Bundesamt
für Sicherheit in der
Informationstechnik

BSI Technische Richtlinie 03125
Beweiswerterhaltung kryptographisch signierter Dokumente

Anlage TR-ESOR-S:
Schnittstellenspezifikation

Bezeichnung	Schnittstellenspezifikation
Kürzel	BSI TR-ESOR-S
Version	1.2.1 (auf Basis der eIDAS-Verordnung)
Datum	15.03.2018

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Tel.: +49 228 99 9582-0
E-Mail: tresor@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

Ziel der Technischen Richtlinie „Beweiswerterhaltung kryptographisch signierter Dokumente“ ist die Spezifikation sicherheitstechnischer Anforderungen für den langfristigen Beweiswerterhalt von kryptographisch signierten elektronischen Dokumenten und Daten nebst zugehörigen elektronischen Verwaltungsdaten (Metadaten).

Eine für diese Zwecke definierte Middleware (TR-ESOR-Middleware) im Sinn dieser Richtlinie umfasst alle diejenigen Module (**M**) und Schnittstellen (**S**), die zur Sicherung und zum Erhalt der Authentizität und zum Nachweis der Integrität der aufbewahrten Dokumente und Daten eingesetzt werden.

Die im Hauptdokument dieser Technischen Richtlinie vorgestellte Referenzarchitektur besteht aus den nachfolgend beschriebenen funktionalen und logischen Einheiten:

- der Eingangs-Schnittstelle S.4 der TR-ESOR-Middleware, die dazu dient, die TR-ESOR-Middleware in die bestehende IT- und Infrastrukturlandschaft einzubetten;
- dem „ArchiSafe-Modul“ (**[TR-ESOR-M.1]**), welches den Informationsfluss in der Middleware regelt, die Sicherheitsanforderungen an die Schnittstellen zu den IT-Anwendungen umsetzt und für eine Entkopplung von Anwendungssystemen und ECM/Langzeitspeicher sorgt;
- dem „Krypto-Modul“ (**[TR-ESOR-M.2]**) nebst den zugehörigen Schnittstellen S.1 und S.3, das alle erforderlichen Funktionen zur Berechnung von Hashwerten und Prüfung elektronischer Signaturen bzw. Siegel bzw. Zeitstempel, zur Nachprüfung elektronischer Zertifikate und zum Einholen qualifizierter Zeitstempel sowie (optional) elektronischer Signaturen bzw. Siegel für die Middleware zur Verfügung stellt. Darüber hinaus kann es Funktionen zur Ver- und Entschlüsselung von Daten und Dokumenten zur Verfügung stellen;
- dem „ArchiSig-Modul“ (**[TR-ESOR-M.3]**) mit der Schnittstelle S.6, das die erforderlichen Funktionen für die Beweiswerterhaltung der digital signierten Unterlagen bereitstellt;
- einem ECM/Langzeitspeicher mit den Schnittstellen S.2 und S.5, der die physische Archivierung/Aufbewahrung und auch das Speichern der beweiswerterhaltenden Zusatzdaten übernimmt.
  Dieser ECM/Langzeitspeicher ist nicht mehr direkt Teil der Technischen Richtlinie, gleichwohl werden über die beiden Schnittstellen, die noch Teil der TR-ESOR-Middleware sind, Anforderungen daran gestellt.
  Ebenso wenig ist die Applikationsschicht, die auch einen XML-Adapter enthalten kann, direkter Teil der Technischen Richtlinie, auch wenn dieser XML-Adapter als Teil einer Middleware implementiert werden kann.

Die in Abbildung Abbildung 1 dargestellte IT-Referenzarchitektur orientiert sich an der ArchiSafe[1] Referenzarchitektur und soll die logische (funktionale) Interoperabilität künftiger Produkte mit den Zielen und Anforderungen der Technischen Richtlinie ermöglichen und unterstützen.

[1] Siehe dazu http://www.archisafe.de

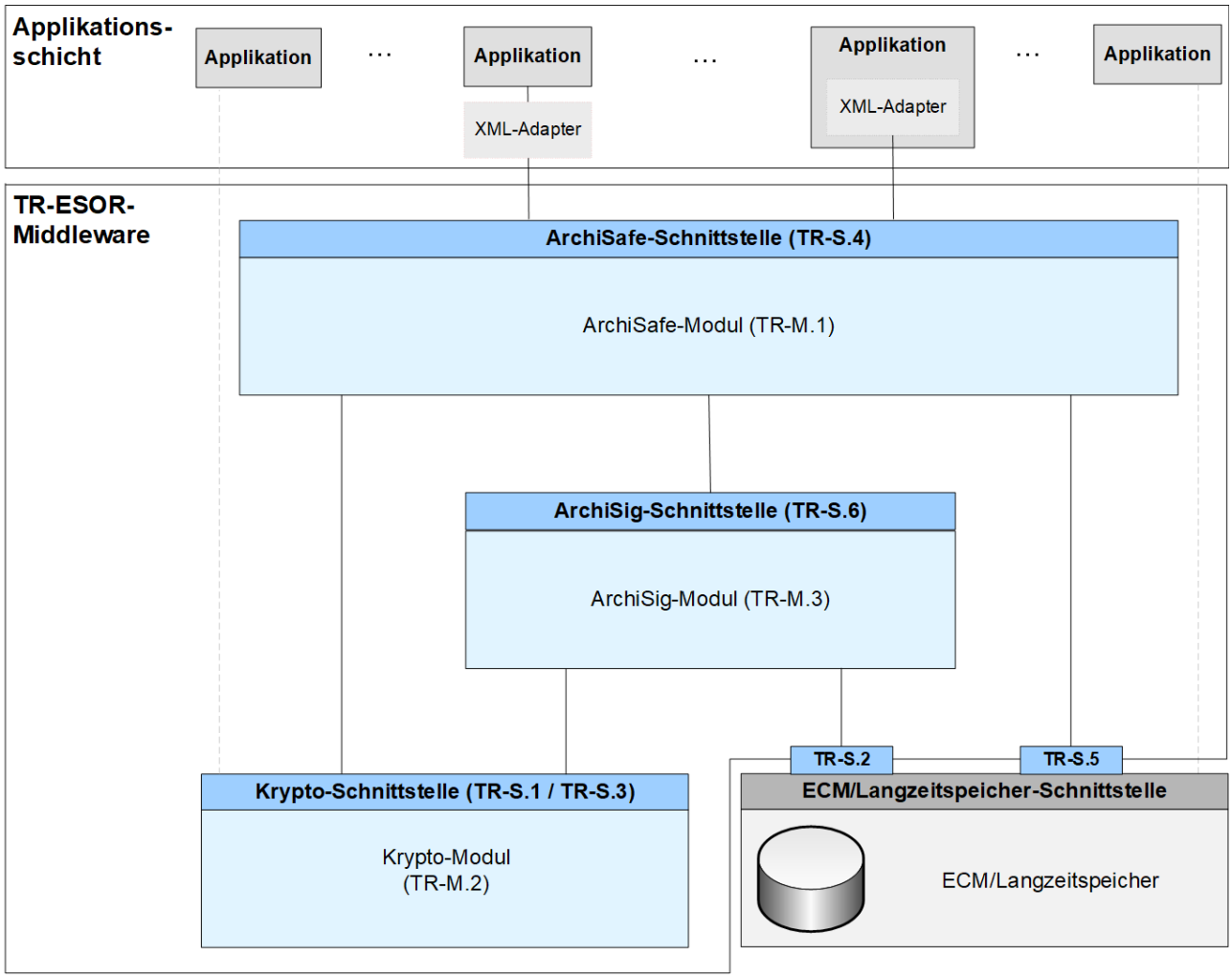

Abbildung 1: Schematische Darstellung der IT-Referenzarchitektur

Diese Technische Richtlinie ist modular aufgebaut und spezifiziert in einzelnen Anlagen zum Hauptdokument die funktionalen und sicherheitstechnischen Anforderungen an die erforderlichen IT-Komponenten und Schnittstellen der TR-ESOR-Middleware. Die Spezifikationen sind strikt plattform-, produkt-, und herstellerunabhängig.

Das vorliegende Dokument trägt die Bezeichnung „Anlage TR-ESOR-S“ und spezifiziert die Schnittstellen zwischen den einzelnen Modulen, wie sie in Abbildung Abbildung 1 dargestellt sind.

2. Übersicht

Ziel der in diesem Dokument beschriebenen Schnittstellen ist, das funktional korrekte Zusammenwirken der einzelnen Module der Referenzarchitektur zu beschreiben und damit eine einheitliche, funktionale Schnittstelle zu definieren. Dieses Dokument beschreibt die Schnittstellen, die in Abbildung 1 innerhalb des angezeigten Geltungsbereiches der TR liegen.

Zunächst werden in Kapitel 3 die funktionalen Anforderungen der TR-ESOR-Middleware beschrieben. Aus diesen funktionalen Anforderungen werden im nachfolgenden Kapitel 4 generische Funktionsspezifikationen abgeleitet. In Kapitel 5 folgt die konkrete Beschreibung der einzelnen Schnittstellen der Referenzarchitektur auf Basis der vorher entwickelten Funktionsspezifikationen. Schließlich sind in Kapitel 6 die Sicherheitsanforderungen für alle Schnittstellen beschrieben.

