Digital Signing without the Headaches

Nick Pope [1] · Juan Carlos Cruellas[2]

[1]Security & Standards Associates
Grays, Essex, United Kingdom
nickpope@secstan.com

[2]Universitat Politècnica de Catalunya
Barcelona, Spain
cruellas@ac.upc.edu

Abstract

Deploying support for digital signatures can be a major headache for any organisation. In many cases signatures are created on behalf of an organisation but may be applied by a constantly changing authorised group of personnel. The need to manage the allocation and certification of the multitude of user keys can be particularly burdensome and difficult to secure. This paper presents an alternative approach to the digital signing, which significantly reduces these headaches, being supported by a number of companies and standardised by OASIS. The OASIS "Digital Signature Services" (DSS) standard specifies the use of a specialised server for the creation and verification of signatures under control of remote clients. Instead of keys having to be held and managed individually, OASIS DSS enables keys and other aspects of the signing service to be managed centrally on a networked server. The OASIS DSS protocol supports a range of signature formats including XML and CMS. It is designed around a basic "Core" set of elements and procedures which can be profiled to support specific uses such as time-stamping (including XML structured timestamps), corporate entity seals, electronic post marks and code signing.

1 Why OASIS DSS?

Electronic documents play a key role in today's modern business environment. No longer does paper form the basis for the day-to-day business. Information is prepared and stored electronically and exchanged on line through email and other online services. E-commerce and the electronic office is no longer just a buzz word but a reality. However, such key business information is generally stored and exchanged in unprotected form. Electronic documents can be readily changed and it can be difficult to prove its authenticity. Such information can be open to fraud and in modern regulatory environment where electronic documentation provides essential part of audit records and regulatory reporting. Organisations are leaving themselves vulnerable to attack and can also have difficulties in providing verifiable documentary evidence against claims of malpractice. The keeping of paper records for key business information is becoming less and less of a practical proposition.

A solution to ensuring authenticity of electronic documents has been available for many years – digital signatures. This enables the source of documentation to be quickly verified as coming from an authentic source and any document tampering is made immediately obvious. Furthermore, being based around public key techniques, digital signatures can be used at the

global level. Given the appropriate public key infrastructure any party can readily verify a signed document's authenticity.

However, the widespread use of digital signatures has yet to be realised. Similar techniques have been generally accepted as the solution of securing web site using SSL. But, whilst use of digital signatures is gradually spreading, digital signatures have yet to obtain widespread adoption. The use of digital signatures is often seen as a major headache. The management of the keys necessary to produce digital signatures can be burdensome and often needs the use of special smart card devices to ensure the security of the keys. This can be particularly difficult in large organisations where there are large numbers of individuals who regularly need new keys because of change in roles as well as people joining and leaving the organisation. Furthermore, individuals often misplace or misuse the keys compromising their security adding the further burden of a major infrastructure for handling revocation.

The OASIS Digital Signature Services (DSS) standard provides a way of significantly reducing the headache of using digital signatures by controlling the application of signatures on an organisational basis through a network based server. Instead of each individual requiring to protect a document having to be allocate a key, with all the difficulties of managing and securing it, using DSS the signing keys are managed on a secure server with all the security controls necessary to minimise the risk of compromise. The creation of a DSS server based signature can be still under the control of an authorised individual but instead of needing specialised signing equipment for each user, the existing user authentication mechanisms (password, two factor, biometric …whatever already is accepted by the organisation) can be used. A DSS signature can secure the organisation's documents, efficiently and effectively, whilst maintaining accountability down to the individual level. Furthermore, the security necessary to protect sensitive signing keys can be targeted at the signing server, for example through the use of tamperproof signing devices and placement in a secure room with controlled access, perhaps with dual control, thereby maximising security and yet reducing costs because the security can be highly localised. This is further enhanced by the ability to manage the auditing of signing events centrally.

