**XMLERS - Description and Implementation on Java**

**Description**

**1. LTANS Overview.**

LTANS stands for Long-Term Archive and Notary Service. It began from an informational RFC ([2]) to respond to requirements of long-term data archiving. A raw data object standing alone is not meaningful, there must be a scheme or method that can specify information about that data object. The associated information of a data object is called meta data. There is not a common format for meta data, it depends on the realm or the context in which the data object is being used. Besides associated information, an archiving data object must be provided a scheme (method) by which some specific functional requirements must be satisfied ([2], section 4). Table 1 lists brief of technical requirements for a data archiving system according to [2]. The details explanation of technical requirements for a data archiving system will be represented in another document (ltans-tech-req.docx)

|  |  |
| --- | --- |
| **Accessing Requirements** | |
| **Requirements** | **Descriptions** |
| Submittable | Permit clients to request to submit data objects for archive. |
| Retrievable | Permit clients to request to retrieve archived data objects |
| Delete-able | Permit clients to request to delete archived data objects. |
| **Policy Requirements** | |
| **Requirements** | **Descriptions** |
| Maintenance policy | Ability of archived data object maintenance.  There should be rules for following operational aspects:  - Preservation activity triggers.  - Default archival period.  - Default handling upon expiration of archival period.  There should be rules for mechanism-specific:  - Where cryptographic mechanisms are employed.  - Policy for cryptographic maintenance. |
| Authorization policy | The entities permitted to exercise services should be defined. The entities including who is permitted to submit, retrieve, or manage specific archived data objects. |
| Service policy | Service policy defines the types of provided services, including:  - Acceptable data types.  - Description of requests that may be accepted.  - Deletion procedure. |
| **Management Requirements** | |
| **Requirements** | **Descriptions** |
| Period requestable | Permit clients to request the operation of specifying an archival period for submitted data objects. |
| Period extendable or shorten-able | Permit clients to request the operation of extending or shorten the archival period for an archived data object. |
| Metadata specify-able | Permit clients to request the operation of specifying the metadata associated with an archived data object. |
| Policy specify-able | Permit clients to request the operation of specifying an archive policy under which the submitted data should be handled. |
| **Evidence Requirements** | |
| **Requirements** | **Descriptions** |
| Providing evidence | Capability of providing evidence that can be used to demonstrate the integrity of data for which it is responsible in the validated archival period. |
| Supporting non-repudiation | - Supporting non-repudiation of data existence, integrity, and origin.  - LTA need NOT be capable of providing ALL evidence necessary to produce non-repudiation proof, in some cases, should NOT be trusted to provide ALL necessary information. |
| **Confidentiality Requirements** | |
| **Requirements** | **Descriptions** |
| Archive Data Object Confidentiality | Providing a means for accepting encrypted data such that future preservation activities apply to the original, unencrypted data. |
| Submitter – Service Confidentiality | Confidentiality between the submitter and LTANS. |
| **Service to Service Requirements** | |
| **Requirements** | **Descriptions** |
| Supporting archive-to-archive data transfer | Supporting transfer of archived data objects, evidence and evidence records from one service to another (From one LTANS to another LTANS). |
| **Groups of Data Requirements** | |
| **Requirements** | **Descriptions** |
| Enable operations upon groups of data objects. | - Support submission of groups of data objects.  - Submitter should be able to indicate which data objects belong together |

***Table 1***

LTANS

LTANS

Figure 1: LTANS correlation.

Request

Response

**Submitter**

Mutual TLS

Request / Response

Mutual TLS

Mutual TLS

Request

Response

Data Object

**+**

**ERS Processing**

ASN.1 format

Timestamped data

Data Object

**+**

**XMLERS Processing**

XML format

Timestamped data

Certificate for TimeStamping

Certificate for TimeStamping

**TSA Server**

**CRL or OCSP Responder**

Verify signed data

(optional)

Verify signed data

(optional)

Meta Data

Signature

(optional)

Meta Data

Signature

(optional)

**From Archive Requestor**

**From Archive Requestor**

Certificate for Digital Signature

Certificate for Digital Signature

**Notary Server**

**Notarized data**

**Notarized data**

**Database**

**Archive Server**

Figure 2: A typical LTANS architecture

Digital sign

Digital sign

**TSP**

End entity certificates for notary server

End entity certificates for TSA Server

**Note:**

TSP: Trust Service provider

TSA: Time Stamping Authority

ERS: Evidence Record Syntax

XML: Extensible Markup Language

XMLERS: XML Evidence Record Syntax

**2. What is ERS?**

ERS stands for Evidence Record Syntax. It is a broadened and generalized version of RFC3126. These two RFCs (Request for Comments) are both for non-repudiation purposes. RFC3126 supports handling just ONE data object, whereas RFC4998 supports handling LARGE SETS of data objects.

In order to easily understand ERS, it is needed to explain how a single data object is handled according to RFC3126.

**2.1. Time Stamping Protocol.**

|  |  |  |
| --- | --- | --- |
| **Possible forms of electronic signature** [9] | | |
| **ES** | **ES-T** | **ES-C** |
| Include the digital signature and other basic information provided by signer. | ES with Time-Stamp, which adds a time-stamp to the electronic signature. | ES with complete validation data, which adds to the ES-T references to the complete set of data supporting the validity of the ES (i.e., revocation status information). |

Table 2

Signature Policy ID

Other Signed Attributes

Digital Signature

Time-Stamp over Digital Signature

Complete certificate and revocation references

ES

ES-T

ES-C

***Note that Digital Signature is DIFFERENT from Electronic Signature. Digital Signature is one of the three mandatory components of Electronic Signature: Digital Signature, Signature Policy ID and Other Signed Attributes.***

An ES-T with its time-stamp can be used as evidence indicating that the associated datum existed before a particular time specified by time-stamp, and the owner of datum, who signed the datum, cannot repudiate the existence of it. The operation of time-stamping processing, which formed an ES-T, is a process of interacting with TSA (Time Stamping Authority).

