Multiple electronic signatures on multiple documents

Antonio Lioy and Gianluca Ramunno

Politecnico di Torino
Dip. di Automatica e Informatica
Torino (Italy)

Abstract. European, international and Internet standards are available to support electronic signatures. The most common signature formats are defined via the ASN.1 syntax with DER encoding, or the XML language. Furthermore PDF is a widespread document format with support for e-signatures. Application of signatures to e-documents must consider several aspects: long term signature validity, non-repudiation, qualified certificates, and many others. This paper focuses on the relationships among multiple documents and multiple signatures and analyses the support provided by current formats to this problem. Where lack of standardization or standard profiling is found, a proposal is made towards better application of e-signatures.

1 Introduction

It's a common belief that the field of electronic signatures (in short *e-signatures*) is an already established one. While this is true for the basic technical and legal foundations, we believe that many aspects have still to be investigated and properly defined. In particular, the flexibility and richness of signature and documents in the paper world are still unmatched, and this can cause unexpected problems in real-life applications. In this paper we focus on the problems related to multiple signatures on the same document (as in the case of countersignatures), signature of document parts or of multiple documents. The complex relationships between multiple signatures and documents are explored, with reference to the major formats, to identify problems and suggest solutions that can improve the practical applicability of electronic signatures.

ISO defines [1] digital signature as *"Data appended to, or a cryptographic transformation of, a data unit that allows the recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient."* This definition is neutral from the technology point of view – for example, it can be implemented with asymmetric encryption or with a Message Authentication Code – and is also neutral with respect to the signer, that can be a machine or a human being. The e-signature definition in the European Directive [2] is also technologically neutral but is related to a specific type of signer: a (natural or legal) person. The same Directive defines an "advanced electronic signature" as a subset of the e-signatures and is equivalent to the ISO definition of digital signature. We can therefore state that a digital signature is an e-signature, when related to a person.

Lioy A. and Ramunno G. (2004).
Multiple electronic signatures on multiple documents.
In *Proceedings of the 1st International Workshop on Electronic Government and Commerce: Design, Modeling, Analysis and Security*, pages 24-34
DOI: 10.5220/0001405300240034

Since the term “electronic signature” is more closely related to the transposition of the handwritten signatures in the electronic world, we prefer this term over “digital signature”. However when we’ll need to refer to specific aspects of a digital signature, this term will be used. If not differently specified, in this paper a digital signature is always implemented via asymmetric encryption.

2 E-signatures and e-documents in real contexts

The simplest use of an e-signature is its application to a single document. However, the real use cases in the paper world are more complex. To gain widespread acceptance, e-signatures should support applications as complex as those supported by handwritten signatures. We will analyse this problem by looking at two of its more complex facets: multiple signatures and multiple documents.

2.1 Relationships among signatures

Several different signatures can be applied at the same time to a single document. Depending on the fact that the order of the signatures is important or not, two categories can be defined: *independent* (i.e. parallel) signatures and *countersignatures* (wrapping and overall signatures).

Independent signatures on a single document do not depend on each other. They are also called “parallel” as they can be computed simultaneously. From a functional perspective, these signatures can be considered as created at the same time even if in practice they aren’t; anyway, it doesn’t matter which signature is created first. In terms of digital signatures, independent signatures are generated by computing a signature algorithm with different keys on the same digest computed over the document being signed (Fig. 1-a).

Countersignatures must be considered in pairs: the embedded signature (the first one in time) covers the document, while the wrapping signature (the second one) covers both the document and the first signature. Many countersignatures can be applied to the same document, each one wrapping the previous one that becomes embedded. In terms of digital signatures, two different forms of countersignatures exist: in the first one, the digest of the wrapping signature is computed only over the embedded signature (Fig. 1-b), while in the other the digest of the wrapping signature is computed over both the embedded signature and the document (Fig. 1-c). To distinguish between the two forms of countersignatures and according to [3], for the rest of this paper we will talk about *independent signatures*, *wrapping/embedded signatures* and *overall signatures*.

