Secure XMaiL or How to get rid of legacy code in secure E-Mail applications

Lars Ewers[1], Wolfgang Kubbilun[2], Lijun Liao[3], Jörg Schwenk[3]

[1] Alsenstr. 23, 44789 Bochum, Germany
lars.ewers@web.de
[2] MediaSec Technologies GmbH, Berliner Platz 6-8, 45127 Essen, Germany
wkubbilun@mediasec.de
[3] Hörst Görtz Institute for IT Security, Ruhr-University Bochum
{lijun.liao, joerg.schwenk}@rub.de

Abstract: E-mail is one of the oldest applications on the internet. Clients have to adhere to message formats that have been defined in RFC 822 [13] back in 1982, and at the same time be able to transport all types of content. Additionally, there are severe restrictions for the use of both encryption and digital signatures due to the adherence to RFC822. In this paper we propose a new approach based on our XMaiL project: Using the XMaiL parser, we transform header and body of the mail into an XML object. This transformation preserves both the MIME and the PKCS#7 structure of the mail. We describe the security enhancements that are possible using XMaiL such as selective encryption and signature of parts of the e-mail, or signature of critical fields in the header of the mail.

1. Introduction

E-mail and the WWW are the two most important applications on the Internet, but e-mail is much older: RFC 822 [13] dates from August 13th, 1982, and the ASCII-based text format described therein is the basis for all mails sent on the internet today.

E-mail is one of the most dangerous internet applications: it is used to send computer viruses, spyware, malware, SPAM and Phishing mails to users [14]. This is partly due to the fact that mail clients must be able to understand many different data formats specific to e-mail.

Legacy Code. To include more than just text in mails, the "Multipurpose Internet Mail Extension" (MIME, RFC 2045 – 2049, [13]) defined a platform independent way to include arbitrary data in RFC 822 based mails. The different multipart message formats can be used to give mails a tree based structure. Although the basic MIME data types are also used in the http protocol, the structuring of data with multipart types and boundary ASCII sequences is only used in e-mail.

In 1998 MIME was extended to Secure MIME (S/MIME, RFC 2311, 2312, 2630, 2632, 2633 [13]) by introducing new, binary MIME types based on PKCS#7 [12], X.509 [15] and ASN.1 [2]. The interpretation of these binary data types can not be delegated to helper applications, but has to be done by the mail application itself. This is not a trivial task: After a good start with S/MIME implementations in at least three

popular e-mail clients (Netscape Messenger, Microsoft Outlook and Outlook Express), the abandonment of the Messenger 4.x product line by Netscape was a serious drawback. It took the Mozilla project years to come up with a new S/MIME implementation.

OpenPGP (RFC 1847, 3156 [13]) is a valid alternative to S/MIME, but it does not add new security features, and it also increases code complexity because of its binary format.

With the rapid development of the WWW, many of its standards were included in e-mail: HTML [7] coded mails have become fairly standard, and they may include hyperlinks and thus force the mail client to understand HTTP. It is to be expected that XHTML and the whole XML [16] standards family will follow soon.

Due to this steadily increasing number of data formats that have to be supported by mail clients, the development and support of mail clients becomes more and more difficult. Additionally, companies who have to store their mails for a long time due to legal reasons will need special databases to store this large amount of data in a structured way, and will also have to archive the client software needed to display it.

Security. S/MIME and OpenPGP have a common weakness when used to secure e-mails: They can only sign and encrypt the `Body` part of an e-mail, most parts of the `Header` remain insecure. The only exception is the address of the sender, included in the "`FROM:`" line: Each e-mail application should check if this address is identical to the one included in the PGP public key, or in the `Subject` or `SubjectAltName` fields of the X.509 certificate. Thus this field is only secure when the e-mail client has been implemented correctly.

All other fields of the Header remain insecure, even such important ones as "`TO:`" and "`DATE:`". Sending signed SPAM mails is thus possible by simply exchanging the "`TO:`"-line with an automated script, and is computationally easy since the signature has to be generated only once. Replay attacks are also possible.

As an example, consider a signed mail from a bank to one customer telling him that his account is nearly empty. Replaying this mail, and changing the "`TO:`" field, will generate a lot of confusion. If in addition the customer is asked to follow a link and enter his password, then combining this attack with a DNS attack on the bank's domain name, can be a serious threat.

