XML Security Time Stamping Protocol

Axelle Apvrille Vincent Girier
Storage Technology European Operations
1 Rd Point Général Eisenhower
31106 Toulouse, France
{Axelle_Apvrille,Vincent_Girier}@storagetek.com

Abstract

Existing time stamp protocols are based on an exchange between a client, sending a document hash, and an authority which signs submitted document with current time. This protects against content and date forgery.

Unfortunately, the protocol relies upon ASN.1 whose encoding is difficult to manipulate - an issue at which XML is very successful. Consequently, this paper proposes an XML adaptation of the time stamping protocol. It focuses on possible integration with XML Signatures and uses XML schemas to describe time stamping messages. Finally, this work provides the basis for wider XML developments of time stamping models.

Keywords

TIME STAMP, XML, DIGITAL SIGNATURE, ASN.1, DER.

1 Introduction

IMPORTANT documents in real life (purchase orders, insurance contracts, partnership agreements...) are always first dated, then signed. Doing so, the signer acknowledges both the document is authentic and its date is valid. Electronic document time stamping sticks to the same philosophy: a date is first appended to the document and then digitally signed. The process protects the document against both content and date forgery.

Unfortunately, computer data outputs are merely byte streams and consequently much more abstract to human beings than handwriting for instance. This probably partly explains the success of XML (eXtensible Markup Language) in the late 90's: simple "human" oriented syntax, extensible, easy to parse... XML is undoubtedly a major improvement to document syntax. Back to time stamping, the idea this paper proposes is to adapt a time stamping protocol to XML and consequently benefit from XML's numerous advantages. This would probably help digital time stamps become less abstract, more human readable and easier to integrate to products.

Hence, this paper studies first time stamping protocol defined in [TSP01] and what points need to be improved. Third section examines possible solutions and limits using existing work such as XML-encoding rules, XML Signatures and XML Advanced Signatures. We then propose our XML time stamping adaptation in section 4, and finally how it integrates to various time stamping models in section 5.

2 Analysis of a time stamping protocol

This section details time stamping protocol defined in [TSP01]. Though the mechanisms this protocol describes work perfectly well, this paper intends to highlight what parts need improvements so as to be more convenient to manipulate.

2.1 The scope of Time Stamping Protocol (TSP)

[TSP01] defines a time stamping protocol shortened *TSP*[1]. Basically, two different entities are involved in time stamping process:

- a client (the user) sends over a time stamp request asking for one of his documents to be time stamped,
- a server, named *time stamp authority* (TSA), is responsible for time stamping. It is trusted to maintain an accurate clock and sends back a *time stamp response*.

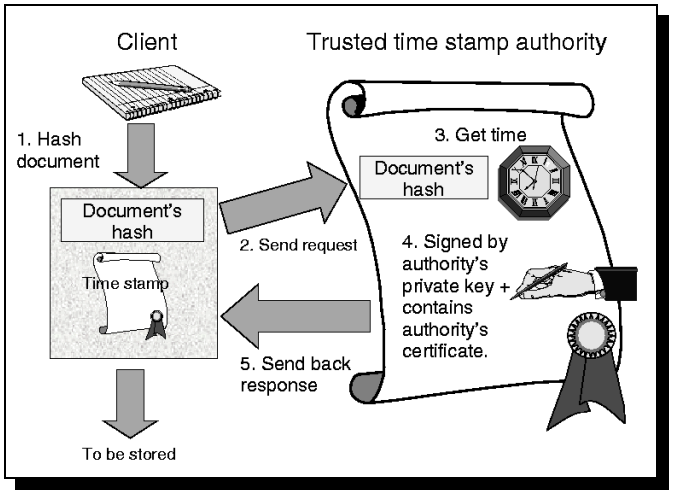

Figure 1: Time stamping process.

Figure 1 illustrates the process. TSP merely describes the *content* of each message sent between the client and the TSA, i.e the format of each message and the precise meaning of each field. It does not focus on *how* messages are transmitted (a few suggestions are provided though).

2.2 Time stamp request

The structure of a time stamp request is very simple. It contains the hash of the document to time stamp, a protocol version number and some more optional parameters. Note the hash of the document is sent across to TSA, not the *full* document. As a matter of fact, for confidentiality and/or efficiency reasons, client might not wish to send over his data in clear text. It's just as simple for him to send over a unique representation of his data.