Hinweis: *Im folgenden Text umfasst der Begriff* ***„Digitale Signatur“*** *„fortgeschrittene elektronische Signaturen“ gemäß* ***[eIDAS-VO, Artikel 3 Nr. 11],*** *„qualifizierte elektronische Signaturen“ gemäß* ***[eIDAS-VO, Artikel 3 Nr. 12]****, „fortgeschrittenen elektronische Siegel“ gemäß* ***[eIDAS-VO, Artikel 3 Nr. 26]*** *und „qualifizierte elektronische Siegel“ gemäß* ***[eiDAS-VO, Artikel 3 Nr. 27]****. Insofern umfasst der Begriff „digital signierte Dokumente“ sowohl solche, die fortgeschrittene elektronische Signaturen oder Siegel bzw. qualifizierte elektronische Signaturen oder Siegel tragen.*

Mit dem Begriff der ***„kryptographisch signierten Dokumente“*** *sind in dieser TR neben den gemäß* ***[eIDAS-VO, Artikel 3 Nr. 12]*** *qualifiziert signierten, den gemäß* ***[eIDAS-VO, Artikel 3 Nr. 27]*** *qualifiziert gesiegelten oder den gemäß* ***[eIDAS-VO, Artikel 3 Nr. 34]*** *qualifiziert zeitgestempelten Dokumenten (im Sinne der eIDAS-Verordnung)) auch Dokumente mit einer fortgeschrittenen Signatur gemäß* ***[eIDAS-VO, Artikel 3 Nr. 11]*** *oder mit einem fortgeschrittenen Siegel gemäß* ***[eIDAS-VO, Artikel 3 Nr. 26]*** *oder mit einem elektronischen Zeitstempel gemäß* ***[eIDAS-VO, Artikel 3 Nr. 33]*** *erfasst, wie sie oft in der internen Kommunikation von Behörden entstehen. Nicht gemeint sind hier Dokumente mit einfachen Signaturen oder Siegeln basierend auf anderen (z. B. nicht-kryptographischen) Verfahren.*

3. Funktionale Anforderungen

Dieses Kapitel definiert die funktionalen Anforderungen an die TR-ESOR-Middleware, die aus Sicht der Fachanwendungen mindestens umgesetzt werden müssen. Dabei werden die funktionalen Anforderungen aus Kapitel 5 des Hauptdokumentes dieser Technischen Richtlinie als Grundlage herangezogen.

Die Spezifikationen lehnen sich an einen Vorschlag der LTANS Arbeitsgruppe der IETF für ein vertrauenswürdiges Archivprotokoll (Trusted Archive Protocol - TAP, **[TAP 03]**) an. Die einzelnen Funktionen werden hier nur sehr abstrakt und rein funktional beschrieben und noch nicht auf einzelne Schnittstellen der Referenzarchitektur abgebildet.

Grundsätzlich nicht angegeben aber immer zulässig sind zusätzliche optionale Parameter, mit welchen weitere (produktspezifische) Optionen festlegt werden können, sofern die im Hauptdokument und den Anlagen zu dieser technischen Richtlinie beschriebenen sicherheitstechnischen Anforderungen nicht umgangen oder kompromittiert werden.

3.1 Externe Funktionen

Die folgenden Funktionen müssen von einer Archiv-Middleware mindestens unterstützt werden.

- **ArchiveSubmissionRequest / -Response**

  ArchiveSubmissionRequest / -Response ist eine Funktion für das Ablegen von zukünftigen Archivdatenobjekten im ECM/Langzeitspeicher. Als Übergabeobjekt wird ein XAIP Container erwartet, aber auch andere Datenformate akzeptiert. Optional kann dabei der Import von einem oder mehreren zu einer bestimmten XAIP-Version bzw. zu den übergebenen Binärdaten gehörenden Evidence Records gemäß **[RFC4998]** oder **[RFC6283]**[2] angestoßen werden. Als Rückgabe wird die eindeutige ID (AOID) des Archivdatenobjektes geliefert.

- **ArchiveRetrievalRequest / -Response**

  ArchiveRetrievalRequest / -Response ist eine Funktion für das Abrufen eines Archivdatenobjektes aus dem ECM/Langzeitspeicher. Übergabeparameter ist die AOID des abzurufenden Archivdatenobjektes und ggf. die Versionsnummern des abzurufenden Archivdatenobjektes. Enthält die VersionID den Wert `all`, dann müssen alle existierenden Versionen eines Archivdatenobjektes zurückgeliefert werden. Als Rückgabe wird im Erfolgsfall das entsprechende Archivdatenobjekt in einem XML-Format (XAIP gemäß Anhang F) geliefert.

- **ArchiveEvidenceRequest / -Response**

  ArchiveEvidenceRequest / -Response ist eine Funktion für das Abrufen von technischen Beweisdaten zu einem Archivdatenobjekt. Übergabeparameter ist eine AOID (und ggf. die Versions-ID) des Archivdatenobjekte,s zu dem die Beweisdaten abgerufen werden sollen. Wurde zusätzlich zu einer AOID noch eine VersionID oder mehrere VersionIDs angegeben, muss das ArchiSig-Modul den Evidence Record für diese VersionID bzw. diese VersionIDs zurück liefern. Sofern das `VersionID`-Element nicht angegeben ist, werden die Beweisdaten für die letzte Version des XAIP zurückgeliefert. Als Rückgabe werden bei Angabe des Wertes „`all`“ im `VersionID-Element` die Evidence Records für alle Versionen erwartet. So ist ein lückenloses Prüfen der Authentizität und Integrität seit dem Zeitpunkt der Archivierung des Archivdatenobjektes möglich.

- **ArchiveDeletionRequest / -Response**[3]

  ArchiveDeletionRequest / -Response ist eine Funktion für das Löschen von einem Archivdatenobjekt. Übergabeparameter ist dieAOIDdes Archivdatenobjektes, das gelöscht werden soll. Sofern ein Archivdatenobjekt vor dem Ablauf der eingestellten Aufbewahrungsfrist gelöscht

[2] **[RFC4998]** muss, **[RFC6283]** kann unterstützt werden.

[3] Sollte der verwendete Langzeitspeicher ein Löschen grundsätzlich nicht zulassen, z.B. weil WORM Medien eingesetzt werden, sollte die TR-ESOR-Middleware bei Aufruf dieser Funktion einen entsprechenden Fehler zurück liefern.

werden soll, muss neben dem Namen (Bezeichner) der aufrufenden Instanz ein Begründungstext für diese Aktion übermitteltwerden, der protokolliert wird. Sind von einem Archivdatenobjekt mehrere Versionen vorhanden, sind durch diesen Aufruf alle Versionen zu löschen. Als Rückgabe wird eine Statusmeldung über das erfolgreiche Löschen erwartet.

Darüber hinaus soll eine Implementierung der TR-ESOR-Middleware die folgenden Funktionen unterstützen:

- **ArchiveUpdateRequest / -Response**

  ArchiveUpdateRequest / -Response ist eine Funktion für das Ändern eines bestehenden Archivdatenobjektes im ECM/Langzeitspeicher. Dabei erzeugt jede Änderung zwangsläufig eine neue Version des Archivdatenobjekts. Mit dieser Funktion können Elemente zu einem bereits archivierten Archivdatenobjekt hinzugefügt oder geändert werden. Eine im Sinne dieser TR zulässige Änderung von Archivdatenobjekten ist

  (1) das Hinzufügen einer neuen Version mit weiteren Dokumenten, Daten, Metadaten, kryptographischen Signaturen, Signaturprüfinformationen oder anderen technischen Beweisdaten zu einem Archivdatenobjekt,

  (2) das Ändern von Dokumenten, Daten, Metadaten in einer neuen Version des Archivdatenobjekt,

  (3) das Löschen der Zuordnung von Dokumente, Daten, Metadaten oder Credentials, die in einer vorausgegangenen Version gegeben sind, aus der neuen Version (durch Nicht-Übernahme).[4]

  Als Übergabeobjekt wird ein Delta-XAIP-Container (siehe **[TR-ESOR-F]**) erwartet, der ein neues `versionManifest`, Verweise auf die Vorgängerversion sowie auf unverändert aus dieser übernommenen Objekte und die zu ergänzenden Elemente enthält, die in einer neuen Version eines bereits abgelegten Archivdatenobjektes ergänzt werden sollen. Als Rückgabe wird die neue Versionsnummer des Archivdatenobjektes erwartet. Eine neue AOID wird nicht vergeben – diese bleibt gleich.

- **ArchiveDataRequest / -Response**

  ArchiveDataRequest / -Response ist eine Funktion zum gezielten Auslesen einzelner Datenelemente aus einem Archivdatenobjekt, ohne jeweils die gesamten Archivdatenobjekte an die IT-Anwendung zurückgeben zu müssen. Die Funktion kann beispielsweise zum Aufbau von Suchindizes, zum Ermitteln des Objekteigentümers, zum Ermitteln der Mindestaufbewahrungsfrist oder zum Auslesen von kryptographischen Signaturen genutzt werden.

  Als Übergabeparameter wird die AOID des entsprechenden Archivdatenobjekts erwartet und die Angabe der auszulesenden Datenelemente.Als Rückgabe werden die ausgelesenen Datenelemente erwartet.

- **VerifyRequest / -Response**

  Mit der Funktion `VerifyRequest/-Response` können XML-basierte Archivdatenobjekte (XAIP) samt der darin enthaltenen oder zusätzlich übergebenen beweisrelevanten Daten (digitalen Signaturen, Zeitstempel, Zertifikate, Sperrlisten, OCSP-Responses etc.) und Beweisdaten (Evidence Records) geprüft werden.

  Die Prüfung der Syntax des XAIP Dokumentes erfolgt dabei im ArchiSafe-Modul.

  Die Überprüfung von beweisrelevanten Daten und von Beweisdaten bereits bei der Archivierung ist notwendig, da nicht garantiert werden kann, dass diese Prüfung bis zum Ende der Aufbewahrungsfrist durchgeführt werden kann. Die Obliegenheit der Vertrauensdiensteanbieter, Informationen über die Zurechenbarkeit und Gültigkeit elektronischer Zertifikate elektronisch vorzuhalten, ist zeitlich begrenzt.

  Übergabewert ist ein Eingabe-Element, vorzugsweise ein XAIP, oder ein kryptographisches SignaturObjekt-Element oder beides. In einem XAIP können in der CredentialSection digitale

[4] Die eigentlichen Metadaten/Datenobjekte/Credentials werden hingegen nicht entfernt.

