The University of Saskatchewan
Department of Computer Science

Technical Report #2011-03

UNIVERSITY OF SASKATCHEWAN

Cryptographic Security for Emails: A Focus on S/MIME

Minhaz Fahim Zibran
Department of Computer Science
University of Saskatchewan
Email: minhaz.zibran@usask.ca

Abstract

In this paper I present a study on "S/MIME", which has become the industry standard for secure email exchange. Based on existing literature review, the study examines S/MIME in depth with specific emphasis on its architecture, strengths, and deficiencies. The study also identifies usability issues related to S/MIME enabled email clients, which indicate scopes for further improvements in those implementations. Obstacles in the adoption of S/MIME are also identified indicating what is required for its successful adoption in the community.

In presenting the study, the paper contributes in two ways: (a) for any newcomer in the field of cryptography this paper will be a useful resource to quickly learn about S/MIME in a fair level of detail, (b) the indication about limitations of S/MIME and its implementations reveals an avenue for further research in the area of email security, which may result in improvement of S/MIME itself, or its implementations in the email clients.

Keywords:

Email Security, S/MIME, MIME, PGP, PKI, Certificate, Email Authentication, Email Encryption, Key Management

1 Introduction

Email has been a very common medium of communication these days. It somewhat replaces the traditional surface mail and many of the traditional ways of communication [32]. Today people send and read emails from their personal computers, business workstation, PDAs and even cell phones. As people do more business communication over email, their requirement for email security increases. A huge volume of information travels over internet among people and organizations. The flow of sensitive information and important business correspondence over emails needs to be secured from any possible forgery. Moreover, there is need for privacy, authentication of authorship, and confirmation of email delivery to the destined recipient [10, 14]. In surface mail privacy is maintained, as messages remain sealed in envelopes. Surface mail delivery may be confirmed by taking signature from the recipient upon delivery of the mail. But such mechanisms are not available for email messages.

Emails can be somewhat protected by restricting their flow over secure communication links within a fixed domain of trusted computing devices. But implementation of such control over large scale distributed management environment like the world wide web is too difficult, if not impossible. Hence, "the only way to protect Internet mail is through the use of cryptography" [10].

Without surprise, significant efforts have been made to apply cryptographic techniques to achieve email security. Among them, Secure/Multipurpose Internet Mail Extension (S/MIME) and Pretty Good Privacy (PGP) are probably the most widely used. Despite the availability of such email security mechanism, in practice, the use of email encryption is still rare [9]. So, it is interesting to investigate what keeps email users away from using secure email system, even though the demand of email security is increasing. This understanding would be useful to develop secure email systems widely adoptable by email users. This paper presents a study on the security mechanism provided by S/MIME, which has been the industry standard for secure email exchange.

The remaining of the paper is organized as follows. Section 2 first points out important security features desired for email security. Previous efforts to achieve these security features are then described in short. Section 3 describes the S/MIME standard in detail with a preliminary discussion on MIME. The security capabilities of S/MIME are then wrapped up in section 4. Security issues and limitations of S/MIME and its implementations are discussed in section 5. Section 6 describes the usability issues related to S/MIME and secure email systems. Finally, section 7 concludes the paper with some remarks on present state of the art, future trend, and open issues.

2 The Quest for Email Security

A secure email application should ensure the following three properties [14]:

i. Confidentiality: message confidentiality or privacy ensures that the message can only be read by the intended recipient, no one else.

ii. Authenticity: message authenticity certifies that the message originally came from the specified author.

iii. Integrity: integrity implies that the message arrived to the recipient in the same state as sent by the sender without having altered on its way.

Having confidentiality, authenticity, and integrity as the primary goal for email security, the following features may be desired in a viable certified email service [32]:

a) Non-repudiation of origin: the receiver must be given a way to prove that a given email indeed originated by the specified sender.

b) Non-repudiation of receipt: the sender must have a way to prove that the intended recipient indeed received the email.

c) Strong fairness: strong fairness in the exchange of email indicates that the recipient should be delivered the email, if and only if the sender obtains a receipt for it.

One of the earliest efforts aiming email security was Privacy Enhanced Mail (PEM), which introduced the concept of using trusted Certification Authority (CA) in secure email systems. Pretty Good Privacy (PGP) was another effort towards the same goal, which was adopted by many individuals.

2.1 Privacy Enhanced Mail (PEM)

Extending the RFC822 internet mail message standard [6], the Internet Activities Board's Privacy Task Force developed new standards known as Privacy Enhanced Mail (PEM) embodied in RFC (Request For Comment) 989 [26], which was issued in 1987. After undergoing two times revision the final set of PEM standards [1, 20, 22, 27] issued in 1993.

PEM defined two major features for ASCII message protection: (1) signed messages, and (2) singled and encrypted messages. Message is encrypted using symmetric key encryption (DES) and signed using RSA public key encryption. Users need to publish their RSA public keys digital certificates conforming the X.509 CCITT standard. These certificates require to be signed using private RSA of trusted Certifying Authority (CA). The CA's public key itself need to be placed in another certificate, which itself could be signed by another CA, and so on, thus forming a chain of certificates with a single trusted root [10].