2.3 Time stamp response

The time stamp response structure is much more complicated: the document hash is joint to the current date and forms a *time stamp information* structure (`TSTInfo`). Then, the resulting structure is encapsulated in a *cryptographic message* which is signed by the TSA. Hence, time stamp's security is not part of TSP but is delegated to *Cryptographic Message Syntax* [CMS99]. TSP lies above CMS. Figure 2 gives an overview of what each document explains.

A time stamp response is secured through the following steps (please refer to figure 3):

1. First, a *status* is included to the time stamp response. It briefly describes the status of the time stamping process. If no error has been encountered, the effective time stamp token is appended to the status.

[1]RFC 3161 suggests this abbreviation and we have therefore chosen it throughout this document. However, please note this does not mean *Trusted Service Providers*.

2. Then, TSP defines the time stamp token field as a cryptographic message (`ContentInfo` structure) carrying signed data information (`SignedData` structure).

3. Useful time stamping data (containing for instance date and document's hash) is packed into a time stamp information structure (`TSTInfo`). This information is first DER-encoded[DER97] then encapsulated in a field of the `SignedData` structure. DER (Distinguished Encoding Rules) are a set of rules which explain how to encode ASN.1 data into a sequence of bytes. So, DER encoding does not secure the time stamping information (`TSTInfo`) but provides a method to pack and transmit data.

4. Then a set of attributes to be signed must be filled up. For time stamping, two attributes are mandatory: a *content type attribute* and a *message digest attribute*. Both attributes refer to the encapsulated information, i.e. `TSTInfo`. The content type attribute indicates encapsulated information's type is `TSTInfo`, and the message digest attribute stores the digest of the `TSTInfo` structure.

5. Finally, this set of attributes is DER encoded, digested and signed by TSA's private key.

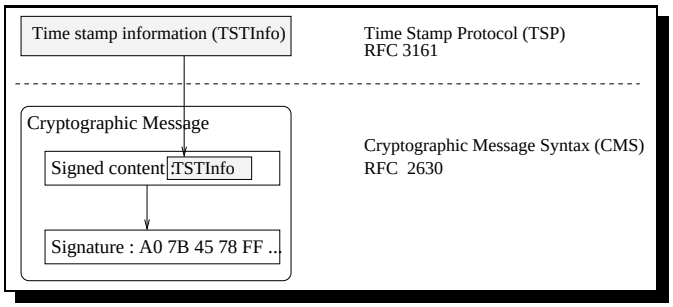

Figure 2: *Definition scope of TSP and CMS. TSP relies upon CMS for security.*

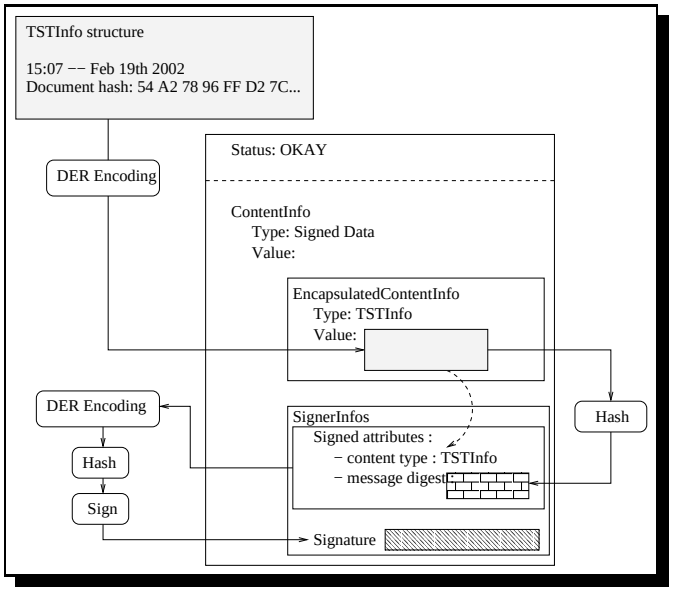

Figure 3: *Building a time stamp response: the different steps.*

2.4 Limits of TSP

Security mechanisms of TSP are not at stake here but rather their format. As a matter of fact, TSP's grammar is based upon ASN.1, which describes structures, types but not values. The real values matching ASN.1 structures need to be encoded using specific set of rules such as BER, CER or DER[DER97]. Time stamping messages therefore encounter encoding rules limitations: BER, CER and DER have been thought in terms of compactness but not for reading or manipulation. For instance, the encoding of a simple structure is shown at figure 4. The

