Usability Studies

Email Encryption

Usability evaluation of PGP 5.0

- Defines a standard of usable security
- Evaluated PGP 5.0 Using
  - Direct evaluation (cognitive walkthrough)
  - User experiments
- Conclusions:
  - PGP 5.0 does not meet the usability security standard
  - Confirms hypothesis that “security-specific user interface design principles and techniques are needed.”

Why Johnny Can’t Encrypt:
A Usability Evaluation of PGP 5.0

Alma Whitten
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
alma@cs.cmu.edu

J. D. Tygar[1]
EECS and SIMS
University of California
Berkeley, CA 94720
tygar@cs.berkeley.edu

Abstract

User errors cause or contribute to most computer security failures, yet user interfaces for security still tend to be clumsy, confusing, or near-nonexistent. Is this simply due to a failure to apply standard user interface design techniques to security? We argue that, on the contrary, effective security requires a different usability standard, and that it will not be achieved through the user interface design techniques appropriate to other types of consumer software.

To test this hypothesis, we performed a case study of a security program which does have a good user interface by general standards: PGP 5.0. Our case study used a cognitive walkthrough analysis together with a laboratory user test to evaluate whether PGP 5.0 can be successfully used by cryptography novices to achieve effective electronic mail security. The analysis found a number of user interface design flaws that may contribute to security failures, and the user test demonstrated that when our test participants were given 90 minutes in which to sign and encrypt a message using PGP 5.0, the majority of them were unable to do so successfully.

We conclude that PGP 5.0 is not usable enough to provide effective security for most computer users, despite its attractive graphical user interface, supporting our hypothesis that user interface design for effective security remains an open problem. We close with a brief description of our continuing work on the development and application of user interface design principles and techniques for security.

1 Introduction

Security mechanisms are only effective when used correctly. Strong cryptography, provably correct protocols, and bug-free code will not provide security if the people who use the software forget to click on the encrypt button when they need privacy, give up on a communication protocol because they are too confused about which cryptographic keys they need to use, or accidentally configure their access control mechanisms to make their private data world-readable. Problems such as these are already quite serious: at least one researcher [2] has claimed that configuration errors are the probable cause of more than 90% of all computer security failures. Since average citizens are now increasingly encouraged to make use of networked computers for private transactions, the need to make security manageable for even untrained users has become critical [4, 9].

This is inescapably a user interface design problem. Legal remedies, increased automation, and user training provide only limited solutions. Individual users may not have the resources to pursue an attacker legally, and may not even realize that an attack took place. Automation may work for securing a communications channel, but not for setting access control policy when a user wants to share some files and not others. Employees can be required to attend training sessions, but home computer users cannot.

Why, then, is there such a lack of good user interface design for security? Are existing general user interface design principles adequate for security? To answer these questions, we must first understand what kind of usability security requires in order to be

[1] Also at Computer Science Department, Carnegie Mellon University (on leave).

Usable Security standard

Definition: Security software is usable if the people who are expected to use it:

1. **are reliably made aware of the security tasks they need to perform;**
2. **are able to figure out how to successfully perform those tasks;**
3. **don’t make dangerous errors; and**
4. **are sufficiently comfortable with the interface to continue using it.**

Design differences

Designing for security has unique challenges that must be accounted for in designing for usability:

- Unmotivated user (security is a secondary goal)
- Abstraction (policies/rules are unintuitive to general population)
- Feedback (security state is complex and difficult to depict)
- “barn door” (cannot make serious mistakes)
- “weakest link” (must attend to all aspects of security; cannot learn/manage incrementally as with other software)

Walkthrough evaluation

- Metaphor issues
  - Keys: a different key is used for encryption than for decryption unlike a single “real” key which does both
  - Signature: not clear that signing (quill pen icon) requires the use of the private key
  - Key types: distinction between RSA keys (blue) and Diffie-Hellman (brass) not clear

