Towards an XML Format for Time-Stamps

Karel Wouters*
karel.wouters@esat.kuleuven.ac.be

Bart Preneel
bart.preneel@esat.kuleuven.ac.be

COSIC research group, Department of Electrical Engineering - ESAT
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Heverlee-Leuven, Belgium
http://www.esat.kuleuven.ac.be/cosic

Ana Isabel González-Tablas
aigonzal@inf.uc3m.es

Arturo Ribagorda

Computer Science Department-GSTI
Carlos III University of Madrid
Avda. de la Universidad 30, 28911 Leganés, Spain

ABSTRACT

XML has become a well-established format for information exchange. Several formats have been defined to secure XML data, such as XML Digital Signatures, XML Encryption and XKMS. In recent work by ETSI on XML digital signatures conforming to European legislation, time-stamps play a key role for qualified digital signatures. Some ASN.1-based formats for time-stamp protocols have been defined within IETF and ISO/IEC. In this paper, we investigate how the wide range of time-stamping protocols in the literature can be embedded into a single XML format; our work is based on existing standardisation efforts. We present our ideas in the form of a concrete XML structure, which can be used as the starting point to develop a mature XML-based time-stamping protocol.

1. INTRODUCTION

Digital time-stamping is a set of techniques that enables us to determine if a certain digital document has been created or signed before a given time. In most practical applications, the time-stamping service is performed by a trusted third party – a Time-Stamping Authority (TSA) – that creates time-stamps. These time-stamps are the digital assertions that a given document was presented to the TSA at a given time. There also exist some distributed time-stamp techniques in which a group of users/TSAs collaborates to compute a time-stamp.

Apart from obvious applications, time-stamping also plays an important role in the classical Public Key Infrastructure (PKI). In this context, it can be used to extend the lifetime of digital signatures: a time-stamp on a digital signature can prove that the signature was generated before the signature key-pair expired or was revoked. Even if the underlying mathematics or the hash algorithm is broken, the signature can still remain valid if the time-stamping algorithm is sufficiently strong. This is clearly taken into account in the relatively new European standards on advanced electronic signatures [17]. Note that this may not be necessary for low-value electronic signatures.

Throughout the past few years, a significant amount of research has been done on the time-stamping problem. In Section 2, we classify existing time-stamping schemes. Section 3 gives an overview of the standards that we studied to construct our time-stamping format. In Section 4, we specify our format and we apply it to some existing schemes. We conclude with an overview of our work and some open issues.

2. CLASSIFICATION OF TIME-STAMPING SCHEMES

We classify time-stamping schemes into three sections: simple schemes, linking schemes and distributed schemes.

2.1 Simple Schemes

Simple schemes generate time-stamp tokens that are independent; they do not include information of other time-stamps. A classical example of this scheme is the digital signature of a TSA on a pair `(time,document)`, which is proposed by Adams *et al.* [6] and in ISO/IEC FDIS 18014-2 [2]. The main limitation of these schemes is that they assume a rather high level of trust in the issuing party, the TSA. Furthermore, nobody can detect possible fraudulent behaviour of the TSA. All of these time-stamps offer so-called *absolute*

*The author was partially supported by the GOA project MEFISTO 2000/6 of the Flemish Government

ACM Workshop on XML Security 2002 Washington D.C. USA

temporal authentication; they include the time at which the time-stamp was made and so they can be situated into a small accuracy interval, if the TSA does not cheat.

2.2 Linking Schemes

Linking schemes try to lower the required trust in the TSA by linking time-stamps. Data from other time-stamps is included into the computation of the issued time-stamp, such that they depend on each other. Linking happens in three phases:

Aggregation: in the first step, all documents received by the TSA within a small time interval – the aggregation round – are being considered simultaneous. The output of the aggregation round is a binary string that securely depends on all the documents submitted in that round. Users receive information on how to compute the aggregation output, using their submitted document. The purpose of aggregation is to lower the load on the TSA, if the linking operation is expensive.

Linking: the output of the aggregation round is taken, and linked to previous aggregation round values, where the output of the linking operation cannot be computed without previous aggregation round values. This establishes a one-way order between aggregation round values, such that so-called *relative temporal authentication* is obtained: time-stamps of different aggregation rounds can be compared. This implies also that a time value is not necessary in linking schemes.

Publication: from time to time (e.g., each week), the TSA publishes the most recent time-stamp in a widely witnessed medium, such as a newspaper. By doing this, the TSA commits itself to all of the previously issued time-stamps. The published values are used for verifying time-stamps and they enable other parties to check if the TSA is behaving properly.

Examples of linking can be found in Bayer *et al.* [8], Benaloh *et al.* [9] and Buldas *et al.* [13]. In these cases, the linking can be visualised by a graph and optimised in time-stamp size. In Benaloh *et al.* [10] and Merkle [24], some aggregation schemes are proposed. These can be based on hash functions in graph-like structures or on number-theoretic problems.