Digital Signatures in S/MIME. Digitally signed content always comes as a pair in S/MIME and CMS. There are two options for signing content: clear signed (S/MIME data type multipart/signed) and opaque signed (CMS/PKCS#7 data type SignedData).

In theory these data structures can be nested, but in practice the mail clients restrict an e-mail to one such pair. (This holds for signature generation and verification.) As a result, header fields can never be signed, because the header must not contain S/MIME or CMS constructs.

Organisation of the paper. Section 2 gives a rough overview of e-mail security, including some research on XML. In Section 3, the basic ideas behind the XMaiL approach are presented. Section 4 goes into detail by discussing security enhancements of the XMaiL data format. Smooth transition scenarios from RFC 822 to XMaiL are discussed in Section 5. Our XMaiL parser described in Section 6 plays an important role here. Section 7 shows how normal mail and webmail can be displayed with XSLT, and future research is described in Section 8.

2. Related Work

The deployment of secure e-mail is limited because both S/MIME and OpenPGP are distributed applications, i.e. the user has to understand the security issues connected to encryption and digital signatures. There is a snapshot of the importance of secure e-mail in today's e-commerce in [6].In addition, encrypted e-mails can no longer be scanned for SPAM, viruses or other malware. Therefore there is a trend to centralize the encryption and signature of e-mails, ranging from automated client applications [5] to secure mail gateways (e.g. [4]). The XMaiL data format allows to combine both distributed and centralized security features, e.g. the content of the mail is signed by the user, but the authenticity of the sender's address and the encryption are guaranteed by the mail gateway.

Secure e-mails are considered to be one component in the fight against SPAM [11]. S/MIME or OpenPGP are used to authenticate the identity of the sender. Additionally, the authors mention that it would be desirable to secure header fields like the "RECEIVED:" lines. We move in this direction by signing security related header fields.

In [10] it has been proposed that in order to fight SPAM, the domain of the sender should be signed. With our approach this is easily possible, since we can split a "name@domain" mail address into a <name> and a <domain> tag, and only select the <domain> element to be digitally signed by the mail gateway.

Since XML is a very general data description language and since the two most important cryptographic primitives encryption and digital signature are available as XML co-standards [17, 18] it is a simple statement that all data formats contained in a secure e-mail can be replaced by XML structures. This idea has been published as the „eXtensible Mail Transport Protocol (XMTP)" [19] in 2000. However, this approach did not take into account the internal structure of e-mails and is thus not applicable to the ideas presented in this paper.

There is an ongoing debate about the advantages of binary and text based data formats [20]. However, the XML and ASN.1 standardization bodies are approaching each other, by defining XML based coding for ASN.1 (XER, see [2]), and by studying the advantages of a binary XML format [21].

3. XMaiL Data Format

Our approach is to model this structure as closely as possible using XML Schema. The final goal is to be able to transform an e-mail to XML, and to be able to transform this XML data back to the original e-mail. Advantages of this approach are the following:

- After transformation, e-mails can be stored in a structured form in any XML database. No special technology is needed to search this database.
- A complex workflow can be applied to the XMaiL inside a company, but it can be transformed to a valid MIME mail when sent over the internet.

- Important header fields can be secured by copying them as invisible Tags into the XML body of the message and signing them, or by first transforming them into an XMaiL, selecting them for signature with XPath, and then transforming them back to a normal mail.
- A complex workflow can be secured by including multiple signatures.
- Knowledge of mail transport protocols (SMTP, POP3, IMAP), RFC 822 and XML is sufficient to build secure e-mail applications.
- XMaiLs can be easily displayed through a webmail interface.

