandatory RecomputedDigestValue's order attribute identifies the specific ds:Reference element that was referencing the data object whose digest value is recomputed with the stronger digest algorithm. Its value is an integer that identifies the order of appearance of the identified ds:Reference when the XAdES signature is serialized, where the value "1" is assigned to the first ds:Reference child of the ds:SignedInfo element.

When implementers suspect that a certain digest algorithm is becoming weak, when one or more detached data objects have been indirectly signed using that algorithm with a signed ds:Manifest, and when they suspect that the aforementioned data objects migh be substituted by others with the same digest due to the weakness of the digest algorithm, they may decide to generate a xadesenv111:RenewedDigests element including the digest values of the aforementioned data objects computed with a stronger algorithm, and incorporate it to the XAdES signature. This recomputation of digest counters the threat explained in clause D.1.14.

When validating a XAdES signature that incorporates this property, the validation process shall include a step consisting in taking each ds:Reference element within xadesenv111:RenewedDigests property, process it according to the XMLDSIG processing model, digest the retrieved data object using the digest method indicated within this ds:Reference, and compare it with the digest value present within this ds:Reference. In absence of any other failure in the validating process, a failure in one of these checks indicates that the corresponding data object indirectly signed has been replaced by another data object.

Figure 2 shows a XAdES signature that incorporates a signed `ds:Manifest`, whose `ds:Reference` children reference two detached data objects, which is generated at time t2 and time-stamped at time t3. The digest values present within the aforementioned `ds:Reference` elements have been computed using a digest algorithm alg1.

The figure shows that at a certain posterior time, it is suspected that algorithm alg1 is becoming too weak and is decided to incorporate in the signature the digest on the two detached data objects computed with a stronger digest algorithm alg2, within the `xadesenv111:RenewedDigests` unsigned property (times t4 and t5 respectively), and afterwards, incorporate a `xadesv141:ArchiveTimeStamp` in t6.

The figure also shows that the XAdES signature includes two `ds:SignedInfo`'s `ds:Reference` children elements (which would be the first and the second `ds:Reference` elements appearing within the serialized signature), and two `ds:Manifest`'s `ds:Reference` children elements, which would appear respectively as third and fourth `ds:Reference` elements (orders 3 and 4) when serializing the signature. In consequence, the values of the `order` attributes of the `xadesenv111:RenewedDigest` elements containing the renewed digest values of the data objects referenced by the two aforementioned `ds:Manifest`'s `ds:Reference` children elements, would be "3" and "4" respectively.

Usage of `xadesenv111:RenewedDigests` unsigned property achieves two effects: first of all it counters the threat resulting of the combination of digest algorithm break and detached data object substitution; and secondly it allows to identify the substituted data object and preserve the validity of the signature for the not subsituted detached data objects.Clause D.1.14 provides additional details.

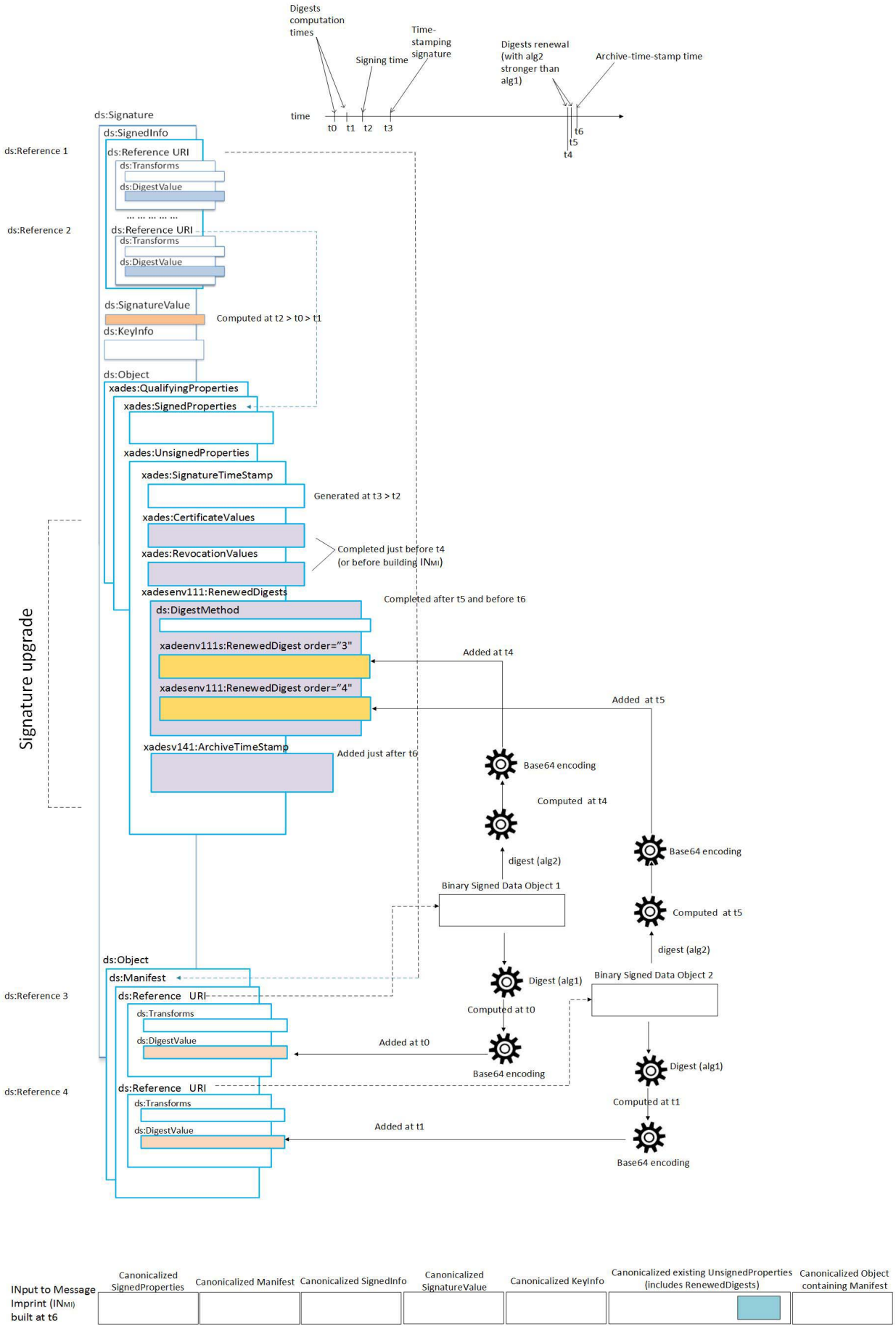


Figure 2: Use of `xadesenv111:RenewedDigests`

.

7. Conformance requirements

The present document defines several conformance levels. Subclauses below define the conformance levels for the XAdES forms specified within clause 4.1 of the normative body part, namely:

- XAdES Basic Electronic Signature (XAdES-BES)
- XAdES Explicit Policy-based Electronic Signature (XAdES-EPES)
- XAdES with trusted Time (XAdES-T)
- XAdES with Archive-time-stamp (XAdES-A)

The normative annex C defines additional conformance levels for XAdES forms that include references to validation material, namely:

- electronic signatures with Complete validation data references (XAdES-C)
- EXtended electronic signature with time forms, Type 1 and Type 2 (XAdES-X Type 1 and XAdES-X Type 2)
- EXtended Long electronic signatures with time forms, Type 1 and Type 2(XAdES-X-L Type 1, and XAdES-X-L Type 2)

NOTE: The conformance levels defined within the present document are more general than the levels defined in the baseline profile, see part 2 of the current series.

An implementation claiming to be conformant to a specific level of the present document shall fulfil the corresponding requirements defined in the current clause or in appendix C.

7.1 XAdES-Basic Electronic Signature (XAdES-BES) conformance level

A XAdES signature claiming conformance to XAdES-BES level shall, at a minimum, consisting of the following components:

- The `ds:Signature` element as specified in [2].
- At least one of the following:
  - One of the signed properties referencing the signing certificate, i.e. the `SigningCertificate` (as defined in clause 6.2.2.1) or `xadesenv111:SigningCertificate` (as defined in clause 6.2.2.2) incorporated (directly or indirectly) to the signature as defined in clause 5.3;
  - the `ds:KeyInfo` element whose contents satisfy the restrictions specified in clause 4.1.1.

7.2 XAdES-Explicitly Policy based Electronic Signature (XAdES-EPES) conformance level

A XAdES signature claiming conformance to XAdES-EPES level shall, at a minimum, consisting of the following components:

- All the components required for fulfilling conformance to XAdES-BES conformance level plus
- The `xades:SignaturePolicyIdentifier` signed property directly or indirectly incorporated.

7.3 XAdES with trusted Time(XAdES-T) conformance level

A signature claiming conformance to xAdES-T level shall be built upon a signature compliant with XAdES-BES or XAdES-EPES conformance level, and, in addition, there shall exist one or more trusted times associated with the signature. An instance of trusted time may be provided by:

- a `xades:SignatureTimeStamp` unsigned property directly or indirectly incoporated to the signature; or
- a time-mark of the electronic signature provided by a Trusted Service Provider.

7.4 XAdES with Archive-time-stamp (XAdES-A) conformance level

A XAdES signature claiming conformance to XAdES-A level shall be built upon signatures compliant with XAdES-T, XAdES-C, XAdES-X (type 1 or 2), and XAdES-XL (type 1 or 2) conformance levels. In addition it:

- shall directly or indirectly incorporate one or more instances of `xadesv141:ArchiveTimeStamp` property.
- may directly or indirectly incorporate one instance of `xades:CertificateValues` property. See clause 6.4.1 for details.
- may directly or indirectly incorporate one instance of `xades:RevocationValues` property. See clause 6.4.2 for details.
- may directly or indirectly incorporate one instance of `xades:AttrAuthoritiesCertValues`. See clause 6.4.3 for details.
- may directly or indirectly incorporate one instance of `xades:AttributeRevocationValues` property. See clause 6.4.4 for details.

Annex <A> (normative): Additional Qualifying Properties Specification

A.1 Qualifying properties for validation data

The following sub-clauses describe in detail qualifying properties that can contain references to certificates and revocation values that have been used in the validation of the electronic signature.

A.1.1 References to CA certificates

Sub-clauses below specify two unsigned properties qualifying the signature used as containers of references and digest values of CA certificates, namely: `CompleteCertificateRefs` and `xadesenv111:CompleteCertificateRefs`.

Only one of these two properties may be incorporated to a XAdES signature: if one of these properties is incorporated, then the other one shall not be incorporated.

EDITOR NOTE: feedback from stakeholders is requested on the solution proposed: define the new `xadesenv111:CompleteCertificateRefs` for acknowledging deprecation of `ds:X509IssuerSerial`, but keep `xades:CompleteCertificateRefs` as in the end, the problem is created by some XML Schema validation tools

A.1.1.1 The `CompleteCertificateRefs` element

The `CompleteCertificateRefs` property is an optional unsigned property that qualifies the signature. There shall be at most one occurrence of this property in the signature.

The `CompleteCertificateRefs` property carries references to the CA certificates that have been used in the validation of the electronic signature.