Time Stamping Authority

(TSA)

Time-Stamping Requestor

(Signer)

TimeStampReq

TimeStampResp

**TST**

1. Check errors.

2. Verify:

+ Fields contained in TST

+ Digital Signature in TST

+ The objects be timestamped.

+ Certificate id of TSA.

+ Data imprint.

+ Hash algorithm OID

+ Timeliness of the response.

+ Value of nonce

Reject TST

failed

OCSP or CRL

TSA’s certificate

has been revoked ?

**TST**: TimeStampToken

The TSA MUST sign each time-stamp message with a key reserved specifically for that purpose, and the corresponding certificate MUST contain extended key usage field with KeyPurposeID having value id-kp-timeStamping.

Figure 3: Common structure of an X.509 (2019) v3 Certificate

(see [13], section 7 and Annex A for details)

Certificate

TBSCertificate

Subject Public Key Info

**Note:**

TBSCertificate: to-be-signed public key certificate

Other Key Purpose: Extended Key Purpose may be defined by any organization with a need, its Id MUST be assigned in

accordance with IANA or ITU-T (see [14], [15])

Algorithm

Public Key params

Extensions

Authority Key Identifier

Subject Key Identifier

Key Identifier

Authority Cert Serial Number

Authority Cert Issuer

Key Usage

Digital Signature

Content Commitment

Key Encipherment

Key Agreement

Data Encipherment

Key CertSign

CRL Sign

Encipher Only

Decipher Only

Version

Serial Number

Signature Algorithm

Issuer

Validity

Subject

Issuer Unique Identifier

Subject Unique Identifier

Extended Key Usage

Server Auth.

Client Auth.

Other Key Purpose

Code Signing

Email Protection

Time Stamping

OCSP Signing

…

Private Key Usage Period

Certificate Policies

CPS Pointer Qualifier

Subject Alternate Name

Signature Algorithm

Signature Value

Policy Mapping

Issuer Alternate Name

Basic Constraints

Policy Constraints

Name Constraints

CRL Distribution Points

Inhibit Any Policy

Authorization Validation

CRL number

Authority Info Access

Subject Info Access

Issuing Distribution Point

CRL Scope

Certificate Issuer

Hold Instruction Code

Invalidity Date

Reason Code

Ordered List

Prot. Restrict

Subject Directory Attributes

Associated Information

Inhibit Any Policy

Delta Info

Status Referrals

CRL Stream Identifier

To Be Revoked

Revoked Group

Expired Certs On CRL

Delta CRL Indicator

Base Update Time

Freshest CRL

Subject Alt. Public Key Info.

Alt. Signature Algorithm

Alt. Signature Value

AA Issuing Distribution Point

Figure 3 illustrates an overview of X.509 certificate, it can be considered as an electronic signature (ES). The details of its components will be presented in another work (x509-v3-cert-review.docx). For time-stamping protocol according to RFC3126, three parts of X.509 certificate will be focused: Time Stamping (in Extended Key Usage), Authority Info Access and Subject Info Access. ~~If subject is an end entity as in figure 2 (end entity certificate created by TSA Server), Subject Info Access will be focused. Otherwise, if the subject is a certificate authority, Authority Info Access will be considered.~~

Extended Key Usage

(OID: 2.5.29.37)

Server Auth.

(OID: 1.3.6.1.5.5.7.3.1 )

Client Auth.

(OID: 1.3.6.1.5.5.7.3.2 )

Code Signing

(OID: 1.3.6.1.5.5.7.3.3 )

Email Protection

(OID: 1.3.6.1.5.5.7.3.4 )

Code Signing

(OID: 1.3.6.1.5.5.7.3.3 )

Time Stamping

(OID: 1.3.6.1.5.5.7.3.8 )

OCSP Signing

(OID: 1.3.6.1.5.5.7.3.9 )

Others

(OID: Assigned by IANA or ITU-T )

**Note:**

id-ce-extKeyUsage OBJECT IDENTIFIER ::= {id-ce 37} -- ce : certificate extension

id-kp-timeStamping OBJECT IDENTIFIER ::= {id-kp 8} -- kp: key purpose

id-ce OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 29}

id-kp OBJECT IDENTIFIER ::= {id-pkix 3} -- pkix: PKI using X.509

id-pkix OBJECT IDENTIFIER ::= {iso(1) identified-organization(3) dod(6) internet(1) security(5)

mechanisms(5) pkix(7)}

Figure 4: X509 v3 Extended Key Usage

(See [14], section 4.2.1.12)

**Note:**

id-pe-subjectInfoAccess OBJECT IDENTIFIER ::= {id-pe 11} -- pe: PKIX certificate extensions

id-pe OBJECT IDENTIFIER ::= {id-pkix 1}

id-ad-timeStamping OBJECT IDENTIFIER ::= {id-ad 3} -- ad: access descriptor

id-ad OBJECT IDENTIFIER ::= {id-pkix 48}

id-pkix OBJECT IDENTIFIER ::= {iso(1) identified-organization(3) dod(6) internet(1) security(5)

mechanisms(5) pkix(7)}

URI: Uniform Resource Identifier (rfc3986)

rfc822Name: Email address in IA5String format (ISO/IEC 646)

dNSName: Domain name in IA5String format

iPAddress: IP address in octet string format

id-ad-\* can be (up to July 2014):

id-ad-ocsp OID: 1.3.6.1.5.5.7.48.1

id-ad-caIssuers OID: 1.3.6.1.5.5.7.48.2

id-ad-timestamping OID: 1.3.6.1.5.5.7.48.3

id-ad-dvcs OID: 1.3.6.1.5.5.7.48.4

id-ad-caRepository OID: 1.3.6.1.5.5.7.48.5

id-ad-http-certs OID: 1.3.6.1.5.5.7.48.6

id-ad-crls OID: 1.3.6.1.5.5.7.48.7

id-ad-xkms OID: 1.3.6.1.5.5.7.48.8

id-ad-signedObjectRepository OID: 1.3.6.1.5.5.7.48.9

id-ad-rpkiManifest OID: 1.3.6.1.5.5.7.48.10

id-ad-signedObject OID: 1.3.6.1.5.5.7.48.11

id-ad-cmc OID: 1.3.6.1.5.5.7.48.12

Figure 5: X509 v3 Subject Info Access

(See [14] sec. 4.2.2.2, [15] sec. 3.33)