Table 1. A small example from RFC 822 and its translation to XMaiL.

```
From: John Doe <jdoe@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06

This is a message just to say hello. So, "Hello".
```

```
<?xml version="1.0" ... ?>
<XMaiL xmlns="http://www.xmlemail.org">
  <Header>
    <From>
      <Mailbox>John Doe &lt;jdoe@machine.example&gt;</Mailbox>
    </From>
    <To><Address>Mary Smith &lt;mary@example.net&gt;</Address></To>
    <Subject>Saying Hello</Subject>
    <Date>Fri, 21 Nov 1997 09:55:06</Date>
    <Mime-Version>1.0</Mime-Version>
  </Header>
  <Body>
    <Text>
      <Plain>
        This is a message just to say hello. So, "Hello".
      </Plain>
    </Text>
  </Body>
</XMaiL>
```

The XMaiL format informally described in Table 1 is defined with an XML Schema (available at xmlemail.org). This Schema is modelled after RFC 2822, MIME and S/MIME. In the XMaiL Parser, Java classes are automatically generated from this Schema, and the mail is parsed with JavaMail. The content of these Java classes is then stored in XMaiL format.

4. Improving the Security of E-Mails

Today's e-mail clients support HTML, which is often the default format for sending mails. Following the development of the World Wide Web, there is a clear migration path to XHTML and XML/XSLT. In these future mail clients, we can also expect support for XML Signature and XML Encryption, or at least it could be added with modest cost.

In XMaiL, the two XML security standards replace both S/MIME and OpenPGP, and for the first time we have a fully text based security standard. Table 2 compares the structure of a multipart/signed S/MIME message with an equivalent signed XMaiL message. (This should not be confused with the XMaiL output by our Parser prototype when applied to a S/MIME message; the result is much more complicated, because we have to adhere to the CMS syntax. See xmlemail.org for example results of our parser.)

Table 2. Equivalent of a clear signed S/MIME message in XMaiL, with two additional signed header fields (marked by underscores). Since the CMS type application/pkcs7-signature is a binary data type, only its structure is give with grey background.

RFC 2822, MIME & S/MIME	XMaiL
	<?xml version="1.0" ...> <XMaiL xmlns="http://..."> <Header>
... From: To: Subject: Date: ... MIME-Version: 1.0	... <From ID="Fromtobesigned"/> <To ID="Totobesigned"/> <Subject/> <Date/> ... <MIME-Version>1.0</Mime-version>
Content-Type: multipart/signed; protocol=”application/x-pkcs7-signature“; micalg=SHA-1; boundary=“-----12345------”	</Header> <Body> <MultipartSigned protocol=”application/x-pkcs7-signature” >
-----12345------ Here is the real MIME mail (optionally with attachments) -----12345------ Content-Type: application/pkcs7-signature; name=”smime.p7s“; Content-Transfer-Encoding: Base64	<MIMEMail ID="Mailtobesigned"> Here is the real MIME mail (optionally with attachments) </MIMEMail> <Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#"> <SignedInfo>
PKCS7: Content-Type: pkcs7-signedData Version: 1	<CanonicalizationMethod Algorithm="..."/> <SignatureMethod Algorithm="..."/> <Reference URI="#Mailtobesigned" Type="..."> <DigestMethod Algorithm="..."/> <DigestValue> 345x3rvEPO0vKtMup4Nbe8nk=... </DigestValue> </Reference>
Digest-Algorithms: Digest-Algorithm: Algorithm: sha1 (2B:0E:03:02:1A)	<Reference URI="#Fromtobesigned" Type="http://www.w3.org/2000/09/xmldsig#XMaiL"> <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

Parameter: Content: PKCS7: Content-Type: pkcs7-data	<DigestValue> hhdudkaslsdi743rundiu23=... </DigestValue> </Reference> <Reference URI="#Totobesigned" Type="http://www.w3.org/2000/09/ xmldsig#XMaiL"> <DigestMethod Algorithm="http://www.w3.org/ 2000/09/xmldsig#sha1"/> <DigestValue> 7dshdiw74hdh3h39j939=... </DigestValue> </Reference> </SignedInfo> <SignatureValue> MC0CFFrVLtRlk34FG6H9OGg5=... </SignatureValue> <KeyInfo> <X509Data> <!--ID public key--> <X509IssuerSerial> <X509SerialNumber> **549** </X509SerialNumber> </X509IssuerSerial> </X509Data> <X509Data> <!-- cert chain --> <!--Client cert--> <X509Certificate> MIICXTCCA.. </X509Certificate> <!-- Intermediate cert --> <X509Certificate> MIICPzCCA... </X509Certificate> <!-- Root cert --> <X509Certificate> MIICSTCCA... </X509Certificate> </X509Data> </KeyInfo> </Signature> </MultipartSigned> </Body> </XMaiL>
Certificates: Certificate: **Root-Zertifikat** Certificate: **CA-Zertifikat** Certificate: **Client-Zertifikat(549)** SignerInfos: SignerInfo: Serial-Number: 549 ... Signature: 0F:9B:46:5B:44:... -----12345------	