Below follows the schema definition for this element.

```
<xsd:element name="CompleteCertificateRefs" type="CompleteCertificateRefsType"/>

<xsd:complexType name="CompleteCertificateRefsType">
  <xsd:sequence>
    <xsd:element name="CertRefs" type="CertIDListType" />
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The `CertRefs` element contains a sequence of `Cert` elements already defined in clause 6.2.2.1, incorporating the digest of each certificate and the issuer and serial number identifier.

If `CompleteCertificateRefs` and `CertificateValues` are present, all the certificates referenced in `CompleteCertificateRefs` shall be present either in the `ds:KeyInfo` element of the signature or in the `CertificateValues` element.

Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain references to the certification chain for any TSUs providing such time-stamp tokens. In this case, an element of this type could be added as an unsigned property to the XML time-stamp token using the incorporation mechanisms defined in the present document.

A.1.1.2 The xadesenv111:CompleteCertificateRefs element

The xadesenv111:ompleteCertificateRefs property is an optional unsigned property that qualifies the signature. There shall be at most one occurrence of this property in the signature.

The xadesenv111:CompleteCertificateRefs property carries references to the CA certificates that have been used in the validation of the electronic signature.

Below follows the schema definition for this element

```
<!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->


<xsd:element name="CompleteCertificateRefs" type="xadesenv111:CompleteCertificateRefsType"/>
<xsd:complexType name="CompleteCertificateRefsType">
    <xsd:sequence>
        <xsd:element name="CertRefs" type="xadesenv111:CertIDListType"/>
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The xadesenv111:CertRefs element contains a sequence of xadesenv111:Cert elements already defined in clause 6.2.2.2, incorporating the digest of each certificate and information (as strings) of the issuer and the serial number.

If xadesenv111:CompleteCertificateRefs and CertificateValues are present, all the certificates referenced in xadesenv111:CompleteCertificateRefs shall be present either in the ds:KeyInfo element of the signature or in the CertificateValues element

NOTE: See NOTE in clause 6.2.2.2 providing rationale for the definition of this new property.

EDITOR NOTE: Feedback is kindly requested to stakeholders on the suitability of defining this new property

A.1.2 The CompleteRevocationRefs element

The CompleteRevocationRefs property is an optional unsigned property that qualifies the signature. There shall be at most one occurrence of this property in the signature. This occurrence shall not be empty.

The CompleteRevocationRefs property will carry references to revocation values used for the validation of the electronic signature.

Below follows the schema definition for this element.

```
<xsd:element name="CompleteRevocationRefs"
  type="CompleteRevocationRefsType"/>

<xsd:complexType name="CompleteRevocationRefsType">
  <xsd:sequence>
    <xsd:element name="CRLRefs" type="CRLRefsType" minOccurs="0"/>
    <xsd:element name="OCSPRefs" type="OCSPRefsType" minOccurs="0"/>
    <xsd:element name="OtherRefs" type="OtherCertStatusRefsType"
      minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CRLRefsType">
  <xsd:sequence>
    <xsd:element name="CRLRef" type="CRLRefType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CRLRefType">
  <xsd:sequence>
    <xsd:element name="DigestAlgAndValue"
      type="DigestAlgAndValueType"/>
    <xsd:element name="CRLIdentifier" type="CRLIdentifierType"
```

```
        minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="CRLIdentifierType">
    <xsd:sequence>
      <xsd:element name="Issuer" type="xsd:string"/>
      <xsd:element name="IssueTime" type="xsd:dateTime" />
      <xsd:element name="Number" type="xsd:integer"  minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="OCSPRefsType">
    <xsd:sequence>
      <xsd:element name="OCSPRef" type="OCSPRefType"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="OCSPRefType">
    <xsd:sequence>
      <xsd:element name="OCSPIdentifier" type="OCSPIdentifierType"/>
      <xsd:element name="DigestAlgAndValue"
        type="DigestAlgAndValueType"
        minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ResponderIDType">
    <xsd:choice>
      <xsd:element name="ByName" type="xsd:string"/>
      <xsd:element name="ByKey" type="xsd:base-64Binary"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="OCSPIdentifierType">
    <xsd:sequence>
      <xsd:element name="ResponderID" type="ResponderIDType"/>
      <xsd:element name="ProducedAt" type="xsd:dateTime"/>
    </xsd:sequence>
    <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="OtherCertStatusRefsType">
    <xsd:sequence>
      <xsd:element name="OtherRef" type="AnyType"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

The `CompleteRevocationRefs` element can contain:

- sequences of references to CRLs (`CRLRefs` element);
- sequences of references to `OCSPResponse` data as defined in RFC 2560 [7] (`OCSPRefs` element);
- other references to alternative forms of revocation data (`OtherRefs` element).

Each element in a `CRLRefs` sequence (`CrlRef` element) references one CRL. Each reference contains:

- the digest of the entire DER encoded CRL (`DigestAlgAndValue` element);
- a set of data (`CRLIdentifier` element) including the issuer (`Issuer` element), the time when the CRL was issued (`IssueTime` element) and optionally the number of the CRL (`Number` element). `CRLIdentifier` element contents shall follow the rules established by XMLDSIG [2] in its clause 4.5.4.1 for strings representing Distinguished Names. In addition, this element can be dropped if the CRL could be inferred from other information. Its `URI` attribute could serve to indicate where the identified CRL is archived.

NOTE: The `number` element is an optional hint helping applications to get the CRL whose digest matches the value present in the reference.

Should one or more of the identified CRLs be a Delta CRL, this property shall include references to the set of CRLs required to provide complete revocation lists.

Each element in an `OCSPRefs` sequence (`OcspRef` element) references one OCSP response. Each reference contains:

- a set of data (`OCSPIdentifier` element) that includes an identifier of the responder and an indication of the time when the response was generated. The responder may be identified by its name, using the `Byname` element within `ResponderID`. The responder may also be identified by the digest of the server's public key computed as mandated in RFC 2560 [7], using the `ByKey` element. In this case the content of the `ByKey` element will be the DER value of the `byKey` field defined in RFC 2560, base-64 encoded. The contents of `ByName` element shall follow the rules established by XMLDSIG [2] in its clause 4.5.4.1 for strings representing Distinguished Names. The generation time indication appears in the `ProducedAt` element and corresponds to the "ProducedAt" field of the referenced response. The optional `URI` attribute could serve to indicate where the OCSP response identified is archived;

- the digest computed on the DER encoded `OCSPResponse` defined in RFC 2560 [7], appearing within `DigestAlgAndValue` element. Applications claiming alignment with the present document should include the `DigestAlgAndValue` element within each `OCSPRef` element.

Alternative forms of validation data can be included in this property making use of the `OtherRefs` element, a sequence whose items (`OtherRef` elements) can contain any kind of information.

If `CompleteRevocationRefs` and `RevocationValues` are present, all the revocation data referenced in `RevocationRefs` shall be present either in the `ds:KeyInfo` element of the signature or in the `RevocationValues` property element.

Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain references to the full set of CRL or OCSP responses that have been used to verify the certification chain for any TSUs providing such time-stamp tokens. In this case, an element of this type could be added as an unsigned property to the XML time-stamp token using the incorporation mechanisms defined in the present document.

A.1.3 References to certificates in the certification path of Attribute Authorities certificates

Sub-clauses below specify two unsigned properties qualifying the signature used as containers of references and digest values of certificates within the certification path of AA certificates, namely: `AttributeCertificateRefs` and `xadesenv111:AttributeCertificateRefs`.

Only one of these two properties may be incorporated to a XAdES signature: if one of these properties is incorporated, then the other one shall not be incorporated.

EDITOR NOTE: feedback from stakeholders is requested on the solution proposed: define the new `xadesenv111:AttributeCertificateRefs` for acknowledging deprecation of `ds:X509IssuerSerial`, but keep `xades:AttributeCertificateRefs` as in the end, the problem is created by some XML Schema validation tools

A.1.3.1 The `AttributeCertificateRefs` element

The `AttributeCertificateRefs` property is an optional unsigned property that qualifies the signature. This property may be used only when a user attribute certificate is present in the signature within the signature. There shall be at most one occurrence of this property in the signature.

The `AttributeCertificateRefs` element will carry the references to the set of Attribute Authorities certificates that have been used to validate the attribute certificates. It may also contain references to the CA certificates within the certification paths of the attribute certificates, if not incorporated elsewhere within the XAdES signature.

EDITOR NOTE: previous versions of XAdES did not make this issue clear. Feedback is requested from stakeholders regarding the suitability of this amendment.

It may also contain references to the full set of revocation data that have been used to validate the signatures of signed assertions within xadesenv111:SignedAssertions element, if not present elsewhere within the XAdES signature.

EDITOR NOTE: it may also contain material for signed assertions. Feedback kindly requested from stakeholders.

Below follows the schema definition for this element.

```
<xsd:element name="AttributeCertificateRefs" type="CompleteCertificateRefsType"/>
```

NOTE: Copies of the certificates referenced in this property may be held using the AttrAuthoritiesCertValues property.

If AttributeCertificateRefs and AttrAuthoritiesCertValues are present, AttrAuthoritiesCertValues, CertificateValues, and ds:KeyInfo properties shall contain all the certificates referenced in AttributeCertificateRefs.

A.1.3.2 The xadesenv111:AttributeCertificateRefs element

The xadesenv111:AttributeCertificateRefs property is an optional unsigned property that qualifies the signature. This property may be used only when a user attribute certificate is present in the signature within the signature. There shall be at most one occurrence of this property in the signature.

The xadesenv111:AttributeCertificateRefs element will carry the references to the set of Attribute Authorities certificates that have been used to validate the attribute certificates. It may also contain references to the CA certificates within the certification paths of the attribute certificates, if not incorporated elsewhere within the XAdES signature. It may also contain references to the certificates of signers of signed assertions present within the xadesenv111:SignedAssertions element, and CA certificates within their certification paths if they are not present elsewhere in the XAdES signature

Below follows the schema definition for this element

```
<!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->

<xsd:element name="AttributeCertificateRefs" type="xadesenv111:CompleteCertificateRefsType"/>
```

The semantics of the elements and types defined above are identical to the elements and types defined in clause A.1.1.1, with the only exception that ds:X509IssuerSerial element is now optional within xadesenv111:CertIDType and in consequence within xadesenv111:CertIDListType and xadesenv111:AttributeCertificateRefs.

NOTE: See NOTE in clause 6.2.2 providing rationale for the definition of this new property

If xadesenv111:AttributeCertificateRefs and AttrAuthoritiesCertValues are present, AttrAuthoritiesCertValues, CertificateValues, and ds:KeyInfo properties shall contain all the certificates referenced in xadesenv111:AttributeCertificateRefs.

A.1.4 The AttributeRevocationRefs element

The AttributeRevocationRefs property is an optional unsigned property that qualifies the signature. This property may be used only when a user attribute certificate is present in the signature within the signature. There shall be at most one occurrence of this property in the signature.

The AttributeRevocationRefs property may carry the references to the full set of revocation data that have been used in the validation of the attribute certificate(s) present in the signature. It may also contain references to the full set of revocation data that have been used to validate the signatures of signed assertions within xadesenv111:SignedAssertions element, if not present elsewhere within the XAdES signature.

Below follows the schema definition for this element.

