Independent Submission
Request for Comments: 5544
Category: Informational
ISSN: 2070-1721

A. Santoni
Actalis S.p.A.

February 2010

Syntax for Binding Documents with Time-Stamps

Abstract

This document describes an envelope that can be used to bind a file (not necessarily protected by means of cryptographic techniques) with one or more time-stamp tokens obtained for that file, where "time-stamp token" has the meaning defined in RFC 3161 or its successors. Additional types of temporal evidence are also allowed.

The proposed envelope is based on the Cryptographic Message Syntax as defined in RFC 5652.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5544.

IESG Note

This RFC is not a candidate for any level of Internet Standard. The standards track specification RFC 4998, Evidence Record Syntax (ERS), specifies an alternative mechanism. Readers are encouraged to also review RFC 4998 when evaluating the suitability of this mechanism.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http:trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

Time-stamping has become the standard technique for proving the existence of a document before a certain point in time. Several legislations around the world embrace the concept and provide for time-stamping services, mainly for the purpose of extending the validity of signed documents. However, while time-stamping enhances digital signatures, its value does not depend on them. It can clearly be useful to time-stamp a document even if it is not signed. And it can also be useful, or even mandatory in some cases, to time-stamp a signed document in its entirety, regardless of how many signatures it contains.

When a time-stamp is related to a digital signature, there already exists a way to keep the two pieces together: RFC 3161 [TSP] describes how one or more TimeStampTokens can be included in a SignerInfo structure as unsigned attributes. On the other hand, there is no standard way to keep together a time-stamped document, whether signed or not, and the related time-stamps.

In such cases, two approaches are typically being adopted:

- time-stamps are kept as separate files (keeping track of what time-stamps belong to what documents is up to the user);
- an ad hoc solution is adopted for specific applications, e.g., a ZIP archive or a proprietary "envelope" of some kind.

Both solutions impede interoperability, which is the objective of this memo.

This document describes a simple syntax for binding one document (actually, any kind of file) to the corresponding temporal evidence; the latter is typically represented by one or more RFC 3161 TimeStampTokens. Additional types of temporal evidence, e.g., an RFC 4998 EvidenceRecord [ERS], are also supported via an "open" syntax. However, for the sake of interoperability, the emphasis in this document is on TimeStampTokens.

The proposed syntax is broadly based on the Cryptographic Message Syntax (CMS) defined in RFC 5652 [CMS].

1.1. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [KWORDS].

The terms "document" and "file" are used interchangeably. The terms "TimeStampToken" and "time-stamp token" are used interchangeably, both referring to the data structure defined in RFC 3161.

2. Syntax for TimeStampedData

The proposed data structure is called TimeStampedData, and it is based on the ContentInfo envelope defined in [CMS]:

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

ContentType ::= OBJECT IDENTIFIER
```

While CMS defines six content types (data, signed-data, enveloped-data, digested-data, encrypted-data, and authenticated-data), this memo defines an additional content type, timestamped-data, identified by the following Object Identifier (OID):

```
   id-ct-timestampedData OBJECT IDENTIFIER ::= {
            iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
            pkcs9(9) id-smime(16) id-ct(1) 31 }
```

This particular OID signals that the content field of the ContentInfo has the following syntax:

```
   TimeStampedData ::= SEQUENCE {
      version              INTEGER { v1(1) },
      dataUri              IA5String OPTIONAL,
      metaData             MetaData OPTIONAL,
      content              OCTET STRING OPTIONAL,
      temporalEvidence     Evidence
   }

   MetaData ::= SEQUENCE {
      hashProtected        BOOLEAN,
      fileName             UTF8String OPTIONAL,
      mediaType            IA5String OPTIONAL,
      otherMetaData        Attributes OPTIONAL
   }

   Attributes ::=
      SET SIZE(1..MAX) OF Attribute -- according to RFC 5652

   Evidence ::= CHOICE {
      tstEvidence    [0] TimeStampTokenEvidence,   -- see RFC 3161
      ersEvidence    [1] EvidenceRecord,           -- see RFC 4998
      otherEvidence  [2] OtherEvidence
   }

   OtherEvidence ::= SEQUENCE {
      oeType               OBJECT IDENTIFIER,
      oeValue              ANY DEFINED BY oeType }