2.3 Distributed Schemes

Another way of lowering the required level of trust in the TSA is to distribute the trust. In that approach, multiple users/TSAs cooperate to generate a time-stamp, possibly using a secure distribution of secret data necessary to generate a time-stamp. In this way, forgery of a time-stamp requires the collusion of a predetermined (high) number of parties, which is considered to be very unlikely. Example of such protocols can be found in Benaloh *et al.* [9, 10] and Ansper *et al.* [7]. They can be extensions of the schemes described above, and as such provide relative temporal authentication or absolute temporal authentication.

2.4 Other schemes

Not all existing time-stamping schemes can be classified into these three sections. For example, the scheme of Haber *et al.* [18] includes the time value in a digitally signed document that contains linking information. This linking information has to be checked "as far as necessary to convince the most suspicious verifier." Obviously, this also includes no checking at all, although the scheme suggest to follow at least one link.

It has to be noticed that not all of these time-stamping schemes are equally secure, see Just [22] for some remarks on that. Some of them have their security goals strictly set, others have not. For a security classification of some recent schemes, we refer to Une [27].

3. STANDARDS FOR XML AND TIME-STAMPING

In this section, we briefly describe some standards upon which we would like to build.

3.1 XMLDSig and XAdES

The mission of the joint W3C and IETF XML Signature working group [14] was to develop an XML compliant syntax used for representing a digital signature of Web resources and portions of protocol messages (anything referencable by a URI), and procedures for computing and verifying such signatures. The resulting document, XML Signature Syntax and Processing (XMLDSig, [14]), specifies the XML digital signature processing rules and syntax. XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere.

ETSI TS 101 903 [16, 17] (better known as XAdES , XML Advanced Electronic Signatures) was built on top of XMLDSig. It defines XML formats for advanced electronic signatures that remain valid over longer periods, are compliant to the Europeans Directive on a community framework for Electronic Signatures [25], and incorporate additional useful information in common use cases.

Obviously, XMLDSig and XAdES play an important role in the context of this paper. Not only do they provide us with useful tools for the definition of our XML time-stamp format, they also provide an interesting application to use time-stamps for the extension of digital signatures. (XAdES explicitely assumes the existence of a TSA).

3.2 IETF Standards

The IETF PKIX Working Group [23] was established in the Fall of 1995 with the intent of developing Internet standards needed to support an X.509-based PKI. The PKIX Time-Stamp Protocol, (PKIX-TSP, RFC 3161 [6]), describes the format of a request sent to a TSA and of the response that is returned. It also establishes several security-relevant requirements for TSA operation, with regard to processing requests to generate responses. This standard describes a simple scheme: time-stamps are digital signatures by the TSA on the submission time and the value of a digital document's message digest.

Another IETF Working Group, S/MIME Mail Security [20], developed the Cryptographic Message Syntax (SMIME-CMS, [21]). This syntax is used to digitally sign, digest, authenticate, or encrypt arbitrary messages. Although we did not base our format directly on this document, it is referenced by PKIX-TSP for the signature functionality. PKIX-TSP and SMIME-CMS are based on ASN.1 [4] and imply DER/BER-encoding [5] of the defined objects.

3.3 ISO Standards

In 1999, the ISO/IEC JTC1/SC27 on security techniques started a project on time-stamping services (ISO/IEC

18014). In this project, three work items have been defined until now:

- **A time-stamping services framework** [1]. In this item, a general framework for time stamping services was built. Communications between the TSA and the client are discussed. The time stamping formats themselves are defined in the following documents.

- **Mechanisms producing independent tokens** [2]. The group of experts decided to integrate the existing IETF PKIX-TSP in this work item. Apart from that, they defined two other time-stamp formats: one where Message Authentication Codes (MACs) replace the digital signatures, and one where the submitted information is archived by the TSA, together with the time of submission (in this case the TSA has to be trusted completely). All of these tokens have the property that they can be verified without access to other tokens. These time-stamp tokens are situated in the simple schemes described above.

- **Mechanisms producing linked tokens** [3]. This document describes time-stamp tokens of a linking scheme. It is still in development and at the time of writing it provides a generic framework that should support several types of aggregation, linking and publication.

As far as we could understand, none of these work items aim to define a distributed time-stamping scheme.

3.4 XER

The ITU-T X.693 Recommendation/International Standard [28] defines two sets of encoding rules that may be applied to values of ASN.1 types and that use XML. These encoding rules are called the XML Encoding Rules (XER) for ASN.1, and both produce an XML document compliant to W3C XML 1.0. The first set is called the Basic XML Encoding Rules. The second set is called the Canonical XML Encoding Rules because there is only one way of encoding an ASN.1 value using these encoding rules. At first sight, it would appear to be interesting to translate the IETF or ISO standards directly into XML using these encoding rules, but that proved to be impossible, because both standards explicitly assume DER-encoding of the time-stamp information. Furthermore, it would be a missed opportunity to ignore the existing rich framework of XMLDSig and XAdES.

Note that, at the time of writing, a complete ASN.1/XML solution for CMS is being defined in the American Bankers Association X9F3 working group [31]. X9.96 XML Cryptographic Message Syntax (XCMS, [30]) will use a single ASN.1 schema for CMS to provide both compact binary encodings using BER/DER, and an XML markup solution using XER. Using this solution, combined with a XER-translation of the IETF or ISO/IEC standards will result in a XML time-stamp format. This may be something to explore in the future. The same group is also working on a Trusted Time-Stamp format (X9.95, [29]), which appears to be based on the ISO/IEC work.