Signaturen bzw. elektronische Zeitstempel gemäß **[eIDAS-VO, Artikel 3 Nr. 33, 34]** sowie ggf. auch ein oder mehrere Evidence Records enthalten sein. Es müssen alle Signaturinformationen (digitalen Signaturen, Zeitstempel, Zertifikate, Sperrlisten, OCSP-Responses etc.) bis hin zu einer vertrauenswürdigen Wurzel geprüft werden. Die hierbei ermittelten Prüfinformationen (Zertifikate, Sperrlisten, OCSP-Responses) werden nach Möglichkeit als unsignierte Attribute bzw. Properties in den entsprechenden kryptogrphischen Signaturen bzw. in die dafür vorgesehenen Datenstrukturen des XAIPs abgelegt.

Die Implementierung der eigentlichen Prüffunktion muss im Krypto-Modul (siehe Anlage **[TR-ESOR-M.2]**) als Komponente der TR-ESOR-Middleware über die in Anlage **[TR-ESOR-S]** beschriebene Schnittstelle S.1 (siehe auch 3.2) erfolgen, um eine entsprechend hohe Vertrauenswürdigkeit zu erreichen. Die für die Prüfung notwendigen Prüfinformationen müssen von den Vertrauensdiensteanbietern abgerufen werden.

3.2 Interne Funktionen

Die folgenden Funktionen sind keine Funktionen, welche die Archiv-Middleware einer Fachanwendung zur Verfügung stellen muss. Es handelt sich um interne Funktionen der TR-ESOR-Middleware, die zum Beweiswerterhalt erforderlich sind.

- **VerifyRequest / -Response**

  VerifyRequest / -Response ist eine Funktion zum Überprüfen von digitale Signaturen bzw. elektronische Zeitstempel gemäß **[eIDAS-VO, Artikel 3 Nr. 33, 34]**, beweisrelevanten Daten (Zertifikaten, Zertifiktsstausinformationen, Zeitstempeln, etc.), Beweisdaten (Evidence Records) und Archivdatenobjekten (XAIPs).. Die Funktion ist notwendig, um beim Archivieren (*ArchiveSubmissionRequest*) die Integrität und Authentizität einer bereits vorhandenen digitalen Signatur zu prüfen und die sich ergebenen Prüfinformationen in in den entsprechenden Signaturen bzw das XAIP einzubetten.[5] Die Überprüfung bereits bei der Archivierung ist notwendig, da nicht garantiert werden kann, dass diese Prüfung bis zum Ende der Aufbewahrungsfrist durchgeführt werden kann. Die Obliegenheit der Vertrauensdiensteanbieter, Informationen über die Zurechenbarkeit und Gültigkeit elektronischer Zertifikate elektronisch vorzuhalten, ist zeitlich begrenzt.

  Übergabewert ist ein Eingabe-Element, vorzugsweise ein XAIP, oder ein kryptographisches SignatureObject-Element (siehe dazu auch Kapitel 4.1) oder beides. In einem XAIP können in den Datenobjekten aber auch in der CredentialSection digitale Signaturen bzw. elektronische Zeitstempel gemäß **[eIDAS-VO, Artikel 3 Nr. 33, 34]** sowie ggf. auch ein oder mehrere Evidence Records enthalten sein (vgl. auch Fußnote 5). Alle kryptographischen Signaturen sind bis hin zu einer vertrauenswürdigen Wurzel zu prüfen.

  Die Implementierung der eigentlichen Prüffunktion muss im Krypto-Modul (siehe Anlage **[TR-ESOR-M.2]**) als Komponente der TR-ESOR-Middleware oder von einem, vom Krypto-Modul aufgerufen, (qualifizierten) Vertrauensdiensteanbieter erfolgen, um eine entsprechend hohe Vertrauenswürdigkeit zu erreichen. Die für die Prüfung notwendigen Prüfinformationen müssen von den Vertrauendiensteanbietern abgerufen werden.

- **SignRequest / -Response (optional)**

  SignRequest / -Response ist eine optionale Funktion zur Anforderung von digitalen Signaturen gemäß **[eIDAS-VO, Artikel 3 Nr. 10, 25]** bei einem Vertrauensdiensteanbieter. Diese Funktion ist notwendig, wenn das ArchiSafe-Modul eine Eingangssignatur an die bei einem *ArchiveSubmissionRequest* übergebenen zu archivierenden Datenobjekte anbringen soll.

  Übergabewert ist ein zu signierendes XAIP (vgl. Fußnote 5 auf Seite 10).

  Die Implementierung der Teil-Funktion „Signatur erstellen“ muss, über die Krypto-Komponente der TR-ESOR-Middleware (siehe **[TR-ESOR-M.2]**) angefordert, beim (qualifizierten) Vertrauensdiensteanbieter erfolgen, um eine entsprechend hohe

[5] Beim Verwenden von proprietären Datenformaten muss hier der jeweilige Produkthersteller eine äquivalente Lösung anbieten.

Vertrauenswürdigkeit zu erreichen. Es müssen für die Teilfunktion „Signatur erstellen“ daher die Dienste eines entsprechenden Dienstleisters genutzt werden.

- **TimestampRequest / -Response**

  TimestampRequest / -Response ist eine Funktion für das Erstellen eines qualifizierten Zeitstempels. Diese Funktion wird für das Übersignieren benötigt.

  Übergabewert ist ein Hashwert der Daten, die mit einem Zeitstempel gesichert werden sollen.

  Die Zeitstempel müssen von einem qualifizierten Vertrauensdiensteanbieter gemäß **[eIDAS-VO, Artikel 3 Nr. 20]** durch das Krypto-Modul (siehe Anlage **[TR-ESOR-M.2]**) als eine Komponente der Middleware angefordert werden.

- **HashRequest / -Response**

  HashRequest / -Response ist eine Funktion zum Errechnen eines Hashwertes aus einem Datenelement. Diese Funktion wird zur Berechnung von Hashwerten für die Hashbaumbildung benötigt.

  Übergabewert ist das Datum (ein Datenelement oder eine Gruppe von Datenelementen), aus dem ein Hashwert berechnet werden soll sowie ggf. ein optionaler Parameter, der den zu verwendenden Hashalgorithmus bezeichnet,.

  Die Implementierung dieser Funktion muss in der Krypto-Komponente der TR-ESOR-Middleware (siehe **[TR-ESOR-M.2]**) erfolgen, um eine entsprechend hohe Vertrauenswürdigkeit zu erreichen.

4. Generische Definition aller Schnittstellenfunktionen

Auf Basis der im vorangegangenen Kapitel rein funktional beschriebenen Anforderungen werden hier die einzelnen Funktionen nun formal auf Basis von ASN.1 spezifiziert. Soweit als möglich wird dabei der OASIS-DSS Standard[6] herangezogen. Die Spezifikationen abstrahieren bewusst von konkreten Implementierungsdetails der in Kapitel 7 des Hauptdokumentes dieser Technischen Richtlinie beschriebenen Referenzarchitektur.

4.1 Allgemeine Datenstrukturen

In diesem Abschnitt werden vorab Datenstrukturen (Datentypen) definiert, die von den nachfolgend beschriebenen Funktionen benutzt werden.

4.1.1 TAPVersion

Der Typ beschreibt die aktuelle Version des vertrauenswürdigen Archivprotokolls (Trusted Archive Protocol, TAP). Die hier angegebene Version 3 bezieht sich dabei auf die Version 1.2 der Technischen Richtlinie.

```
TAPVersion ::= INTEGER { v3(1) }
```

4.1.2 ArchiveData

Der Typ *ArchiveData* enthält im Feld *data* ein Archivdatenobjekt (XAIP-Dokument) und optional im Feld *type* eine Typinformation bzw. eine Referenz auf eine Typinformation (bspw. einen URI oder eine OID auf das zugrunde liegende XML Schema). Ein anderes Datenformat kann vereinbart werden und muss dann in der Typinformation angegeben werden.

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

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

4.1.3 DeltaArchiveData

Der Typ *DeltaArchiveData* enthält im Feld *deltaData* ein Delta-Archivdatenobjekt (DXAIP-Dokument) und optional im Feld *type* eine Typinformation bzw. eine Referenz auf eine Typinformation (bspw. einen URI oder eine OID auf das zugrunde liegende XML Schema).

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

DeltaArchiveDataType ::= CHOICE
{
     oid     OBJECT IDENTIFIER, -- DEFAULT dxaip
     uri     IA5String
}
```

[6] Siehe http://www.oasis-open.org

4.1.4 AOID

Der Datentyp *AOID*[7] ist ein eindeutiges Identifikationsmerkmal für ein Archivdatenobjekt. Für jedes erfolgreich gespeicherte Archivdatenobjekt muss eine *AOID* erzeugt worden sein.

```
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
}
```

Der strukturierte Datentyp *GeneralName* kann dazu genutzt werden, verschiedene Formen von Namen oder Bezeichnern zu kodieren. Die vorgeschlagene Struktur ermöglicht die Bildung lokal definierter Namen oder Bezeichner.

Die ASN.1 Syntax für den Datentyp *GeneralName* ist in **[RFC2459]** definiert.[8]

Das Feld *archiveInfos* kann (optional) zusätzliche Angaben über die Archivumgebung enthalten, bspw. eine Angabe des Zeitpunktes der Archivierung oder einen Identifikator (Adresse) des für die dauerhafte Ablage vorgesehenen Langzeitspeichers.

4.1.5 ArchiveVersionTokenSequence und ArchiveTokenSequence

Der Datentyp ArchiveTokenSequence enthält jeweils ein AOID-Element und optional eine Lise von VersionID(s).