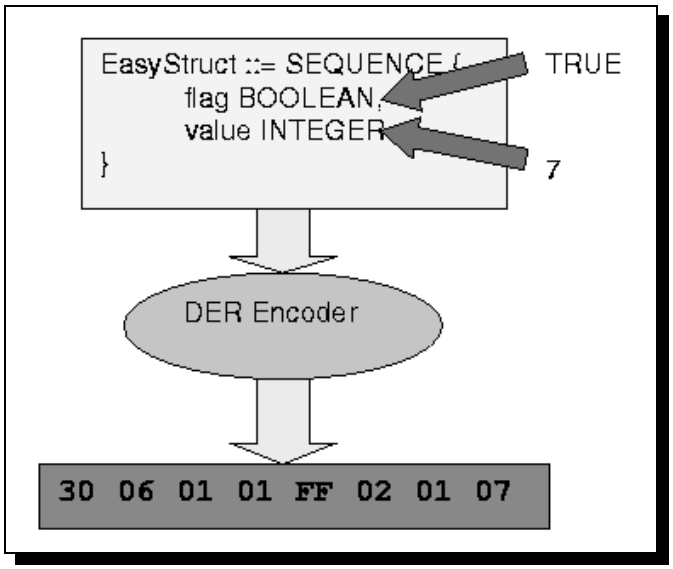

Figure 4: *DER encoding of a simple data values of type* `EasyStruct`.

result is definetely abstract to human mind, and we believe that, in the context of documents that are meant to be manipulated by many different products, displayed, parsed etc, this will only hold back users or implementors from using it. There are probably other ways to produce a more convenient output, but XML seemed to us perfectly suited to solve those problems: its markup language is easy to understand, it is extensible and therefore adapts to various contexts, it offers a large panel of existing tools (XML parsers, XML editors, XML validators, XML Stylesheets...). This paper will consequently restrict to an XML solution.

3 A guided tour of existing work to time stamp XML documents

3.1 XER: a simple solution ?

The most obvious solution which comes up is to replace use of DER encoding by an XML encoding. To do so, another set of encoding rules, named XER for *XML Encoding Rules*[XER01], have been defined. From an ASN.1 structure and data values, XER produces XML output. For instance, the encoding of data value in figure 4 is:

```
<EasyStruct>
 <flag> true </flag>
 <value> 7 </value>
</EasyStruct>
```

No doubt this is much more clear than `30 06 01 01 FF 02 01 07`!
Down into TSP's grammar, DER encoding is used to:

1. fill up the `EncapsulatedContentInfo` structure,

2. calculate the signature of the signed attributes,

3. and finally, of course, to transmit the global time stamp response data.

So, if we can replace DER with XER for those 3 points, the trick is done.

Unfortunately, TSP strictly requires the encapsulated field is *DER encoded* (see [TSP01, §2.4.2, page 7]) (first two points), and for the signature, CMS explicitly mentions use of DER encoding ([CMS99, §5.4, page 10]). Switching to XER encoding would mean breaking conformity to either TSP or CMS, and would consequently require re-writing them.

Hence, it is only possible to replace DER encoding of the global time stamp response structure (third point). To do so, ASN.1 to XML translators exist (see work of Imamura and Maruyama[IM00]). However, as the inner encapsulated content information and signature are stuck with DER encoding, the XER encoded response would approximatively look like figure 5. The result is not satisfactory for three reasons. First, both XER and DER

```
- <TimeStampResp>
  + <status>
  - <timeStampToken>
      <contentType>id-signedData</contentType>
    - <SignedData>
        <version>3</version>
        <digestAlgorithms>...</digestAlgorithms>
        <encapContentInfo>30 06 01 01 ... (DER encoding)</encapContentInfo>
        <certificates>...</certificates>
        <crls>...</crls>
      - <signerInfos>
        - <signerInfo>
            <version>1</version>
            <sid>...</sid>
            <digestAlgorithm>...</digestAlgorithm>
          - <signedAttrs>
            - <Attribute>
                <attrType>id-contentType</attrType>
              - <attrValues>
                  <attrValue>id-ct-TSTInfo</attrValue>
                </attrValues>
              </Attribute>
            - <Attribute>
                <attrType>id-messageDigest</attrType>
              - <attrValues>
                  <attrValue>4A 2F 75 ... (digest of DER encoded TSTInfo)</attrValue>
                </attrValues>
              </Attribute>
            </signedAttrs>
            <signature>A7 78 45 B4 FF 99 10 ...</signature>
          </signerInfo>
        </signerInfos>
      </SignedData>
    </timeStampToken>
  </TimeStampResp>
```