```
<xsd:element name="AttributeRevocationRefs" type="CompleteRevocationRefsType"/>
```

NOTE: Copies of the revocation values referenced in this property may be held using the `AttributeRevocationValues` property.

If `AttributeRevocationRefs` and `AttributeRevocationValues` are present, `AttrAuthoritiesCertValues`, `CertificateValues`, and `ds:KeyInfo` shall contain the values of all the objects referenced in `AttributeRevocationRefs`

Should one or more of the identified CRLs be a Delta CRL, this property shall include references to the set of CRLs required to provide complete revocation lists.

A.1.5 Time-stamps on references to validation data

Clauses below specify containers for time-stamp tokens that cover the properties that encapsulate references to validation material. Two elements are specified depending on which are the elements that are actually time-stamped, namely the `SigAndRefsTimeStamp` and the `RefsOnlyTimeStamp`.

A.1.5.1 The `SigAndRefsTimeStamp` element

The `SigAndRefsTimeStamp` property is an optional unsigned property qualifying the signature. Clause B.2.1 proposes a XAdES form that can incorporate one or more `SigAndRefsTimeStamp` elements.

Below follows the schema definition for this element.

```
<xsd:element name="SigAndRefsTimeStamp" type="XAdESTimeStampType"/>
```

This property contains a time-stamp token that covers the following data objects: `ds:SignatureValue` element, all present `SignatureTimeStamp` elements, `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, `CompleteRevocationRefs`, and when present, `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs`, and `AttributeRevocationRefs`.

Depending whether all the aforementioned time-stamped unsigned properties and the `SigAndRefsTimeStamp` property itself have the same parent or not, its contents may be different. Details are given in clauses below

A.1.5.1.1 Not distributed case

When `SigAndRefsTimeStamp` and all the unsigned properties covered by its time-stamp token have the same parent, this property uses the Implicit mechanism. The input to the computation of the digest value shall be the result of taking in order each of the data objects listed below, canonicalize each one as specified in clause 5.4, and concatenate the resulting octet streams:

1) The `ds:SignatureValue` element.

2) Those among the following unsigned properties that appear before `SigAndRefsTimeStamp`, in their order of appearance within the `UnsignedSignatureProperties` element:

   - The `SignatureTimeStamp` elements.
   - The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` element.
   - The `CompleteRevocationRefs` element.
   - The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs` element if this property is present.
   - The `AttributeRevocationRefs` element if this property is present.

Below follows the list -in order- of data objects that contribute to the digest computation. Elements within [] contribute in their order of appearance within the `UnsignedSignatureProperties` element, not in the order they are enumerated below:

```
(ds:SignatureValue, [SignatureTimeStamp+, (CompleteCertificateRefs |
xadesenv111:CompleteCertificateRefs), CompleteRevocationRefs,
(AttributeCertificateRefs |xadesenv111:AttributeCertificateRefs)?,
AttributeRevocationRefs?]).
```

A.1.5.1.2 Distributed case

When `SigAndRefsTimeStamp` and some of the unsigned properties covered by its time-stamp token DO NOT have the same parent, applications shall build this property as indicated below:

1) No `Include` element will be added for `ds:SignatureValue`. All applications shall implicitly assume its contribution to the digest input (see below in this clause).

2) Generate one `Include` element per each unsigned property that shall be covered by the time-stamp token in the order they appear listed below:

   - The `SignatureTimeStamp` elements.
   - The `CompleteCertificateRefs` element.
   - The `CompleteRevocationRefs` or `xadesenv111:CompleteCertificateRefs` element.
   - The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs` element if this property is present.
   - The `AttributeRevocationRefs` element if this property is present.

Applications shall build URI attributes following the rules stated in clause 6.1.4.3.1.

Generating applications shall build digest computation input as indicated below:

1) Initialize the final octet stream as an empty octet stream.

2) Take the `ds:SignatureValue` element and its content. Canonicalize it it as specified in clause 5.4, and put the result in the final octet stream.

3) Take each unsigned property listed above in the order they have been listed above (this order shall be the same as the order the `Include` elements appear in the property). For each one extract comment nodes, canonicalize it as specified in clause 5.4, and concatenate the resulting octet string to the final octet stream.

Below follows the list of the data objects that contribute to the digest computation. Super index ^{e} means that this property is referenced using explicit mechanism, i.e. that the property contains an `Include` element that references it:

```
(ds:SignatureValue, SignatureTimeStamp^e+, (CompleteCertificateRefs^e,|
xadesenv111:CompleteCertificateRefs^e), CompleteRevocationRefs^e,
(AttributeCertificateRefs^e | xadesenv111:AttributeCertificateRefs^e)?,
AttributeRevocationRefs^e?).
```

A.1.5.2 The `RefsOnlyTimeStamp` element

The `RefsOnlyTimeStamp` property is an optional unsigned property qualifying the signature. Clause B.2.1 proposes a XAdES form that can incorporate one or more `RefsOnlyTimeStamp` elements.

Below follows the schema definition for this element.

```
<xsd:element name="RefsOnlyTimeStamp" type="XAdESTimeStampType"/>
```

This property contains a time-stamp token that covers the following data objects: `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, `CompleteRevocationRefs`, and when present, `AttributeCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, and `AttributeRevocationRefs`.

Depending whether all the aforementioned time-stamped unsigned properties and the `RefsOnlyTimeStamp` property itself have the same parent or not, its contents may be different. Details are given in clauses below.

A.1.5.2.1 Not distributed case

When `RefsOnlyTimeStamp` and all the unsigned properties covered by its time-stamp token have the same parent, this property uses the Implicit mechanism. The input to the computation of the digest value shall be the result of taking those of the unsigned properties listed below that appear before the `RefsOnlyTimeStamp` in their order of appearance within the `UnsignedSignatureProperties` element, canonicalize each one as specified in clause 5.4, and concatenate the resulting octet streams:

- The `CompleteCertificateRefs` element.
- The `CompleteRevocationRefs` or `xadesenv111:CompleteCertificateRefs` element.
- The `AttributeCertificateRefs` or `xadesenv111:CompleteCertificateRefs` element if this property is present.
- The `AttributeRevocationRefs` element if this property is present.

Below follows the list of data objects that contribute to the digest computation:

```
([(CompleteCertificateRefs | xadesenv111:CompleteCertificateRefs),
CompleteRevocationRefs, (AttributeCertificateRefs|
xadesenv111:CompleteCertificateRefs)?, AttributeRevocationRefs?]).
```


A.1.5.2.2 Distributed case

When `RefsOnlyTimeStamp` and some of the unsigned properties covered by its time-stamp token DO NOT have the same parent, applications shall build this property generating one `Include` element per each unsigned property that shall be covered by the time-stamp token in the order they appear listed below:

- The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` element.
- The `CompleteRevocationRefs` element.
- The `AttributeCertificateRefs` or `xadesenv111:CompleteCertificateRefs` element if this property is present.
- The `AttributeRevocationRefs` element if this property is present.

Applications shall build URI attributes following the rules stated in clause 6.1.4.3.1.

Generating applications shall build digest computation input as indicated below:

1) Initialize the final octet stream as an empty octet stream.
2) Take each unsigned property listed above in the order they have been listed above (this order shall be the same as the order the `Include` elements appear in the property). For each one extract comment nodes, canonicalize it as specified in clause 5.4, and concatenate the resulting octet stream to the final octet stream.

Below follows the list -in order- of the data objects that contribute to the digest computation. Superindex [e] means that this property is referenced using explicit mechanism, i.e. that the property contains a Include element that references it:

((CompleteCertificateRefs[e], | xadesenv111:CompleteCertificateRefs[e]),
CompleteRevocationRefs[e], (AttributeCertificateRefs[e] |
xadesenv111:CompleteCertificateRefs[e])?, AttributeRevocationRefs[e]?).

A.2 Obsoleted qualifying properties

A.2.1 The ArchiveTimeStamp element

The ArchiveTimeStamp element is an optional unsigned property qualifying the signature. This is an obsoleted property. Below follows the schema definition for this element.

