BSI Technical Guideline 03125
Preservation of Evidence of Cryptographically Signed Documents

Annex TR-ESOR-S:
Interface Specifications

<table>
<thead>
<tr>
<th>Designation</th>
<th>Interface Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation</td>
<td>BSI TR-ESOR-S</td>
</tr>
<tr>
<td>Version</td>
<td>1.2</td>
</tr>
<tr>
<td>Date</td>
<td>19.12.14</td>
</tr>
</tbody>
</table>
# Table of contents

1. Introduction .................................................................................................................. 5

2. Overview ....................................................................................................................... 7

3. Functional requirements ............................................................................................... 8
   3.1 External functions ....................................................................................................... 8
   3.2 Internal functions ...................................................................................................... 10

4. Generic definition of all interface functions ................................................................. 12
   4.1 General data structures .............................................................................................. 12
      4.1.1 TAPVersion .......................................................................................................... 12
      4.1.2 ArchiveData ......................................................................................................... 12
      4.1.3 DeltaArchiveData ................................................................................................. 12
      4.1.4 AOID .................................................................................................................. 13
      4.1.5 ArchiveVersionTokenSequence and ArchiveTokenSequence .......................... 13
      4.1.6 SignatureObject ................................................................................................. 14
      4.1.7 Controls .............................................................................................................. 14
      4.1.8 ResponseStatus ..................................................................................................... 15
      4.1.9 ArchiveVersion .................................................................................................... 15
      4.1.10 ArchiveVersionToken ........................................................................................ 15
      4.1.11 ArchiveToken ...................................................................................................... 15
      4.2 Definition of interface functions ............................................................................. 15
         4.2.1 ArchiveSubmissionRequest .............................................................................. 15
         4.2.2 ArchiveSubmissionResponse ........................................................................... 15
         4.2.3 ArchiveUpdateRequest ..................................................................................... 16
         4.2.4 ArchiveUpdateResponse ................................................................................... 16
         4.2.5 ArchiveRetrievalRequest ................................................................................. 17
         4.2.6 ArchiveRetrievalResponse .............................................................................. 17
         4.2.7 ArchiveEvidenceRequest ............................................................................... 17
         4.2.8 ArchiveEvidenceResponse ............................................................................. 18
         4.2.9 ArchiveDeletionRequest .................................................................................. 19
         4.2.10 ArchiveDeletionResponse ............................................................................... 19
         4.2.11 ArchiveDataRequest ........................................................................................ 19
         4.2.12 ArchiveDataResponse ...................................................................................... 20
         4.2.13 VerifyRequest .................................................................................................. 21
         4.2.14 VerifyResponse ............................................................................................... 22
         4.2.15 SignRequest ..................................................................................................... 23
         4.2.16 SignResponse ................................................................................................... 23
         4.2.17 TimestampRequest ............................................................................................ 23
         4.2.18 TimestampResponse ....................................................................................... 24
         4.2.19 HashRequest .................................................................................................... 25
         4.2.20 HashResponse ................................................................................................. 25

5. Concrete definition of interfaces .................................................................................. 26
   5.1 S.1 interface ............................................................................................................... 26
      5.1.1 VerifyRequest ..................................................................................................... 26
      5.1.2 VerifyResponse .................................................................................................. 26
      5.1.3 SignRequest ....................................................................................................... 27
      5.1.4 SignResponse ..................................................................................................... 27
   5.2 S.2 interface ............................................................................................................... 27
      5.2.1 The interface for storing the archival information packages ............................. 27
      5.2.2 The interface for storing the cryptographic evidence records ............................ 28
5.3 S.3 interface ........................................................................................................... 29
5.3.1 TimestampRequest .......................................................................................... 29
5.3.2 TimestampResponse ......................................................................................... 29
5.3.3 VerifyRequest .................................................................................................. 29
5.3.4 VerifyResponse ............................................................................................... 29
5.3.5 HashRequest .................................................................................................. 29
5.3.6 HashResponse ................................................................................................ 29
5.4 S.4 interface ........................................................................................................ 30
5.4.1 ArchiveSubmissionRequest ............................................................................ 30
5.4.2 ArchiveSubmissionResponse ........................................................................ 30
5.4.3 ArchiveEvidenceRequest ............................................................................... 30
5.4.4 ArchiveEvidenceResponse ............................................................................. 30
5.4.5 ArchiveRetrievalRequest ............................................................................... 30
5.4.6 ArchiveRetrievalResponse ............................................................................. 30
5.4.7 ArchiveDeletionRequest ................................................................................ 31
5.4.8 ArchiveDeletionResponse ............................................................................... 31
5.4.9 ArchiveDataRequest ....................................................................................... 31
5.4.10 ArchiveDataResponse ................................................................................... 31
5.4.11 VerifyRequest .............................................................................................. 31
5.4.12 VerifyResponse ............................................................................................ 31
5.5 S.5 interface ........................................................................................................ 31
5.5.1 ArchiveRetrievalRequest ............................................................................... 31
5.5.2 ArchiveRetrievalResponse ............................................................................. 31
5.5.3 ArchiveDeletionRequest ................................................................................ 31
5.5.4 ArchiveDeletionResponse ............................................................................... 31
5.5.5 ArchiveDataRequest ....................................................................................... 32
5.5.6 ArchiveDataResponse ..................................................................................... 32
5.6 S.6 interface ........................................................................................................ 32
5.6.1 ArchiveSubmissionRequest .......................................................................... 32
5.6.2 ArchiveSubmissionResponse ........................................................................ 32
5.6.3 ArchiveUpdateRequest .................................................................................. 32
5.6.4 ArchiveUpdateResponse ............................................................................... 32
5.6.5 ArchiveEvidenceRequest .............................................................................. 32
5.6.6 ArchiveEvidenceResponse ............................................................................. 33

6. Security-related requirements ................................................................................. 34
1. Introduction

The goal of the Technical Guideline "Preservation of Evidence of Cryptographically Signed Documents" is to specify security-related requirements for the long-term preservation of evidence of cryptographically signed electronic documents and data along with associated electronic administrative data (meta data).

A Middleware defined for this purpose (TR-ESOR-Middleware) in the sense of this Technical Guideline includes all the modules (M) and interfaces (S) ["S" for the German word "Schnittstellen"] used for securing and preserving the authenticity and proving the integrity of the stored documents and data.

The Reference Architecture introduced in the Main Document of this Technical Guideline consists of the functions and logical units described below:

- The S.4 input interface of the TR-ESOR-Middleware which serves to embed the TR-ESOR-Middleware in the existing IT and infrastructure landscape;
- The "ArchiSafe-Module" ([TR-ESOR-M.1]) which regulates the flow of information in the Middleware, implements the security requirements for the interfaces with the IT applications and ensures that the application systems are decoupled from the ECM/Long-Term Storage;
- The "Cryptographic-Module" ([TR-ESOR-M.2]) and the associated S.1 and S.3 interfaces that provide all the functions needed for creating (optional) and verifying electronic signatures, post-verifying electronic certificates and for obtaining qualified time stamps for the Middleware. Furthermore, it can provide functions for the encryption and decryption of data and documents;
- The "ArchiSig-Module" ([TR-ESOR-M.3]) with the S.6 interface that provides the functions needed for the preservation of evidence of the digitally signed documents;
- An ECM/Long-Term Storage with the S.2 and S.5 interfaces that assumes the physical archiving/storage and also the storage of the meta data that preserve evidence. This ECM/Long-Term Storage is no longer directly a part of the Technical Guideline, but requirements will be made for it through the two interfaces that are still part of the TR-ESOR-Middleware.

