Security and Usability Engineering with Particular Attention to Electronic Mail

Volker Roth [a] Tobias Straub [b] Kai Richter [c]

[a] *OGM Laboratory LLC*
Omaha, NE 68106, USA

[b] *Technische Universität Darmstadt*
Hochschulstr. 10, 64289 Darmstadt, Germany

[c] *Zentrum für Graphische Datenverarbeitung e.V.*
Fraunhoferstr. 5, 64283 Darmstadt, Germany

Abstract

Support for strong electronic mail security is widely available yet only few communicants appear to make use of these features. Apparently, the operational overhead of security outweighs its perceived benefits. Towards increasing the benefits versus overhead ratio we follow an approach that considers security and usability tradeoffs from the outset. We separate key exchange from binding keys to identities. The best effort key exchange and key maintainance scheme that we devise operates transparently for the user. We also describe complementary visualization and interaction techniques that communicate the security state of sent and received mail to users in a non-intrusive fashion. Towards a practical assessment of the overheads of binding keys to identities, we conducted a quantitative analysis of users' mail behavior of which we present the results. We argue that for individual non-commercial users, out-of-band verification of keys could be more economical than building trust in public key certificates issued by third parties.

Key words: Security engineering, usability and security, secure electronic mail, human computer interaction, transparent encryption, transparent digital signatures

1 Introduction

A multitude of mail user agents ("MUA") and mail transfer agents ("MTA") with support for encryption and digital signatures, and supporting public key infrastructures ("PKI") have been—and still are—researched, developed, and deployed. However, it is generally agreed that the amount of "secure" electronic mail (briefly, secure mail) is only a negligible fraction of the total amount of mail. The obvious lack of suffi cient incentives for securing mail has two reasonable explanations.

Email address: volker.roth@acm.org (Volker Roth).

First, the *anticipated benefits* of secure mail are too low, or second, the *costs* of applying the existing security technology are too high.

The benefits appear to be low if the threat perception is weak, which is generally the case. Although many subjects may be aware of *potential* eavesdropping on their mail communication, they may dismiss that as improbable ("I'm not so important") or they may be unbothered by it ("I have nothing to hide") [20]. Consequently, the costs of security must be low as well for security to be attractive.

The costs can be measured in monetary terms or in terms of the overhead users have due to security technology. Monetary costs can be discounted since numerous free and affordable commercial mail encryption tools are available or encryption capabilities come with the software at no extra monetary cost. The costs are determined primarily by the *cognitive effort required to operate electronic mail security.* This implies that users install the necessary software and generate cryptographic keys. The difficulty lies in exchanging keys, building trust in the received keys, authorizing cryptographic operations, and in diligently and competently reacting to prompts and warnings of the security software. However, the majority of users only has limited understanding of the underlying trust models and concepts (see e.g., [13]). Consequently, they avoid or improperly operate the security software [34].

This situation is not likely to change in the foreseeable future, and leads researchers to believe that a trade-off between security and usability must be made in favor of usability. This may yield a level of security that is less than perfect, but which is still "good enough" [28]. Smetters and Grinter are rather straightforward in this respect, and ask "if you put usability first, how much security can you get?" [31]. We believe that these comments mark a profound change in the way many "secure" systems will be engineered in the future. Rather than being driven by a conservative assessment of threats and security requirements, the safety and usability of such systems will be at the focus of many low to medium risk application areas.

The research question then becomes what the optimum trade-off should be and how the security benefits are maximized with minimum damage to usability. We argue that public key certificates signed by intermediaries (e.g., certification authorities ("CA") with limited or nonexistent liability, or peers in PGP's Web of Trust model) and digital signatures account for a considerable cognitive or operational overhead, but they contribute marginally to the practical level of security. By eliminating these primitives, the costs versus benefits ratio can be improved. Towards more user--friendly secure mail, we explore a non-intrusive approach that works without either of these primitives, and rather focuses on transparent message encryption and integrity protection. In our approach, we separate key *exchange* from *binding keys to identities.*

We take an in-depth look at the implications this choice has on security and usability. Particularly, this enables us to implement a large portion of the security mechanisms in a transparent and opportunistic (e.g., best effort) fashion while binding keys to identities becomes a trade-off the user can make at his or her discretion.

Part of our approach may be compared to an intrusion detection system based on principles of cryptographic coherence: mail from a peer is secured with the same key as prior mail unless a new key is introduced, and a new key may only be used if it is introduced with a key that has been used before. Acceptance of the initial key is a "leap of faith" [4]. In a "greedy" fashion, faith can be confirmed by verifying keys in an order that maximizes the security benefits per interaction. For this purpose, a user's communication behavior and social network is monitored. One metric that we apply is the amount of mail exchanged per peer: verifying the key of the peer with whom most of the mail is exchanged has the greatest immediate quantitative benefit. Although, this is only a first step. Research in the information-theoretic and psychological foundations of applicable metrics will certainly be an interesting and rewarding endeavor.

We believe that opportunities come along to build more effective user interfaces. Rather than leading users to the Minotaur of cryptology and public key infrastructures (e.g., by educating users in the art of encryption [33]) or eliminating all direct control (e.g., by delegating all responsibilities to a transparent proxy [10,17]) the chores of secure communication can be broken down into pieces with a clearer focus and purpose.

2 Background

We begin this section by summarizing our threat model. Our engineering effort is directed at making suitable tradeoffs between security and usability. As part of our analysis what suitable tradeoffs might be, we look at the economics of contemporary defense mechanisms in the light of prudent engineering principles for the design of secure information systems [27]. The conclusions we draw from our analysis guides the design choices we make in subsequent parts of our paper.

2.1 Threat Model

We focus on *individual non-commercial users* of electronic mail (briefly, mail) e.g., Alice and Bob who communicate by mail from their homes. Both connect to the Internet through a telecommunications provider ("Telco") and an Internet service provider ("ISP"). Alice sends her mail by transacting with a mail exchanger ("MX") e.g., a *sendmail* daemon run by her mail provider. *En route* to Bob, her mail may be forwarded several times from one MX to another until it is delivered to Bob's mail drop. The mail drop stores mail for Bob until he retrieves it with his mail client e.g., by means of the POP3 or IMAP protocols or a Web front end. Figure 1 illustrates that model. We distinguish between three classic categories of threats as suggested by Anderson [2]:

Unauthorized information release: The adversary, Mallory, intercepts and reads mail sent by Alice to Bob. We do not address the threat of traffic analysis, that is beyond our scope.