```
<xsd:element name="ArchiveTimeStamp" type="XAdESTimeStampType"/>
```

Should the XAdES signature incorporate a CounterSignature unsigned property, implementers should ensure that all the required material for conducting the validation of the counter-signature is incorporated to the XAdES signature before generating the first ArchiveTimeStamp property. This may be done within the counter-signature itself or within the containers available within the counter-signed XAdES signature.

Should a CounterSignature unsigned property be time-stamped by the ArchiveTimeStamp, any ulterior change of their contents (by addition of unsigned properties if the counter-signature is a XAdES signature, for instance) would make the validation of the ArchiveTimeStamp, and in consequence of the countersigned XAdES signature, fail. Implementers should, in consequence, not change the contents of the CounterSignature property once it has been time-stamped by the ArchiveTimeStamp. Implementors may, in these circumstances, to make use of the detached counter-signature mechanism specified in clause 6.2.7.1.

In addition it has to be noted that the present document allows to counter-sign a previously time-stamped countersignature with another CounterSignature property added to the embedding XAdES signature after the time-stamp container.

Depending whether all the unsigned properties covered by the time-stamp token and the ArchiveTimeStamp property itself have the same parent or not, its contents may be different. Details are given in clauses below.

A.2.1.1 Not distributed case

When ArchiveTimeStamp and all the unsigned properties covered by its time-stamp token have the same parent, this property uses the Implicit mechanism for all the time-stamped data objects. The input to the computation of the digest value shall be built as follows:

1) Initialize the final octet stream as an empty octet stream.

2) Take all the ds:Reference elements in their order of appearance within ds:SignedInfo referencing whatever the signer wants to sign including the SignedProperties element. Process each one as indicated below:

   - Process the retrieved ds:Reference element according to the reference processing model of XMLDSIG.
   - If the result is a XML node set, canonicalize it as specified in clause 5.4.
   - Concatenate the resulting octets to the final octet stream.

3) Take the following XMLDSIG elements in the order they are listed below, canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream:

- The `ds:SignedInfo` element.
- The `ds:SignatureValue` element.
- The `ds:KeyInfo` element, if present.

4) Take any of the following unsigned signature properties that appear before the current `ArchiveTimeStamp` in the order they appear within the `UnsignedSignatureProperties`, canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream:

- Any present `SignatureTimeStamp` property.
- Any present `CounterSignature` property.
- The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, and `CompleteRevocationRefs` properties if present.
- The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs`, and `AttributeRevocationRefs` properties if present.
- Any present `SigAndRefsTimeStamp` and `RefsOnlyTimeStamp` property.
- The `CertificateValues` property. This property shall be added if it is not already present AND the `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic signature.
- The `RevocationValues` property. This property shall be added if it is not already present AND the `ds:KeyInfo` element does not contain the full set of revocation data used to validate the electronic signature.
- The `AttrAuthoritiesCertValues` property. This property shall be added if not already present and the following conditions are true: there exist an attribute certificate in the signature AND a number of certificates that have been used in its validation do not appear in `CertificateValues` or `ds:KeyInfo`.
- The `AttributeRevocationValues` property. This property shall be added if not already present and there the following conditions are true: there exist an attribute certificate AND some revocation data that have been used in its validation do not appear in `RevocationValues` or `ds:KeyInfo`.
- Any previous `ArchiveTimestamp` property.

5) Take any `ds:Object` element in the signature that is not referenced by any `ds:Reference` within `ds:SignedInfo`, except that one containing the QualifyingProperties element. Canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream.

A.2.1.2 Distributed case

When `ArchiveTimeStamp` and some of the unsigned properties covered by its time-stamp token DO NOT have the same parent, applications shall use the explicit (Include) mechanism only for referencing the unsigned properties. Applications shall build this property generating one `Include` element per each unsigned property that must be covered by the time-stamp token in the order they appear listed below:

1) Any present `SignatureTimeStamp` property.
2) Any present `CounterSignature` property.
3) The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` property if present.
4) The `CompleteRevocationRefs` property if present.

5) The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs` property if present.

6) The `AttributeRevocationRefs` property if present.

7) Any present `SigAndRefsTimeStamp` property.

8) Any present `RefsOnlyTimeStamp` property.

9) The `CertificateValues` property. This property shall be added if it is not already present AND the `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic signature.

10) `RevocationValues` property. This property shall be added if it is not already present AND the `ds:KeyInfo` element does not contain the full set of revocation data used to validate the electronic signature.

11) The `AttrAuthoritiesCertValues` property. This property shall be added, if not already present, when the conditions mentioned in the non distributed case are met.

12) The `AttributeRevocationValues` property. This property shall be added, if not already present, when the conditions mentioned in the non distributed case are met.

13) Any previously present `ArchiveTimestamp` property.

No `xades:Include` elements are generated for any other XMLDSIG element present in the signature, although they are actually time-stamped as they contribute to the generation of the message imprint computation input.

Generating applications MUST build digest computation input as for the Implicit case (clause A2.1.1) substituting step 4) by the one specified below:

4) Take the following unsigned signature properties in the order they are listed below, extract comment nodes, canonicalize each one as specified in clause 5.4 and concatenate each resulting octet stream to the final octet stream. The order of appearance of their referencing `Include` elements in the `ArchiveTimeStamp` property shall be identical to the order that unsigned properties are actually processed:

- Any present `SignatureTimeStamp` property.
- Any present `CounterSignature` property.
- The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` property if present.
- The `CompleteRevocationRefs` property if present.
- The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs` property if present.
- The `AttributeRevocationRefs` property if present.
- Any present `SigAndRefsTimeStamp` property.
- Any present `RefsOnlyTimeStamp` property.
- The `CertificateValues` property. This property shall be added if it is not already present AND the `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic signature.
- The `RevocationValues` property. This property shall be added if it is not already present AND the `ds:KeyInfo` element does not contain the full set of revocation data used to validate the electronic signature.
- The `AttrAuthoritiesCertValues` property. This property shall be added, if not already present, when the conditions mentioned in the non distributed case are met.

- The `AttributeRevocationValues` property. This property shall be added, if not already present, when the conditions mentioned in the non distributed case are met.

- Any previous `ArchiveTimestamp` property.

Annex <B> (normative): XAdES signature forms with references

B.1 Electronic signature with complete validation data references (XAdES-C)

XML Advanced Electronic Signature with Complete validation data references (XAdES-C) in accordance with the present document adds to the XAdES-T the `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` and `CompleteRevocationRefs` unsigned properties as defined by the present document. If attribute certificates appear in the signature, then XAdES-C also incorporates the `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs`,and the `AttributeRevocationRefs` elements.

Below follows the structure for XAdES-C built by direct incorporation of properties on a XAdES-T containing the `SignatureTimeStamp` signed property. A XAdES-C form based on time-marks MAY exist without such element.

Henceforth [Ref. to certificates] denotes the choice between `CompleteCertificateRefs` and `xadesenv111:CompleteCertificateRefs`, and [Ref. to AttrAuth. certs.] denotes the choice between `AttributeCertificateRefs` and `xadesenv111:AttributeCertificateRefs`


```
                                XMLDISG
                                   |
<ds:Signature ID?>- - - - - - - - +- - - - - - +-+-+
  <ds:SignedInfo>                  |            | | |
    <ds:CanonicalizationMethod/>   |            | | |
    <ds:SignatureMethod/>          |            | | |
   (<ds:Reference URI? >           |            | | |
      (<ds:Transforms>)?           |            | | |
      <ds:DigestMethod/>           |            | | |
      <ds:DigestValue/>            |            | | |
    </ds:Reference>)+              |            | | |
  </ds:SignedInfo>                 |            | | |
  <ds:SignatureValue/>             |            | | |
  (<ds:KeyInfo>)? - - - - - - - - +            | | |
                                               | | |
  <ds:Object>                                  | | |
                                               | | |
    <QualifyingProperties>                     | | |
                                               | | |
      <SignedProperties>?                      | | |
                                               | | |
        <SignedSignatureProperties>            | | |
        (SigningTime)?                         | | |
        ([Ref. to signing certificate])?       | | |
        (SignaturePolicyIdentifier)?           | | |
        (SignatureProductionPlace)?            | | |
        ([Signer Attrs.])?                     | | |
      </SignedSignatureProperties>             | | |
                                               | | |
      <SignedDataObjectProperties>             | | |
        (DataObjectFormat)*                    | | |
        (CommitmentTypeIndication)*            | | |
        (AllDataObjectsTimeStamp)*             | | |
        (IndividualDataObjectsTimeStamp)*      | | |
      </SignedDataObjectProperties>            | | |
                                               | | |
      </SignedProperties>                      | | |
                                               | | |
```

```
      <UnsignedProperties>                          | | |
                                                    | | |
        <UnsignedSignatureProperties>               | | |
          (CounterSignature)*- - - - - - - - - - + | | |
          (SignatureTimeStamp)+- - - - - - - - - - + |
          ([Ref. to certificates])                  |
          (CompleteRevocationRefs)                  |
          ([Ref. to AttrAuth. certs.])?             |
          (AttributeRevocationRefs)?                |
        </UnsignedSignatureProperties>- - - - -+-+  |
                                                    | | |
       </UnsignedProperties>                        | | |
                                                    | | |
    </QualifyingProperties>                         | | |
                                                    | | |
  </ds:Object>                                      | | |
                                                    | | |
</ds:Signature>- - -  - - - - - - - - - - - - - -+-+-+
                                                    | | |
                                  XAdES-BES(-EPES)  | |
                                                    | |
                                           XAdES-T  |
                                                    |
                                              XAdES-C
```

The XAdES-C form is the XAdES instantiation of the AdES-C form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause C.1.

NOTE 1: When the signer does not provide the XAdES-C, the verifier may create the XAdES-C when the required components of revocation and validation data become available. This may require a grace period.

B.2 Extended signatures with time forms (XAdES-X)

Extended signatures with time indication forms (XAdES-X) in accordance with the present document build on signatures containing `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` and `CompleteRevocationRefs` properties, by adding one or more unsigned properties encapsulating time-stamp tokens.

Depending of what is time-stamped, there are two different types of XAdES-X signatures, namely, XAdES-X type 1 and XAdES-X type 2. Time-stamps in both types cover, among other elements, `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, and `CompleteRevocationRefs` properties. Time-stamps provide an integrity and trusted time protection over everything that is time-stamped. They protect the referenced certificates, CRLs and OCSP responses in case of a later compromise of a CA key, CRL key or OCSP issuer key.

XAdES-X type 2 is built by adding one or more `RefsOnlyTimeStamp` properties each containing one time-stamp obtained from different TSAs. These time-stamps are computed on the `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, and `CompleteRevocationRefs` properties

B.2.1 EXtended Electronic Signature with Time Type 1 (XAdES-X Type 1)

XAdES-X type 1 is built by adding one or more `SigAndRefsTimeStamp` properties each containing one time-stamp obtained from different TSAs. These time-stamps are computed on the `SignatureValue` element, `SignatureTimeStamp` if present, `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, and `CompleteRevocationRefs` properties. Below follows the structure of this form.