Figure 5: *XER-encoding of a time stamp response record.*

are used, which is not very homogeneous. It forces implementors to cope with two different document syntaxes. Second, notice the most important part of the time stamp response (i.e. the time stamp information contained in `TSTInfo`) is still DER encoded. If one portion were to be improved, it's this one... By DER encoding the time

stamp information, we're completely missing our initial goal: we have only complicated time stamping mechanism without really improving its output. Finally, the signature is quite misleading. It is neither a signature of the XML document nor a signature of XER-encoded signed attributes: it is the signature of *the DER encoding of all signed attributes*. This is not very convenient because signature verification will have first to DER encode the signed attributes, and then verify the signature.

As a conclusion, simply switching from DER to XER encoding cannot solve our problems. Time stamp adaptation to XML requires a time stamp protocol is re-written without ASN.1. Time stamp's security being guaranteed by digital signatures, next paragraph will focus on adaptation of digital signatures to XML i.e *XML signatures*.

3.2 Using XML Signatures for time stamps

Time stamp's security is guaranteed by a digital signature. In the case of TSP, this signature is done in a cryptographic message which relies on ASN.1 structures. For an all-XML scheme, the W3 Consortium recommends using *XML Signatures*[DSIG02]. Please note at the time this paper is written, the XML Signature draft standard has just been issued (March 2002) and how new all the concern of security over XML is.

This recommendation describes how to perform signatures of XML documents, with signature itself complying to an XML syntax. The signature can be defined as a *detached document* (signature is separated from the document), *enveloping* the document (the document is found *inside* the signature), *or enveloped* in the document (the signature is "appended" to the document). With XML Signatures, signatures are really part of - or a complete - XML document, therefore keeping XML advantages.

In the case of time stamping, basically, a timed XML Signature of an XML document needs to be done. Luckily, the W3C recommendation provides us with a time stamping example[DSIG02, §2.2] shown at figure 6. This is a

```
- <Signature Id="MySecondSignature">
  - <SignedInfo>
      <!--  ...  -->
      <Reference URI="http://www.w3.org/TR/xml-stylesheet/" />
      <!--  ...  -->
    - <Reference URI="#AMadeUpTimeStamp" Type="http://www.w3.org/2000/09/xmldsig#SignatureProperties">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>k3453rvEPO0vKtMup4NbeVu8nk=</DigestValue>
      </Reference>
    </SignedInfo>
    <!--  ...  -->
  - <Object>
    - <SignatureProperties>
      - <SignatureProperty Id="AMadeUpTimeStamp" Target="#MySecondSignature">
        - <timestamp xmlns="http://www.ietf.org/rfcXXXX.txt">
            <date>19990908</date>
            <time>14:34:34:34</time>
          </timestamp>
        </SignatureProperty>
      </SignatureProperties>
    </Object>
  </Signature>
```

Figure 6: *A detached timed signature example.*

detached signature example. The `SignedInfo` part contains a list of references to be signed, and a small time stamp is included in the *signature properties*. The signature value is not shown in that example, but signs all `SignedInfo`. Having a closer look to the various references that are signed, one can notice the second reference actually points to the signature properties: this means the time stamp will be signed too, along with the other URIs. However, this timed signature example cannot quite be compared to a time stamp in TSP. The major differences are:

- the example makes a direct reference to the document to time stamp, whereas a TSA has only access to its hash,
- the time stamp in the example is simplistic: time stamping policy, accuracy etc are missing,
- the example is actually more a *timed signature* (time is a property of the signature) than a *signed time stamp* (time stamp is signed) like in TSP.