An example of wrapping/embedded signature is the signature carried by the counterSignature attribute defined by CMS [4] and supported by ETSI [5]. Examples of overall signatures are those implemented by nesting several S/MIME objects [6] and the time-stamps (repeatedly) applied to a signature for long term validation as in the ETSI ES-A format.

The distinction between wrapping and overall signatures makes sense only in the electronic world and only when implemented as digital signatures. In the real world, a handwritten counter-signature on a paper document is implicitly applied both to the

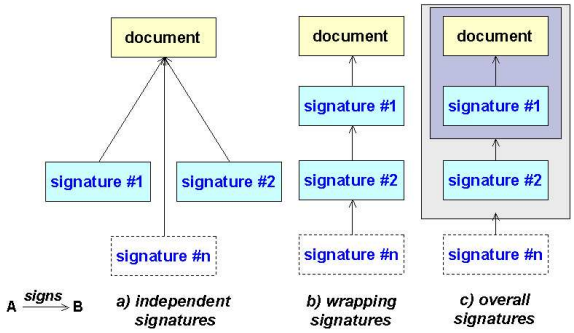

Fig. 1. Independent, wrapping and overall signatures

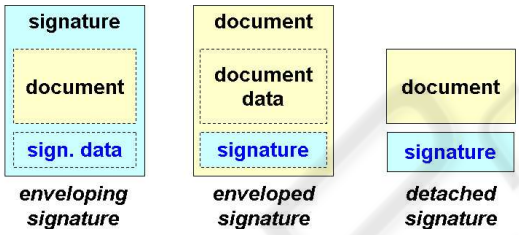

Fig. 2. Signatures-documents relationships

existent signature(s) and to the document. Moreover, from a semantic point of view, the wrapping and overall signatures perform the same function as both sign the document with the previous signature(s): the wrapping signature does it indirectly (via a signature chain) while the overall signature does it directly (by an encapsulation procedure).

The three basic relationships among electronic signatures can be used to build any complex set of relationships among any number of signatures.

2.2 Signature-document relationships

By looking at the relationship between the signature and the document being signed, it is possible to classify signatures as *enveloped*, *enveloping* and *detached* (Fig. 2). An enveloped signature is embedded within the document. An enveloping signature contains the document. In both cases, the signature and the document are stored together in the same "envelope", while a detached signature is stored separately from the document being signed.

2.3 Multi-part documents / dossiers

Another relationship is the one among documents or parts of them. Documents related to a single topic or file can be grouped in a unique dossier (Fig. 3-a). Moreover, depending on the application, some documents may require to have many signatures, each one applied to a single part of the document (Fig. 3-b).

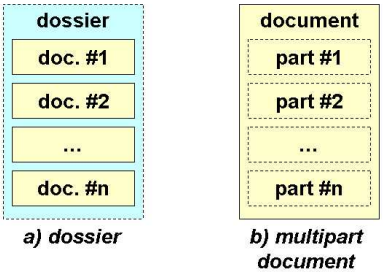

Fig. 3. Dossiers and multi-part documents

An electronic dossier can be implemented in different ways. One of them is by creating a multi-part MIME [7] object, where each document is a MIME part. Another way is by linking together the dossier documents via an indirect digital signature. To calculate this signature the document digests must be concatenated and a digital signature computed over them. This cryptographically binds the documents together even if they are physically stored separately.

The implementation of a multi-part document to allow signing each part individually, is strictly related to the document and signature formats. The document format must support a method to identify the boundaries of each part, while the signature format must support a mechanism to clearly store these data.

3 Current formats for electronic signatures

By combining the basic relationships (signatures-signatures, signatures-documents and documents-documents) it is possible to create in the electronic world the variety of real cases found in the paper world. However, the current e-signature formats give a limited support to these schemes. In general they work well with one document and one signature but hit limits when creating complex schemes. In the next sections we will analyse some widespread e-signature formats from the perspective of the relationships among e-signatures and e-documents.

