Global Information Assurance Certification Paper

E-MAIL SECURITY WITH S/MIME

George Kuzmowycz

August 31, 2001

INTRODUCTION

Since I attended my first SANS Institute class the week after the 10th anniversary of the first release of PGP, and since I found no course material relating to S/MIME, this topic seemed to make sense. The intent of this paper is to present an overview of the history, design, usage and the current state of market and community acceptance of S/MIME while contrasting it, where appropriate, to PGP. A basic understanding of public-key concepts is assumed, as is some familiarity with the Internet RFC (Request for Comment) process and the X.500 standards..

PGP OVERVIEW

PGP (Pretty Good Privacy) was developed by Philip Zimmermann in 1991, as an attempt to bring encryption to the masses. The developer's own writings present the best overview of his motivations and of the "politics" of encryption at the time. [17]

S/MIME OVERVIEW

S/MIME (Secure Multipurpose Internet Mail Extensions) was originally proposed by RSA Data Security, Inc. in 1995, which then led an industry consortium including most of the major e-mail software and Internet browser vendors, such as Microsoft, Netscape and Lotus. Development work is now being coordinated by the IETF S/MIME Working Group.

ENCRYPTION/SIGNATURE

Although both PGP and S/MIME are referred to as "public-key" systems, this is not entirely accurate with respect to either specification. Due to the relative computational complexity of the calculations involved in the common public-key algorithms (RSA, ElGamal, etc.), message bodies are encrypted using one of several commonly-accepted symmetric-key algorithms, such as DES and its variants, IDEA, etc. The public-key encryption is used only for a small, fixed portion of the message: the session key used for the symmetric-key encryption of the full message body. Likewise with digital signatures – the public-key encryption is applied not to the full message body, but only to the message digest, most commonly computed using the MD5 algorithm. The following brief outline of the encryption processes is taken from RFC 1991 [1]; although that RFC is specific to PGP, this procedural description applies equally well to S/MIME

- the sender creates a message
- the sending [program] generates a random number to be used as a session key for this message only
- the sending [program] encrypts the message using the session key

- the session key is encrypted using the recipient's public key and prepended to the encrypted message
- the receiving [program] decrypts the session key using the recipient's private key
- the receiving [program] decrypts the message using the session key

S/MIME calls for a different set of encryption and signature algorithms than PGP. Because S/MIME development began in 1995, and the specification needed to work within US government export controls which existed until recently, S/MIME implementations have been required to support 40-bit RC2, which is known to be a very weak algorithm. Although tripleDES is also a supported algorithm, and is in fact recommended, some have criticized S/MIME for being cryptographically "weak," but it is weak only if a weak algorithm is chosen. The specification is very clear on the subject: "40-bit encryption is considered weak by most cryptographers. Using weak cryptography in S/MIME offers little actual security over sending plaintext. However, other features of S/MIME, such as the specification of tripleDES and the ability to announce stronger cryptographic capabilities to parties with whom you communicate, allows senders to create messages that use strong encryption." [12]

PGP KEY MANAGEMENT

The model for management of private and public keys in PGP was originally called the "Web of Trust." The developer understood that this one area was administratively the most difficult, and the most likely to retard large-scale adoption of encryption. The PGP documentation says: "This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. It is the 'Achilles heel' of public key cryptography, and a lot of software complexity is tied up in solving this one problem." [18]

Since the SANS/GIAC course material provides substantial detail regarding the generation and management of keys and the "man in the middle" attack, those subjects will not be discussed here.

S/MIME KEY MANAGEMENT

By contrast with PGP's "web" model, with interlocking trust relationships which can be assigned a "weight" or value by the user, S/MIME was designed from the outset as a purely hierarchical model. Keys or certificates are trusted based on the "trustworthiness" of the issuer, which is assumed to be of a higher value than that of the user. The line of trust can be followed up the chain of certificates to some root, which is generally a large commercial organization – a Certificate Authority (CA) – engaged purely in the business of verifying identity and assuring the validity of keys or certificates, e.g Verisign. To a large degree this solves the key identity problem discussed under PGP above, since in order to obtain a certificate (key pair) from such an entity I have to provide some rudimentary identification which can conceivably be verified, at least by the certificate authority, such as an address and credit card information. The commercial nature of the transaction imposes minimal controls, and may even be associated with

some legal liability on the part of the issuer.