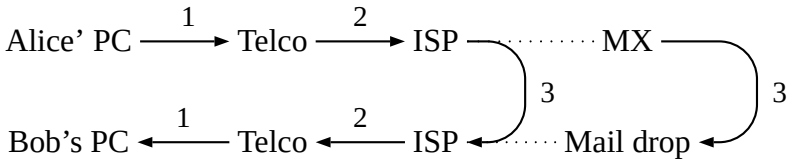

Fig. 1. The physical and logical path (dotted extension lines) of an e-mail from Alice to Bob. Each entity besides Alice' and Bob's PC can collaborate with the adversary in an active or passive attack. Additionally, the adversary may tap the communication channels between these entities (a passive attack).

Unauthorized information modification: Mallory modifies mail sent from Alice to Bob while it is in transit or in storage, or she sends mail to Bob that is forged in the name of Alice.

Unauthorized denial of use: Mallory can prevent Alice from sending mail to Bob, or can prevent Bob from receiving mail from Alice. Note that a powerful adversary can always deny service e.g., to Alice by saturating or severing her communication channels. We limit ourselves to denial of service through regular interaction with Alice's and Bob's mail system e.g., by sending them forged or unsolicited mail.

Mallory may or may not collaborate with any of the entities involved in forwarding or storing mail with their consent (e.g., implied by search warrant) or without their consent (e.g., because Mallory compromised a MX or even Bob's computer). Mallory may launch passive attacks (eavesdropping only) or active attacks (which involve the modification or generation of messages).

Although we discuss threats and countermeasures in terms of Alice and Bob, a particular *pair* or users, our intention is to address large scale attacks. In other words, Mallory attacks multiple mail "sinks" for a mail "source," multiple mail sources for a mail sink, or multiple sources and sinks.

2.2 Potential Tradeoffs

Before we discuss tradeoffs between security and usability that we have identified, we would like to briefly recite a few prudent engineering rules for designing protection mechanisms as Saltzer and Schroeder summarized them in [27]:

(1) The design should be as simple and small as possible.
(2) The mechanism should have fail-safe defaults.
(3) The mechanism should be easy to understand and use.

From an economic or rational perspective, public key certificates are worthwhile if their benefits outweigh their costs. The primary benefit of certificates is the prevention of impersonation attacks by *active* adversaries who substitute their keys for the keys of legitimate users. For comparison, public key cryptography without certificates foils *passive* eavesdroppers completely [15], and it uncovers active adversaries unless the adversary intercepts and re-encodes all mail *at all times*, particularly at the time when the communicants first exchange their keys.

In order to fulfill their objective, certificates require a complicated public key infrastructure ("PKI") which is typically backed by certification authorities with limited or nonexistent liability (compare e.g., [19,16]). Building trust in certificates, exchanging keys, and managing keys accounts for the majority of the interactions and decisions that interfere with the goal oriented tasks of a user, and which the user has difficulties to understand (compare e.g., [13,19,34]).

Since a PKI seems to be neither simple (rule 1) nor particularly easy to understand and use (rule 3), we decided to investigate what it would take to achieve adequate security without that primitive. Our objective is not to attain 100% security but to attain *significantly improved security* at a *high degree of usability*,

Our first measure was to separate concerns that would be uniquely addressed by a PKI, which is the process of binding a key to an identity. Once that was determined, other separations of concerns followed suit:

(1) exchanging keys and maintaining current keys
(2) protecting mail
(3) communicating and responding to potential security breaches
(4) *optionally* binding keys to an identity

Here, "binding" implies verification of an ad hoc association between a key and an alleged identity. For as long as the user is not concerned about active attacks, he or she may disregard the binding and trust the ad hoc association. This does not necessarily mean that the association is not trustworthy, each concern has elements that attempt to detect security breaches based on certain *coherence principles* that we define along with our mechanisms. For instance, if Alice uses public key k in her communication with Bob then she may not use public key $k' \neq k$ unless she introduced k' and proved that she knows the private keys that belong both to k and k'. A variety of more general coherence principles have been summarized nicely by Arkko and Nikander [4] under the notion of *weak authentication*.

In summary, our approach is to make key exchange and encryption a largely *transparent default behavior* and to design protocols that make it difficult for Mallory:

(1) to initially compromise a communication channel,
(2) to maintain continuous control over the channel,
(3) to leave the compromised channel undetected,

even if no binding between keys and identities is provided through a PKI. Thereby, we aim to reduce the overhead (costs) associated with the operation of mail security to a degree where mail security becomes ubiquitous. At the same time, we raise the risks and costs of successful attacks so that a rational adversary must concentrate his or her resources on a small number of targets with a high overall yield — and thus does not invade the privacy of non-commercial individual users on a large scale.

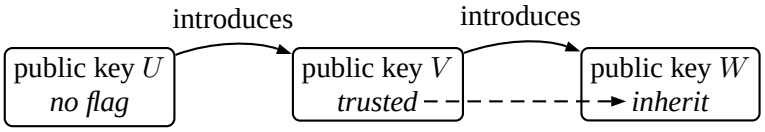

Fig. 2. Key structures are kept in doubly-linked lists. Each successor key is introduced by its predecessor. A key can be flagged as *trusted*. The flag carries forward, which means that a successor key with no flags set inherits the flag settings from its predecessor. A *trusted* flag indicates that the key has been successfully bound to an identity.

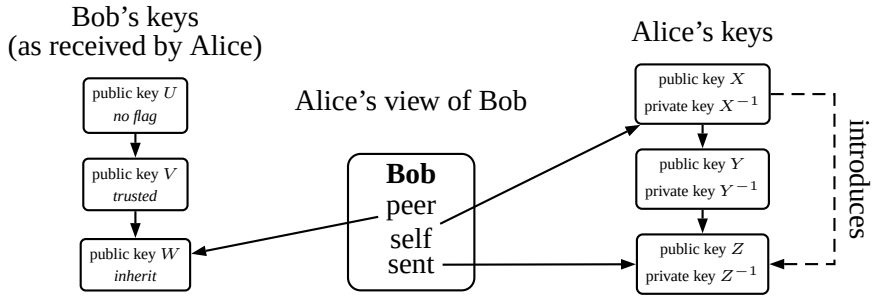

Fig. 3. Alice keeps a structure for her peers e.g., Bob. The *peer* key is the most recent key that Alice received from Bob. The *self* key is the most recent key of Alice that Alice saw Bob use. The *sent* key is the most recent key that Alice sent to Bob (but Bob may not have used that key). Whenever Alice sends a more recent key (e.g., Z) to Bob, she introduces that key using the *self* key.

