Donald Eastlake on XML Digital Signatures

An interview with one of the specification's pioneers

Larry Loeb (larryloeb@prodigy.net)
Author, Secure Electronic Transactions
March 2002

In this exclusive developerWorks interview, XML Digital Signatures pioneer Donald Eastlake responds to Larry Loeb's recent article on the topic by clarifying a number of issues about how this technology is used.

Author's note: In a recent article on XML Digital Signatures, I questioned their utility and usefulness. Since the proposal has just been recommended for passage into general usage, I decided it was time to check back on the topic again. This time, I talked with Donald Eastlake, the editor of the XML Digital Signature (XMLDSIG) RFC, and someone who should know something about the subject, since he has been intimately involved with XML specifications since the effort began in the 1990s. He has also served on IETF efforts too numerous to list. His responses appear unedited, except for minor grammatical changes. Questions posed to him appear in bold.

According to the RFC, the goal of a digital signature in XML is to provide authentication services for data 'that may or may not be located within the XML.' What did you envision by this statement?

There were a number of different constituencies represented in the XML Digital Signature Working Group. Some were "Forms" people whose goal was the ability to insert into documents signatures that signed all or a subpart of the document (excluding the signature itself) without changing the root element. Others were protocol-oriented people who wanted, for example, to use XML signatures to sign specific elements in relatively short XML-encoded protocol messages without extreme verbosity. Others wanted to construct detached XML signatures that authenticated and perhaps asserted some properties of remote data which might not be XML at all but might, for example, be a binary executable file.

To satisfy these constituencies, XMLDSIG was made very flexible. Because some in these constituencies see their point of view as the only one and forget about the others, verbiage such as the phrase you quote above from the RFC, was included in the requirements and specification to make clear that all the bases were going to be covered.

The RFC also notes that this specification is not sufficient to address all application security-trust concerns, and additional requirements will be necessary. What do you envision as additionally necessary for a truly secure XML?

What is "truly secure XML?" The phrase is meaningless without a definition of what security properties you are trying to achieve and what your threat model is. XMLDISG provides a building block. It is a flexible mechanism for the cryptographic binding of data to a key.

Meeting some particular application's security requirements might mean specifying particular algorithms to be used, type of keying information allowed, semantic tags to be bound to signatures, security of key generation and distribution, security of the execution environment including TEMPEST considerations,

level of background checks on the personnel involved, administrative procedures including how to respond to compromises, choice of legal venue for resolving disputes, etc., etc., etc.

What you need to do to authenticate an anonymous political pamphlet (i.e., nothing) is not the same as what you need to do to authenticate a meeting room reservation inside a local network which isn't what you need to do to authenticate million dollar transactions on an multiparty international funds transfer system operating over the global Internet. The confidentiality you want for a press release (i.e., none) is not the confidentiality you want for an ordinary user's click trail which isn't the confidentiality you want for information about an as yet unannounced merger of major corporations, and that isn't the security you want when you are distributing authentication material for Emergency War Orders, the possession of which would theoretically enable you to launch armed nuclear missiles. There is no magic formula or pixie dust you can use to make something "secure" without knowing the security policy you want, the threats you want to secure against, etc. Most everything in my remarks above is a general characteristic of digital signatures or authentication codes and doesn't really have anything to do with whether they are XML, S/MIME, or whatever.

What do you see a digital signature adding (or subtracting) for XML message transport?

XMLDSIG cryptographically associates data with a key and can provide authentication and/or integrity. Since you ask specifically about message transport, an XML message sent from party X to party Y can, through an XML digital signature attached by party X, be authenticated by party Y as having come from party X and not been tampered with. This assumes that the key used by party X is only in the possession of party X and other trusted parties, if any; this only applies to the data elements of the message covered by the signature; and this assumes that party Y trusts the key in use to indicate signing by party X and trusts and has implemented the cryptographic algorithms in use.

To take a slightly less message-oriented and more document-oriented point of view, X could be the originator of an XML object/document. Then the signature affixed by X could authenticate and assure the integrity of that document to an unlimited number of recipients despite the signed object possibly being embedded in other XML and/or forwarded multiple hops or the like.

Data transport's goal is to assure message integrity (i.e., "not tampered with"). Now, can a XMLDSIG assure "non-repudiation" (ignoring that term's legal definition for the moment) the way that some vendors have been telling users?

Yes. To do that, you have to select an asymmetric signature algorithm, like RSA, have the originator sign with their private key, have those who want to verify the signature be able to know and trust the corresponding public key (possibly by having it be in a certificate sent in the KeyInfo part of the Signature signed by a Certification Authority the receiver trusts). If that is all true, then you have what is commonly referred to as non-repudiability by the signer of their signature.