3.1 Cryptographic Message Syntax (CMS)

CMS is a multi-purpose format for cryptographic services: it supports digital signatures, Message Authentication Codes, encryption, etc. Here we will consider only the signed-data content type. CMS is specified by ASN.1 [8] with data encoded via DER [9]. The SignedData content type supports many signers (signerInfos) of the same content (encapContentInfo). This basic structure (Fig. 4-a) makes easy creating multiple independent signatures. Among the data included in each signer's structure (SignerInfo), there are signed (signedAttrs) and unsigned (unsignedAttrs) attributes. The former are covered by the digest in the signature and are used to cryptographically bind some data to the content. Therefore the signed attributes can't be modified later without invalidating the signature. On another hand, the unsigned attributes are used to transport data

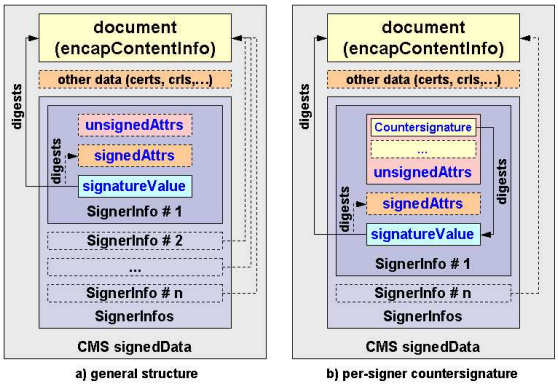

Fig. 4. CMS signed-data

that don't need to or cannot be signed (e.g. a counter-signature that can be generated only after the generation of the first signature). If the signedAttrs structure is present, at least two signed attributes must be present: content-type and message-digest. The former carries the type of data being signed, while the latter contains the digest computed only over the document.

Signatures relationships. The signed-data format natively supports several independent signatures. Furthermore CMS defines the Countersignature unsigned attribute that contains a countersignature (Fig. 4.b). Since attributes are per-signer, the countersignatures are per-signer in either form, wrapping or overall. Therefore a signature that countersigns two independent signatures is not supported by CMS but by nesting several signed-data envelopes. In this case, the inner object contains the independent signatures and becomes the "document" being signed of the outer CMS object. The latter will include only one signature that represents the countersignature in the overall form. Nesting CMS objects is the S/MIME approach for digital signature and message encryption.

Signatures-documents relationships. CMS supports only enveloping and detached signatures: the former when the structure encapContentInfo contains the document being signed, the latter when this structure is empty.

Multi-part documents and dossiers relationships. The document can be in whatever format, as it is considered a single binary object (a "blob"). Even if the possibility of signing only a specific part of a document depends on the document format, CMS doesn't provide any support to include the reference to a document fragment, in order to signal that only a fragment has been signed. Furthermore CMS doesn't provide native support for signing dossiers.

3.2 XML Signature (XMLdsig)

XML Signature [10] is the format designed to support digital signatures encoded via XML. No other security service is supported. This format is an XML application, defined by using the Document Type Definition (DTD) syntax and the XML Schema language [11].

XMLdsig has been designed to support any document format: a document in a generic format is considered as a single blob while special procedures are used when signing XML documents. In fact several forms of an XML document with the same semantics may exist. These forms have a different binary representation but the same logical representation, such as the DOM [12] node tree generated by a parser. Different binary representations lead to different digests and consequently to different signature values. Therefore, before being signed, an XML document must be transformed into a canonical representation by using a proper algorithm, such as C14n [13]. Moreover a generic transformation - such as an XSLT [14] one - could have been applied over the document before being signed. XMLdsig takes into account these issues: the canonicalisation algorithms and pre-signature transformations are specified within the signature structure and are signed together with the document.

XMLdsig is an indirect signature: it is implemented by calculating an overall digest over the set of the digests computed on each document being signed. Complex schemes with more than two levels of digest calculation are also supported. This core structure, the intrinsic nature of XML and the extensive use of URIs to refer to the data covered by the digital signature make XMLdsig capable to sign multiple documents, parts of documents, or documents stored elsewhere and accessed through a reference (URI), as well as to flexibly control how a signature has to be validated.

