Waterhouse: Enabling Secure E-mail with Social Networking

Alex P. Lambert
Department of Computer Science
University of Illinois
Urbana, Illinois
alamber3@uiuc.edu

Stephen M. Bezek
Department of Computer Science
University of Illinois
Urbana, Illinois
sbezek2@uiuc.edu

Karrie G. Karahalios
Department of Computer Science
University of Illinois
Urbana, Illinois
kkarahal@cs.uiuc.edu

CHI 2009, April 4 – 9, 2009, Boston, MA, USA
ACM 978-1-60558-246-7/09/04.

Abstract
We present Waterhouse, a system for sending and receiving cryptographically protected electronic mail ("secure e-mail"). We show how an existing e-mail interface can be modified to make exchanging secure e-mail nearly effortless. Our system integrates with social networking services (such as Facebook) to automatically exchange cryptographic keys between friends. When a user sends a message to a friend, our system automatically encrypts the contents to thwart eavesdroppers. When a user receives a message from a friend, Waterhouse uses the recipient's social network to verify the sender's identity. Our prototype shows senders' photos as an intuitive indicator of message authenticity. We describe our planned user study and conclude with directions for future work.

Keywords
Security, privacy, social computing, cryptography

ACM Classification Keywords
D4.6. Security and privacy protection: Cryptographic controls. H5.2. User interfaces: Graphical user interfaces (GUI).

Introduction

Conventional electronic mail is not secure: it is easy to forge a sender's address, and eavesdropping on messages is not difficult [6]. Cryptographically protected electronic mail ("secure e-mail") offers two important advantages over unprotected e-mail [3]. First, the recipient can verify the sender's identity with a cryptographic signature; a valid signature proves that the message was not forged. Second, the message contents can be protected from eavesdroppers using encryption. To exchange secure e-mail messages, the sender and recipient must each have a cryptographic "key pair" (consisting of a "public key" and a related, secret "private key"). Furthermore, the sender must know the recipient's public key, and the recipient must know the sender's public key.

The underlying mathematical principles for cryptographically protected communication were published more than three decades ago [2]. Standards for sending and receiving secure e-mail have existed for over twenty years [8], and popular e-mail clients include interoperable implementations of the standards [9]. But critical usability issues prevent widespread use of cryptographically protected e-mail [10].

Meanwhile, e-mail users have remained vulnerable. In 1998, an executive of a rare book listing service allegedly "directed [his] employees to intercept and copy all incoming communications to [his customers] from Amazon.com." Prosecutors alleged that the company "intercepted thousands of messages" and that employees "routinely read the e-mail messages sent to [customers] in the hope of gaining a commercial advantage." [11]

In practice, two challenging problems prevent widespread deployment of secure e-mail: secure e-mail clients are not intuitive [12], and there is no widespread, reliable method of locating others' public keys [5]. We present Waterhouse, a system that addresses both issues by integrating with social networking services such as Facebook.

Related work

"Why Johnny Can't Encrypt" [12]

Whitten and Tygar's 1999 usability evaluation of a popular secure e-mail package (Pretty Good Privacy 5.0 [14] with Eudora) found significant problems with the product's user interface. The evaluators questioned the product's analogy between cryptographic keys and physical keys and its metaphor for sender authentication.

Their testing also exposed practical usability challenges: of the twelve participants in their user study, only a third successfully sent a secure e-mail message in the 90-minute test period. A quarter accidentally sent confidential information in a non-secure message. The authors concluded that the interface "[did] not come even reasonably close to achieving our usability standard" and that it "does not make [exchanging secure e-mail] manageable for average computer users."

Whitten and Tygar formalized five problems that security technology designers must address; two are especially relevant to our discussion. First, the "unmotivated user" problem is straightforward: for most users, security is not a primary goal. Second, security technologies must provide understandable,

actionable feedback to the user to prevent dangerous errors; this is the "feedback" problem.

