COMMON ISIS-MAILTRUST SPECIFICATIONS
FOR INTEROPERABLE PKI APPLICATIONS

FROM T7 & TELETRUST


# ISIS-MTT SPECIFICATION

# PRELIMINARY SPECIFICATION

## LONG-TERM CONSERVATION OF ELECTRONIC SIGNATURES

FINAL DRAFT – JUNE 30TH 2004

# Contact Information

MailTrusT Working Group of the TeleTrusT Deutschland e.V.: www.teletrust.de
The up-to-date version of ISIS-MTT can be downloaded from the above web-site.
Please send comments and questions to isis-mtt@teletrust.de

Editors:

Ralf Brandner, InterComponentWare AG, ralf.brandner@intercomponentware.com
Brian Hunter, Fraunhofer Institute SIT, brian.hunter@sit.fraunhofer.de
Ulrich Pordesch, Fraunhofer Institute SIT, pordesch@sit.fraunhofer.de
Michael Tielemann, DATEV eG, michael.tielemann@datev.de

The following people have contributed to the ISIS-MTT Specification:

Alexander Rossnagel, Universität Kassel, a.rossnagel@uni-kassel.de

# Document History

| VERSION | DATE | CHANGES |
|---|---|---|
| 0.1 | 22.03.2004 | First draft |
| 0.2 | 02.05.2004 | Corrections of phrases and layout, considering several comments |
| Final Draft | 30.06.2004 | Corrections of phrases |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Table of Contents

# 1 Preface

If signed documents are to be stored for long periods, specific problems have to be considered. First of all, already after some years certificates, certificate paths or revocation information may no longer be available, because certification service provider may delete these data from the directory according to its certificate policy, the address of online directory or status services may change or because of cessation of business. Secondly, after a longer period of time hash algorithms or public key algorithms, which are used in electronic signatures, may become weak. For these reasons it may become impossible to verify electronic signatures which were provable years ago. Long-term conservation of signatures means the preservation of the power of evidence of electronic signatures over long periods. In order to retain the power of evidence of the electronic signatures over a long period of time, some additional measures have to be applied.

Firstly, data which are needed to verify an electronic signature (verification data), but which will probably not be available online in the future, must be stored in a secure and persistent way. All required verification data should be saved within the electronic signature, so that these data are always available when the electronic signature is verified. Furthermore, the electronic signatures must be renewed before their respective cryptographic algorithms become insecure or the related certificates expire.

This preliminary ISIS-MTT Specification part provides methods for preserving the power of evidence of electronic signatures over long periods based on the Cryptographic Message Syntax (CMS) [RFC 3369].

The first part of this preliminary ISIS-MTT Specification part defines how the verification data can be integrated into signed documents according to [RFC 3369]. The structure is based on the "Electronic Signature Formats" Specification [ETSI TS 101 733]. The goal of this preliminary ISIS-MTT Specification part is to provide an ISIS-MTT profile of the [ETSI TS 101 733] that only deals with long-term conservation and verification of electronic signatures. This profile complies with the existing ISIS-MTT Specification and reduces the complexity in the [ETSI TS 101 733].

The second part of this preliminary ISIS-MTT Specification part defines methods to renew signatures and to verify renewed signatures. Due to the differing needs of application areas, two specifications are profiled. Firstly, according to the above mentioned [ETSI TS 101 733], an enhanced signature format containing archive time-stamps is profiled. The specification of the ETSI is meaningful for a small number of documents, which are present in the CMS format. Secondly, the "Evidence Record Syntax" (ERS) [LTANS ERS], which is currently discussed within the IETF Working Group "Long Term Archive and Notary Services" (LTANS), is profiled. This specification supports signature renewal for data of any format (signed documents, externally stored verification data like CRLs or policy statements), considers the needs of large document archives and is conforming to the German Signature Law (SigG) [SigG 01].

Because this preliminary ISIS-MTT Specification part deals with long-term conservation of signatures, which is related to various parts of the core specification as well as to the optional SigG-Profile, it is suggested to publish this part as an appendix to the ISIS-MTT Specification.