3 Exchanging and Maintaining Current Keys

Our first concern is to distribute public keys that can be used to prevent unauthorized information release or modification. At this point, we are *not* concerned about the binding of a key to an identity, we cover that aspect further below in §6. That enables us to design the exchange of public keys in a fashion that is entirely transparent to users, unless of course there is evidence of a potential attack. The (fully automatic) key exchange process of Alice ("Kex") has three duties:

(1) maintain the "security state" of Alice, which is the known set of keys, peers, and their relationships;
(2) decide which keys must be used and which keys must be exchanged at any given point in time;
(3) update the security state based upon the keys that are received;

The principal idea is to design Kex such that updates of the security state (in other words a "transition" in the state space) occur only if the new state is at least as secure as the previous state. A transition is triggered if, as part of a mail, a new key is "introduced," which requires a message of the following form: [1]

$$\text{Bob} \rightarrow \text{Alice}: \ \{\text{"Bob"}, \text{"Alice"}, t, h(k_{\text{old}}), k_{\text{new}}\} \text{ signed with } k_{\text{old}}^{-1}, k_{\text{new}}^{-1}$$

[1] For simplicity, we assume that the same key pair is used for signing and encryption, as is common practice e.g., in S/MIME [26] applications. The approach can readily be extended to support dual key pairs.

Here, t is the current time, k^{-1} is the private key of Bob, and k is his public key. In practice, "Alice" and "Bob" denote Alice's and Bob's canonical mail addresses. What Bob tells Alice is that he wishes to replace his older key k_{old} with the more recent key k_{new}. As a shorthand, we write $k_{\text{old}} \succ k_{\text{new}}$, provided that the signature is valid, and $k_{\text{old}} \not\succ k_{\text{new}}$ otherwise. The first key is introduced self-signed i.e., k_{old} is empty. Old keys and the order in which they were introduced are kept (see fig. 2) until they are not referenced any longer in a peer structure and are garbage collected e.g., based on expiry time and storage quotas.

Kex intercepts incoming mail and processes any valid key introductions contained therein using three types of data (see fig. 3) that together comprise the security state:

(1) the list of Alice's key pairs ordered by recency
(2) for each peer (e.g., Bob) his associated public keys ordered by recency
(3) a peer structure for each peer

Peer structures are persistent and created on demand when the first mail of a peer is received. For instance, the peer structure that Kex keeps on Bob stores the following information:

self: the most recent key of Alice that Bob used to protect his mail to Alice;
sent: the most recent key of Alice the Kex introduced (sent) to Bob;
peer: the most recent key of Bob known to Kex;

When Kex receives $k \succ k'$ it enforces a simple rule. If no key is known for Bob then a self-signed key is accepted as Bob's initial key and an association with Bob's identity is formed. Otherwise, if k is the known key of Bob then that key introduces k' which becomes the new most recent key of Bob. Of course, the introduced key k' must not already exist in Alice's security state or else an error occurs (that prevents the inadvertent or adverse introduction of loops in the history of keys). If an error occurs then Kex tags the mail that contained the key introduction with a descriptive error code.

Kex also intercepts outbound mail and determines for each recipient whether a key must be introduced (e.g., if Bob has not yet used any of Alice's keys or uses a key that is outdated). Once Kex selected a key for introduction, it keeps introducing that key with the most recent key of Alice that Bob knows until Bob begins using the introduced key. At this point, Kex either stops adding key introductions to outbound mail or starts introducing a more recent key of Alice (if such a key has been generated meanwhile).

Figures 4 and 5 give a more precise and formal presentation of how Kex processes incoming and outgoing key introductions. The processes are designed to keep Alice's and Bob's security states synchronized even if an active adversary drops legitimate mails with key introductions, or reorders such mail.

Security Obviously, a valid key introduction can be computed only by someone who knows the private keys that correspond to both public keys, the old and the new

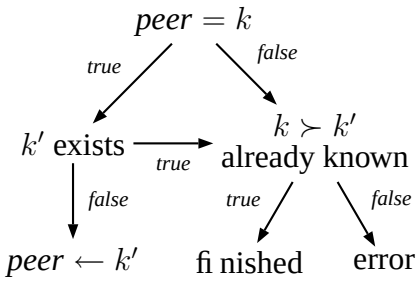

Fig. 4. Decision graph for incoming key introductions $k \succ k'$. Note that in a self-signed key introduction the old key k is nil. If no peer key is known then *peer* is also nil (compare fig. 3). This allows a very compact presentation of the graph.

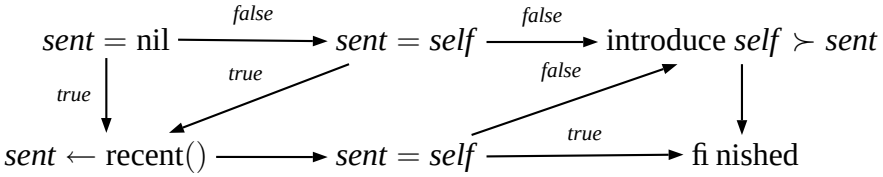

Fig. 5. Decision graph for outgoing mails. The *peer*, *self*, and *self* variables refer to the peer data structure that we introduced in fig, 3. As an invariant, the *sent* key must always be at least as recent as the *self* key.

one. Alice does not really know whether Bob or Mallory sent the key introduction, Mallory could have generated both the old and the new key. However, by simple induction we can then reason that Alice must have received the first self-signed key with Bob's name on it from Mallory (for now we disregard the possibility that Bob adversely or inadvertently disclosed his private key to Mallory or that Mallory has broken Bob's key pair). On the other hand, if Alice received the initial key introduction actually from Bob (whether or not Alice is certain about it) then Mallory cannot send valid key introductions in Bob's name unless she breaks Bob's key pair.

Assume now that Mallory sent the initial key introduction. Alice attempts to send mail to Bob in Mallory's key which triggers alarms on Bob's side unless Mallory performs a man-in-the-middle ("MITM") attack and transcodes all such mail. A single missed mail reveals Mallory's presence. Once Mallory starts her attack he cannot leave the communication channel between Alice and Bob without his presence being noticed. In order to do so, he would have to introduce Bob's public key to Alice which requires forgery of Bob's signature. Alice's advantage is that Murphy's Law is in her favor — eventually Mallory will make a mistake, perhaps due to a configuration error, a route change, a computer failure, or because Alice and Bob followed procedures we outline in §6.

Finally, assume that Mallory broke Bob's key. That means she can read Bob's mail to Alice — until Bob introduces a new key. At this point, security is either restored or Mallory must launch an active attack with all the implications discussed above.