One must keep in mind XML Signatures have been thought only for *signing* and certainly not as a *time stamping protocol*. To compare the XML model to the ASN.1/DER model, XML Signatures should be related to CMS, but not TSP.

3.3 XML Advanced Electronic Signatures Draft

Previous section has shown XML Signatures could be suited to secure time stamps over XML documents. However XML Signatures are not a time stamping protocol and this leaves us with the question of whom to compare TSP with. This section will try and investigate if *XML Advanced Electronic Signatures*[XADES02] proposed by the ETSI Working Group are better suited to our needs.

Actually, as XML Signatures are very open (you can barely put anything in the signature, or add whatever property you want), the XML Advanced Electronic Signatures draft tries to suggest common and useful structures to include to signatures. Intended for electronic documents that remain valid over long periods, it furthermore explicitly mentions use of time stamps (for instance, in the XAdES-T form [XADES02, §4.3,§7.3]).

Dating documents is achieved through two different mechanisms (see Figure 7):

1. signer is requested to mention the date at which he performed the signature. The date is written as a signed property (named `SigningTime`), and is signed with all content (`SignedInfo`) and signing properties (`SignedProperties`) by the signer.
2. and optionally an external time stamp, issued by a time stamp authority, may be added as an unsigned property.

One should note that roles of signer and TSA are strictly separated. On one side, signer is trusted for its signature of the document, but not specifically for the signing time it mentions. On the other side, TSA is trusted for time stamping. The time stamp time stamps the XML signature value and is signed by the TSA, but not by the signer (which explains why time stamp is referred to as an "unsigned property"). In this paper, we are rather concerned by this form of time stamping, as signer's signing time is not trusted.

Unfortunately, having a close look at the draft, we notice time stamp's format is not defined at all. The draft merely suggests either to include an ASN.1 time stamp (!) stored in Base 64 format, or references to a future XML type. As a matter of fact, the draft is not meant to define XML time stamping : *"since at the time being there is no standard for an XML time stamp, we provide a placeholder for future use"*[XADES02, §7.1.4].

As a conclusion, the XML Advanced Signatures draft cannot be considered as TSP for XML documents. It suggests how to include time stamps (and other parameters) in signed XML documents, but leaves time stamps undefined.

3.4 Comparison between proposals

We have demonstrated that XER encoding (§3.1) does not offer a proper adaptation of TSP to the XML world. XML signatures (§3.2) and XML Advanced Signatures (§3.3) are both interesting. Unfortunately, they are both dedicated to defining how to sign XML documents - eventually including time stamps - but do not propose any substitute for a time stamping protocol. Currently, we are not aware of any other work concerning time stamps in XML.

Figure 7: *An XML Advanced Electronic Signature including an external time stamp (XAdES-T form). The '+' sign indicates that leaf information exists but is not shown (collapsed mode). The '-' sign indicates an expanded mode.*

In next section, we consequently intend to propose a solution to this missing part: a definition to time stamping messages with XML. This solution is based on XML schemas for message description, and secured by XML Signatures. Resulting time stamp could also be included in XML Advanced Signatures. We believe this proposal (please refer to table 1):

- proposes an adaptation of TSP to XML,
- benefits from XML's advantages in terms of manipulation and simplicity,
- is homogeneous, i.e not a mixture of ASN.1/DER and XML, but 100% XML,
- uses XML Signatures, and is compatible with XML Advanced Signatures.

4 XML Schemas for time stamping

Our time stamping protocol is directly inspired by [TSP01] and, likewise, the transport means are left aside, the securisation of the data being realised thanks to *XML Signatures* [DSIG02] as it was by *CMS* [CMS99] for the original protocol. Hence, the protocol is basically defined by the structure of the request sent by a client to the TSA and the response of the TSA to the client which includes the time stamp information.

These structures are defined using XML schemas, which provides great readability of the structure definitions.

The namespace definitions for all the schemas included in this section are given in Figure 8. Note that the target namespace is defined here as *"http://ournamespace"* as a provisional definition. As indicated, `ds` introduces *XML Signatures* types and `tsp` introduces our own types.