# 2 Generation and Processing Model

Interoperability in the context of the ISIS-MTT Specification means that electronic signatures created with software from one vendor can be verified using software from another vendor. In order to achieve this interoperability, the model will be divided into two processes, namely, the signature creation process (generation) and the signature verification process (processing). Suitable requirements will be defined for these two processes. However, this basic two-process model is insufficient for long-term conservation of electronic signatures. Foremost the verifier shall be able to add the verification data to the document, after he has verified the signature. This means that the verifier does not only read the values from the signature, but he continues "creating" the signature by adding verification data to it. Secondly, signature renewal is completely independent of the signature creation and the verification processes. By taking these considerations into account, this preliminary ISIS-MTT Specification part continues to subdivide the generation, as well as the processing processes:

- Generation

1. *Signature creation:* This process generates the initial signature over the document. It adds all necessary signed attributes to the signature. It may or may not add verification data or a time-stamp as unsigned attributes to the signature.
2. *Signature completion:* This process retrieves the verification data (certificate path and revocation information) for the signing certificate and may request a signature time stamp. It adds this data as unsigned attributes to the signature.
3. *Signature renewal:* Signatures or the whole document are archive time-stamped, if necessary. The archive time-stamp may be integrated in the signature immediately after it was generated or may be stored and managed separately by an archive system.

- Processing

4. *Signature verification:* Signatures are validated using the verification data included in the signature by signature completion (2). A result of the signature verification process is a point in time were signature and signed content has to have existed because algorithms became weak or involved certificates expired.
5. *Verification of the renewed signatures:* Renewed signatures (archive time-stamps) are verified, in order to ensure that the given data existed at the necessary time in the past.

Signature completion could be part of an initial verification by a receiver of the signed document but could also be done by an archive system later and maybe without a (cryptographically) verification of the signatures.

Sub-processes 1 and 2 could be combined for one component; however this increases the complexity of all generation modules. Also, sub-processes 4 and 5 could be combined.

*Interoperability* in the context of long-term signatures means that the signatures created and completed by sub-processes 1 and 2 are correctly verifiable by sub-process 4. Furthermore, interoperability implies that the renewed signatures created by sub-process 3 are verifiable by sub-process 5.

Thus, *conformity* to this preliminary ISIS-MTT Specification part is defined as follows: A software component conforms to this preliminary ISIS-MTT Specification part, if it fulfils all the requirements on the sub-processes that it implements. A software product conforms to this preliminary ISIS-MTT Specification part, if it consists of software components, which fulfil the requirements on all of the sub-processes.

# 3 Integration and Verification of Long-Term Verification Data

## 3.1 Scope of Verification Data

According to ISIS-MTT Part 5, the following data are necessary to verify electronic signatures:

- Certificates: Signer Certificate, relevant attribute certificates and all certificates of their certificate paths including cross-certificates and root certificates;
- Revocation information: CRLs or OCSP-Responses for the certificates and their certificate paths;
- Certificate policies.

Furthermore, it makes sense to verify a signature to acquire the latest signature creation time from the time stamp, since the *signing-time* in the signature itself may not be considered trustworthy.

## 3.2 Underlying Specifications

Since the [RFC 3369] is inadequate for the storage of all verification data, the [ETSI TS 101 733] is used as well. However, not all the fields defined in [ETSI TS 101 733] are considered, only those that are needed for long-term conservation of signatures and those needed for the conformance requirements with [ETSI TS 101 733].

For signature creation, either the "Basic Electronic Signature" type (see Section 8.1) or the "Explicit policy based Electronic Signature" type (see Section 8.2) must be supported, according to the Conformance Requirements of [ETSI TS 101 733]. For the verification of electronic signatures, either the "Verification using time-stamping" type (see Section 8.3) or the "Verification using secure records" type (see Section 8.4) must be supported. In order to meet the minimal requirements of [ETSI TS 101 733] and the requirements of long-term conservation, the following profile is based on the "Basic Electronic Signature" and the "Verification using time-stamping".