The main elements of XMLdsig are:

- <SignedInfo> references the documents being signed and their digests;
- <SignatureValue> is the signature computed by applying a digital signature algorithm to the digest of the canonicalised <SignedInfo> element;
- <Object> is an optional element that can appear in multiple instances to include any data type; it is generally used to:
  - carry the document being signed in the case of enveloping signatures
  - carry some signature properties or assertions as <SignatureProperties> to be optionally signed together with the documents
  - carry one or many <Manifest> objects
- <Manifest> is an optional element very similar to <SignedInfo> (it can include references to the documents being signed and their digests) but has a different semantics in the signature validation process, described later.

The basic structure of an XML signature is given in Fig. 5 including the digest calculation while the various transformations and the structure carrying the signer's key data are omitted. The validation of an XML signature containing only the mandatory <SignedInfo> must be performed as follows:

1. check that every reference within <SignedInfo> is valid
2. re-calculate the digests over every referenced document and verify that they match with the ones stored within <SignedInfo>

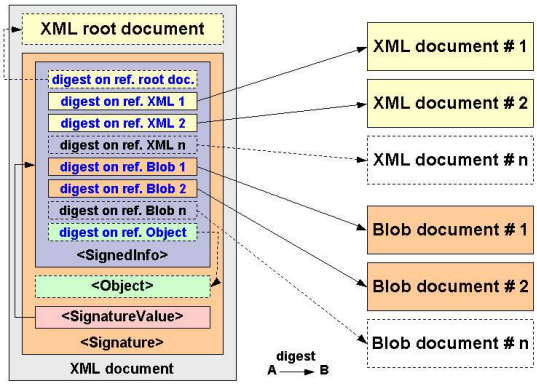

Fig. 5. XML Signature general structure

3. re-calculate the digest over the <SignedInfo> after having applied the transformations specified within the same element
4. use this digest and the signature as input of the proper verification function of the chosen signature algorithm.

A scheme that includes <Object> and <Manifest> is shown in Fig. 6. When the <Manifest> is present within <Object>, the reference to <Manifest> is included within <SignedInfo>. During signature validation, the latter is a reference to be mandatory checked together with the related digest while the verification logic of the references in <Manifest> is left to the application: it can check all the references and digests, only part of them or none.

More complex schemes can be implemented by hierarchically nesting multiple <Manifest> up to the last one that is referenced within <SignedInfo>.

It is also possible to have any number of <Object>, each one including any number of elements of any type. The elements to be signed must have the "id" attribute and the related reference inserted into <SignedInfo> or <Manifest>. The reference is an URI with the fragment identifier equal to the value of the "id" attribute. XMLdsig provides very powerful and flexible mechanisms. However it doesn't natively offer standardized structures to support signed and unsigned attributes as CMS does. Furthermore CMS defines a wide set of attributes that can be used. Therefore we can say that XMLdsig is at a lower functional level than CMS.

Signatures relationships. XMLdsig doesn't provide any native support for independent signatures and countersignatures but, given its flexibility, it allows the implementation of countersignatures in both forms. Wrapping signatures can be implemented by placing an XML signature object within an <Object> element of a signature applied to a number of documents. The value of the inner signature is computed over the <SignatureValue> of the outer signature. Overall signatures can also be easily implemented by nesting the signatures in a manner similar to S/MIME. In both cases the definition of a profile of the XMLdsig specification can help to implement the countersignatures in a standard way.

