A Case (Study) For Usability in Secure Email Communication

Apu Kapadia
Dartmouth College

As a network security researcher, I find it very disappointing that most users can't, or simply don't, secure their everyday Internet communications. For good reason, usability in security has received a fair deal of attention in the past few years (see the September 2004 special issue on this topic[1]). To push the issue further, I decided to initiate my own informal case study on the usability and practical relevance of standard security mechanisms for email communication.

I focused my attention on available public-key cryptography techniques for digitally signing and encrypting email. My first step was to establish a public–private key pair to use with email. I chose to use Secure/Multipurpose Internet Mail Extensions (S/MIME), a standard for signing and encrypting email, because it's already supported by popular email clients such as Apple Mail, Outlook Express, and Mozilla's Thunderbird. Unlike S/MIME, I found that Pretty Good Privacy (PGP) and the GNU Privacy Guard (GPG) were unusable with nontechnical correspondents because it required them to install additional software. S/MIME, it seemed, was the better solution for these "everyday users," for whom the concepts of public-key infrastructure (PKI), PGP, certificates, keys, and so on remain elusive. Additionally, I decided to get my public key certified by Thawte (www.thawte.com), an online certificate authority (CA).

Digital signatures

After months of signing email, I've realized that, currently, everyday users seldom need to do so, as we will see if we examine email signatures more closely.

For any message, Alice can use her private key to generate a cryptographic package that a recipient can verify only by using her public key and the original message. This package is called a *digital signature* and provides two basic properties: nonrepudiation and integrity.

Nonrepudiation

Nonrepudiation is the idea that, in theory, a signer such as Alice can't later deny that she signed the message. For example, occasionally I submit reviews for conference papers over email. I could digitally sign my messages to claim responsibility for my words. But as any security researcher would be quick to point out, digital signatures' nonrepudiability is just an illusion. Alice can always claim that someone stole her private key and that the signature is a forgery. And if that's not enough, Alice can publish her key in *The New York Times*, letting potentially anybody sign a message using it. In such situations, Alice can be penalized for negligence or irresponsible behavior, but she can't be held responsible for the contents of messages signed with her private key. Even if Bob tries to hold Alice to her original contract by proving that the signature he possesses was created before Alice published her key—perhaps by using a time-stamping service[2] or an online notary—Alice can still claim that she didn't know her key was stolen. More sophisticated protocols for nonrepudiation are needed, but as it now stands with standard S/MIME, nonrepudiation for casual email users doesn't work in practice.

Integrity

Forging email messages on today's Internet is surprisingly easy, and forgeries such as phishing emails are a direct threat to everyday users. In theory, if messages are digitally signed, recipients can reject those with spoofed "From" addresses because their signatures won't be valid—that is, only Paypal can sign messages that appear to come from paypal.com. Digital signatures also provide protection against adversaries who modify parts of the message in transit, although I would argue that such email modifications present very little threat to everyday users—for them, digital signatures' main utility is in countering forged sender addresses.

In practice, however, digital signatures are a weak line of defense. Phishers can use cleverly crafted email addresses such as customer-service@paypal-help.com to trick users into believing that they're corresponding with Paypal. Because phishers can legitimately own a domain such as paypal-help.com, a phisher can obtain a certificate and generate emails from that domain that have valid signatures (this is just a hypothetical example, but at the time of writing, paypal-help.com was registered under a for-

eign mailing address). Any mechanism that combats phishing must look beyond the integrity protection that digital signatures provide. Given that most email that users receive is unsigned, users routinely verify a message's integrity based on its contents and context. In fact, I find myself verifying messages' integrity based on their overall content, even when they are digitally signed. For the lack of a better term, I call this form of integrity "semantic integrity," in contrast to the standard notion of (syntactic) integrity that digital signatures provide. When corresponding with familiar people, verifying the semantic integrity of email messages is surprisingly easy—digitally signed or not, strange text from a friend that contains an attached virus looks suspicious. I routinely ignore signatures from family, friends, and acquaintances simply because I'm confident that I can sniff out forgeries.