```
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

Die folgende ASN.1- Definition stellt ein Signaturobjekt dar.

```
SignatureObject ::= ContentInfo
ContentInfo ::= SEQUENCE {       -- CMS envelope of the signature
      contentType  ContentType
      content     [0] EXPILCIT ANY DEFINED BY contentType
}
ContentType ::= OBJECT IDENTIFIER  -- identifies the type of signed data
```

Bzgl. der verwendeten Signaturformate gilt **[TR-ESOR-M.2]**, Kap. 5.1.1.

---

[7] Durch die Übergabe eines `AOID`-Elementes beim *ArchiveSubmissionRequest* kann die AOID von der aufrufenden Anwendung vergeben werden. Im Regelfall fehlt dieses Element aber und die AOID wird vom aufgerufenen Modul bereitgestellt. Dann wird die AOID entweder vom ArchiSig-Modul oder vom Langzeitspeicher erzeugt

[8] Siehe unter http://www.ietf.org/rfc/rfc2459.txt

4.1.7 Controls

Das Datenfeld `Controls` ist für optionale Ein- und Ausgabeelemente vorgesehen und **kann** beispielsweise entsprechende Steuerelemente enthalten, die im Rahmen einer Profilierung der vorliegenden Spezifikation definiert werden sollen.

Der Typ kann sowohl bei Anfragen (Request) als auch bei Antworten (Response) verwendet werden.

Der Typ kann u.a dazu benutzt werden, zusätzliche Informationen zu übergeben,

- die den Request/Response über eine ID identifizieren,
- den Request/Response näher qualifizieren,
- zur Identifikation und Authentifikation zwischen dem aufrufenden und dem ausführenden Modul dienen,
- eine Geschäftsanwendungs-interne ID des Datenobjektes oder die von der Geschäftsanwendung vergebene Archidatenobjekt ID (AOID) sowie
- zusätzliche Anweisungen enthalten, die bspw. die zusätzliche Ausführung bestimmter Operationen anfordern.

Das entsprechende Feld darf nicht dazu benutzt werden, die im Hauptdokument und den Anlagen zu dieser technischen Richtlinie beschriebenen sicherheitstechnischen Anforderungen zu umgehen oder zu kompromittieren.

Das ausführende Modul muss jeden *Request* abweisen, der eine unbekannte oder auch nicht unterstützte Anweisung im `Controls` Typ enthält.

Das ausführende Modul muss keine Informationen in einem *Response* zu Aktionen zurückgeben, die nicht im dazugehörigen *Request* definiert wurden.

```
Controls ::= SEQUENCE SIZE (1..MAX) OF Control

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

ControlType ::= OBJECT IDENTIFIER
```

4.1.8 ResponseStatus

Der Typ beschreibt das Ergebnis eines *Requests* und enthält einen Statuscode sowie (optional) einen Statustext.

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

Alternativ kann für die Spezifikation des Ergebnisfeldes auch die Definition des `<Result>` Elementes des OASIS DSS Standards herangezogen werden.

4.2 Definition der Schnittstellenfunktionen

4.2.1 ArchiveSubmissionRequest

Ein *ArchiveSubmissionRequest* übergibt ein Archivdatenobjekt (XAIP) oder ein Datenobjekt in einem beliebigen anderem Format an das aufgerufene Modul. Dieses aufgerufene Modul legt das Datenobjekt direkt im ECM/Langzeitspeicher ab oder übergibt das Datenobjekt an ein weiteres Modul, welches die eigentliche Ablage übernimmt. Der *ArchiveSubmissionRequest* muss mit einer *ArchiveSubmissionResponse* beantwortet werden.

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

(A4.2-1) Gemäß der vorliegenden Spezifikation soll es möglich sein, in den Controls des *ArchiveSubmissionRequests* eine AOID sowie einen oder mehrere Evidence Records zu übergeben bzw. einen Prüfbericht gemäß **[TR-ESOR-VR]** zu dem übergebenen Archivdatenobjekt anzufordern.

(A4.2-2) Im Zuge des Imports von Evidence Records müssen diese von der TR-ESOR-Middleware vollständig geprüft werden. Dies umfasst die im entsprechenden ERS-Standard vorgesehenen Prüfungsschritte[9], wobei die jeweiligen Zertifikate der Zeitstempel vollständig bis hin zu einer vertrauenswürdigen Wurzel geprüft werden müssen.

4.2.2 ArchiveSubmissionResponse

Die Antwort auf einen *ArchiveSubmissionRequest* wird wie folgt definiert:

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

Bestandteil der *ArchiveSubmissionResponse* ist im Falle einer erfolgreichen Ablage im ECM/Langzeitspeicher die Rückgabe einer eindeutigen Archivdatenobjekt ID (AOID), die künftig für alle weiteren Transaktionen, die das Archivdatenobjekt betreffen, vorgelegt werden muss. Für eine mögliche zukünftige Versionierung des Archivdatenobjektes wird die Versionsnummer auf 1 gesetzt. Mit der *ArchiveSubmissionResponse* wird die Version ID nicht mit zurückgeliefert, da sie immer auf 1 gesetzt ist.

(A4.2-3) Für den Fall, dass im *ArchiveSubmissionRequest* die Signaturprüfung fehlgeschlagen ist, kann in den Controls ein Prüfbericht gemäß **[TR-ESOR-VR]** zurückgegeben werden. Die Anforderung eines Prüfberichtes kann in den Controls des *ArchiveSubmissionRequests* gesteuert werden.

4.2.3 ArchiveUpdateRequest

Ein *ArchiveUpdateRequest* übergibt ein „Delta-“Archivdatenobjekt (XAIP) mit den gewünschten Änderungen/Ergänzungen an das aufgerufene Modul, um eine neue Version zu erzeugen. Das aufgerufene Modul in der TR-ESOR Middleware sorgt dafür, dass die Änderungen / Ergänzungen in eine neue Version des Archivdatenobjektes im ECM/Langzeitspeicher übernommen werden. Der *ArchiveUpdateRequest* muss mit einer *ArchiveUpdateResponse* beantwortet werden.

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

(A4.2-4) Gemäß der vorliegenden Spezifikation soll in den Controls des *ArchiveUpdateRequests* die Eingabe einer AOID sowie die Eingabe von Evidence Records bzw. die Eingabe der Anforderung eines Prüfberichts möglich sein.

4.2.4 ArchiveUpdateResponse

Die Antwort auf einen *ArchiveUpdateRequest* wird wie folgt definiert:

[9] Siehe Abschnitt 3.3 in **[RFC4998]** und Abschnitt 2.3 in **[RFC6283]**.

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

VersionID ::= STRING
```

Bestandteil der *ArchiveUpdateResponse* ist im Falle einer erfolgreichen Aktualisierung im Langzeitspeicher die Rückgabe der neuen Versionsnummer des entsprechenden Archivdatenobjektes.

(A4.2-5) Für den Fall, dass im *ArchiveUpdateRequest* die Prüfung der übergebenen Daten fehlgeschlagen ist, kann in den Controls ein Prüfbericht gemäß **[TR-ESOR-VR]** zurückgegeben werden. Die Anforderung eines Prüfberichtes kann in den Controls des *ArchiveSubmissionRequests* gesteuert werden.

4.2.5 ArchiveRetrievalRequest

Ein *ArchiveRetrievalRequest* ist ein Aufruf an das System , um ein Archivdatenobjekt (ein XAIP Dokument) an die aufrufende Geschäftsanwendung zurückzugeben. Das Archivdatenobjekt wird dabei mittels der AOID und ggf. noch mit einer oder mehreren Versionsnummer(n) identifiziert.

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

(A4.2-6) Das ArchiveToken Sequence-Element kann eine Folge von Versions-Identifikatoren enthalten, durch die angegeben wird, welche Versionen des Archivdatenobjektes genau zurückgeliefert werden sollen. Sofern das `VersionID`-Element nicht angegeben ist, werden die zur letzten Version gehörigen Datenobjekte und Verwaltungsinformationen zurückgeliefert. Durch die Angabe von „all“ für die VersionID werden alle existierenden Versionen eines Archivdatenobjektes zurückgeliefert.

(A4.2-7) In den requestControls soll es eine Eingabemöglichkeit geben, dass das zurückgelieferte XAIP den bzw. die entsprechenden Evidence Record(s) enthalten soll.

4.2.6 ArchiveRetrievalResponse

Die Antwort des Systems auf einen *ArchiveRetrievalRequest* wird wie folgt definiert:

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

Das zugehörige Archivdatenobjekt wird bei erfolgreicher Ausführung der Anfrage im optionalen Feld ArchiveData im XAIP-Format zurückgegeben. Im Fehlerfall kann dieses Feld entfallen.

4.2.7 ArchiveEvidenceRequest

Ein *ArchiveEvidenceRequest* dient der Abfrage technischer (kryptographischer) Beweisdaten für die Authentizität und Integrität von einem gespeicherten und durch eine *AOID* eindeutig identifizierten Archivdatenobjekt. Optional kann für das *AOID*-Element noch die Versionsnummer angegeben werden, auf die sich der abzufragende Beweisdatensatz beziehen soll. Wird keine Versionsnummer angegeben, muss sich der Beweisdatensatz auf die aktuelle (letzte) Version beziehen. Durch die Angabe von „`all`“ für die VersionID werden die Evidence Records für alle existierenden Versionen eines Archivdatenobjektes zurückgeliefert.

Ein *ArchiveEvidenceRequest* ist wie folgt definiert:

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

(A4.2-8) In den *requestControls* soll die Eingabe eines Parameters für das Format der abgefragten Beweisdaten möglich sein. Hier sind Formatangaben für Evidence Records gemäß **[RFC4998]** oder **[RFC6283]** die möglichen Werte.

(A4.2-9) Ein konkretes Produkt muss Evidence Records gemäß **[RFC4998]** unterstützen, kann aber auch beide Formate implementieren.[10] Wird in den Controls kein Format angegeben, wird standardmäßig ein Evidence Record gemäß **[RFC4998]** zurückgegeben.

4.2.8 ArchiveEvidenceResponse

Ein *ArchiveEvidenceRequest* wird im Falle einer erfolgreichen Authentifizierung mit einer *ArchiveEvidenceResponse* beantwortet.

Ein *ArchiveEvidenceResponse* wird wie folgt definiert:

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

4.2.8.1 responseData

Das Feld *responseData* gibt für das erfolgreich durch eine *AOID* eindeutig identifiziertes Archivdatenobjekt die angeforderten Evidence Records zurück. Das Format dieser Evidence Records richtet sich nach dem entsprechenden Parameterwert im Request. Das *status* Feld beschreibt das Ergebnis des *ArchiveEvidenceRequest* und enthält einen Statuscode sowie (optional) einen Statustext.

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

VersionID ::= String

EvidenceData ::= SEQUENCE{
     credentialID       String OPTIONAL,
     relatedObjects     SEQUENCE OF String OPTIONAL,
     evidenceRecord     CHOICE
                        { ersEvidenceRecord EvidenceRecord,
                          xmlEvidenceRecord XMLEvidenceRecord }
}
```

Das Feld `credentialID` enthält die ID des Evidence Records, insbesondere dann, wenn dieser in einem Archivdatenobjekt eingefügt wird und über diese ID ansprechbar sein soll.