The development of the OASIS DSS has involved the leaders in the digital security market including RSA, IBM, BEA Systems, Entrust, Surety, Cybertrust. Also, OASIS DSS has worked closely with the Universal Postal Union to facilitate the use of DSS within their Electronic Post Mark system [UPU EPM]. Several implementations exist and interoperability trials are being carried out to demonstrate the practicability of the standard. Much of the work of DSS has been aimed at building on the simplicity of time-stamping services to provide the full capabilities of digital signatures and support a range of signature forms.

This technique of network based signing is particularly appropriate where information is released on behalf of an organisation, for example with signing of code to indicate that the a program is created by a trusted corporation with the appropriate development and release organisational controls. By placing the organisational signature in shared server, the creation of the signature can be linked to the appropriate controls for proper authorisation and release of signed code. Similarly, DSS signatures can be used in electronic invoicing to sign electronic invoices where the appropriate release procedures have been met.

The use of DSS is not limited, however, to the signing of documents with corporate signatures. It can be used to provide general facilities to protect the integrity of key documents within any organisation by sealing or time-stamping data to ensure that it cannot be modified once protected. The approach of network based signing is also being recognised as form of proxy signing in legal environments such as court filing and notarial services.

2 What Does OASIS DSS Do?

The basic aim of the OASIS Digital Signature Service (DSS) draft standard is to define protocols for a networked web service to support digital signatures. It also supports a variety of variations on basic digital signature services such as time-stamping.

DSS is designed to support a range of signature formats. Not only does DSS support the World Wide Web consortium XML Signature [W3C XMLDSig], but also the widely used Cryptographic Message Syntax (CMS) binary signed data format [IETF CMS]. It can even be extended to support other forms of signature such as PGP. The protocol is also designed to be easily extensible to enable support of advanced forms of CMS and XML based electronic signatures such as defined by ETSI [ETSI TS 101 733] & [ETSI TS 101 903].

DSS supports two basic protocols one for the creation of digital signatures, the other for verification of signatures. The basic operation of a DSS sign and verify requests are illustrated below:

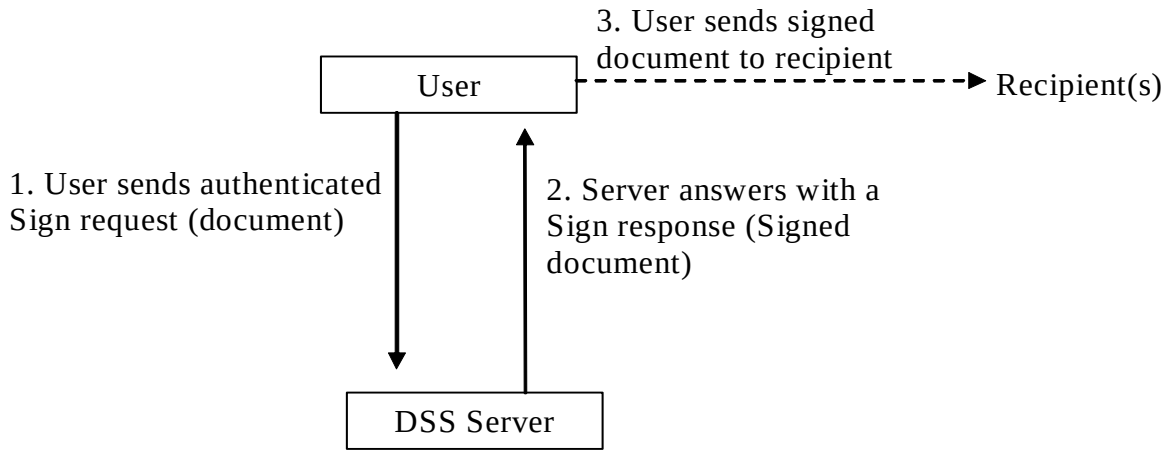

Fig 1: Illustration of DSS Sign Protocol (to be updated with prettified version)

1. The user sends the request for the document to be signed through a secure channel that authenticates the user (e.g. SSL with client authentication).
2. The server checks that the authenticated user is allowed to sign the document and if acceptable signs the document on behalf of the user with a corporate signing key or a key which the server holds on behalf of the user.
3. The signature is added to the document by the server and returned to the user back through the same secure channel.