In summary, we presented secure key exchange and key maintenance scheme that operates *continuously* and *transparently* on Alice's inbound and outbound mail.

No user interaction is required unless evidence of an attack is found. Powerful active adversaries may still attack but their chances of evading eventual detection are limited.

4 Protecting Mail

Mail consists of headers and the mail body [12]. Both elements need protection, which is achieved by transforming them into a separate *envelope* for each intended recipient. The envelope consists of a minimal set of headers and a protected body, and is sent to the recipient (e.g., Bob) via the simple mail transport protocol ("SMTP"). How exactly the envelope is represented is largely a matter of implementation choice. On an abstract level, the process is quite simple. If Bob requires a key introduction then it is added to the mail. Key introductions, headers, and mail body are jointly signed with the introduced key (the *sent* key) or otherwise with the most recent key of Alice that is available to Bob (the *self* key).

Finally, if a public key of Bob is known (*peer* $\neq$ nil) then the signed information including signature is encrypted with that key and the cipher text is placed in the body of the envelope. The information necessary for decryption (e.g., key identifiers and encrypted content encryption keys) is added where appropriate as per implementation choice. Necessary and sufficient information to fulfill mail transport requirements are copied to the envelope's header section.

If no suitable key of Bob is known then the augmented mail (and not the envelope) is sent (unencrypted) or not sent at all based on a prior policy decision that we explain in greater detail in §5 along with user interaction principles.

Inbound mail is processed in reverse order, which is a more involved process due to the various error conditions that may arise. For simplicity, let "Kin" be a shorthand for that process. If all applicable verification operations turn out correctly and the encryption key is more recent than the key previously used by Bob then Kin updates the *self* key in Alice's view of Bob accordingly. In case of errors, Kin signals them by adding appropriate headers to the mail, which must then be interpreted by the MUA. In addition to rather obvious error conditions such as invalid signatures and missing keys, there are some subtle conditions that Kin must verify. For instance, if Bob once sent mail to Alice which was signed or encrypted then any subsequent mail without that protection is flagged with an error. Otherwise, adversaries could simply bypass the security mechanisms by sending plain text mail, which is easily forged.

Due to space constraints, a detailed discussion of the processes or an exhaustive analysis of error conditions and potential responses is out of the scope of this paper. The important point that we wish to stress, though, is that protection against unauthorized manipulation or disclosure of mail contents is the *default* mode of operation. As we will explain in greater detail in §5, the user is not given the choice *not* to encrypt whenever encryption is possible. His decision is required solely when mail must be sent to recipients for whom no valid key is known.

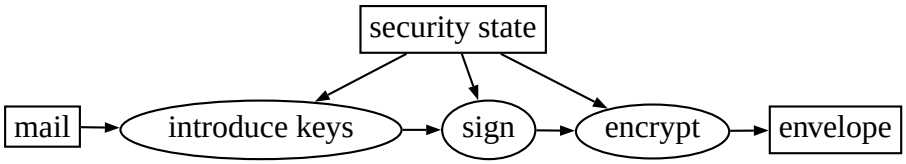

Fig. 6. The process of protecting outbound mail.

5 Key Introduction and User Interaction

Up to this point, our discussion focused on the key exchange and mail protection processes that underpin our engineering approach. Most importantly, we separated the key exchange from binding keys to identities (in a strong sense) and we made transparent signatures and encryption the default mode of operation. In this section, we discuss how that approach enables us to simplify the interaction with users.

The usability of contemporary mail security software is limited primarily by two factors:

(1) required interactions with security functions are perceived as an obstacle to accomplishing goal-oriented tasks (see e.g., Sasse et al. [29]);
(2) users have limited understanding of the concepts that underpin trust models and cryptographic primitives (compare e.g., [13,33]).

We illustrate these limitations with two examples (compare e.g., [18]). First, warning messages and confirmation boxes frequently appear and must be "clicked away" when no suitable keys are available even though the user may be well aware of the risks of sending mail in the clear and is willing to accept them. Second, users are presented unintelligible information e.g., certificate contents which are easily set to arbitrary and superficially convincing values by adversaries who generate self--signed certificates.

In contrast, our user interface design concepts require fewer interactions than contemporary approaches. We let users interact with concepts they know ("mail" rather than "keys"). Our objective is to provide a familiar context and mental representation [24]. All decisions that a user must make are related to a purpose (e.g., confidentiality) rather than unfamiliar concepts (e.g., certificates as the means to an end). Furthermore, users are not forced to make decisions immediately, they are at leisure to invoke security functions when deemed appropriate (which is influenced by users' curiosity or risk perception). We cover two user interaction aspects in this section: output (i.e, alerts and visualizing status information such as the availability of a particular key) and input (i.e., making policy decisions).

Alerts and Status Display Users (i.e., Alice) must be alerted for instance if the key exchange and maintenance processes detected incoherent key usage, invalid signatures, or other error conditions. Such alerts are associated with a particular mail that has been received. A simple and unintrusive method to alert Alice to such an event is to highlight the suspicious mail prominently in the mail display e.g.,

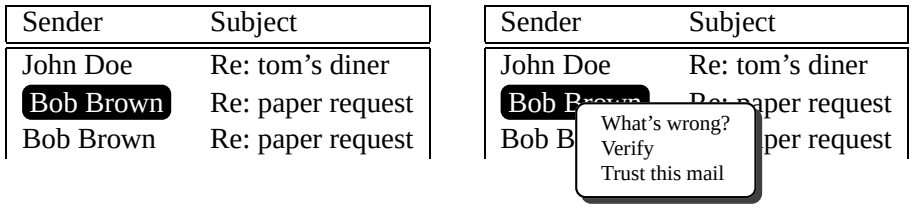

Fig. 7. Mail flagged with an error e.g., as a consequence of incoherent key usage. *Left:* the mail with the error is highlighted in reverse; *right:* a context menu is displayed upon right-clicking the flagged mail. The context menu allows to inquire about the reason of the error, to verify the association of a key and an identity, or to define the underlying key as trusted (which binds the key to the identity of the sender).