Subject Info Access

(OID: 1.3.6.1.5.5.7.1.11)

Access Description

Access Method

(OID: 1.3.6.1.5.5.7.48.3)

URI (for http-based, ftp-based services)

rfc822Name (for email-based service)

dNSName or iPAddress (for TCP/IP based service)

Access Location

Access Description

Access Method

(OID: 1.3.6.1.5.5.7.48.x)

URI

rfc822Name

dNSName

Access Location

iPAddress

directoryName

**…**

Figure 6: X509 v3 Authority Info Access

(See [14] sec. 4.2.2.1)

**Note:**

id-pe- authorityInfoAccess OBJECT IDENTIFIER ::= {id-pe 1}

id-pe OBJECT IDENTIFIER ::= {id-pkix 1}

id-pkix OBJECT IDENTIFIER ::= {iso(1) identified-organization(3) dod(6) internet(1) security(5)

mechanisms(5) pkix(7)}

URL: Uniform Resource Locators (rfc1738) - See [16] sec. 3; [17] sec. 1.1.3)

(URL is a subset of URI)

id-ad-ocsp OID: 1.3.6.1.5.5.7.48.1

id-ad-caIssuers OID: 1.3.6.1.5.5.7.48.2

Authority Info Access

(OID: 1.3.6.1.5.5.7.1.1)

Access Description

Access Method

(OID: 1.3.6.1.5.5.7.48.1)

Access Location

Access Description

Access Method

(OID: 1.3.6.1.5.5.7.48.2)

Access Location

URL (OCSP Responder Location)

URI

dNSName

directoryName

**…**

**Certificate for TimeStamping**

Extensions

Extended Key Usage

Time Stamping

(OID: 1.3.6.1.5.5.7.3.8 )

Subject Info Access

(OID: 1.3.6.1.5.5.7.1.11)

Access Description

Access Method

(OID: 1.3.6.1.5.5.7.48.3)

URI

rfc822Name

dNSName or iPAddress

Access Location

Authority Info Access

(OID: 1.3.6.1.5.5.7.1.1)

Access Description

Access Method

(OID: 1.3.6.1.5.5.7.48.1)

Access Location

URL

(OCSP Responder Location)

**TimeStamp Requestor**

TBSData

**Trusted Service Provider**

**Verifier**

Certificate for what purpose?

Certificate for what subject?

Who verifies the certificate?

Generate end entity certificate for TSA Server

Timestamp request

Figure 7: X509 v3 certificate for Time Stamping

Timestamped data

**rfc3126**

Return to the requestor

Request to verify

Respond the result

**2.2. Electronic Signature Structure**

Compared to short-term archiving, long-term archiving may encounter some risks due to many reasons which can occur in long period of time. Especially, when data needs to be linked together, the risk increases. For example, a signed data object needs to connect to verifier to verify the signature bearing on it, but it cannot access to. In another case, the type of keys or hash algorithm used in a specific electronic signature may be compromised and they are no longer in used. For those reasons, a type of Electronic Signature has been approved: ES-X, which is extended type of ES-C (see table 2). ES-X is divided into three types as described below:

Signature Policy ID

Other Signed Attributes

Digital Signature

Time-Stamp over Digital Signature

Complete certificate and revocation references

ES

ES-T

ES-C

Complete certificate and revocation data

ES-X Long

Figure 8: Electric Signature ES-X Long

All the associated revocation status information

Full CA-chain from root CA to parent CA of signer’s certificate

signer’s certificate

TS for overall ES-C

TS for DS

Signature Policy ID

Other Signed Attributes

Digital Signature

Time-Stamp over Digital Signature

Complete certificate and revocation references

Time-stamp over CES

ES

ES-T

ES-C

ES-X 1

Figure 9: Electric Signature ES-X Type 1

Signature Policy ID

Other Signed Attributes

Digital Signature

Time-Stamp over Digital Signature

Complete certificate and revocation references

ES

ES-T

ES-C

Time-stamp over Complete certs and revoc. refs

ES-X 2

Figure 10: Electric Signature ES-X Type 2

TS for DS

TS for complete cert. and revoc. refs

Before explaining the details of ES-X, it is necessary to take an overview of major parties which are involved in a business transaction supported by electronic signatures. This also makes illustrating figures 8, 9, 10 to be clearer.

|  |  |
| --- | --- |
| **Signer** | An entity that initially creates the electronic signature. |
| **Verifier** | An entity that verifies EVIDENCE. In the context of this work, it is an entity that validates an electronic signature. |
| **The Arbitrator** | An entity which arbitrates disputes between a signer and verifier when there is a disagreement on the validity of a digital signature. |
| **Trusted Service Provider**  **(TSPs)** | One or more entities that help to build trust relationships between the signer and verifier. |

Table 3

**Verifier**

Electronic Signature

create

validate

Figure 11: Major parties participate in the business transaction supported by Electronic Signatures

Have disputes between Signer and Verifier?

**Arbitrator**

yes

public key certificates (2)

register (1)

Certificate Revocation List (CRL)

(to be published)

**TSPs**

Certification Authorities (CAs)

Repository Authorities (e.g., Directory)

Time-Stamping Authorities (TSAs)

Online Certificate Status Protocol Responder

(OCSP Responder)

Attribute Authorities

Signature Policy Issuers

Registration Authorities (RAs)

issue

publish

Cross-certificates

issue

Signatures policies

(Defined by each of specific businesses)

Technical / procedural requirements for ES

Verification / Validation

issue

**Signer**

**Users**

attributes linked to public key certificates

attest that some data was formed before a given trusted time

cert. status

unknown

revoked

not revoked

check status

check status

cert. status

unknown

revoked

not revoked

no

Back to figures 8, 9, and 10, an ES by itself does not satisfy the non-repudiation requirement due to lack of time point at which the data is signed. So, whether or not the ES has Timestamp created by signer, it MUST be sent to TSA via a timestamp request to check if it has timestamp which indicates the time point at that the signature of the signer is created. Of course, a specific error for timestamp is acceptable due to the delay of timestamping, the tolerance for timestamp depends on the context of electronic signature is being used. The diagram by which an ES be timestamped is as following:

ES-T

**TSA**

Timestamp Response

yes

Has Electric Signature had Timestamp yet ?