Having obtained the signed document from the DSS server the user can then pass it on to one or more recipients who may verify the signature themselves or use the DSS verify protocol

The recipient may verify the signature himself or use the DSS verify protocol as indicated below:

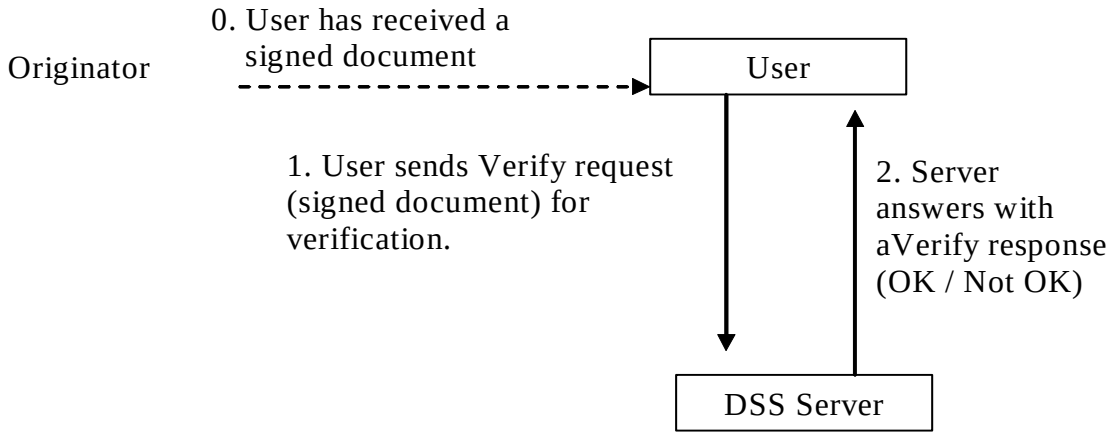

Fig 2**: Illustration of DSS Verify Protocol**

The user may be passed a signed document coming from a user with its own signing capability or one using the DSS Sign protocol as described above.

1. The user sends the request for the signed document to be verified through a secure channel (e.g. SSL).
2. The server verifies the validity of the signed document including checking the validity and revocation checks on any keys or certificates as necessary.
3. The results of this verification is returned back to the user through the same secure channel.

The DSS protocol removes from the user all the burdens normally associated with digital signatures. There is no need for the management of large numbers of keys distributed throughout the organisation, and no special cryptographic code or keys are needed on the client system. Where it is necessary to authenticate the client existing mechanisms can be used. All the problems of maintaining the security of the keys and cryptographic functions associated with digital signatures can be managed by the organisation through centralised controls.

DSS servers can be used to maintain an audit record to confirm that signatures are verifiable at the time of receipt, and through use of time-stamping ensure that the validity of archived signed documents can be assured long after the applicable keys have expired.

3 DSS specification set structure

The DSS specification set is formed by the so-called core document ("Digital Signature Service Core Protocols, Elements and Bindings") and a number of additional documents defining specific profiles of the aforementioned core protocols.

The core document defines the (XML-based) syntax and semantics for the basic services, namely: signature generation and signature verification. This includes:

- Definition of four basic messages: SignRequest, SignResponse, VerifyRequest and VerifyResponse. They are defined to easily manage the most common signatures formats, ie, [XMLSig] and [CMS].
- Definition of an extensibility mechanism that allows the clients to further qualify or even increase the extent of the requests through optional inputs. It also allows the servers to answer with extended responses through the corresponding optional outputs.