PEM took too long, nearly five years to complete the IETF (Internet Engineering Task Force) standardization, and by this time it was overtaken by events, and so it could never be deployed to any great extent [5]. The major problem with PEM was the requirement that the users' public keys needed to be certified by their local CAs. These local CAs needed to be certified by a policy CA, which itself needed to be registered to the root CA named IPRA (Internet Policy Registration Authority). Nonexistence of such an infrastructure resulted the inertia in establishing PEM based email systems [5]. RIPEM (Riordan's Internet Privacy Enhanced Mail) is a public domain implementation of PEM [40].

2.2 Pretty Good Privacy (PGP)

PGP was primarily developed by Phillip R. Zimmermann in 1991. It allows encrypting and digitally signing email messages, individual files or protecting complete file systems [41]. PGP uses both public key cryptography and private key cryptography to protect information against eavesdropping or forgery [15]. Every user uses a pair of keys, one public and another private to securely exchange message. To send a message, the author generates a session key and with that encrypts the message using symmetric key encryption technique. Then he applies asymmetric key encryption technique to encrypt the session key with the recipient's public key. Optionally the sender may sign the digest of the message using his private key. Then the encrypted message along with the encrypted session key and the signed digest is sent to the recipient. The receiver recovers the session key by decrypting

using her private key. She uses the session key to decrypt the message. Using the sender's public key she verifies sender's signature on the message digest.

Today implementation of PGP is available for use on all major operating systems and integrates as a plug-in to email systems, such as Microsoft Outlook, Lotus Notes, Eudora mail system, and Novel GroupWise.

PGP's approach to certification is its primary advantage, which is also its major difference from PEM. Unlike PEM, PGP does not require any centralized Public Key Infrastructure (PKI) with single root. PGP does not necessitate the users' public keys certified by CAs. PGP users can independently certify keys belonging to other users. It is the user who decides his or her level of trust on certain user's key or certificate. Consequently PGP establishes a trust model described as "web of trust" [10, 41], "public trust model" [30], "user centric trust" [31], and "web of confidence" [45].

A salient drawback of PGP is in its key distribution mechanism using public PGP key servers, where anyone can create and use a key having any name on it [5, 10]. For example, there may be numerous keys on the PGP key servers with the name "Bill Gates" on them, but none of them may actually belong to the founder of Microsoft Corporation.

3 S/MIME

PEM, which is based on the RFC822 [6] message standard aimed to provide *textual* email message security. Similarly, "non-textual objects cannot be exchanged with PGP" [44]. However, with the increase of power in terms of computation and networking capability, people's need for using non-texual objects such as image, audio, and video in emails became a fair demand. To support diversity of content, multipart message structure, and non-English text, the Multipurpose Internet Mail Extension (MIME) was proposed by IETF in 1992 [3]. Then a number of works have been done to combine the security feature of PGP with MIME, for example, OpenPGP and PGP/MIME [44]. S/MIME (Secure/Multipurpose Internet Mail Extension) is an enhancement of MIME to provide cryptographic security [8] for the MIME based emails. To better understand S/MIME, let us first take look at MIME.

3.1 MIME

MIME allows non-ASCII data to be sent through emails. It transforms non-ASCII data at the sender's MTA (Message/Mail Transfer Agent) [2] to ASCII data and delivers to the client MTA over Internet. The message in the receiving site is transformed back to the original data. We may think MIME as a set of software functions that that transforms non-ASCII data to ASCII data and vice versa.

MIME defines five hearders, which can be added to the original email header section to define the content transformation parameters. These hearders are as follows:

i. MIME-Version defines the version of MIME used. The current version is 1.1.

Type	Sub-type	Description
	Plain	Unformatted
	HTML	HTML format
Multipart	Mixed	Body contains ordered parts of different data types.
	Parallel	Same as above but without order.
	Digest	Similar to Mixed but the default is message/RFC822
	Alternative	Parts are different versions of the same message.
Message	RFC822	Body is an encapsulated message.
	Partial	Body is a fragment of a larger message.
	External-Body	Body is a reference to another message.
Image	JPEG	Image is in JPEG format.
	GIF	Image is in GIF format.
Video	MPEG	VIdeo is in MPEG format.
Audio	Basic	Single channel encoding of voice at 8 KHz.
Application	PostScript	Adobe PostScript.
	Octet-Stream	General binary data (eight bit bytes).

Table 1: MIME Content 'types' and 'sub-types' [8]

ii. Content-Type defines the type of data used in the body of the message. It also defines content 'sub-type', which follows the 'type' separated by a slash. MIME allows seven different types of data, which are listed in Table 1

iii. Content-Transfer-Encoding defines the method used to encode the messages into binary bits (0s and 1s) for transport. The five types of encoding methods are listed in Table 2.

iv. Content-Id uniquely identifies the entire message in a multiple message environment.

v. Content-Description defines whether the body of the message is image, audio, or video.

3.2 S/MIME

S/MIME adds some additional content types to the MIME to provide security services. The current S/MIME version 3.1 obsoletes all earlier versions. However, most implementations still bear version 3.0 features for digital signature processing [24]. Today, popular email clients such as MS Outlook XP, MS Outlook 2007, MS Outlook Express, Lotus Notes, Netscape Communicator, Eudora 7.1, and Mozilla Thunderbird 1.5 support S/MIME enabled messages.