4. PROTOCOL AND SYNTAX

4.1 Overview

The main structures of a time-stamping scheme are:

- A TimeStampRequest, the message a user sends to the TSA(s), requesting a time-stamp of type X of some data. The core component of this request is a digest value that should be time-stamped. Optionally, the user can specify other elements, such as the preferred TSA policy, a nonce to thwart replay attacks and in the PKI case an indication if the user wants to receive additional certificate information in the response. The replay attack occurs when a middleman is replaying legitimate TSA responses. It may also be desirable to send multiple digest values (using a different hash function) of the same document to protect against –unknown– attacks on a single hash function, see for example the TIMESEC project [26]. In the request there should be no information that could reveal the content of the document to be time-stamped. There is one exception to this rule: the user might want the TSA to act as a notary authority, so the document is sent (over an authenticated encrypted channel) to be stored by the TSA.

- A TimeStampResponse, the response message generated by the TSA, should contain a field indicating the response status and, if the time-stamp could be generated, the time-stamp itself. Note that this may be a partial time-stamp in the case of a distributed scheme.

- A VerifyRequest and a VerifyResponse, in the case where the explicit cooperation of the TSA is required to verify a time-stamp.

In the next sections, we elaborate the main structures. Note that if XAdES or XMLDSig elements were fit to do the job, we adopted them in our scheme. From here on, elements borrowed from XMLDSig will be prefixed with "`ds:`"; elements from XAdES will be prefixed by "`xades:`". Our scheme has been written in W3C's Schema language [19]; schema elements shall be prefixed with "`xs:`", and our newly defined elements will be prefixed by "`tsp:`". For presenting our ideas, we will work with simplified structures; the full schema can be found in the appendix A.

4.2 Common syntax

The `tsp:TimeStampRequest` element is sent to the TSA when a client wants to have a document time-stamped. Normally, the request will contain the digest value of the document to be time-stamped. It has the following structure:

```
<tsp:TimeStampRequest Type? CertReq?>
    <tsp:MessageImprints Id?>
    (<xades:SignaturePolicyIdentifier>)?
    (<tsp:Nonce>)?
    (<ds:Object>)?
</tsp:TimeStampRequest>
```

- The `tsp:TimeStampRequest` element has an attribute `Type` that indicates the requested time-stamp type, and a `CertReq` attribute to indicate if detailed certificate information is required if the TSA uses digital signatures.

- `tsp:MessageImprints` contains the digest values to be time-stamped. This element is described further on.

- `xades:SignaturePolicyIdentifier` identifies the signature policy that the user wants the TSA to apply. A default policy will be applied if none is specified in the request.
- `tsp:Nonce` contains a random value to prevent replay attacks. It should be copied into the response of the TSA.
- `ds:Object` will contain the documents to be time-stamped in the case of a notary authority.

When a user submits a `tsp:TimeStampRequest`, the TSA responds with a `tsp:TimeStampResponse`. This element is structured as follows:

```
<tsp:TimeStampResponse>
    <tsp:Status>
       <tsp:MajorStatus Code>
       (<tsp:FailInfo Code>)?
    </tsp:Status>
    (<tsp:TimeStampToken Id?>)?
</tsp:TimeStampResponse>
```

The `Status` element contains information about how the request was handled. Its children contain machine-readable information in the attribute `Code` and human-readable information in their (textnode) children. `MajorStatus` indicates general information, such as "Time-stamp granted", while `FailInfo` can indicate the reason why a request failed.
The `tsp:TimeStampToken` contains the time-stamp itself. Its structure is as follows:

```
<tsp:TimeStampToken Id?>
     (<tsp:References>
       [(<ds:Reference>)+ OR <tsp:XadesTSTLink Idref>]
     </tsp:References>)?
    (<tsp:MessageImprints Id?>
       (<tsp:DigestAlgValue Id?>)+
     </tsp:MessageImprints>)?
    <tsp:TSTInfo Id?>
       (<xades:SignaturePolicyIdentifier>)?
       (<tsp:SerialNumber>)?
       (<tsp:GenTime MilliSeconds? MicroSeconds?>)?
       (<tsp:Accuracy>
          (<Seconds>)? (<MilliSeconds>)? (<MicroSeconds>)?
        </tsp:Accuracy>)?
       (<tsp:Ordering />)?
       (<tsp:Nonce>)?
       (<tsp:TSA URI?>)?
       (<tsp:Id Name>)?
    </tsp:TSTInfo>
    (<ds:Signature>)?
    (<tsp:BindingInfo Algorithm Id?>)?
</tsp:TimeStampToken>
```