- Definition of a XML format for a time-stamp token, fully based on XML signatures as specified in [XMLSig].
- Definition of mechanisms for managing generation and verification of digital signatures carrying time-stamp tokens (both CMS-based as defined in [RFC 3161] and the XML-based specified in the core document itself) computed on the signatures themselves (signature time-stamps).
- Definition of bindings for transport and security. The first ones specify how DSS messages are encoded and carried over the most popular transport protocols (it defines bindings for HTTP –through HTTP POST exchanges- and SOAP 1.2). The security bindings establish rules for providing confidentiality, authentication and integrity to the transport binding; TLS 1.0 support is mandatory and SSL 3.0 support is optional. In this way clients may use wide-spread tools that do not jeopardize their implementation.

The profile documents further develop the basic messages so that they may be easily tailored to meet the requirements of a specific application or use case. Profiles may restrict the values ranges of certain message elements, or, if required, extend the basic core protocols defining new optional inputs, outputs and/or bindings.

The final result is not only a set of protocols targeting a number of relevant scenarios but also a set of generic protocols which may be easily further profiled as new uncovered use cases are identified.

4 Variations and Profiling DSS

The DSS protocol supports a number of variations in this protocol. For example, the signature may be passed back to the user on its own, detached from the document to which it applies, or placed within the document to which it applies. Another variation is that the document is reduced to a simple hash fingerprint for sending to the server instead of the document for either signing or verification, thereby reducing bandwidth requirements and reducing the opportunity for the confidentiality of the document to be compromised.

When signing a document the DSS server may add additional attributes or properties to the signature such as the claimed signing time or a time-stamp against the content applied immediately before signing.

Due to the number of variations a specific set of options can be selected in the DSS protocol to support a particular mode of operation or application requirement. This selection from the DSS protocol is defined in separate DSS profile specification. The DSS protocol is also designed to facilitate extensions and so DSS Profiles may also extend the protocol, as well as selecting specific options, defining its own profile specific input or outputs for profile specific attributes of a signature.

One of the simplest DSS profiles provides time-stamping equivalent to the existing time-stamping standard – RFC 3161, but incorporating the use of XML. In the DSS time-stamping profile selects from the core to provide simple time-stamping as follows:

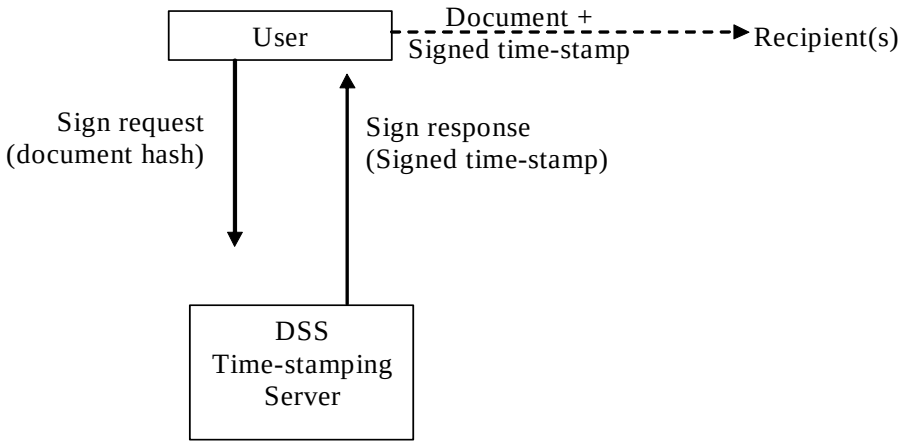

Fig 2: Illustration of DSS Time-stamp Profile

1. The user calculates the hash of the document to be time-stamped locally
2. This document hash is sent to the DSS time-stamping server.
3. The server creates a signed object containing the document hash and the signing time.
4. The signed time-stamp is returned to the user, and if the DSS server is not accessed through a secure channel the user may verify the signature as being valid and from a trusted server.

The user would pass the document and signed time-stamp on to one or more recipients. If required a profile of the DSS verify protocol may be used to verify the time-stamp at the recipient.

