Why Don't We Encrypt Our Email?

Stephen Farrell • *Trinity College Dublin*

In this installment of Practical Security, I briefly review the security features of common mail user agents (MUAs) and then consider what a user might actually need for email security. We fairly quickly see a mismatch between the implemented features and user requirements, which could explain why so few of us use any of these ubiquitously deployed security features.

In most cases, I describe a situation in which the user has a local MUA — for example, Mozilla Thunderbird (www.mozilla.com/en-US/thunderbird) or Microsoft Outlook (www.microsoft.com/outlook) — and isn't using a Web-based mail client (in which case, the MUA is actually running on the Web server).

Existing Mail Security Features

We can discuss current email security features under different headings: local filtering, local protocol security, domain-applied security, and, lastly, the main topic that I'll discuss here: end-to-end security, meaning security services that a message author applies and that a message recipient checks.

MUAs carry out various local filtering actions on the user's behalf — for example, running antivirus tools or antispam filters to reduce the amount of time the user spends dealing with unwanted messages. Many people use these security features, although such techniques can be somewhat problematic — for example, if the filter mistakenly quarantines an incoming message, it can be hard to find.

Locally, MUAs often set up secure links to mail servers for submission, usually via Simple Mail Transfer Protocol (SMTP), and delivery, usually via Internet Message Access Protocol (IMAP) or Post Office Protocol, version 3 (POP3). Although users do employ these security features, generally they have little control over the security settings for their MUA-to-server connections, so they simply do whatever their service provider tells them to do to get their mail working.

At the domain level, inbound or outbound mail transfer agents (MTAs) can carry out several security tasks, mainly related to the (attempted) control of spam. For example, an MTA might scan incoming messages for malware or digitally sign outgoing messages using the Domain Keys Identified Mail (DKIM) protocol.[1] However, users don't see most of these features, so they aren't that relevant to this discussion.

From the end-to-end perspective, users can apply encryption and origin authentication services to their messages. Before we consider why users don't use these features, I'll briefly outline how they work. Mail encryption is fairly easy to understand — once users have a recipient's cryptographic key, they can choose to encrypt a message to that recipient so that no one except the intended recipient (including no en route mail administrators) can see the message's content. This is a service that sounds like it should be attractive — essentially, it's the difference between a postcard (most email) and a closed letter (encrypted mail). However, note that I've described encryption as being easy after some key management has happened — more on that later.

Origin authentication is somewhat harder to explain but basically amounts to adding a digital signature to the message so that a recipient who verifies the signature can be confident that the actual message's author is the claimed message author — that is, the recipient gets evidence that the message isn't a forgery. Detecting forgeries sounds useful, too, though perhaps somewhat less useful than the ability to keep a message secret. Extending the snail-mail analogy, origin authentication is somewhat like

 1089-7801/09/$25.00 © 2009 IEEE

registered mail, at least in terms of being able to identify the message's source. There are two standard cryptographic schemes for applying these services: secure Multipurpose Internet Mail Extensions (S/MIME)[2] and OpenPGP,[3] both of which can apply encryption, origin authentication, or both to any outgoing message. S/MIME is the form of end-to-end mail security that is built in to most MUAs, so support for S/MIME is almost ubiquitous. OpenPGP is generally available as a plug-in for most MUAs and so is also widely available, although it's nowhere near as widely deployed as the S/MIME functionality. In addition, almost all MUAs have interoperated for at least a basic set of S/MIME and OpenPGP uses (www.imc.org). In other words, these protocols are deployed, and they work. Yet they remain unused.

User Requirements for Email Security

The fact that people don't use these security services might mean that users don't find the set of services useful. So we might start by asking, "What do users actually want from email security?" Well, it's hard to know what people want because it appears that the industry hasn't really asked them. In fact, it seems to have, for at least the past decade and a half, worked from the same set of assumptions that produced early secure email efforts (privacy enhanced mail [PEM][4]) and has simply replicated those security services through each iteration of email security specifications.

PEM was essentially driven by enterprise- and government-networking requirements, as laid down by security experts, and not, as far as I'm aware, by user requirements. It's no wonder that users didn't adopt PEM (although there were other reasons, not least the lack of MIME support due to some unfortunate timing in producing the PEM and MIME specifications) — a failure that seems to be repeated each time a new MUA deploys the same old set of security services. Maybe it's now time to actually ask some users what they want and, more importantly, how much inconvenience they're willing to suffer to get what they want.