"Johnny 2" [10]
In 2005, Garfinkel and Miller created and evaluated a system based on key continuity management (KCM). Their prototype, CoPilot, addressed the problem of finding others' public keys by automatically learning senders' keys from incoming secure messages. They expanded Whitten and Tygar's user study to test subjects' responses to forged messages from attackers.

Garfinkel and Miller's interviews revealed that after interacting with CoPilot for less than an hour, users generally understood the benefits of secure e-mail. They found that while the KCM approach generally improved security, users had trouble responding correctly to forged messages. Additionally, only a third of the experimental subjects elected to use encryption when sending confidential data; most sent the information without protecting it from eavesdroppers. Some participants expected the e-mail client to protect them from mistakes: they said that if encryption were important for their scenario, a system administrator would have configured the e-mail client to send only encrypted messages. These results suggest that future systems should carefully consider forgery issues, provide clear information about the security of outgoing messages, and automatically send messages securely whenever possible.

Waterhouse: three radical changes

Our system, Waterhouse, significantly reduces the difficulty of exchanging secure e-mail. Waterhouse acts as an extension to existing e-mail clients. Our system incorporates three radical changes that distinguish it from other secure e-mail solutions:

- Waterhouse automates security tasks. For outgoing messages, Waterhouse automatically employs the strongest security possible. For example, when sending a message to another Waterhouse user, the message contents are automatically encrypted to prevent eavesdropping. The sender does not have to explicitly choose to protect his message. Waterhouse requires only minimal configuration, and it automatically generates a cryptographic key pair when first installed.
- Waterhouse distributes public keys using social networking services (such as Facebook). Before exchanging secure e-mail, the sender and recipient must know one another's public key. Waterhouse exploits social networking services to automatically exchange these keys between friends. (Waterhouse *only* exchanges keys with friends. We assume that if two users are friends on a social networking service, they have verified one another's identity. An attack and a potential solution are discussed in the future work section.)
- Waterhouse integrates social network data into its user interface. For example, when a user reads a secure message, the sender's photo appears in the message window. In contrast, another popular system displays opaque, non-intuitive information (including a hexadecimal "key ID") in this area [4].

A user scenario

The easiest way to understand Waterhouse is to consider a typical, concrete use case: Maria and her friend Don would like to communicate securely. Maria

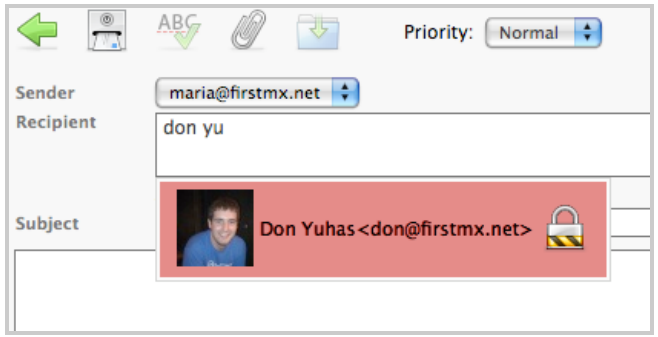

Figure 1. Facebook friends who are Waterhouse users are automatically added to Maria's address book.

and Don are both casual Facebook users. Maria, who already uses Waterhouse, asks Don to try Waterhouse.

First, Don obtains the Waterhouse software. (In our current prototype, Waterhouse is integrated into an existing open-source, web-based e-mail client.) Next, Don links Waterhouse to his Facebook account. This step allows our system to look up information about his Facebook friends. In our prototype, Waterhouse only accesses friends' names, profile photos, e-mail addresses, and public keys. At this point, Waterhouse automatically generates a cryptographic key pair for Don. It then publishes Don's public key to the Facebook service. Don's public key is now available to his friends. Don and Maria can now exchange secure e-mail.

Later, when Maria composes a message to Don, our system retrieves a list of her Facebook friends. Waterhouse users are automatically merged into Maria's address book; each friend's entry includes his profile picture and public key. (This aspect is similar to Lieberman and Miller's Facemail system [7].)