In practice, this turns out to be somewhat less trustworthy than the ideal, though, since a large CA dealing with a large volume of low-value transactions has limited resources to do the validity checking that is implied by the model. A Verisign certificate for an individual user costs less than $15/year; a server certificate or code-signing certificate costs $300-400. It may be unreasonable to expect much in the way of identity verification for such sums. Several examples clearly illustrate the possibility of fraud or misrepresentation: the well-known case which came to light in March of 2001, where Verisign issued two code-signing certificates to someone falsely claiming to represent Microsoft [4]; or a 1997 case in which cryptographic researcher Kevin McCurley obtained a Verisign certificate for the email address root@localhost [9]. The potential for fraud is clear in either case: a false code-signing certificate can persuade an unsuspecting user to install, for example, a malicious ActiveX control, and a "signed" message from the very common and very often highly-privileged e-mail ID of "root@localhost" can plausibly and successfully ask a user to reveal information which should be secure. Recommendations that users exercise "due diligence" seldom produce any meaningful results, as most users don't know whom or what to trust, when or under what circumstances.

The ability of CA's to revoke certificates though the publication of Certificate Revocation Lists (CRLs) likewise is problematic, since it requires an affirmative action on the part of the user to download and apply a CRL to their certificate store. If they do not download and apply CRLs regularly, then they cannot possibly be aware of certificates deemed to be invalid. No specification calls for automatically contacting a CA to receive a CRL.

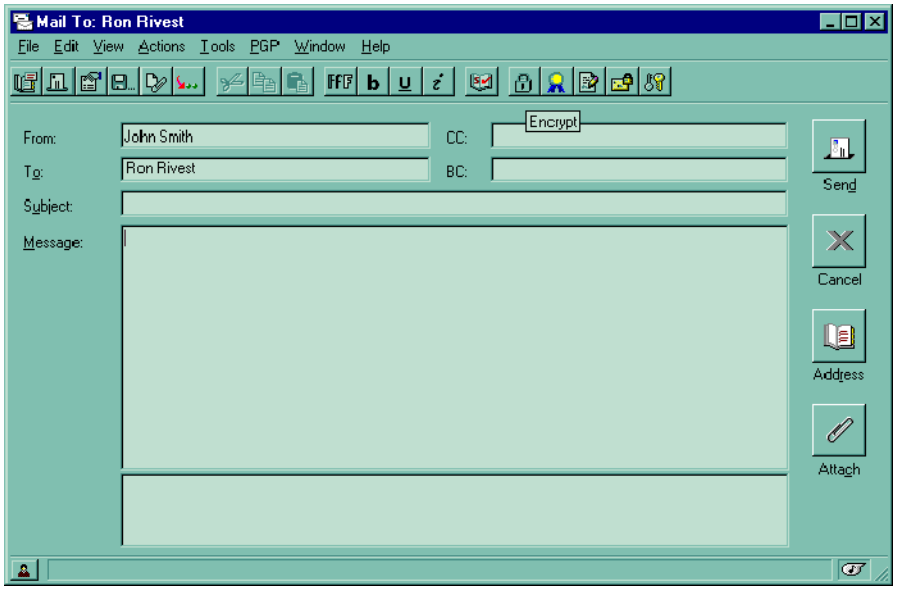
The chain of trust or authority with X.509 certificates will very quickly lead back to the "Trusted Root Certification Authorities," a large number of which are installed by default with an Internet browser installation. This (presumably knowledgeable) author's installation of IE 5 has, at present, 40 "Trusted Root" certificates; all a malicious user needs to do is to deceive one of those trusted root CA's, and I have a problem. The operation of each CA as it relates to trust is a legal and not a technical issue, and thus is not susceptible to technical remedies. The X.509 specification deals in detail with the format of the certificates, but does not deal with the trust or verification issue at all. This is subject to each CA's Certification Practice Statement (CPS); to the extent that these operating rules vary by CA, not only the expectations of each user can vary, but this can also affect each application's interpretation of what a certificate actually certifies.

Despite these drawbacks with X.509 certificate trust and management, the weight of the market is heavily influencing their widespread adoption, and therefore tilting the scales toward S/MIME. X.509v3 certificates are used to secure SSL sessions and SET transactions. The major Internet browsers (Microsoft and Netscape) have X.509 certificate support built-in, and the certificate infrastructure is extended to their respective e-mail products (the Outlook series of products and Netscape Messenger.) Even though there are elegant plug-ins integrating PGP with nearly every common e-mail application on most

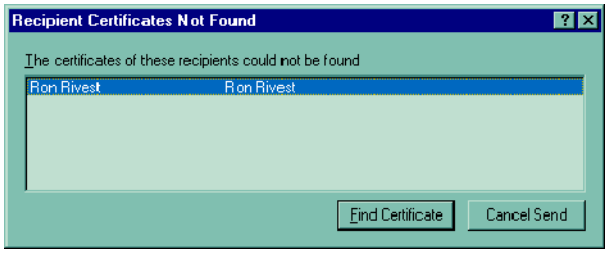
operating systems, the need to install and understand an extra product is always a stumbling block. When it comes to an area as intrinsically complex as encryption and security, it is probably insurmountable. An often-cited study by Alma Whitten of Carnegie Mellon and J.D. Tygar of U.C. Berkeley [16] showed that a majority of users in their sample could not correctly use PGP 5.0, and in some cases actually compromised security by improper use or misunderstanding. One-fourth of the people in the sample sent messages intended to be encrypted as plain text because they did not understand the program. Only one-third of the subjects were able to generate a key pair, obtain others' public keys and correctly sign and encrypt an e-mail message even when given 90 minutes to do so. Even an authority as prominent as Simson Garfinkel, author of several books on UNIX security and one book on PGP, has written "I hate getting encrypted email; it's a pain." [5]