```
                                  XMLDISG
                                    |
<ds:Signature ID?>- - - - - - - - +- - - - - - +-+-+-+
  <ds:SignedInfo>                   |           | | | |
    <ds:CanonicalizationMethod/>    |           | | | |
    <ds:SignatureMethod/>           |           | | | |
```

The XAdES-X-Type 1 form is the XAdES instantiation of the AdES-X-Type 1 form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause C.2.1.

B.2.2 EXtended Electronic Signature with Time Type 2 (XAdES-X Type 2)

XAdES-X type 2 is built by adding one or more `RefsOnlyTimeStamp` properties each containing one time-stamp obtained from different TSAs. These time-stamps are computed on the `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, and `CompleteRevocationRefs` properties.

The XAdES-X-Type 2 form is the XAdES instantiation of the AdES-X-Type 2 form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause C.2.2.

B.3 Extended long electronic signatures with time forms (XAdES-X-L type 1 or 2)

Extended long electronic signatures with time (XAdES-X-L) forms in accordance with the present document build up on XAdES-X types 1 or 2 by adding the `CertificateValues` and `RevocationValues` unsigned properties.

The structure for the most complete XAdES-X-L type 1, built on the most complete XAdES-X signature, is shown below.

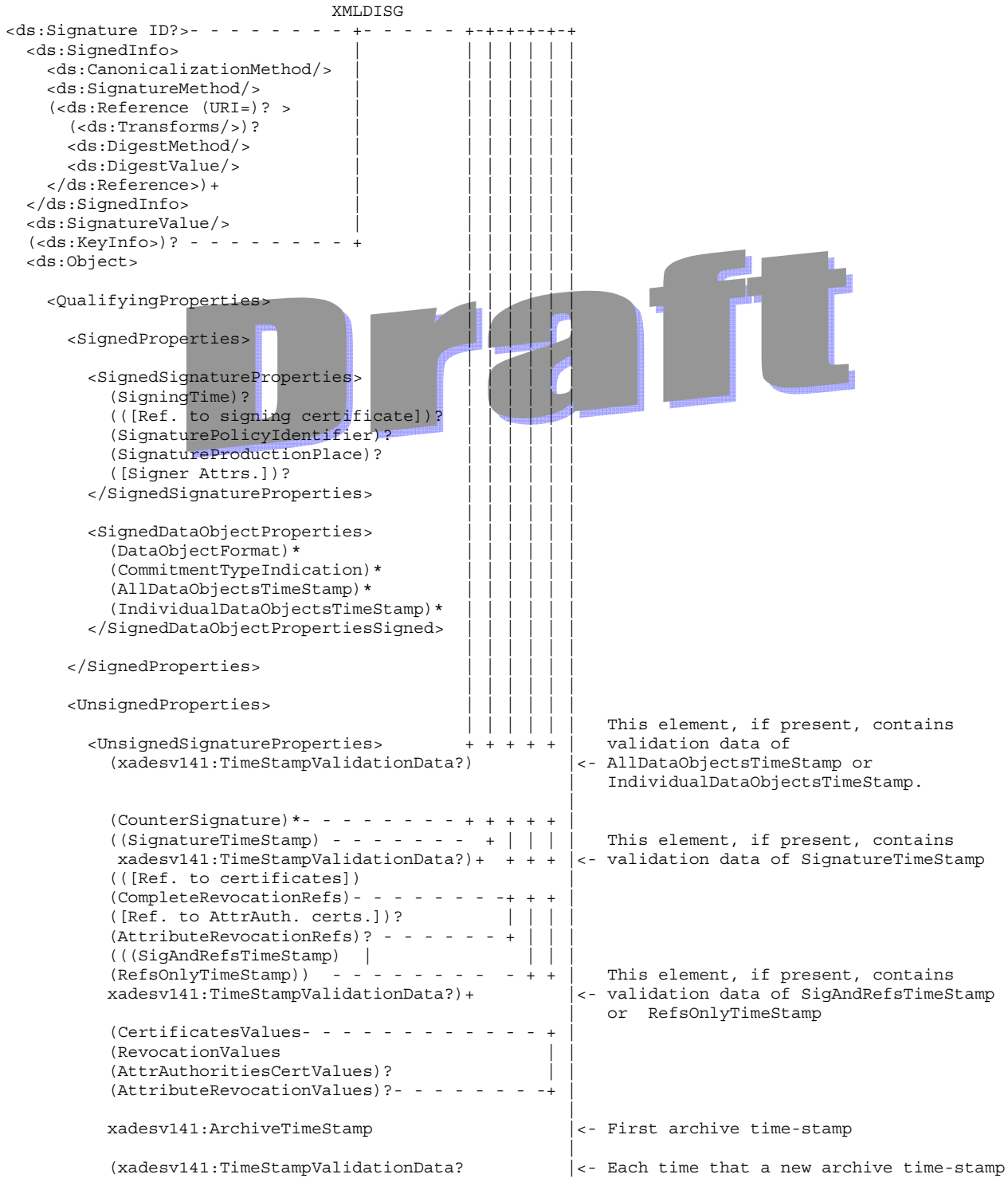
The structure for the most complete XAdES-X-L type 2, built on the most complete XAdES-X signature, is shown below.

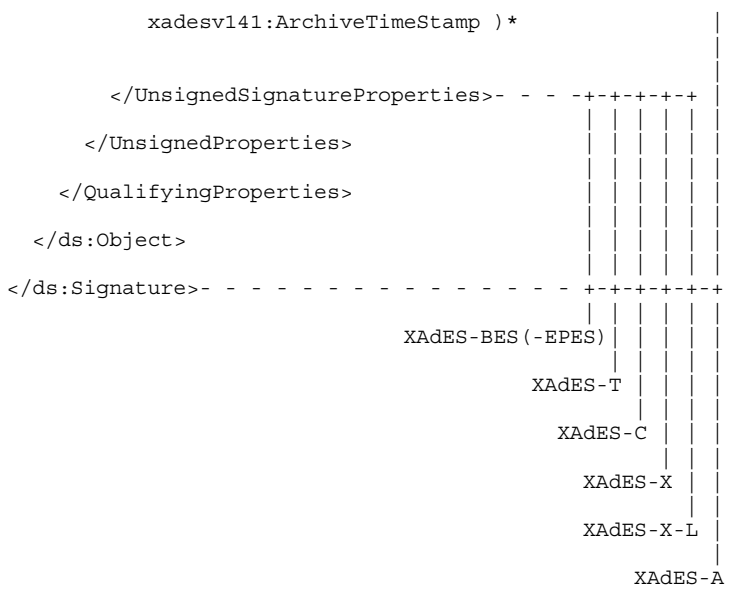
The XAdES-X-L-Type 1 and XAdES-X-L-Type 2 forms are the XAdES instantiations of the AdES-X-L-Type 1 and AdES-X-L-Type 2 forms specified within ETSI EN 319 102.

Conformance requirements for these forms of XAdES signatures are specified in clause C.3.1 and C.3.2 respectively.

B.4 Archival Electronic Signature complete

Below follows the structure of a XAdES-A built on a XAdES-X-L, as an example of the most complete archival form.





Annex <C> (normative): Conformance requirements for additional Electronic Signature Forms.

In addition to the base conformance levels defined in clause 7, the present document defines the following additional core conformance levels:

- Electronic signatures with Complete validation data references (XAdES-C)
- EXtended electronic signature with time forms, Type 1 (XAdES-X Type 1)
- EXtended signature with time forms, Type 2 (XAdES-X Type 2)
- EXtended Long signature (XAdES-X-L)
- EXtended Long signatures with time forms, Type 1 (XAdES-X-L Type 1)
- EXtended Long signatures with time forms, Type 2 (XAdES-X-L Type 2)

An implementation claiming to be conformant to one of them shall implement the corresponding.

C.1 Electronic signatures with Complete validation data references (XAdES-C)

A signature claiming conformance to XAdES-C level shall be built upon a signature compliant with XAdES-T conformance level. In addition, it:

- shall incorporate (directly or indirectly) one instance of `xades:CompleteCertificateReferences` unsigned property.
- shall incorporate (directly or indirectly) one instance of `xades:CompleteRevocationRefs` unsigned property.
- may incorporate (directly or indirectly) one instance of `xades:AttributeCertificateReferences` unsigned property.

- may incorporate (directly or indirectly) one instance of `xades:AttributeRevocationRefs` unsigned property.

C.2 EXtended signatures with time forms (XAdES-X)

The extended signatures with time forms build on XAdES-C and protect the certificate and revocation references with a time-stamp.

C.2.1 XAdES-X Type 1

A signature claiming conformance to XAdES-X Type 1 level shall be built upon a signature compliant with XAdES-C conformance level. In addition, it shall directly or indirectly incorporate:

- one or more instance of `xades:SigAndRefsTimeStamp` unsigned property.

C.2.2 XAdES-X Type 2

A signature claiming conformance to XAdES-X Type 2 level shall be built upon a signature compliant with XAdES-C conformance level. In addition, it shall directly or indirectly incorporate:

- one or more instance of `xades:RefsOnlyTimeStamp` unsigned property.

C.3 EXtended Long signatures with time forms (XAdES-X-L)

C.3. 1 EXtended Long signatures with time forms, Type 1 (XAdES-X-L Type 1)

A signature claiming conformance to XAdES-X-L Type 1 level shall be built upon a signature having XAdES-X Type 1 level. In addition it shall incorporate the full set of certificate values and revocation values required for validating the signature. In consequence it:

- may directly or indirectly incorporate one instance of `xades:CertificateValues` property. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:RevocationValues` property. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:AttrAuthoritiesCertValues`. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:AttributeRevocationValues` property. See clause XXX for details.

C.3.2.1 EXtended Long signatures with time forms, Type 2 (XAdES-X-L Type 2)

A signature claiming conformance to XAdES-X-L Type 2 level shall be built upon a signature having XAdES-X Type 2 level. In addition it shall incorporate the full set of certificate values and revocation values required for validating the signature. In consequence it:

- may directly or indirectly incorporate one instance of `xades:CertificateValues` property. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:RevocationValues` property. See clause XXX for details.

- may directly or indirectly incorporate one instance of `xades:AttrAuthoritiesCertValues`. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:AttributeRevocationValues` property. See clause XXX for details.

Annex <D> (informative): General Description

D.1 The `SigningTime` element

By adding this signed property to the XAdES signature, the signatory claims to have generated this signature at the indicated time. Readers are reminded that this is not a time indication coming from a trusted source like a time-stamp token.

D.2 References to the signing certificate

In many real life environments users will be able to get from different CAs or even from the same CA, different certificates containing the same public key for different names. The prime advantage is that a user can use the same private key for different purposes. Multiple use of the private key is an advantage when a smart card is used to protect the private key, since the storage of a smart card is always limited. When several CAs are involved, each different certificate may contain a different identity, e.g. as a national or as an employee from a company. Thus when a private key is used for various purposes, the certificate is needed to clarify the context in which the private key was used when generating the signature. Where there is the possibility of multiple uses of private keys it is necessary for the signer to indicate to the verifier the precise certificate to be used.

Many current schemes simply add the certificate after the signed data and thus are subject to various substitution attacks. An example of a substitution attack is a "bad" CA that would issue a certificate to someone with the public key of someone else. If the certificate from the signer was simply appended to the signature and thus not protected by the signature, any one could substitute one certificate by another and the message would appear to be signed by some one else. In order to counter this kind of attack, the identifier of the certificate has to be protected by the digital signature from the signer.

The `SigningCertificate` and `xadesenv111:SigningCertificate` properties are designed to prevent the simple substitution of the certificate.

D.3 The `CommitmentTypeIndication` element

The commitment type can be indicated in the electronic signature either:

- explicitly using a commitment type indication in the electronic signature;
- implicitly or explicitly from the semantics of the signed data object.

If the indicated commitment type is explicit by means of a commitment type indication in the electronic signature, acceptance of a verified signature implies acceptance of the semantics of that commitment type. The semantics of explicit commitment types indications are specified either as part of the signature policy or may be registered for generic use across multiple policies.

The commitment type may be:

- defined as part of the signature policy, in which case the commitment type has precise semantics that is defined as part of the signature policy;
- a registered type, in which case the commitment type has precise semantics defined by registration, under the rules of the registration authority. Such a registration authority may be a trading association or a legislative authority.

The definition of a commitment type includes an identifier (URI) and an optional sequence of qualifiers, which may provide additional information (for instance information about the context, be it contractual/legal/application specific).

If an electronic signature does not contain a recognized commitment type then the semantics of the electronic signature is dependent on the data object being signed and the context in which it is being used. How commitment is indicated using the semantics of the data object being signed is outside the scope of the present document.

D.4 The `DataObjectFormat` element

When presenting signed data to a human user it may be important that there is no ambiguity as to the presentation of the signed data object to the relying party. In order for the appropriate representation (text, sound or video) to be selected by the relying party a content hint may be indicated by the signer. If a relying party system does not use the format specified to present the data object to the relying party, the electronic signature may not be valid. Such behaviour may have been established by the signature policy, for instance.

The `DataObjectFormat` element provides information that describes the format of the signed data object. The presence of this element is indeed beneficial when the signed data is to be presented to human users on validation if the presentation format is not implicit within the data that has been signed.

D.5 The `SignatureProductionPlace` element

