2016-07-01

The Quest to Secure Email: A Usability Analysis of Key Management Alternatives

Jeffrey Thomas Andersen
Brigham Young University

The Quest to Secure Email: A Usability Analysis of Key Management Alternatives

Jeffrey Thomas Andersen

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Kent Seamons, Chair
Daniel Zappala
Brian Morse

Department of Computer Science

Brigham Young University

July 2016

ABSTRACT

The Quest to Secure Email: A Usability Analysis
of Key Management Alternatives

Jeffrey Thomas Andersen
Department of Computer Science, BYU
Master of Science

The current state of email security is lacking, and the need for end-to-end encryption of email is clear. Recent research has begun to make progress towards usable, secure email for the masses (i.e., novice users without IT support). In this paper, we evaluate the usability implications of three different key management approaches: PGP, IBE, and passwords. Our work is the first formal A/B evaluation of the usability of different key management schemes, and the largest formal evaluation of secure email ever performed. Our results reveal interesting inherent usability trade-offs for each approach to secure email. Furthermore, our research results in the first fully-implemented PGP-based secure email system that has been shown to be usable for novice users. We share qualitative feedback from participants that provides valuable insights into user attitudes regarding each key management approach and secure email generally. Finally, our work provides an important validation of methodology and design principles described in prior work.

Keywords: Secure email, key management, usability testing

ACKNOWLEDGMENTS

I would like to thank Scott Ruoti and Tyler Monson for their work on this key management study, and the BYU Usability Center for the use of their facilities. I would also like to acknowledge the administrative support provided by the Computer Science graduate program manager, Jen Bonnett. I thank Kent Seamons for his support in advising me throughout my program, and Scott Ruoti for his mentorship. I would also like to thank my family for their support, and my uncle, Iroh, for his timeless wisdom.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

When email was first developed in 1971, no attention was paid to secure it. In recent years, there has been a strong push to address this by adopting technologies that enhance the security of email during transmission. Still, recent research has shown that these approaches (i.e., STARTTLS, SPF, DKIM, DMARC) are often configured incorrectly and have weaknesses that can be exploited by attackers [7, 8, 11]. Even if these technologies were properly deployed and configured, they do nothing to protect email at rest or while it is being processed by an email server.

These factors motivate the need for secure email that uses end-to-end encryption.[1] In secure email, the sender encrypts email messages before transferring them to the email provider, ensuring that no one but the intended recipients are able to read the messages. While secure email protocols such as PGP and S/MIME are nothing new, adoption by the masses has been nearly non-existent.[2]

One explanation for this poor adoption is secure email's long history of usability issues [16, 24, 30]. Recent research has begun making progress towards usable, secure email for the masses. Ruoti et al. evaluated design principles for creating a highly-usable version of IBE-based secure email [16, 20].[3] Similarly, Atwater et al. [1] explored how PGP-based secure email could be made more usable.

[1] End-to-end encryption of email refers to content-based encryption of email as opposed to connection-level encryption (e.g., TLS, HTTPS).

[2] We note that S/MIME is widely used in certain organizations (e.g., US government), but this adoption has not spread to the masses.

[3] Identity-based encryption (IBE) is described in more detail in Section 3.3.

In this paper, we build upon this prior research and explore how different key management approaches affect the usability of secure email. To this end, we build and evaluate three secure email systems, each based on a different key management approach: PGP, IBE, and passwords. The systems largely have identical functionality and interfaces, different only as required by their particular key management scheme. Moreover, these systems incorporate all prior research into usable, secure email, giving each system its best chance at succeeding. To evaluate the systems, we conduct a within subjects, paired-participant [19] A/B user study that involved 47 participant pairs (94 total participants) using all three systems, the largest study of secure email to date.

The results of our study show that users find both our PGP- and IBE-based secure email systems to be highly usable. This is the first time a fully-implemented PGP-based secure email system has been shown to be usable for novice users. Our study also reveals interesting details on user attitudes regarding secure email.

The contributions of this paper are,

1. **First A/B comparison of usability of key management in secure email.** In this paper, we conduct a formal A/B comparison of three different secure email key management approaches: PGP, IBE, and passwords. Our results demonstrate that each of these approaches are viable for novice users, though passwords are rated as slightly less usable. We also evaluate participants' qualitative responses and identify several intrinsic usability trade-offs between each system.

2. **First empirically verified, usable, PGP-based secure email system.** Early examinations of PGP-based secure email found it to be unusable [24, 30]. More recently, research has made progress towards usable, PGP-based secure email, but the studies of these systems did not correctly simulate the experience of a novice user. In this paper, we use a fully-implemented system and a formal paired-participant methodology [19] to accurately evaluate the ability of novices to use PGP-based secure email. Our results demonstrate that participants viewed our PGP-based system as highly usable, with

nearly a third of participants preferring it over our IBE- and password-based secure email systems.

3. **User attitudes regarding secure email.** Our study elicits user attitudes regarding the three key management approaches we evaluate. This includes security and usability trade-offs identified by participants. For example, even after understanding that PGP provides more security than IBE, many users indicate that they do not need that level of security and prefer IBE because they don't have to wait for the recipient to first install the system. Participant responses also reveal attitudes regarding secure email generally, such as the fact that many participants will only feel comfortable installing a secure email system if they know it has been verified by security-conscious individuals.

4. **Validation of prior research.** Recent research has proposed several design principles for making secure email usable for novices. In this paper we implement principles described by Atwater et al. [1] and Ruoti et al. [16, 20]. The positive results of our user studies provide validation of these design principles. More particularly, our work demonstrates that the design principles described by Ruoti et al. are generally applicable, and not just limited to IBE-based secure email. Finally, we replicate Ruoti et al.'s paired-participant methodology and provide further evidence that it has significant benefits over traditional methodologies where a study coordinator simulates one end of an email conversation.

Chapter 2

Background

In this section we first describe the current state of email security. We then describe the threat model for secure email. Finally, we discuss related work on analyzing the usability of secure email.

2.1 Email Security

When email was first designed in 1971[1] no meaningful attention was paid to security. As such, it was originally trivial for an attacker to steal email during transit or to send messages with falsified sender information. In recent years, there have been attempts to patch security into email. For example, TLS is now used to protect email during transmission, and DKIM and DMARC are used to authenticate the sender of an email. However, the deployment of these technologies is limited and they are often misconfigured.

In an analysis of email delivery security (i.e., TLS, DKIM, DMARC, SPF), Durumeric et al. found that a majority of email is still vulnerable to attack [7]. They showed that only 35% of SMTP servers are configured to use TLS, and these servers are often vulnerable to a downgrade attack. Similarly, they demonstrated that the adoption of DKIM and DMARC are so low that they provide no practical benefits. These results were further confirmed by concurrent work by both Foster et al. [8] and Holz et al. [11].

As such, email is still an easy target for attackers. For example, Durumeric et al. found that in seven countries over 20% of inbound Gmail messages are being stolen [7].

[1] `http://openmap.bbn.com/~tomlinso/ray/firstemailframe.html`

Additionally, the inability to authenticate the sender of an email increases the likelihood of email phishing, a multi-billion-dollar problem.[2] Perhaps most troubling, even if TLS, DKIM, and DMARC were to be widely adopted and configured correctly, these technologies do nothing to protect email when at rest.[3]

End-to-end encryption of email solves each of the above problems. Namely, by encrypting her email with Bob's public key, Alice is sure that only Bob can read her email. Similarly, if Alice has signed the email with her private key, Bob can verify that the email actually came from Alice. The most common forms of public key encryption are PGP, S/MIME, and IBE. Descriptions of PGP and IBE are given in Section 3.

2.2 Threat Model

In the threat model for secure email there are four possible entities:

1. **User** — The user's computer, operating system, and secure email software are considered part of the trusted computing base.

2. **Email provider** — The email provider can be treated as either fully-malicious, honest-but-curious,[4] or *sometimes-malicious*. We define a sometimes-malicious entity as one that is for the most part honest-but-curious, but from time to time can also take malicious action and collude with other sometimes-malicious entities. The sometime-malicious model is helpful for two reasons. First, this is a relatively accurate model of the largest email providers, which are unlikely to attack users' security unless forced to do so by an outside entity (e.g., court order). Second, this allows us to more accurately

[2] http://krebsonsecurity.com/2016/04/fbi-2-3-billion-lost-to-ceo-email-scams/

[3] At rest, email can be stolen as the result of a breach (https://www.washingtonpost.com/world/national-security/chinese-hackers-who-breached-google-gained-access-to-sensitive-data-us-officials-say/, a malicious insider (http://gawker.com/5637234/gcreep-google-engineer-stalked-teens-spied-on-chats, or a subpoena (https://en.wikipedia.org/wiki/PRISM_(surveillance_program), http://www.law360.com/articles/488725/post-snowden-google-report-shows-data-requests-growing).

[4] An honest-but-curious entity will gather any information available to them (e.g., Gmail scans email messages), but will not attempt to break the secure email system (e.g., impersonating the user to the key server) or collude with other honest-but-curious parties).

analyze systems that only transiently rely on the email provider, such as using email to verify a user's identity before they are allowed to post a public key to a key directory.

3. **Key server (optional)** — Some secure email schemes rely on the use of a trusted third-party key server, which—similar to the email provider—can be treated as either fully-malicious, sometimes-malicious, or honest-but-curious. The key server can be responsible for the generation and storage of key pairs (i.e., key escrow), or it can act as a lookup directory for individuals to find public keys (i.e., key directory). In either case, the reliance on this trusted third-party reduces the overall security of the secure email system. Still, we do note that there are methods for reducing potential harm from a sometimes-malicious third-party key server (e.g., thresholded-IBE [12], CONIKS [13]).

4. **Adversary** — The adversary is free to eavesdrop on any communication between users, email providers, and key servers.[5] Additionally, the adversary can attempt to compromise the email provider or key server. The adversary wins if she is able to use these resources to access the plaintext contents of the encrypted email body.

We do not consider attacks directly against the user or trusted computing base (i.e., phishing credentials, installing malicious software). Similarly, we do not consider an attacker who can compromise fundamental networking primitives (i.e., TLS, DNS). While these are valid concerns, if the attacker can accomplish these types of attacks, they can already do far more damage than they could by breaking the secure email system.[6] We also note that data needed by the email provider to transmit email (e.g., recipient addresses) cannot be encrypted, and may be available to the adversary (e.g., this information may be passed over an unencrypted channel). Our threat model instead focuses on ensuring that the data in the encrypted body is safe from an attacker.