Das Feld `relatedObjects` enthält eine Liste der IDs der Datenelemente des Archivdatenobjektes, über die der Evidence Records eine Aussage zur Integrität macht.

Die Definition des Datentyps `EvidenceRecord` findet sich in **[RFC4998]** und im Kapitel 5.5 der Anlage **[TR-ESOR-F]**. Die Definition des Datentyps `XMLEvidenceRecord` ist definiert im Internet Draft „Extensible Markup Language Evidence Record Syntax“ **[RFC6283]**

[10] Für eine Prüfsoftware, die nicht Teil dieser TR ist, ist es sehr empfehlenswert, dass diese beide Formate implementiert.

4.2.9 ArchiveDeletionRequest

Die Funktion *ArchiveDeletionRequest* ermöglicht das Löschen eines archivierten Datenobjektes.[11] Ein *ArchiveDeletionRequest* muss sich auf ein gespeichertes und durch ein *AOID*-Element eindeutig identifizierte Archivdatenobjekt beziehen. Der *ArchiveDeletionRequest* muss mit einer *ArchiveDeletionResponse* beantwortet werden, der zuverlässig Auskunft über das (erfolgreiche) Löschen gibt.

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

4.2.9.1 requestControls

(A4.2-10) Das Feld *requestControls* muss im Falle eines *ArchiveDeletionRequests* vor Ablauf der für das jeweilige Archivdatenobjekt gesetzten Aufbewahrungsfrist eine Begründung (*ReasonOfDeletion*) für das Löschen enthalten. Ansonsten muss der Löschauftrag abgelehnt werden.

(A4.2-11) Die Begründung soll wenigstens folgende Angaben enthalten:

- ein eindeutiges Identifikationsmerkmal der aufrufenden Anwendung in *requestorName*
- und einen protokollierbaren Begründungstext in *requestInfo*.

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

4.2.10 ArchiveDeletionResponse

Die Antwort auf einen Löschauftrag, *ArchiveDeletionResponse*, ist wie folgt definiert:

```
ArchiveDeletionResponse ::= SEQUENCE{
     version            TAPVersion DEFAULT v3,
     status             ResponseStatus,

     responseControls   [0] Controls OPTIONAL
}
```

Im Feld *ResponseStatus* wird der aufrufenden Anwendung das Ergebnis des Löschauftrages zurückgegeben. Im Fehlerfalle kann das Element *Controls* eine ausführliche Fehlermeldung enthalten.

4.2.11 ArchiveDataRequest

Das Auslesen von diskreten Datenelementen eines Archivdatenobjektes, *ArchiveDataRequest*, ist wie folgt definiert:

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

4.2.11.1 requestData

Das Feld *requestData* enthält in diesem Fall die AOID des Archivdatenobjektes, von dem einzelne Datenelemente ausgelesen werden sollen. Zusätzlich müssen die abzufragenden Datenelemente angegeben werden. Dies erfolgt über *dataLocation*. Jedes auszulesende Datenelement (Datenknoten) des Archivdatenobjektes muss eindeutig identifiziert werden. In *dataLocation* ist hier der komplette und eindeutige Pfad innerhalb der Baumstruktur des Archivdatenobjektes oder ein eindeutiger Abfrageausdruck[12] anzugeben.

[11] Natürlich kann diese Funktion keine Daten auf Backup-Medien löschen. Sollten diese Daten aus bestimmten Gründen ebenfalls zu vernichten sein, muss dies anderweitig gelöst werden.

Die explizite (optionale) Angabe einer Versions ID ist bei dieser Funktion nicht notwendig, da über das Feld *dataLocation* auch auf Datenelemente einzelner Versionen direkt zugegriffen werden kann. Es können so auch Datenelemente mehrerer Versionen eines Archivdatenobjektes gleichzeitig abgefragt werden.

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

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

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

4.2.12 ArchiveDataResponse

Ein *ArchiveDataRequest* wird im Falle einer erfolgreichen Authentifizierung mit einer *ArchiveData-Response* beantwortet.

Ein *ArchiveDataResponse* wird wie folgt definiert:

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

4.2.12.1 responseData

Das Feld *responseData* gibt die Daten im Feld *value* zurück, die erfolgreich aus dem Archivdatenobjekt ausgelesen werden sollten und konnten. Das Rückgabefeld *value* kann dabei auch BASE64 kodierte Binärdaten enthalten.

Im Feld *location* muss der entsprechende, in der Suchanfrage übergebene Identifizierungsparameter (dort der Typ *locationValue*) wieder mit zurück geliefert werden.

