Enabling Email Confidentiality through the use of Opportunistic Encryption

Simson L. Garfinkel <simsong@lcs.mit.edu>
http://www.simson.net/
MIT Laboratory for Computer Science, NE43-536, Cambridge, MA 02139

ABSTRACT

Software for encrypting email messages has been widely available for more than 15 years, but the email-using public has failed to adopt secure messaging. This failure can be explained through a combination of technical, community, and usability factors. This paper proposes a new approach to email security that employs opportunistic encryption and a security proxy to facilitate the opportunistic exchange of keys and encryption of electronic mail. While it appears that this approach offers less security than established systems that employ certificates, the security is in fact equivalent to today's systems based on PGP, PGP/MIME and S/MIME.

1. Background

It is widely believed that security and usability are two antagonistic goals in system design. A classic example of this argument is the failure of Internet users to adopt secure email systems such as PGP![14] and S/MIME![2]. Despite the obvious security problems arising from unencrypted email, the argument goes, users do not encrypt the vast majority of their electronic correspondence because the added security results in a system that is inherently too difficult to use.

Certainly, most Internet users have failed to adopt encrypted email. This failure persists despite the widespread availability of free email encryption software since 1991[14], the direct support of email encryption algorithms in popular email client software [6][7], the publishing of popular documentation and training materials (e.g., [4][8]) to teach users the importance and techniques for sending encrypted mail, and ongoing efforts within the Internet Engineering Task Force to standardize formats for email encryption dating back to 1987![2][3][5]. Moreover, the difficulty of the popular PGP encryption system has been well-documented [13].

To encrypt or not to encrypt is the question that today's email clients present their users. Given this choice, most users chose the "zero-click" alternative—not to encrypt. The perceived risks that encryption protects against presumably pale when compared to the difficulty of encrypting

But security need not be complicated. The popular SSL/TLS encryption protocol![1] provides ubiquitous and reasonably transparent encryption on many websites. Users of eBay and Amazon.com do not make a conscious decision to encrypt their account passwords before sending them over the Internet: the passwords are encrypted by default. Meanwhile, unencrypted TELNET has been largely supplanted by the cryptographically secure SSH remote access protocol for Unix servers![11]. Rather than being antagonistic, usability and security are synergistic: secure systems that are usable enjoy great success, and those that are not are simply avoided.![9]

2. The Need for Encrypted Email Today

Encryption provides two primary functions for email. *Signing* attaches to the email a digital signature. This signature can to verify the authenticity of the message sender and to detect message tampering. *Sealing* scrambles the content of a message so that it cannot be deciphered by anyone other than the intended recipient.

Figure 1 traces the path followed by many mail messages on the Internet today. Although all unencrypted communications moving across the Internet are vulnerable to interception and analysis, the volume of Internet traffic and the relative inaccessibility of these communication links makes such attacks relatively difficult. On the other hand, any message from a given Sender (S) to a Recipient (R) will usually travel through the Sender's SMTP Server (SS) and then be spooled for final delivery in a mailbox

on the Recipient's POP Server (RS). These computers provide ideal locations for targeted monitoring or message modification.

There is no reason to use encryption unless there is a risk of sender spoofing, unauthorized message adulteration, or unauthorized message interception. These risks are real: accounts of email interception can be found in many popular works, including [10] and [12].

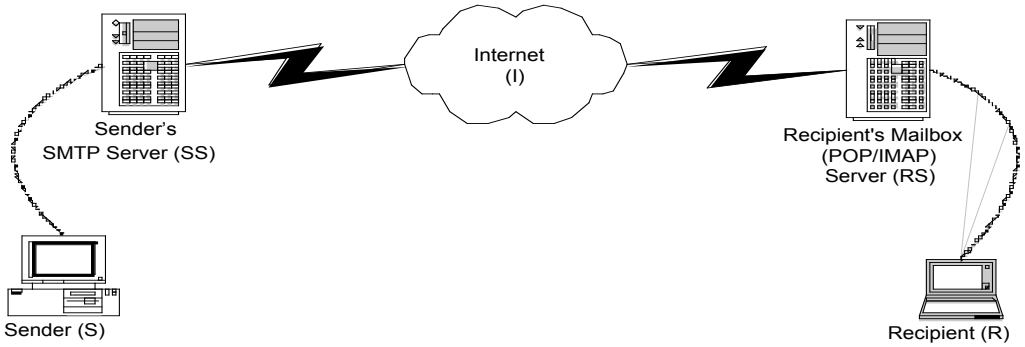

Figure 1 Internet Email is vulnerable to interception at many points

Another risk not shown in Figure 1 is that a hostile third party will send a message to R that appears to come from the S, but in fact does not. A special case occurs when the third party sends mail to R that appears to come from R herself! Many senders of unsolicited commercial email (i.e. *spam*) use such techniques in an attempt to bypass anti-spam filters.