PKIStatusInfo

rfc3161. Sec. 2.4.2

no

**+**

TimeStampToken

yes

**Signer**

Create

Electronic Signature

Figure 12: ???

Figure 13: Signer – TSA interacting via Request / Response

**Signer**

**TSA**

**Timestamp request**

(rfc3161. sec. 2.4.1 )

version (1)

messageImprint

hashAlgorithm

hashMessage

reqPolicy (optional)

nonce (optional)

certReq

extensions (optional)

**Timestamp response**

(rfc3161. sec. 2.4.2 )

**statusInfo**

statusString ::= PKIFreeText (Optional)

status ::= INTEGER

granted (0)

grantedWithMods (1)

waiting (3)

rejection (2)

revocationWarning (4)

revocationNotification (5)

failInfo ::= PKIFailureInfo (Optional)

badAlg (0)

badRequest (2)

badDataFormat (5)

timeNotAvailable (14)

unacceptedPolicy (15)

unacceptedExtension (16)

addInfoNotAvailable (17)

systemFailure (25)

**timeStampToken ::= ContentInfo**

(optional)

id-signedData

(oid: 1.2.840.113549.1.7.2)

SignedData

version

1

5

4

3

2

digestAlgorithms

SET OF Alg. IDs

encapContentInfo

eContent

eContentType

id-ct-TSTInfo

(oid: 1.2.840.113549.1.9.16.1.4)

DER-encoded TSTInfo

version (1)

policy

messageImprint

genTime

serialNumber

tsa

accuracy

ordering

none

extensions

+ Check permission

(is a subscriber?)

+ Check hashAlgorithm

+ Check data format

+ Check TSA Policy

+ Check nonce

+ Check certReq

+ Check the extensions

Apply specific modifications

Have something not valid or warning

All be valid

Waiting

check statusInfo

status = 1

status = 0

Decide to reject TST ?

Check nonce

Reject TST

Accept TST

yes

no

match

not match

Check failInfo, statusString and decide suitable processing

Mutual TLS

|  |  |  |
| --- | --- | --- |
| **timeStampReq** | | |
| **Fields** | | **Meanings** |
| version | | The version of the Time-Stamp Request, currently be 1 |
| mesageImprint | hashAlgorithm | A hash algorithm which is applied to hash the datum. It SHOULD be a known hash algorithm that is recognized by TSA (see figure 15). |
| hashedMessage | The hash of the datum to be time-stamped. |
| reqPolicy | | Indicate the TSA policy under which the TimeStampToken (TST) SHOULD be provide |
| nonce | | A random big integer (at least 64 bits) that allows the client (signer) to verify the timeliness of the response when no local clock is available. The client sends a nonce to TSA via timeStampReq then TSA MUST send back the same nonce value via the corresponding timeStampResp, otherwise that timeStampResp shall be rejected. The nonce is an optional, but it SHOULD be added to the transaction (Signer - TSA) for relay attack resistance. |
| certReq | | - If certReq field is present and set to true, the TSA’s public key certificate that is referenced by the ESSCertID (or ESSCertIDv2) identifier inside a SigningCertificate attribute in the response MUST be provided by the TSA in the certificates field from the SignedData structure in that response. That field may also contain other certificates.  - If the certReq field is absent or set to false then the certificates field from the SignedData structure MUST NOT be present in the response. (see figure 16) |
| extensions | | The extensions field is a generic way to add additional information to the request in the future. Extensions is defined in [10]. If an extension, whether it is marked critical or not critical, is used by a requester (signer / client) but is not recognized by a time-stamping server, the server SHALL not issue a token and SHALL return a failure (unacceptedException). |

Table 4

|  |  |  |
| --- | --- | --- |
| **timeStampResp – statusInfo – status** (see [22], sec. 3.2.3; [8], Appendix C) | | |
| **Values** | **Meanings** | |
| granted (0) | Indicate that the timeStampToken (TST) contained in timeStampResp is exactly what requester asked for | A TimeStampToken MUST be present in TimeStampResp |
| grantedWithMods (1) | Indicate that the timeStampToken, with modifications; the requester is responsible for ascertaining the differences rejection. |
| rejection (2) | Indicate that the requester DOES NOT get TST, more information is returned in failInfo and statusString. | A TimeStampToken MUST NOT be present in TimeStampResp |
| waiting (3) | Indicate that the request body part has not been processed yet, expect to hear more later. |
| revocationWarning (4) | Indicate that a revocation is imminent. |
| revocationNotification (5) | Indicate that a revocation has occurred. |

Table 5

|  |  |
| --- | --- |
| **timeStampResp – statusInfo – failInfo** (see [22], sec. 3.2.3; [8], Appendix C) | |
| **Values** | **Meanings** |
| badAlg (0) | Unrecognized or unsupported Algorithm Identifier. |
| badRequest (2) | Transaction not permitted or supported. |
| badDataFormat (5) | The data submitted has the wrong format. |
| timeNotAvailable (14) | The TSA’s time source is not available |
| unacceptedPolicy (15) | The requested TSA policy is not supported by the TSA |
| unacceptedExtension (16) | The requested extension is not supported by the TSA |
| addInfoNotAvailable (17) | The additional information requested could not be understood or is not available. |
| systemFailure (25) | The request cannot be handled due to system failure. |