The application layer that can include an XML adapter is not a direct part of the Technical Guideline either, even though this XML adapter can be implemented as part of a Middleware.

The IT Reference Architecture depicted in Figure 1 is based on the ArchiSafe\(^1\) Reference Architecture and is supposed to make possible and support the logical (functional) interoperability of future products with the goals and requirements of the Technical Guideline.

\(^1\) For further information, see [http://www.archisafe.de](http://www.archisafe.de).
This Technical Guideline has a modular design and the individual annexes to the Main Document specify the functional and security-related requirements for the needed IT components and interfaces of the TR-ESOR-Middleware. The specifications are strictly platform-, product-, and manufacturer-independent.

This document bears the designation "Annex TR-ESOR-S" and specifies the interfaces between the individual modules as depicted in Figure 1.
2. Overview

The goal of the interfaces described in this document is to describe the functionally correct interaction of the individual modules of the Reference Architecture and thus define a standardised, functional interface. This document describes the interfaces that are within the area of application of this Technical Guideline as shown in Figure 1.

First, chapter 3 will describe the functional requirements of the TR-ESOR-Middleware. Generic specifications of functions will be derived from the functional requirements in chapter 4. The concrete description of the individual interfaces of the Reference Architecture follows in chapter 5 on the basis of the functional specifications that have been developed before. Finally, security requirements are stipulated for all interfaces in chapter 6.
3. Functional requirements

This chapter defines the functional requirements for the TR-ESOR-Middleware that shall be implemented at least from the technical applications' point of view. In doing so, the functional requirements from chapter 5 of the Main Document of this Technical Guideline are used as a basis. The specifications are based on a suggestion of the LTANS working group of IETF for a Trusted Archive Protocol - TAP. [TAP 03]. Here, the individual functions are described in a very abstract and functional manner and are not mapped to the individual interfaces of the Reference Architecture yet. Additional optional parameters, with which additional (product-specific) options can be defined provided that the security-related requirements described in the Main Document and the annexes to this Technical Guideline are not circumvented or compromised, are not specified in general, but are always allowed.

3.1 External functions

The following functions shall at least be supported by an archive Middleware.

- **ArchiveSubmissionRequest / -Response**
  
  *ArchiveSubmissionRequest / -Response* is a function for storing future archive data object (group) in the ECM/Long-Term Storage. The expected transfer object is an XAIP container, but other data formats are also accepted. Optionally, the import from one or several Evidence Records pursuant to [RFC4998] or [RFC6283] and belonging to a specific XAIP version or to the binary data transferred may be initiated. The unique ID (AOID) of the archive data object (group) is returned as response.

- **ArchiveRetrievalRequest / -Response**
  
  *ArchiveRetrievalRequest / -Response* is a function for the retrieval of an archive data object (group) from the ECM/Long-Term Storage. The input parameter is the AOID of the archive data object (group) to be requested and, if applicable, the version numbers of the archive data object (group) to be requested. If the VersionID contains the value "all", all existing versions of an archive data object (group) shall then be returned. In case of success, the corresponding archive data object (group) is delivered in an XML format as response (XAIP pursuant to Annex F).

- **ArchiveEvidenceRequest / -Response**
  
  *ArchiveEvidenceRequest / -Response* is a function for requesting evidence records for an archive data object (group). The input parameter is an AOID (and, if applicable, the VersionID) of the archive data object (group) for which the evidence records should be requested. If one VersionID or several VersionIDs was specified in addition to an AOID, the ArchiSig-Module shall return the Evidence Record(s) for this VersionID or these VersionIDs. If the VersionID element is not specified, the evidence records for the last version of the XAIP are returned. When the value "all" is specified in the VersionID element, the Evidence Records are expected as response for all versions. This enables complete verification of the authenticity and integrity since the time at which the archive data object (group) was archived.

- **ArchiveDeletionRequest / -Response**
  
  *ArchiveDeletionRequest / -Response* is a function for deletion of an archive data object (group). The input parameter is the AOID of the archive data object (group) which is to be deleted. If an archive data object (group) is to be deleted prior to the expiry of the retention period set, a reason for this action, which is logged, shall be transferred for this action in

---

2. [RFC4998] shall, [RFC6283] may be supported.
3. If the Long-Term Storage used does not allow deletion in general, for example because WORM media are used, then the TR-ESOR-Middleware should return a corresponding error when this function is requested.
addition to the name (identifier) of the instance making the request. If there is more than one
version of an archive data object (group), all versions are to be deleted by this request. As
response, a status message regarding the successful deletion is expected.

Furthermore, an implementation of the TR-ESOR-Middleware should support the following functions:

- **ArchiveUpdateRequest / -Response**
  
  *ArchiveUpdateRequest / -Response* is a function for changing an existing archive data object
  (group) in the ECM/Long-Term Storage. In doing so, each change necessarily creates a new
  version of the archive data object (group). With this function, elements may be added to an
  already existing archive data object (group) or changed. In the sense of this Technical
  Guideline, permissible changes to archive data objects (groups) are
  
  1. Adding a new version with additional documents, data, meta data, signatures, signature
     verification information or other evidence records to an archive data object (group),
  2. Changing documents, data, meta data in a new version of the archive data object
     (group),
  3. Deleting the assignment of documents, data, meta data or credentials which are
     available in a previous version from the new version (by not taking them over).

  The expected transfer object is a Delta-XAIP container (see [TR-ESOR-F]) that includes a
  new *versionManifest*, references to the previous version as well as to objects that have
  been taken over without any change and the elements to be supplemented which are to be
  supplemented in a new version of an already stored archive data object (group). As response,
  the new version number of the archive data object (group) is expected. A new AOID is not
  issued, it remains the same.

- **ArchiveDataRequest / -Response**

  *ArchiveDataRequest / -Response* is a function for the targeted retrieval of individual data
  elements from an individual archive data object (group) without having to return the
  respective entire archive data object (group) to the IT application. This function can be used,
  for example, to create search indices, determine the object owner, determine the minimum
  retention period, or retrieve electronic signatures.

  As input parameters, the AOID of the respective archive data object (group) and an indication
  of the data elements to be retrieved are expected. As response, the data elements retrieved are
  expected.

- **VerifyRequest / -Response**

  With the function *VerifyRequest/-Response*, XML-based archival information packages
  (XAIP) including the supplemental evidence data (signatures, time stamps, certificates,
  certificate revocation lists, OCSP responses etc.) and Evidence Records that are contained
  therein or were additionally transferred can be verified.

  The syntax of the XAIP document is verified in the ArchiSafe-Module.

  Verification of supplemental evidence data and evidence records is already necessary upon
  archiving, because it cannot be guaranteed that this verification can be carried out until the end
  of the retention period. The obligation of the certification service provider to keep information
  about the attribution and validity of electronic certificates has a time limit.

  The submitted value is an input element, preferably an XAIP, or a SignatureObject element or
  both. In an XAIP, digital signatures and, if applicable, also one or several evidence records
  may be included in the CredentialSection. All signature information (signatures, time
  stamps, certificates, certificate revocation lists, OCSP responses etc.) up to a trustworthy root
  shall be verified. If possible, the verification information (certificates, certificate revocation
  lists, OCSP responses) is stored as unsigned attributes or properties in the corresponding
  signatures or in the provided data structures of the XAIP.