- The `tsp:References` element contains references to time-stamped content. It is a set of digest values that are used as input of a hash function that will produce the digest values in the `tsp:MessageImprints` element. Or, it can contain a `tsp:XadesTSTLink`, a link to XAdES time-stamp information (a `xades:HashDataInfo` element) that specifies the octet stream to be time-stamped, again used as input for the `tsp:MessageImprints` element. The `tsp:References` element can contain several references to the same content, but digested with different hash functions. Note that this element is added by the user after receiving the time-stamp response.
- `tsp:MessageImprints` contains the digest values and the algorithms to produce those values from the concatenated digest values or the octet stream above. This construction allows for several 'imprints' with different hash functions.
- The `tsp:TSTInfo` element contains TSA-specific time-stamp information; we expect it to be present in each type of time-stamp. Its format is almost the same as the `TSTInfo` element in PKIX-TSP. Note that `<tsp:GenTime>` is an extension of `xs:dateTime`.
- `ds:Signature`. Depending on the time-stamping scheme, this element is included to sign `tsp:MessageImprints`, `tsp:TSTInfo` and/or `tsp:BindingInfo`.
- `tsp:BindingInfo`. This element contains the binding information for linked time-stamps.

With these elements every time-stamp token of the simple and linked schemes can be constructed.

4.3 Simple schemes

For signed time-stamps or time-stamps computed with a MAC algorithm, the TSAs response will contain a `tsp:MessageImprints`, a `tsp:TSTInfo` (including a time), and a `ds:Signature` element. The references in the signature will point to the `tsp:MessageImprints` and the `tsp:TSTInfo` element. It is preferable that Exclusive Canonicalization (exc-C14N, [11]) is used, as well for the references in the signature, as for the C14N of the `ds:SignedInfo` in the signature. This is because, more than likely, the time-stamp is going to be embedded into another XML document, which will corrupt the signature if the enveloping document introduces additional namespaces, and if ordinary C14N is applied.

For archiving schemes, the `tsp:MessageImprints` and `tsp:TSTInfo` elements are required. Of course, the document to be archived should be protected. This can be obtained by using a secure connection such that the entire communication is confidential and authenticated (client and server), or by using XML Encryption [15] to encrypt the document only, combined with an authentication mechanism.

4.4 Linking schemes

A linking scheme takes a digest value as input. If a `tsp:TSTInfo` element is present in the response, it can also be taken into account in the digest value that is presented to the linking scheme. The response of the TSA was described above, and the `tsp:BindingInfo` has the following structure:

```
<tsp:BindingInfo Algorithm Id?>
  <tsp:DigestAlgValue Idrefs? Id?>
  (<tsp:AggregationInfo Algorithm? Id?>)?
  <tsp:LinkingInfo Algorithm?>
    (<tsp:Head Id?>)?
    (<tsp:Tail Id?>)?
    (<ds:Object>)?
  </tsp:LinkingInfo>
  (<tsp:PublishedInfo Id? Location?>)?
</tsp:BindingInfo>
```

- `tsp:DigestAlgValue` contains the digest value that is passed on to the linking scheme. This is the result

of selecting the nodes referred to by the attribute `Idrefs`, and use their concatenation as the input of the specified hash function. The attribute can point to digest values within `tsp:MessageImprints` and to the `tsp:TSTInfo` element. In the first case, the octet stream representing the referenced digest value is taken, in the last case, the output of exc-C14N of the referenced element is taken.

- `tsp:AggregationInfo`: if present, this element specifies the aggregation algorithm and the necessary data to compute the output of the aggregation round with the `tsp:DigestAlgValue` element.
- `tsp:LinkingInfo` contains the algorithm and data to compute the value of the linking round, given the output of the aggregation round. If no aggregation is specified, the value from the `tsp:DigestAlgValue` element is taken. `tsp:Head` contains linking information from time-stamps, issued before this one. `tsp:Tail` contains information from time-stamps after this one, which is transmitted by the TSA at the end of the linking round. `ds:Object` contains information that we thought was 'unnatural' to include directly into `tsp:Head` or `tsp:Tail`. It can be referenced from within these elements.
- `tsp:PublishedInfo`: contains round values for linking rounds, plus the location where they can be retrieved or verified.

The elements `tsp:AggregationInfo`, `tsp:Head`, `tsp:Tail` and `tsp:PublishedInfo` all have the same structure, `tsp:ChainingType`, which is defined as follows:

```
<xs:complexType name="ChainType">
  <xs:sequence>
    <xs:element name="Node"
                type="tsp:NodeType"
                maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:ID" use="optional"/>
</xs:complexType>

<xs:complexType name="NodeType">
  <xs:choice>
    <xs:sequence>
      <xs:element ref="ds:DigestMethod" minOccurs="0"/>
      <xs:element ref="ds:DigestValue" minOccurs="0"/>
    </xs:sequence>
    <xs:element name="BinaryContent"
                type="xs:base64Binary"/>
  </xs:choice>
  <xs:attribute name="Id"
              type="xs:ID" use="optional"/>
  <xs:attribute name="Reference"
                type="xs:IDREF" use="optional"/>
  <xs:attribute name="Alignment"
                type="xs:string" use="optional"/>
</xs:complexType>
```

- An element of type `tsp:ChainType` consists of a series of `tsp:Node`s, following the ideas of Buldas *et al.* [12].
- We chose `tsp:Node` to be a multi-functional basis element; it occurs in different contexts, depending on the linking, aggregation or publishing algorithm. As many linking schemes are based on hash functions, the `tsp:Node` should be able to hold a digest value. The `tsp:BinaryContent` element is provided to specify other binary information, such as the result of a modular exponentiation in the aggregation scheme of Benaloh *et al.* [10].
- The attribute `Reference` of the `tsp:Node` is included to avoid over-definition of the `tsp:NodeType`. This can be used to refer to structured data, embedded in the `ds:Object` element in the linking information. `Alignment` is included to provide location information in Merkle trees, used for aggregation.