Table 6

Figure 14: time-stamping request details

TimeStampReq

(rfc3161 sec. 2.4.1)

version ::= INTEGER

messageImprint

nonce ::= INTEGER (64 bits)

(optional)

hashAlgorithm ::= AlgorithmIdentifier (\*)

reqPolicy ::= OBJECT IDENTIFIER

(optional)

certReq ::= BOOLEAN

(default false)

extensions [0] ::= SEQUENCE SIZE (1 .. MAX) OF Extension (optional)

Extension

extnID ::= OBJECT IDENTIFIER

critical ::= BOOLEAN

(default false)

extnValue ::= OCTET STRING

hashedMessage ::= OCTET STRING

**Note:**

(\*): Refer to figure 15

1

**Note:**

(\*) MD5 and SHA-1 algorithms are not considered secure ([13], page 158)

(\*\*) hashAlgs: joint-iso-itu-t(2).country(16).us(840).organization(1).gov(101).csor(3).nistAlgorithms(4).hashalgs(2)

Example: OID: hashAlgs.2 implies OID: 2.16.840.1.101.3.4.2.2

DigestAlgorithm/HashAlgorithm(ITU-T X.509 (2019) Annex B, NIST.FIPS.180-4 sec.6)

id-sha1 (OID: 1.3.14.3.2.26)

SHA-2 family (\*\*)

id-sha256

(OID: hashAlgs.1)

id-sha512

(OID: hashAlgs.3)

id-sha384

(OID: hashAlgs.2)

id-sha224

(OID: hashAlgs.4)

id-sha512-224

(OID: hashAlgs.5)

id-sha512-256

(OID: hashAlgs.6)

id-sha3-224

(OID: hashAlgs.7)

SHA-3 family (\*\*)

id-shake128

(OID: hashAlgs.11)

id-sha3-256

(OID: hashAlgs.8)

id-sha3-384

(OID: hashAlgs.9)

id-sha3-512

(OID: hashAlgs.10)

id-shake256

(OID: hashAlgs.12)

id-shake256-len

(OID: hashAlgs.18)

ShakeOutputLen ::= INTEGER

(output length in bits)

Figure 15: Digest Algorithms / Hash Algorithms recommended by ITU-T X.509 and NIST.FIPS 180-4

SHA-1 (\*)

Alg-SHAKE256-LEN

id-shake128-len

(OID: hashAlgs.17)

ShakeOutputLen ::= INTEGER

(output length in bits)

Alg-SHAKE128-LEN

OID: 1.2.840.113549.2.5

MD5 (\*)

Figure 16: time-stamping response details (see [8] sec. 2.4.2, [19] sec. 5)

status

failInfo ::= BIT STRING (optional)

TimeStampResp

(rfc3161 sec. 2.4.2)

status ::= INTEGER

granted (0)

grantedWithMods (1)

rejection (2)

waiting (3)

revocationWarning (4)

revocationNotification (5)

statusString ::= SEQUENCE SIZE (1 .. MAX) OF UTF8String (optional)

badAlg (0)

badRequest (2)

badDataFormat (5)

timeNotAvailable (14)

unacceptedPolicy (15)

unacceptedExtension (16)

addInfoNotAvailable (17)

systemFailure (25)

timeStampToken

(optional)

id-signedData

(OID: 1.2.840.113549.1.7.2)

digestAlgorithms ::= SET OF DigestAlgorithmIdentifier

signedData

version ::= INTEGER

1

5

4

3

2

encapContentIno

eContentType = id-ct-TSTInfo

(OID: 1.2.840.113549.1.9.16.1.4)

eContent = DER-encoded TSTInfo

version ::= INTEGER

1

policy ::= OBJECT IDENTIFIER (see figure 17)

messageImprint

hashAlgorithm ::= AlgorithmIdentifier

hashedMessage ::= OCTET STRING

serialNumber ::= INTEGER (up to 160 bits)

genTime ::= GeneralizedTime (in YYYYMMDDhhmmss[.s…]Z format - expressed as UTC time)

accuracy

seconds ::= INTEGER

(optional)

millis [0] ::= INTEGER (1 .. 999)

(optional)

micros [1] ::= INTEGER (1 .. 999)

(optional)

ordering ::= BOOLEAN (default false)

nonce ::= INTEGER (optional) -- MUST equal to the value provided in the TimeStampReq

tsa [0] ::= GeneralName (optional) -- give a hint in identifying the name of TSA (rfc5816, sec. 2.2.2)

(optional)

actual identification of the entity that signed the response

ESSCertIDv2

hashAlgorithm ::= AlgorithmIdentifier (default: id-sha256)

certHash ::= OCTET STRING (certHash = hash(DER-encoded(certificate including signature))

)

issuerSerial

issuer :: GeneralNames

serialNumber ::= CertificateSerialNumber (INTEGER)

ESSCertID

certHash ::= OCTET STRING (certHash = SHA-1(DER-encoded(certificate including signature))

)

issuerSerial

issuer :: GeneralNames

serialNumber ::= CertificateSerialNumber (INTEGER)

extensions [1] ::= SEQUENCE SIZE (1 .. MAX) OF Extension (optional)

Extension

extnID ::= OBJECT IDENTIFIER

critical ::= BOOLEAN

(default false)

extnValue ::= OCTET STRING

certificates ::= CertificateSet (optional) (see figure 18)

crls ::= RevocationInfoChoices (optional) (see figure 19)

signerInfos ::= SET OF SignerInfo (see figure 20)

See [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

timeStampResp-timeStampToken-signedData-encapContentInfo-eContent-policy

Figure 17: details of timeStampResp – timeStampToken-signedData – encapContentInfo – eContent - policy (in progress …)

timeStampResp-timeStampToken-signedData-certificates ::= SET OF CHOICES { .. }

Figure 18: details of timeStampResp – timeStampToken – signedData - certificates

TSA’s public key certificate (refer to figure 3)

X509Version

Serial Number

Signature Algorithm

Issuer

Validity

Subject

Subject Public Key Info

Issuer Unique Identifier (v2 or v3)