Konnte das gesuchte Datenelement nicht identifiziert und ausgelesen werden, entfällt *value*.

```
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

Ein V*erifyRequest* übergibt ein Archivdatenobjekt (XAIP-Dokument) samt der darin enthaltenen oder zusätzlich übergebenen beweisrelevanten Daten (Signaturen, Siegel, Zeitstempel, Zertifikate, Sperrlisten, OCSP-Responses etc.) und Beweisdaten (Evidence Records) zur Verifikation.[13]

[12] Ausgehend von einem XML-basierten Archivdatenobjekt bieten sich für die diskrete Adressierung von XML-Datenelementen hier XPath (siehe unter: http://www.w3.org/TR/2007/REC-xpath20-20070123/), XQuery (siehe unter: http://www.w3.org/TR/2007/REC-xquery-20070123/) oder die XML Pointer Language XPointer (siehe unter: http://www.w3.org/TR/2003/REC-xptr-framework-20030325/) an. Über ein *requestControl* kann spezifiziert werden, welche Abfragemethode genau verwendet wird.

Neben dem ArchiveData-Element kann dabei auch ein Sequenz von SignatureObjects (SignatureObjectSequence) übergeben werden, in der eigenständige Signaturen (detached signatures) zur Prüfung übergeben werden können. Wenn Signaturen bereits im ArchiveData-Element enthalten sind, kann die optionale Sequenz von SignatureObjects (SignatureObjectSequence) entfallen.

Im `SignatureObject` kann auch ein `Evidence Record`-Element übergeben werden, um die entsprechende Prüfung des Evidence Record anzustoßen. In diesem Fall müssen die Attribute `AOID` und `VersionID` vorhanden sein und das zugehörige `XAIP`-Element muss im requestData übergeben werden.

Sofern das `signatureObjectSequence`-Element fehlt, muss genau ein requestData-Element vorhanden sein, das die zu prüfenden Signaturobjekte enthält.

Ein *VerifyRequest* wird wie folgt definiert:

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

(A4.2-12) In den Controls soll insbesondere die Eingabe von Steuererlementen eines Policy-Elementes bzw. die Anforderung eines Prüfberichtes gemäß **[TR-ESOR-VR]** möglich sein.

(A4.2-13) Es müssen alle Signaturinformationen (Signaturen, Siegel, Zeitstempel, Zertifikate, Sperrlisten, OCSP-Responses etc.) bis hin zu einer vertrauenswürdigen Wurzel geprüft werden. Die hierbei ermittelten Prüfinformationen (Zertifikate, Sperrlisten, OCSP-Responses) werden als unsignierte Attribute bzw. Properties in den entsprechenden Signaturen bzw. in den dafür vorgesehenen Datenstrukturen in der credentialSection des XAIP Dokumentes abgelegt.

(A4.2-14) Sofern in der `credentialSection` des übergebenen XAIP-Containers ein oder mehrere `xaip:EvidenceRecord`-Elemente gemäß **[TR-ESOR-F]** enthalten sind, müssen diese entsprechend geprüft werden.

4.2.13.1 requestControls

Das *requestControls* Feld kann

- Angaben über den Signaturtyp,
- das Signaturformat und
- Prüfoptionen enthalten, bspw. auf eine vordefinierte Policy oder eine Liste von Parametern verweisen, die das Vorgehen bei der Prüfung signierter Objekte bestimmen und zur Prüfung der Signaturen herangezogen werden müssen,
- oder zusätzliche Aktionen definieren, die vom Krypto-Modul durchzuführen sind, bspw. für die Rückgabe der Prüfergebnisse den in der eCard-API des BSI spezifizierten Prüfbericht **[TR-ESOR-VR]** bzw. **[eCard-2]** zu nutzen.

4.2.14 VerifyResponse

Die Antwort auf einen *VerifyRequest* wird wie folgt definiert:

[13] Wird statt des XAIP Formats ein proprietäres Format von einem Produkt genutzt, kann an dieser Stelle auch nur dieses Format übergeben werden. Es ist dann anderweitig sicher zu stellen, dass eine äquivalente Funktionalität gegeben ist.

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

4.2.14.1 responseData

Das Feld *responseData* kann das XAIP-Dokument mit den in die dafür vorgesehenen XML-Elemente (siehe **[TR-ESOR-F]**) eingetragenen Prüfinformationen enthalten. Alternativ können hier auch nur die Prüfinformationen zurückgeliefert werden, die dann vom aufrufenden Modul in die XAIP-Datenstruktur in die dafür vorgesehenen Elemente oder in alternative Datenformate eingetragen werden müssen.

4.2.15 SignRequest

Ein *SignRequest* übergibt ein Archivdatenobjekt (XAIP-Dokument) zur Erstellung einer digitalen Signatur.[14]

Ein *SignRequest* wird wie folgt definiert:

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

4.2.15.1 requestControls

Das *requestControls* Feld kann dazu benutzt werden, zusätzliche Informationen zu übergeben,

- die Angaben über den Signaturtyp (z.B. XML-Signatur, CMS-Signatur oder PDF-Signatur),
- das Signaturformat sowie
- weitere Optionen enthalten, die bei der Erstellung der Signatur herangezogen werden müssen oder zu beachten sind,
- oder zusätzliche Aktionen definieren, die durchzuführen sind, wie bspw. mit der Ausführung der Signatur zugleich auch die zugrundeliegenden Signaturzertifikate bis hin zu einem vertrauenswürdigen Wurzelzertifikat zu prüfen und die Prüfergebnisse in die dafür vorgesehenen Abschnitte des XAIP-Dokumentes (siehe **[TR-ESOR-F]**) einzutragen.

4.2.16 SignResponse

Die Antwort auf einen *SignRequest* wird wie folgt definiert:

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

4.2.16.1 responseData

Das Feld *responseData* enthält im Erfolgsfall das signierte XAIP-Dokument mit den in die dafür vorgesehenen XML-Elemente (siehe **[TR-ESOR-F]**) eingetragenen Signaturdaten bzw. ein alternatives Datenformat inkl. der erzeugten Signatur.

[14] Wird statt des XAIP Formats ein proprietäres Format von einem Produkt genutzt, kann an dieser Stelle auch nur dieses Format übergeben werden. Es ist dann anderweitig sicher zu stellen, dass eine äquivalente Funktionalität gegeben ist.

Alternativ können hier auch nur die Signaturdaten zurückgeliefert werden, die dann vom aufrufenden Modul in die XAIP-Datenstruktur in die dafür vorgesehenen Elemente oder in alternative Datenstrukturen eingetragen werden müssen.

4.2.17 TimestampRequest

Ein *TimestampRequest* übergibt einen Hashwert zur Erstellung bzw. zur Anforderung eines Zeitstempels über diesen Hashwert.

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

4.2.17.1 requestData

Das Feld *requestData* enthält den Hashwert, der mit einem Zeitstempel versehen werden soll, und eine Information über den verwendeten Hashalgorithmus (siehe **[RFC3161]**).

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

4.2.17.2 requestControls

Das optionale Feld *requestControls* kann dazu benutzt werden, zusätzliche Informationen zu übergeben,

- die Angaben über den Zeitstempeltyp und das Zeitstempelformat (**[RFC3161]**-Zeitstempel oder **[RFC4998]** bzw. **[RFC6283]**-Archivzeitstempel) sowie
- zusätzliche Anweisungen enthalten, die bspw. bestimmen, dass ein Archivzeitstempel auf Basis von **[RFC4998]** bzw. **[RFC6283]** eine qualifizierte Zeitstempel gemäß **[eIDAS VO, Artikel 41 and 42]** sein muss, oder
- die zusätzlich bei der Erstellung des Zeitstempels herangezogen werden müssen oder zu beachten sind.

4.2.18 TimestampResponse

Die Antwort auf einen *TimestampRequest* wird auf Basis von **[RFC3161]** wie folgt definiert:

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

4.2.18.1 responseData

Das Feld *responseData* enthält ein *TimeStampToken* gemäß **[RFC3161]** und muss enthalten sein, wenn bei der Erstellung kein Fehler aufgetreten ist.

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

Die entsprechende Typdefinition zu *id-signedData* und *SignedData* finden sich in der CMS Spezifikation **[RFC3852]**. Das *TimeStampToken* darf keine anderen Signaturen als die Signatur des Zeitstempeldiensteanbieters enthalten. Zugehörige Zertifikatsinformationen müssen durch die Signatur ge-

schützt sein. Darüber hinausgehende Prüfinformationen zum Zeitstempel müssen in den nichtsignierten Feldern des *TimeStampToken* abgelegt werden.[15]

4.2.19 HashRequest

Ein *HashRequest* übergibt an das Modul einen binären Wert, von dem der Hashwert berechnet werden soll. Auch XML-Daten werden so übergeben, auch wenn es um das Erzeugen einer eingebettete XML-Signatur oder ähnliches geht.

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

4.2.19.1 requestData

Die zu hashenden Daten werden im binären Format übergeben. Die Erstellung der binären Darstellung bleibt dem Aufrufer der Funktion überlassen.

4.2.19.2 requestControls

Das *requestControls* Feld kann dazu benutzt werden, zusätzliche Informationen zu übergeben, die

- Angaben über den zu verwendenden Hashalgorithmus machen.

4.2.20 HashResponse

Die Antwort auf einen *HashRequest* wird wie folgt definiert:

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

4.2.20.1 responseData