Type	Description
7 bit	NVT ASCII characters and short lines.
8 bit	Non-ASCII characters and short lines.
Binary	Non-ASCII characters with unlimited-length lines.
Radix-64	6-blocks of data are encoded into 8-bit ASCII characters using Radix-64 conversion.
Quoted-printable	Non-ASCII characters are encoded as an equal sign followed by an ASCII code.

Table 2: MIME Content-Transfer-Encoding [8]

This paper concentrates on S/MIME version 3.0 and 3.1. S/MIME version 3.1 is defined by the following five major specifications.

i Cryptographic Message Syntax (RFC 3852 [18] updated by RFC 4853 [19])

ii Cryptographic Message Syntax (CMS) Algorithms (RFC 3370 [17])

iii S/MIME Version 3.1 Message Specification (RFC 3851 [37])

iv S/MIME Version 3.1 Certificate Handling (RFC 3850 [36])

v Diffie-Hellman Key Agreement Method (RFC 2631 [38])

An additional protocol, Enhanced Security Services for S/MIME (RFC 2634 [16]) was proposed describing four optional security services extensions to S/MIME to allow signed receipts, security labels, and secure mailing lists. By the use of signed receipts S/MIME is expected to provide non-repudiation of recipient. The purpose of security labels is to support access control and routing decisions related to the encrypted message. Detail of the specifications can be found in the S/MIME mail security charter of the IETF website at http://www.ietf.org/html.charters/smime-charter.html.

3.2.1 Cryptographic Message Syntax (CMS)

To define how security services, such as confidentiality or integrity, can be added to MIME content types, S/MIME defines Cryptographic Message Syntax (CMS). The syntax in each case defines the exact encoding scheme for each content type. Discussed below are the different types of messages and different sub-types that are created from these messages [8].

Data Content Type is an arbitrary string. The object created is called 'Data'.

Digested-Data Content Type is used to provide *integrity* for the message. The result is typically used as the content for the enveloped-data content type. The encoded result is an object called 'digestedData'. The process of creating 'digestedData' involves the following two steps.

i Using the hash algorithm of the user's choice a message digest is created from the content.

ii The message digest, the algorithm, and the content are added together to create the 'digestedData' object.

Figure 1 shows the process of creating 'digestedData'.

Signed Data Content Type provides *authenticity* and *integrity* of data. It contains any type and zero or more signature values. The encoded result is an object called 'signedData'. A signed data message can only be viewed by the recipient of with S/MIME capability [39]. Figure 2 shows the process of creating 'signedData'. The following are the steps in the process.

i For each signer, a message digest is created from the content using the specific hash algorithm chosen by the signer.

ii Each message digest is signed by the signer with his or her private key.

iii The content, signature values, certificates, and the algorithms are then collected to create the 'signedData'.

Enveloped-Data Content Type is used to provide *privacy* for the message. It contains encrypted content of any type, and zero or more encrypted keys and certificates. The encoded result is an object called 'envelopedData'. Figure 3 shows the process of creating an object of this type. The steps involved in the process is are as follows.

i A pseudorandom session key is created for the symmetric-key algorithm to be used to encrypt the content.

ii The content is encrypted using the defined algorithm and created session key.

iii For each recipient, a copy of the session key is encrypted with the public key of the recipient.

iv The encrypted contents, encrypted session keys, algorithm used, and certificates are encoded using Radix-64.

Authenticated-Data Type is used for providing *authentication* of the data. The resultant object is called 'authenticatedData'. Figure 4 shows the process of creating 'authenticatedData' object. The steps to create this type of object are as follows.

i For each recipient, a MAC (Message Authentication Code) key is generated using a pseudorandom generator.

ii For each recipient, A MAC (encrypted content) is created applying the MAC algorithm to the content.

iii The MAC key is encrypted with the public key of each recipient.

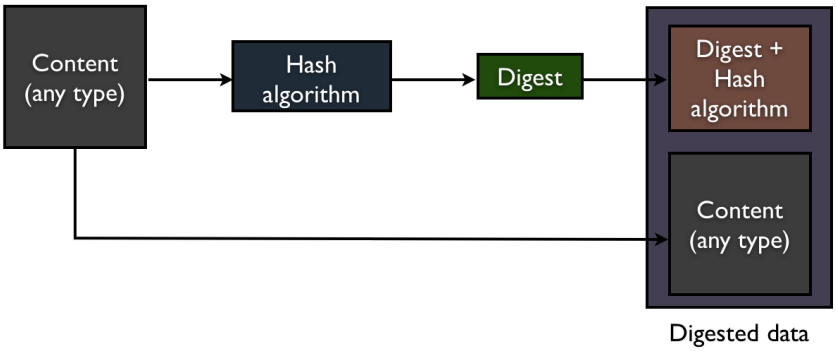

Figure 1: Creation of digested data

iv The content, MAC, algorithms, and other information are collected together to form the 'authenticatedData'.