## 3.3 Verification Time Reference

During the verification process, the certificate path is verified for a specific point of time in the past (reference time). The following rules are to be used to ascertain this reference time, which will be used in the verification of the signature validity.

### 3.3.1 Ascertainment of Reference Time

If the signature time-stamp is present, then the reference time will be the time within the time-stamp. Otherwise, the reference time is to be obtained in one of the following ways (since the current time is, in general, not practical to use for the verification of signed data after a long time):

1. the signed attribute *signing-time* (asserted by signer)
2. the unsigned attribute *signing-time* (the signature does not protect this time)
3. an out-of-band date passed to the program.

ISIS-MTT conforming products (signature verification components) must support methods 1 and 3 and this must be a configurable user option. Method 2 may be supported, because signing-time as an unsigned attribute is still possible, according to ISIS-MTT Part 3. The program must inform the user, through a verification dialog box or a verification protocol, how the reference time was obtained.

### 3.3.2 Verification of Signature Validity

The following conditions are to be checked, based on the reference time, while verifying the signature validity:

1. Certificate path validity: A valid certificate path, according to ISIS-MTT Part 5 (PKIX-model), must be found.
2. OCSP-Responses and CRLs, which were used in the path validation, must have been produced after the reference time.
3. The algorithms' security must have been suitable at the reference time (see chapter 5).

## 3.4 Additional unsigned Attributes

The following table defines the unsigned attributes, in which the necessary verification data can be stored. Certificates and CRLs should be continued to be stored in the corresponding fields in SignedData (certificates, crls).

**Table 1: Additional unsigned attributes for the integration of verification data**

| Fields | | | References | | | | | ISIS-MTT Long Term Conservation | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | Name | Semantics | Document | Chap. | Status | | Table | Support | | | Notes |
| | | | | | Gen | Proc | | Gen[1] | Proc[2] | Values | |
| 1 | *signature-time-stamp id-aa-signatureTimeStampToken* | Time stamp over the signature. | TS 101 733 | 6.1.1 | +- | ++ | | +- | ++ | | |
| 2 | *complete-certificate-references id-aa-ets-certificateRefs* | References all certificates used in the validation of the signing certificate. | TS 101 733 | 6.2.1 | +- | ++ | | +- | ++ | | [1] |
| 3 | *complete-revocation-references id-aa-ets-revocationRefs* | References all revocation used in the validation of the signing certificate. | TS 101 733 | 6.2.2 | +- | ++ | | +- | ++ | | [2] |
| 4 | *attribute-certificate-references id-aa-ets-attrCertificateRefs* | Attribute certificate reference. | TS 101 733 | 6.2.3 | +- | +- | | +- | +- | | [3] |
| 5 | *attribute-revocation-references id-aa-ets-attrRevocationRefs* | Reference to attribute certificate revocation lists. | TS 101 733 | 6.2.4 | +- | +- | | +- | +- | | [3] |

[1] Gen means here the described sub-process 2: Signature completion (see chapter 2).

[2] Proc means here the described sub-process 4: Signature verification (see chapter 2).

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | *certificate-values id-aa-ets-certValues* | Certificates in path, including signer certificate. | TS 101 733 | 6.3.3 | +- | ++ | | +- | ++ | | [4] |
| 7 | *revocation-values id-aa-ets-revocationValues* | Revocation information used in validation. | TS 101 733 | 6.3.4 | +- | ++ | | +- | ++ | | [5] |

[1] Includes certificate references in path validating signing certificate, but not includes signing certificate (it is included in *signing-certificate*). Compliant components SHALL include the issuerSerial element.

[2] Includes references to all revocation information used for validation of certificate path.

[3] May be included if the signature contains an attribute certificate.

[4] When this attribute is included, it MUST contain the signing certificate and all CA certificates (including the trust anchor used) in the validation path.