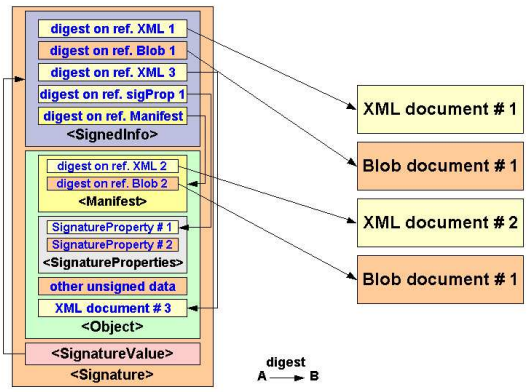

Fig. 6. XML Signature with Object and Manifest

Signatures-documents relationships. XMLdsig supports all the signatures-documents relationships. Enveloping and detached signatures can be applied over any kind of document, while the enveloped signature applies only to XML documents. In fact an XML signature can be placed within an XML document and the <SignatureValue> can be computed over a portion of the document as it can occur when signing document fragments (see next subsection). The signature is enveloped because the root element of the document substantially wraps the signature.

Multi-part documents and dossiers relationships. The core syntax and processing of the XMLdsig specification natively support the signature applied over a number of documents (dossiers). It is also possible to sign parts of documents if the format of the document being signed supports the selection of document portions by means of fragment identifiers compatible with the URI syntax. HTML, XML and XHTML are examples of such document formats: the fragments are identified by the "id" attribute of the tag enclosing the portion being signed, according to the syntax in [15, 16].

3.3 Portable Document Format (PDF)

PDF [17] is a widespread proprietary format for electronic documents whose specification is publicly available. It supports electronic signatures but only the enveloped type, due to its nature of document format. A PDF document is stored in the file as a collection of objects (e.g. the pages) organized in a hierarchical structure. The support for electronic signatures was first introduced in PDF v. 1.3. The signature can be computed over many byte ranges of the PDF file and it is stored in a structure called "signature dictionary". In the latest version of PDF (v. 1.5) a new digest mechanism has been added. Now it is possible to calculate a digest over an object or an object hierarchy, as they are or after being transformed: for example it is possible to apply a selective digest to include certain object types and to exclude other one. In this way it is possible to implement schemes as the one where an author creates a form to be filled and signs it

excluding the empty fields from the signature. Later a user can check the signature to authenticate the form and then fill the form and sign the whole resulting document.

Among the properties signed with the document, there is the registered name of the signature handle, that is the application that generated the signature and that should verify it later on.

The PDF features related to the e-signatures are numerous and very flexible. As for XMLdsig, these basic mechanisms can be considered as powerful building blocks to implement complex signature schemes. However the PDF specification doesn't provide any predefined scheme. Each signature handler, as the signature plug-ins for the Acrobat product, can implement its own set of signature schemes.

Signatures relationships. Since it is possible to sign any physical portion of the PDF file, any number of independent signatures are supported. Moreover, since a signature is contained within an object (the signature dictionary) that can be in turn signed, PDF supports countersignatures too.

Signatures-documents relationships. PDF supports only enveloped signatures.

Multi-part documents and dossiers relationships. PDF signatures are applied only to the content of the file containing the signature itself.

3.4 ETSI e-signature formats

ETSI has defined two specifications for "Electronic Signature Formats" fully equivalent from the functional point of view: one [5] uses the DER encoding and is based upon CMS, while the other [18], called XAdES, uses the XML encoding and it is based upon XMLdsig.

The DER-encoded formats don't offer any additional feature for multiple signatures and documents over the ones already provided by CMS. Instead, in order to be equivalent to the DER-based format, the XAdES specification supports new structures and functions, as the unsigned attributes and specifically the countersignature one, not natively provided by XMLdsig. This is caused by the fact that CMS and XMLdsig are structurally different and XMLdsig is at a lower functional level than CMS.

4 Proposals

4.1 Cryptographic Message Syntax

To give CMS the capabilities to manage signatures on multiple document, we exploit the fact that the data being signed can be a MIME object with multipart/mixed media type. Each document to be signed can be included within a part of the MIME object. By signing this object, a signature over several documents is applied, either as enveloped or detached signature, depending if encapContentInfo includes the MIME object or is empty.

This proposal can be improved to implement features similar to the ones of Manifest in XMLdsig.