  However, the actual meta data/information packages/credentials are not removed.
The implementation of the actual verification function shall be carried out in the Cryptographic-Module (see Annex TR-ESOR-M.2) as a component of the TR-ESOR-Middleware via the S.1 interface (see also chapter 3.2) described in Annex TR-ESOR-S in order to achieve a correspondingly high level of trustworthiness. The verification information needed for the verification shall be requested from the certificate service providers.

### 3.2 Internal functions

The following functions are not functions that the archive Middleware has to provide to a specialised application. They are internal functions of the TR-ESOR-Middleware that are necessary for the preservation of evidence.

- **VerifyRequest / -Response**

  *VerifyRequest / -Response* is a function for verifying advanced or qualified electronic signatures, supplemental evidence data (certificates, certificate status information, time stamps etc.), Evidence Records and archive data objects (groups) (XAIPs). The function is necessary in order to verify the integrity and authenticity of an already existing digital signature during archiving (*ArchiveSubmissionRequest*) and to embed the resulting verification information in the corresponding signatures or in the XAIP.\(^5\) Verification is already necessary upon archiving, because it cannot be guaranteed that this verification can be carried out until the end of the retention period. The obligation of the certification service provider to keep information about the attribution and validity of electronic certificates has a time limit.

  The submitted value is an input element, preferably an XAIP, or a SignatureObject element (in this respect, see also chapter 4.1) or both. In an XAIP, digital signatures and, if applicable, one or several evidence records may be included in the information packages, but also in the CredentialSection (see also footnote 5). All signatures are to be verified up to a trustworthy root.

  The implementation of the actual verification function shall be carried out in the Cryptographic-Module (see Annex TR-ESOR-M.2) as a component of the TR-ESOR-Middleware in order to achieve a correspondingly high level of trustworthiness. The verification information needed for the verification shall be requested from the certificate service providers.

- **SignRequest / -Response**

  *SignRequest / -Response* is a function for creating (advanced) electronic signatures. This function is needed when information packages to be archived submitted by an *ArchiveSubmissionRequest* are to be furnished by the ArchiSafe-Module with an entry signature.

  The submitted value is an XAIP to be signed (see footnote 5 on page 10).

  The implementation of the partial function "hash value generation" shall, the partial function "create signature" can be carried out in the cryptographic component of the TR-ESOR-Middleware (see TR-ESOR-M.2) in order to achieve a correspondingly high level of trustworthiness. For the partial function "create signature", the services of a corresponding service provider may also be used.

- **TimestampRequest / -Response**

  *TimestampRequest / -Response* is a function for creating a qualified time stamp, which bears a qualified electronic signature. This function is needed for signature renewal.

  The submitted value is a hash value of the data which is to be secured with a time stamp.

  The time stamps can be requested from a (trustworthy) certification service provider or created by the Cryptographic-Module itself (see Annex TR-ESOR-M.2) as a component of the

---

\(^5\) When proprietary data formats are used, the respective product manufacturer shall offer an equivalent solution.
Middleware. In this case, this Middleware component shall be able to create qualified electronic signatures and have a reliable time signal.

- **HashRequest / -Response**

  HashRequest / -Response is a function for calculating a hash value from a data element. This function is also needed to calculate hash values for the hash tree generation.

  The submitted value is the piece of data (a data element or a group of data elements) based on which a hash value is to be calculated and, if applicable, an optional parameter which identifies the hash algorithm to be used.

  The implementation of this function shall be carried out in the cryptographic component of the TR-ESOR-Middleware (see [TR-ESOR-M.2]) in order to achieve a correspondingly high level of trustworthiness.
4. Generic definition of all interface functions

Based on the requirements described in the previous chapter, the individual functions will be specified formally in this chapter on the basis of ASN.1. The OASIS-DSS standard is used as far as possible. The specifications consciously abstract from concrete implementation details of the Reference Architecture described in chapter 7 of the Main Document of this Technical Guideline.

4.1 General data structures

In this section, data structures (data types) will be defined so that they can be used by the functions described below.

4.1.1 TAPVersion

This type describes the current version of the Trusted Archive Protocol (TAP). The Version 3 indicated here is based on Version 1.2 of the Technical Guideline.

TAPVersion ::= INTEGER { v3(1) }

4.1.2 ArchiveData

The ArchiveData type contains an archival information package (XAIP document) in the data field and, optionally, a type information or a reference to a type information (for example a URI or an OID to the XML schema on which it is based) in the type field. Another data format may be agreed and, in that case, it shall be indicated in the type information.

ArchiveData ::= SEQUENCE
{
    type   ArchiveDataType OPTIONAL,
    data   OCTET STRING
}

ArchiveDataType ::= CHOICE
{
    oid    OBJECT IDENTIFIER, -- DEFAULT xaip
    uri    IA5String
}

4.1.3 DeltaArchiveData

The DeltaArchiveData type contains a delta archival information package (DXAIP document) in the deltaData field and, optionally, a type information or a reference to a type information (for example a URI or an OID to the XML schema on which it is based) in the type field.

DeltaArchiveData ::= SEQUENCE
{
    type   DeltaArchiveDataType OPTIONAL, -- DEFAULT dxaip
    deltaData   OCTET STRING
}

DeltaArchiveDataType ::= CHOICE

See http://www.oasis-open.org
4.1.4 AOID

The `AOID` data type is a unique identifier for an archive data object (group). An `AOID` shall have been generated for each successfully stored archive data object (group).

`AOID ::= SEQUENCE
{  
aoid GeneralName,
archiveInfos [0] ArchiveInfos OPTIONAL
}

ArchiveInfos ::= SEQUENCE SIZE (1..MAX) OF ArchiveInfo

ArchiveInfo ::= SEQUENCE {
    archiveInfoType OBJECT IDENTIFIER,
    archiveInfoValue [0] ANY DEFINED BY archiveInfoType
}

The `GeneralName` structured data type may be used in order to code different forms of names or identifiers. The suggested structure enables the creation of locally defined names or identifiers.

The ASN.1 syntax for the `GeneralName` data type is defined in [RFC2459].

The `archiveInfos` field may (optionally) contain additional information about the archive environment, for example such as information about the time of archiving or an identifier (address) of the Long-Term Storage stipulated for the permanent storage.

4.1.5 ArchiveVersionTokenSequence and ArchiveTokenSequence

The `ArchiveTokenSequence` data type contains an AOID element and, optionally, a list of `VersionID(s)`. 

---

7 With the submission of an AOID element during the `ArchiveSubmissionRequest`, the AOID can be issued by the application making the request. As a rule, however, this element is missing and the AOID is provided by the module receiving the request. The AOID is then generated either by the ArchiSig-Module or by the Long-Term Storage.

8 See http://www.ietf.org/rfc/rfc2459.txt
ArchiveTokenSequence ::= SEQUENCE SIZE (1..MAX) {
    aoid AOID,
    archiveVersionTokenSequence ArchiveVersionTokenSequence OPTIONAL
}

ArchiveVersionTokenSequence ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
    versionID VersionID OPTIONAL
}

VersionID ::= String

4.1.6 SignatureObject

The following ASN.1 definition constitutes a signature object.

SignatureObject ::= ContentInfo

ContentInfo ::= SEQUENCE {
    contentType ContentType
    content [0] EXPLICIT ANY DEFINED BY contentType
}

ContentType ::= OBJECT IDENTIFIER -- identifies the type of signed data

[TR-ESOR-M.2], chapter 5.1.1, applies with respect to the signature formats used.