At this point I will re-emphasize my focus on everyday users. Certainly, defense contractors, network administrators, and so on are well advised to digitally sign messages to correspondents who expect them. You can instruct employees to reject any message from the security officer without a valid signature—certain job functions rely on baseline security mechanisms for which you can provide training. For everyday users, however, using digital signatures to verify messages' integrity is both overkill and prone to error, the former because using signatures for detecting alterations doesn't address a tangible threat, and the latter because telling everyday users to "ensure that the signature is valid" to detect forgeries is a misguided heuristic. Focusing on tools that will help them verify semantic integrity, instead, is more promising.

Incrimination

Given that the two most important properties of digital signatures don't seem useful in practice, why might everyday users continue to sign email? The property of incrimination, although anecdotal, has made a lasting impact on my use of signatures and highlights the need for more research on usability in security.

By default, some email clients attempt to digitally sign replies to signed messages. While responding to my signed email, a correspondent who works for the military was told to "insert cryptocard." Because the correspondent was not familiar with digital signatures, I received a reply with a suspicious tone (whether intended or not, this is how I interpreted it). With the prospect of a potentially peeved military official, I found myself obliged to explain that I was not trying to do anything sneaky with government computers, and that the email client was the culprit with its automated behavior. A couple of test emails, with and without signatures, convinced the correspondent of my theory—that the email client was indeed trying to automatically sign replies to my signed messages.

In a separate incident, another correspondent, also unfamiliar with PKI, was facing problems after encountering a certificate signed by an untrusted CA. After clicking on "examine certificate," and a stray click later, my certificate was presented for examination. The email client automatically obtained this certificate from earlier messages I had signed. From my correspondent's viewpoint, however, the problem with connecting to an untrusted email server was somehow linked to my name. Again, I found myself obliged to explain that I wasn't trying to do anything sneaky with my correspondent's email client. These incidents have taught me an important lesson: sign your messages only to people who understand the concept. Until more usable mechanisms are integrated into popular email clients, signatures using S/MIME should remain in the domain of "power users."

Encryption and the key distribution problem

Now, more than ever, the privacy of our communications is at risk. The government is increasingly interested in our conversations, and in an open system such as the Internet, we must take added measures to ensure our privacy rights. With the confi-

dentiality of my electronic conversations in mind, I convinced some of my research colleagues to encrypt their email conversations with me.

By pre-installing third-party CA certificates into email clients without rigorous auditing procedures, vendors are breaking the trust model required for PKI to be successful.

While exchanging public keys, the most important step is to verify that a man-in-the-middle isn't subverting your exchange. If we assume that an adversary can control our conversations, we must verify the exchanged public keys' authenticity.

Charlie, a man-in-the-middle, can pretend to be Bob with respect to Alice and Alice with respect to Bob. Alice and Bob communicate "securely," except that they're both communicating through Charlie without realizing that he's decrypting and re-encrypting their messages. The most secure way for Alice and Bob to verify their keys' authenticity is to do so in person; this, however, is impractical, giving rise to the key-distribution problem—how can users distribute their public keys to other parties reliably? The PKI world has developed two solutions: either rely on a trusted third party (or a more elaborate network of trusted third parties) such as Thawte or VeriSign (www.verisign.com) to certify that your correspondent's public key is bound to his or her identity, or verify the authenticity yourself by checking the public key's *fingerprints* through an out-of-band (OOB) channel—that is, by a separate means of communication.

Third-party "trust"

Verifying the authenticity of keys with my correspondents was surprisingly error-prone. Let's analyze the PKI solution that relies on CAs first. If Alice's public key is certified (digitally signed) by a CA that Bob trusts, then Bob will accept Alice's certificate as being authentic. If Alice's key is certified by a CA that's not on Bob's trusted list, Bob can try to find a *trusted path* to Alice's certificate by starting at a CA that he does trust. Let's say that Bob trusts only CA_1 and encounters Alice's certificate signed by CA_3. Bob can try to find a chain of trust in which CA_1 certifies CA_2, who in turn certifies CA_3 (certificate chains can be much longer in practice). This certificate chain lets Bob establish a path of trust to Alice's certificate, even though he doesn't explicitly trust CA_3. PKI proposes meshes of CAs established by certification relationships. Meshes can also include hierarchies of higher-level CAs certifying lower-level CAs and cross-certification authorities which can bridge trust hierarchies into a mesh to aid in building trust paths.