Encrypted-Data Content Type is used to create an encrypted version of any content type. Although, it looks similar to the enveloped-data content type, the encrypted-data content type has no recipient. It can be used to store the encrypted data, rather than transmitting it. The process is very simple. The user employs any key (typically driven from the password) and any algorithm to encrypt the content. This encrypted content is then stored without including the key or algorithm. The resultant object is called 'encryptedData'.

3.2.2 Cryptographic Algorithms

S/MIME specifies several cryptographic algorithms for use, which are listed in Table 3. The term 'must' indicates absolute requirement, and the term 'should' implies recommendation only. S/MIME recommends using SHA-1 (not MD5) as hash function for creating message digests. For content encryption triple DES is recommended.

3.2.3 S/MIME Messages

The set of new content types that S/MIME uses are listed in Table 4. Table 5 shows an example of MIME header and Table 6 shows an example of S/MIME header for a message with enveloped data. The portion between the the original email header and the email body represents the MIME and S/MIME headers.

3.2.4 Key Management and Certification

The key management in S/MIME is a combination of key management used by X.509 and PGP. S/MIME uses public key certificates signed by the Certification Authorities (CA)

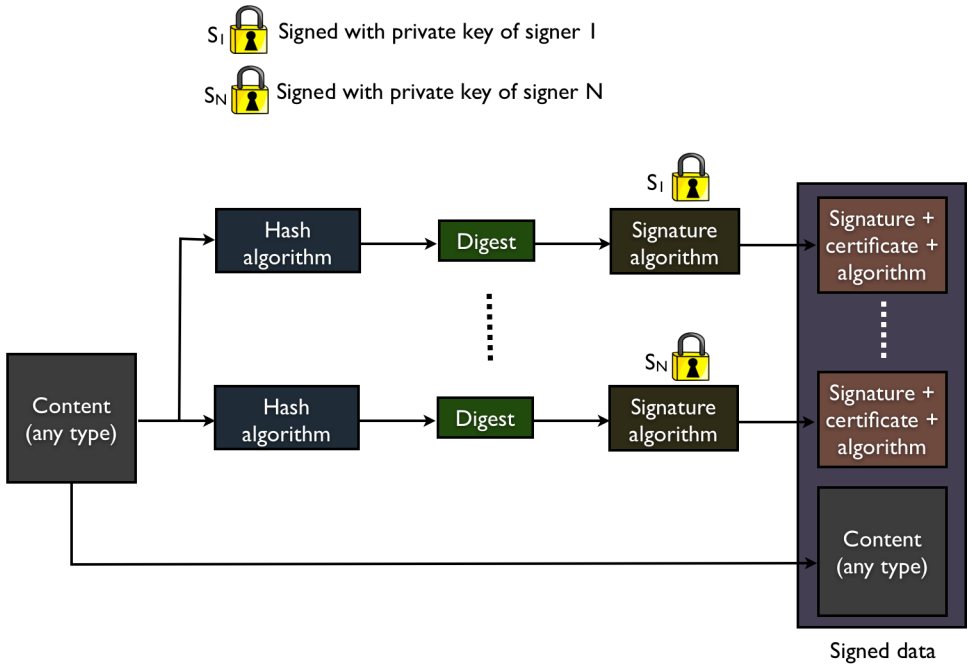

Figure 2: Creation of signed data

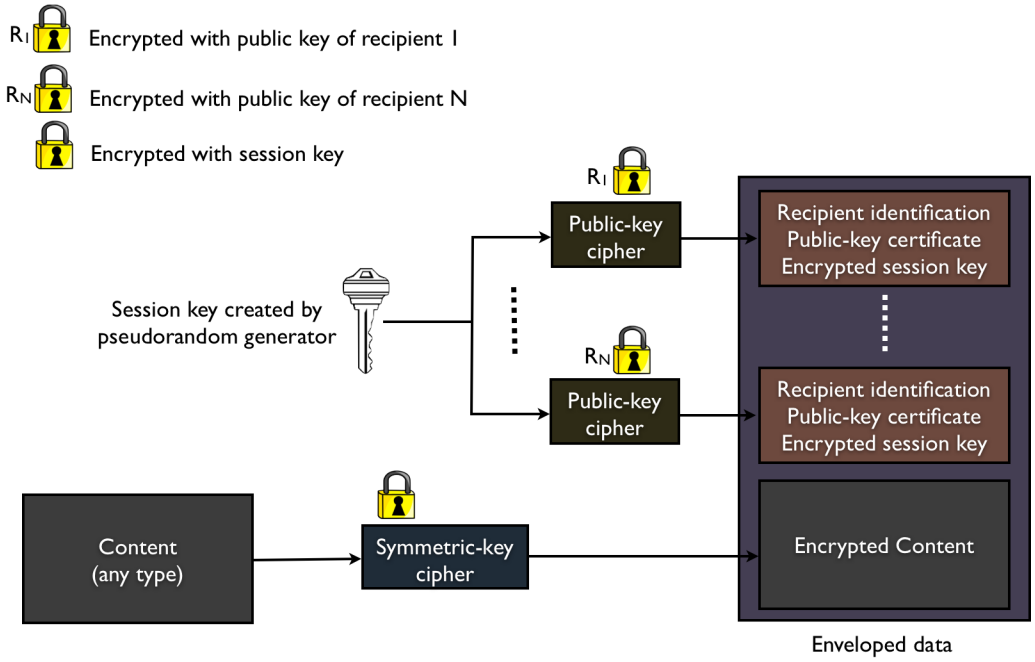

Figure 3: Creation of enveloped data

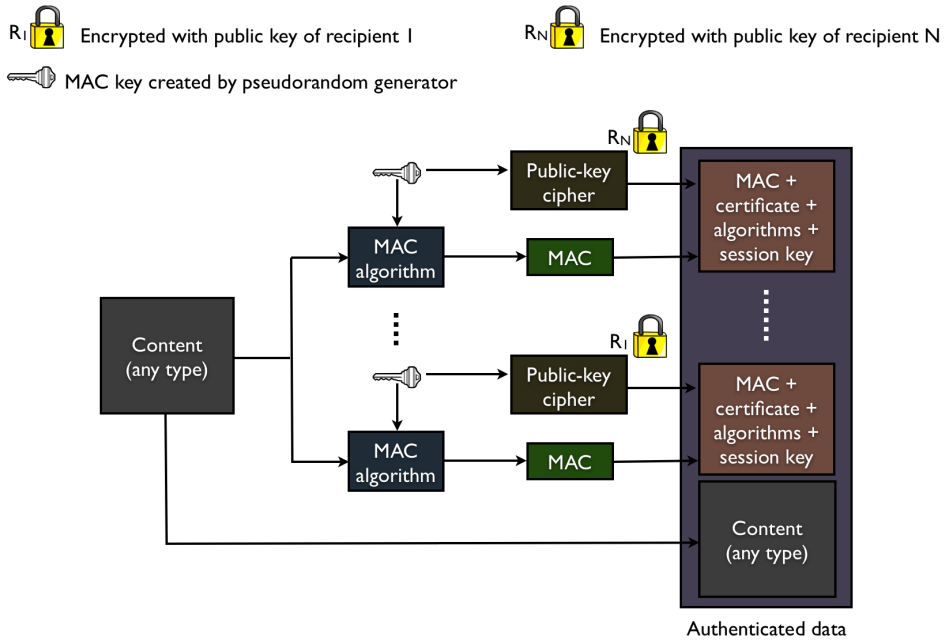

Figure 4: Creation of authenticated data

Algorithm	**Sender must support**	**Receiver must support**	**Sender should support**	**Receiver should support**
Content encryption	Triple DES	Triple DES		1.AES 2. RC2/40