[5] In nearly all cases, this communication will be encrypted using TLS, and the adversary only has access to the encrypted packets.

[6] For example, if an attacker can arbitrarily compromise TLS they can replace all software downloaded by a user, including the secure email system, with versions that contain malware.

To steal the user's sensitive data, the adversary must obtain both the encrypted email and the key material needed to decrypt the email. The former can be accomplished by either compromising the email provider, or intercepting an encrypted email that is not transmitted using TLS. The latter can be accomplished by users revealing this information, or in the case of key escrow by compromising the key escrow server. In that case, just as the adversary must collect the data from both the email provider and key escrow server, neither of these parties alone has enough information to unilaterally steal the user's sensitive data.

When classifying the security of a system against this threat model, there are four possible classification: satisfies the threat model when the third-party entities—email provider and optional key server—are fully-malicious, satisfies the threat model when those entities are sometimes-malicious, satisfies the threat model when those entities are honest-but-curious, and does not satisfy the threat model. We prefer this fine-grained classification, as it allows more precision in discussing the security of a system. This is important because a system that only protects against some adversarial models (e.g., honest-but-curious) may be sufficient for some use cases and have higher usability than systems which protect against more malicious models.

2.3 Related Work

Whitten and Tygar [30] conducted the first formal user study of a secure email system (i.e., PGP 5), which uncovered serious usability issues with key management and users' understanding of the underlying public key cryptography. They found that a majority of users were unable to successfully send encrypted email in the context of a hypothetical political campaign scenario. The results of their study took the security community by surprise and helped shape modern usable security research.

Seven years later, Sheng at al. demonstrated that despite improvements made to PGP (i.e., PGP 9), key management was still a challenge for users [24]. Furthermore, they showed that in the new version of PGP, encryption and decryption had become so transparent that users were unsure if a message they received had actually been encrypted.

Garfinkel and Miller created a secure email system using S/MIME and used this system to replicate Whitten and Tygar's earlier study [10]. Their work demonstrated that automating key management provides significant usability gains compared to earlier studies that burdened users with key management tasks. Still, they observed that their tool "was a little too transparent" in how well it integrated with Outlook Express, and sometimes users failed to read the instructions accompanying the visual indicators.

Ruoti et al. used IBE to explore the design principles necessary to create usable, secure email. In their first study, they demonstrated that users strongly preferred that secure email be tightly integrated with their existing email systems [16]. In a continuation of this work, Ruoti et al. demonstrated that usability could be further enhanced by adding context-sensitive inline tutorials, artificial delays on encryption while users are instructed regarding the security of their messages, and contextual clues inserted into the underlying webmail interfaces [20]. In our systems, we adopt the features discussed by Ruoti et al. and evaluate whether they are beneficial to non-IBE-based secure email (i.e., PGP- and password-based).

Atwater et al. evaluated the usability of PGP using a mocked secure email tool that automatically generates key pairs for users, shares the generated public key with a key server, and retrieves the recipient's public key as needed. Their results showed that with these modifications, users could successfully use PGP to send and receive secure email. Unfortunately, their mock-up did not correctly simulate PGP's key management, failing to require users to wait for their recipients to establish key pairs before they could be sent email. This makes it unclear if their positive results are valid, as this is one of PGP's pain points. In our PGP-based system, we adopt many of the usability features introduced in Atwater et al.'s system, but unlike Atwater et al. we fully implement PGP key management. This allows us to test whether PGP-based secure email can be usable by novices, or whether Atwater et al.'s result was an artifact of their incomplete simulation of PGP.

Bai et al. explored user attitudes towards different models for obtaining a recipient's public key in PGP [2]. In their study, they built two PGP-based secure email systems, one that used the traditional key exchange model,[7] and one that used a registration model based on a key directory.[8] Users were provided with instructions on how to use each tool and given several tasks to complete. Afterwards, participants shared their opinions regarding the key exchange models. The results of this study showed that, overall, individuals preferred the key directory-based registration model, though they were not averse to the traditional key exchange model either. In our study we adopt the key directory model, which was found to be preferable by Bai et al. Unlike our work, Bai et al.'s study only gathered data on user attitudes regarding key management, and did not evaluate their usability.

Ruoti et al. developed a novel paired-participant methodology for evaluating the usability of secure email [19]. Unlike other methodologies that have participants interact with study coordinators, this methodology brought in pairs of participants and observed whether these users could collaboratively begin using secure email. Each pair of participants were required to know each other before the study, better simulating how grassroots adoption of secure email would likely progress. Their results showed that this methodology was preferable to past methodologies for several reasons: first, users acted more naturally during the study; second, it identified additional pain points in the systems tested, that would not have surfaced in a traditional study; third, the methodology allowed researchers to observe both a) participants who are introducing their friends to secure email, and b) participants who are being introduced to secure email. In our study, we use this methodology to evaluate the systems we built.

[7] In this model, users must manually exchange their public keys with each other. This model is based on the idea of a "web of trust."

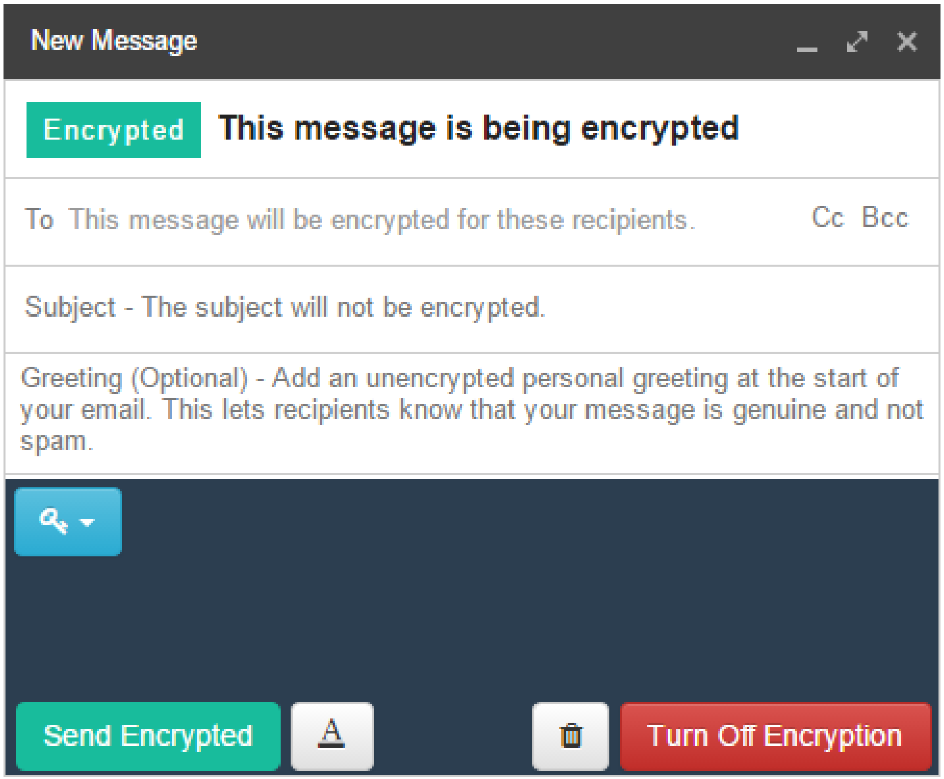
[8] In this model, users prove their identity to a trusted third-party key directory, which will then host their public key. Senders can then look up a recipient's public key in this directory.

Chapter 3

Secure Email Systems

To compare the usability of PGP-, IBE-, and password-based secure email, we developed secure email prototypes that were each implemented with one of these key management approaches. To ensure that the interface and functionality of each prototype would be largely identical, we built each using the MessageGuard platform. This allowed us to conduct an A/B evaluation of each system, which restricted variations to intrinsic properties of the key management schemes we were testing. While not described below, we also conducted several cognitive walkthroughs and pilot studies to refine the usability of MessageGuard as a whole, and each of the three secure email variants individually.

In this section, we first describe the MessageGuard platform including the overall system look and feel. We then describe the three secure email variants we developed: PGP, IBE, and Passwords. For each of these, we describe the security model of the key management scheme as well as interface elements and functionality that are unique to each version. Each of these systems is available for testing at `https://{`*`pgp,ibe,passwords`*`}.messageguard.io` and source code for each system is available at
`https://bitbucket.org/isrlemail/messageguard-WebClient`.



This message is being encrypted

To This message will be encrypted for these recipients.

Subject - The subject will not be encrypted.

Greeting (Optional) - Add an unencrypted personal greeting at the start of your email. This lets recipients know that your message is genuine and not spam.

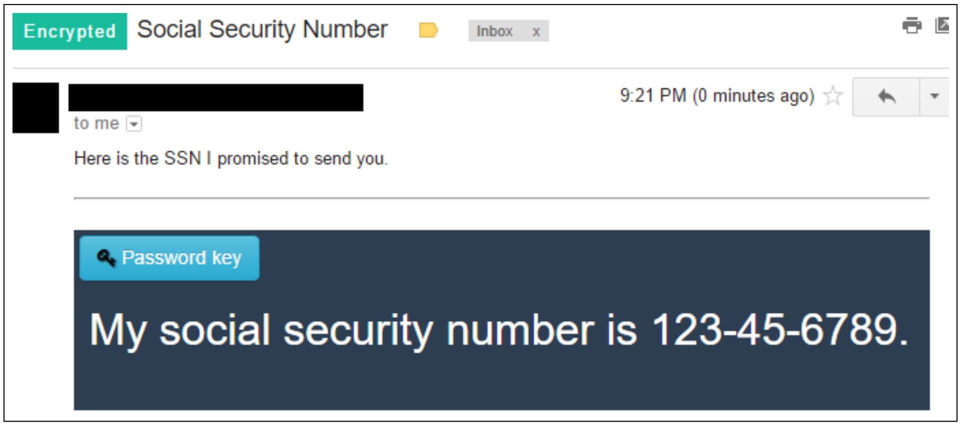
Figure 3.1: Composition Overlay in MessageGuard.

3.1 The MessageGuard Platform

The MessageGuard platform [21][1] is designed to allow researchers to rapidly prototype secure email systems.[2] These prototypes can then be deployed as browser extensions in all major browsers except IE. By building systems using MessageGuard, it is easy to ensure that the systems have overall identical interfaces and functionality, an important factor in A/B studies. MessageGuard also has support for pluggable key management, simplifying our job in developing the three secure email systems. In the remainder of this subsection we give a high level description of MessageGuard; for further details regarding its security we invite readers to refer to the MessageGuard paper [21].