4.1.7 Controls

The Controls data field is provided for optional input and output elements and may, for example, contain corresponding control elements which are to be defined within the scope of a profiling of this specification.

This type can be used both for requests and for responses.

Among other things, this type may be used to submit additional information:

- that identifies the request/response with an ID,
- that qualifies the request/response in more detail,
- that serves for identification and authentication between the requesting and executing modules,
- that contains an internal Business Application ID of the archive data object (group) or the archive data object ID (AOID) issued by the Business Application, and
- that contains additional instructions that, for example, request the additional execution of certain operations.

The corresponding field shall not be used in order to circumvent or compromise the security-related requirements described in the Main Document and the annexes to this Technical Guideline.

The executing module shall deny every Request that contains an unknown or unsupported instruction in the Controls type.

The executing module shall not return any information in a Response to actions that were not defined in the associated Request.
Controls ::= SEQUENCE SIZE (1..MAX) OF Control

Control ::= SEQUENCE{
  controlType  ControlType,
  controlValue ANY DEFINED BY controlType OPTIONAL
}

ControlType ::= OBJECT IDENTIFIER

4.1.8 ResponseStatus
This type describes the result of a Request and contains a status code and (optionally) a status text.

ResponseStatus ::= SEQUENCE {
  code StatusCode,
  statusMessage String OPTIONAL
}

Alternatively, the definition of the <Result> element of the OASIS DSS standard can also be used for the specification of the result field.

4.2 Definition of interface functions

4.2.1 ArchiveSubmissionRequest

An ArchiveSubmissionRequest submits an archive data object (XAIP) or an archive data object (group) in any other format to the module requested. This requested module stores the data object directly in the ECM/Long-Term Storage or submits the data object to another module that takes over the actual storing. The ArchiveSubmissionRequest shall be answered with an ArchiveSubmissionResponse.

ArchiveSubmissionRequest ::= SEQUENCE {
  version TAPVersion DEFAULT v3,
  requestData ArchiveData,
  requestControls [0] Controls OPTIONAL
}

(A4.2-1) According to this specification, it should be possible to submit an AOID as well as one or several evidence records or to request a verification report pursuant to [TR-ESOR-VR] for the archive data object submitted by using the Controls of the ArchiveSubmissionRequest.

(A4.2-2) When importing evidence records, they shall be verified completely by the TR-ESOR-Middleware. This includes the verification steps provided in the corresponding ERS standard, whereby the respective certificates of the time stamps shall be verified completely up to a trustworthy root.

4.2.2 ArchiveSubmissionResponse

The answer to an ArchiveSubmissionRequest is defined as follows:

---

9 See section 3.3 in [RFC4998] and section 2.3 in [RFC6283].
ArchiveSubmissionResponse ::= SEQUENCE
{
    version TAPVersion DEFAULT v3,
    status ResponseStatus,
    responseData AOID OPTIONAL,
    responseControls [0] Controls OPTIONAL
}

In the event of successful storage in the ECM/Long-Term Storage, the return of a unique archive data object (group) ID (AOID) that shall be presented for all further transactions concerning the archive data object (group) is part of the ArchiveSubmissionResponse. For the possible versioning of the archive data object (group) in the future, the version number is set to 1. The VersionID is not returned with the ArchiveSubmissionResponse, because it is always set to 1. (A4.2-3) If the signature verification failed in the ArchiveSubmissionRequest, a verification report pursuant to [TR-ESOR-VR] may be returned in the Controls. The request for a verification report may be controlled in the Controls of the ArchiveSubmissionRequests.

4.2.3 ArchiveUpdateRequest

An ArchiveUpdateRequest submits a "delta" archival information package (XAIP) with the desired changes/additions to the module requested in order to generate a new version. The requested module in the TR-ESOR Middleware ensures that the changes/additions are taken over in a new version of the archival information package in the ECM/Long-Term Storage. The ArchiveUpdateRequest shall be answered with an ArchiveUpdateResponse.

ArchiveUpdateRequest ::= SEQUENCE {
    version TAPVersion DEFAULT v3,
    requestData DeltaArchiveData,
    requestControls [0] Controls OPTIONAL
}

(A4.2-4) According to this specification, entering an AOID and entering the evidence records or entering the request for a verification report should be possible in the Controls of the ArchiveUpdateRequest.

4.2.4 ArchiveUpdateResponse

The answer to an ArchiveUpdateRequest is defined as follows:

ArchiveUpdateResponse ::= SEQUENCE
{
    version TAPVersion DEFAULT v3,
    status ResponseStatus,
    responseData VersionID OPTIONAL,
    responseControls [0] Controls OPTIONAL
}

VersionID ::= STRING

In the event of a successful update in the Long-Term Storage, the return of the new version number of the corresponding archive data object (group) is part of the ArchiveUpdateResponse.
If the verification of the data submitted has failed in the *ArchiveUpdateRequest*, a verification report pursuant to [TR-ESOR-VR] may be returned in the *Controls*. The request for a verification report may be controlled in the *Controls* of the *ArchiveSubmissionRequests*.

### 4.2.5 ArchiveRetrievalRequest

An *ArchiveRetrievalRequest* is a request to the system to return an archival information package (an XAIP document) to the Business Application making the request. The archival information package is identified by means of its *AOID* and, if applicable, with one or several version number(s).

**ArchiveRetrievalRequest** ::= SEQUENCE

\{
  version TAPVersion DEFAULT v3,
  requestData ArchiveTokenSequence,
  requestControls [0] Controls OPTIONAL
\}

The *ArchiveTokenSequence* element can contain a sequence of version identifiers indicating which versions of the archival information package exactly are to be returned. If the *VersionID* element is not specified, data objects and administrative information associated to the last version are returned. By indicating "all" for the *VersionID*, all existing versions of an archival information package are returned.

In the *requestControls*, there should be an input unit which states that the returned XAIP should contain the corresponding evidence record(s).

### 4.2.6 ArchiveRetrievalResponse

The answer of the system to an *ArchiveRetrievalRequest* is defined as follows:

**ArchiveRetrievalResponse** ::= SEQUENCE

\{
  version TAPVersion DEFAULT v3,
  status ResponseStatus,
  responseData ArchiveData OPTIONAL,
  responseControls [0] Controls OPTIONAL
\}

In the event of a successful execution of the request, the associated archival information package is returned in the optional *ArchiveData* field in the XAIP format. This field can be omitted in the event of an error.

### 4.2.7 ArchiveEvidenceRequest

An *ArchiveEvidenceRequest* serves to request technical (cryptographic) evidence records for the authenticity and integrity of an archive data object (group) stored and identified unambiguously by an *AOID*. Optionally, the version number to which the requested evidence record should refer may be indicated for the *AOID* element. If no version number is indicated, the evidence record shall refer to the current (last) version. By indicating "all" for the *VersionID*, the evidence records are returned for all existing versions of an archive data object (group).

An *ArchiveEvidenceRequest* is defined as follows:
BSI TR-ESOR-S: Interface Specifications

ArchiveEvidenceRequest ::= SEQUENCE{
  version TAPVersion DEFAULT v3,
  requestData ArchiveTokenSequence,
  requestControls [0] Controls OPTIONAL
}

(A4.2-8) It should be possible to indicate a parameter in the requestControls for the format of the evidence records requested. Format specifications for Evidence Records pursuant to [RFC4998] or [RFC6283] are possible values here.