Despite the risks, email encryption remains relatively rare. We believe that the reason is systematic *usability barriers.* Consider, for S to send an encrypted message to R, S must:

1. Determine if R in fact desires to receive encrypted messages and can decrypt them.
2. Determine the email encryption system that R is using.
3. Obtain a copy of R's public key (K_R), being careful to verify that the key in fact belongs to R (and not to some other Internet user with a similar name or email address).
4. Verify that R still possesses the corresponding private key for K_R.
5. Load K_R into either an email client or special-purpose encryption program.
6. Create a private/public key pair (K_S) for signing the message, if one does not already exist.
7. Compose the message.
8. Sign the message with K_S and encrypt it with K_R.
9. Send the message.

To decrypt this message and verify the sender, R must:

1. Provide the encrypted message as input to a suitable decryption program.
2. If R's private key is encrypted, R must enter a pass phrase to decrypt it.
3. If S's signature is to be verified, R must obtain S's public key to verify the message.
4. View the decrypted/verified message.

Additional barriers exist that prevent S from even systematically signing all outgoing email messages as a matter of course, thanks to the way that programs such as Microsoft Outlook and Outlook Express handle signed messages received in PGP/MIME or S/MIME format. Microsoft's programs frequently display a signed message as a blank message with two attachments: one for the "signed" message and one for the signature. Viewing such messages can be quite cumbersome. Outlook and Outlook Express may also generate a warning if the sender's key is not registered or the key's Certificate Authority is not trusted. In either of these cases, the Microsoft programs force the user to acknowledge a security warning before the message content is displayed. As a result, some people who receive messages that are merely signed will ask the sender to stop sending digitally signed messages because they are annoying to the recipient.

3. A Zero-Click Interface for Encrypted Email

Stream is an email encryption system designed to overcome the usability problems that have heretofore plagued the world of cryptographically-protected Internet email through the use of a zero-click interface. Stream operates as a filter on outgoing email messages through the use of an SMTP proxy, and on incoming email messages through the use of a POP proxy.

As an outgoing filter, Stream automatically performs these actions for each outgoing message (M):

1. Determines the sender's email address E.
2. Creates a public/private key pair for address E (K_E) if one does not exist.
3. Places a copy of K_E in M's message header.
4. Evaluates the recipients ($R_{1..n}$) of message M. If a public key (K_R) for R is known, Stream:
    a. Encapsulates M's original mail header within message M.
    b. Adds to this encapsulated header the key fingerprint for each recipient's encryption key.
    c. Creates a new *sanitized mail header* for message M containing a single To: address and a nondescript Subject: line.
    d. Encrypts M for the recipient and sends the message out through SMTP server SS.
5. Finally, the original message is sent to any recipients for whom K_R is not known, if there are any.

Stream provides *opportunistic encryption*: if the email message can be encrypted, it is. If it cannot be encrypted, it is sent without encryption. We believe that this behavior mimics the behavior of existing users of encryption: they use it if they can, but if they can't, they send their message anyway.

As an incoming filter, Stream automatically performs these actions on each incoming message (M):

1. Determines if an encryption key is present in the mail header.
    a. If so, the key is added to the user's key database.
    b. If this is a new key for an existing email address in the database, the user is warned through an additional email message alerting them to the fact. This warning is similar to the warning that SSH generates when a server's public key is changed.
2. If the message is encrypted:
    a. Stream decrypts the message.
    b. Unencapsulates the encapsulated mail headers.
    c. If key fingerprints were present, Stream verifies that the fingerprints on the encapsulated message headers match those for the copies of the keys in the key database.
    d. If the fingerprints do not match, a warning is sent to the recipient.
    e. Finally, Stream inserts a special character (currently the plus sign) at the beginning of the Subject: line.

Stream treats this special character (+) as the *mandatory encryption character*. If this character is present at the beginning of a Subject: line (ignoring any number of repeating "Re:" sequences), Stream will omit outgoing step 5 — that is, it will not transmit an unencrypted message. In this manner, a reply to an encrypted message is always encrypted unless a user takes specific actions to the contrary.