Session-key encryption	RSA	RSA	Diffie--Hellman	Diffie--Hellman
Hash	SHA-1	SHA-1		MD5
Digest encryption	DSS	DSS	RSA	RSA
Message authentication		HMAC with SHA-1		

Table 3: Cryptographic Algorithms for S/MIME [8]

Type	Sub-type	Parameter	Description
Multipart	Signed		a clear signed message in two parts: the message and the signature.
Application	pkcs7-mime	signedData	a signed S/MIME entity
	pkcs7-mime	envelopedData	encrypted S/MIME entity
	pkcs7-mime	degenerate signedData	an entity containing only public key certificates
	pkcs7-mime	compressedData	a compressed S/MIME entity
	pkcs7-signature	signedData	the content type of the signature subpart of a multipart/signed message

Table 4: S/MIME Content Types [39]

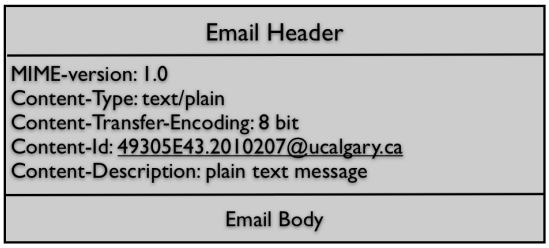

Table 5: MIME Header

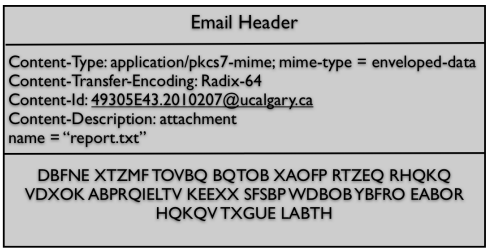

Table 6: S/MIME Header

defined by X.509. Unlike PEM, S/MIME does not require a single trusted root among the CAs. Some S/MIME enabled email clients support self-signed certificates though their use is discouraged [10, 12]. However, the user is responsible for maintaining the web of trust to verify signatures as defined by PGP [8]. In general, for *key establishment*, a user needs to get his or her pair of public and private keys certified by trusted CAs such as Verisign, Thawte, and Globalsign. S/MIME version 3.0 mandates *Digital ID*, which is a digital certificate that includes the following at the minimum [39]: (a) owner's public key, (b) owners name or alias, (c) expiration date of the Digital ID, (d) serial number of the Digital ID, (e) name of the CA that issued the Digital ID, and (f) digital signature of the CA that issued the Digital ID. Moreover, a Digital ID may also contain other user-supplied information, including address, email address, basic registration information such as country, zip code, age, gender, etc.