4.5 Application to some time-stamping schemes

This section illustrates our XML time-stamping format by applying it to some recent/used time-stamping schemes.
Scenario: let a user A send a `TimeStampRequest` of some type containing a digested value X_n (X_n) to the TSA. Suppose that the TSA is able to issue the time-stamp and that he/she sends to A a `TimeStampResponse` with the time-stamp contained in the `TimeStampToken` element. This element will be different for each of the schemes applied, and it is this element that will be illustrated in the following examples. Note that some descendant elements of `ds:Signature` were omitted to save space. Exc-C14N should be included for all references and the C14N algorithm in `ds:SignedInfo`.

4.5.1 Linear linking

We apply our XML format to the Linear linking scheme of Haber *et al.* [18]. For the details of this scheme and the meaning of `L_n`, we refer to Sect. 5.1 of the cited paper. This example is a time-stamp for the variant on the basic scheme, with $k = 2$.

```
<TimeStampToken xmlns="our-timestamp-URI">
  <MessageImprints Id="messageImprintsID">
    <DigestAlgValue Id="digestvalue1">
      <DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue> X_n </DigestValue>
    </DigestAlgValue>
  </MessageImprints>
  <TSTInfo Id="TSTInfoID">
    <SerialNumber>n</SerialNumber>
    <GenTime>2002-11-22Z12:00:00:00</GenTime>
    <ID Name="ID-of-A"/>
  </TSTInfo>
  <ds:Signature>
    <ds:SignedInfo>
      <ds:Reference URI="#messageImprintsID"/>
      <ds:Reference URI="#TSTInfoID"/>
      <ds:Reference URI="#BindingInfoID"/>
    </ds:SignedInfo>
    <ds:SignatureValue>
      msbOAt...NwdrSJX9fcL6=
    </ds:SignatureValue>
  </ds:Signature>
  <BindingInfo Algorithm="LinearLinking-URI-HS91"
               Id="BindingInfoID">
    <DigestAlgValue Idrefs="digestvalue1 TSTInfoID">
      <DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>
        dQ8Nx7zFrCmyiCQ9vYk0N2N88MI=
      </DigestValue>
    </DigestAlgValue>
    <LinkingInfo Algorithm="LinearLinking-k2">
      <Head>
        <Node Id="hash(L_(n-1))">
```

```
        <DigestValue>H(L_(n-1))</DigestValue>
      </Node>
      <Node Reference="tstinfo-n-1"/>
      <Node Reference="imprint-n-1"/>
      <Node Id="hash(L_(n-2))">
        <DigestValue>H(L_(n-2))</DigestValue>
      </Node>
      <Node Reference="tstinfo-n-2"/>
      <Node Reference="imprint-n-2"/>
    </Head>
    <Tail>
      <Node Reference="tstinfo-n+1"/>
      <Node Reference="tstinfo-n+2"/>
    </Tail>
    <ds:Object>
      <TSTInfo Id="tstinfo-n-1">...</TSTInfo>
      <MessageImprints Id="imprint-n-1">
        ...
      </MessageImprints>
      <TSTInfo Id="tstinfo-n-2">...</TSTInfo>
      <MessageImprints Id="imprint-n-2">
        ...
      </MessageImprints>
      <TSTInfo Id="tstinfo-n+1">
        <Id Name="ID-of-n+1"/>
      </TSTInfo>
      <TSTInfo Id="tstinfo-n+2">
        <Id Name="ID-of-n+2"/>
      </TSTInfo>
    </ds:Object>
  </LinkingInfo>
</BindingInfo>
</TimeStampToken>
```

4.5.2 *Binary Linking*

For the details of this scheme and meaning of the symbols, we refer to Buldas *et al.* [12], Sect. 5.

```
<TimeStampToken xmlns="our-timestamp-URI">
  <MessageImprints Id="MessageImprintsID">
    <DigestAlgValue Id="digestvalue1">
      <DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue> X_n </DigestValue>
    </DigestAlgValue>
  </MessageImprints>
  <TSTInfo Id="TSTInfoID">
    <SerialNumber> n </SerialNumber>
  </TSTInfo>
  <ds:Signature>
    <ds:SignedInfo>
      <ds:Reference URI="#TSTInfoID"/>
      <ds:Reference URI="#BindingInfoID"/>
    </ds:SignedInfo>
  </ds:Signature>
  <BindingInfo Id="BindingInfoID"
               Algorithm="BinaryLinking-URI-BLLV98">
    <DigestAlgValue Idrefs="TSTInfoID digestvalue1">
      ...
    </DigestAlgValue>
    <LinkingInfo Algorithm="m-value">
      <Head>
        <Node Id="L_n">
          <DigestValue> L_n </DigestValue>
        </Node>
        <Node Id="L_n_1">
          <DigestValue> L_n_1 </DigestValue>
        </Node>
         ...
        <Node Id="L_n_q">
          <DigestValue> L_n_q </DigestValue>
        </Node>
        <Node Reference="prev-round"/>
      </Head>
    </LinkingInfo>
    <PublishedInfo>
      <Node Id="prev-round">
        <DigestValue>L_(eps_(r-1))</DigestValue>
      </Node>
    </PublishedInfo>
  </BindingInfo>
</TimeStampToken>
```

The full time-stamp, including the tail, will be sent when the current linking round finishes.