[5] When this attribute is included, conforming components SHALL include all revocation information used for the validation of the certificate path. Conforming components SHALL include OCSP responses for all certificates (that can be revoked) and may include CRLs or other revocation information.

# 4 Generation and Verification of Renewed Signatures

## 4.1 Introduction

For the signature renewal exist different specifications, which are suitable for different application areas:

- Signature oriented [ETSI TS 101 733]: verifiable with signature, no separate data management, everything encapsulated in one object, however, access to signature is necessary and is expensive –> in particular, suitable for small numbers of documents
- Document oriented [LTANS ERS]: efficient for large numbers of documents, no access to signatures required, however, separate data management

ISIS-MTT supports both options. However, both cannot be used simultaneously.

## 4.2 Conditions for Signature Renewal

Conditions for signature renewal must be defined if signature renewal lines up, so that the document does not lose its power of evidence and can be successfully validated in the future. The related renewal period here is based upon the PKIX validity model of certificate path validation. There are two situations when a signature renewal waits for treatment:

- Before the expiry of the signing certificate or the latest time-stamp certificate: A new time-stamp must be retrieved.
- Before a cryptographic algorithm (public key algorithm or hash algorithm) used within the signature or time-stamp becomes weak: A new algorithm must be used and a new time-stamp has to be retrieved.

Signature renewal may occur more regularly, according to local policy.

## 4.3 Integrated Archive Time-Stamps

[ETSI TS 101 733] defines an enhanced signature format (called ES-A), using the additional unsigned-attribute archive-time-stamp, that allows the long-term conservation of individual signatures. Products conforming to this preliminary ISIS-MTT Specification part may optionally support the creation of ETSI archive time-stamps, but must support the verification of such archive time-stamps.

Since the archive time-stamp is an extension to the “Basic Electronic Signature”, as described in section 3.2, all the additional attributes listed in section 3.4 are included in archive time-stamps and must be supported by verification components. The following table shows the additional unsigned attribute that is needed for archive time-stamping.

**Table 2: Additional unsigned attribute for the integration of archive time-stamps**

| Fields | | | References | | | | | ISIS-MTT Long Term Conservation | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | Name | Semantics | Document | Chap. | Status | | Table | Support | | | Notes |
| | | | | | Gen | Proc | | Gen[3] | Proc[4] | Values | |
| 1 | *archive-time-stamp id-aa-ets-archiveTimestamp* | Time stamp over the encapContentInfo, Certificate and crls in SignedData, and all signed and unsigned attributes in signerInfos. | TS 101 733 | 6.4.1 | +- | +- | | +- | ++ | | |

[3] Gen means here the described sub-process 3: Signature renewal (see chapter 2).

[4] Proc means here the described sub-process 5: Verification of renewed signatures (see chapter 2).

## 4.4 Evidence Records

Evidence Records [LTANS ERS], defined in the IETF working group LTANS, are suitable for conservation of large numbers of documents [LTANS REQ]. The basic structure uses hash trees, which allows archive time-stamps to relate to many documents, thus reducing the cost (namely the number of time-stamps needed).

The structure creates a hash tree from several documents and includes a time-stamp of the root hash of this tree. This time-stamp is sufficient to protect the power of evidence of the documents (signed data, data, certificates, CRLs, etc.) until one of the algorithms in the time-stamp becomes weak or the time-stamp certificate expires. The structure must then be extended with a new time-stamp, which protects the original time-stamp. This is called time-stamp renewal. Another step occurs when the hash algorithm becomes weak. At this point a hash tree renewal takes place, in which all the conserved documents and the Evidence Records are rehashed with a new hash algorithm and a new time-stamp retrieved. These processes are described in [LTANS ERS].

Conforming components must support the generation of hash tree renewals. They may support time-stamp renewals, which is very easy but can be substituted by complex hash tree renewal. Verification components must support the verification of both kinds of renewal.

In the table below, the main elements of an EvidenceRecord are shown.