[1] https://bitbucket.org/isrlemail/messageguard

[2] MessageGuard supports adding content-based encryption to most applications on the Web, and not just secure email.



Social Security Number

[redacted] [redacted]

9:21 PM (0 minutes ago)

to me

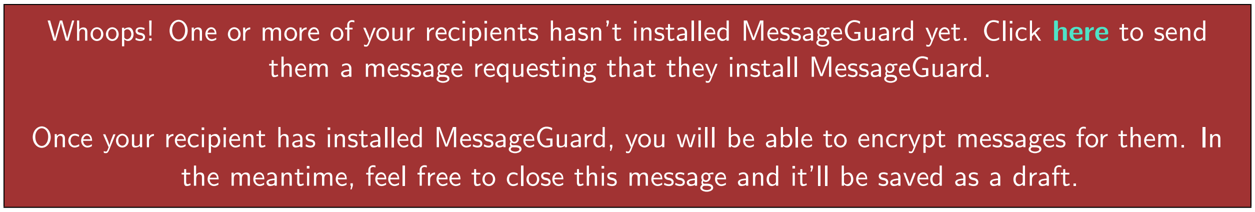
Here is the SSN I promised to send you.

My social security number is 123-45-6789.

Figure 3.2: Read Overlay in MessageGuard.

MessageGuard tightly integrates with existing web applications—in this case Gmail—using *security overlays* [28]. Security overlays function by replacing portions of Gmail's interface with secure interfaces that are inaccessible to Gmail. Users then interact with these secure overlays to create and read encrypted email (composition–Figure 3.1, read–Figure 3.2). The overlays themselves use a distinctive color scheme to help users identify them as the interfaces to use when encrypting their email.

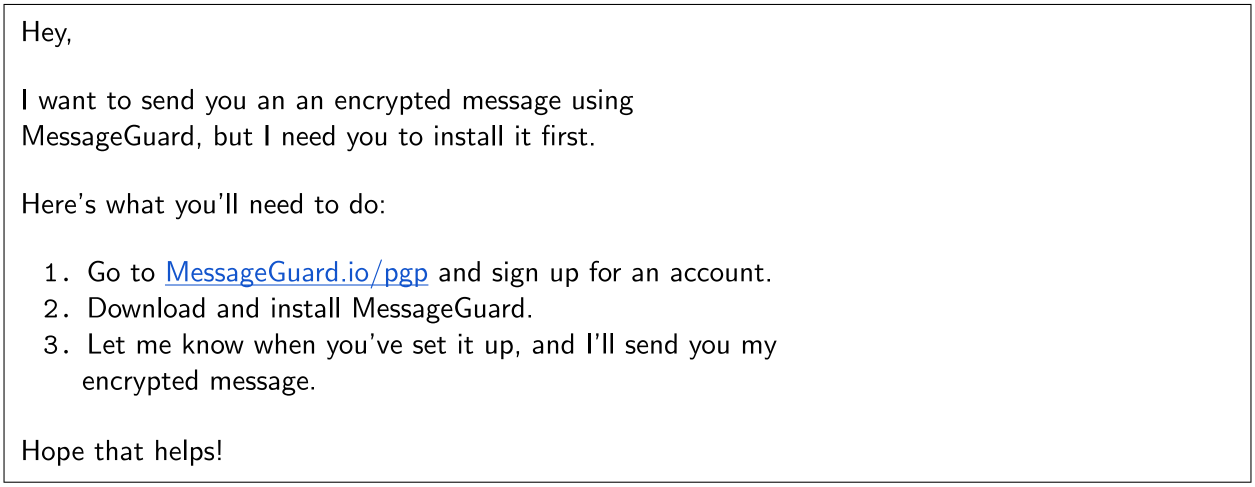
In addition to the distinctive interface, MessageGuard incorporates the design principles identified by Ruoti et al. as being necessary for secure email to be usable [16, 20]. First, when a message is encrypted, an artificial delay is added to this process; during the delay participants are shown informative text helping them understand how their message is being protected. Second, MessageGuard includes context-sensitive, inline tutorials that appear the first time a user initiates a specific task; research has shown that tutorials employing this style are more effective at getting users to read and understand them [19, 20]. Third, MessageGuard uses automatic encryption, but briefly shows users ciphertext as part of encryption and decryption, helping users feel confident that their messages have been encrypted. Fourth, users can include an unencrypted greeting with their secure email, helping their friends have confidence to install MessageGuard and decrypt the email. Fifth, encrypted emails contain information that help non-MessageGuard-users know how to set up and get started with MessageGuard.



Whoops! One or more of your recipients hasn't installed MessageGuard yet. Click **here** to send them a message requesting that they install MessageGuard.

Once your recipient has installed MessageGuard, you will be able to encrypt messages for them. In the meantime, feel free to close this message and it'll be saved as a draft.

Figure 3.3: Error Shown to PGP Users when Encrypting Message for Recipient Without Public Key Available in the Key Directory.



Hey,

I want to send you an an encrypted message using
MessageGuard, but I need you to install it first.

Here's what you'll need to do:

1. Go to MessageGuard.io/pgp and sign up for an account.
2. Download and install MessageGuard.
3. Let me know when you've set it up, and I'll send you my encrypted message.

Hope that helps!

Figure 3.4: Invitation Email Generated to Invite Recipient to Install PGP.

3.2 PGP

One of the best known approaches for providing end-to-end encryption is Pretty Good Privacy [9], better known by its acronym PGP. PGP was developed in 1991 by Phil Zimmerman, and allows users to encrypt and sign their email messages using public key cryptography. In PGP, users generate a key pair and can then share their public key in a number of ways, such as sending the key directly to other users, posting the key to a personal website, or uploading the key to a key directory.

PGP actually has a variety of valid configurations—for example, public keys can be verified using a web-of-trust, the certificate authority system, or by retrieval from a trusted key directory. In line with work by Atwater et al. and Bai et al., we chose a configuration that maximized usability, while still maintaining the most important security features: First, keys are shared using a key directory [2]; users verify that they have permission to upload a key by creating an account at the key directory and verifying their email address using email-based identification and authentication [27]. Second, a user's private key is only stored on their local computer, but it is not password encrypted [1]. Third, if a user attempts to send an email to a recipient who hasn't yet uploaded a public key to the key directory, the user is prompted to send an email to the recipient with instructions on how to set up MessageGuard [1] (see Figure 3.3 and Figure 3.4). The design of our PGP system satisfies our threat model when third-party entities are sometimes-malicious, though we do note that there would be a need to monitor the key directory (e.g., with CONIKS [13]).

The following is the workflow for our *MessageGuard–PGP* system, for a new user sending email to a non-user.

1. The user visits the MessageGuard website. They are instructed to create an account with their email address. Their address is then verified by having the user click a link in an email sent to them. They are then able to download MessageGuard–PGP.
2. After installation, the user is told that the system will generate a PGP key pair for them. The public key is automatically uploaded to the key directory, as the user is already authenticated to the key directory from the previous step.
3. The user attempts to send an encrypted email, but is informed the recipient hasn't yet installed the system (see Figure 3.3). They are then prompted to send their recipient an email inviting them to install MessageGuard–PGP (see Figure 3.4).
4. Once the recipient has installed MessageGuard–PGP, which generates and publishes their public key, they inform the sender that they are ready to proceed. The sender can now finish encrypting the email for the recipient.

3.3 IBE

Identity-based encryption (IBE) is a public key system wherein a user's public key is simply their email address [23]. Private keys are generated by a trusted third-party key server, which authenticates the identity of the user before providing them with their private key. IBE satisfies our threat model when third-party entities are honest-but-curious. As compared to PGP, IBE is less secure, but potentially more usable—IBE allows anyone to be sent secure email, regardless of whether they have already installed MessageGuard.

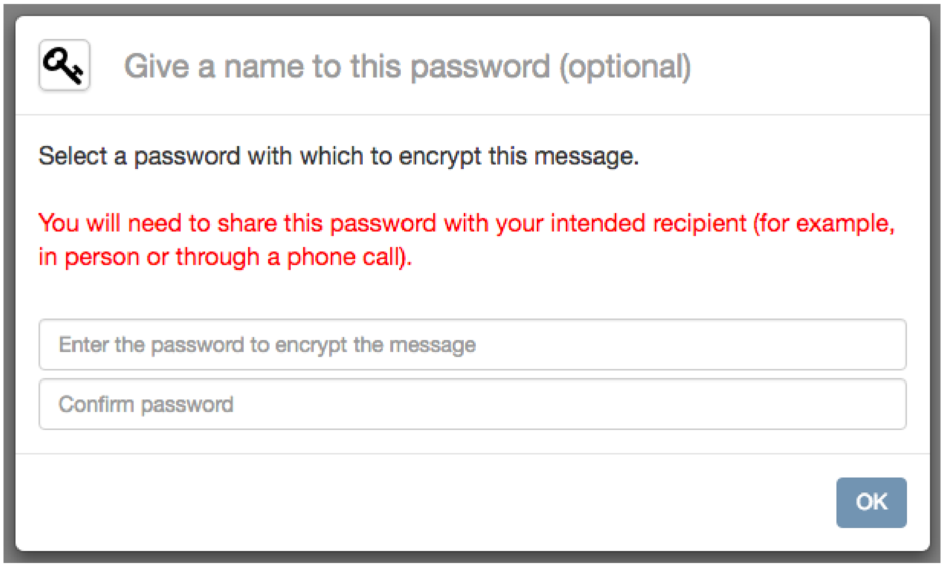
The workflow for *MessageGuard–IBE* is as follows:

1. The user visits the MessageGuard website. They are instructed to create an account with their email address. Their address is then verified by having the user click a link in an email sent to them. They are then able to download MessageGuard–IBE.
2. After installation, the user is told that the system will retrieve their IBE key from the key server. This happens automatically, as the user is already authenticated to the key server from the previous step.
3. The sender is able to send encrypted email to any address.

3.4 Passwords

Password-based encryption refers to allowing a user to select a password that will be used to encrypt their email.[3] This approach satisfies our threat model when third-parties are fully-malicious, and has the highest theoretical security of the three systems we built. Practically, the security of password-based encryption is limited by the strength of the user-chosen passwords and security of the out-of-band channels use to communicate those password. For example, if users choose easy-to-guess passwords, then an attacker could obtain an encrypted message and then brute force the password.