   TimeStampTokenEvidence ::=
      SEQUENCE SIZE(1..MAX) OF TimeStampAndCRL

   TimeStampAndCRL ::= SEQUENCE {
      timeStamp   TimeStampToken,          -- according to RFC 3161
      crl         CertificateList OPTIONAL -- according to RFC 5280
   }
```

The version field contains the version number of the TimeStampedData syntax. It SHALL be 1 for this version of the document.

The dataUri field contains a URI reference conforming to [URI]. When the content field is absent, dataUri MUST be present and contain a URI allowing retrieval of the document that was time-stamped (unless the document is later moved). When the content field is present, this field MAY also be present.

The metaData field contains metadata related to the document that was time-stamped, if applicable. In particular:

- The hashProtected field indicates whether the metadata have been included in the computation of the digest within the first TimeStampToken (see further on). This makes it possible to detect a subsequent alteration of the metadata.

- The fileName field contains the original filename of the document that was time-stamped.

- The mediaType field contains a media type/subtype and possible parameters for the time-stamped document, according to [MIME]. This information may help decide how to "open" or deal with the time-stamped document.

- The otherMetaData field contains further attributes of the time-stamped document (e.g., a description, claimed author, etc.), where each attribute is specified by an object identifier and a corresponding set of values, as described in [CMS]. When this field is present, it MUST contain at least one Attribute.

Within the metaData field (if present), at least one of the fileName, mediaType, and otherMetaData sub-fields MUST be present.

The Attribute values within the otherMetaData field MUST be DER encoded, even if the rest of the structure is BER encoded.

The content field, when present, carries the entire contents, in its original format and encoding, of the document that was time-stamped. This can actually be any kind of data, e.g., a text document, an executable, a movie, a message, etc. The omission of the content field makes it possible to bind the temporal evidence to external data. In such a case, the temporal evidence is computed as though the content field were present.

The temporalEvidence field carries the evidence that the time-stamped document did exist before a certain point in time. Several types of evidence are allowed, but compliant applications are only required to support the RFC 3161 type -- namely, the tstEvidence choice.

The TimeStampTokenEvidence sequence MUST contain at least one element of type TimeStampAndCRL.

The elements of the TimeStampTokenEvidence sequence MUST conform to the following rule:

- if the metaData field is absent or the value of its hashProtected field is FALSE, then the TimeStampToken within the first element SHALL be computed over the value octets of the content field (if this field is absent, use the octets retrieved via the dataUri field);

- otherwise (the metaData field is present and the value of its hashProtected field is TRUE), the TimeStampToken within the first element SHALL be computed over the concatenation of the following fields:

  - the DER encoding of the metaData field;

  - the value octets of the content field (if this field is absent, use the octets retrieved via the dataUri field);

- the TimeStampToken within the second element SHALL be computed over the first element;

- the TimeStampToken within each subsequent element SHALL be computed over its preceding element in the sequence.

Within the TimeStampAndCRL construct, the optional crl field carries a suitable CRL (Certificate Revocation List) demonstrating that the certificate of the TSA (Time-Stamping Authority) that issued the TimeStampToken was not revoked at the time when the subsequent element in the TimeStampTokenEvidence sequence was added. See the Security Considerations section for further discussion on this topic.

3. Compliance Requirements

Compliant applications MUST support at least the RFC 3161-based type of evidence (i.e., the tstEvidence CHOICE).

4. Recommended Processing

This section is focused on the RFC 3161-based type of evidence. Processing of the structure for other types of evidence would be done in a similar manner.

4.1. Generating a New TimeStampedData Structure

In this case, applications are supposed to behave as follows:

- populate the version field with the integer value v1(1);
- if a self-contained envelope is to be generated, always populate the content field with the content of the file in its original format and encoding; depending on the application, the dataUri field may also be added;
- otherwise (a detached envelope is to be generated), always populate the dataUri field with the URI of the time-stamped document (e.g., http://foo.example.com/Contract12345.pdf); using an absolute URI or a relative reference depends on the application;
- if the metaData field is being added, decide on the value of its hashProtected field; set its value to TRUE if the application needs the remaining fields of the metaData construct to be hash-protected as described in Section 2; otherwise, set it to FALSE;
- if the metaData field is being added, optionally populate the fileName field (e.g., "Contract12345.pdf"), the mediaType field with a suitable media type/subtype and possible parameters according to [MIME], and the otherMetaData field, depending on the application;
- select a suitable one-way hash function and compute a hash value using that function over the content, or the concatenation of the metadata and the content, as described in Section 2; this hash value will then be used for requesting the first TimeStampToken;
- obtain the first temporal evidence from a TSA and add it to the temporalEvidence field;
- insert the TimeStampedData into a ContentInfo structure, with the id-ct-timestampedData OID in the contentType field;