**Table 3: Main elements of the Evidence Record**

| Fields | | | Refs To ERS | ISIS-MTT Long Term Conservation | | | |
|---|---|---|---|---|---|---|---|
| # | Name | Semantics | | Support | | | Notes |
| | | | | Gen | Proc | Values | |
| 1 | *digestAlgorithms* | Sequence of all hash algorithms used. | 2.1 | ++ | ++ | | |
| 2 | *cryptoInfos* | Useful information for verification, e.g. certificates, crls, algorithm suitability, etc. | 2.1 | +- | +- | | [1] |
| 3 | *encryption* | Encryption information used to decrypt data. | 2.1 | - | +- | | [1] |
| 4 | *archiveTimeStampSequence* | Sequence of ArchiveTimeStampChain. | 4.1 | ++ | ++ | | |
| 5 | *archiveTimeStampChain* | Sequence of ArchiveTimeStamp. | 4.1 | +- | ++ | | [2] |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6 | *archiveTimeStamp* | Hash algorithm, reducedHashTree, time-stamp. | 3.1 | ++ | ++ | | [3] |
| [1] Components must be able to at least parse and ignore these elements. | | | | | | | |
| [2] Of course archiveTimestampChain must consist of at least one archive time-stamp. +- for generation means, that it is not necessary to support chains with more then one archive time-stamp | | | | | | | |
| [3] See following table for elements of *ArchiveTimeStamp*. | | | | | | | |

The following table profiled the ArchiveTimeStamp.

**Table 4: ArchiveTimeStamp (ERS)**

| # | Name | Semantics | Refs to ERS | Support | | | Notes |
|---|---|---|---|---|---|---|---|
| | | | | Gen | Proc | Values | |
| 1 | *digestAlgorithm* | AlgorithmIdentifier of hash algorithm used. | 3.1 | ++ | ++ | | [1] |
| 2 | *reducedHashTree* | Sequence of sequence of hashes (Octet String). | 3.1 | ++ | ++ | | |
| 3 | *timestamp* | ContentInfo. | 3.1 | ++ | ++ | | |
| [1] Need not be added to structure if *timeStamp* contains the hash algorithm that was used for the *reducedHashTree*. | | | | | | | |

According to [LTANS ERS], if the document is a CMS signed message, then the Evidence Record may be added to it after generation as an unsigned attribute. This is not necessary, but it is an option to avoid separate data objects (CMS message file, ERS file), if desired. ISIS-MTT conforming components may realize this feature.

**Table 5: Archive Time Stamps Attributes (ERS)**

| # | Name | Semantics | Refs | Support | | Notes |
|---|---|---|---|---|---|---|

| | | | to ERS | Gen | Proc | Values | |
|---|---|---|---|---|---|---|---|
| 1 | *id-ATS-Attribute 1* | CMS-Object including content selected and archive-timestamped as single object. | Appendix A | +- | +- | | |
| 2 | *id-ATS-Attribute 2* | CMS-Object and content selected and archive-timestamped as two separate objects. | Appendix A | +- | +- | | |

# 5 Security Suitability Check of Cryptographic Algorithms

Over long periods of time, the security of algorithms and their parameters decreases. This is the main reason for signature renewal: renewed electronic signatures (here: archive time-stamps) have to be generated before an algorithm or one of its parameters becomes insecure. It is obviously, that we need algorithm security policies in order to avoid contrary results within verification components trying to verify old signatures. In Part 6, ISIS-MTT established the algorithms to be supported by ISIS-MTT conforming components. But this part does not define which hash algorithms or public key algorithms in connection with their parameters today are regarded as secure enough or up to a certain time.

There are still only two official security suitability evaluations of security suitability of cryptographic algorithms used for qualified electronic signatures:

- Evaluation published by European Standardization Institute (ETSI) relevant for qualified signatures covering the requirements of the EU Directive [ETSI SR 002 176],
- Publications of the Regulatory Authority for Telecommunication and Post (RegTP) for qualified electronic signatures covering the requirements of the SigG [REGTP 01].