Most S/MIME enabled email clients include in them certificates of well-known CAs. S/MIME standard automates a rudimentary form of *key distribution* [11]: when a digitally signed message is sent, it is accompanied by a copy of the public key certificate of the sender, which can be used to verify the message. The certificate is copied from the message into the recipients address book. In fact, peer certificates are obtained by S/MIME applications

in any of the following three ways [12]:

i Extracting certificates from incoming signed messages from peers

ii Loading certificates from *.p7c files

iii Lookup of peer certificates from a LDAP (Lightweight Directory Access Protocol) Repository

4 S/MIME Capabilities

Clearly, S/MIME is capable of providing privacy in emails by enveloped messages, and authenticity by signed messages. Signed messages also ensure non-repudiation of sender. Message integrity can be verified using message digest. The use of signed receipts is claimed to provide non-repudiation of the recipient, though this claim is not beyond question [4, 33]. However, "strong fairness" as mentioned in section 2 is yet to achieve.

Nikita Borisov and et. al. [4] argues that besides authenticity, repudiation of sender may also be desired in situations. For instance, Alice may want to send a message to Bob, which Bob needs to verify for authenticity. But Alice wants to make sure that Bob will not be able to prove to a third person that the message originated from Alice. S/MIME's 'authenticatedData' (see section 3.2.1) is capable to satisfy such requirements, as well.

S/MIME supports both 'opaque' and 'clear' (multipart) signing format. In case of clear signed format, only the digital signature is encoded using radix-64 [39]. Consequently, the recipients without S/MIME capability can still view the message, although they cannot verify the signature [12].

5 Limitations of S/MIME

This section points out some security issues and limitations of S/MIME and its implementations.

Non-repudiation of recipient is claimed to have achieved by the use of signed receipt. However, this assumes the recipient to be a fair participant in the sense that the recipient returns a signed receipt, if the sender asks for it. So, non-repudiation in this case is completely dependent on the recipient's will. Rolf Oppliger [33] argues, "this assumption is somewhat difficult, for if one could assume fair participants, then there would be no need for receipt in the first place".

Multi-recipient message support in S/MIME is not efficient enough. To send a private message to multiple recipients, the sender has to produce different envelopes for different recipients. Then all envelopes are combined in a stack and delivered to each of the recipients. The recipient has to seek from the stack the envelope intended

for him, and then decrypt it to read the message. This approach of S/MIME incurs *computation cost* at the sender's end linear to the number of recipients of the message [42]. The *length of the message* to transfer to each recipient over communication channel is also linear to the number of recipients. Moreover, the need for the recipients decrypting their own envelopes ruins the important feature of *carbon-copy* [42]. Carbon-copy is used not only to save from recomposing the same message, but also used as a means to imply that exactly the same message is equally delivered to each of the recipients.

Partial content signature is not supported by S/MIME. But such type of flexibility may be desired in many cases [25]. For example, when a computer sales company sends a digital invoice to the customer, it may be more sensible if the technical department signs the technical configuration part, sales department signs the total price and delivery address, whereas the shipping department signs the delivery date.

Email header protection provided by S/MIME is not sufficient. RFC 2633 [35] clearly declares that S/MIME version 3 is to secure the email content, not the header. As S/MIME version 3 provides protection for the email content only. The content does not contain the header, and so the subject, to, from (the name), cc, and date fields of the header can be altered without affecting the verifiability of existing signature over the message content [24, 25]. Alteration of only the email address of the from field would make the the signature non-verifyable because of S/MIME's email address crosscheck [24]. S/MIME 3.1 (RFC 3851 [37]) provides an optional mechanism for header protection to some extent. In this method the actual full message (including the header) is encapsulated into a single message/rfc822 MIME type object, and the S/MIME signature is applied to it. This signed message/rfc822 object is attached to another email message, which is sent to the recipient. In this case, the header of the actual message becomes protected but the outer header remains unprotected [24, 25]. Moreover, this makes it complicated for the email client in the recipient's end to determine how to present to the user the inner and outer headers. It also becomes difficult to determine whether the message within the message/rfc822 wrapper is the top level message, or the complete message/rfc822 MIME entity is another encapsulated message [25]. Although RFC 3851 recommends the email clients to display the encapsulated message as the only message, this does not comply with the current email related IETF standards and most email client implementations [24].

Possible use of bogus name for sender is not prevented in S/MIME, due to the fact that class-1 certificates do not contain validated names [23, 24], and email clients allow any name for sender while sending an email message. Moreover, the S/MIME version 3 signature verification process (RFC 2632 [34]) does not mandate a comparison between the name in the certificate and the name in the email header. The only connection between a certificate and an email message is the email address [24]. So,

it is possible to send a signed email message using the email address existing in the certificate, but with a different name.

Email storage in encrypted form with the original encryption key is a design flaw of all current S/MIME client implementations [9, 10]. S/MIME enabled email clients store emails as they receive emails in the form of encrypted with the recipient's public key, which can be decrypted with the corresponding private key only. But if the recipient's private key gets lost, or expires and he revokes new key then the stored emails become inaccessible.