4. Analysis of Usability, Security, and Related Work

Considerable complexity in existing email encryption systems comes from issues of key management and the perceived need to verify the identity of public key holders. For example, [4] devotes one chapter each to email signing and sealing, but five chapters to key management issues. By eliminating explicit key management from the user experience, the task of sending and receiving encrypted email is dramatically simplified.

PGP's "Web of Trust" allows "trusted introducers" to certify the authenticity of email addresses and human-readable names associated with keys. Although technically elegant, PGP's approach ignores the fact that most electronic introductions take place when the introducer sends an email message to the other two participants. Stream's use of signed, sealed and encapsulated message headers and key fingerprints

gives this introduction protocol the security of PGP's original key signing approach. Furthermore, the tendency of PGP users to upload keys to many signatures on them to PGP key servers makes both the key holders and key signers vulnerable to network analysis; such practices are discouraged by Stream.

S/MIME's use of certificate authorities provides for the verification of legal names when an individual's identity is appropriately verified prior to the certificate's issuance. In our experience, most correspondents are more concerned with the *continuity of identification* than absolute identity. That is, people wish to know that their correspondent today is the same as yesterday; if needed, they rely on procedures other than certificate validation to determine the legal identity of their correspondents.

One potential criticism of Stream is that it cannot detect a perfect man-in-the-middle attack. That is, if *every* outgoing message from S is intercepted by an attacker and modified, R will never detect this fact. However, the same is true of all other cryptographic systems: The man-in-the-middle could just as easily modify the public keys stored in the downloaded copy of Outlook Express or Netscape Navigator. If an out-of-band mechanism is used to install software or download keys for a Certificate Authority, that method can be used to download Stream's initial email message as well.

We are unaware of any other system for embedding email cryptographic keys in mail headers, although some Netnews software signs Usenet mail headers with the "x-pgp-sig:" header. While this system distributes signatures for message headers, it does not distribute keys or seal message content.

Finally, while most PGP and S/MIME implementations leave messages encrypted except when being viewed, our system removes cryptographic protection once a message reaches its final destination. We believe that providing special security for email is not appropriate given the large amount of other confidential information on a typical desktop or laptop computer. A better approach is to provide a unified security system for all of a user's confidential documents.

5. References and Acknowledgements

[1] Dierks, T., "RFC 2246: The TLS Protocol," January 1999. http://www.ietf.org/rfc/rfc2246.txt

[2] Dusse, S., et al, "RFC 2311: S/MIME Version 2 Message Specification," March 1998. http://www.ietf.org/rfc/rfc2311.txt

[3] Elkins, M., et al, "RFC 2015: MIME Security with Pretty Good Privacy (PGP)," October 1996. http://www.ietf.org/rfc/rfc2015.txt

[4] Garfinkel, Simson. PGP: Pretty Good Privacy, O'Reilly & Associates, January 1995

[5] Linn, John, "RFC 989: Privacy Enhancement for Internet Electronic Mail" February 1987. http://www.ietf.org/rfc/rfc989.txt

[6] Microsoft Corp, "Outlook Express," Versions 3.0 through 6.0, 1996—2003

[7] Netscape Corp., "Netscape Communicator," Version 4.0 through 6.0, 1996—2003

[8] Oppliger, Rolf, *Secure Messaging with PGP and S/MIME*, 15 December 2000, Artech House.

[9] Saltzer , J. and Schroeder, M. "The Protection of Information in Computer Systems," *Proc. IEEE*, vol. 63, no. 9, 1975, pp. 1278–1308.

[10] Shimomura, Tsutomu, and Markoff, John, Takedown: The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaw—By the Man Who Did It, Hyperion, February 1996.

[11] SSH Communications, http://www.ssh.com/

[12] Stoll, Clifford, The Cuckoo's Egg: Tracking a Spy Through the maze of Computer Espionage, Pocket Books, October 2000.

[13] Whitten, Alma and Tygar, J.D. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," in Proceedings of the 8th USENIX Security Symposium, August 1999. http://citeseer.nj.nec.com/whitten99why.html

[14] Zimmermann, Phil, "Pretty Good™ Privacy – RSA public key cryptography for the masses," 5 June 1991, PGP 1.0 Beta test version

Previous versions of this paper were reviewed by Jean Camp, David Clark, Kevin Fu, Rob Miller, Ronald Rivest and Jerome Saltzer. I have greatly benefited from their input, advice and ideas.