Table 2 also shows the additional possibilities of secure XMaiL compared to S/MIME and OpenPGP. The hash value which is signed not only covers the mail body, but also two important fields in the mail header, the “TO:” and the “FROM:” field. In our example there is still only one signature, but in principle there can be different signatures by different signers, e.g. a signature by the sender of the mail covering the mail body, and another signature from the company mail gateway covering the “FROM:” header field and thus certifying that the e-mail is indeed valid for the company.

The verification and visualization of signatures must no longer be implemented by the mail client, but can use standard XML technology: standard libraries (e.g. [1]) to

decrypt or verify signatures, and XSLT stylesheets which display the successfully verified parts of an e-mail in a different colour.

The only part of Table 2 which is not XML are the different X.509 certificates, which are only included in base64 coded form. This is an approach also followed in WS-Security (binary X.509 security tokens), but one can also think of a Certification Authority which issues the same certificate both in X.509 and XML format.

5. XMaiL and SMTP

If we transform the RFC 822 header of an e-mail to XMaiL as in Table 2, the current SMTP infrastructure will no longer be able to transport it. There are two ways to cope with this problem:

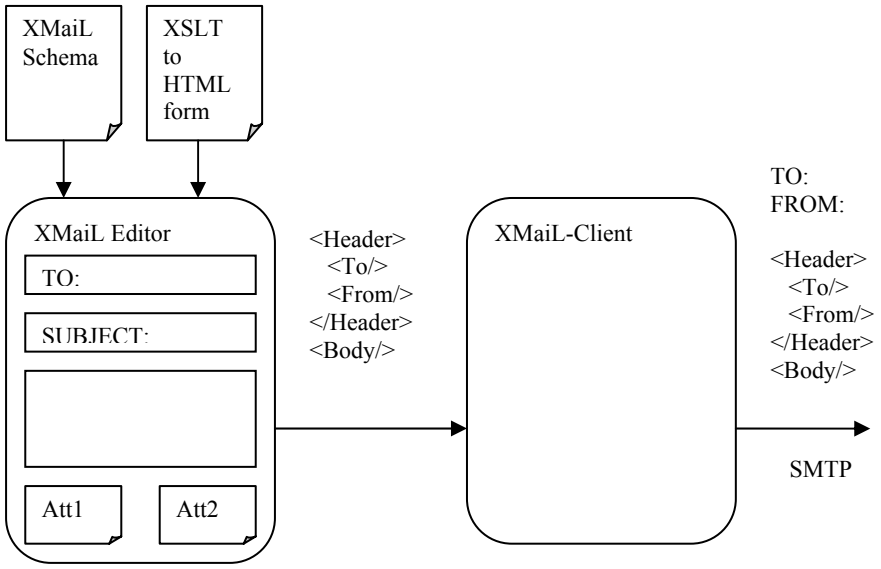

Fig. 1. Generating an RFC 822 header from an XMaiL.

The first possibility is to transform the RFC 822 header only to XMaiL to compute the signature, and then transform it back for SMTP transport. This approach works if the signed RFC 822 header fields are not modified during transport. To verify the signature, the header can be transformed to XMaiL again. If some unsigned header fields have changed (e.g. the "RECEIVED:" fields), this does not affect the signature.

The second possibility is to generate the RFC 822 header from the XMaiL header part for SMTP transport. Figure 1 shows this possibility. When such an e-mail is received, the signed parts of the <Header> element are displayed rather than the corresponding fields in the RFC 822 header.

6. XMaiL Parser Prototype

The XMaiL API structure is illustrated in Figure 2. It begins with the XMaiL schemas: XMaiL_Schema.xsd and smime.xsd. The former defines the basic data formats of an XMaiL, and includes the latter schema which is to be extended with the PKCS#7 [12] data formats. Using the XML binding compiler (*xjc*) of Sun [9], Java objects can be generated from the schemas.