S/MIME OPERATION

The built-in and nearly "automatic" nature of S/MIME-X.509 will presumably obviate the usability problem. The necessary cryptographic infrastructure is installed in default installations of Internet Explorer and Outlook Express as well as Netscape Navigator and Messenger, and the major CA root certificates are installed as well. From that point on, simply sending a digitally signed message is sufficient to install the sender's public key into the recipient's e-mail application. Encrypting a message using an S/MIME-compliant application is simply a matter of entering an addressee and then choosing the "Encrypt" function. An example from Novell's GroupWise e-mail program is shown below



The ScreenTip saying “Encrypt” is from the toolbar icon which looks like a padlock. (Sharp-eyed readers will note that this installation of GroupWise also includes the PGP plug-in) Once this option is selected and a message is composed, selecting “Send” will cause the application to search the local certificate store to locate an appropriate X.509 certificate for the addressee. If a certificate is found, the message is encrypted and sent. If a certificate is not found, the application notifies the user and prompts for a search, as follows:



MIME MESSAGE TYPES

S/MIME, as implied by the name, is a set of extensions to the MIME message format, as originally specified in RFC 1521 [2]. While the development of PGP pre-dates the MIME message format, and in its operation PGP can create messages that can be delivered and processed even by non-MIME-compliant mail systems, the encryption and signing functions specified by S/MIME cannot be handled by a non-MIME-compliant mail system. The S/MIME specification provides for several MIME content types to do this. “Content Types” are defined in RFC 1521 as follows:

> The Content-Type header field is used to specify the nature of the data in the body of an entity, by giving type and subtype identifiers, and by providing auxiliary information that may be required for certain types. After the type and subtype names, the remainder of the header field is simply a set of parameters, specified in an attribute/value notation. The set of meaningful parameters differs for the different types.
>
> In general, the top-level Content-Type is used to declare the general type of data, while the subtype specifies a specific format for that type of data. Thus, a Content-Type of "image/xyz" is enough to tell a user agent that the data is an image, even if the user agent has no knowledge of the specific image format "xyz".

In practical usage, the user’s application program (e-mail reader, browser, etc.) looks at the Content-Type header and at the file name to determine how to present the data to the user. Most commonly this is used, for example, to display an HTML message with its formatting, to display a JPEG file in the appropriate image display program, and so on. The various S/MIME content types likewise automatically invoke the relevant features of the e-mail program.

There are two major S/MIME content types in common use, two subtypes and several parameters. For digital signatures, the message being signed can be sent “in the clear” (as readable plain text) with a signature attached, or as signed data, such that the body text is not transmitted in the clear. The approach taken determines the MIME content types. A clear-signed message shows a content type of Multipart, with a subtype of Signed, and no parameters. The message is then divided into parts as any MIME multipart message would be, with the text identified with a content type of text/plain, and the signature showing a content type of application/pkcs7-signature. A signed message that is not “clear signed” is one that has been transformed by an encryption algorithm, but is actually encrypted using the sender’s private key and can thus be decrypted by anyone having the corresponding public key – so it is not secure, but is also not transmitted as clear text. Such a message is sent with a content type of “application,” a subtype of “pkcs7-mime” and a parameter of “signedData.” A comparison of the two formats is shown below, with many of the RFC 822 headers omitted for clarity.

CLEAR-SIGNED MESSAGE

```
Content-Type: multipart/signed

–boundary
Content-Type: text/plain

This is the clear text.

–boundary
Content-Type: application/pkcs7-signature; name=smime.p7s

BRA4g6qrhDg8gGAwINUBAXnXAv9dOVbhfDgrjaj2qD4cDTUQ1skHE0XTryzZ
tMjTa7
xWn3e2yvmO8qu23FXcyPJFvd86BTwv0hEPmbeiYA4Ki3jT3955Hv2iUBfyvA
....

–boundary
```

SIGNED-DATA MESSAGE

```
Content-Type: application/pkcs7-mime; smime-type=signed-
     data;
     name=smime.p7m

mQCNAziDqqsAAAEEAJbbaOUM4XXlMTM3f2q92jeFxNylCF8c94Ij7gAAsuF2
2V
yfXJOIfhPvTltGsjObE72Z7s3XFYafy54lIVyyIqtCNTXRs9xB6pHjtANvXd
....
```