Walkthrough evaluation

- Key management issues
  - Key server:
    - no top-level visibility;
    - not identified as a remote operation;
    - no history of access
  - Key rating:
    - Validity (completely, marginally, invalid) – degree of confidence that key belongs to given user
    - Trust (completely, marginally, untrusted) – degree of confidence in another user as certifier of keys
    - Assigned automatically
    - Problems
      - User may assign now meaning to “validity” and “trust”
      - Automatic assignment not visible

Walkthrough evaluation

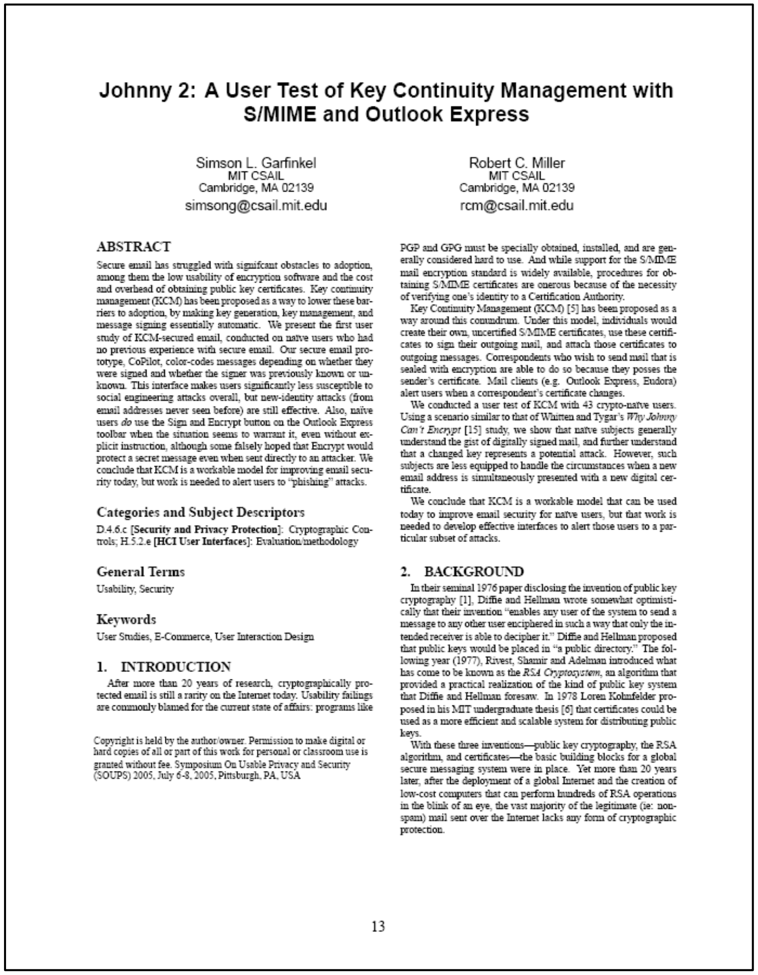
- Reversibility:
  - Insufficient notice (e.g., deleting the private key)
  - Insufficient guidance on what actions are need to undo the effects of an otherwise irreversible operation (accidental key revocation)
- Consistency (terminology: “encode” vs. “encrypt”)
- Too much information (does not separate information relevant to novice vs. advanced users)

User experiment

- Task: send sensitive political campaign information via encrypted email to five others.
- Participants: 12, email proficiency, security novices
- Results:
  - only 1/3rd of the subjects were able to complete the task in 90 minutes
  - 1/4th of the subjects accidentally exposed the sensitive information
  - Subjects' difficulties stemmed from inadequate understanding of the public-key model

Email encryption redux

- Repeats Whitten/Tygar experiments with 43 crypto-naïve users
- Uses newer systems (S/MIME) in combination with Key Continuity Management (KCM)
- Claims/Results:
  - Less secure (in principle) but more usable (in practice)
  - Better interfaces needed for a specific situation



Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express

Simson L. Garfinkel
MIT CSAIL
Cambridge, MA 02139
simsong@csail.mit.edu

Robert C. Miller
MIT CSAIL
Cambridge, MA 02139
rcm@csail.mit.edu

ABSTRACT

Secure email has struggled with signifcant obstacles to adoption, among them the low usability of encryption software and the cost and overhead of obtaining public key certificates. Key continuity management (KCM) has been proposed as a way to lower these barriers to adoption, by making key generation, key management, and message signing essentially automatic. We present the first user study of KCM-secured email, conducted on naïve users who had no previous experience with secure email. Our secure email prototype, CoPilot, color-codes messages depending on whether they were signed and whether the signer was previously known or unknown. This interface makes users significantly less susceptible to social engineering attacks overall, but new-identity attacks (from email addresses never seen before) are still effective. Also, naïve users *do* use the Sign and Encrypt button on the Outlook Express toolbar when the situation seems to warrant it, even without explicit instruction, although some falsely hoped that Encrypt would protect a secret message even when sent directly to an attacker. We conclude that KCM is a workable model for improving email security today, but work is needed to alert users to "phishing" attacks.

Categories and Subject Descriptors

D.4.6.c [**Security and Privacy Protection**]: Cryptographic Controls; H.5.2.e [**HCI User Interfaces**]: Evaluation/methodology

General Terms

Usability, Security

Keywords

User Studies, E-Commerce, User Interaction Design

1. INTRODUCTION

After more than 20 years of research, cryptographically protected email is still a rarity on the Internet today. Usability failings are commonly blamed for the current state of affairs: programs like PGP and GPG must be specially obtained, installed, and are generally considered hard to use. And while support for the S/MIME mail encryption standard is widely available, procedures for obtaining S/MIME certificates are onerous because of the necessity of verifying one's identity to a Certification Authority.

Key Continuity Management (KCM) [5] has been proposed as a way around this conundrum. Under this model, individuals would create their own, uncertified S/MIME certificates, use these certificates to sign their outgoing mail, and attach those certificates to outgoing messages. Correspondents who wish to send mail that is sealed with encryption are able to do so because they posses the sender's certificate. Mail clients (e.g. Outlook Express, Eudora) alert users when a correspondent's certificate changes.

We conducted a user test of KCM with 43 crypto-naïve users. Using a scenario similar to that of Whitten and Tygar's *Why Johnny Can't Encrypt* [15] study, we show that naïve subjects generally understand the gist of digitally signed mail, and further understand that a changed key represents a potential attack. However, such subjects are less equipped to handle the circumstances when a new email address is simultaneously presented with a new digital certificate.

We conclude that KCM is a workable model that can be used today to improve email security for naïve users, but that work is needed to develop effective interfaces to alert those users to a particular subset of attacks.

 Symposium On Usable Privacy and Security (SOUPS) 2005, July 6-8, 2005, Pittsburgh, PA, USA

2. BACKGROUND

In their seminal 1976 paper disclosing the invention of public key cryptography [1], Diffie and Hellman wrote somewhat optimistically that their invention "enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it." Diffie and Hellman proposed that public keys would be placed in "a public directory." The following year (1977), Rivest, Shamir and Adelman introduced what has come to be known as the *RSA Cryptosystem*, an algorithm that provided a practical realization of the kind of public key system that Diffie and Hellman foresaw. In 1978 Loren Kohnfelder proposed in his MIT undergraduate thesis [6] that certificates could be used as a more efficient and scalable system for distributing public keys.