[3] The password is transformed into a symmetric encryption key using a password-based key derivation function.


Give a name to this password (optional)

Select a password with which to encrypt this message.

You will need to share this password with your intended recipient (for example, in person or through a phone call).

Enter the password to encrypt the message
Confirm password
OK

Figure 3.5: Dialog for Entering a New Password with Which to Encrypt Email.

The *MessageGuard–Passwords* workflow is as follows:

1. The user visits the MessageGuard website. They are prompted to download MessageGuard–Passwords.
2. After installation, the system is immediately ready to work.
3. When the user attempts to send an encrypted email, they are informed that they need to create a password with which to encrypt the email (see Figure 3.5). After creating the password, the user can then send their encrypted email.
4. The user must communicate to the recipient the password used to encrypt the email message. This should happen over an out-of-band (i.e., non-email) channel.

Chapter 4

Key Management

In this section we describe the design and implementation of key management for MessageGuard. We will first cover the overall architecture of key management, followed by a description of the internals for each of the implemented key management schemes. Finally, we will address some solutions to engineering challenges that we designed and implemented.

4.1 Architecture

MessageGuard's key management is designed in an object-oriented manner. Encryption keys are represented as instances of a *key scheme*. These instances implement the base cryptographic operations required to secure messages according to their respective key management schemes. The basic architecture is presented in Figure 4.1.

For performance reasons, MessageGuard does not utilize these key management schemes to encrypt the entire message contents. To encryt a message, a random AES key is generated, designated as the *message key*. This key is used to symmetrically encrypt the contents of the message, and it is this key that is encrypted with whatever particular key management scheme MessageGuard is configured to use.[1] The application of message keys is the approach taken by most secure email systems.

[1] This approach is most beneficial with IBE, because encryption with that scheme is quite slow relative to other schemes.

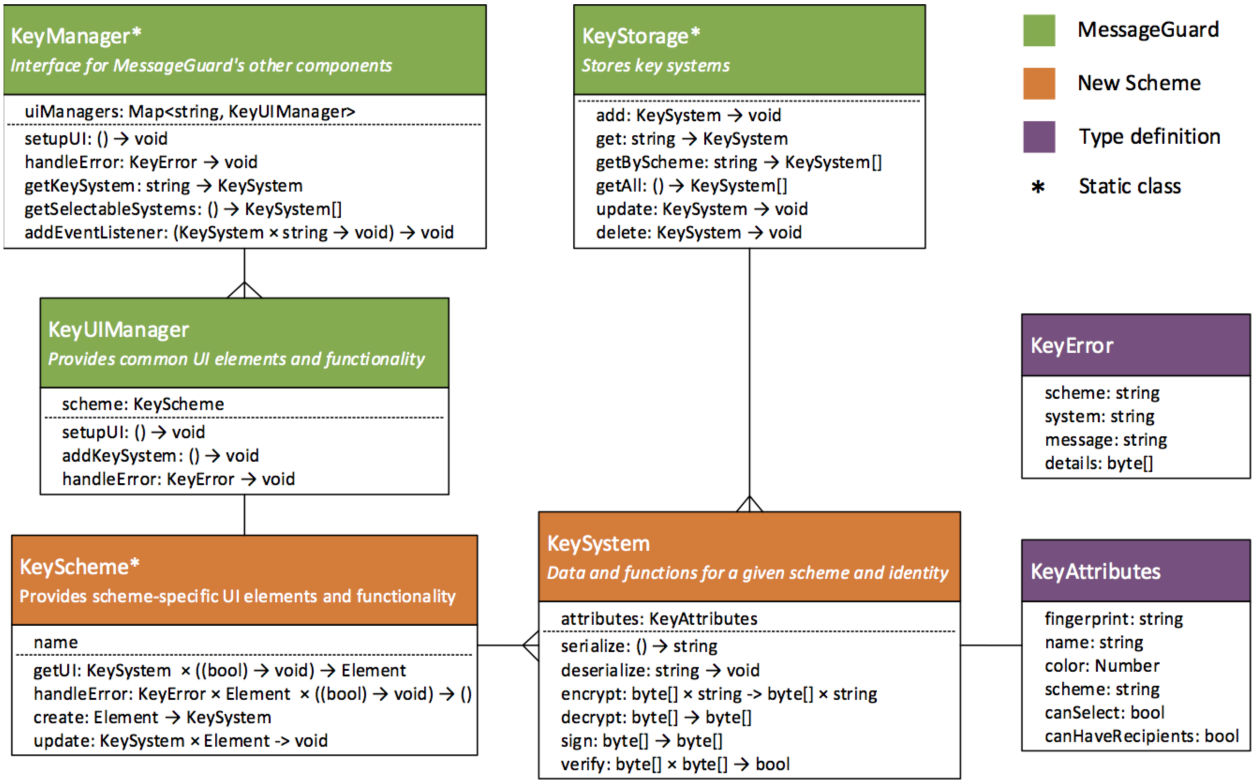

KeyManager*
Interface for MessageGuard's other components
uiManagers: Map<string, KeyUIManager>
setupUI: () → void
handleError: KeyError → void
getKeySystem: string → KeySystem
getSelectableSystems: () → KeySystem[]
addEventListener: (KeySystem × string → void) → void
KeyStorage*
Stores key systems
add: KeySystem → void
get: string → KeySystem
getByScheme: string → KeySystem[]
getAll: () → KeySystem[]
update: KeySystem → void
delete: KeySystem → void
MessageGuard
New Scheme
Type definition
* Static class
KeyUIManager
Provides common UI elements and functionality
scheme: KeyScheme
setupUI: () → void
addKeySystem: () → void
handleError: KeyError → void
KeyError
scheme: string
system: string
message: string
details: byte[]
KeyScheme*
Provides scheme-specific UI elements and functionality
name
getUI: KeySystem × ((bool) → void) → Element
handleError: KeyError × Element × ((bool) → void) → ()
create: Element → KeySystem
update: KeySystem × Element -> void
KeySystem
Data and functions for a given scheme and identity
attributes: KeyAttributes
serialize: () → string
deserialize: string → void
encrypt: byte[] × string -> byte[] × string
decrypt: byte[] → byte[]
sign: byte[] → byte[]
verify: byte[] × byte[] → bool
KeyAttributes
fingerprint: string
name: string
color: Number
scheme: string
canSelect: bool
canHaveRecipients: bool

Figure 4.1: MessageGuard Key Management Framework

4.1.1 Key Manager

The *KeyManager* is a singleton and acts as the main interface for the rest of MessageGuard key management. It is responsible for fetching *KeySystem*s from storage and using them to perform encryption and decryption operations.

4.1.2 Key UI Manager

KeyUIManager instances are responsible for managing the user interface presented to users when creating encryption keys, and editing existing keys. This class directly manages common UI elements—the key name field, key color button, and save/create buttons. UI elements specific to the particular key management schemes are managed by their *KeyScheme* instance.

4.1.3 Key Storage

KeyStorage is responsible for encryption key storage and retrieval. It is implemented as a simple key-value store, backed by `localStorage`. The “key” for each encryption key is that key’s fingerprint—see Section 4.3 for details on fingerprint generation.

4.2 Key Schemes

We implemented key management schemes for PGP, IBE, and passwords.

4.2.1 PGP

The PGP key management scheme is implemented on top of `OpenPGP.js`.[2] MessageGuard uses this library to generate a PGP key pair, and then stores that key pair in key storage. The public key is then published to the PGP key directory, provided the user has authenticated with the key directory and proved ownership of their identity.

[2] `https://openpgpjs.org`

To improve usability, PGP private keys are not password-protected. Instead of enforcing password protection for PGP keys alone, we can introduce password protection of these keys through the use of a master password, as described in Section 6.5.

When encrypting a message key, MessageGuard will query the key server for the recipient's public key. It will use this public key to encrypt the message, after using its own private key to sign it. Recipients will look up the sender's public key to verify this signature.[3]

4.2.2 IBE

The IBE key management scheme is implemented on top of `ibejs`.[4] Users provide their identity to MessageGuard, which queries the IBE key escrow server to retrieve the private key associated with that identity. The key escrow server will verify that the user has an authenticated session and has proved ownership of that identity before replying with both the private key and the public parameters necessary for generating public keys for other identities.

When encrypting a message key, MessageGuard will generate a public key for the recipient, then use it to encrypt the key. Currently, `ibejs` does not support message signing, so our IBE implementation does not yet support sender identity verification. Including this functionality would not be a large undertaking, however.

4.2.3 Passwords

The password-based key management scheme is implemented on top of the `PBKDF2` algorithm.[5] Users supply passwords to be used for symmetric encryption. These passwords are used to derive encryption keys, which are stored in key storage.[6] The `PBKDF2` algorithm is used to

[3] While the email transport mechanism reveals email metadata, MessageGuard's design anonymizes message sender identities; this allows MessageGuard to facilitate use cases such as a dead drop. To inform recipients which identity should be used to look up the public key from the key server for signature verification, the sender's identity is encrypted along with the message key. Only the recipient will be able to read this identity.

[4] `https://bitbucket.org/isrlemail/ibejs`

[5] `https://en.wikipedia.org/wiki/PBKDF2`

[6] MessageGuard discards the original password value.

derive these keys, with 5000 rounds of iteration.[7] These symmetric encryption keys are used to encrypt the message keys.

4.3 Fingerprints

To assist MessageGuard on a recipient's machine in selecting the right key to decrypt a message, a "fingerprint" is attached to each outgoing message that identifies which encryption key can decrypt the message. This concept is widely used in PGP key distribution and verification; each key pair has a short fingerprint based on the public key, which users can verify when establishing trust in a public key.

Each key management scheme in MessageGuard handles fingerprints differently. PGP generates fingerprints as the hash of the public key. IBE generates fingerprints as the hash of both the identity (e.g. email address) and the public parameters of the IBE key escrow server.

Using passwords presents a problem for generating key fingerprints. PGP and IBE are both public key schemes, so the fingerprint reveals nothing about the private key used to decrypt messages. However, because password encryption is symmetric, if we were to use the hash of the password as the fingerprint we would allow anyone with access to that fingerprint the ability to brute-force the original password value. However, since the fingerprint accompanies the encrypted ciphertext, and anyone with access to that ciphertext can launch a brute-force attack to learn the password, no new vulnerability would be introduced. We therefore implemented password fingerprints as the `bcrypt` hash of the password.