ENCRYPTED MESSAGE

To send secure, encrypted data using S/MIME, the application will utilize the content type of "Application" with a subtype of "pkcs7-mime" and a parameter of "envelopedData." This would be a message transformed by an appropriate encryption algorithm using a session key encrypted with the recipient's public key, which would thus be readable only by the person possessing the corresponding private key (presumably the intended recipient.) The relevant S/MIME headers would appear as follows:

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-
     data;
     name=smime.p7m

ftsHInozruBF2WeIUwp7QgqE9PcKBrl4QXO6AE7OwXjdqh6gqIQ4PIBgMCDV
AAID
```

```
tCpBbm9ueW1vdXMgUmVtYWlsZXIgPG1peEBtaXhtYXN0ZXIuY2V0aS5wb...
.
```

Although there are at least two other S/MIME object types existing in older specifications, and intended for transmitting certificate registration requests or certificate revocation lists, these are rarely used and have not remained in the S/MIME v3 specification, so they are not discussed here.

THE STANDARDS PROCESS

Both PGP (through the OpenPGP consortium) and S/MIME (through the efforts of RSA and other vendors, and now the IETF S/MIME Working Group) have been on the IETF standards track since at least 1997. Earlier versions of each specification encountered insurmountable objections due to the use of algorithms subject to patent (IDEA in the case of PGP and RSA in the case of S/MIME.) As the specifications evolved, the reliance on patented algorithms has been eliminated, either due to revisions in specifications or, in the case of RSA, due to the expiration of patents. S/MIME, in particular, has undergone many revisions and developments, and is currently at v3. Neither, however, can lay claim to being a "standard." Both are still only at stage two of a four-step standards adoption process, and are at the time of this writing deemed to be "proposed standards," not yet even "draft standards." [10]

THE FUTURE

As the Internet Mail Consortium, which has been involved in the standards process since 1997, says: having two protocols that do the same thing is much worse than having one. [8] It is unclear which standard will prevail – the "Internet community" is fond of PGP because of its age and its impeccable (i.e. non-commercial) pedigree, but the weight of the market is pushing toward S/MIME. As we have seen in a wide variety of markets, the "best" standard is often not the prevailing standard. Only one fact seems clear: the vast majority of Internet e-mail is not routinely encrypted. This seems unlikely to change in the near future.

REFERENCES

[1] D. Atkins, W. Stallings, P. Zimmermann, "PGP Message Exchange Formats (RFC 1991)", http://www.ietf.org/rfc/rfc1991.txt

[2] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One (RFC 1521)", http://www.ietf.org/rfc/rfc1521.txt

[3] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, "OpenPGP Message Format (RFC 2440)",
http://www.ietf.org/rfc/rfc2440.txt

[4] Brian Fonseca, "VeriSign issues false Microsoft digital certificates", InfoWorld, March 22, 2001,
http://infoworld.com/articles/hn/xml/01/03/22/010322hnmicroversign.xml

[5] Simson Garfinkel, "Pretty Good Politics",
http://hotwired.lycos.com/packet/garfinkel/97/18/index2a.html

[6] Ed Gerck, "Overview of Certification Systems", July 18, 2000,
http://www.mcg.org.br/certover.pdf

[7] International Telecommunications Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", March, 2000,
http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.509-200003-P

[8] Internet Mail Consortium, "S/MIME and OpenPGP", http://www.imc.org/smime-pgpmime.html

[9] Kevin McCurley, "DigiCrime is now known as root@localhost",
http://www.digicrime.com/id.html

[10] "Official Internet Protocol Standards", http://www.rfc-editor.org/rfcxx00.html

[11] OpenPGP Alliance, http://www.openpgp.org/about_openpgp/history.shtml

[12] B. Ramsdell, "S/MIME Version 3 Message Specification (RFC 2633)"
http://www.ietf.org/rfc/rfc2633.txt

[13] RSA Security, Inc., "S/MIME Frequently Asked Questions",
http://www.rsasecurity.com/standards/smime/faq.html

[14] RSA Security, Inc., "Public-Key Cryptography Standards",
http://www.rsasecurity.com/rsalabs/pkcs/index.html

[15] William Stallings, “Cryptography and Network Security:Principles and Practice, Second Edition.” Upper Saddle River, NJ: Prentice Hall, 1999.

[16] Alma Whitten and J.D. Tygar, “Why Johnny Can't Encrypt.” August, 1999, http://www.cs.cmu.edu/~alma/johnny.pdf

[17] Philip Zimmermann, “Why I wrote PGP”, http://web.mit.edu/prz/essays-WhyIWrotePGP.shtml

[18] Philip Zimmermann, “PGP User’s Guide, Volume 1”, ftp://ftp.pgpi.org/pub/pgp/2.x/doc/pgpdoc1.txt