6 Usability Barriers

Today S/MIME technology is widely deployed but email encryption is rarely used [9, 11]. A major reason is usability obstacles [9]. S/MIME standard and its implementations incur a number of usability barriers, which hinder it being adopted by email users. One can send S/MIME secured email only if the recipient has a public key [21]. Moreover, the additional effort needed for obtaining Digital ID from established CA is an obstacle, which pushes users away from using secure email systems [11]. People's names are used in Digital IDs, and it is impractical to expect globally unique names of people. This restricts Digital IDs good enough for use in small communities only [7].

In general, users of secure systems are recommended to change their password or PIN (Personal Identification Number) periodically to save from accidental compromise. *Certificate revocation* involves the time-consuming process of public key distribution, which needs much effort. This certificate revocation problem is often mistakenly ignored [7, 33].

Interoperability of S/MIME enabled email clients used across different users is important to enable users conveniently exchange S/MIME secured email messages. RSA Data Security Inc. established in 1997 the S/MIME Interoperability Test Center, which allows vendors get their email clients undergo S/MIME interoperability testing, and have the results published [12]. A number of factors found to have affected interoperability of S/MIME enabled email clients [12], for example,

- Currently different email clients provide implementation of different versions (2, 3.0, or 3.1) of S/MIME.
- Some email clients support the use of implicit trust model using self-signed certificates, while others don't.
- Some implementations support both opaque and clear (multipart) signing format, while others support any one of them.
- Different email clients support different subsets of encryption and hash algorithms.
- Different email clients support variable RSA modulus and key size (512, or 768, or 1024, or 2048 bit).

- Some email clients support X.509v3 certificate path validation for parsing complex certificate chains to establish trust in peer certificates. However, many implementations do not support the validation of certificates that are part of a multilevel hierarchy.

All S/MIME enabled email clients maintain some sort of repository for user certificates and keys. Such repositories are tightly coupled with the corresponding computing devices. As AOL's client and webmails do not support S/MIME [10], mobile users who do not carry their personal computing devices such as Laptops or PDAs cannot exchange S/MIME secured emails via webmails [10]. In order to facilitate such facility certificates and keys must be available on all the computing devices that may be used for email exchange. This *key availability and migration* appears to be challenging to implement [10].

A very small portion of email users have the cryptographic concepts about keys, certificates, encryption, and decryption. Therefore, S/MIME implementation needs to provide well designed *graphical user interface (GUI)* capable of hiding the cryptographic complexities from the general users by automating most of the key management, signature verification, encryption and decryption process. However, careful analysis [9, 28] and usability tests [10, 11, 24, 43] indicate that usable GUI implementation for email security has been still a big challenge to achieve. "Until more usable mechanisms are integrated into popular email clients, signatures using S/MIME should remain in the domain of power users" [21].

7 Conclusion

Technically S/MIME provides a good level of security in email exchange. However, it is not bulletproof. The identity management issues and low usability in the interfaces of S/MIME enabled email clients pose significant barrier against its adoption in the community [11, 24]. key management activities such as key establishment, sharing, revocation, and migration are still challenging. *Key Continuity Management (KCM)* [11, 13] was proposed to relax the stringent identity certification rules of S/MIME and to automate the process of key generation, key management, and message signing. However, "real security is hard work. There is no cure-all, specially not PKI" [7].

Security mechanisms are effective only when those are used correctly [43]. S/MIME enabled email clients need usable interfaces for allowing an average user to properly apply the S/MIME security mechanisms. Current implementations lack this level of usability, which is a significant obstacle to adoption [11, 24]. Attractive graphical user interface does not ensure usability of secure systems like S/MIME email clients. Effective security requires a different usability standard [43]. It is often believed that security and usability are two antagonistic goals in system design [9], which makes it more difficult to design usable secure systems.

Moreover, in case of S/MIME, "ease of use is not just having a better GUI, it's also dealing with the user's inability to grasp abstractions of public-key infrastructures. Non-

technical users don't want to learn key certification and certificate authorities. It's the conceptual hurdles that are stumbling the blocks, not the quality of the graphical user interface" [29]. Therefore, mass users' gaining the understanding on public key cryptography is necessary to leverage the adoption of S/MIME for secure email exchange.

References

[1] D. Balenson. RFC 1423: Privacy enhancement for internet electronic mail: Part III: Algorithms, modes, and identifiers. February 1993.

[2] R. Blum. *Open Source Email Security*. Sams Publishing, USA, 2002.

[3] N. Borenstein and N. Freed. RFC 1522: MIME (multipurpose internet mail extensions) Part One: Mechanisms for specifying and describing the format of internet message bodies. 1993.

[4] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84, New York, NY, USA, 2004. ACM.

[5] D. Chadwick, A. Young, and N. Cicovic. Merging and extending the PGP and PEM trust models-the ICE-TEL trust model. *Network, IEEE*, 11(3):16–24, May/Jun 1997.

[6] D. Crocker. RFC 822: Standard for the format of ARPA internet text message. 1982.

[7] C. Ellison and B. Schneier. Inside risks: risks of PKI: secure email. *Commun. ACM*, 43(1):160, 2000.

[8] B. A. Forouzan. *Cryptography and Network Security*. McGraw-Hill, New York, USA, 2008.

[9] S. L. Garfinkel. Enabling email confidentiality through the use of opportunistic encryption. In *dg.o '03: Proceedings of the 2003 annual national conference on Digital government research*, pages 1–4. Digital Government Society of North America, 2003.