Although this approach can provide a high level of assurance in enterprise-level communications, it has a few limitations when applied to email exchanges between everyday users. Mainly at fault is the list of "trusted" CAs that the email client's software vendor has pre-installed. A colleague of mine, Scott Rea, calls this a list of "third parties" as opposed to a list of "trusted third parties" because this list doesn't correspond to the set of CAs that the email client's users trust. After all, I chose not to get my public key certified from an authority that I had never heard of (and hence didn't trust), but rather had it certified by Thawte. My correspondents, however, don't know my trusted CA a priori. A powerful man-in-the-middle attack could indeed create a bogus certificate for my identity, certified by a malicious CA that I don't trust, but that is on the list of installed third-party CAs. Because the S/MIME email client would trust the certificate, were my colleagues accepting a fake certificate signed by another CA or were they accepting my Thawte certificate?

Clearly, users must first trust the CAs installed in their email clients. Second, if Alice and Bob are exchanging keys, they should use a CA that they both trust. Absent a common trusted CA, the just-mentioned man-in-the-middle attack is still possible, with or without certificate chains. PKI has been plagued by its end-points—by pre-installing third-party CA certificates into email clients without rigorous auditing procedures, vendors are breaking the trust model required for PKI to be successful.

Now, consider enterprise systems, in which organizations can make rigorous policy decisions about a CA's certification procedures and thereby outsource the key management functions to a trusted CA. They can also make rigorous policy decisions regarding valid trust paths to other CAs. For example, the Higher Education Bridge Certification Authority (HEBCA) has a stringent process of assigning *levels of assurance* (LOA) to CAs that are part of the bridge. Higher education organizations can then trust HEBCA, and the organizations that are part of HEBCA can trust each other's certificates. In other words, HEBCA "bridges" trust between different organizations operating under their own PKIs by certifying their CAs' practices. Training employees within an organization to recognize valid certificates is feasible because the organization has a financial incentive to do so. Everyday users, however, don't have the time or motivation for rigorous bookkeeping about various CAs' certification procedures. CA-certified keys and

trusted paths are less meaningful if users don't understand the certifying CA's procedures and are willing to accept any certificate that their email client trusts. (Note, however, that PKI can be quite successful as a means for an enterprise-level organzation to authenticate everyday users—the organization can have rigorous policies about which CA's certificates it should accept, without including everyday users in these trust decisions.)

As I've argued, exchanging keys using current implementations of S/MIME is risky for everyday users because their trust in their email clients is misplaced. We must take a long-term approach toward building usable key-management methods and educating everyday users about trusting CAs and establishing a common root of trust with their correspondents. An independent organization such as HEBCA can audit CAs carefully and help establish a common root of trust. Users and email client vendors can then be instructed to trust only CAs with the auditing organization's approval. In the short-term, however, because most everyday users don't have mutually trusted CAs, they should use the second solution, fingerprint verification, to foil man-in-the-middle attacks.

Fingerprint verification

A fingerprint is a secure hash of the public key and is a smaller, digested form. Verifying that the exchanged key's fingerprint matches the original key's fingerprint is a much faster way to verify the key's authenticity.

The recently proposed concept of *key continuity management* (KCM)[3,4] is an emerging alternative to the CA-based approach. KCM posits that once Bob has verified a key's fingerprints, he can be sure that the key he uses for encryption is the same one he's verified in the past. Users needn't rely on an elaborate network of CAs to certify keys. As with SSH, users of email clients are assumed to verify a newly observed key's fingerprint, after which key continuity gives the user a sense of security. This approach has limitations, however: what can Alice do if her key is compromised? In a CA-based approach, before using Alice's key to secure communications, Bob can check the CA's revocation list or use the Online Certificate Status Protocol (OCSP) to ensure that it hasn't been compromised. KCM, however, relies on Alice informing all her correspondents that her key has been compromised. KCM proponents argue that the added benefit of an infrastructureless approach outweighs the reduction in security from potentially compromised keys. If users verify fingerprints often enough, they can limit the amount of damage a compromised key causes.

This brings us to one final question: how can users verify a key's fingerprints reliably? One option is to verify fingerprints for email over IM and fingerprints for IM over email. However, this approach still won't protect us against motivated adversaries (or our employers!) who can intercept both communication lines and subvert our attempted OOB fingerprint verification.