(optional)

Subject Unique Identifier (v2 or v3)

(optional)

Extensions (v3)

(optional)

extendedCertificate (obsolate)

extendedCertficateInfo

CMSVersion

Certificate

**Note:**

X509Version ::= ENUMERATED {1, 2, 3}

CMSVersion ::= ENUMERATED {1, 2, 3, 4, 5}

GeneralizedTime format: YYYYMMDDhhmmss[.s…]Z ) - refer to [13] sec. 9 and annex E for details

attributes ::= UnauthAttributes (SET SIZE (1 .. MAX) OF Attribute)

attrType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY

attrValues ::= SET OF AttributeValue

attributeCertificateV1 (obsolate) (refer to figure 18-a)

signatureValue ::= BIT STRING

signatureAlgorithm ::= SignatureAlgorithmIdentifier (refer to figure 21)

attributeCertificateV2 (refer to figure 18-b)

other ::= OtherCertificateFormat

otherCertFormat ::= OBJECT IDENTIFIER

otherCert ::= ANY DEFINED BY otherCertFormat

attributeCertificateV1 (definition from X.509-1997) – (obsolate)

attributeCertificateInfoV1

version ::= INTEGER

0

subject ::= CHOICE {…}

baseCertID [0] ::= IssuerSerial

(associate with a Public Key Certificate)

subjectName [1] ::= GeneralNames

(associated with a name)

issuer ::= GeneralNames

signatureAlgorithmIdentifier (see figure 21)

certificateSerialNumber ::= INTEGER

attCertValidityPeriod

notBeforeTime :: GeneralizedTime

notBeforeTime :: GeneralizedTime

attributes ::= SEQUENCE OF { .. }

attribute

attrType ::= OBJECT IDENTIFIER

attrValue ::= ANY DEFINED BY attrType

issuerUniqueID (optional)

extensions (optional)

signatureAlgorithm ::= SignatureAlgorithmIdentifier

signatureValue ::= BIT STRING

Figure 18-a: details of timeStampResp-timeStampToken-signedData-certificates – attributeCertificateV1

attributeCertificateV2 (definition from X.509-2000)

attributeCertificateInfoV2

version ::= INTEGER

1

holder ::= SEQUENCE { .. }

baseCertID [0] ::= IssuerSerial ::= SEQUENCE { .. } (optional)

(the issuer and serial number of the holder’s Public Key Certificate)

issuer ::= GeneralNames

certificateSerialNumber ::= INTEGER

entityName [1] ::= GeneralNames (optional)

(the name of the claimant or role)

issuerUID ::= UniqueIdentifier (optional)

objectDigestInfo ::= SEQUENCE { .. } (optional)

digestObjectType ::= ENUMERATED { .. }

otherObjectType (2)

publicKeyCert (1)

publicKey (0)

otherObjectTypeID ::= OBJECT IDENTIFIER (optional)

digestAlgorithm ::= AlgorithmIdentifier (refer to figure 15)

objectDigest ::= BIT STRING

issuer ::= GeneralNames (IssuerName)

signatureAlgorithmIdentifier (refer to figure 21)

certificateSerialNumber ::= INTEGER

attCertValidityPeriod

notBeforeTime :: GeneralizedTime

notBeforeTime :: GeneralizedTime

attributes ::= SEQUENCE OF { .. }

attribute

attrType ::= OBJECT IDENTIFIER

attrValue ::= ANY DEFINED BY attrType

issuerUniqueID (optional)

extensions (optional)

signatureAlgorithm ::= SignatureAlgorithmIdentifier (refer to figure 21)

signatureValue ::= BIT STRING

**Note:**

See [13] sec.9.3.2.1 (page 38) for GeneralNames explanation.

Figure 18-b: Details of timeStampResp-timeStampToken-signedData-certificates-attributeCertificateV2

timeStampResp-timeStampToken-signedData-crls ::= RevocationInfoChoices ::= SET OF CHOICE { .. }

Figure 19: Details of timeStampResp-timeStampToken-signedData-crls

certificateList ::= SIGNED (certificateListContent)

certificateListContent ::= SEQUENCE { .. }

signatureAlgorithm ::= SignatureAlgorithmIdentifier (refer to figure 21)

signatureValue ::= BIT STRING

version

(if present, version shall be 2))

signatureAlgorithmIdentifier ::= SignatureAlgorithmIdentifier

issuer ::= Name ::= SEQUENCE OF RelativeDistinguishedName

(see [40] sec. 9 – page 23 for details)

thisUpdate ::= Time ::= CHOICE { .. }

utcTime ::= UTCTime

generalTime ::= GeneralizedTime

**Note:**

UTCTime format: YYMMDDHHMMSSZ

GeneralizedTime format: YYYYMMDDhhmmss[.s…]Z)

(Refer to [19] sec. 11.3 page 41 for details)

CARLs: Certification Authority Revocation Lists

EPRLs: End-Entity Public Key Certificate Revocation Lists

(Refer to [13] sec. 9 and annex E for details)

nextUpdate ::= Time ::= CHOICE { .. } (optional)

utcTime ::= UTCTime

generalTime ::= GeneralizedTime

revokedCertificates ::= SEQUENCE OF SEQUENCE { .. } (optional)

certificateSerialNumber ::= INTEGER

revocationDate

utcTime ::= UTCTime

generalTime ::= GeneralizedTime

crlEntryExtensions ::= Extensions (including CARLs and EPRLs which are defined for CRLs) (optional)