- BER-encode the ContentInfo structure (except for the fields that are required to be DER encoded) and save it with a reasonable file name (e.g., derived from the name of the time-stamped file).

4.2. Verifying an Existing TimeStampedData Structure

In this case, applications are supposed to behave as follows:

- check that the contentType field of the ContentInfo structure has the expected value (id-ct-timestampedData) in its contentType field; then, extract the inner TimeStampedData structure and continue processing;

- check the version field (it should be v1);

- check that the temporalEvidence field is not empty;

- check whether the content is present; if it is not, use the dataUri field to retrieve the file;

- open the first element of the TimeStampTokenEvidence sequence, open the time-stamp token within it and use the hash function that was used to obtain it to re-compute the hash of the fields indicated in Section 2; if the re-computed hash value matches the one within the time-stamp token, continue processing; otherwise, the TimeStampedData structure has been modified;

- validate the temporalEvidence by checking that:

  - each TimeStampToken in the chain does contain the correct digest value (according to the rule described in Section 2) and it was signed by a trusted TSA,

  - the corresponding TSA signing certificate was not revoked at the time when the subsequent TimeStampToken was issued, based on the associated CRL;

- depending on the application, use the temporal evidence for whatever purpose the application was designed for;

- depending on the application, show the dataUri, the fileName, the mediaType, the otherMetaData, and the temporal evidence to the user;

- depending on the application, save the content to a separate file;

- depending on the application, store at a different place the content that has been retrieved using the dataUri field, then update the dataUri field accordingly;

- depending on the application, show the time-stamped file to the user, possibly by activating a suitable "viewer".

4.3. Extending the Validity of an Existing TimeStampedData Structure

In this case, applications are supposed to behave as follows:

- validate the TimeStampedData structure as described above;
- select the time-stamp token from the last TimeStampAndCRL element in the chain and obtain the latest available CRL for the corresponding TSA certificate (if this CRL is not fresh enough, wait until the next one is available), then store it in the TimeStampAndCRL element;
- instantiate a new TimeStampAndCRL element and obtain a new time-stamp token computed over the previous one, according to the rule described in Section 2; insert the new time-stamp token into the new TimeStampAndCRL element, then append the latter to the end of the chain.

See the Security Considerations section for further discussion on extending the validity of an existing TimeStampedData structure.

5. Security Considerations

When the metaData field is present and the hashProtected sub-field is set to TRUE, the metadata are also included in the computation of the digest within the first time-stamp token, so that any subsequent alteration of the metadata will be easily detected. However, the integrity of hash-protected metadata does not imply that the metadata were correct at the time when the TimeStampedData object was created. That can only be inferred by other means (e.g., from context). For instance, when TimeStampedData objects are created by an archival service provider, it may be reasonable to assume that the metadata are correct at creation time. Instead, when a TimeStampedData object is received from an unknown party, the recipient cannot safely assume that the metadata are correct, lacking further information.

In general, a time-stamp token should not be considered valid after the certificate of the issuing TSA is expired (also, this consideration depends on the legislation and the policy under which the TSA operates). However, a time-stamp token can itself be time-stamped to extend the validity of the TSA's signature. By repeatedly applying this technique, a whole chain of time-stamp tokens can be grown to extend the validity of the first one ad libitum. Thus, this approach can be adopted to extend the validity of a TimeStampedData structure beyond the expiry date of the first TimeStampToken within it, by adding further elements to the TimeStampTokenEvidence sequence

according to the rule described in Section 2. Of course, each additional TimeStampToken must be added in a timely manner (before the previous one is expired or has been revoked).

The validity extension technique described above requires that the TSA signing certificates can still be verified long after they have expired, typically by checking a CRL. The CRL must be captured at the suitable time, because expired certificates are typically removed from the CRL regardless of their being revoked. The TimeStampAndCRL construct allows adding a CRL next to the related TimeStampToken, so that the TSA certificate will still be verifiable at any later time. The CRL must be captured at the time when another element is about to be added to the TimeStampTokenEvidence sequence, or even later -- to allow for a last-minute revocation request to be processed by the CA (see the discussion about "grace periods" in [CADES]).

6. Normative References

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[ERS] Gondrom, T., Brandner, R., and U. Pordesch, "Evidence Record Syntax (ERS)", RFC 4998, August 2007.

[KWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[PKIX1] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[TSP] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

7. Informative References

[CADES] Pinkas, D., Pope, N., and J. Ross, "CMS Advanced Electronic Signatures (CAdES)", RFC 5126, March 2008.

Appendix A. ASN.1 Module

The ASN.1 module contained in this appendix defines the structures that are needed to implement this specification. It is expected to be used in conjunction with the ASN.1 modules in [CMS], [TSP], [PKIX1], and [ERS].