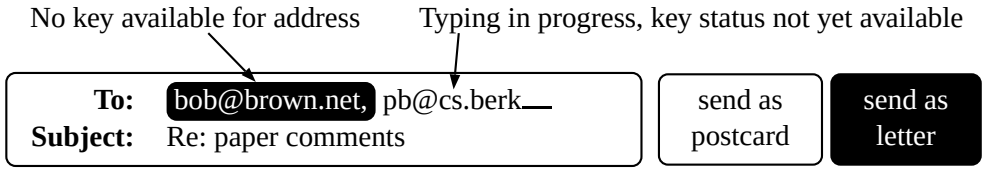

Fig. 8. Immediate feedback on key status through highlighting. *Left:* no key is known for bob@brown.net, therefore the address is highlighted. Entry of the second address is still in progress, no highlighting has taken place yet. *Right:* mail can be sent under the postcard or letter policy. Since no key is known for at least one recipient, the letter policy yields a conflict and is therefore highlighted.

as illustrated in fig. 7. For example, assume that Bob introduced his key to Alice in one mail, and Mallory subsequently sends a key introduction of her own forged key in Bob's name. However, Kex would flag Mallory's mail with an error and the mail would be highlighted to bring the fact to Alice's attention. On the other hand, if Mallory sent her mail before Bob then Bob's mail—the genuine one—would be flagged. In either case would Alice be alerted to the fact that some sort of attack is ongoing. Which key is which—and consequently which mail is genuine—can be determined by appropriate subsequent verification. However, it is upon Alice to decide whether or not to investigate the alert, and she may do so at a time of her choice. Her regular tasks remain largely uninterrupted.

Alice may investigate the reason for the alert and the conclusions that can be drawn from it e.g., by invoking a context menu (see fig. 7 for illustration). Subsequently, she may wish to invoke a verification procedure (see §6) which allows her to confirm the authenticity of the mail. It is worth noting that technically the authenticity of the *key* would be verified that the sender of the highlighted mail used. However, Alice would have the illusion that the verification refers to the mail rather than the keys. Based on the outcome of the verification procedure, Alice may define the mail (or rather the underlying key) as trusted. Once a key of Bob is trusted to be authentic (and marked appropriately in Alice's security state) that trust extends to all keys that Bob introduces with the trusted key (see fig. 2 again).

Input of Policy Decisions If Bob's key is unavailable to Alice at the time when Alice wishes to send him mail then current mail security implementations typically

alert Alice by raising an alert panel. The panel then requires that Alice confirms or cancels the send operation by a mouse click. That forces Alice to divert her attention to handling the interruption before she can proceed with her original work process. If Alice is an unexperienced user then she may even be forced to switch input modalities (e.g., from keyboard to mouse) in order to handle the event. It is less intrusive to communicate the alert by highlighting Bob's mail address appropriately while Alice enters it. For illustration, see fig. 8. Bob's address is displayed in reverse video to indicate the fact that his key is unavailable. Alice may then decide to keep or delete Bob's address (or to ignore the event) without a major interruption of her task.

Additionally, we consider analogies to the physical world when requiring input of policy decisions i.e., which alternative action is to be taken when no key is available for a mail's intended recipients. In the physical world, we send postcards or letters. Postcards are readable by anyone involved in the delivery process and consequently the expectation of confidentiality is low when sending them. Letters have been opened regularly (and re-sealed imperceptibly) for reasons of counterespionage during the Second World War [21], and there is no reason to believe that letters are not intercepted and read today. However, for all practical purposes, letters offer a higher degree of confidentiality, and consequently the users' expectations of confidentiality are higher as well. Electronic mail sent in clear text resembles postcards whereas encrypted mail resembles letters.

Consequentially, we substitute the common "Send" button by two buttons: a "Send as postcard" and a "Send as letter" button (see also fig. 8 for illustration). The former encrypts mail for all recipients whose keys are known and sends clear text to all other recipients, whereas the latter sends encrypted mail to those recipients whose keys are known and no mail to all other recipients. In either case, mail is encrypted automatically whenever a key is available. In order to prevent users from inadvertently not sending mail to a particular recipient whose key is unknown, the "Send as letter" button is highlighted if the key of one or more recipients is unknown. As in the case of handling alerts, this approach conveys all necessary information to Alice without necessarily demanding additional user interactions. Sending mail still only requires one click and no alert boxes interrupt Alice's work task subsequent to sending a mail—all security policy decisions are made beforehand.

6 Binding Keys to Identities

Above, we described how mail can be protected against unauthorized information release and modification in a fashion that is largely transparent and non-intrusive. Although, we have not obtained absolute security—a determined adversary who manages to intercept the first key exchange between Alice and Bob may launch a MITM attack against them, and may even evade detection for an unspecified period of time. The reason is that our approach does not, in a strong sense, bind a key to the identity of its legitimate owner. We expect that Alice and Bob achieve such a strong binding by verifying their keys over an authenticated channel.

A legitimate question one may ask is whether that approach is a scalable and what burden is placed upon users by it. Below, we present results of an analysis of users' mail behavior that we conducted to find an answer to said question. More precisely, we analyzed the *setup costs* and the *continuous costs* of pair-wise key verification:

(1) the setup costs are measured by the size of the peer group that a subject has when the system is introduced;
(2) the continuous costs are measured by the rate at which the subject encounters new peers.

Our results, which we summarize and interpret below, led us to believe that pair-wise key verification may indeed be scalable. Furthermore, the analysis leads to a *greedy* algorithm which maximizes the security benefit per key verification so that users can determine an optimal trade-off between their desired level of security and interaction overhead.

6.1 Materials and Methods

We developed a set of tools (using Java, Perl, MatLab, gnuplot, and Bourne Shell) to analyze mail stored in IMAP, POP3, and file system based mail drops (mbox and related formats). A scanner extracts the *To, Cc, Bcc, Reply-To, Date, References,* and *In-Reply-To* headers of all mails. Electronic mail addresses are anonymized by substituting them with a running identity number ("ID") that is the same for equal addresses. Equality of email addresses is determined by a lower-case comparison of their local part and domain [12].

Each ID stands for a potential *peer* of the mail owner (for ease of description we refer to the mail owner as Alice). A peer of Alice is a communicant with whom Alice has a bidirectional communication i.e., Alice sent mail to her peer and the peer sent mail to Alice (or vice versa). Our tools determine the set of Alice's peers according to the following rules:

(1) Alice provides her own addresses (one or many)
(2) A mail is an answer if it has a References or In-Reply-To header
(3) A sender is a communicant who sent mail to Alice
(sender $\in$ From $\cup$ Reply-To and Alice $\in$ To $\cup$ Cc $\cup$ Bcc)
(4) A *strong* sender is a communicant who replied to Alice
(sender $\in$ From $\cup$ Reply-To and Alice $\in$ To and the mail is an answer)
(5) A receiver is a communicant to whom Alice sent mail
(Alice $\in$ From $\cup$ Reply-To and receiver $\in$ To $\cup$ Cc $\cup$ Bcc)
(6) A *strong* receiver is a communicant to whom Alice replied
(Alice $\in$ From $\cup$ Reply-To and receiver $\in$ To and the mail is an answer)
(7) A peer is a strong sender, a strong receiver, or in sender $\cap$ receiver
(8) A peer is **not** a mailing list e.g., an address that includes:
majordomo@, listserv@, -request@, -list@, etc.