In the area of medical communication, users do seem to want secure messaging and would increasingly use it the more they trust the medical and technology service providers involved.[5] However, the same study also notes that the medical service providers considered actually use proprietary Web-based messaging rather than standard email. Presumably, this is partly because no one does, or can, use the standard MUA security features. Richard Klein also states that perceived ease of use can affect perceived usability, which certainly resonates with current work on usable security in general, although the focus of usability has so far mainly been on Web interactions rather than end-to-end email security features.[5]

Another security researcher has documented his own requirements in the sense that he's tracked and commented on his experiences using MUA security services for several months.[6] He concludes, sensibly enough, that origin authentication isn't very important at all because he doesn't really react to whether or not mail he receives has an accompanying digital signature. Rather, he treats all mail in a wider context — for example, whether he knows the apparent sender and whether the content is appropriate for that apparent sender. Such contextual evaluation is actually something that users are becoming better at because the constant bombardment of spam is quite an efficient way to train humans (at least in the long run).

Context use also brings up the awkward fact that S/MIME and OpenPGP security services apply only to the message body and not at all to message headers — the fact that the digital signature doesn't cover an origin-authenticated message's subject line isn't something that users will easily understand — users don't, and shouldn't, need to know the difference between RFCs 5322[7] and 5321.[8] The same criticism also applies to the use of S/MIME and OpenPGP confidentiality — most users would expect the service to encrypt the subject line when sending an encrypted message, but this isn't what happens. In any case, because we do require message confidentiality, we must then ask if our current local protocol security services meet our needs. Again, it's hard to know how users actually perceive this security — do they think that because they access and send a message securely, it remains secure all the way to the recipient? Or perhaps, more likely, they don't really have strong confidentiality requirements for most messages that they send. After all, knowing that people will meet at the pub at 6 p.m. on Friday is hardly private information that people wouldn't guess on their own.

The question of confidentiality brings up two real mail requirements that a useful and usable secure mail service must meet — users must be able to easily control the level of security applied to a given message, and it must also work when the message has multiple recipients, regardless of how we ultimately implement security. Because email works that way, secure email must also, or else it's a misnomer. However, at present, I believe we simply don't know how to provide a general but usable multirecipient confidentiality service for email.

How to handle cases in which one recipient is a list agent and not a human at all might also be difficult. I'm not at all sure what transitivity requirements might really exist for mail that's forwarded through such mailing lists — should the service

also encrypt forwarded mail, or can the service send it on without encryption because it's already been delivered to the list agent? Neither answer seems quite right, the former because the message's originator is no longer in the loop (so why bother), the latter because of the obvious loss of confidentiality. Perhaps a better answer would be to include the originator's confidentiality requirements with the message itself — that way, the originator could at least influence the decision at the list agent, without forcing software developers to make that decision.

Indeed, this line of reasoning leads toward a content-security model in which secured messages carry with them the rules that their creators want enforced as the security service processes the messages in the network. Although this might be disturbingly like a digital-rights-management (DRM) scenario, the fact that anyone in the network is a valid source of messages-with-rules might make it sufficiently different to be less threatening. Time will tell as people who are interested in content-based networking[9] start to work on generic security models.

It's hard to know how users perceive security — do they think that because they access and send a message securely, it remains secure all the way to the recipient?

A Small Change

But if we restrict ourselves to interpersonal messaging with only one recipient for now, can we maybe employ the S/MIME technology that's already deployed? If we assume that our network takes most of the messages that we want to protect and securely moves them from one (human) sender to one (human) recipient, then perhaps we can make some progress with a few relatively small MUA changes.

Although the existing public-key-infrastructure- (PKI-) based model for S/MIME can work well for enterprise users in which an administrator generates and certifies keys, this model doesn't work for casual users — expecting users to know that they must generate key pairs before anyone can encrypt messages to them is simply unrealistic. (The same is essentially true of OpenPGP — people have used it successfully in various small communities over the years, but it can't really scale to casual Internet-wide use.) Therefore, current MUAs should first be able to automatically generate and self-sign a key pair whenever a user adds a new mail account locally. The user can always choose to replace these keys later on and should be able to migrate them just like any other piece of account-state information.

Were that done, then the next — and harder — problem is how to set up an exchange of keys in a way that a user might find intuitive. This clearly involves both MUAs, the sender's and the recipient's. And of course, many of both will actually be Web mail users, of which there are many forms. A fairly obvious answer could be to simply sign every outbound message and include the relevant public key (because each mail account now has a key pair associated). Crucially, however, those signatures shouldn't be visible in the MUA because they're solely used to distribute public keys and not really to provide origin authentication. That way, it would be highly likely that anyone who wanted to send me an encrypted message could already do so because they already had a key that is, or purports to be, mine. However, it's important to note that even if you use the current S/MIME or OpenPGP format here, the fact that you've signed the message is, in itself, meaningless. All we're doing is distributing the sender's public key in a way that could be verifiable; we aren't providing origin authentication, but that's okay because users don't want that anyway.