RegTP publishes its evaluation annually and it is stated in the SigG that only these algorithms and parameters may be used for qualified signatures. The modus of ETSI publications (periods, changes) and its legal binding is still not fixed.

ISIS-MTT does not want to publish its own security suitability evaluation yet. The existing ones are sufficient for qualified signatures and for simple and advanced signatures. Users or organizations may also define their own policies to meet their security needs.

As a minimum requirement for ISIS-MTT conforming components, it is demanded:

- ISIS-MTT conforming products must support RegTP-algorithm security policy,
- ISIS-MTT conforming products should support ETSI-algorithm security policy,
- ISIS-MTT conforming products may support other algorithm security policies,
- ISIS-MTT conforming products must indicate in the verification result, which security suitability policy has been used.

For security policies published by RegTP, the following algorithm, derived from German Signature Ordinance (SigV) [SigV 01], should be used to determine the algorithm suitability, in the German legislative realm for qualified certificates:

- Search the German Federal Gazette, which contains the most recent algorithm suitability evaluation (for the desired cryptographic algorithm).
- If it contains a date indicating until when the algorithm was suitable, then this is the desired date.
- If it contains a date indicating until when the algorithm is anticipated to be suitable and this date is earlier than the next publication, then this is the desired date.
- Otherwise the desired date is the date of the next publication of the German Federal Gazette.

ETSI has published the suitability of algorithms on the European level, however it is not clearly as there will be further publications in the future or how a sequence of such publications should be processed.

# 6 SigG-Profiles

The ISIS-MTT SigG-Profile restricts and enhances the core part of the ISIS-MTT Specification, considering the requirements of the SigG regarding qualified electronic signatures. According to this paragraph, the profiles of the above-mentioned specifications will be restricted.

## 6.1 Legal Requirements

§ 17 Signature Ordinance contains guidelines on how signature renewal is to be done. These guidelines are interpreted as follows:

1. The renewed signature must be created before the used cryptographic algorithms lose their security suitability.
2. The renewed signatures must use hash and public key algorithms whose security is currently suitable.
3. The security suitability of the algorithms is published by the RegTP. Since the suitability of an algorithm may be revised in a future publication, future signature verifications may need to use another publication. The publication that must be used, is always the latest publication, in which the algorithm was evaluated.
4. A qualified time-stamp is necessary. An additional qualified signature is legally unnecessary, if the qualified time-stamp self has a qualified signature. Qualified time-stamps that are created using a pseudonym certificate fulfill these formal requirements.
5. The renewed signature must at least have the quality of a qualified signature. Qualified time-stamps from an accredited CA are necessary for qualified signatures with CA accreditation.
6. All earlier signatures (multiple parallel or serial signatures and earlier renewed signatures) of a document must be included.
7. Multiple documents may receive a new signature together. It is generally irrelevant, if the signature renewal relates to other documents, as long as one wishes only to prove that the document existed at a given time. Moreover, hash values of the documents instead of the document themselves could be signed, provided that the security suitability of the hash algorithm is sufficient.

SigG and SigV contain no specific guidelines on how verification data should be archived or used in the verification process. However, the following guidelines are indirectly relevant (see also ISIS-MTT SigG-Profile)

8. Certificates and their revocation information must be available from qualified CAs for at least 5 years after the certificate expires and at least 30 years from accredited CAs. Therefore, the user must consider that after this time period the online verification data are no longer present. Thus, he must securely archive this data before this period ends.
9. The validity model is important for the certificate path validation. Valid signatures according to the chain model can be verified in accordance to the shell model as invalid. Under the aspect of long-term conservation a signature must be always verified according to the same validity model.

## 6.2 Integration and Verification of Long-Term Verification Data

During the process of completing (e.g. adding necessary verification data, or other attributes) qualified signatures, only the following data must be included, according to the SigG:

- Signing certificate and related attribute certificates
- CA certificate
- OCSP relating to the signing certificate and attribute certificates

Time-stamps must also include their signing certificates.