Das Feld *responseData* enthält den berechneten Hashwert. Der Hashwert wird im `MessageImprint` Format, gemäß **[RFC3161]**, zurückgegeben. Die Rückgabe kann leer sein, wenn beim Erzeugen des Hashwertes ein Fehler aufgetreten ist.

[15] Hierfür sollen die Felder certificates und crls genutzt werden, die in der Struktur `SignedData` im `TimeStampToken` enthalten sind.

5. Konkrete Definition der Schnittstellen

Die Spezifikationen der Schnittstellen enthalten keine Definitionen von Sicherheitsmechanismen auf der Vermittlungs-, Transport- oder Anwendungsschicht, sondern setzen den Einsatz zuverlässiger Sicherheitsmechanismen wie z. B. **[RFC4346]** oder **[RFC2401]** für die gegenseitige Authentisierung der Kommunikationsendpunkte voraus.

Diejenigen Module, die eine der nachfolgend aufgeführten Schnittstellen anbieten, müssen die hier aufgeführten Funktionen der Schnittstelle komplett und vollständig umsetzen.

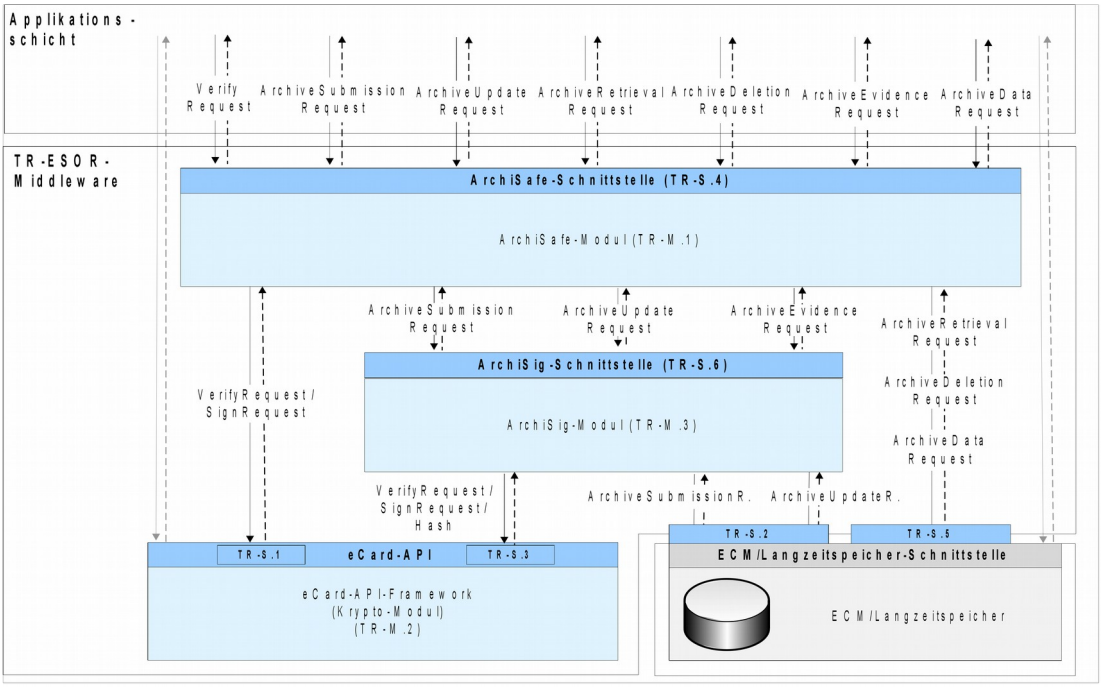

Abbildung 2: Funktionen der Schnittstellen

Abbildung 2 zeigt, welche Schnittstellen welche Funktionen anbieten müssen.

Falls nicht anderes angegeben, ist die Spezifikation der einzelnen Schnittstellenfunktionen identisch zu den entsprechenden Funktions- und Typ-Spezifikationen in Kapitel 4.

5.1 Schnittstelle S.1

Vornehmlicher Zweck der Schnittstelle **TR-ESOR-S.1** zwischen dem ArchiSafe-Modul und dem Krypto-Modul ist das Verifizieren und Erstellen digitale Signaturen, die den zu archivierenden elektronischen Daten (XAIP-Dokumenten) beigefügt wurden oder beigefügt werden sollen. Die Schnittstelle muss folgende Funktionen umsetzen:

5.1.1 VerifyRequest

Siehe Kapitel 4.2.13.

5.1.2 VerifyResponse

Siehe Kapitel 4.2.14.

5.1.3 SignRequest

Siehe Kapitel 4.2.15.

5.1.4 SignResponse

Siehe Kapitel 4.2.16.

5.2 Schnittstelle S.2

Vornehmlicher Zweck der Schnittstelle **TR-ESOR-S.2** zwischen dem ArchiSig-Modul und dem ECM/Langzeitspeicher ist es, dem ArchiSig-Modul die notwendigen lesenden und schreibenden Zugriffe sowohl auf den ArchiSig-eigenen Datenbestand als auch auf den Archivdatenbestand im ECM/Langzeitspeicher zu ermöglichen.

Die Schnittstelle S.2 kann davon ausgehen, dass

- Informationen und Daten im ECM/Langzeitspeicher weder vorsätzlich noch fahrlässig in unzulässiger Weise manipuliert werden können,
- Abgerufene Daten aus dem ECM/Langzeitspeicher die bitgenaue Reproduktion der ursprünglich archivierten Nutzdaten (insbesondere der Primärdaten) darstellen,
- Veränderungen an den ursprünglich archivierten Daten nachweislich protokolliert werden[16],
- die Integrität und Verfügbarkeit (Schutz vor Verlust und Zerstörung)[17] der gespeicherten Informationen und Daten nicht kompromittiert werden können.

(A5.2-1) Für die Aufbewahrung der Archivdatenobjekte und der kryptographischen Beweisdaten muss das ArchiSig-Modul über eine (oder mehrere) sichere und performante Schnittstelle(n) zu einem (oder mehreren) vertrauenswürdigen elektronischen ECM/Langzeitspeicher(n) verfügen.

(A5.2-2) Die Schnittstelle S.2 muss durchsetzen, dass bei der Ablage von neu zu archivierenden Archivdatenobjekten entweder eine bereits generierte AOID mit übergeben werden kann, über welche später ein Zugriff auf das Archivdatenobjekt möglich ist, oder vom ECM/Langzeitspeicher eine AOID vor der eigentlichen Ablage erzeugt und an das aufrufende Modul übergeben wird.[18]

5.2.1 Die Schnittstelle zur Speicherung der Archivdatenobjekte

(A5.2-3) Die Speicherung der Archivdatenobjekte muss so erfolgen, dass der Zusammenhang zwischen dem gespeicherten Archivdatenobjekt und der zugehörigen Archivdatenobjekt-ID (AOID) jederzeit technisch nachvollziehbar bleibt.

Die Schnittstelle muss folgende Funktionen bezüglich der Behandlung von Archivdatenobjekten bereitstellen:

- eine Funktion zum Speichern eines Archivdatenobjektes und der zugehörigen Archivdatenobjekt ID (AOID),
- eine Funktion zum lesenden Zugriff auf die Archivdatenobjekte. Dies ist notwendig, um im Falle eines drohenden Verlustes der Sicherheitseigenschaften der eingesetzten Hashalgorith-

[16] Der Nachweis soll über aussagekräftige Protokolle und Log-Dateien erfolgen, die grundsätzlich außerhalb des für die Speicherung der Archivdatenobjekte vorgesehenen Speichersystems abgelegt und vorgehalten werden sollen. Die Dauer der Aufbewahrung der Protokolle und Log-Dateien ist in einer technischen Organisationsrichtlinie festzulegen.

[17] Für den Betrieb des ECM/Langzeitspeichers wird die Prüfung und Bescheinigung der Ordnungsmäßigkeit zum dauerhaften Nachweis der Vollständigkeit, der Echtheit, der Unverfälschtheit und der Verfügbarkeit der elektronisch abgelegten und aufbewahrten Informationen und Daten auf der Grundlage einer ausführlichen Systemdokumentation durch einen unabhängigen Gutachter empfohlen.

[18] Durch die Übergabe eines `AOID`-Elementes beim *ArchiveSubmissionRequest* kann die AOID von der aufrufenden Anwendung vergeben werden. Im Regelfall fehlt dieses Element aber und die AOID wird vom aufgerufenen Modul bereitgestellt. Dann wird die AOID entweder vom ArchiSig-Modul oder vom Langzeitspeicher erzeugt. Im letzten Fall muss die AOID vom ECM/Langzeitspeicher erzeugt und an ArchiSig übergeben werden, damit ArchiSig die neue AOID in das Archivdatenobjekt einfügen und dann über die schützenswerten Bestandteile dieses Objektes einen Hashwert berechnen kann. Erst danach wird das Archivdatenobjekt tatsächlich im ECM/Langzeitspeicher abgelegt.

men die Hashwerte der gespeicherten Archivdatenobjekte und damit die Hashbäume neu berechnen zu können.

Die Schnittstelle soll folgende Funktionen bezüglich der Behandlung von Archivdatenobjekten bereitstellen:

- eine Funktion zum Aktualisieren bereits archivierter Archivdatenobjekte

5.2.1.1 ArchiveSubmissionRequest

Grundsätzlich siehe Kapitel 4.2.1.

Ein *ArchiveSubmissionRequest* der Schnittstelle S.2 übergibt ein Archivdatenobjekt und zusätzlich optional die zugehörige Archivdatenobjekt ID (AOID) an den elektronischen ECM/Langzeitspeicher, wenn diese nicht vom ECM/Langzeitspeicher selbst, sondern vorab vom ArchiSig-Modul generiert wurde.

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

5.2.1.2 ArchiveSubmissionResponse

Siehe Kapitel 4.2.2.

5.2.1.3 ArchiveUpdateRequest

Siehe Kapitel 4.2.3.

5.2.1.4 ArchiveUpdateResponse

Siehe Kapitel 4.2.4.

5.2.1.5 ArchiveRetrievalRequest

Siehe Kapitel 4.2.5.

5.2.1.6 ArchiveRetrievalResponse

Siehe Kapitel 4.2.6.

5.2.2 Die Schnittstelle zur Speicherung der kryptographischen Beweisdaten

Die aufgeführte Liste von Funktionen zur Speicherung kryptographischer Beweisdaten ist nur eine mögliche Ausprägung von Funktionen dieser Schnittstelle. Daher wurde auf eine formale Spezifikation dieser Schnittstelle, z.B. in ASN.1 Notation, bewusst verzichtet. Der für die Beweisdatensicherung herangezogene Speicher und seine internen Datenstrukturen sowie seine Leistungsfähigkeit hinsichtlich der oben genannten Funktionen können von äußerst unterschiedlicher Ausprägung sein. Eine allgemein gültige Schnittstellenspezifikation wäre deshalb nur auf einem sehr hohen Abstraktionsniveau möglich.

Die Speicherung der kryptographischen Beweisdaten sollte so erfolgen, dass

- jeder zu einem Archivdatenobjekt nativ erzeugte und gespeicherte Hashwert eindeutig mit der zu dem Archivdatenobjekt gehörenden Archivdatenobjekt ID (AOID) und VersionID verknüpft wird und
- jederzeit technisch nachvollziehbar bleibt, welche Hashwerte in welcher Baumstruktur unter einem Archivzeitstempel subsumiert werden.

Diese Schnittstelle sollte daher (je nach technischer Ausprägung) folgende Funktionen bereitstellen:

- eine Funktion zur zuverlässigen Speicherung eines über ein Archivdatenobjekt berechneten Hashwertes H sowie der zugehörigen Archivdatenobjekt ID (AOID),
- eine Funktion zur Abfrage einer Liste von Hashwerten, die noch nicht mit einem Archivzeitstempel gemäß des ERS-Standards gesichert wurden,

- eine Funktion zur zuverlässigen Speicherung von gemäß des ERS-Standards erzeugten oder erneuerten Archivzeitstempeln,
- eine Funktion zur Abfrage einzelner oder mehrerer Archivzeitstempel, für die eine Signaturerneuerung nach § 15 des Vertrauensdienstegesetzes aufgrund eines drohenden Verlustes der Sicherheitseigenschaften der diesen Archivzeitstempeln zugrunde liegenden kryptographischen Algorithmen und Parameter erforderlich ist,
- eine Funktion zur Abfrage eines technischen Beweisdatensatzes (Evidence Record) gemäß des ERS-Standards zum Nachweis der Authentizität und Integrität eines durch seine Archivdatenobjekt ID (AOID) eindeutig bestimmten Archivdatenobjektes.

5.3 Schnittstelle S.3

Vornehmlicher Zweck der Schnittstelle **TR-ESOR-S.3** zwischen dem ArchiSig-Modul und dem Krypto-Modul ist das Erzeugen von Hashwerten sowie das Erzeugen und Verifizieren von qualifizierten Zeitstempeln. Beide Datenarten werden für den Aufbau der Merkle-Hashbäume **[MER 1980]** benötigt.