(A4.2.9) A concrete product shall support Evidence Records pursuant to [RFC4998], but may also implement both formats. If no format is specified in the Controls, an Evidence Record pursuant to [RFC4998] is returned by default.

4.2.8 ArchiveEvidenceResponse

In the event of successful authentication, an ArchiveEvidenceRequest is answered with an ArchiveEvidenceResponse.

An ArchiveEvidenceResponse is defined as follows:

ArchiveEvidenceResponse ::= SEQUENCE {
  version TAPVersion DEFAULT v3,
  status ResponseStatus,
  responseData ArchiveEvidenceResults OPTIONAL,
  responseControls [0] Controls OPTIONAL
}

4.2.8.1 responseData

The responseData field returns the Evidence Record(s) requested for the archive data object (group) which was successfully identified unambiguously by an AOID. The format of this(these) Evidence Record(s) is based on the corresponding parameter value in the request. The status field describes the result of the ArchiveEvidenceRequest and contains a status code and (optionally) a status text.

ArchiveEvidenceResults ::= SEQUENCE{
  aoid AOID OPTIONAL,
  versionID VersionID OPTIONAL,
  status ResponseStatus OPTIONAL,
  evidenceData SEQUENCE OF EvidenceData OPTIONAL
}

VersionID ::= String

10 It is highly recommended for verification software that is not part of this Technical Guideline to implement both of these formats.
EvidenceData ::= SEQUENCE{
    credentialID String OPTIONAL,
    relatedObjects SEQUENCE OF String OPTIONAL,
    evidenceRecord CHOICE
      { ersEvidenceRecord EvidenceRecord,
        xmlEvidenceRecord XMLEvidenceRecord }
}

The credentialID field contains the ID of the evidence record, in particular if it is included in an archival information package and should be addressable with this ID.

The relatedObjects field contains a list of IDs of the data elements of the archival information package, by means of which the evidence record makes a statement about integrity.

The definition of the EvidenceRecord data type can be found in [RFC4998] and in chapter 5.5 of the Annex [TR-ESOR-F]. The definition of the XMLEvidenceRecord data type is defined in the Internet draft "Extensible Markup Language Evidence Record syntax" [RFC6283].

4.2.9 ArchiveDeletionRequest

The ArchiveDeletionRequest function makes it possible to delete an archive data object (group). An ArchiveDeletionRequest shall refer to an archive data object (group) that is stored and identified unambiguously by an AOID element. The ArchiveDeletionRequest shall be answered with an ArchiveDeletionResponse that provides reliable information about the (successful) deletion.

ArchiveDeletionRequest ::= SEQUENCE{
    version TAPVersion DEFAULT v3,
    requestData AOID,
    requestControls [0] Controls OPTIONAL
}

4.2.9.1 requestControls

(A4.2-10) In the event of an ArchiveDeletionRequest before the expiry of the retention period set for the respective archive data object (group), the requestControls field shall contain a reason (ReasonOfDeletion) for the deletion. Otherwise, the order for deletion shall be denied.

(A4.2-11) The reason should at least contain the following information:

• A unique identifier of the application making the request in requestorName and
• A loggable reason in requestInfo.

ReasonOfDeletion ::= SEQUENCE {
    requestorName GeneralName,
    requestInfo UTF8String
}

4.2.10 ArchiveDeletionResponse

The answer to an order for deletion, ArchiveDeletionResponse, is defined as follows

---

11 Of course, this function cannot delete data on backup media. If this data should also be destroyed for certain reasons, another solution shall be found.
ArchiveDeletionResponse ::= SEQUENCE{
    version          TAPVersion DEFAULT v3,
    status           ResponseStatus,
    responseControls [0] Controls OPTIONAL
}

In the ResponseStatus field, the result of the order for deletion is returned to the application making
the request. In the event of an error, the Controls element may contain a detailed error message.

4.2.11 ArchiveDataRequest

The retrieval of discrete data elements of an individual archival information package, ArchiveDataRequest, is defined as follows:

ArchiveDataRequest ::= SEQUENCE{
    version          TAPVersion DEFAULT v3,
    requestData      RequestDataSequence,
    requestControls  [0] Controls OPTIONAL
}

4.2.11.1 requestData

In this case, the requestData field contains the AOID of the archival information package from which individual data elements are to be retrieved. In addition, the data elements to be requested shall be indicated. This is done in dataLocation. Each data element (data node) of the archival information package to be retrieved shall be identified unambiguously. Here, the complete and unambiguous path within the tree structure of the archival information package or an unambiguous request expression shall be indicated in dataLocation\(^\text{12}\).

The explicit (optional) specification of a VersionID is not necessary for this function, since the data elements of individual versions can also be directly accessed via the dataLocation field. Data elements of several versions of an archival information package can also be requested at the same time.

RequestData ::= SEQUENCE{
    aoid          AOID,
    dataLocation  DataLocationSequence
}

DataLocationSequence ::= SEQUENCE SIZE (1..MAX) OF DataLocation

DataLocation ::= SEQUENCE {
    locationType   IA5String,
    locationValue  ANY DEFINED BY locationType
}

---

\(^\text{12}\) Assuming an XML based archival information package, the following may be suitable for the discrete addressing of XML data elements: XPath (see http://www.w3.org/TR/2007/REC-xpath20-20070123/), XQuery (see http://www.w3.org/TR/2007/REC-xquery-20070123/) or the XML Pointer Language XPointer (see http://www.w3.org/TR/2003/REC-xptr-framework-20030325/). By means of requestControl, it can be specified which request method exactly is to be used.
4.2.12 ArchiveDataResponse

In the event of successful authentication, an ArchiveDataRequest is answered with an ArchiveDataResponse. An ArchiveDataResponse is defined as follows:

```
ArchiveDataResponse ::= SEQUENCE {
    version             TAPVersion DEFAULT v3,
    status              ResponseStatus,
    responseData       ResponseDataElement OPTIONAL,
    responseControls    [0] Controls OPTIONAL
}
```

4.2.12.1 responseData

The responseData field returns the data that should and could be successfully retrieved from the archival information package in the value field. The value return field may also contain BASE64-encoded binary data.

In the location field, the corresponding identification parameter submitted in the search request (in the locationValue type) shall be returned.

If the data element searched for was not identified and retrieved, the value element is omitted.

```
ResponseDataElement ::= SEQUENCE SIZE (1..MAX) OF XAIPData
```

```
XAIPData ::= SEQUENCE{
    status             ResponseStatus OPTIONAL,
    location           [0] UTF8String,
    value              CHOICE {
        xmlData          UTF8String,
        binary           OCTET STRING
    } OPTIONAL
}
```

4.2.13 VerifyRequest

A VerifyRequest submits an archival information package (XAIP document) including the supplemental evidence data (signatures, time stamps, certificates, certificate revocation lists, OCSP responses etc.) and evidence records that are contained therein or were additionally transferred for verification.\(^{13}\)

In addition to the ArchiveData element, a sequence of SignatureObjects (SignatureObjectSequence) can also be submitted, in which detached signatures can be submitted for verification. If signatures are already contained in the ArchiveData element, the optional sequence of SignatureObjects (SignatureObjectSequence) can be omitted.

An evidence record element may also be submitted in the SignatureObject to initiate the corresponding verification of the evidence record. In this case, the attributes AOID and VersionID shall be available and the associated XAIP element shall be returned in the requestData.

If the SignatureObjectSequence element is missing, exactly one requestData element that contains the signature objects to be verified shall be available.