Since `bcrypt` generates a random salt with each hash, the recipient cannot simply take the `bcrypt` hash of an existing password and compare it to an incoming fingerprint to see whether that password will be able to decrypt a given message. In addition, once

[7] Many security products use more rounds, (`http://security.stackexchange.com/q/3959`) but the JavaScript-based implementation of `PBKDF2` we use is incapable of completing a higher number of rounds in a reasonable amount of time. In the future we could adjust this value, or allow users to make this security-usability tradoff themselves.

a recipient has already set up a password key, that original password value will have been discarded, and cannot be used with bcrypt to compare its hash to the incoming fingerprint.

Therefore, recipients will not be able to confirm whether any existing password keys can decrypt a message. As a result, when users receive their first password-encrypted message from a given sender, MessageGuard always prompts the recipient that they need to enter a password to decrypt that message, regardless of whether that user might have already stored a decryption password identical to the password used by the sender.

When recipients enter a potential password to use for decryption, MessageGuard will confirm it can decrypt the message by checking it against the incoming fingerprint. The bcrypt salt is extracted, and used with the password value as inputs to bcrypt. If the given password matches the password used by the sender to encrypt the message, the output of bcrypt will match the hash contained in the incoming fingerprint.

At this stage, MessageGuard must ensure that the encryption key derived from the password is identical to that derived by the sender. The key derivation function, PBKDF2, also utilizes a random salt. If the recipient derives an encryption key using a salt that differs from that used by the sender, the passwords might match but the encryption keys will not, and the receiver will be unable to decrypt their message.

To resolve this, both the sender and recipient use the bcrypt salt as the salt provided to PBKDF2. This ensures that both parties can generate the same encryption key from the same password.[8]

4.4 Key Generation

To improve usability, we hide as many key management details from users as possible. This includes PGP key generation and IBE key retrieval. Upon startup, MessageGuard queries the key server for the user's current state: whether the user is logged in, and if so what

[8] Note that this does not cause password-based key derivation to be deterministic for all parties. Salts are allowed to be randomly generated when a sender enters a new encryption password. Recipients are simply forced to use the same salt as the sender to generate their bcrypt and PBKDF2 output.

identities have been verified by the key server. For each of these identities, keys are generated or fetched, for PGP or IBE respectively.[9]

In order for key autogeneration to succeed, we had to ensure that users would have already registered with the key server and validated an identity prior to installing and running the MessageGuard extension; otherwise, the key server would refuse to accept a generated PGP key and refuse to supply a user's IBE private key. To ensure that first-time MessageGuard users are already authenticated to the key server, we designed the landing page to force visitors to register for an account and validate an identity before presenting them with an extension download link. Since this link is only presented after the user has registered and proved an identity, keys for that identity can be generated by MessageGuard immediately upon startup.[10]

During implementation, we had to ensure that the extension would be able to make authenticated AJAX requests to the key server's API. To do this, we instructed MessageGuard's key initialization code to include cookies with AJAX calls to the key server. On the key server, we configured CORS[11] to accept incoming requests from the MessageGuard extension, and allow cookies within those requests to be used for authentication.

4.5 Session Storage

When designing MessageGuard, we wanted to avoid training users to enter passwords into potentially malicious web pages. If our tool provided in-page password fields, this would broaden the attack surface for a phishing attack. As such, MessageGuard opens a new window that contains all necessary interface elements for users to enter encryption or decryption keys. For example, the password-based encryption variant presents Figure 3.5 in a new window, and not as part of a MessageGuard overlay.

[9] PGP key generation uses `window.crypto.getRandomValues` as a source for strong random numbers. This function has wide support across major browsers.

[10] Currently, MessageGuard does not support multiple users. All encryption keys are available to any user of the browser.

[11] `https://en.wikipedia.org/wiki/Cross-origin_resource_sharing`

This approach became problematic when we wanted to introduce short-term encryption keys. For example, instead of storing encryption passwords forever, we wanted to allow these passwords to be purged from memory after the user closes their browser window. In addition, if we were to implement MessageGuard with a master password that protects all stored encryption keys, this master password would also need to be evicted after the user completed their session.

The problem arose because `sessionStorage`, the native browser storage object that is designed to store short-term data, does not allow cross-window shared storage. Each window or tab receives its own session storage area. If key storage were to add an encryption key to the `sessionStorage` object while running on a separate window, the original window with MessageGuard security overlays would be unable to read that key from its own `sessionStorage`. Similarly, a master password entered in one window would be unable to be used to protect keys created in another window.

To implement shared-yet-ephemeral storage, we rapidly prototyped four variants. Each suffers from drawbacks, some debilitating. The final variant has the most advantages and is the version that is integrated within MessageGuard in our study.

- The first variant operated by using `localStorage` as a persistence layer, and clearing the contents during the `onbeforeunload` event, signifying that the window was about to close. The contents are only cleared, however, if the window being closed is the last window left open. This is done by maintaining a count of all open MessageGuard windows. Whenever a MessageGuard window loads, it advertises this to all other open MessageGuard windows, which then increment their in-memory count; these advertisements are broadcast as storage events. MessageGuard window closures trigger a similar notification, which results in each remaining window decrementing their counter.

  This approach suffers from two major drawbacks. First, `onbeforeunload` handlers are used to present a confirmation message to users, allowing them to cancel closing a window.

Modifications of localStorage are not guaranteed to succeed before the window closed. As such when a MessageGuard window closes and attempts to broadcast this event to the other active windows through a storage event, that event is not guaranteed to fire. If any window fails to broadcast their closure, the in-memory count of active windows maintained by other MessageGuard instances will cease being accurate; when the last MessageGuard window closes, it will erroneously believe that there is at least one more active window, and will refuse to evict the contents of localStorage.

Even if windows can consistently broadcast their closure, when only two MessageGuard windows are active and both are closed at once, they will both believe that another window will remain open, and will refuse to clear localStorage.

- The second variant addresses some failings in the first by eschewing the utilization of the onbeforeunload event, in favor of a complex coordination system where each active MessageGuard window is in frequent communication with each other. One "master window" is responsible for tracking how many other MessageGuard windows are currently responding to heartbeats. When the count reaches zero, localStorage is cleared, while its contents remain in memory. When this master window closes, the contents are lost.

  If the master window closes while it is not the last window left alive, the remaining MessageGuard windows utilize a stochastic process to "elect" a new master window, in a first-come-first-serve basis. Essentially, each window waits a random amount of time before electing itself as the new master window. The first MessageGuard window to perform this is recognized as the new master window by the rest.

  This approach is susceptible to race conditions. One window may elect itself as the master window and broadcast this to the rest of the waiting windows, while in the meantime another window also elects itself, before having received the broadcast message from the first window. Both would believe they are the master window, and resolving this would prove difficult.

Additionally, like the first variant this option fails to operate properly when all active windows close simultaneously. No window would have the opportunity to note that it is the only window left alive, and `localStorage` would not be cleared.

- The third variant keeps all session storage in memory. When a new MessageGuard window loads, it broadcasts a request for the current contents of session storage to all currently-active windows. If a response is received, this new window then populates its session storage contents with the data in the response. Any modifications to this in-memory store are also broadcast to all active instances. Any window that closes leaves no trace of this store on disk, and when the last window closes the contents are lost.

  While this approach is sufficient to fulfill the requirements of session storage, we found that—from a user's perspective—it is a bit too eager to clear its contents. If a user loads MessageGuard in their email provider and provides a decryption key to read their mail, then reloads their email tab, the session storage clears itself, since it only exists in memory and no other MessageGuard window was active to maintain its state.

- We realized at this point that Chrome did provide a native storage object that fulfills the requirements for MessageGuard: session cookies. These cookies are ephemeral, clearing themselves after the browser window closes; and they allow cross-window sharing of data. Unfortunately, cookies are not suitable for storing large amounts of data. To use the large storage capacity of `localStorage` with the ephemerality of session cookies, we applied symmetric AES encryption. Upon startup, a random AES key is generated and placed in a session cookie, which is configured to be visible across the MessageGuard extension origin. This key is used to encrypt data placed in `localStorage`. Any MessageGuard window that opens in the same session has access to this AES key, and is able to decrypt data retrieved from `localStorage`, giving it access to the current state of session storage. When the browser closes, the `localStorage` contents persist, but the encryption key necessary to read these contents is lost. The next time MessageGuard opens and attempts to read the contents of session storage, it will be unable to do so and will behave as if that data does not exist.

This appraoch fulfills all requirements of session storage, and behaves well for users. As such, it is the approach adopted for MessageGuard during the study, and is the persistence layer used for password-based secure email.

One drawback to this approach is that, since the AES key is stored in a session cookie, that key will be transmitted along with every HTTP resource request to the domain used by MessageGuard. Since MessageGuard is deployed as a browser extension, these requests never travel over the network and the key is not made visible to any third party. If MessageGuard were to be deployed as a bookmarklet and run from a dedicated domain, that domain would have access to the ephemeral encryption key. However, without access to the contents of `localStorage`, the key has no use. And since the security model of MessageGuard as a bookmarklet requires trusting the dedicated domain (since this domain is responsible for serving up all MessageGuard code), the risk of exposing the ephemeral AES encryption key to that server is minimal.

4.6 Concurrency

In Chrome, all iframes on the same origin run on the same thread. This also applies to extension domains. Since key manager operations can be computationally expensive, we needed to have key management run in its own thread. In Chrome, this is accomplished by utilizing web workers. These web worker theads are initialized by passing the URL of a JavaScript file to the WebWorker constructor. Communication with this thread is implemented as simple message passing.

To implement key management as a web worker, we created a module with a collection of key management “tasks”. These include encrypting a message key, encrypting a draft, performing decryption, and responding to heartbeat requests. This module is required by a “thread message responder” module, which is the target script provided to the WebWorker constructor. A “thread message transmitter” module passes requests to the web worker, and awaits responses from that thread.

When migrating the key management codebase to run in the context of a web worker, a major issue we encountered was the unavailability of local storage and cookies. Local storage is widely used across key management—for key storage, advertising the creation of a key, or implementing session storage. The latter of these also relies heavily on cookies. In the web worker environment, these objects did not exist.

To resolve this, we implemented a proxy that passes commands or requests between the main origin thread and a web worker thread. The worker thread is able to request, set, or delete local storage or cookie data from this proxy, and broadcast or subscribe to local storage events. This proxy used Promises[12] to allow asynchronous inter-thread communication.