We first consider the process to convert an RFC 2822 e-mail to the corresponding XMaiL. The core component is the XMaiL Parser. It takes RFC 2822 e-mails as input. To convert an RFC 2822 email without cryptographic components, the JavaMail API [8] converts it to JavaMail objects. The XMaiL API then picks up information from the JavaMail objects and generates the XMaiL objects. The current JavaMail API is not able to handle the PKCS#7 data formats. Hence to handle S/MIME e-mails the Bouncy Castle Crypto API [3] is additionally applied.

The XMaiL objects can be accessed by applications for some purposes, e.g. to verify the signatures, spam check, etc.. However, they are invisible to users and not storable. Hence an XMaiL writer based on JAXB API [9] is implemented. It converts the XMaiL objects to the corresponding XMaiL in XML format or represents them in an XMaiL Viewer.

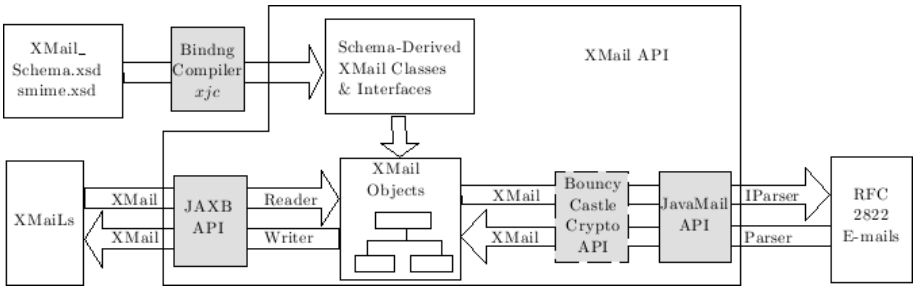

Fig. 2. XMaiL API

The inverse direction, converting an XMaiL to the corresponding RFC 2822 e-mail, can be also achieved by the API using the XMaiL IParser. First the Reader generates XMaiL objects from an XMaiL. The XMaiL can be from a file, or from an Editor, as in Figure 1. The XMaiL IParser either provides an interface for the normal email-client, e.g. Firefox, Outlook Express, to access the XMaiL, or converts the XMaiL to the RFC 2822 e-mail.

7. Displaying XMaiLs with XSLT

The XMaiL format closes the gap between “normal“ mail and webmail: The e-mail is displayed in the same way in the mail client and in the browser, only the retrieval method is different. It will also be possible to display the validity of the signature in a web browser, provided this browser is capable of verifying XML signatures.

Figure 3 shows a MIME message transformed to XMaiL and displayed with the Mozilla Browser using an “Outlook Express” XSLT transformation.

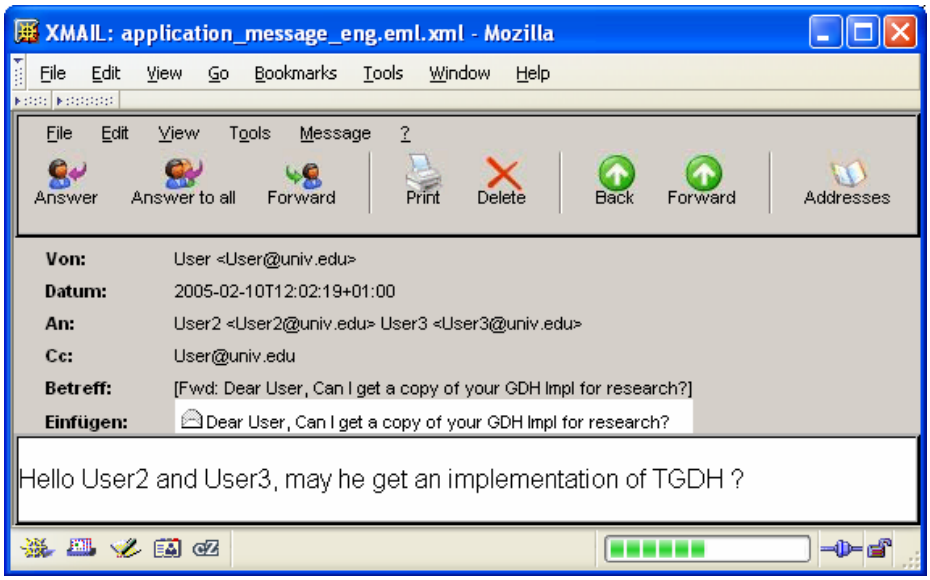

Fig. 3. The example mail transformed to XMaiL format and displayed using XSLT. The text part needs some more work.