A VerifyRequest is defined as follows:

---

\(^{13}\) If a proprietary format of a product is used instead of the XAIP formats, only this format can be submitted in this place. It shall then be ensured in another way that there is equivalent functionality.
BSI TR-ESOR-S: Interface Specifications

```
VerifyRequest ::= SEQUENCE{
    version             TAPVersion DEFAULT v3,
    requestData        ArchiveData OPTIONAL,
    signatureObjectSequence SignatureObjectSequence OPTIONAL,
    requestControls     [0] Controls OPTIONAL
}
```

(A4.2-12) Entering control elements of a policy element or the request of a verification report pursuant to [TR-ESOR-VR] in Controls element should be possible.

(A4.2-13) All signature information (signatures, time stamps, certificates, certificate revocation lists, OCSP responses etc.) up to a trustworthy root shall be verified. The verification information (certificates, certificate revocation lists, OCSP responses) which was identified in this context is stored as unsigned attributes or properties in the corresponding signatures or in the data structures provided for this purpose in the credentialSection of the XAIP document.

(A4.2-14) If one or several xaip:EvidenceRecord elements pursuant to [TR-ESOR-F] are contained in the credentialSection of the XAIP containers submitted, they shall be verified correspondingly.

4.2.13.1 requestControls

The requestControls field may contain

- Information about the signature type,
- The signature format and
- Verification options, that e.g. refer to a pre-defined policy, or to a list of parameters that determine the procedure when signed objects are verified and which shall be used to verify the signatures,
- Or define additional actions that are to be carried out by the Cryptographic-Module, for example to use the verification report specified in the eCard-API of the BSI [TR-ESOR-VR] or [eCard-2] in order to return the verification results.

4.2.14 VerifyResponse

The answer to a VerifyRequest is defined as follows:

```
VerifyResponse ::= SEQUENCE{
    version             TAPVersion DEFAULT v3,
    status              ResponseStatus,
    responseData       ResponseData OPTIONAL,
    responseControls    [0] Controls OPTIONAL
}
```

4.2.14.1 responseData

The responseData field can contain the XAIP document with the verification information entered into the XML elements provided for this purpose (see [TR-ESOR-F]). Alternatively, only the verification information can also be returned, which then has to be entered into the elements provided in the XAIP data structure for this purpose or in alternative data formats by the module making the request.

4.2.15 SignRequest

A SignRequest submits an archival information package (XAIP document) for the creation of an electronic signature.\[14\]

---

\[14\] If a proprietary format of a product is used instead of the XAIP formats, only this format can be submitted in this place. It shall then be ensured in another way that there is equivalent functionality.
A `SignRequest` is defined as follows:

```
SignRequest ::= SEQUENCE{
    version TAPVersion DEFAULT v3,
    requestData RequestData,
    requestControls [0] Controls OPTIONAL
}
```

### 4.2.15.1 requestControls

The `requestControls` field may be used to submit additional information that contains

- Information about the signature type (e.g. XML signature, CMS signature or PDF signature),
- The signature format and
- Additional options that have to be included or observed when the signature is created,
- Or define additional actions that are to be carried out if, for example, the signature certificate is to be verified up to a trusted root certificate simultaneously with the signature and the verification results are to be entered in the sections of the XAIP document provided for this purpose (see [TR-ESOR-F]).

### 4.2.16 SignResponse

The answer to a `SignRequest` is defined as follows:

```
SignResponse ::= SEQUENCE{
    version TAPVersion DEFAULT v3,
    status ResponseStatus,
    responseData ResponseData OPTIONAL,
    responseControls [0] Controls OPTIONAL
}
```

### 4.2.16.1 responseData

In case of success, the `responseData` field contains the signed XAIP document with the signature data entered into the XML elements provided for this purpose (see [TR-ESOR-F]) or an alternative data format including the signature created.

Alternatively, only the signature data can also be returned, which then has to be entered into the elements provided in the XAIP data structure for this purpose or in alternative data structures by the module making the request.

### 4.2.17 TimestampRequest

A `TimestampRequest` submits a hash value for the creation or request of a time stamp for this hash value.
TimestampRequest ::= SEQUENCE{
    version        TAPVersion DEFAULT v3,
    requestData    MessageImprint,
    requestControls [0] Controls OPTIONAL
}

4.2.17.1 requestData
The requestData field contains the hash value which is to be time stamped and information about the hash algorithm used (see [RFC3161]).

MessageImprint ::= SEQUENCE{
    hashAlgorithm  AlgorithmIdentifier,
    hashedMessage  OCTET STRING
}

4.2.17.2 requestControls
The optional requestControls field may be used to submit additional information that contains
- The information about the time stamp type and the time stamp format ([RFC3161] time stamp or [RFC4998] or [RFC6283] archive time stamp) and
- Additional instructions that, for example, determine that an archive time stamp pursuant to [RFC4998] or [RFC6283] shall bear a qualified electronic signature or
- That also have to be included or observed during the creation of the time stamp.

4.2.18 TimestampResponse
The answer to a TimestampRequest is defined as follows on the basis of [RFC3161]:

TimestampResponse ::= SEQUENCE{
    version        TAPVersion DEFAULT v3,
    status         ResponseStatus,
    responseData  TimeStampToken OPTIONAL,
    responseControls [0] Controls OPTIONAL
}

4.2.18.1 responseData
The responseData field contains a TimeStampToken pursuant to [RFC3161] and shall be contained if there was no error during creation.

TimeStampToken ::= ContentInfo
    -- [RFC3161]
    -- contentType is id-signedData  ([RFC3852])
    -- content    is SignedData     ([RFC3852])

The corresponding type definition for id-signedData and SignedData can be found in the CMS specification [RFC3852]. The TimeStampToken shall not contain any signatures besides the signature of the time stamp service provider. The associated certification information shall be protected by the signature. Verification information that goes beyond that shall be stored in unsigned fields of the TimeStampToken.\(^\text{15}\)

\(^{15}\) For this purpose, the certificates and crls fields which are contained in the SignedData structure in the TimeStampToken should be used.
4.2.19 HashRequest

A HashRequest submits a binary value from which the hash value is to be calculated to the module. XML data is also submitted in such a manner even if it concerns the generation of an embedded XML signature or the like.

\[
\text{HashRequest} ::= \text{SEQUENCE}\{
\quad \text{version} \quad \text{TAPVersion DEFAULT v3,}
\quad \text{requestData} \quad \text{BIT STRING,}
\quad \text{requestControls} \quad [0] \text{ Controls OPTIONAL}
\}\n\]

4.2.19.1 requestData

The data to be hashed is submitted in binary format. The creation of the binary depiction is left to the requester of the function.

4.2.19.2 requestControls

The requestControls field may be used to submit additional information that
• Provides information about the hash algorithm to be used.

4.2.20 HashResponse

The answer to a HashRequest is defined as follows:

\[
\text{HashResponse} ::= \text{SEQUENCE}\{
\quad \text{version} \quad \text{TAPVersion DEFAULT v3,}
\quad \text{status} \quad \text{ResponseStatus,}
\quad \text{responseData} \quad [0] \text{ MessageImprint OPTIONAL,}
\quad \text{requestControls} \quad [1] \text{ Controls OPTIONAL}
\}\n\]

4.2.20.1 responseData

The responseData field contains the calculated hash value. The hash value is returned in the MessageImprint format pursuant to [RFC3161]. The response can be empty if there was an error during the generation of the hash value.
5. Concrete definition of interfaces

The specifications of the interfaces do not contain any definitions of security mechanisms on the network, transport, or application layers, but assume the use of reliable security mechanisms such as [RFC4346] or [RFC2401] for mutual authentication of the communication end points.