	Time Stamp Protocol support	Ease of manipulation, human readable ...	Homogenous	Compatible with XML Signatures
XER solution (§3.1)	Yes	Medium (time stamp information is still in DER)	No (DER & XER)	No
XML signatures (§3.2)	No	Yes	Yes	Yes
XML Advanced Signatures (§3.3)	No	Yes (but complex)	Not always (may contain DER & XML)	Yes
Our proposal : XML Time stamp	Yes	Yes	Yes	Yes

Table 1: Features of different XML time stamping solutions

4.1 Time stamp request

The XML schema definition of the format of the request is given in Figure 9. Similarly to *TSP*, which used *CMS* [CMS99] structures to describe the format of the `HashedMsg` and `Algorithm` fields, we use the `DigestValueType` (which is in fact a base 64 binary string) and `DigestMethodType` defined in *XML Signatures* [DSIG02].

The `Nonce` field is defined as a long integer (64 bits in XML) so as to ensure that it has a high probability of uniqueness and thus provides a means to check that a Request and a Response actually match.

```
- <schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://ournamespace"
    xmlns:tsp="http://ournamespace"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    elementFormDefault="qualified">
```

Figure 8: *Namespace definitions*

```
- <element name="TimeStampReq" type="tsp:TimeStampReqType">
  - <complexType name="tsp:TimeStampReqType">
    - <sequence>
        <element name="MsgImprint" type="tsp:MsgImprintType" />
        <element name="ReqPolicy" type="tsp:TSAPolicyType"
          minOccurs="0" />
        <element name="Nonce" type="long" minOccurs="0" />
      </sequence>
    </complexType>
    <attribute name="Version" type="positiveInteger" />
    <attribute name="CertReq" type="boolean" default="false" />
  </element>
- <complexType name="tsp:MsgImprintType">
  - <sequence>
      <element name="HashedMsg" type="ds:DigestValueType" />
    </sequence>
    <attribute name="Algorithm" type="ds:DigestMethodType"
      use="required" />
  </complexType>
- <complexType name="TSAPolicyType">
  - <sequence>
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
    <attribute name="Policy" type="anyURI" />
  </complexType>
```

Figure 9: *Request definition*

The `ReqPolicy` type which was originally defined as an OID is here defined as a reference to a URI describing a specific `Policy`, and an unbounded set of implicit parameters, by similarity with the way algorithm was defined in *XML Signatures*.

Note the time stamp request does not identify the requester as it is not required by the TSA. In situations where the TSA requires the identity of the requesting entity, alternate identification and authentication means should be used.

4.2 Time stamp response

The format of the response is defined in Figure 10. It enumerates all possible `Status` and `FailInfo` values which appear as explicit strings in the response, thus providing immediate information for a human end-user.

```
– <element name="TimeStampResp" type="tsp:TimeStampRespType">
  – <complexType name="tsp:TimeStampRespType">
    – <sequence>
        <element name="StatusInfo" type="tsp:StatusInfoType" />
        <element name="TimeStampDoc" type="ds:Signature" />
      </sequence>
    </complexType>
  </element>
– <complexType name="tsp:StatusInfoType">
  – <sequence>
      <element name="Status" type="tsp:StatusType" />
      <element name="StatusString" type="string" minOccurs="0" />
      <element name="FailInfo" type="tsp:FailInfoType" minOccurs="0" />
    </sequence>
  </complexType>
– <simpleType name="tsp:StatusType">
  – <restriction base="string">
      <enumeration value="granted" />
      <enumeration value="granted with modifications" />
      <enumeration value="rejection" />
      <enumeration value="waiting" />
      <enumeration value="revocation warning" />
      <enumeration value="revocation notification" />
    </restriction>
  </simpleType>
– <simpleType name="tsp:FailInfoType">
  – <restriction base="string">
      <enumeration value="bad algorithm" />
      <enumeration value="bad request" />
      <enumeration value="bad data format" />
      <enumeration value="time not available" />
      <enumeration value="unaccepted policy" />
      <enumeration value="additional information not available" />
      <enumeration value="system failure" />
    </restriction>
  </simpleType>
```

Figure 10: *Time stamp response definition*

When the status is either *"granted"* or *"granted with modifications"*, a time stamp token must be present. When status is neither *"granted"* nor *"granted with modifications"*, the time stamp token must be omitted. The `StatusString` field is merely defined as containing a string which can provide any additional information (such as *"The MsgImprint field is not correctly formatted"*). A `TimeStampDoc` is a `Signature` as defined in [DSIG02]. More precisely, the signature (figure 11) envelops time stamp information (figure 12) and information relative to the signature itself (algorithm, value, certificates...).