Root certificates should be saved as well. Additional data (OCSP responses relating to the CAs or root certificates) need not to be included, according to this preliminary ISIS-MTT Specification part section 3.2 and ISIS-MTT Part 5 Certificate Path.

It is recommended for users, that verification data are included from the start, if the signature will be needed in the future when these data are no longer available. For qualified signatures, this information must be available for at least 5 years after certificate expiry. For qualified signatures with CA accreditation, this period is 30 years after the signing certificate expires.

## 6.3 Generation and verification of Renewed Signatures

In addition to the remarks in section 4 of this preliminary ISIS-MTT Specification part, the following restrictions are applicable:

1. Qualified signature renewal according to SigG based on the technique within the ETSI-Specification is not conforming to SigV. Thus, the LTANS method should be used instead. Verification processes must indicate, which renewal technique was used.
2. Archive time-stamps must be issued by a qualified CA using a signing certificate that displays the CA's accreditation. If the document contains signatures with CA accreditation, then accredited time-stamps should be used. Generation processes for renewed signatures must either only use accredited time-stamps or offer the choice to vary the types of time-stamps. Generation processes must show the security level of all the renewed signatures (at least all the renewed signatures pertaining to one document).
3. All the verification data defined in section 3.2 may be included in the time-stamp. However, at a minimum, the generating process must include the signing certificate in the time-stamp.
4. When a document consists of separate data objects (e.g. document file and multiple signature files), the signature renewal must occur for all objects together. This means, that the generation process must place the hash values of the individual objects under one father node and must not place any other hash within this group. Similarly, the verification process must verify that no other hash value is included under this father node, besides the hashes relating to the data objects that are to be verified.
5. While processing signatures and renewal signatures ISIS-MTT conforming components must use the security policy published by RegTP (see section 5).

# 7 ASN.1 Definitions

The following definitions are parts of [ETSI TS 101 733] and [LTANS ERS]. Only those parts of these specifications are laid down here, which are used in this preliminary ISIS-MTT Specification part.

## 7.1 Enhanced Electronic Signature Formats

### Signature Timestamp

```
id-aa-signatureTimeStampToken OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 14}

SignatureTimeStampToken ::= TimeStampToken
```

### Complete Certificate Refs.

```
id-aa-ets-certificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 21}

CompleteCertificateRefs ::= SEQUENCE OF OtherCertID

OtherCertID ::= SEQUENCE {
      OtherCertHash     OtherHash,
      IssuerSerial      IssuerSerial OPTIONAL }

OtherHash ::= CHOICE {
      sha1Hash OtherHashValue, -- This contains a SHA-1 hash
      otherHash OtherHashAlgAndValue}

OtherHashValue ::= OCTET STRING

OtherHashAlgAndValue ::= SEQUENCE {
      hashAlgorithm     AlgorithmIdentifier,
      hashValue         OtherHashValue }
```

### Complete Revocation Refs

```
id-aa-ets-revocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 22}

CompleteRevocationRefs ::=  SEQUENCE OF CrlOcspRef

CrlOcspRef ::= SEQUENCE {
      Crlids            [0] CRLListID     OPTIONAL,
      Ocspids           [1] OcspListID    OPTIONAL,
      otherRev          [2] OtherRevRefs  OPTIONAL}

CRLListID ::=  SEQUENCE {
      Crls              SEQUENCE OF CrlValidatedID}

CrlValidatedID ::=  SEQUENCE {
      CrlHash           OtherHash,
      CrlIdentifier     CrlIdentifier OPTIONAL}

CrlIdentifier ::= SEQUENCE {
      Crlissuer         Name,
      CrlIssuedTime     UTCTime,
      CrlNumber         INTEGER OPTIONAL}
```

```
OcspListID ::=  SEQUENCE {
      OcspResponses     SEQUENCE OF OcspResponsesID}

OcspResponsesID ::=  SEQUENCE {
      OcspIdentifier    OcspIdentifier,
      OcspRepHash       OtherHash   OPTIONAL}

OcspIdentifier ::= SEQUENCE {
      ocspResponderID   ResponderID, -- As in OCSP response data
      producedAt        GeneralizedTime -- As in OCSP response data}

OtherRevRefs ::= SEQUENCE {
      otherRevRefType   OtherRevRefType,
      otherRevRefs      ANY DEFINED BY otherRevRefType}

OtherRevRefType ::= OBJECT IDENTIFIER
```