5. CONCLUSION AND OPEN ISSUES

The purpose of this paper was to study existing time-stamping schemes and standards, and, based on that study, to propose ideas for an XML format for time-stamps. We note that there are many different time-stamp protocols with widely varying data structures. Therefore, it was not an easy task to provide one single structure that covers them all. The resulting format can be a basis for the definition of a mature time-stamp protocol. We focused on the time-stamp token format, so there remain some gaps to be filled:

- We did not define a protocol for verifying time-stamps. This is necessary if the cooperation of the TSA is required or wanted for verification.
- For processing rules, TSA behaviour and polices, we refer to the ETSI and ISO documents covering this subject. Also, in PKIX-TSP, some requirements for the TSA can be found, for simple schemes based on digital signatures.
- We studied some of the proposals for distributed time-stamps. In most of them, the TSAs return a partial time-stamp that can be used to compute the full token. We believe that these types of time-stamps can be fitted in our structures. If there exist time-stamping schemes based on distributed computations, or a real splitting of secret information among the TSAs, we might run into problems. A possible solution would be to introduce a proxy that performs the distribution and recombination tasks for the user.
- The identifiers for the algorithms are not yet defined.

It was also noted that our proposal depends heavily on ID/IDREF pairs. A document will be broken if a time-stamp containing an ID that already exists in the document is added to that document. Moreover, if the same IDs are used for each instance of a specific type of time-stamp, it is not possible to add two time-stamps of that kind to one document. Note that this action is very likely in XAdES. Even if we would resolve this problem by choosing appropriate IDs, it would still be possible to break a document by adding the same time-stamp twice. We can eliminate some ID instances by defining custom transformations for signatures on time-stamps, as suggested by one of the referees. At least two transformations are necessary to accommodate all possible schemes. Furthermore, the ID/IDREFS construction to indicate which digest values are taken for the production of linking information can be replaced by URIs. The rest of the IDs are used to add structure to the linking information, but that structure can be assumed to be known by the application.

6. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for their useful comments and suggestions.

7. REFERENCES

[1] ISO/IEC 18014-1. Information technology – Security techniques – Time-stamping services – Part 1: Framework. Draft available at `http://oberon.postech.ac.kr/kiisc-sis/timestamp/`, 2002.

[2] ISO/IEC FDIS 18014-2. Information technology – Security techniques – Time-stamping services – Part 2: Mechanisms producing independent tokens. Draft available at `http://oberon.postech.ac.kr/kiisc-sis/timestamp/`, 2002.

[3] ISO/IEC WD 18014-3. Information technology – Security techniques – Time Stamping Services – Part 3: Mechanisms producing linked tokens. Draft available at `http://oberon.postech.ac.kr/kiisc-sis/timestamp/`, 2002 (working draft).

[4] ISO/IEC 8824-1. Information Technology – Abstract Syntax Notation One (ASN.1): Specification of Basic Notation, 1998.

[5] ISO/IEC 8825-1. Information Technology – ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), 1998.

[6] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). Available at `http://www.ietf.org/html.charters/pkix-charter.html`, April 2002, work in progress.

[7] Arne Ansper, Ahto Buldas, Märt Saarepera, and Jan Willemson. Improving the availability of time-stamping services. In *The 6th Australasian Conference on Information Security and Privacy - ACISP'2001*, Lecture Notes in Computer Science, pages 360–375, Sydney, Australia, July 2001. Springer-Verlag.

[8] Dave Bayer, Stuart Haber, and W. Scott Stornetta. Improving the Efficiency and Reliability of Digital Time-Stamping. In R. Capocelli, A. De Santis, and U. Vaccaro, editors, *Sequences II: Methods in Communication, Security and Computer Science*, pages 329–334. Springer-Verlag, 1993. Available at `http://www.surety.com/solutions/DN/presentation.html`.

[9] J. Benaloh and M. de Mare. Efficient Broadcast Time-Stamping. Technical Report TR-MCS-91-1, Clarkson University, Department of Mathematics and Computer Science, April 1991.

[10] J. Benaloh and M. de Mare. One-way Accumulators: A Decentralized Alternative to Digital Signatures. In T. Helleseth, editor, *Advances in Cryptology - Proceedings of EuroCrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, Lofthus, Norway, May 1993. Springer-Verlag.