In some transactions the purported place where the signer was at the time of signature creation may need to be indicated. In order to provide this information the `SignatureProductionPlace` signed property is defined. It specifies an address associated with the signer at a particular geographical (e.g. city) location.

D.6 The `SignerRole` element

While the name of the signer is important, the position of the signer within a company or an organization is of paramount importance as well. Some information (i.e. a contract) may only be valid if signed by a user in a particular role, e.g. a Sales Director. In many cases, who the sales Director really is, is not that important, but being sure that the signer is empowered by his company to be the Sales Director is fundamental.

The present document defines two different ways for providing this feature:

- by placing a *claimed* signer attribute / role;
- by placing a an assertion that includes signer attribute / role, signed by an entity which does not satisfy the requirements needed to be considered an Attribute Authority or
- by placing an attribute certificate issued by an Attribute Authority.

NOTE: Another possible approach would have been to use additional attributes containing the attribute or roles name(s) in the signer's identity certificate. However, it was decided not to follow this approach as it significantly complicates the management of certificates. For example, by using separate certificates for the signer's identity and roles means new identity keys need not be issued if a user's role changes.

D.6.1 Claimed signer attribute/role

The signer may be trusted to state his own attribute or role without any third party certifying this claim; in which case, the claimed role can be added to the signature as a signed attribute.

D.6.2 Certified signer attribute/role

A certified signer attribute / role is certified by a trusted Attribute Authority (AA). The certification is done either by the AA issuing an Attribute.

Unlike public key certificates that bind an identifier to a public key, Attribute Certificates bind the identifier of a certificate to some attributes, like a role. An Attribute Certificate is NOT issued by a CA but by an Attribute Authority (AA). The Attribute Authority, in most cases, might be under the control of an organization or a company that is best placed to know which attributes are relevant for which individual. The Attribute Authority may use or point to public key certificates issued by any CA, provided that the appropriate trust may be placed in that CA. Attribute Certificates may have various periods of validity. That period may be quite short, e.g. one day. While this requires that a new Attribute Certificate be obtained every day, valid for that day, this can be advantageous since revocation of such certificates may not be needed. When signing, the signer will have to specify which Attribute Certificate it selects. In order to do so, the Attribute Certificate will have to be included in the signed data in order to be protected by the digital signature from the signer. In order to unambiguously identify the attribute certificate(s) to be used for the verification of the signature, an identifier of the attribute certificate(s) from the signer is part of the signed data

D.7 Multiple signatures and countersignatures

Some electronic documents have full effect only if they bear more than one signature. This is generally the case, for example, when a contract is signed between two parties. The ordering of the signatures may or may not be important, i.e. one may or may not need to be applied before the other. This allows establishing two basic categories for multiple signatures:

- independent signatures;
- countersignatures.

Independent signatures are parallel signatures where the ordering of the signatures is not important. The computation of these signatures is performed on exactly the same input but using different private keys.

Countersignatures are signatures that are applied one after the other and are used where the order the signatures are applied is important. In these situations the first signature signs the signed data object. Each additional signature may sign in turn the latest previously generated signature, or all the previously generated signatures and the signed document.

The referencing mechanism present in XMLDSIG gives full support to countersignatures. Using them, the countersignatures may be placed and kept in different ways: they may be embedded one within the other, or they may be detached from the rest as long as their corresponding `<ds:Reference>` elements ensure that each signature actually signs the previously generated signature (or all the previously generated signatures and the signed document if this is the requirement). While XMLDSIG supports these features, it does not propose any standard format for countersignatures as it considers this topic being out of its scope.

The present document defines a new URI value, which, when assigned as value of the `Type` attribute of a `ds:Reference` element, denotes that the enclosing XAdES signature is in fact, a countersignature of another signature.

In addition the present document defines with the `CounterSignature` property a standard way of managing countersignatures that:

- are computed on the values of the latest previously generated signatures;
- are embedded within the signatures that they countersign so that the first electronic signature (the one computed on the data objects actually signed) contains all the additional countersignatures that have to be verified.

Independent signatures do not appear as `CounterSignature` properties of another independent one.

This proposal does not, of course, satisfy all the potential requirements that real situations may pose in terms of relationships among electronic signatures and documents. This would require more complexity, which is out of scope of XAdES. Readers are reminded, though, that ASiC containers specify a standard way of packaging together documents and their detached signatures (be them independent or countersignatures).

D.8 Time-stamps on the signed data objects

The signed properties `IndividualDataObjectsTimeStamp` and `AllDataObjectsTimeStamp` provide proof of the existence of the signed data object, at the time indicated by the encapsulated time-stamp token.

Using them, a trusted secure time may be obtained before the data objects are signed and included under the digital signature. This solution requires an online connection to a trusted time-stamping service before generating the signature and may not represent the precise signing time, since it can be obtained in advance. However, these properties may be used by the signer to prove that the signed data objecst existed before the date included in the time-stamp token (see clause 6.2.8).

D.9 The SignaturePolicyIdentifier element

The signature policy is a set of rules for the creation and validation of an electronic signature, under which the signature can be determined to be valid. A given legal/contractual context may recognize a particular signature policy as meeting its requirements.

When a comprehensive signature policy used by the verifier is either explicitly indicated by the signer or implied by the data being signed, then a consistent result can be obtained when validating an electronic signature. When the signature policy being used by the verifier is neither indicated by the signer nor can be derived from other data, or the signature policy is incomplete then verifiers, including arbitrators, may obtain different results when validating an electronic signature. Therefore, comprehensive signature policies that ensure consistency of signature validation are valuable for both the signers and verifiers.

For assessing whether a signature policy meets the requirements of the legal and contractual context in which it is being applied, a human readable form of such a policy is required. Additionally, if the parts of the signature policy that specify the rules for the creation and validation of the electronic signature are expressed in a form that a computer may understand and process, this will facilitate the automatic processing of an electronic signature.

If no signature policy is identified then the signature may be assumed to have been generated/verified without any policy constraints, and hence may be given no specific legal or contractual significance through the context of a signature policy.

The present document specifies two unambiguous ways for identifying the signature policy that a signature follows:

- The electronic signature can contain an explicit and unambiguous identifier of a signature policy together with a hash value of the signature policy, so it can be verified that the policy selected by the signer is the one being used by the verifier. An explicit signature policy has a globally unique reference, which, in this way, is bound to an electronic signature by the signer as part of the signature calculation. In these cases, for a given explicit signature policy there shall be one definitive form that has a unique binary encoded value. A signature policy identified in this way may be qualified by additional information.

- Alternatively, the electronic signature can avoid the inclusion of the aforementioned identifier and hash value. This will be possible when the signature policy can be unambiguously derived from the semantics of the type of data object(s) being signed, and some other information, e.g. national laws or private contractual agreements, that mention that a given signature policy has to be used for this type of data content. In such cases, the signature will contain a specific empty element indicating that this implied way to identify the signature policy is used instead the identifier and hash value.

D.10 The `SignatureTimeStamp` element

An important property for long standing signatures is that a signature, having been found once to be valid, will continue to be so months or years later. A signer, verifier or both may be required to provide on request, proof that an electronic signature was created or validated during the validity period of all the certificates that make up the certificate path. In this case, the signer, verifier or both will also be required to provide proof that all the user and CA certificates used were not revoked when the signature was created or validated.