```
TimeStampedDataModule
   { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) smime(16) modules(0) 35 }

   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   IMPORTS

      -- Imports from RFC 5652 [CMS]
      Attribute
         FROM CryptographicMessageSyntax2004
            { iso(1) member-body(2) us(840) rsadsi(113549)
              pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) }

      -- Imports from RFC 3161 [TSP]
      TimeStampToken
         FROM PKIXTSP
            { iso(1) identified-organization(3) dod(6) internet(1)
              security(5) mechanisms(5) pkix(7) id-mod(0)
              id-mod-tsp(13)}

      -- Imports from RFC 5280 [PKIX1]
      CertificateList
         FROM PKIX1Explicit88
            { iso(1) identified-organization(3) dod(6) internet(1)
              security(5) mechanisms(5) pkix(7) id-mod(0)
              id-pkix1-explicit-88(18)}

      -- Imports from RFC 4998 [ERS]
      EvidenceRecord
         FROM ERS
            { iso(1) identified-organization(3) dod(6) internet(1)
              security(5) mechanisms(5) ltans(11) id-mod(0)
              id-mod-ers88(2) id-mod-ers88-v1(1) };

   -- TimeStampedData Content Type and Object Identifier

   id-ct-timestampedData OBJECT IDENTIFIER ::= {
      iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
      id-smime(16) id-ct(1) 31 }
```

```
TimeStampedData ::= SEQUENCE {
   version              INTEGER { v1(1) },
   dataUri              IA5String OPTIONAL,
   metaData             MetaData OPTIONAL,
   content              OCTET STRING OPTIONAL,
   temporalEvidence     Evidence
}

MetaData ::= SEQUENCE {
   hashProtected        BOOLEAN,
   fileName             UTF8String OPTIONAL,
   mediaType            IA5String OPTIONAL,
   otherMetaData        Attributes OPTIONAL
}

Attributes ::=
   SET SIZE(1..MAX) OF Attribute -- according to RFC 5652

Evidence ::= CHOICE {
   tstEvidence    [0] TimeStampTokenEvidence,   -- see RFC 3161
   ersEvidence    [1] EvidenceRecord,           -- see RFC 4998
   otherEvidence  [2] OtherEvidence
}

OtherEvidence ::= SEQUENCE {
   oeType            OBJECT IDENTIFIER,
   oeValue           ANY DEFINED BY oeType }

TimeStampTokenEvidence ::=
   SEQUENCE SIZE(1..MAX) OF TimeStampAndCRL

TimeStampAndCRL ::= SEQUENCE {
   timeStamp   TimeStampToken,          -- according to RFC 3161
   crl         CertificateList OPTIONAL -- according to RFC 5280
}

END
```

Appendix B. Acknowledgments

Thanks to Stephen Kent for encouraging the author in the early stages of this work.

Thanks to Russ Housley for reviewing this memo, suggesting useful amendments and assigning a value to the OIDs herein defined.

Thanks are also due to other people who reviewed this memo and helped improving it, but prefer not to be mentioned.

Author's Address

Adriano Santoni
Actalis S.p.A.
Via Taramelli 26
I-20124 Milano
Italy

EMail: adriano.santoni@actalis.it