For most purposes, you also have to add a semantic tag which says what the signature "means" (i.e., contractual agreement, witness, counter-signature to other signatures, authorship, etc.).

Perhaps part of the problem is that, to avoid cumbersome wording, XMLDSIG uses the term "signature" to include not only what's described above, which is what people usually mean by "signature," but also to include symmetric key authentication codes, or biometric-based authentication, or whatever. Even though, with a symmetric key authentication code, anyone who can verify the authentication code can forge one, so it can be repudiated, you might want to use this because it is so much more efficient.

If you are just trying to get trust in the integrity of communication from box A to box B over the Internet and you control both boxes, having each add an authentication code before sending a message to the other and then checking and stripping off such a code on receipt is reasonable. [That is, use of a MAC in the data stream can sometimes be a preferred method of authentication -- ed.]. This can give you a pipe secure against tampering (but not eavesdropping unless you also add encryption) at a relatively low cost in computational overhead.

Exactly so. XMLDSIG allows -- encourages, really -- multiple methods of authentication . I assume when you mentioned earlier that "to satisfy these [differing -- ed.] constituencies, XMLDSIG was made very flexible" that this property was part of that flexibility. And the specific method can be pointed to by a URI element, yes?

Yes. Some of the flexibility comes from the ability to specify cryptographic algorithms and the working group chose to use URIs as the names of algorithms. But there is also flexibility afforded by other places you can use URIs, by making some elements and attributes optional, by accepting a wide variety of keying information, etc. XMLDSIG is a toolkit with a wide variety of uses.

But it is important to note that in most real world systems, particular verifiers will probably have very strict policies about what of these many options they will accept. If one is looking for an RSA public key signature with public key modulus at least 1024 bits long and a currently valid certificate from your local company Certification Authority for that key, you would not consider an XMLDSIG which specifies another algorithm or key information type to authenticate what was signed for your purposes, even though it might be acceptable to some other verifier.

What uses do you think that the real world will have for XML digital signatures? Where do you think they will find their greatest usage?

I believe there will be a full spectrum of usage but two areas of particular prominence seem likely: documents and messages. While these sometimes blend into each other, documents tend to be longer lived and the signatures on them tend to indicate human approval of all or part of the document, although they may also have time stamp signatures affixed automatically. Messages tend to be transient and would more commonly have authentication automatically affixed and removed. Of course, a message could include one or more documents, in the sense I'm using the word here, in its content.

Documents are most likely to use public key techniques while messages, depending on the application, could use public key or symmetric secret key techniques. Documents are more likely to be something "important" like a mortgage or court filing. But if XML digital signatures are widely used for messages, messages could be several orders of magnitude more numerous.

What do you see as the minimal necessary security superstructure for an XML digital signature that will make it useful for real world applications?

It really depends on the application. It is always necessary that the signer and verifier have keying material, so the minimal "superstructure" is the manual configuration of the signer and verifier with keying material. While I'm sure this will be done in some cases, for most applications, you would want either a certificate system or a key distribution center as a source of trust in keys.

Resources

- In XML signatures: Behind the curtain, Larry Loeb questions the utility and usefulness of XML Digital Signatures.
- The W3C's XML Signature Working Group page is an excellent resource for more information on the topic, including the XML-Signature Syntax and Processing Recommendation.
- Donald Eastlake is a chair of the Digital Signature Working Group, whose charter explains what they hope to accomplish.
- The OASIS organization's Robin Cook compiled this compendium (with dates) on the digital signature effort.
- VeriSign's key management system uses XML signature as a trust basis. A downloadable Java client SDK is available here.
- Of course, you can find a wide range of security articles in the developerWorks Security topic, and our XML zone is always an excellent resource for developers working with the language.
- The XML Security Suite, available on alphaWorks, provides security features such as digital signature, encryption, and access control for XML documents.
- IBM Security Services can help you determine what your risks are, and then design a security program to address them.

About the author

Larry Loeb has written for many of the last century's major "dead tree" computer magazines, having been -- among other things -- a consulting editor for BYTE magazine and senior editor for the launch of WebWeek. He's been online since uucp "bang" addressing (where the world existed relative to !decvax), serving as editor of the Macintosh Exchange on BIX, and the VARBusiness Exchange. He's also written a book on the Secure Electronic Transaction Internet protocol. His first Mac had 128K of memory. His first 1130 had 4K, as did his first 1401. You can e-mail him at larryloeb@prodigy.net.

e-mail it!

What do you think of this article?

Killer! (5) Good stuff (4) So-so; not bad (3) Needs work (2) Lame! (1)

Comments?

Submit feedback