For each mail and each sender and receiver referenced therein, our tool generates a sample which consists of the communicant's ID and the time stamp of the mail

that is extracted from its Date header. Next, double entries are removed which may occur e.g., if mail is filed multiple times. Finally, we eliminate all samples that are not peers and all samples with an age of more than two years measured from the most recent sample.

Three cumulative distributions are computed from the remaining set of samples:

(1) percent of mail exchanged with $n \leq x$ peers;
(2) percent of weeks where mail was exchanged with $n \leq x$ peers;
(3) percent of weeks where $n \leq x$ new peers were observed.

Multiple sets of samples (one set per mailbox owner) are combined by averaging the values of the distributions. The output are the mean $\bar{y}$ for each position x and the standard deviations for samples above (s_a) and below (s_b) the mean. The results are plotted as: $f_a(x) = \bar{y}(x) + s_a(x)$, $f(x) = \bar{y}(x)$, and $f_b(x) = \bar{y}(x) - s_b(x)$.

6.2 Description of Study

We generated a Web page from which our mail drop scanning tool and instructions for its installation and configuration could be downloaded. The subjects we invited to participate in our study were colleagues, fellow researchers, and students at Technical University Darmstadt, Fraunhofer Institute for Computer Graphics, and the Peter Kiewit Institute at the University of Nebraska at Omaha. Overall 34 individuals responded by sending us their anonymized mailbox extracts. The sizes of the sample sets varied considerably as some participants were communicating significantly more by mail than others. For instance, the participant who provided the largest dataset is a system administrator responsible for an IT infrastructure that serves about 200 researchers, students, and IT personnel at spin-off companies. Other participants (including the authors, who also provided samples) are actively engaged in collaborative research and project work.

We sorted the datasets by the number of samples our tool extracted from each set and chose the 17 sets with the largest numbers of samples, expecting to obtain more reliable results than would probably have been the case if the smaller datasets were included. All retained datasets had more than 750 samples after applying our filter procedure. Table 1 lists the sizes of the extracted samples and the samples that were retained after samples of non-peers were removed.

The results of our analysis are displayed in fig. 9. We found that on average 50% of all mail was exchanged with 10 or fewer peers ($\min(x \in \mathbb{N} \mid f(x) \geq 0.5) = 10$), and with 21 or fewer peers in our "worst case" scenario ($\min(x \in \mathbb{N} \mid f_b(x) \geq 0.5) = 21$). In 50% of all weeks, the number of peers with whom subjects exchanged mail was 10 or fewer on average and 22 or fewer in the worst case ($f_b(22) \approx 0.545$). In 50% of all weeks, subjects encountered 2 or fewer peers on average ($f(1) \approx 0.47$) and 3 or fewer in the worst case scenario ($f_b(3) \approx 0.54$).

6.3 Interpretation of Results

Our analysis is based on subjects' saved mailboxes, which necessarily captures a limited view on their actual communication since they almost certainly have stored

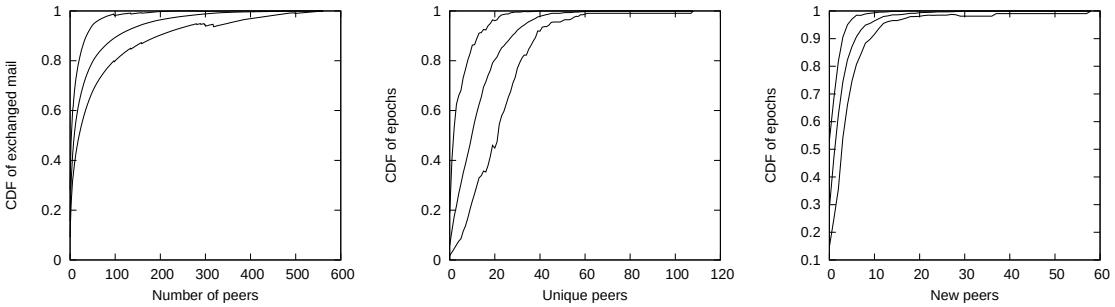

Fig. 9. Graphs of cumulative distribution functions of mail statistics. *Left:* the cumulated percentage of exchanged mails (ordinate) is plotted against the number of peers (abscissa) with whom the mails were exchanged. *Middle:* The cumulated percentage of weeks (ordinate) is plotted against the number of different peers per week (abscissa). *Right:* The cumulated number of weeks (ordinate) is plotted against the number of new peers encountered per week (abscissa). The graphs generally answer the question, whether in y percent of all cases a given condition held for x or fewer peers.

filtered	all	ratio	filtered	all	ratio	filtered	all	ratio
2651	3624	0.73	2072	3184	0.65	7657	13007	0.59
18650	29695	0.63	2766	3453	0.80	783	1727	0.45
2473	3926	0.63	3359	3785	0.89	5076	9721	0.52
2463	3225	0.76	4031	4521	0.89	1204	2259	0.53
874	1265	0.69	1604	2073	0.77	8194	11431	0.72
			8349	14484	0.58	2465	7171	0.34

Table 1
Numbers of extracted samples per dataset/user (“all”), number of selected samples which can be attributed to bidirectional communication (“filtered”), ratio between extracted and selected samples per dataset.

sent and received mail selectively. On the other hand, the type of subjects whose mail communication we analyzed uses mail more professionally and likely more extensively than private non-commercial users. It is probably fair to say that the results we obtained are an upper bound of what we could expect to see when analyzing the mail communication of private non-commercial users, a view that has been confirmed by occasional informal interviews we conducted with such users. Judging by our results, it appears that:

(1) a reasonably small number of key verifications would be required to achieve an “ideal” level of security for the majority of all exchanged mail;
(2) a fairly small number of key verifications would be required on an ongoing basis to achieve an “ideal” level of security with new peers.

Also, our results must be interpreted in the light of the actual threat level. Users may not feel—nor have—the need to verify their keys with all of their peers, which further reduces the costs of obtaining a level of security that is comparable to (or even exceeds that of) certificates issued by a certification authority.