Those modules that offer the interfaces listed below shall completely and fully implement the functions of the interfaces listed below.

Figure 2: Functions of the interfaces

Figure 2 shows which interfaces have to offer which functions.

If not indicated otherwise, the specification of the individual interface functions can be identical to the corresponding function and type specifications in chapter 4.

5.1 S.1 interface

The primary purpose of the TR-ESOR-S.1 interface between the ArchiSafe-Module and the Cryptographic-Module is the verification and creation of electronic signatures that were or should be attached to electronic data to be archived (XAIP documents). The interface shall implement the following functions:

5.1.1 VerifyRequest

See chapter 4.2.13.

5.1.2 VerifyResponse

See chapter 4.2.14.
5.1.3 SignRequest
See chapter 4.2.15.

5.1.4 SignResponse
See chapter 4.2.16.

5.2 S.2 interface
The primary purpose of the TR-ESOR-S.2 between the ArchiSig-Module and the ECM/Long-Term Storage is to make the necessary read and write access to ArchiSig’s own database and the archive database in the ECM/Long-Term Storage possible for the ArchiSig-Module.

The S.2 interface can assume that

- Information and data in the ECM/Long-Term Storage can be neither intentionally nor negligently manipulated,
- Requested data from the ECM/Long-Term Storage represents a reproduction of the originally archived payload data (in particular the primary data) down to the last bit,
- Changes to the originally archived data are logged in a verifiable manner\(^\text{16}\) and
- The integrity and availability (protection against loss and destruction)\(^\text{17}\) of the information and data stored cannot be compromised.

(A5.2-1) For the storage of the archive data objects (groups) and the cryptographic evidence records, the ArchiSig-Module shall have one (or several) secure and efficient interface(s) to one (or several) trustworthy electronic ECM/Long-Term Storage(s).

(A5.2-2) For the first-time storage of archive data objects (groups), the S.2 interface shall enforce the option to submit an AOID that has already been generated and by means of which access to the archive data object (group) is possible at a later point in time or the creation of an AOID by the ECM/Long-Term Storage and submission of it to the module making the request prior to actual storage.\(^\text{18}\)

5.2.1 The interface for storing the archive data object (group)

(A5.2-3) The storage of the archive data object (groups) shall be carried out in such a manner that the relationship between the archive data object (group) stored and the associated archive data object (group) ID (AOID) remains technically traceable at all times.

The interface shall provide the following functions regarding the treatment of archive data objects (groups):

- A function for storing an archive data object (group) and the associated archive data object (group) ID (AOID),
- A function for read access to the archive data objects (groups). This is necessary in order to be able to re-calculate the hash values of the archive data objects (groups) stored and thus the

---

\(^\text{16}\) Proof should be demonstrated in the form of clear protocols and log files that, in general, should be stored and kept outside the storage system intended for the storage of the archive data objects (groups). The retention period of the protocols and log files is to be defined in a technical organisation policy.

\(^\text{17}\) For the operation of the ECM/Long-Term Storage, the verification and certification of the correctness for permanent proof of the completeness, authenticity, unadulteredness and availability of the electronically stored and preserved information and data on the basis of an extensive system documentation by an independent expert is recommended.

\(^\text{18}\) With the submission of an AOID element during the ArchiveSubmissionRequest, the AOID can be issued by the application making the request. As a rule, however, this element is missing and the AOID is provided by the module receiving the request. The AOID is then generated either by the ArchiSig-Module or by the Long-Term Storage. In the latter case, the AOID shall be generated by the ECM/Long-Term Storage and submitted to ArchiSig so that ArchiSig can insert the new AOID into the archive data object and then calculate a hash value for the part of this worthy of protection. Only then is the archive data object actually stored in the ECM/Long-Term Storage.
hash trees in the event of the imminent loss of the security characteristics of the hash algorithms used.

The interface should provide the following functions regarding the treatment of archive data objects (groups):

- A function for updating already archived archive data objects (groups).

5.2.1.1 ArchiveSubmissionRequest

See chapter 4.2.1 for general information.

An ArchiveSubmissionRequest of the S.2 interface submits an archive data objects (groups) and, optionally, also the associated archive data object (group) ID (AOID) to the electronic ECM/Long-Term Storage if it has not been generated by the ECM/Long-Term Storage itself, but rather beforehand by the ArchiSig-Module.

```
ArchiveSubmissionRequest ::= SEQUENCE{
    version           TAPVersion DEFAULT v3,
    aoid              AOID OPTIONAL,
    requestData       ArchiveData,
    requestControls   [0] Controls OPTIONAL
}
```

5.2.1.2 ArchiveSubmissionResponse

See chapter 4.2.2.

5.2.1.3 ArchiveUpdateRequest

See chapter 4.2.3.

5.2.1.4 ArchiveUpdateResponse

See chapter 4.2.4.

5.2.1.5 ArchiveRetrievalRequest

See chapter 4.2.5.

5.2.1.6 ArchiveRetrievalResponse

See chapter 4.2.6.

5.2.2 The interface for storing the cryptographic evidence records

The list of functions presented for storing cryptographic evidence records is only one possible design of the functions of this interface. Thus, a formal specification of this interface, for example in ASN.1 notation, has been omitted deliberately. The storage used for the securing of evidence records and its internal data structures as well as its performance regarding the functions mentioned above can have designs of various types. Thus, it would only be possible to make a generally applicable interface specification with a very high level of abstraction.

The storing of the cryptographic evidence records should be carried out in such a manner that

- Each natively computed and stored hash value for an archive data object (group) is unambiguously linked to the archive data object (group) ID (AOID) and VersionID belonging to the archive data object (group) and
- It remains technically traceable at all times which hash values were subsumed in which tree structure under an archive time stamp.

Thus, this interface should provide the following functions (based on its technical design):

- A function for the reliable storing of a hash value H calculated for an archive data object (group) and the associated archive data object (group) ID (AOID),
- A function for requesting a list of hash values that have not been secured with an archive time stamp pursuant to the ERS standard yet,
• A function for the reliable storing of archive time stamps computed or renewed pursuant to the ERS standard,
• A function for requesting individual or several archive time stamps for which signature renewal pursuant to § 17 SigV is necessary on account of the imminent loss of the security characteristics of the cryptographic algorithms and parameters these archive time stamps are based on,
• A function for requesting an evidence record pursuant to the ERS standard for proving the authenticity and integrity of an archive data object (group) determined unambiguously by its archive data object (group) ID (AOID).

5.3 S.3 interface

The primary purpose of the TR-ESOR-S.3 interface between the ArchiSig-Module and the Cryptographic-Module is the generation of hash values and the generation and verification of qualified time stamps. Both kinds of data are needed for the development of the Merkle hash trees [MER 1980].

(A5.3-1) For the creation of hash values and the request/retrieval and the verification of qualified time stamps, the ArchiSig-Module shall be able to access a cryptographic module through a secure and efficient interface that at least fulfils the obligatory requirements described in Annex [TR-ESOR-M.2] to this Technical Guideline.

This interface shall provide the following functions:
• A function for creating qualified time stamps,
• A function for verifying the qualified signature of qualified time stamps,
• A function for calculating hash value.

5.3.1 TimestampRequest

See chapter 4.2.17.