However, this leads to what the Secure Shell would consider a leap of faith. When a sending user first wants to send a secure mail to a given recipient for whom he has a public key, then the (probably quite remote) possibility exists that the public key isn't the intended recipient's but is rather an interloper's. How do we explain this to a random user who doesn't (and shouldn't have to) know anything about asymmetric cryptography? Perhaps an answer here is as follows: if everything is already in place, we've done the "leap of faith" before, and nothing has changed in the meantime, just give the user an *encrypt* button they can turn on as they author a message. If not, then give users a *how to encrypt this* link to a Web page where they can find step-by-step instructions and explanations for that particular pair of sending and receiving MUAs and for the particular state in which they currently find themselves. It's highly likely that the sending MUA has some information about the receiving MUA because some message from the intended recipient is very likely to be in the sender's set of stored messages.

That way, the MUA can give the sender a link that's specific to the pair of MUAs involved, and the sender can follow the instructions accordingly. Building such a Web site could

start anytime — all that the designers would need is some convention to map between pairs of MUA identifiers — for example, User-Agent: Thunderbird 2.0.0.16 (X11/20080707) and the relevant instructions. Some set of MUA vendors (or a community effort) could provide URLs and helpful instructional text on those pages. Indeed, such Web content would be useful even with totally unmodified MUAs. The link might also incorporate other state information, especially whether the sending MUA appears to have a key for the recipient and what to do if not.

Of course, if that Web content simply tried to steer users toward existing PKIs — for example, to get them to buy a certificate — that would, I believe, be counterproductive. Because users simply don't want to use this functionality very often, seeding its use should be more important than trying to make money at this stage. At some future time, if users employ their encrypt buttons, then there could be a market for selling public-key certificates for email purposes, but today there isn't. So, my small proposal for a change is something that MUA vendors could consider — if they really want to encourage broader use of all those lines of code they've already deployed.

Even after all the effort that the security community and MUA developers have devoted to developing and deploying S/MIME, its use in the wild remains miniscule. This is partly, I believe, because the set of services S/MIME and even OpenPGP provide are those that security experts (like myself — mea culpa) envisioned two decades ago. However, casual use of end-to-end security is reduced further by essentially unimaginative implementation of the features in MUAs that really only support corporate users, and then only if they have well-resourced and security-aware administrators. MUA developers should go back to basics and ask what users really want here — the ability to occasionally encrypt an email without much trouble at all.

References

1. E. Allman et al., *DomainKeys Identified Mail (DKIM) Signatures*, IETF RFC 4871, May 2007; www.rfc-editor.org/rfc/rfc4871.txt.
2. B. Ramsdell, ed., *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification*, IETF RFC 3851 July 2004; www.ietf.org/rfc/rfc3851.txt.
3. J. Callas et al., *OpenPGP Message Format*, IETF RFC 2440, Nov. 1998; www.ietf.org/rfc/rfc2440.txt.
4. J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*, IETF RFC 1421, Feb. 1993; http://dret.net/rfc-index/reference/RFC1421.
5. R. Klein, "Internet-Based Patient-Physician Electronic Communication Applications: Patient Acceptance and Trust," *e-Service J.*, vol. 5, no. 2, 2007, pp. 27–52.
6. Apu Kapadia, "A Case (Study) For Usability in Secure Email Communication," *IEEE Security and Privacy*, vol. 5, no. 2, 2007, pp. 80–84.
7. P. Resnick, ed., *Internet Message Format*, IETF RFC 5322, Oct. 2008; www.ietf.org/rfc/rfc5322.txt.
8. J. Klensin, *Simple Mail Transfer Protocol*, IETF RFC 5321, Oct. 2008; www.ietf.org/rfc/rfc5321.txt.
9. A. Carzaniga and A.L. Wolf, "Content-Based Networking: A New Communication Infrastructure," LNCS 2538, Springer-Verlag, 2002, pp. 59–68.

Stephen Farrell is a research fellow at Trinity College Dublin. His research interests include security and delay/disruption-tolerant networking. Farrell has a PhD in computer science from Trinity College Dublin. Contact him at stephen.farrell@cs.tcd.ie.