(refer to [13]; figure 22 for details of CRL Entry Extension )

crlExtensions [0] ::= Extensions (optional) (see [13] sec. 9.5; figure 23 for details of CRL Extension)

otherRevocationInfoFormat [1] ::= SEQUENCE { .. }

otherRevInfoFormat ::= OBJECT IDENTIFIER

otherRevInfo ::= ANY DEFINED BY otherRevInfoFormat

timeStampResp-timeStampToken-signedData-signerInfos ::= SET OF SignerInfo

Figure 20: Details of timeStampResp-timeStampToken-signedData-signerInfos (in progress)

signatureAlgorithmIdentifier (refer to [21]) - part 1

RSASSA-PKCS1-v1.5 signature algorithms

DSA signature algorithms

ECDSA signature algorithms

Figure 21: Standardized Signature Algorithms (according to [21]) continue…

sha256WithRSAEncriptionAlgorithm

ID: sha256WithRSAEncryption

(OID : pkcs-1.11)

sha384WithRSAEncriptionAlgorithm

ID: sha384WithRSAEncryption

(OID : pkcs-1.12)

sha512WithRSAEncriptionAlgorithm

ID: sha512WithRSAEncryption

(OID : pkcs-1.13)

sha224WithRSAEncriptionAlgorithm

ID: sha224WithRSAEncryption

(OID : pkcs-1.14)

rSASSA-PPSAlgorithm

ID: id-RSASSA-PPS

(OID : 1.2.840.113549.1.1.10)

PARAMS

hashAlgorithm: AlgorithmIdentifier

(see figure 15)

saltLLeangth ::= INTEGER

(default 20)

trailerField ::= INTEGER

(default 1)

dsa-with-sha224

ID: id-dsa-with-sha224

(OID : 2.16.840.1.101.3.4.3.1)

dsa-with-sha256

ID: id-dsa-with-sha256

(OID : sigAlgs.2)

ecdsa-with-SHA224

ID: ecdsa-with-sha224

(OID : 1.2.840.10045.4.3.1)

ecdsa-with-SHA256

ID: ecdsa-with-sha256

(OID : id-ecSigType.3.2)

ecdsa-with-SHA384

ID: ecdsa-with-sha384

(OID : : id-ecSigType.3.3)

sha1WithRSAEncriptionAlgorithm

ID: sha1WithRSAEncryption

(OID : 1.2.840.113549.1.1.5)

with SHA-1 (rfc3279)

with SHA-2 (1\*)

with SHA-3 (2\*)

sha3-224-WithRSAEncriptionAlgorithm

ID: id-rsassa-pkcs1-v1-5-with-sha3-224

(OID : 2.16.840.1.101.3.4.3.13)

sha3-256-WithRSAEncriptionAlgorithm

ID: id-rsassa-pkcs1-v1-5-with-sha3-256

(OID : sigAlgs.14)

sha3-384-WithRSAEncriptionAlgorithm

ID: id-rsassa-pkcs1-v1-5-with-sha3-384

(OID : sigAlgs.15)

sha3-384-WithRSAEncriptionAlgorithm

ID: id-rsassa-pkcs1-v1-5-with-sha3-512

(OID : sigAlgs.16)

with SHA-2 (2\*)

dsa-with-sha384

ID: id-dsa-with-sha384

(OID : sigAlgs.3)

dsa-with-sha512

ID: id-dsa-with-sha512

(OID : sigAlgs.4)

dsa-with-sha3-224

ID: id-dsa-with-sha3-224

(OID : 2.16.840.1.101.3.4.3.5)

dsa-with-sha3-256

ID: id-dsa-with-sha3-256

(OID : sigAlgs.6)

with SHA-3 (2\*)

dsa-with-sha3-384

ID: id-dsa-with-sha3-384

(OID : sigAlgs.7)

dsa-with-sha3-512

ID: id-dsa-with-sha3-512

(OID : sigAlgs.8)

dsa-with-sha1

ID: id-dsa-with-sha1

(OID : 1.2.840.10040.4.3)

with SHA-1 (rfc3279)

with SHA-2 (3\*) (rfc3279)

with SHA-3 (2\*)

ecdsa-with-SHA512

ID: ecdsa-with-sha512

(OID : : id-ecSigType.3.4)

with SHA-1 (3\*) (rfc3279)

ecdsa-with-SHA1

ID: ecdsa-with-SHA1

(OID : : id-ecSigType.1)

ecdsa-with-sha3-224

ID: id-ecdsa-with-sha3-224

(OID : 2.16.840.1.101.3.4.3.9)

ecdsa-with-sha3-256

ID: id-ecdsa-with-sha3-256

(OID : sigAlgs.10)

ecdsa-with-sha3-384

ID: id-ecdsa-with-sha3-384

(OID : sigAlgs.11)

ecdsa-with-sha3-512

ID: id-ecdsa-with-sha3-512

(OID : sigAlgs.12)

ecdsa-with-shake128 (4\*)

ID: id-ecdsa-with-shake128

(OID : 1.3.6.1.5.5.7.6.32)

ecdsa-with-shake256 (4\*)

ID: id-ecdsa-with-shake256

(OID : rfc8692SigAlgs.33)

signatureAlgorithmIdentifier (refer to [21]) - part 2

EdDSA Signatures (rfc8410)

sa-Ed25519

ID: id-Ed25519

(OID : 1.3.101.112)

ML-DSA Signatures (see [21])

Figure 21: Standardized Signature Algorithms (according to [2])

Figure 21: Standardized Signature Algorithms (according to [2])

SLH-DSA Signatures (see [21])

**Note:**

(1\*) pkcs-1 : iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 1

(2\*) sigAlgs : joint-iso-ccitt(2) country(16) us(840) organization(1) gov(101) csor(3) nistAlgorithm(4) 3

(3\*) id-ecSigType: iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4)

(4\*) rfc8692SigAlgs : iso(1) identified-organization(3) dod(6) internet(1) security(5) mechnisms(5) pkix(7) algorithms(6)

Figure 21: Standardized Signature Algorithms (according to [21])

timeStampResp-timeStampToken-signedData-signerInfos ::= SET OF SignerInfo

Figure 22: Details of timeStampResp-timeStampToken-signedData-signerInfos (in progress)

extensionId

crlExtensions ::= Extensions ::= SEQUENCE OF Extension { .. }

cRLNumber

number ::= INTEGER (0 .. MAX)