We wanted to ensure that key management could function properly even if it were to run in a browser where web workers is not available. To allow this, by default key management loads as a “web worker loader”, with only enough know-how to initialize a web worker and pass it requests. If this loader detects that web workers are not available, it reloads itself in “stand-alone” mode, with all key management code available. In this mode, concurrency will not be available, but key management will still function normally.

Since key management may run in either a worker thread or the main origin thread, the local storage and cookie proxy interfaces must be identical to that of the native versions. Although native local storage and cookies are accessed synchronously, we introduced an indirection layer that exposes these objects’ APIs with Promises. In this way, key management

[12] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

can access local storage and cookies using Promises, regardless of whether it runs in a worker thread or the main origin thread.

Chapter 5

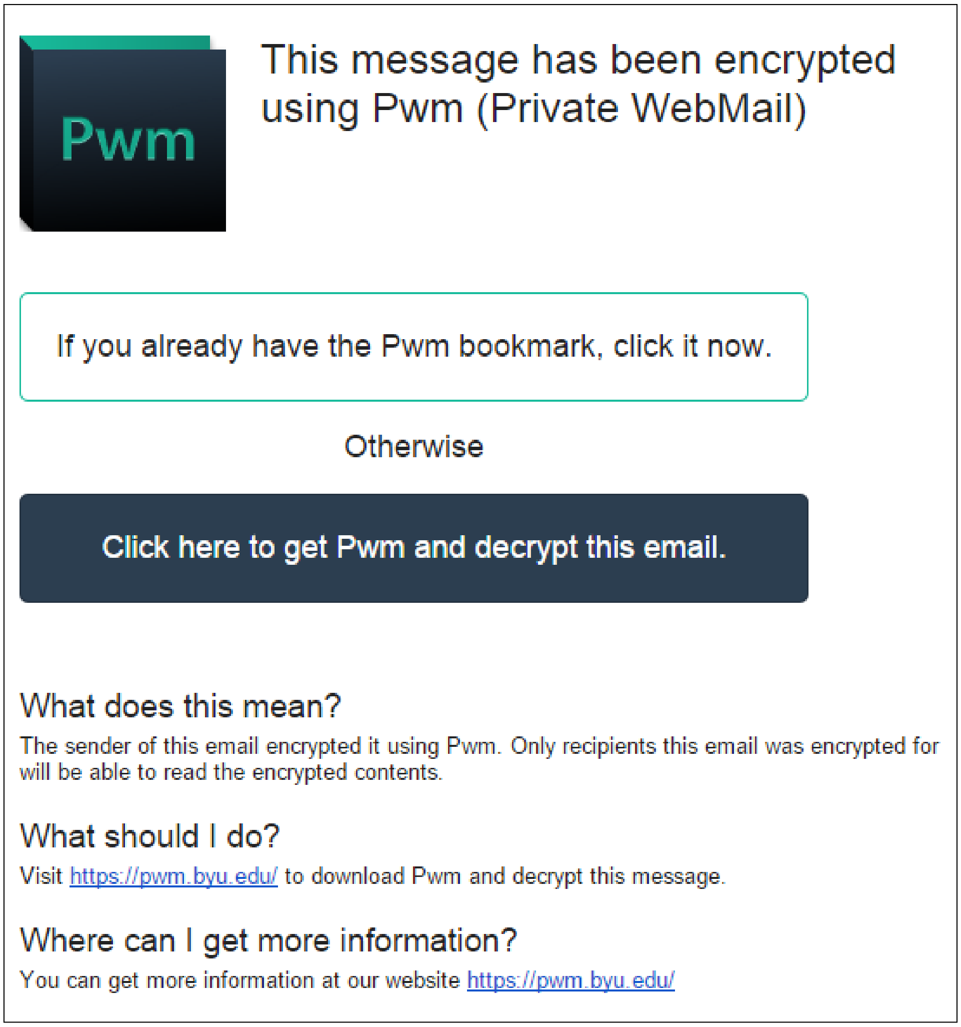
Design Iteration

As part of the engineering effort toward implementing MessageGuard, we evaluated our design decisions through the use of pilot studies and a cognitive walkthrough. Over the course of these studies, we gained valuable insight on the usability impact of our approaches, and were able to make changes that significantly improved the usability of MessageGuard.

5.1 Pilot Studies

Once the MessageGuard system was sufficiently implemented so as to be functional, we conducted a number of pilot studies with friends and family. These studies where highly informal, and consisted of simply attempting to use the system to send an encrypted message. All pilot study participants were successfully able to send and receive encrypted messages.

There were four minor changes that arose from these studies. First, we introduced more "friendly", colloquial wording to the system's status and error messages. For example, Figure 3.3 illustrates the error message shown to PGP users when attempting to encrypt messages for users that had no public key registered to the MessageGuard key server. Previously, this was a rather stark error message, reading, *"This recipient has not installed MessageGuard. Before you can send them encrypted messages, they must be registered. Have them visit MessageGuard.io/pgp to download and install the MessageGuard tool."* One pilot study participant felt unease at this message, and as a result we expanded it to include instructions on waiting for recipients to register before trying again; we also included reassurances that



This message has been encrypted using Pwm (Private WebMail)

If you already have the Pwm bookmark, click it now.

Otherwise

Click here to get Pwm and decrypt this email.

What does this mean?

The sender of this email encrypted it using Pwm. Only recipients this email was encrypted for will be able to read the encrypted contents.

What should I do?

Visit https://pwm.byu.edu/ to download Pwm and decrypt this message.

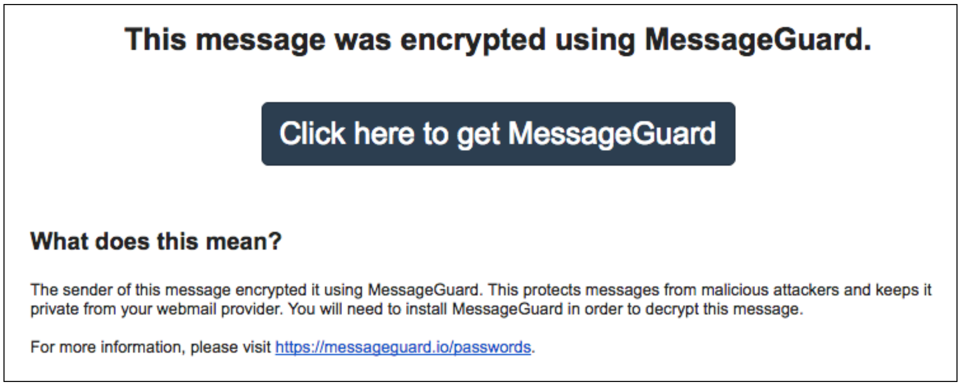
Where can I get more information?

You can get more information at our website https://pwm.byu.edu/

Figure 5.1: Instructions for Pwm and Early Iterations of MessageGuard.

the message draft has been saved, and the user should feel free to close the message and come back to it later.

Second, we received feedback on the instructions included in an encrypted email, indicating that it was not visually well-designed. These instructions, inherited from Pwm [20] and presented in Figure 5.1, provide directions for non-MessageGuard-users on how to proceed with decrypting their encrypted email. As a result of this feedback, we redesigned these instructions and stripped them down to the bare essentials. The results of this effort are presented in Figure 5.2



This message was encrypted using MessageGuard.

What does this mean?

The sender of this message encrypted it using MessageGuard. This protects messages from malicious attackers and keeps it private from your webmail provider. You will need to install MessageGuard in order to decrypt this message.

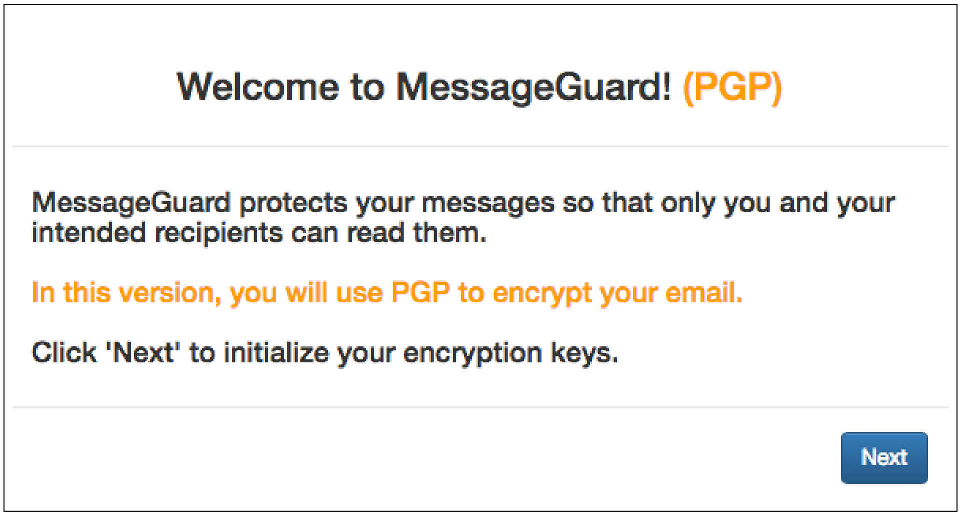
For more information, please visit https://messageguard.io/passwords.

Figure 5.2: Revised Instructions for MessageGuard-encrypted Messages.

Third, we gained evidence that the password name field was non-obvious and would need to be made optional. The password entry form, presented in Figure 3.5, gives users the option to name their encryption passwords, to help them differentiate between them when encrypting email for different parties. Pilot study participants never voluntarily interacted with this field, and questioned its purpose. As a result of this, MessageGuard's key creation was altered to allow nameless keys, which would be given default names differentiated by a numeral suffix (e.g. "Encryption Password", "Encryption Password (2)").

Finally, we noted that the term "key" appeared in multiple contexts within MessageGuard. This was problematic, as we never introduced users to this term, nor provided any context for users to understand how encryption keys are used in general. MessageGuard was designed to be as usable as possible; ensuring user understanding was not a design goal. As such, we decided to restructure MessageGuard's labeling to avoid using this term, in favor of terms such as "encryption password".

We did not validate the effectiveness of these changes with pilot study participants. We took the feedback received from these pilot studies and applied the changes we believed were necessary to address the issues observed.



Welcome to MessageGuard! (PGP)

MessageGuard protects your messages so that only you and your intended recipients can read them.

In this version, you will use PGP to encrypt your email.

Click 'Next' to initialize your encryption keys.