It would be quite unacceptable to consider a signature as invalid even if the keys or certificates were only compromised later. Thus there is a need to be able to demonstrate that the signature key was valid around the time that the signature was created to provide long term evidence of the validity of a signature. Time-stamping by a Time-Stamping Authority (TSA) can provide such evidence.

Time-stamping an electronic signature before the revocation of the signer's private key and before the end of the validity of the certificate provides evidence that the signature has been created while the certificate was valid and before it was revoked.

If a recipient wants to keep the result of the validation of an electronic signature valid, he will have to ensure that he has obtained a valid time-stamp for it, before that key (and any key involved in the validation) is revoked. The sooner the time-stamp is obtained after the signing time, the better. It is important to note that signatures may be generated "off-line" and time-stamped at a later time by anyone, for example by the signer or any recipient interested in the signature. The time-stamp can thus be provided by the signer together with the signed data object, or obtained by the recipient following receipt of the signed data object.

The validation mandated by the signature policy can specify a maximum acceptable time difference which is allowed between the time indicated in the `SigningTime` element and the time indicated by the `SignatureTimeStamp` element.

D.11 Elements containing references to validation data

When dealing with long term electronic signatures, all the data used in the validation (namely, certificate path and revocation information) of such signatures have to be stored and conveniently time-stamped for arbitration purposes. Similar considerations apply to attribute certificates if they appear within the signature.

In some environments, it can be convenient to add these data to the electronic signature (as unsigned properties) for archival purposes (see archival electronic signatures XAdES-A).

Systems implementing the present document may alternatively consider to archive validation data outside the XAdES e.g. to prevent redundant storage and to reduce the size of the signatures. In such cases each electronic signature incorporates references to all these data within the signature, reducing accordingly the size of the stored electronic signature. This format builds up taking XAdES-T signature by incorporating additional data required for validation:

- the sequence of references to the full set of CA certificates that have been used to validate the electronic signature up to (but not including) the signer's certificate;
- the sequence of references to the full set of revocation data that have been used in the validation of the signer and CA certificates;
- the references to the full set of Attribute Authorities and the CA certificates that have been used to validate the attribute certificate(s), if present;
- the references to the full set of revocation data that have been used in the validation of the attribute certificate(s), if present.

The full set of references to the revocation data that have been used in the validation of the signer and CAs certificates, provide means to retrieve the actual revocation data archived elsewhere in case of dispute and, in this way, to illustrate that the verifier has taken due diligence of the available revocation information.

Currently two major types of revocation data are managed in most of the systems, namely CRLs and responses of on-line certificate status servers, obtained through protocols designed for these purposes, like OCSP protocol. In consequence, means are provided in the present document for referencing both types of revocation data.

D.12 Time-stamps on references to validation data

Electronic signatures incorporating time-stamps on validation data references are needed when there is a requirement to safeguard against the possibility of a CA key in the certificate chain ever being compromised. A verifier may be required to provide, on request, proof that the certification path and the revocation information used at the time of the signature were valid, even in the case where one of the issuing keys or OCSP responder keys is later compromised.

The present document defines two ways of using time-stamps to protect against this compromise:

- Time-stamp the sequence formed by the digital signature (`ds:SignatureValue` element), the `SignatureTimeStamp` element when present in the XAdES-T form, the certification path references, the Attribute Authorities and CA certificate references and the revocation data references (for both the certificates in the certification path and in the list of Attribute Authorities certificate.

- Time-stamp only the references.

For both ways signer, verifier or both may request, obtain and add the time-stamp to the electronic signature.

When an OCSP response is used, it is necessary to time-stamp in particular that response in the case the key from the responder would be compromised. Since the information contained in the OCSP response is user specific and time specific, an individual time-stamp is needed for every signature received. Instead of placing the time-stamp only over the certification path references and the revocation information references, which include the OCSP response, the time-stamp is placed on the digital signature (`ds:SignatureValue` element), the signature time-stamp(s) present in the XAdES-T form, the certification path references and the revocation status references (by adding a `+SigAndRefsTimeStamp` to the electronic signature). For the same cryptographic price, this will provide an integrity mechanism over the electronic signature. Any modification can be immediately detected. It should be noticed that other means of protecting/detecting the integrity of the electronic signature exist and could be used.

Despite the fact that this is the best scenario for using a time-stamp that covers both the references and the signature elements, it may also used in scenarios where the revocation data are CRLs. However, time-Stamping each electronic signature with the complete validation data references as defined above may not be efficient, particularly when the same set of CA certificates and CRL information is used to validate many signatures.

Time-Stamping CA certificates will stop any attacker from issuing bogus CA certificates that could be claimed to exist before the CA key was compromised. Any bogus time-stamped CA certificates will show that the certificate was created after the legitimate CA key was compromised. In the same way, time-stamping CA CRLs, will stop any attacker from issuing bogus CA CRLs which could be claimed to exist before the CA key was compromised.

Time-Stamping references to commonly used certificates and CRLs (the time-stamp token that `RefsOnlyTimeStamp` unsigned property encapsulates), can be done centrally, e.g. inside a company or by a service provider. This method reduces the amount of data the verifier has to time-stamp, for example it could reduce to just one time-stamp per day (i.e. in the case were all the signers use the same CA and the CRL applies for the whole day). As said before, the information that needs to be time-stamped is not the actual certificates and CRLs but the unambiguous references to those certificates and CRLs.

D.13 Validation data

A verifier will have to prove that the certification path was valid, at the time of the validation of the signature, up to a trust point according to the naming constraints and the certificate policy constraints from an optionally specified signature validation policy. It will be necessary to capture all the certificates from the certification path, starting with those from the signer and ending up with those of the certificate from one trusted root. Also the certificates used for validating any attribute certificate and/or time-stamp present within the electronic signature.

When dealing with long term electronic signatures, all the data used in the validation (including the certification paths of the signing certificate, and any attribute certificate and/or time-stamp present) have to be conveniently stored and time-stamped.

Several elements within a XAdES signature are entitled to contain certificate values that have been used to validate it, namely: the `ds:KeyInfo`, the `CertificateValues`, `AttrAuthoritiesCertValues` and the `CertificateValues` child of `xadesv141:TimeStampValidationData`.

The `ds:KeyInfo`, the `CertificateValues`, and the , `AttrAuthoritiesCertValues` element contain the full set of certificates that have been used to validate the electronic signature, until the first archive time-stamp is added. The `CertificateValues` child of `xadesv141:TimeStampValidationData` contains certificates required for validating the time-stamp token present in the archive time-stamp.

For dealing with long term signatures, it is also needed to store and conveniently time-stamp all the revocation data used in the validation of such signatures.

When using CRLs to get revocation information, a verifier will have to make sure that he or she gets at the time of the first validation the appropriate certificate revocation information from the signer's CA. Usually this is done as soon as possible, after the grace period, to minimize the time delay between the generation and validation of the signature. This involves checking that the signer certificate serial number is not included in the CRL. The signer, the verifier or any other third party may obtain either this CRL. If obtained by the signer, then it will be conveyed to the verifier. Additional CRLs for the CA certificates in the certificate path need to also be checked by the verifier. It may be

convenient to archive these CRLs within an archived electronic signature for ease of subsequent validation or arbitration.

When using OCSP to get revocation information, a verifier will have to make sure that she or he gets at the time of the first validation an OCSP response that contains the status "valid". Usually this is done as soon as possible after the generation of the signature, after the grace period. The signer, the verifier or any other third party may fetch this OCSP response. Since OCSP responses are transient and thus are not archived by any TSP including CA, it is the responsibility of every verifier to make sure that it is stored in a safe place.