Conforming time stamping requesters must be able to recognize version 1 time stamp tokens with all the optional fields present, but are not mandated to understand the semantics of any expression, if present. The `MsgImprint` field must contain the same value as the similar field in the request.

Figure 11: *An enveloping signature template of the time stamp information*

The `Policy` field must indicate the TSA's policy under which the response was produced. If a `ReqPolicy` was specified in the request, the policy used by the TSA must be the one specified in the request otherwise, the error *"unaccepted policy"* must be returned.

The `SerialNumber` field is an integer assigned by the TSA for each time stamp token. It must be unique for each time stamp issued by a given TSA. One should notice that the property must be preserved even after a possible interruption of the service.

The restriction of the `dateTime` type ensures that we will always be able to determine the ordering of two time stamps according to the `Time` field (which is not the case if the timezone is omitted), provided that the difference of the values of the `Time` field is greater than the `Accuracy` described hereafter.

The `Accuracy` added/substracted to/from the `Time` value provides an upper/lower bound for the actual time of the time stamp. When the accuracy optional field is omitted, the accuracy may be available through other means, e.g. the `TSAPolicy` field.

If the ordering field is missing or is present and set to false, the `Time` field only indicates the time at which the time stamp token has been created by the TSA. In this case, the lack of accuracy of two time stamps can prevent from determining which one was produced first. If the ordering field is set to true, every time stamp token from the same TSA can always be ordered based on the `Time` field, regardless of the `Accuracy`.

The `Nonce` field must be present if it was present in the request. In such a case, it must equal the value of the request.

A client receiving a time stamp response (figure 10) should of course first check the status of the response. If everything is okay, he can store the `TimeStampDoc` as a detached time stamp, or within his document as an envelopped time stamp (note an envelopping time stamp is impossible as the TSA does not have access to the document, but only to its hash).

5 Extensions

The main issue with the elementary request/response exchange is finding a way to limit the trust put in the TSA : there is no way to detect a fake time stamp issued by the Authority. Several ways to increase reliability on these timestamps were explored by Haber and Stornetta in [HS91] and further publications. They combine two basic ideas to limit the trust put in a TSA : sending time stamping requests to several distinct entities and linking the

```
– <complexType name="tsp:TimeStampInfoType">
  – <sequence>
      <element name="TSAPolicy" type="tsp:TSAPolicyType" />
      <element name="MsgImprint" type="ds:DigestValue" />
      <element name="Time" type="tsp:TSPTimeType" />
      <element name="Nonce" type="long" minOccurs="0" />
      <element name="Accuracy" type="AccuracyType" minOccurs="0" />
      <element name="X509Qualifiers" type="ds:X509Data" />
    </sequence>
    <attribute name="Version" type="positiveInteger" />
    <attribute name="SerialNumber" type="integer" />
    <attribute name="Ordering" type="boolean" default="false" />
  </complexType>
– <simpleType name="TSPTimeType">
  – <restriction base="dateTime">
      <pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:d{2}.\d{3}Z" />
      <pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:d{2}.\d{3}+\d
        {2}:\d{2}" />
      <pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:d{2}.\d{3}-\d
        {2}:\d{2}" />
    </restriction>
  </simpleType>
– <complexType name="AccuracyType">
  – <sequence>
    – <element name="Seconds" type="SecondType">
      – <simpleType name="SecondType">
        – <restriction base="nonNegativeInteger">
            <maxInclusive value="59" />
          </restriction>
        </simpleType>
      </element>
    – <element name="Millis" type="MilliType">
      – <simpleType name="MilliType">
        – <restriction base="nonNegativeInteger">
            <maxInclusive value="999" />
          </restriction>
        </simpleType>
      </element>
    </sequence>
  </complexType>
```

Figure 12: *Time stamp info type definition*

```
– <complexType name="DistributedTimeStampDocType">
  – <sequence>
      <element ref="TimeStampDoc" maxOccurs="unbounded" />
    </sequence>
  </complexType>
```

Figure 13: *Distributed Time stamp template*

time stamps emitted by a TSA to the previously emitted time stamps. In the following section, we have a rapid look at the adaptations we need to define for our protocol to support these schemes.