With the application of the XMaiL IParser as plug-ins (described in Section 6), the XMaiLs can also be displayed with normal e-mail clients, e.g. Firefox and Outlook Express.

8. Future Work

The main goal of the XMaiL project is to give a proof-of-concept implementation that demonstrates a possible migration path from RFC 822 to XML based mail formats. The first step in this path is the XMaiL gateway which transform all kinds of mails (including the binary formats of S/MIME and OpenPGP) to XML and back. The interaction between Parser and iParser still needs some tuning, and OpenPGP has to be included.

The next step will be the implementation of an XMaiL editor with SMTP support, and an XMaiL viewer with POP3. Both tools will add and remove RFC 822 header lines, and will use Apache XML security routines. We will investigate if with XForms and XSLT a standard browser can be used, and how different signature levels can be displayed.

9. References

[1] Apache XML Security. http://xml.apache.org/security/.

[2] ASN.1 Information Site. http://asn1.elibel.tm.fr.

[3] The Legion of the Bouncy Castle, Bouncy Castle Crypto APIs. http://www.bouncycastle.org/

[4] IronMail Gateway. http://www.ciphertrust.com

[5] Lars Eilebrecht, *Ciphire Mail: Email Encryption and Authentication*. Financial Cryptography and Data SecurityNinth International Conference, February 28-March 3, 2005, Roseau, The Commonwealth Of Dominica.

[6] Simson L. Garfinkel, Jeffrey I. Schiller, Erik Nordlander, David Margrave, and Robert C. Miller, *Views, Reactions and Impact of Digitally-Signed Mail in e-Commerce*. Financial Cryptography and Data SecurityNinth International Conference, February 28-March 3, 2005, Roseau, The Commonwealth Of Dominica.

[7] World Wide Web Consortium, *Hypertext Markup Language*. http://www.w3c.org/MarkUp/

[8] SUN Microsystems, *JavaMail API*. http://java.sun.com/products/javamail/

[9] SUN Microsystems, *Java Architecture for XML Binding (JAXB)*. http://java.sun.com/xml/jaxb/

[10] Jason Levitt, *Tech Guide: Many Strategies Against Spam Can't Stem Frustration*. http://www.informationweek.com/story/showArticle.jhtml?articleID=13101046&pgno=3

[11] Barry Leiba, Nathaniel Borenstein, *A Multifaceted Approach to Spam Reduction*. First Conference on Email and Anti-Spam (CEAS) ***2004*** Proceedings *Mountain View, CA July 30 and 31, 2004*

[12] PKCS #7: Cryptographic Message Syntax Standard. http://www.rsasecurity.com/rsalabs/node.asp?id=2129

[13] Internet Engineering Task Force, *Request for Comments No. vwxy*. www.ietf.org/rfc/rfcvwxy.txt .

[14] SANS Institute, *The Twenty Most Critical Internet Security Vulnerabilities*. http://www.sans.org/top20

[15] http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.509

[16] World Wide Web Consortium, *eXtended Markup Language*. http://www.w3.org/XML/

[17] XML Signature WG, http://www.w3.org/Signature/

[18] XML Encryption WG, http://www.w3.org/Encryption/2001/

[19] Mediaone, *eXtensible Mail Transport Protocol*. http://xml.coverpages.org/xmtp20000508.html

[20] Darren P Mundy, David Chadwick and Andrew Smith, *Comparing the Performance of Abstract Syntax Notation One (ASN.1) vs eXtensible Markup Language (XML).* TERENA Networking Conference, 19-22 May 2003, Zagreb, Croatia.

[21] XML Binary Characterization Working Group, http://www.w3.org/XML/Binary/