5.3.2 TimestampResponse

See chapter 4.2.18.

5.3.3 VerifyRequest

See chapter 4.2.13.

5.3.4 VerifyResponse

See chapter 4.2.14.

5.3.5 HashRequest

See chapter 4.2.19.

5.3.6 HashResponse

See chapter 4.2.20.

5.4 S.4 interface

The TR-ESOR-S.4 interface should make it possible for the Business Application to access the ECM/Long-Term Storage system in a standardised and functional manner. Furthermore, the interface should reliably prevent unauthorised access to the ECM/Long-Term Storage system.

(A5.4-1) The TR-ESOR-S.4 interface shall at least provide the following functions:
• A function for the secure and reliable storing of archive data object (group),
• A function for retrieving archive data object (group) (in XAIP format),
• A function for retrieving technical (cryptographic) evidence records,
A function for deleting archived data.

The **TR-ESOR-S4** interface should provide the following functions:

- A function for updating already archived archive data object (group),
- A function for retrieving data elements of individual archive data object (group),
- A function for verifying an archival information package (XAIP document) including the supplemental evidence data (signatures, time stamps, certificates, certificate revocation lists, OCSP responses etc.) and Evidence Records that are contained therein or were additionally transferred.

**A5.4.2** The TR-ESOR-Middleware may also be realised (in part) on the basis of asynchronous requests.\(^{19}\)

**5.4.1 ArchiveSubmissionRequest**
See chapter 4.2.1.

**5.4.2 ArchiveSubmissionResponse**
See chapter 4.2.2.

**5.4.3 ArchiveUpdateRequest**
See chapter 4.2.3.

**5.4.4 ArchiveUpdateResponse**
See chapter 4.2.4.

**5.4.5 ArchiveRetrievalRequest**
See chapter 4.2.5.

**5.4.6 ArchiveRetrievalResponse**
See chapter 4.2.6.

**5.4.7 ArchiveEvidenceRequest**
See chapter 4.2.7.

**5.4.8 ArchiveEvidenceResponse**
See chapter 4.2.8.

**5.4.9 ArchiveDeletionRequest**
See chapter 4.2.9.

**5.4.10 ArchiveDeletionResponse**
See chapter 4.2.10.

**5.4.11 ArchiveDataRequest**
See chapter 4.2.11.

**5.4.12 ArchiveDataResponse**
See chapter 4.2.12.

**5.4.13 VerifyRequest**
See chapter 4.2.13.

**5.4.14 VerifyResponse**
See chapter 4.2.14.

\(^{19}\) See also TR-ESOR-F, chapter 6.
5.5 S.5 interface

The **TR-ESOR-S.5** interface enables access from the ArchiSafe-Module to the ECM/Long-Term Storage without technical dependence on the cryptographically secured evidence records.

(A5.5-1) The **TR-ESOR-S.5** interface shall at least provide the following functions:

- A function for retrieving archive data object (group) (in XAIP format),
- A function for deleting archived data, and
- A function for retrieving data elements of individual archive data object (group).

(A5.5-2) The S.5 interface shall ensure that

- Information and data in the ECM/Long-Term Storage can be neither intentionally nor negligently manipulated,
- Requested data represents a reproduction of the originally archived payload data (in particular the primary data) down to the last bit,
- The deletion of archived data is verifiably logged, which includes, in particular, logging the reason in the event of premature deletion\(^{20}\) and
- The integrity, availability (protection against loss and destruction) and ability to audit\(^{21}\) the information and data stored cannot be compromised.

5.5.1 ArchiveRetrievalRequest

See chapter 4.2.5.

5.5.2 ArchiveRetrievalResponse

See chapter 4.2.6.

5.5.3 ArchiveDeletionRequest

See chapter 4.2.9.

5.5.4 ArchiveDeletionResponse

See chapter 4.2.10.

5.5.5 ArchiveDataRequest

See chapter 4.2.11.

5.5.6 ArchiveDataResponse

See chapter 4.2.12.

5.6 S.6 interface

The **TR-ESOR-S6** interface is used by the ArchiSafe-Module to access the ArchiSig-Module.

Archiving (new) archive data object (group) is possible with the **TR-ESOR-S.6** interface described here, which can be used to involve the ArchiSig-Module directly in the archiving procedure. This is a direct way to compute the securing hash values. Thus, it is impossible to circumvent this security function.

(A5.6-1) For the storage, in particular of electronically signed documents, and the retrieval of cryptographic evidence records for the technical proof of the authenticity and integrity of the archive

\(^{20}\) Proof should be demonstrated in the form of clear protocols and log files that, in general, should be stored and kept outside the storage system intended for the storage of the archive data objects (groups). The retention period of the protocols and log files is to be defined in a technical organisation policy.

\(^{21}\) For the operation of the ECM/Long-Term Storage, the verification and certification of the correctness of the procedure for permanent proof of the completeness, authenticity, unadulteratedness and availability of the electronically stored and preserved information and data on the basis of an extensive system documentation by an independent expert is recommended.
data object (group) stored in the electronic ECM/Long-Term Storage, the ArchiSafe-Module shall be able to access a secure and efficient interface to an ArchiSig-Module that at least fulfils the requirements described in Annex [TR-ESOR-M.3] to this Technical Guideline.

This interface shall provide the following functions:

- A function for storing archive data object (group),
- A function for retrieving Evidence Records pursuant to the ERS standard [RFC4998] or [RFC6283].

The interface should provide the following functions:

- A function for updating already archived archive data object (group),

5.6.1 ArchiveSubmissionRequest

See chapter 4.2.1 for general information.

An ArchiveSubmissionRequest of the S.6 interface submits an archive data object (group) and, optionally, also the associated archive data object (group) ID (AOID) to the ArchiSig-Module if it has not been generated by the ArchiSig-Module, but rather beforehand by the ArchiSafe-Module.

\[
\text{ArchiveSubmissionRequest ::= SEQUENCE}
\]
\[
\begin{align*}
\text{version} & : \text{TAPVersion DEFAULT v3}, \\
\text{aoid} & : \text{AOID OPTIONAL}, \\
\text{requestData} & : \text{ArchiveData}, \\
\text{requestControls} & : [0] \text{Controls OPTIONAL}
\end{align*}
\]

5.6.2 ArchiveSubmissionResponse

See chapter 4.2.2.

5.6.3 ArchiveUpdateRequest

See chapter 4.2.3.

5.6.4 ArchiveUpdateResponse

See chapter 4.2.4.

5.6.5 ArchiveEvidenceRequest

See chapter 4.2.7.

5.6.6 ArchiveEvidenceResponse

See chapter 4.2.8.
6. Security-related requirements

In this section, the security-related requirements for all the interfaces described in chapter 5 are listed.

(A6.1-1) The target module shall only be accessed after successful mutual authentication between the source module and the target module (if applicable, the specialised application making the request). Authentication shall be repeated for each request; alternatively, a secure tunnel can be maintained.

A Request or Response shall not be executed if the authentication between the requesting and the executing module was not possible or not successful on the network, transport, or application layers.

(A6.1-2) Mutual authentication shall be cryptographically sufficient so that it is impossible to exchange individual components without being noticed.

(A6.1-3) It shall not be possible to replay authentication or other messages.

(A6.1-4) Unauthorised access to authentication or payload data during communication shall be prevented reliably. The interface shall be implemented in such a manner that denial of service (DoS) or consequential errors such as buffer overflow or SQL injections can be ruled out.