5.1 Distributed model

Using time stamps in a distributed scheme involves only minor effort for integration. In this scheme trust in the TSA is limited by the fact that the request is sent to several independent TSAs. The responses are then checked for coherence and the resulting time stamp consists in gathering the responses from the different TSAs together in a unique structure. Thus we can define the time stamp structure which will be saved by the client with the structure defined in Figure 13.

5.2 Linked model

In a linked model, the information signed by a TSA involves data from previously emitted time stamps. The response has to take this change in account and the format of the time stamp element must be redefined by adding a reference to some link information depending on the linking scheme used. For example in the simplest case, the link information is the hash of the previous time stamp which gives the schema of Figure 14.

```
- <complexType name="tsp:TimeStampInfoType">
  - <sequence>
      <element name="TSAPolicy" type="tsp:TSAPolicyType" />
      <element name="MsgImprint" type="ds:DigestValue" />
      <element name="Time" type="tsp:TSPTimeType" />
      <element name="Nonce" type="long" minOccurs="0" />
      <element name="Accuracy" type="AccuracyType" minOccurs="0" />
      <element name="X509Qualifiers" type="ds:X509Data" />
      <element name="LinkInfo" type="ds:Signature" minOccurs="0" />
      <!--  this Signature is actually the previously emitted time-stamp
        -->
    </sequence>
    <attribute name="Version" type="positiveInteger" />
    <attribute name="SerialNumber" type="integer" />
    <attribute name="Ordering" type="boolean" default="false" />
  </complexType>
```

Figure 14: *Modifications to TimeStampInfoType for a Linked Model*

There has been, besides these two examples, many schemes designed to enhance the elementary Request/Response exchange in order to limit the trust put in the TSA. But as they have most often evolved from the two examples we gave, we believe that it will be easy to adapt the original protocol to these elaborate schemes, with only a few changes in the structures defined above.

6 Conclusion

Designing a time stamp protocol benefits from XML's most evident advantages : readability and extensibility. Readability is necessary for a protocol designed to be widely used; it is achieved thaks to an output often consisting of explicit names instead of numbers or codes. Extensibility is even more critical as protocols evolve permanently. In this perspective, it is important that Time Stamp Protocol benefits from *XML Signatures*' evolutions as security is at stake.

Moreover, we believe efficiency of the protocol is not sacrificed. In the 80's, ASN.1's compactness was an important featuer, but nowadays, the difference of compactness between ASN.1 and XMLnotations only introduces minimal additional processing. Actually, the efficiency of the overall process mainly depends on the TSA's ability to perform quickly computations like hashing and signing.

7 Acknowledgements

We'd like to thank Jim Hughes for both his ideas concerning this work and his support.

References

[CMS99] R. Housley, *Cryptographic Message Syntax*, Network Working Group, RFC 2630, June 1999.

[DER97] ITU-T Recommendation X.690, *Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, OSI

networking and system aspects - Abstract Syntax Notation One (ASN.1), Series X: Data networks and open system communications, December 1997.

[DSIG02] D. Eastlake, J. Reagle, D. Solo, *(Extensible Markup Language) XML-Signature Syntax and Processing*, Network Working Group, RFC 3275, March 2002.

[HS91] S. Haber, W. S. Stornetta, *How to time stamp a digital document*, Journal of Cryptology, Vol.3, No. 2, pp. 99-111. 1991

[IM00] T. Imamura and H. Maruyama, *Mapping between ASN.1 and XML*, IBM Research Report, RT 0362, 2000.

[TSP01] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, *Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP)*, Network Working Group, RFC 3161, August 2001.

[XADES02] ETSI Technical Committee Security (SEC), *XML Advanced Electronic Signatures (XAdES)*, Technical Specification 101 903 v1.1.1, February 2002.

[XER01] ITU-T Recommendation X.693, *Information technology - ASN.1 encoding rules: XML encoding rules (XER)*, OSI networking and system aspects - Abstract Syntax Notation One (ASN.1), Series X: Data networks and open system communications, Prepublished version at http://www.itu.int/ITU-T/studygroups/com17/languages/X.693_0901.pdf, December 2001.