When Maria starts typing Don's name, Don's picture appears with a lock icon, indicating that she can securely communicate with him (Figure 1). Waterhouse adds a green strip to the top of the message; the strip's text informs Maria that her message will be encrypted and therefore protected from eavesdroppers (Figure 2).

Finally, when Don receives Maria's message, Waterhouse automatically decrypts the message and checks Maria's digital signature. If this process succeeds, Waterhouse adds a green notification to the e-mail window (Figure 3). This notification includes Maria's picture; we expect that users will quickly associate the presence of a photo and green strip with a secure message.

If the verification process fails or if the message from Maria was not sent securely, Waterhouse instead adds a warning that the message may have been forged. To avoid displaying misleading warnings, our system shows the forgery notification only when the purported sender is capable of sending secure messages.

Figure 2. In our interface text, we avoid technical terms in favor of meaningful, understandable descriptions. For example, we replace the word "encrypted" with a description of the benefit of encryption.

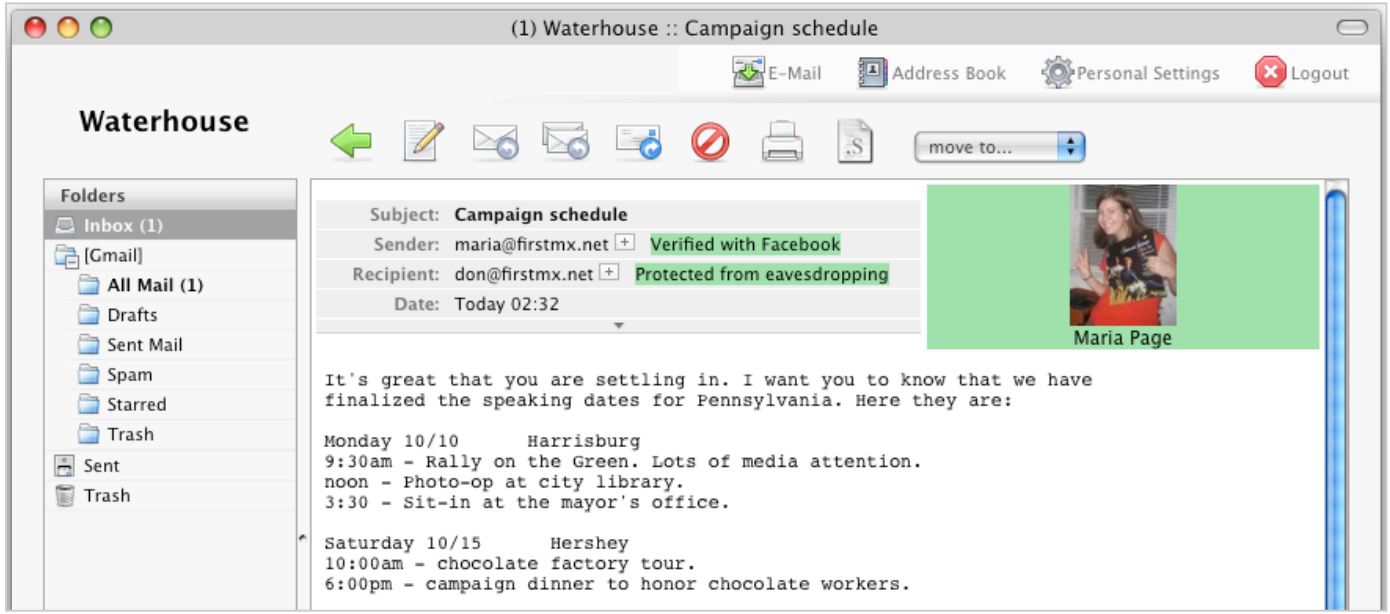

Figure 3. Waterhouse displays a secure message. The sender's photo and name is displayed along with a notification that the message was protected from eavesdroppers.