[10] S. L. Garfinkel, D. Margrave, J. I. Schiller, E. Nordlander, and R. C. Miller. How to make secure email easier to use. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 701–710, New York, NY, USA, 2005. ACM.

[11] S. L. Garfinkel and R. C. Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, pages 13–24, New York, NY, USA, 2005. ACM.

[12] S. Gupta, J. Mulvenna, S. Ganta, L. Keys, and D. Walters. Interoperability characteristics of S/MIME products. In *Proceedings of the International Exhibition and Congress on Secure Networking - CQRE (Secure) '99*, pages 229–241, London, UK, 1999. Springer-Verlag.

[13] P. Gutmann. Why isn't internet secure yet, dammit. In *AusCERT Asia Pacific Information Technology Security Conference 2004; Computer Security: Are we there yet?*, Gold Coast, Australia, May 2004. AusCERT.

[14] L. Harn and J. Ren. Design of fully deniable authentication service for e-mail applications. *Communications Letters, IEEE*, 12(3):219–221, March 2008.

[15] K. Henry. Getting started with PGP. *Crossroads*, 6(5):8, 2000.

[16] P. Hoffman. RFC 2634: Enhanced security services for S/MIME. June 1999.

[17] R. Housley. RFC 3370: Cryptographic message syntax (CMS) algorithms. August 2002.

[18] R. Housley. RFC 3852: Cryptographic message syntax (CMS). July 2004.

[19] R. Housley. RFC 3852:cryptographic message syntax (CMS) multiple signer clarification. April 2007.

[20] B. Kaliski. RFC 1424: Privacy enhancement for internet electronic mail: Part IV: Key certication and related services. February 1993.

[21] A. Kapadia. A case (study) for usability in secure email communication. *Security & Privacy, IEEE*, 5(2):80–84, March-April 2007.

[22] S. Kent. RFC 1422: Privacy enhancement for internet electronic mail: Part II: Certificate-based key management. February 1993.

[23] A. Levi and Çetin Kaya Koç. Inside risks: Risks in email security. *Commun. ACM*, 44(8):112, 2001.

[24] A. Levi and C. B. Güder. Understanding the limitations of S/MIME digital signatures for e-mails: A GUI based approach. *Comput. Secur. (2008)*, 2008.

[25] L. Liao and J. Schwenk. Secure emails in XML format using web services. *Web Services, 2007. ECOWS '07. Fifth European Conference on*, pages 129–136, Nov. 2007.

[26] J. Linn. RFC 989: Privacy enhancement for internet electronic mail: Part I: Message encipherment and authentication procedures. February 1987.

[27] J. Linn. RFC 1421: Privacy enhancement for internet electronic mail: Part I: Message encryption and authentication procedures. February 1993.

[28] C. Masone and S. Smith. Towards usefully secure email. *Technology and Society Magazine, IEEE*, 26(1):25–34, Spring 2007.

[29] L. McLaughlin. Philip Zimmermann on what's next after PGP. *IEEE Security and Privacy*, 4(1):10, 2006.

[30] S. Mendez and C. Huitema. A new approach to the X.509 framework: Allowing a global authentication infrastructure without a global trust model. *IEEE*, pages 172–189, 1995.

[31] R. A. Mollin. *An Introduction to Cryptography, 2nd Edition*. Chapman & Hall/CRC, Boca Raton, Fl, USA, 2007.

[32] A. Nenadić, N. Zhang, and S. Barton. Fair certified e-mail delivery. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 391–396, New York, NY, USA, 2004. ACM.

[33] R. Oppliger. Certified mail: the next challenge for secure messaging. *Commun. ACM*, 47(8):75–79, 2004.

[34] B. Ramsdell. RFC 2632: S/MIME version 3 certificate handling. June 1999.

[35] B. Ramsdell. RFC 2633: S/MIME version 3 message specification. June 1999.

[36] B. Ramsdell. RFC 3850: S/MIME version 3.1 certificate handling. July 2004.

[37] B. Ramsdell. RFC 3851: S/MIME version 3.1 message specification. July 2004.

[38] E. Rescorla. RFC 2631: Diffie-Hellman key agreement method. June 1999.

[39] W. Stallings. *Cryptography and Network Security: Principles and Practices, 4th Edition*. Prentice Hall, USA, 2006.

[40] H. F. Tipton and M. Krause. *Information Security Management Handbook, Fifth Edition*. CRC Press, Boca Raton, Fl, USA, 1997.

[41] H. C. van Tilborg. *Encyclopedia of Cryptography and Security*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[42] W. Wei, X. Ding, and K. Chen. Multiplex encryption: A practical approach to encrypting multi-recipient emails. In *ICICS*, pages 269–279, 2005.

[43] A. Whitten and J. D. Tygar. Why johnny can't encrypt: a usability evaluation of PGP 5.0. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

[44] K. Yamamoto. An integration of PGP and MIME. *Proceedings of the Symposium on Network and Distributed System Security*, pages 17–24, Feb 1996.

[45] P. R. Zimmermann. PGP user's guide, volume 1, essential topics. *available free with PGP software.*