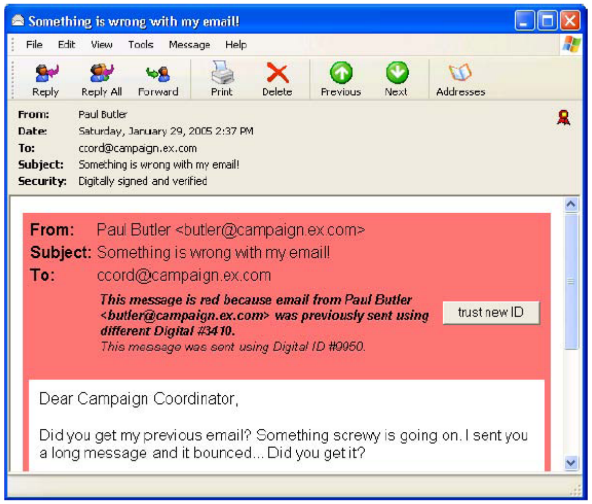
With these three inventions—public key cryptography, the RSA algorithm, and certificates—the basic building blocks for a global secure messaging system were in place. Yet more than 20 years later, after the deployment of a global Internet and the creation of low-cost computers that can perform hundreds of RSA operations in the blink of an eye, the vast majority of the legitimate (ie: non-spam) mail sent over the Internet lacks any form of cryptographic protection.

S/MIME and KCM

- S/MIME
  - Automatically attaches certificate (with public key) of user whose private key encrypted (signs) outgoing email
  - Automatically decrypts received email which has an attached certificate and stores certificate in address book
  - Obtaining a certificate is still difficult (requires trusted third party, certificate chains)
- KCM
  - Ignore certificate chains ("users are on their own"); directly associate identity in certificate with public key in certificate for email purposes
  - Notify user if public key changes for that identity
  - Tradeoff: less secure but more usable and scalable
  - Added to Eudora mail client via CoPilot

Attacks and Feedback

- Anticipated attacks
  - New key attack (trust email which has a key different from one seen previously?)
  - New identity attack (trust new key and new identity)?
  - Unsigned message attack (trust unsigned message from known source)
- Feedback (message border color-coded)
  - Red (message contains new key from known identity)
  - Yellow (first signed message from identity)
  - Green (current message signed with known key)
  - Gray (unsigned message from identity with known key)
  - White (unsigned message from unknown identity)



Experiment

- Three groups
  - No KCM
  - KCM
  - KCM + briefing
- Results
  - KCM help users to resist attacks
  - Users were able to explain signing and sealing in follow-up interviews, however…
  - KCM users are less likely to encrypt message than those without KCM (apparently not understanding the difference between sealing and signing despite results of interviews)

Cohort	n	% subjects resisting attacks: sometimes	% subjects resisting attacks: always	Clicked "encrypt" to seal email: sometimes	Clicked "encrypt" to seal email: always
No KCM	14	43%	0%	50%	21%
Color	14	50%	29%	36%	36%
Color+Briefing	15	87%	33%	20%	13%
χ^2		6.13	3.61	2.96	0.29
$p =$		0.013	0.57	0.087	0.59

Table 3: Summary Results of Johnny 2 User Study

Attack Types

	% of subjects that tried to send the schedule when requested by:					
Group	Maria 1	Maria 2	Ben	new key attack	new identity attack	unsigned message attack
No KCM	100% (14/14)	92% (11/12)	100% (14/14)	71% (10/14)	79% (11/14)	75% (9/11)
Color	93% (13/14)	100% (13/13)	92% (11/12)	64% (9/14)	50% (7/14)	58% (7/12)
Color+Briefing	100% (13/15)	100% (14/15)	100% (13/14)	13% (2/15)	60% (9/15)	43% (6/14)
χ^2	2.20 $p = 0.14$	0.018 $p = 0.89$	0.79 $p = 0.37$	10.61 $p = 0.001$	1.02 $p = 0.31$	3.98 $p = 0.046$

- KCM more successful against new key attack and unsigned message attacks
- KCM not more successful against new identity attack

Reflections

- To what extent is "technology" the answer? (What is the difference between the user performance in the two experiments?)
- Does usability engineering for security require a different set of methods/tools?
- Is a tradeoff between security and usability required (as suggested in the use of KCM)?
- Importance of repeatability.
- Utility of an experimental framework (the "*Johnny2* Experimenter's Workbench").