6.4 Greedy Binding

The analysis we conducted based on subjects' saved mail can be implemented as an ongoing process e.g., in a MUA or in a mail security proxy. That process may then display on demand a ranked list of Alice's peers for whom a key verification would provide the maximum "utility." In other words, the process would suggest the peers with whom the largest amount of mail is being exchanged and with whom no key verification has yet taken place. The MUA may additionally visualize a security "score" (e.g., the percentage of mail that is exchanged securely) which may spur a special effort of the user to increase one's security so that the "high score" is reached (in analogy to playing a computer game).

Human-computer interaction ("HCI") support for key verification opens several avenues for further research and a wealth of alternatives can be explored. It is not our objective to give a complete solution in this paper, we rather wish to stimulate further research in this direction below.

For instance, Anderson and Lomas devised a scheme to fortify key negotiation with passwords by combining a collision resistant one-way hash function with a collision-rich hash function [3]. Short passwords can be exchanged easily in a telephone conversation, much easier than a 160 bits key fingerprint. While a short password does not provide the same level of security, the odds of an adversary being able to forge the protocol should be adequately small for the given threat level. Protocols with similar objectives were devised by Bellovin and Merritt [7] and Katz et al. [22]. At the time of writing, such protocols have been investigated primarily from the perspective of cryptographic strength. We argue that an investigation of usability issues of such protocols has merit. In a similar vein, user interaction support for interlock protocols based on personal entropy [15] can be explored.

Last not least, the relative rate at which users exchange mail is not the only conceivable metric to optimize the ratio of interaction and security—other behavioral or information-theoretic metrics may be applicable.

7 Related Work

The tension between security and usability has long been recognized, yet interest in that topic appears to have gained momentum recently [5,14,19,25,31,36]. Yee [36], Smetters, and Grinter [31] advocate an approach where user actions implicitly trigger security policy decisions based on the assumption that user intention conveys policy choices towards achieving a particular goal. This approach is further explored by Balfanz [5] in the Web context. We are in favor of this approach although it is of limited expressiveness in the mail context where sending or not sending mail are the primary choices. In this sense, we already stretched the limits of implicit authorization by suggesting two alternative send buttons (metaphors of mail transport) which both convey different security policies.

With respect to communicating a system's security state to users, Dourish and Redmiles [14] follow an approach that increases the amount of contextual information

a user must cope with, in the expectation that this improves his intuitive understanding of the system. We, on the other hand, seek to minimize the distraction a user faces due to security overhead. Our approach bears some similarity to the staging approach described by Whitten and Tygar [33]. Both approaches give the user the freedom of choice when to improve his security, either by verifying keys or by progressing to the next stage. However, Whitten and Tygar expect that the user eventually masters the concepts of certification, something that we avoid entirely.

Gutmann advocates that the choice of trusted certificates in a Web browser is left to the user's Internet Service Provider [19]. This simplifies the operation of Web security for the user but also reduces the security benefits when compared to the case in which the user is in control.

The *Secure Communication System* ("Secos") [1] provides secure mail and other services in a unified application. At the time of mail composition, users receive feedback whether mail can be sent securely. Secos relies on a Web of Trust augmented by a certification authority. A "best efforts" approach similar to ours is supported, although without transparent key updates or a key history.

The approach of Levien et al. [23] towards transparent Internet mail security requires a formalization of trust by certificates and explicit policies which is the opposite of what we aim to achieve. We believe that their concept is closer to the guard approach frequently encountered in military settings. Another recent instance of the guard approach is described by Wolthusen [35]. Our approach also differs from the one of Schneier and Hall [30], which accounts for public keys with a limited lifetime, but limits its discussion of key management to password disciplines. Recently, Boneh and Franklin [9] presented a practical identity-based encryption ("IBE") scheme. However, IBE requires a trusted party for the computation of private keys. This again has the type of trust implications that we seek to avoid.

Arkko and Nikander [4] formalize the notion of weak authentication and discuss the economic impacts from the attacker's point of view. The secure shell (SSH) is a typical example which follows the leap-of-faith method in that it alerts users of key changes. However, SSH does not support transparent roll-over of keys. Applications of weak authentication are discussed in the ad hoc wireless networks [32] and the IPv6 community [11].

Bentley et al. [8] developed a security patch for sendmail which opportunistically encrypts SMTP sessions between MTAs, yet does not provide end-to-end security. Stream [17] uses SMTP and POP proxies that transparently handle encryption on behalf of the MUA. It thus provides end-to-end security with a zero user interface. However, Stream does not support digital signatures nor key updates. The Enigma system of Brown and Snow [10] has a similar proxy architecture. Enigma adheres to the PGP format in contrast to Stream and our system which both encode all cryptographic information in proprietary email headers.

8 Conclusions and Future Work

It has been widely acknowledged that security is as much a human factors problem as it is a technical one, and some authors even suggest that retrofitting usability to existing security mechanisms is no more likely to be successful than retrofitting security mechanisms to existing applications [31].

In this paper, we presented an innovative design approach towards non-intrusive secure mail which considers security and usability trade-offs for a particular group of users (private non-commercial users of mail) from the outset and is closely modeled along the lines of prudent engineering rules for designing protection mechanisms [27]:

The design should be as simple and small as possible: A major source of technical and operational complexity—public key infrastructures or a Web of Trust—are avoided in our approach. We separate key exchange (which is performed transparently) from binding keys to identities (which becomes a trade-off between usability and security that the user can make at his own discretion).

The mechanism should have fail-safe defaults: Mail is encrypted transparently and automatically whenever possible. The user must only decide how mail is handled for recipients for whom no suitable key is known. Only active attacks will succeed, and only if the adversary launches a MITM attack prior to the first mail exchange. Even in the case of a successful attack, it is infeasible for the adversary to leave the communication channel undetected.

The mechanism should be easy to understand and use: Required user interaction is kept at a minimum and is unintrusive. The user (i.e., Alice) has the choice to continue her work task uninterrupted. All interaction metaphors are based on concepts and mental representations with which Alice is familiar. Based on statistics of her mail communication, she can elevate her security (e.g., detect MITM attacks) in a greedy fashion: the software recommends peers with whom a key verification has the greatest immediate security "utility."

Our approach also adheres to prudent practice as postulated e.g., by Norman [24] and Balfanz et al. [6]. Our main thesis, the low comparative benefit/cost ratio of certification infrastructures for private non-commercial users, and the potential scalability of pair-wise key verification, is supported by the reduction in user interactions that our approach achieves as well as by the results of a study of users' mail communication, the results of which we presented in the paper. We conclude that security and usability of mail is not necessarily mutually exclusive if best practices in the design of secure and usable systems are adhered to dilligently.