A number of profiles have been defined for DSS. This includes:

a) Time-stamp profile

As described above, including support for XML format time-stamps.

b) DSS Entity Seal Profile

This profile is a variation on a signed time-stamp, where the signed object includes not only the time but the identity of the authenticated user requesting the "entity seal". This provides further traceability and provides a form of "proxy" signature where the signature is produced on behalf of another identifiable party.

c) Advanced Electronic Signature Profile

This profile produces signatures that have the attributes needed for legally qualified and long-term signatures

d) Code signing Profile

This profile is designed to support the signing of code authorised for distribution with an organisational signature indicating its authenticity.

e) Electronic Post Mark Profile

This profile is for providing an electronic post mark used confirm authenticity of email, as promoted by the Universal Postal Union (UPU).

f) Signature Gateway Profile

This profile supports the creation of signatures at a gateway from a form only recognised internally to a standard form which can be recognised externally.

5 Technical Details

The DSS core document defines two protocols: one for signature generation (Sign) and one for signature verification (Verify). Each protocol defines two XML messages: one for requesting the provision of the service to the server (Request), and other for giving the result back to the client (Response).

5.1 Sign protocol

The `SignRequest` message has two different parts:

1. `InputDocuments`. This element contains information on the documents that must be signed. Binary documents are encoded in base-64 within `Base64Data`. XML documents may be escaped (`EscapedXML`), base-64 encoded (`Base64XML`) or without any previous processing (`InlineXML`). `InputDocuments` may also contain the digest of the documents (`DocumentHash`) or even a transformed version of the original document (`TransformedData`).
2. `OptionalInputs`. The core document defines some contents that may be useful to any profile. The core defines inputs for indicating the **identity** of the requester; for indicating the signing **key to be used** by the server; for requesting to the server the generation and incorporation **of a time-stamp token on the signature**; for requesting generation of multiple `ds:Reference` elements for a single `Document`, etc.

The core document specifies how the server must behave when receiving a `SignRequest` for generating the requested signature and building the corresponding `SignResponse`.

The `SignResponse` message has three relevant parts:

1. `Result`, with details of the result of the server's operation: a mandatory `ResultMajor`, notifying whether the server executed properly, an optional `ResultMinor` giving specific details, and an optional string (`ResultMessage`).

2. `SignatureObject`, which may enclose the signature created to be passed to the client. This may be a XML (`ds:Signature`) or a base-64 encoded CMS signature (`Base64Signature`). It may also contain a RFC 3161time-stamp or a XML based time-stamp as defined by the DSS core itself. Finaly, it may also contain a reference(`SignaturePtr`) to an enveloped XML signature within a document.

3. `OptionalOutputs`. An example is the one for including documents enveloping the signature requested.