[11] J. Boyer, D. Eastlake III, and J. Reagle. Exclusive XML Canonicalization Version 1.0 (EXC-C14N). `http://www.w3.org/TR/xml-exc-c14n/`, Juli 2002.

[12] Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-Stamping with Binary Linking Schemes. In Hugo Krawczyk, editor, *Advances on Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 486–501, Santa Barbara, USA, August 1998. Springer-Verlag.

[13] Ahto Buldas, Helger Lipmaa, and Berry Schoenmakers. Optimally Efficient Accountable Time-Stamping. In *Public Key Cryptography - PKC'2000*, Lecture Notes in Computer Science, pages 293–305, Melbourne, Australia, 2000. Springer-Verlag.

[14] D. Eastlake, J. Reagle, and D. Solo. XML-Signature Syntax and Processing. `http://www.w3.org/Signature/`, February 2002.

[15] Donald Eastlake and Joseph Reagle (eds). XML Encryption Syntax and Processing. `http://www.w3.org/Encryption`, August 2002.

[16] ETSI. European Telecommunications Standards Institute, Security Technical Committee(ETSI-SEC). `http://www.etsi.org/sec`.

[17] ETSI-SEC-ESI. XML Advanced Electronic Signatures (XAdES), ETSI TS 101 903. Available at `http://portal.etsi.org/sec/el-sign.asp`, February 2002.

[18] S. Haber and W. S. Stornetta. How to Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2):99–111, 1991. Available at `http://www.surety.com/solutions/DN/presentation.html`.

[19] Dave Hollander and C. M. Sperberg-McQueen (chairs). XML Schema Working Group. `http://www.w3.org/XML/Schema`.

[20] R. Housley. S/MIME Mail Security (smime), IETF Working Group. `http://www.ietf.org/html.charters/pkix-charter.html`.

[21] R. Housley. Cryptographic Message Syntax. Available at `http://www.ietf.org/html.charters/smime-charter.html`, April 2002, work in progress.

[22] Michael Just. Some timestamping protocol failures. In *Proceedings of the Symposium on Network and Distributed Security (NDSS 98)*, pages 89–96, San Diego, CA, USA, March 1998.

[23] S. Kent and T. Polk (chairs). Public-Key Infrastructure(X.509)(pkix), IETF Working Group. `http://www.ietf.org/html.charters/pkix-charter.html`.

[24] Ralph C. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 122–134, 1980.

[25] European Parliament. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. Available at `http://europa.eu.int/information_society/topics/ebusiness/ecommerce/3in%formation/law&ecommerce/legal/digital/index_en.htm`, Januari 2000.

[26] Bart Preneel, Bart Van Rompay, Jean-Jacques Quisquater, Henri Massias, and J. Serret Avila. Specification and Implementation of a Timestamping System. Technical Report TIMESEC WP4, Université Caltholique de Louvain, 1999.

[27] Masashi Une. The Security Evaluation of Time

Stamping Schemes: The Present Situation and Studies, IMES Discussion Paper Series. Technical Report 2001-E-18, Institute for Monetary and Economic Studies, Japan, December 2001. http://www.imes.boj.or.jp/english/publication.html.

[28] ITU-T X.693. Information technology – ASN.1 encoding rules: XML encoding rules (XER). Available at http://www.itu.int/ITU-T/studygroups/com17/languages/, December 2001.

[29] BSR X9.95-200x. Trusted Timestamp. to be published.

[30] BSR X9.96-200x. XML Cryptographic Message Syntax (XCMS). to be published.

[31] X9F3. Data and Information Security – Protocols. http://www.x9.org/.

APPENDIX

A. TIMESTAMP SCHEMA

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE schema PUBLIC
 "-//W3C//DTD XMLSchema 200102//EN"
 "http://www.w3.org/2001/XMLSchema.dtd" [
  <!ATTLIST schema
    xmlns:tsp CDATA #FIXED
      "http://www.cosic.be/2002/08/xmltsp#"
    xmlns:ds CDATA #FIXED
      "http://www.w3.org/2000/09/xmldsig#"
    xmlns:xades CDATA #FIXED
      "http://uri.etsi.org/01903/v1.1.1#"> ]>
<xs:schema
 targetNamespace="http://www.cosic.be/2002/08/xmltsp#"
 xmlns:xades="http://uri.etsi.org/01903/v1.1.1#"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:tsp="http://www.cosic.be/2002/08/xmltsp#"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">
  <xs:import
   namespace="http://www.w3.org/2000/09/xmldsig#"
   schemaLocation="xmldsig-core-schema.xsd"/>
  <xs:import
   namespace="http://uri.etsi.org/01903/v1.1.1#"
   schemaLocation="XAdES.xsd"/>