Figure 5.3: MessageGuard Initialization Modal

5.2 Cognitive Walkthrough

The cognitive walkthrough [29] is a formal method for exposing potential usability hurdles in a system. Conducted by system designers, the cognitive walkthrough is organized into tasks that users will need to perform. Each task consists of a list of correct actions. At each step, the system designers will consider what action a user might take, based on available clues and past instructions. Designers note if a typical user would deviate from the correct action. Based on these notes, designers can improve the usability of their system by addressing the mistakes users would make in using their system.

We conducted a cognitive walkthrough of MessageGuard, and made one major change as a result of this walkthrough, having to do with the initialization process. As described in Section 4.4, MessageGuard will auto-generate PGP keys, and auto-fetch IBE keys, upon startup. This vastly simplifies the instructions a user needs to send to a recipient who has not installed MessageGuard yet and needs to generate PGP keys. All the recipient has to do is install MessageGuard and then start it once; at that point, the recipient's PGP keys are automatically generated, and the sender can proceed with encrypting email for that recipient.

Prior to the cognitive walkthrough, MessageGuard would not “start up” until it was loaded as an overlay, for the purposes of displaying or composing an encrypted message. Only once a user loaded MessageGuard in order to read or write a message would the key autogeneration take place. The problem with this approach was that it was possible for a recipient to believe they had properly installed MessageGuard without having performed key initialization. If recipients notify the sender that they have installed MessageGuard—which they have—without ever having loaded MessageGuard as an overlay, the recipient’s PGP key will not have been generated and published. At that point, the sender would try to encrypt for the recipient and be unable to do so, even though both sender and recipient have installed MessageGuard.

Essentially, the installation phase and key generation phase were separate stages. To address this, we extended installation to include key generation. Upon installation of MessageGuard, an initialization window opens. This window is responsible for carrying out key generation, and is presented in Figure 5.3. By funneling users through this initialization window, we are able to ensure that all users who complete MessageGuard installation will also have completed key generation, preventing the issue described above. A further benefit of the initialization window is that it explicitly informs users that MessageGuard has been installed. Before the initialization window was introduced, users would need to visit Gmail before they received confirmation that MessageGuard had been correctly installed.

Chapter 6

Methodology

We conducted an IRB-approved user study wherein pairs of participants used secure email to communicate sensitive information to each other. Our study methodology is taken from work by Ruoti et al. [19]. We use this methodology for its various benefits and because it allows us to compare the results from our systems with the results produced by Ruoti et al.

In the remainder of this section we give an overview of the study and describe its scenario, tasks, study questionnaire, and post-study interview. In addition, we discuss the development and limitations of the study.

6.1 Study Setup

The study ran for two and a half weeks—beginning Monday, May 23, 2016 and ending Tuesday, June 7, 2016. In total, 55 pairs of participants (110 total participants) took the study. Due to various reasons discussed later in this section, we excluded results from eight participant pairs. For the remainder of this paper, we refer exclusively to the remaining 47 pairs (94 participants).

Participants took between fifty and sixty-five minutes to complete the study, and each participant was compensated $15 USD for their participation. Participants were required to be accompanied by a friend, who served as their counterpart for the study. For standardization and to satisfy requirements of the systems tested in the study, both participants were required to use their own Gmail accounts.

When participants arrived, they were given a consent form to sign, detailing the study and their rights as participants. Participants were informed that they would be in separate rooms during the study and would use email to communicate with each other. Participants were also informed that a study coordinator would be with them at all times and could answer any questions they might have.

Using a coin flip, one participant was randomly assigned as Participant A (referred to as “Johnny” throughout the paper) and the other as Participant B (referred to as “Jane” throughout the paper). The participants were then led to the appropriate room to begin the study; each room had identical equipment. For the remainder of the study, all instructions were provided in written form. Participants completed the task on a virtual machine, which was restored to a common snapshot after each study task, ensuring that the computer started in the same state for all participants and that no participant information was inadvertently stored.

During the study, participants were asked to complete a multi-stage task three times, once for each of the secure email systems being tested: PGP, IBE, and Passwords. The order in which the participants used the systems was randomized.

6.2 Demographics

We recruited Gmail users for our study at a local university, as well as through Craigslist. Participants were evenly split between male and female: male (47; 50%), female (47; 50%). Participants skewed young: 18 to 24 years old (75; 80%), 25 to 34 years old (18; 19%), 35 to 44 years old (1; 1%).

We distributed posters across campus to avoid biasing our participants towards any particular major. Most participants were college students: high school graduates (1; 1%), undergraduate students (71; 76%), college graduates (15; 16%), graduate students (7; 7%). Participants were enrolled in a variety of majors, including both technical and non-technical majors.

6.3 Scenario Design

During the study, participants were asked to role-play a scenario about completing taxes. Johnny was told that his friend, Jane, had graduated in accounting and was going to help Johnny prepare his taxes. To do so, Johnny needed to send her his social security number and his last year's tax PIN. Johnny was told that because this information was sensitive, he should encrypt it using a secure email system we gave him.[1] Jane was told that she would receive some information regarding taxes from Johnny, but was not informed that the information would be encrypted.

6.4 Task Design

Based on the scenario, participants were asked to complete a two-stage task.

1. Johnny would encrypt and send his SSN and last year's tax PIN to Jane.
2. Jane would decrypt this information, then reply to Johnny with a confirmation code and this year's tax PIN. The reply was required to be encrypted. After Johnny received this information, he would inform Jane that he had received the necessary information, and then the task would end.[2].

During each stage, participants were provided with worksheets containing instructions regarding the task and space for participants to record the sensitive information they received.[3] Both participants were provided with the information they would send (e.g., SSN and PIN), but were told to treat this information as they would their own sensitive information. Participants completed the same task for each of the three systems being tested.

Before beginning any tasks, participants were informed that other than the sensitive information they were provided, which would need to be transmitted over email, they were

[1] Johnny was provided a URL for the secure email system to use.

[2] This confirmation step is added to ensure that Johnny could decrypt Jane's message. We did not require the confirmation message to be encrypted.

[3] These instructions did not include directions on how to use any of the systems.

free to communicate with each other however they normally would. Additionally, participants were informed that they could browse the Internet, use their phones, or engage in other similar activities while waiting for email from their friend. This was done to provide a more natural setting for the participants, and to avoid frustration if participants had to wait for an extended period of time while their friend figured out an encrypted email system.

Study coordinators were allowed to answer questions related to the study but were not allowed to provide instructions on how to use any of the systems being tested. If participants became confused regarding the system, coordinators would tell them that they could answer questions regarding the task, but could not describe how to use the systems being tested.

6.5 Study Questionnaire

We administered our study using the Qualtrics web-based survey software. As part of this survey, participants answered a set of demographic questions.

Immediately upon completing the study task for a given secure email system, participants were asked several questions related to their experience with that system. First, participants completed the ten questions from the System Usability Scale (SUS) [5, 6]. Multiple studies have shown that SUS is a good indicator of perceived usability [26], is consistent across populations [18], and has been used in the past to rate secure email systems [1, 16, 19]. After providing a SUS score, participants were asked to describe what they liked about each system, what they would change, and why they would change it.

After completing the task and questions for all three secure email systems, participants were asked to select which of the encrypted email systems they had used was their favorite, and to describe why they liked this system. Participants were next asked to rate the following statements using a five-point Likert scale (Strongly Disagree–Strongly Agree): “I want to be able to encrypt my email,” and “I would encrypt email frequently.”

Finally, the survey told participants that MessageGuard could be enhanced with a master password, which they would be required to enter before MessageGuard would function.

This would help protect their sensitive messages from other individuals who might also use the same computer. After reading the description about adding a master password to MessageGuard, users were asked to describe whether they would want this feature and why they felt that way.

6.6 Post-Study Interview

After completing the survey, participants were interviewed by their respective study coordinators. The coordinators asked participants about their general impressions of the study and the secure email systems they had used. Furthermore, the coordinators were instructed to note when the participants struggled or had other interesting events occur, and during the post-study interview the coordinators reviewed and further explored these events with the participants.

To assess whether participants understood the security provided by each secure email system, coordinators questioned participants regarding what an attacker would need to do to read their encrypted messages. Coordinators would continue probing participants' answers until they were confident whether or not the user correctly understood the security model of each system.

After describing their perceived security models, participants were then read short descriptions detailing the actual security models of each system. Participants were encouraged to ask questions if they wanted further clarification for any of the described models. After hearing these descriptions, participants were then asked to indicate whether their opinions regarding any of the systems had changed. Participants were also asked whether they would change their answer regarding their favorite system on the survey.

Upon completion of the post-study interview, participants were brought together for a final post-study interview. First, participants were asked to share their opinions on doing a study with a friend, as opposed to a traditional study. Second, participants were asked to describe their ideal secure email system. While participants are not system designers, we hoped that this question might elicit responses that participants had not yet felt comfortable sharing.

6.7 Quality Control

We excluded responses from eight pairs of participants.[4] First, three pairs were removed because the secure email tools became inoperative during the study, making it impossible for participants to complete the study. Second, two pairs were removed because the participants were non-native English speakers, making it difficult for them to understand the instructions they were given.

Third, we removed three participant pairs that were clearly not paying attention to the study survey. One participant answered "neither agree nor disagree" to all Likert items and did not fill in answers to any other question. Another participant admitted at the end of the study that he hadn't noticed that the SUS questions alternated between positive and negative phrasings. One of the participants from the third pair gave nonsense answers to several questions. Rather than try to extract the few answers that might have been meaningful from these participants, we felt it was best to remove them from our data.

6.8 Limitations

Our study involved a single user sending email to one other user. This approach was helpful in understanding the basic usability of the systems tested, but it might not reveal all the usability issues that would occur in other communication models, such as a user sending email to multiple individuals. Future work could expand this research and examine other usage scenarios.

Our study also has limitations common to all existing secure email studies. First, our populations are not representative of all groups, and future research could broaden the population (e.g., non-students, non-Gmail users). Second, our study was a short-term

[4] When we excluded a participant's results, we also excluded their partner's results.

study, and future research should look at these issues in a longer-term longitudinal study. Third, our study is a lab study and has limitations common to all studies run in a trusted environment [14, 25].