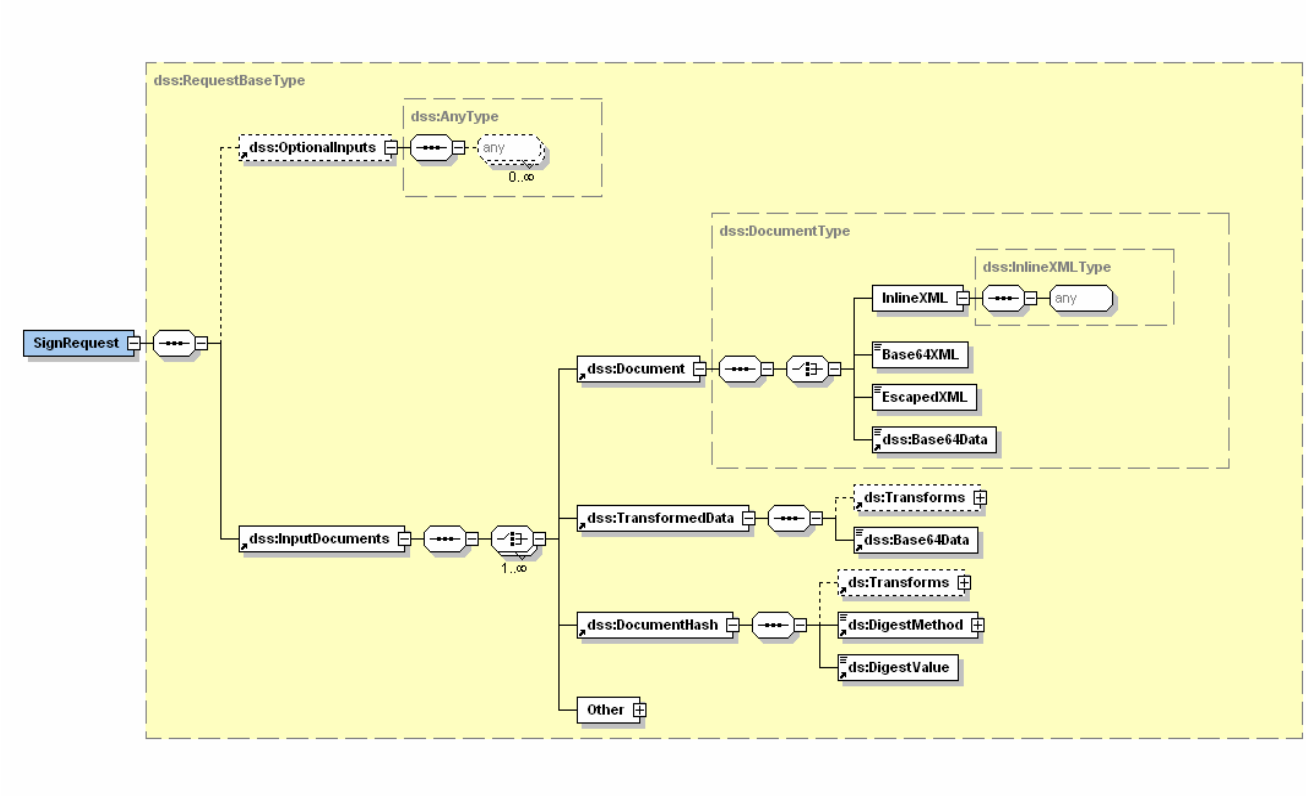

Fig 4: SignRequest message structure

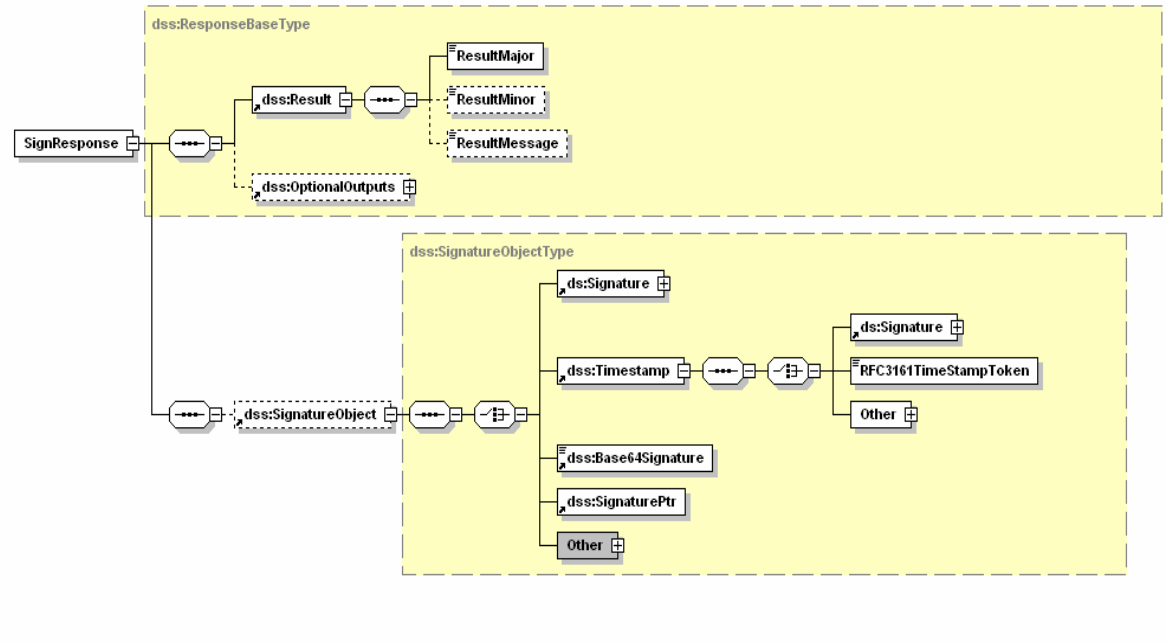

Fig 5: SignResponse message structure

5.2 Verify protocol

The `VerifyRequest` message has three main parts:

1. `SignatureObject`. This element is used for passing the signature to be verified to the server. Its structure is the same as the one present in `SignResponse` message.
2. `InputDocuments`. This element is used for passing the signed documents that the server must deal with while verifying the signature. They may even incorporate enveloped signatures.
3. `OptionalInputs`. The core defines inputs for instructing the server to verify any ds:Manifest element present in [XMLSig] signatures; for requesting to the return, of details on the verification process; for requesting the return of the signatory's identification, for requesting the incorporation of unauthenticated properties (cryptographic material used for verification, for instance) to the signature; and for requesting the generation and incorporation of a signature time-stamp as an unauthenticated property, among other purposes.

The `VerifyResponse` message shares its two elements with `SignResponse`:

1. `Result` gives details of the result of the server's operation.
2. `OptionalOutputs`. Generaly speaking, each optional input of VerifyRequest is related with thecorresponding optional output where the server passes to the client the result of the specific processing requested.

5.3 XML Time-stamp token

DSS core protocol also defines a format for XML time-stamps. A XML time-stamp is an XML Signature (as defined in [XMLSig]) that signs an ds:Object element enclosing the current time and related information in a TSTInfo element.