OID: 2.5.29.20 (id-ce-cRLNumber)

statusReferrals

statusReferrals ::= SEQUENCE SIZE (1 .. MAX) OF CHOICE { .. }

issuer [0] ::= GeneralName (optional)

cRLFeferral [0] ::= SEQUENCE { .. }

location [1] ::= GeneralName (optional)

deltaRefInfo [2] ::= DeltaRefInfo (optional) (see figure 23-a)

cRLScope ::= CRLScopeSyntax (see figure 23-b)

OID: 2.5.29.45 (id-ce-statusReferrals)

critical ::= BOOLEAN (default false)

extensionValue ::= OCTET STRING

**Note:**

id-ce OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 29}

Figure 23: Details of timeStampResp-timeStampToken-signedData-crls – certificateList – certificateListContent – crlExtensions

(in progress)

**2.3. Timestamp Request format**

**2.4. Time Stamp Token (TST) format**

**2.5. Merkle Tree Overview**

**2.6. Combining group of data objects using Merkle Tree**

**2.7. Standardized Evidence Record Syntax (RFC4998)**

**3. What is XMLERS?**

**4. Where are ERS and XMLERS in LTANS architecture?**

**5. ERS components**

**3. XMLERS components**

**4. Table of ERS APIs**

**5. Table of XMLERS APIs**

**Implementation**

**References**

[1]: <https://datatracker.ietf.org/wg/ltans/documents/>

[2]: rfc4810 – Long-term Archive Service Requirements - C. Wallace, et al. – March 2007

[3]: rfc4998 – Evidence Record Syntax (ERS) – Gondrom, et al. – August 2007

[4]: rfc5276 – Using the Server-Based Certificate Validation Protocol (SCVP) to Convey Long-Term Evidence Records – C. Wallace, et al. – August 2008.

[5]: rfc5698 – Data Structure for the Security Suitability of Cryptographic Algorithms (DSSC). – Kunz, et al. – November 2009

[6]: rfc6283 – Extensible Markup Language Evidence Record Syntax (XMLERS). – Blazic, et al. – July 2011

[7]: rfc3280 – X509 - Certificate and Certificate Revocation List (CRL) Profile. – Housley, et al. – April 2002

[8]: rfc3161 – Internet X.509 Public Key Infrastructure. Time-Stamp Protocol (TSP). – Adams, et al. – Aug. 2001

[9]: rfc3126 – Electronic Signature formats for Long-term electronic signatures. – Pinkas, et al. – Sep. 2001

[10]: rfc2459 – Internet X.509 Public Key Infrastructure – Certificate and CRL Profile. – Housley, et al. – Jan. 1999

[11]: rfc3280 - Internet X.509 Public Key Infrastructure – Certificate and CRL Profile. – Housley, et al. – April 2002

[12]: <https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.509>

[13]: Recommendation ITU-T X.509 – ITU 10/2019

[14]: rfc5280 - Internet X.509 Public Key Infrastructure – Certificate and CRL Profile. - D. Cooper – NIST

S. Santesson – Microsoft

S. Farrell – Trinity College Dublin

S. Boeyen – Entrust

R. Housley – Vigil Security

W. Polk – NIST

May 2008

[15]: rfc7299 – Object Identifier Register for the PKIX Working Group. – R. Housley – Vigil Security. – July 2014

[16]: rfc2560 – X.509 Internet Public Key Ỉnastructure: Online Certificate Status Protocol – Myers, et al. – June 1999

[17]: rfc3986 – Uniform Resource Identifier (URI): Generic Syntax – Berners-Lee, et al. – Jan. 2005

[18]: rfc1738 – Uniform Resource Locators (URL) Berners-Lee, Masinster & McCahill – Dec. 1994

[19]: rfc5652 – Cryptographic Message Syntax (CMS) – R. Housley – Sep. 2009

[20]: FIPS PUB 180-4 – Secure Hash Standard – U.S. DoC, NIST – Aug. 2015

[21]: <https://csrc.nist.gov/projects/computer-security-objects-register/algorithm-registration>

[22]: rfc2510 – Internet X.509 Public Key Infrastructure: Certificate Management Protocols – Adams & Farrell – Mar. 1999

[23]: rfc3628 – Policy Requirements for Time-Stamping Authorities (TSAs) – Pinkas, et al. – Nov. 2003

[24]: Policy requirements for time-stamping authorities - ETSI TS 102 023 v1.1.1 – April. 2002

[25]: Electronic Signatures and Infrastructures (ESI); Policy and Security Requirements for Trust Service Providers issuing Time-Stamps – ETSI EN 319 421 v1.2.1 – May 2023

[26]: Electronic Signatures and Infrastructures (ESI); General Policy Requirements for Trust Service Providers – ETSI EN 319 401 v2.3.1 – May 2021

[27]: rfc5816 – ESSCertIDv2 Update for RFC3161 – Santesson & Pope – Mar. 2010

[28]: <https://pki.digitalsign.pt/TSPS_V2.1_EN.pdf>

[29]: <https://wipoproof.wipo.int/wdts/docs/wipo_proof_timestamping_policy_and_practice_statement.pdf>

[30]: <https://exkasan.com/document/ThaiAI_AdvanCert_e-Timestamp%20Practice%20Statement_version_1.0_stamp.pdf>

[31]: rfc2634 – Enhanced Security Services for S/MIME – Hoffman – Jun. 1999

[32]: rfc5035 – Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility – Schaad – Aug. 2007

[33]: Electronic Signatures and Infrastructures; Time stamping profile – ETSI TS 101 861 v1.4.1 – July. 2011

[34]: rfc4210 – PKIX; Certificate Management Protocol (CMP) – Adams, et al. – Sep. 2005

[35]: rfc5912 – New ASN.1 Modules for PKIX – Hoffman & Schaad. – Jun. 2010

[36]: rfc9480 – Certificate Management Protocol (CMP) Updates. – Brockhaus, et al. – Nov. 2023

[37]: rfc4055 –

[38]: rfc3279 –

[39]: rfc3281 –

[40]: Recommendation ITU-T X.501 – ITU 10/2019