Exchanging SMS messages is a viable option[5] because the mobile phone network is clearly separated from our organizations' networks (or are they?). After hearing about the purported collaboration between the NSA and AT&T, however, relying on phone companies to deliver electronic fingerprints also seems risky against capable adversaries. In the end, if you can't verify fingerprints in person, it seems safest to verify them over the phone. This is the standard method of fingerprint verification in voice-over-IP (VoIP) services such as Philip Zimmermann's Zfone (www.philzimmermann.com/EN/zfone/), given that it's very difficult for a man-in-the-middle to subvert a voice conversation in real time. Additionally, humans can easily verify the semantic integrity of a voice conversation with a known correspondent because a man-in-the-middle would have trouble impersonating your correspondent's voice. (Caveat: humans are poor at verifying the semantic integrity of conversations with unknown correspondents, a weakness that is exploited in social engineering attacks.) It would be prudent, however, to expect computers in the not-too-distant future to be able to synthesize voice in real time. A dedicated man-in-the-middle could possibly replace the part of your conversation related to fingerprint verification. Soon, we will need more sophisticated methods for verifying a remote correspondent's fingerprints, but until then, relying on real-time voice verification seems to be the best option. In my personal experience, my correspondents seemed rather uncomfortable with the "geekiness" of reading random numbers over the phone. However, with VoIP software becoming more popular among everyday users, a mechanism to use the same verified keys for email communications will be a great solution to the problem of OOB fingerprint verification.

Although neither the trusted third-party nor fingerprint solutions in their current forms seem sufficiently secure for everyday users, perhaps a hybrid approach is needed in the short term. As I suggested with CA-based PKI, everyday users

In the end, if you can't verify fingerprints in person, it seems safest to verify them over the phone.

should verify a key's fingerprints. Mechanisms developed for KCM can bolster trust in CA-certified keys and ensure that users verify fingerprints to secure communication.

There are several barriers for everyday users who wish to secure their communications. S/MIME is supported by popular email clients, but casual users are lulled into a false sense of security; accepting "valid" signatures without comprehending the underlying trust assumptions or being content with encrypted email without being diligent about fingerprint verification highlights the mismatch between the user's expectations and their communication's underlying security.

On the optimistic front, PKI awareness is increasing—here at Dartmouth College, all first-year students are issued PKI tokens, and research on usability for secure communication is gaining momentum. One promising approach uses attribute-based annotations to help users make better trust decisions about their email communication.[6] Until such usable mechanisms are introduced into popular email clients, however, proceed with caution and verify those fingerprints. ❑

Acknowledgments

The author thanks Scott Rea for his insightful comments and willingness to read multiple drafts of this article. He also thanks Sean Smith, Patrick Tsang, and Phoebe Wolfskill for their helpful comments.

References

1. *IEEE Security & Privacy*, special issue on usable security, vol. 2, no. 5, 2004.
2. S. Haber and W.S. Stornetta, "How to Time-Stamp a Digital Document," *J. Cryptology*, vol. 3, no. 2, 1991, pp. 99–111.
3. P. Gutmann, "Why Isn't the Internet Secure Yet, Dammit," *Proc. AusCERT Asia Pacific Information Technology Security Conf.*, AusCERT, May 2004; http://conference.auscert.org.au/conf2004/.
4. S.L. Garfinkel and R.C. Miller, "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express," *Proc. Symp. Usable Privacy and Security* (SOUPS 05), ACM Press, 2005, pp 13–24.
5. A.J. Nicholson et al.,"LoKey: Leveraging the SMS Network in Decentralized, End-to-End Trust Establishment, *Proc. 4th Int'l Conf. Pervasive Computing* (Pervasive 06), LNCS 3968, Springer-Verlag, pp. 202–219.
6. C. Masone and S.W. Smith, "Towards Usefully Secure Email," *IEEE Technology and Society Magazine*, to be published, Mar. 2007.

***Apu Kapadia** is a post-doctoral research fellow at the Institute for Security Technology Studies, Dartmouth College. His research interests include systems security and privacy, and he is particularly interested in anonymizing networks and usable mechanisms for enhancing privacy. Kapadia has a PhD in computer science from the University of Illinois at Urbana-Champaign. He is a member of the IEEE and the ACM. Contact him at akapadia@cs.dartmouth.edu.*