(A5.3-1) Für die Erzeugung von Hashwerten und die Anforderung /den Abruf und die Verifikation von qualifizierten Zeitstempeln muss das ArchiSig-Modul über eine sichere und performante Schnittstelle auf ein kryptographisches Modul zugreifen können, dass mindestens die in der Anlage **[TR-ESOR-M.2]** dieser TR beschriebenen obligatorischen Anforderungen erfüllt.

Diese Schnittstelle muss die folgenden Funktionen bereitstellen:

- Eine Funktion zur Erzeugung von qualifizierten Zeitstempeln bei einem qualifizierten Vertrauensdiensteanbieter, über die Krypto-Komponente der TR-ESOR-Middleware **[TR-ESOR-M.2]** angefordert .
- Eine Funktion zur Verifikation der digitalen Signatur von qualifizierten Zeitstempeln
- eine Funktion zur Berechnung eines Hashwertes

5.3.1 TimestampRequest

Siehe Kapitel 4.2.17.

5.3.2 TimestampResponse

Siehe Kapitel 4.2.18.

5.3.3 VerifyRequest

Siehe Kapitel 4.2.13.

5.3.4 VerifyResponse

Siehe Kapitel 4.2.14.

5.3.5 HashRequest

Siehe Kapitel 4.2.19.

5.3.6 HashResponse

Siehe Kapitel 4.2.20.

5.4 Schnittstelle S.4

Die Schnittstelle **TR-ESOR-S.4** soll es den Geschäftsanwendungen ermöglichen, in einer einheitlichen und funktionalen Weise auf das ECM-/Langzeitspeichersystem zuzugreifen. Die Schnittstelle soll weiterhin unberechtigte Zugriffe auf das ECM-/Langzeitspeichersystem zuverlässig verhindern.

(A5.4-1) Die Schnittstelle **TR-ESOR-S.4** muss mindestens die folgenden Funktionen bereitstellen:

- Eine Funktion zur sicheren und zuverlässigen Ablage von Archivdatenobjekten
- Eine Funktion zum Abruf von Archivdatenobjekten (im XAIP Format)
- Eine Funktion zum Abruf technischer (kryptographischer) Beweisdaten
- Eine Funktion für das Löschen archivierter Daten

Die Schnittstelle **TR-ESOR-S4** soll folgende Funktionen bereitstellen:

- Eine Funktion zur Aktualisierung bereits archivierter Archivdatenobjekte
- Eine Funktion zum Abruf von Datenelementen einzelner Archivdatenobjekte
- Eine Funktion zur Verifikation eines Archivdatenobjekt (XAIP-Dokument) samt der darin enthaltenen oder zusätzlich übergebenen beweisrelevanten Daten (Signaturen, Siegel, Zeitstempel, Zertifikate, Sperrlisten, OCSP-Responses etc.) und Beweisdaten (Evidence Records)

(A5.4-2) Die TR-ESOR-Middleware kann auch (teilweise) auf Basis von asynchronen Aufrufen realisiert werden.[19]

5.4.1 ArchiveSubmissionRequest

Siehe Kapitel 4.2.1.

5.4.2 ArchiveSubmissionResponse

Siehe Kapitel 4.2.2.

5.4.3 ArchiveUpdateRequest

Siehe Kapitel 4.2.3.

5.4.4 ArchiveUpdateResponse

Siehe Kapitel 4.2.4.

5.4.5 ArchiveRetrievalRequest

Siehe Kapitel 4.2.5.

5.4.6 ArchiveRetrievalResponse

Siehe Kapitel 4.2.6.

5.4.7 ArchiveEvidenceRequest

Siehe Kapitel 4.2.7.

5.4.8 ArchiveEvidenceResponse

Siehe Kapitel 4.2.8.

5.4.9 ArchiveDeletionRequest

Siehe Kapitel 4.2.9.

5.4.10 ArchiveDeletionResponse

Siehe Kapitel 4.2.10.

5.4.11 ArchiveDataRequest

Siehe Kapitel 4.2.11.

5.4.12 ArchiveDataResponse

Siehe Kapitel 4.2.12.

[19] Vgl. auch **[TR-ESOR-F]**, Kapitel 6.

5.4.13 VerifyRequest

Siehe Kapitel 4.2.13.

5.4.14 VerifyResponse

Siehe Kapitel 4.2.14.

5.5 Schnittstelle S.5

Die Schnittstelle TR-ESOR.S-5 ermöglicht die Zugriffe vom ArchiSafe-Modul aus auf den ECM/Langzeitspeicher, die keine fachlichen Abhängigkeiten zu den kryptographisch gesicherten Beweisdaten besitzen.

(A5.5-1) Die Schnittstelle **TR-ESOR-S.5** muss mindestens die folgenden Funktionen bereitstellen:

- Eine Funktion zum Abruf von Archivdatenobjekten (im XAIP Format),
- eine Funktion für das Löschen archivierter Daten und
- eine Funktion zum Abruf von Datenelementen einzelner Archivdatenobjekte

(A5.5-2) Die Schnittstelle S.5 muss sicher stellen, dass

- Informationen und Daten im ECM/Langzeitspeicher weder vorsätzlich noch fahrlässig in unzulässiger Weise manipuliert werden können,
- abgerufene Daten die bitgenaue Reproduktion der ursprünglich archivierten Nutzdaten (insbesondere der Primärdaten) darstellen,
- das Löschen von archivierten Daten nachweislich protokolliert wird, was insbesondere auch das Protokollieren der Begründung bei vorzeitigem Löschen mit einschließt[20] und
- die Integrität, Verfügbarkeit (Schutz vor Verlust und Zerstörung) und die Revisionsfähigkeit[21] der gespeicherten Informationen und Daten nicht kompromittiert werden können.

5.5.1 ArchiveRetrievalRequest

Siehe Kapitel 4.2.5.

5.5.2 ArchiveRetrievalResponse

Siehe Kapitel 4.2.6.

5.5.3 ArchiveDeletionRequest

Siehe Kapitel 4.2.9.

5.5.4 ArchiveDeletionResponse

Siehe Kapitel 4.2.10.

5.5.5 ArchiveDataRequest

Siehe Kapitel 4.2.11.

5.5.6 ArchiveDataResponse

Siehe Kapitel 4.2.12.

[20] Der Nachweis soll über aussagekräftige Protokolle und Log-Dateien erfolgen, die grundsätzlich außerhalb des für die Speicherung der Archivdatenobjekte vorgesehenen Speichersystems abgelegt und vorgehalten werden sollen. Die Dauer der Aufbewahrung der Protokolle und Log-Dateien ist in einer technischen Organisationsrichtlinie festzulegen.

[21] Für den Betrieb des ECM/Langzeitspeichers wird die Prüfung und Bescheinigung der Ordnungsmäßigkeit des Verfahrens zum dauerhaften Nachweis der Vollständigkeit, der Echtheit, der Unverfälschtheit und der Verfügbarkeit der elektronisch abgelegten und aufbewahrten Informationen und Daten auf der Grundlage einer ausführlichen Systemdokumentation durch einen unabhängigen Gutachter empfohlen.

5.6 Schnittstelle S.6

Die Schnittstelle **TR-ESOR-S6** dient dem Zugriff des ArchiSafe Moduls auf das ArchiSig Modul.

Über die hier beschriebene Schnittstelle **TR-ESOR-S.6**, über die eine Archivierung von (neuen) Archivdatenobjekten möglich ist, kann das ArchiSig-Modul direkt beim Archivierungsvorgang eingebunden werden. Die sichernden Hashwerte können so unmittelbar erzeugt werden. Ein Umgehen dieser Sicherheitsfunktion ist damit ausgeschlossen.

(A5.6-1) Für die Ablage insbesondere digital signierter Dokumente und den Abruf kryptographischer Beweisdaten zum technischen Nachweis der Authentizität und Integrität der im elektronischen ECM/Langzeitspeicher aufbewahrten Archivdatenobjekte muss das ArchiSafe-Modul über eine sichere und performante Schnittstelle auf ein ArchiSig-Modul zugreifen können, dass mindestens die in der Anlage TR-ESOR-M.3 dieser TR beschriebenen Anforderungen erfüllt.

Diese Schnittstelle muss die folgenden Funktionen bereitstellen:

- eine Funktion zur Ablage von Archivdatenobjekteneine Funktion zum Abruf von Evidence Records nach dem ERS-Standard **[RFC4998]** bzw. **[RFC6283]**

Die Schnittstelle soll folgende Funktionen bereitstellen:

- Eine Funktion zur Aktualisierung bereits archivierter Archivdatenobjekte

5.6.1 ArchiveSubmissionRequest

Grundsätzlich siehe Kapitel 4.2.1.

Ein *ArchiveSubmissionRequest* der Schnittstelle S.6 übergibt ein Archivdatenobjekt und zusätzlich optional die zugehörige Archivdatenobjekt ID (AOID) an das ArchiSig-Modul, wenn diese nicht vom ArchiSig-Modul sondern vorab vom ArchiSafe-Modul generiert wurde.

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

5.6.2 ArchiveSubmissionResponse

Siehe Kapitel 4.2.2.

5.6.3 ArchiveUpdateRequest

Siehe Kapitel 4.2.3.

5.6.4 ArchiveUpdateResponse

Siehe Kapitel 4.2.4.

5.6.5 ArchiveEvidenceRequest

Siehe Kapitel 4.2.7.

5.6.6 ArchiveEvidenceResponse

Siehe Kapitel 4.2.8.

6. Sicherheitstechnische Anforderungen

Im Folgenden werden die sicherheitstechnischen Anforderungen für alle Schnittstellen aus Kapitel 5 aufgeführt.

(A6.1-1) Der Zugriff auf das Ziel-Modul darf erst nach einer erfolgreichen gegenseitigen Authentifizierung zwischen dem Quell-Modul und dem Ziel-Modul (ggf. der aufrufenden Fachanwendung) erfolgen. Die Authentifizierung ist für jeden Aufruf zu wiederholen, alternativ kann ein sicherer Tunnel aufrecht erhalten werden.

Ein *Request* oder *Response* darf nicht ausgeführt werden, wenn die Authentifizierung zwischen der aufrufenden und dem ausführenden Modul auf der Vermittlungs-, Transport- oder Anwendungsschicht nicht möglich oder nicht erfolgreich war.

(A6.1-2) Die wechselseitige Authentisierung muss kryptographisch ausreichend sein, so dass ein unbemerkter Austausch einzelner Komponenten nicht möglich ist.

(A6.1-3) Das Replay einer Authentisierung oder anderer Nachrichten darf nicht möglich sein.

(A6.1-4) Ein unberechtigter Zugriff auf Authentisierungs- oder Nutzdaten während der Kommunikation muss zuverlässig verhindert werden. Die Schnittstelle muss so implementiert werden, dass auch Blockierungen (DoS) oder Folgefehler, wie Buffer Overflow oder SQL-Injections ausgeschlossen werden können.