### Attribute certificate references

```
id-aa-ets-attrCertificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 28}

AttributeCertificateRefs ::=  SEQUENCE OF OtherCertID
```

### Attribute revocation references

```
id-aa-ets-attrRevocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 29}

AttributeRevocationRefs ::=  SEQUENCE OF CrlOcspRef
```

### Certificate Values

```
id-aa-ets-certValues OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 23}

CertificateValues ::=  SEQUENCE OF Certificate
```

### Certificate Revocation Values

```
id-aa-ets-revocationValues OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 24}

RevocationValues ::=  SEQUENCE {
      CrlVals           [0] SEQUENCE OF CertificateList OPTIONAL,
      OcspVals          [1] SEQUENCE OF BasicOCSPResponse OPTIONAL,
      otherRevVals      [2] OtherRevVals OPTIONAL}

OtherRevVals ::= SEQUENCE {
      otherRevValType   OtherRevValType,
      otherRevVals      ANY DEFINED BY otherRevValType}

OtherRevValType ::= OBJECT IDENTIFIER
```

### Archive Timestamp

```
id-aa-ets-archiveTimestamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 27}

ArchiveTimeStampToken ::= TimeStampToken
```

## 7.2 Evidence Record Syntax

```
ArchiveTimeStamp ::= SEQUENCE {
      digestAlgorithm   AlgorithmIdentifier,
      reducedHashtree   [0] SEQUENCE OF {SEQUENCE OF OCTET STRING} OPTIONAL,
      timeStamp         ContentInfo}

ArchiveTimeStampChain ::=  SEQUENCE OF ArchiveTimeStamp

ArchiveTimeStampSequence ::= SEQUENCE OF ArchiveTimeStampChain

EvidenceRecord ::= SEQUENCE {
      Version           INTEGER {v1(1)},
      digestAlgorithms  SEQUENCE OF AlgorithmIdentifier,
      cryptoInfos       [0] CryptoInfos OPTIONAL,
      encryption        [1] EncryptionMethod OPTIONAL,
      archiveTimeStampSequence     ArchiveTimeStampSequence}

id-EvidenceRecord ::= {id-ATS-Attribute 1}
id-EvidenceRecord ::= {id-ATS-Attribute 2}
```

# References


[ETSI TS 101 733] ETSI TS 101 733 V1.5.1 (2003-12) Technical Specification: Electronic Signatures and Infrastructures (ESI); Electronic Signature Formats.

[ETSI SR 002 176] ETSI SR 002 176 V1.1.1 (2003-03) Special Report: Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures.

[LTANS ERS] Evidence Record Syntax (ERS), draft for IETF Working Group LTANS (Long term archive and notary services), February 2004.

[LTANS REQ] Long-term Archive Service Requirements, draft for IETF Working Group LTANS (Long term archive and notary services), February 2004.

[RegTP 01] Regulierungsbehörde für Telekommunikation und Post (RegTP): Geeignete Algorithmen, Amtliche Veröffentlichungen der Regulierungsbehörde RegTP, Technische Regulierung Telekommunikation, Amtliche Veröffentlichungen, Bonn 5.7.2001, Bundesanzeiger Nr. 158 - Seite 18562 vom 24.August 2001, Bonn 5.7.2001.

[RFC 3369] Housley, R., Cryptographic Message Syntax (CMS), RFC 3369, 2002.

[SigG 01] Law Governing Framework Conditions for Electronic Signatures and Amending Other Regulations, Bundesgesetzblatt Nr. 22, 2001, S.876.

[SigV 01] Ordinance on Digital Signatures, Bundesgesetzblatt Nr. 59, 2001, S. 3074.