Several elements within a XAdES signature are entitled to contain revocation data values that have been used to validate it, namely: the `ds:KeyInfo`, the `RevocationValues`, `AttributeRevocationValues` and the `RevocationValues` child of `xadesv141:TimeStampValidationData`.

The `ds:KeyInfo`, the `RevocationValues`, and the , `AttributeRevocationValues` element contain the full set of revocation data that have been used to validate the electronic signature, until the first archive time-stamp is added. The `RevocationValues` child of `xadesv141:TimeStampValidationData` contains revocation data required for validating the time-stamp token present in the archive time-stamp.

D.13 Time-stamps for archival

Advances in computing increase the probability of being able to break algorithms and compromise keys. There is therefore a requirement to be able to protect electronic signatures against this possibility.

Over a period of time weaknesses may occur in the cryptographic algorithms used to create an electronic signature (e.g. due to the time available for cryptoanalysis, or improvements in cryptoanalytical techniques). Before such weaknesses become likely, a verifier should take extra measures to maintain the validity of the electronic signature.

Several techniques could be used to achieve this goal depending on the nature of the weakened cryptography. In order to simplify matters, the present document specifies the Archive validation data technique, covering all the cases.

Archive validation data consists of the complete validation data and the complete certificate and revocation data, time-stamped together with the electronic signature. The Archive validation data is necessary if the hash function and the crypto algorithms that were used to create the signature are no longer secure. Also, if it cannot be assumed that the hash function used by the Time-Stamping Authority is secure, then nested time-stamps of Archived Electronic Signature are required.

The potential for Trusted Service Provider (TSP) key compromise should be significantly lower than for user keys, because TSP(s) are expected to use stronger cryptography and better key protection. It can be expected that new algorithms (or old ones with greater key lengths) will be used. In such a case, a sequence of time-stamps will protect against forgery. Each time-stamp needs to be affixed before either the compromise of the signing key or of the cracking of the algorithms used by the TSA. TSAs (Time-Stamping Authorities) should have long keys and/or a "good" or different algorithm.

Nested time-stamps will also protect the verifier against key compromise or cracking the algorithm on the old electronic signatures.

The process will need to be performed and iterated before the cryptographic algorithms used for generating the previous time-stamp are no longer secure. Archive validation data MAY thus bear multiple embedded time-stamps.

D.14 The `xadesenv111:RenewedDigests` element

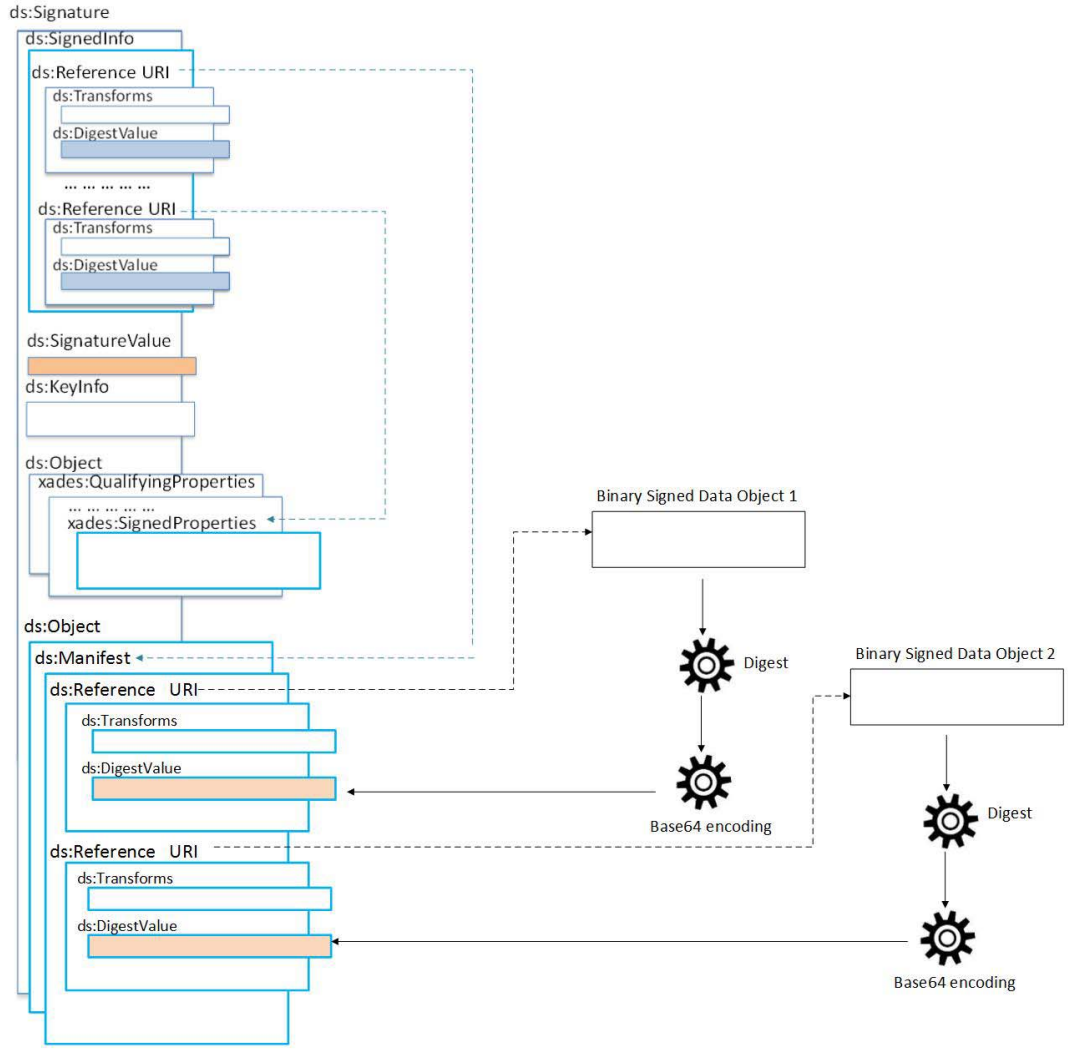
Figure 3 shows a XAdES-A signature that indirectly signs two detached binary data objects through one signed `ds:Manifest` element.

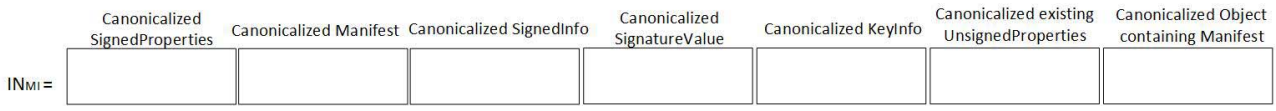
The figure also shows the input to the message imprint computation (IN_{MI}) that is used for generating the time-stamp token encapsulated within the `xadesv141:ArchiveTimeStamp` unsigned property, namely the concatenation of : the canonicalized `xades:SignedProperties` element, the canonicalized `ds:Manifest` element, the canonicalized `ds:SignedInfo` element, the canonicalized `ds:SignatureValue` element, the canonicalized `ds:KeyInfo` element, the canonicalized unsigned properties that have been incorporated just before the computation of the `xadesv141:ArchiveTimeStamp` in their order of appearance, and the the canonicalized `ds:Object` that contains the signed `ds:Manifest`. It is worth to emphasize that the contents of the two detached data objects are not

559 part of the message imprint computation input; however their digest values, present within the `ds:Manifest`, are part
560 of the message imprint computation input.

561

562

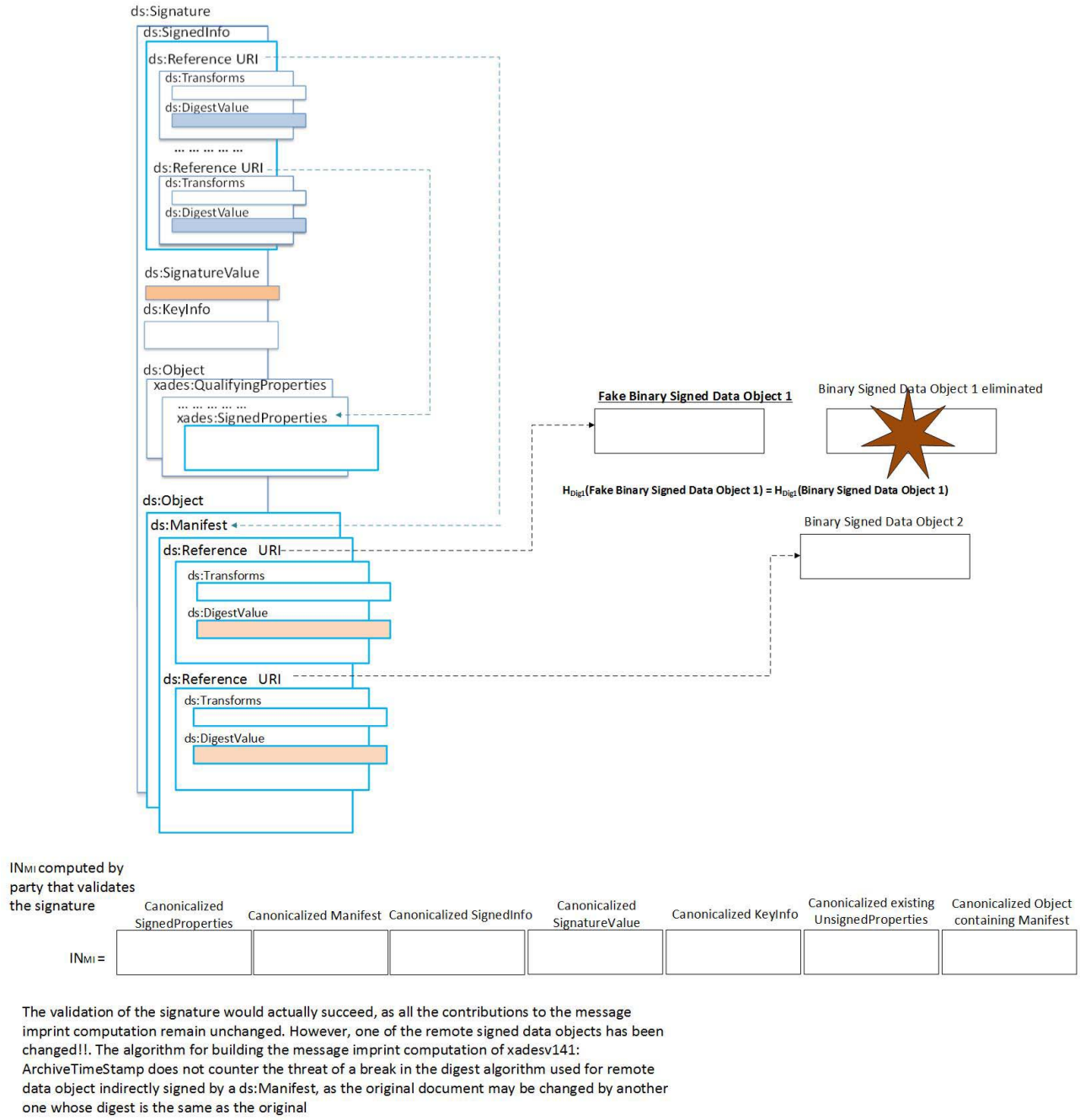




563

564 **Figure 3: A XAdES signature signing two detached data objects through one signed `ds:Manifest`**

565 Figure 4 shows a potential attack to XAdES signatures built as shown in Figure 3. Such an attack will remain unnoticed
566 if no additional measures are taken.



The validation of the signature would actually succeed, as all the contributions to the message imprint computation remain unchanged. However, one of the remote signed data objects has been changed!!. The algorithm for building the message imprint computation of xadesv141: ArchiveTimeStamp does not counter the threat of a break in the digest algorithm used for remote data object indirectly signed by a ds:Manifest, as the original document may be changed by another one whose digest is the same as the original

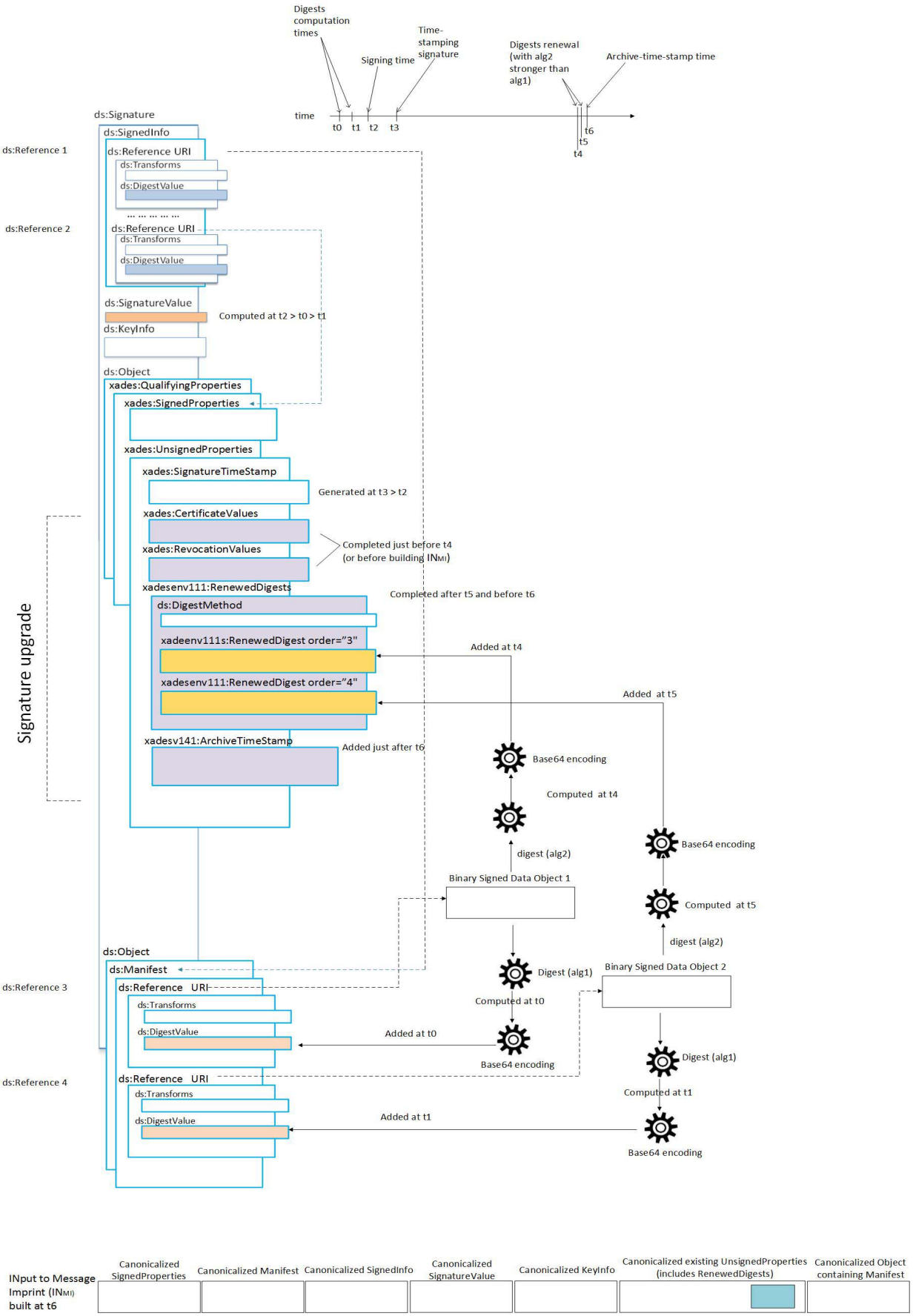
Figure 4: A successful attack: alg1 is broken and one of the data objects is substituted by another one with the same digest value

The figure shows what would happen if at a certain point in time the digest algorithm alg1 that was used for computing the digest values of the two detached data objects is broken, and if one of the aforementioned detached data objects is substituted with a fake data object whose digest value is exactly the same as the original one.

Under these circumstances, the validation of the XAdES signature would give a valid result. In first place, the check of the digest values within the signed `ds:Manifest`'s `ds:Reference` children would succeed, as the digest value of the fake data object would be the same as the digest value of the original data object. In second place the validation of the `xadesv141:ArchiveTimeStamp` would also succeed, as the actual contents of the detached data object referenced from the signed `ds:Manifest` did not contribute to the computation of the message imprint.

The `xadesenv111:RenewedDigests` unsigned property prevents that this attack succeeds, as it is shown in figure 5 below.

581



582

583 **Figure 5: Use of `xadesenv111:RenewedDigests` for countering the threat of figure 4**

If the entity in charge of upgrading the XAdES signature by incorporating a `xadesv141:ArchiveTimeStamp` suspects that the algorithm alg1 is near to be broken and that some of the detached data objects referenced within the signed `ds:Manifest` might be substituted by another, that entity may build the `xadesenv111:RenewedDigests` unsigned property and incorporate it to the signature before computing the message imprint. Under these circumstances, the input to the message imprint would be: the canonicalized `xades:SignedProperties` element, the canonicalized `ds:Manifest` element, the canonicalized `ds:SignedInfo` element, the canonicalized `ds:SignatureValue` element, the canonicalized `ds:KeyInfo` element, the canonicalized present unsigned properties in their order of appearance, and the the canonicalized `ds:Object` element that contains the signed `ds:Manifest`. It is important to emphasize two things:

1) That the `xadesenv111:RenewedDigests` unsigned property is part now of the input to the message imprint and that its contents (the new digest values on the detached data objects) are secured by the time-stamp token.

2) That the actual contents of the detached data objects are not part of the message imprint. This is an important fact that allows that if the attack explained above succeeds, the global validation of the signature succeeds and yet identify that oneo of the detached signed data objects has been substituted, as explained below.

If an attacker achieves to substitute one of the original detached data objects signed through the signed `ds:Manifest` with another one with the same digest value computed with the broken algorithm alg1, then an entity validating this XAdES signature will:

1) Successfully validate the time-stamp token encapsulated within the `xadesv141:ArchiveTimeStamp`, as none of the elements that were concatenated for computing the input to its message imprint have been changed. This ensures that whatever is covered by the time-stamp token has not been altered.

2) When checking the digest values of the `xadesenv111:RenewedDigests` unsigned property, the validator will detect a failure, as the fake detached data object does not have the same digest value than the original one using the algorithm alg2, stronger than alg1. This will allow the validator to conclude that the data object referenced by first the `ds:Manifest`'s `ds:Reference` child element, is not the original one, and in consequence, the signature on this particular object has to be considered as not valid.

3) Allow the validator still to consider the signature valid on the rest of the signed data objects if the rest of the processes performed for validating the signature succed, as the success of the archive time-stamp validation ensures that nothing within the signature has changed.

Usage of `xadesenv111:RenewedDigests` unsigned property achieves two effects: first of all it counters the threat resulting of the combination of digest algorithm break and detached data object substitution; and secondly it allows to identify the substituted data object and preserve the validity of the signature for the not subsitutied detached data objects.

<PAGE BREAK>

Annex E (informative): Change History

date	Version	Information about changes
November 2013	V0.0.3	Include an extension mechanism to `xades:SignedSignatureProperties` and `xades:SignedDataObjectProperties`. Restructuring of the document to align with EN 319 122, addition of new properties: `xadesenv111:SigningCertificate`, `xadesenv111:SignerRole`, `xadesenv111:SignaturePolicyStorage`, `xadesenv111:RenewedDigest`, and `xadesenv111:SPDocSpecification` qualifier and the introduction of different conformance levels.

History

Document history		
<Version>	<Date>	<Milestone>
0.0.4	30/11/2013	Stable draft version for public comments.