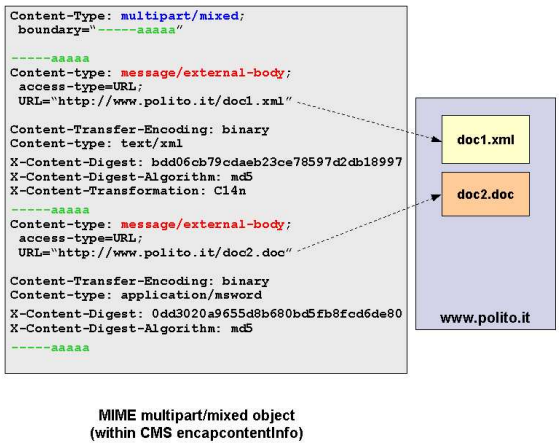

Fig. 7. Proposal for multi-part MIME object

Each part of the MIME multi-part object can be of type message/external-body to reference the documents rather than including them. By using the URL Access Type [19] it is possible to refer to the external documents through the URIs. Moreover, if the format of the documents supports a method compliant with the URI syntax to identify the document fragments (as occurs with HTML, XML and XHTML) it is also possible to sign document fragments.

To complete the proposal, three header fields should be defined to be used within each message/external-body envelope: X-Content-Digest, X-Content-Digest-Algorithm and X-Content-Transformation (optional). In this way it is possible to specify the digest computed over each document being signed, after having optionally transformed it via the X-Content-Transformation. The MIME object should be transformed into a canonical form as described in [20] before being signed. The CMS signature computed over the described MIME object implements an indirect signature over multiple documents. The signature verification would be similar to the one done over the XMLdsig <Manifest>: the verification of the references to the documents and their digests would be left to the applications while the verification of the signature over the MIME object would be mandatory. An example of the complete proposal is shown in Fig. 7.

4.2 XML Signature

XAdES builds many structures upon XMLdsig to support most of the requirements given in section 2. Therefore no proposal is given but it would be useful if XAdES would become a true international specification as complement of XMLdsig. This could be achieved by the future ETSI/W3C joint group.

5 Conclusions

Although many formats for e-signatures are available, this paper shows that, for practical applications, there are aspects to be further specified to implement complex schemes with multiple signatures and documents. Therefore profiles should be produced to define complex schemes in a standardized way. Some proposals (for CMS and XML) have been given in this paper.

References

1. ISO/IEC 13888 (1997). Information technology - Security techniques - Non-repudiation.
2. EU El-Sign Directive (1999). Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures.
3. CEN/ISSS CWA 14171 (2004). CWA 14171 - General Guidelines for Electronic Signature Verification.
4. RFC-3369 (2002). Cryptographic Message Syntax (CMS).
5. ETSI TS 101 733 (2004). Electronic Signature Formats.
6. RFC-2633 (1999). S/MIME Version 3 Message Specification.
7. RFC-2046 (1996). MIME Part Two: Media types.
8. ITU X.680 (2002). Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.
9. ITU X.690 (2002). Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).
10. RFC-3275 (2002). XML-Signature Syntax and Processing.
11. W3C Recommendation XMLSchema (2001). XML Schema.
12. W3C Recommendation DOM (2004). Document Object Model (DOM) Level 3.
13. W3C Recommendation C14n (2001). Canonical XML Version 1.0.
14. W3C Recommendation XSLT (1999). XSL Transformations (XSLT) - Version 1.0.
15. RFC-2854 (2000). The “text/html” Media Type.
16. RFC-3236 (2002). The “application/xhtml+xml” Media Type.
17. Adobe PDF v. 1.5 (2003). PDF Reference, fourth edition - Adobe Portable Document Format.
18. ETSI TS 101 903 (2004). XML Advanced Electronic Signatures (XAdES).
19. RFC-2017 (1996). Definition of the URL MIME External-Body Access-Type.
20. RFC-2049 (1996). MIME Part Five: Conformance criteria and examples.