Future directions

We have created a complete, functional prototype, including Facebook connectivity and strong cryptography. We plan to begin user testing in early 2009 and publish our results in a later paper. We plan to base our experiments on Garfinkel and Miller's protocols so we can compare our technique against their KCM results.

Our work focuses on the key exchange and user interface aspects of secure messaging. Many closely related problems remain open. Because a security system is only as strong as its weakest component, successful attacks against other related systems could compromise the security of a user's messages. For example, our system does not prevent a user from divulging his Facebook account password at a phishing site. Easy-to-use, phishing-resistant authentication methods are not yet popular, but technologies like InfoCard [1] appear promising. Our current prototype also does not address the problem of key management; it is up to the user to store his private key securely. Our system is designed to work in tandem with solutions to these problems.

Waterhouse assumes that if two users are friends on a social networking service, they have verified one another's identity. If a malicious user wanted to attack

Maria by impersonating Don, he could create a Facebook profile with Don's name and photo and send a friendship request to Maria. If Maria approved the request, the malicious user could send secure messages to Maria as Don. To address this problem, Waterhouse could ignore the public key of any friend with less than *n* friends in common with the recipient (for some value of *n*). With this change, a friend is trusted only when other friends vouch for him; this intuition is similar to the "web of trust" ideal in the Pretty Good Privacy package. [14]

Our system is subject to network effects; a user will find Waterhouse compelling only if his friends are also using it. Therefore, it may be difficult to convince the first users to install Waterhouse. Including it with future versions of existing popular e-mail clients could overcome this problem. Yahoo recently released a software development kit for its web-based e-mail service [13]. We plan to investigate the feasibility of porting Waterhouse to this new platform.

Acknowledgements

Eric Gilbert and Tony Bergstrom provided valuable feedback. Simson Garfinkel and Robert Miller generously released their study protocols and documentation.

References

[1] Bertocci, V., Serack, G., Baker, C. *Understanding Windows CardSpace: an introduction to the concepts and challenges of digital identities.* Addison-Wesley Professional, Upper Saddle River, NJ, USA, 2007.

[2] Diffie, W., Hellman, M.E. New directions in cryptography. *IEEE Trans. Inform. Theory,* IEEE Press (1976). 644-654.

[3] Dusse, S., et al. S/MIME version 2 message specification (RFC 2311). The Internet Society (1998).

[4] Enigmail Project. Enigmail 0.95.6 screenshots. http://enigmail.mozdev.org/documentation/screenshots.php

[5] Gutmann, P. How to build a PKI that works. In *Proc. Third Annual PKI R&D Workshop*, NTIS (2004).

[6] Klensin, J. Simple mail transfer protocol (RFC 2821). The Internet Society (2001).

[7] Lieberman, E., Miller, R.C. Facemail: showing faces of recipients to prevent misdirected email. *Proc. Usable Privacy and Security 2007*, ACM Press (2007), 122-131.

[8] Linn, J. Privacy enhancement for Internet electronic mail (RFC 989). IAB Privacy Task Force (1987).

[9] Réseaux IP Européens Network Coordination Centre. E-mail client testing for S/MIME compliance. http://www.ripe.net/db/support/security/mail_client_tests.html

[10] Simson, G.L., Miller, R.C. "Johnny 2:" a user test of key continuity management with S/MIME and Outlook Express. *Proc. Usable Privacy and Security 2005*, ACM Press (2005), 13-24.

[11] U.S. Court of Appeals for the First Circuit. *United States of America v. Bradford C. Councilman*, 418 F.3d 67.

[12] Whitten, A., Tygar, J.D. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. *Proc. USENIX Security 1999*, USENIX Association (1999), 14.

[13] Yahoo. Yahoo! Mail Web Services. http://developer.yahoo.com/mail/

[14] Zimmermann, P.R. *The official PGP user's guide.* MIT Press, Cambridge, MA, USA, 1995.