  <xs:element name="TimeStampRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tsp:MessageImprints"/>
        <xs:element
          ref="xades:SignaturePolicyIdentifier"
          minOccurs="0"/>
        <xs:element name="Nonce"
                    type="xs:int" minOccurs="0"/>
        <xs:element ref="ds:Object" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="CertReq"
                    type="xs:boolean" use="optional"/>
      <xs:attribute name="Type"
                    type="xs:anyURI" use="optional"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="MessageImprints">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DigestAlgValue"
                    type="tsp:DigestAlgValueType"
                    maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Id"
                    type="xs:ID" use="optional"/>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="DigestAlgValueType">
    <xs:complexContent>
      <xs:extension base="xades:DigestAlgAndValueType">
        <xs:attribute name="Id"
                      type="xs:ID" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="TimeStampResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Status">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="MajorStatus">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name="Code"
                                    type="xs:int"
                                    use="required"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="FailCode" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name="Code"
                                    type="xs:int"
                                    use="required"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="TimeStampToken"
                    type="tsp:TimeStampTokenType"
                    minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="TimeStampTokenType">
    <xs:complexContent>
      <xs:restriction base="ds:SignatureType">
        <xs:sequence>
          <xs:element ref="tsp:References"
                      minOccurs="0"/>
          <xs:element ref="tsp:MessageImprints"
                      minOccurs="0"/>
          <xs:element name="TSTInfo"
                      type="tsp:TSTInfoType"/>
          <xs:element ref="ds:Signature"
                      minOccurs="0"/>
          <xs:element ref="tsp:BindingInfo"
                      minOccurs="0"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="References">
    <xs:complexType> <xs:choice>
```

```
        <xs:element ref="ds:Reference"
                    maxOccurs="unbounded"/>
        <xs:element name="XADESInfoLink">
          <xs:complexType>
            <xs:attribute name="idref"
                          type="xs:IDREF"
                          use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:choice> </xs:complexType>
    </xs:element>

    <xs:complexType name="TSTInfoType">
      <xs:sequence>
        <xs:element ref="xades:SignaturePolicyIdentifier"
                    minOccurs="0"/>
        <xs:element name="SerialNumber"
                    type="xs:integer" minOccurs="0"/>
        <xs:element name="GenTime"
                    type="tsp:ExtendedDateTimeType"
                    minOccurs="0"/>
        <xs:element name="Accuracy" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Seconds" type="xs:int"/>
              <xs:element name="MilliSeconds"
                          minOccurs="0">
                <xs:simpleType>
                  <xs:restriction base="xs:short">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="999"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="MicroSeconds"
                          minOccurs="0">
                <xs:simpleType>
                  <xs:restriction base="xs:short">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="999"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Ordering"
                    type="xs:boolean" minOccurs="0"/>
        <xs:element name="Nonce"
                    type="xs:int" minOccurs="0"/>
        <xs:element name="TSA" minOccurs="0">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute name="URI"
                              type="xs:anyURI"
                              use="optional"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="Id" minOccurs="0">
          <xs:complexType>
            <xs:attribute name="Name"
                          type="xs:string"
                          use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="Id"
                    type="xs:ID" use="optional"/>
    </xs:complexType>

    <xs:complexType name="ExtendedDateTimeType">
      <xs:simpleContent>
        <xs:extension base="xs:dateTime">
          <xs:attribute name="MilliSeconds"
                        use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:short">
                <xs:pattern value="[0-9]{3}"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="MicroSeconds"
                        use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:short">
                <xs:pattern value="[0-9]{3}"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

    <xs:element name="BindingInfo">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DigestAlgValue">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="tsp:DigestAlgValueType">
                  <xs:attribute name="IdRefs"
                                type="xs:IDREFS"
                                use="optional"/>
                </xs:extension>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
          <xs:element name="AggregationInfo"
                      minOccurs="0">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="tsp:ChainType">
                  <xs:attribute name="Algorithm"
                                type="xs:anyURI"
                                use="optional"/>
                </xs:extension>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
          <xs:element name="LinkingInfo">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Head"
                            type="tsp:ChainType"
                            minOccurs="0"/>
                <xs:element name="Tail"
                            type="tsp:ChainType"
                            minOccurs="0"/>
                <xs:element ref="ds:Object"
                            minOccurs="0"/>
              </xs:sequence>
              <xs:attribute name="Algorithm"
                            type="xs:anyURI"
                            use="optional"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="PublishedInfo" minOccurs="0">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="tsp:ChainType">
                  <xs:attribute name="Location"
                                type="xs:anyURI"
                                use="optional"/>
```

```
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="Id"
                    type="xs:ID" use="optional"/>
      <xs:attribute name="Algorithm"
                    type="xs:anyURI" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="ChainType">
    <xs:sequence>
      <xs:element name="Node"
                  type="tsp:NodeType"
                  maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Id"
                  type="xs:ID"
                  use="optional"/>
  </xs:complexType>

  <xs:complexType name="NodeType">
    <xs:choice>
      <xs:sequence>
        <xs:element ref="ds:DigestMethod"
                    minOccurs="0"/>
        <xs:element ref="ds:DigestValue"
                    minOccurs="0"/>
      </xs:sequence>
      <xs:element name="BinaryContent"
                  type="xs:base64Binary"/>
    </xs:choice>
    <xs:attribute name="Id"
                  type="xs:ID" use="optional"/>
    <xs:attribute name="Reference"
                  type="xs:IDREF" use="optional"/>
    <xs:attribute name="Alignment"
                  type="xs:string" use="optional"/>
  </xs:complexType>
</xs:schema>
```