We regard three directions for future research as particularly promising: first, innovative human-computer interaction support for usable key verification; second, finding alternative and improved metrics to maximize users' security per interaction in a customized and adaptive fashion; third, innovative approaches to provide user interface "incentives" which promote ubiquitous "best effort" security as a base level of (mail) protection.

Acknowledgments

We would like to thank all individuals who supported our study and allowed us to analyze their anonymized mailbox contents.

References

[1] Secure Communication System. `http://www.secos.org`, Jan. 2005.

[2] ANDERSON, J. Information security in a multi-user computer environment. *Advances in Computers* (1973), 1–35. New York: Academic Press.

[3] ANDERSON, R., AND LOMAS, T. Fortifying key negotiation schemes with poorly chosen passwords. *Electronics Letters 30*, 13 (June 1994), 1040–1041.

[4] ARKKO, J., AND NIKANDER, P. How to authenticate unknown principals without trusted parties. In *Proc. Security Protocols Workshop 2002* (Cambridge, UK, April 2002).

[5] BALFANZ, D. Usable access control for the world wide web. In *Proc. Nineteenth Annual Computer Security Applications Conference* (Dec. 2003), IEEE, pp. 406–415.

[6] BALFANZ, D., DURFEE, G., GRINTER, R., AND SMETTERS, D. K. In search of usable security: Five lessons from the field. *IEEE Security and Privacy 2*, 5 (September/October 2004), 19–24.

[7] BELLOVIN, S. M., AND MERRITT, M. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy* (1992), pp. 72–84.

[8] BENTLEY, D., ROSE, G., AND WHALEN, T. ssmail: Opportunistic encryption in sendmail. In *Proc. Thirteenth Systems Administration Conference (LISA XIII)* (Seattle, WA, USA, Nov. 1999), USENIX Association.

[9] BONEH, D., AND FRANKLIN, M. Identity based encryption from the Weil pairing. *SIAM J. of Computing 32*, 3 (2003), 586–615.

[10] BROWN, I., AND SNOW, C. R. A proxy approach to e-mail security. *Software – Practice and Experience 29*, 12 (1999), 1049–1060.

[11] CASTELLUCCIA, C., MONTENEGRO, G., LAGANIER, J., AND NEUMANN, C. Hindering eavesdropping via IPv6 opportunistic encryption. In *ESORICS* (2004), pp. 309–321.

[12] CROCKER, D. H. Standard for the format of ARPA Internet text messages. Request for Comments 822, Internet Engineering Task Force, August 1982.

[13] DAVIS, D. Compliance defects in public key cryptography. In *Proc. 6th USENIX Security Symposium* (1996), pp. 171–178.

[14] DOURISH, P., AND REDMILES, D. An approach to usable security based on event monitoring and visualization. In *Proc. New Security Paradigms Workshop* (Virginia Beach, VA, USA, Sept. 2002), ACM, pp. 75–81.

[15] ELLISON, C. Establishing identity without certification authorities. In *Proc. Sixth USENIX Security Symposium* (July 1996).

[16] ELLISON, C., AND SCHNEIER, B. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal 16*, 1 (2000), 1–7. Available online at URL `http://www.counterpane.com/pki-risks.html`.

[17] GARFINKEL, S. Enabling email confidentiality through the use of opportunistic encryption. In *National Conference on Digital Government Research* (Boston, MA, May 2003).

[18] GRINTER, R. E., AND SMETTERS, D. Three challenges for embedding security into applications. [25].

[19] GUTMANN, P. Plug-and-play PKI: A PKI your mother can use. In *Proc. 12th USENIX Security Symposium* (Washington, DC, USA, Aug. 2003).

[20] K. KARVONEN, L. CARDHOLM, S. K. Cultures of trust: A cross-cultural study on the formation of trust in an electronic environment. In *Proceedings of the 3rd Nordic Workshop on Security (NordSec 2000)* (Reykjavik, Iceland, October 2000).

[21] KAHN, D. *The Codebreakers*, 2nd ed. Scribner, 1996.

[22] KATZ, J., OSTROVSKY, R., AND YUNG, M. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT* (2001), pp. 475–494.

[23] LEVIEN, R., MCCARTHY, L., AND BLAZE, M. Transparent Internet e-mail security. Tech. rep., AT&T Laboratories, Murray Hill, NJ 07974, 1996. Draft version.

[24] NORMAN, D. *The Psychology of Everyday Things*. Basic Books, USA, 1988.

[25] PATRICK, A., LONG, C., AND (ORGANIZERS), S. F. Workshop on human-computer interaction and security systems at acm chi 2003. Web pages at URL `http://www.andrewpatrick.ca/CHI2003/HCISEC/index.html`, Apr. 2003.

[26] RAMSDELL, B. S/MIME Version 3 Message Specification. Request for Comments 2633, Internet Engineering Task Force, June 1999.

[27] SALTZER, J. H., AND SCHROEDER, M. D. The protection of information in computer systems. *Communications of the ACM 17*, 7 (July 1974).

[28] SANDHU, R. Good-enough security. *IEEE Internet Computing* (Jan. 2003), 66–68.

[29] SASSE, M. A., BROSTOFF, S., AND WEIRICH, D. Transforming the "weakest link:" a human-computer interaction approach to usable and effective security. *BT Technical Journal 19*, 3 (July 2001), 122–131.

[30] SCHNEIER, B., AND HALL, C. An improved e-mail security protocol. Tech. rep., Counterpane Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419, 1997.

[31] SMETTERS, D. K., AND GRINTER, R. E. Moving from the design of usable security technologies to the design of useful secure applications. In *Proceedings of the New Security Paradigms Workshop* (Virginia Beach, VA, USA, Sept. 2002), ACM, pp. 82–89.

[32] STAJANO, F., AND ANDERSON, R. J. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proc. 7th International Security Protocols Workshop* (1999), pp. 172–194.

[33] WHITTEN, A., AND TYGAR, J. Safe staging for computer security. [25].

[34] WHITTEN, A., AND TYGAR, J. D. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proc. 9th USENIX Security Symposium* (August 1999).

[35] WOLTHUSEN, S. A Distributed Multipurpose Mail Guard. In *Proceedings from the Fourth Annual IEEE SMC Information Assurance Workshop, United States Military Academy* (West Point, NY, USA, June 2003), IEEE Press, pp. 258–265.

[36] YEE, K.-P. User interaction design for secure systems. In *Proc. 4th International Conference on Information and Communications Security* (Singapore, Dec. 2002), R. Deng, S. Qing, F. Bao, and J. Zhou, Eds., vol. 2513 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 278–290. ISBN 3-540-00164-6.