Contents of this TSTInfo element are equivalent to the fields defined in the binary timestamp structure defined in RFC3161. `CreationTime` contains the time when the token was issued; `SerialNumber` contains a unique serial number across all the tokens generated by a particular TSA. `ErrorBound` indicates the TSA's estimation of the maximum error in its local clock. `Policy` identifies the policy under which the token has been issued. `Ordered` indicates whether the time-stamps generated by a TSA are ordered according to the value of `CreationTime`. TSA contains TSA's name.

6 Conclusion

At the time of writing the DSS set of standards are currently in Committee Draft status and just about ready to be submitted for public comments. In parallel, certain members of the TC have started an interoperability initiative for assessing the protocols under a practical perspective, and suppliers are already working on bringing implementations to market. By implementing DSS, the power of digital signatures can be provided without the headaches of installing PKI capabilities at every user system and ensuring signing keys and devices are managed securely.

References

[OASIS DSS] OASIS Digital Signature Services Technical Committee
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss

[W3C XMLDSig] XML-Signature Syntax and Processing – W3C Recommendation 12 february 2002, D Eastlake et al, http://www.w3.org/TR/xmldsig-core/

[IETF CMS] IETF RFC 3852 Cryptographic Message Syntax (CMS), July 2004 R. Housley, http://www.ietf.org/rfc/rfc3852.txt

[ETSI TS 101 733] CMS Advanced Electronic Signatures (CAdES) http://www.etsi.org/services_products/freestandard/home.htm

[ETSI TS 101 903] XML Advanced Electronic Signatures (XAdES) http://www.etsi.org/services_products/freestandard/home.htm

[UPU EPM] Universal Postal Union – Electronic Post Mark http://www.upu.int/news_centre/documents/en/brochure_the_electronic_post_mark_en.pdf

Keywords

Electronic signatures

Digital signatures

XML

Public key cryptography

Web services

Timestamping

Electronic Postmark