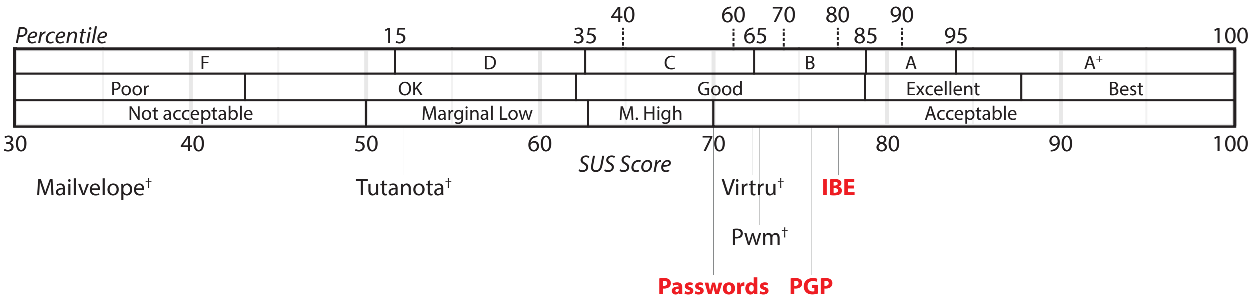
Chapter 7

Results

In this section we report on quantitative results from our our study. First, we report the SUS score for each system. Next, we give task completion times and details on how often participants mistakenly sent sensitive information in the clear. We then detail users' understanding of each system's security model. Finally, we report on users' favorite systems and several other minor results.

7.1 System Usability Scale

We evaluated each system using the System Usability Scale (SUS). A breakdown of the scores are given in Table 7.1. To give context to these scores, we leverage the work of several researchers that correlated SUS scores with more intuitive descriptions of usability [3, 4, 22, 26]. The descriptions are presented in Figure 7.1.



† These SUS scores are for other secure email systems tested with the same methodology, and are reference points for the performance of our systems. Mailvelope is a PGP-based system, Tutanota a password-based system, Pwm an IBE-based system, and Virtru a custom key escrow scheme.

Figure 7.1: Adjective-based Ratings to Help Interpret SUS Scores

	Participant	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)	Range	Percentile
PGP	Johnny	47	75.0	16.2	±4.6	70.4–79.6	73%
	Jane	47	76.5	13.5	±3.9	72.6–80.4	78%
	Both	94	75.7	14.9	±3.0	72.7–78.7	76%
IBE	Johnny	47	77.7	13.8	±3.9	73.8–81.6	82%
	Jane	47	76.9	13.3	±3.8	73.1–80.7	80%
	Both	94	77.3	13.5	±2.7	74.6–80.0	81%
Passwords	Johnny	47	72.7	15.4	±4.4	68.3–77.1	66%
	Jane	47	67.2	14.2	±4.1	63.1–71.3	48%
	Both	94	70.0	15.0	±3.0	67.0–73.0	56%

Percentiles are calculated by looking up the SUS score in a table [22]. When a SUS score is not in the table we estimate the percentile based on the available data.

Table 7.1: SUS Scores

PGP's SUS score of 75.7 is rated as having "Good" usability, receives a "B" grade, and falls in the 76th percentile of systems tested with SUS. IBE's score of 77.3 is also rated as having "Good" usability, receives a "B+" grade, and is in the 81st percentile. Finally, Passwords' score of 70.0 is rated as having "Good" usability, receives a "C" grade, and reaches the 56th percentile.

The difference between PGP's and IBE's SUS scores are not statistically significant (two-tailed student t-test, matched pairs—Johnny–$p = 0.12$, Jane–$p = 0.83$, Both–$p = 0.21$). In contrast, the difference between Jane's PGP and Passwords scores are significant (two-tailed student t-test, matched pairs—Johnny–$p = 0.28$, Jane–$p < 0.001$, Both–$p < 0.001$). Also, the difference between IBE's and Passwords' SUS scores are significant for both Johnny and Jane (two-tailed student t-test, matched pairs—Johnny–$p = 0.01$, Jane–$p < 0.001$, Both–$p < 0.001$)

We also compared the SUS scores for our systems against the SUS scores for other systems previously tested with the same methodology [19]. Specifically, we compared systems that had the same key management approach: PGP to Mailvelope, IBE to Pwm, IBE to

	Participant	Count	Mean When First	Mean When Not First	Effect Size	$p^\dagger$
PGP	Johnny	47	81.6	71.6	−10.0	0.04
	Jane	47	79.7	74.8	−4.9	0.25
	Both	94	80.6	73.2	−7.4	0.02
IBE	Johnny	47	81.0	75.8	−5.2	0.22
	Jane	47	79.6	75.3	−4.3	0.30
	Both	94	80.3	75.6	−4.7	0.10
Passwords	Johnny	47	65.9	75.6	+9.6	0.04
	Jane	47	66.4	67.6	+1.2	0.80
	Both	94	66.1	71.3	+5.2	0.11

†Two-tailed student t-test, equal variance.

🟩 Result is statistically significant

Table 7.2: SUS Score By Ordering

Virtru,[1] and Passwords to Tutanota. In each case, our system out-performed these other systems (two-tailed student t-test, unequal variance—PGP and Mailvelope–$p < 0.001$, IBE and Pwm–$p < 0.09$, IBE and Virtru–$p = 0.04$, Passwords and Tutanota–$p < 0.001$). This gives strong evidence that the design principles integrated into MessageGuard [20, 21] lead to strong usability in secure email systems, regardless of the key management approach used.

We also tested to see whether there was a statistically significant difference between the SUS score ratings of Johnny and Jane. We found no significant difference for either PGP or IBE (two-tailed student t-test, equal variance—PGP–$p = 0.63$, IBE–$p = 0.76$). For Passwords, we found a nearly significant result, with Johnny rating Passwords 5.5 points higher than Jane (two-tailed student t-test, equal variance—Passwords–$p = 0.08$), though this difference disappears when Passwords was the first system tested (two-tailed student t-test, equal variance—Passwords–$p = 0.92$).

[1] Virtru uses key escrow which from the user's perspective is functionally indistinguishable from IBE.

We also explored whether the order in which systems were tested had an effect on their SUS scores. The results of this analysis are summarized in Table 7.2. Overall, Johnny's experience was significantly affected by system ordering. Jane's experience was also affected, but the effect was never statistically significant.

Johnny rated PGP 10 points higher when it was the first system he tested; he also rated Passwords 9.6 points lower when it was the first system tested. More specifically, we found that PGP's score only dropped when it followed Passwords, and this drop was statistically significant (two-tailed student t-test, equal variance—Johnny–$p < 0.01$, Jane–$p = 0.13$, Both–$p < 0.01$). Similarly, Passwords' score only rose when it followed PGP, though in this case the relation only existed if Passwords immediately followed PGP; this difference was also statistically significant (two-tailed student t-test, equal variance—Johnny–$p < 0.01$, Jane–$p = 0.23$, Both–$p < 0.01$).

IBE's scores were relatively stable regardless of ordering, except when the systems were tested in this order: Passwords, IBE, PGP; in this case, IBE's score was 14.0 points lower than the mean score for all other orderings. This treatment had a similar effect on PGP; when tested in this order, PGP's score was 11.4 points lower than its mean score for other orderings. Based on our analysis of the data, it is possible that this treatment's low scores for PGP and IBE is an anomaly that would disappear over the course of additional studies.

7.2 Time

We recorded the time it took each participant to finish the assigned task with each system. For timing purposes the tasks were split into two stages. The first stage started when Johnny first visited the MessageGuard website and ended when he had successfully sent an encrypted email with his SSN and last year's tax PIN. The second stage started when Jane received her first encrypted email and ended when she had decrypted it, replied with the appropriate information, and received the confirmation email from Johnny. It is possible for stage one and

	Stage	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)	Range
PGP	1	47	8:02	3:06	±0:53	7:09–8:55
	2	45	3:24	1:28	±0:26	2:58–3:50
	Both	45	11:33	3:53	±1:08	10:25–12:41
IBE	1	46	3:30	1:30	±0:26	3:04–3:56
	2	44	5:58	2:36	±0:46	5:12–6:44
	Both	43	9:30	3:50	±1:09	8:21–10:39
Passwords	1	46	3:31	1:25	±0:25	3:06–3:56
	2	44	6:54	3:34	±1:03	5:51–7:57
	Both	43	10:22	4:00	±1:12	9:10–11:34

Table 7.3: Time Taken to Complete Task (min:sec)

two to overlap; if Johnny first sends an encrypted message without the required information, this will start the timer for stage two without stopping the timer for stage one.[2]

Timings were calculated using the video recordings of the participants' screen. In one instance the video file of the recording was corrupted, making it impossible to gather task completion times. In three other instances the study coordinators forgot to start the video recording, causing either a complete (one instance) or partial (two instances) loss of data. Task completion time data from the remaining recordings is given in Table 7.3 and Figure 7.2.

By design, PGP shifts a significant portion of user effort from stage two to stage one—Jane installs PGP in stage one instead of stage two. As such, the difference in stage completion times between PGP and the other two systems are statistically significant (two-tailed student t-test, matched pairs—both stages, both systems–$p < 0.001$). For total task completion time (stage one + stage two), only the difference between PGP and IBE is statistically significant (two-tailed student t-test, matched pairs—PGP and IBE–$p = 0.05$, PGP and Passwords–$p = 0.14$, IBE and Passwords–$p = 0.321$).

[2] We took this approach as stage one is clearly not finished, but Jane is also able to start making progress on completing stage two.

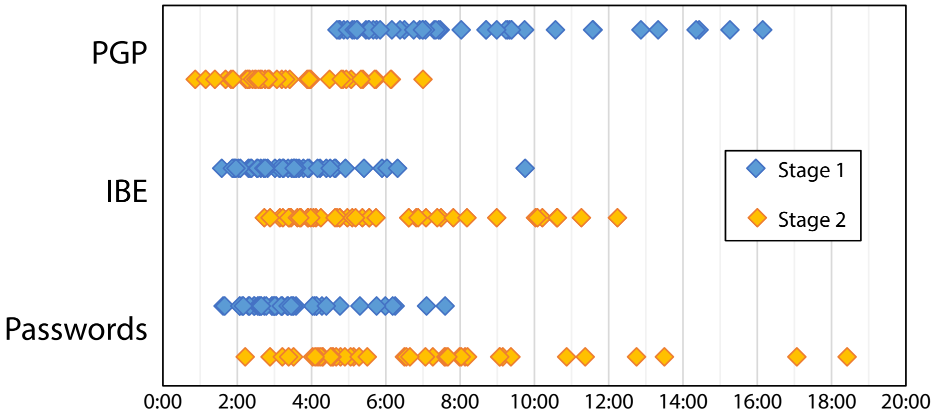


Figure 7.2: Individual Participant Task Completion Times

	Stage	Count	Mean When First	Mean When Not First	Effect Size	$p^{\dagger}$
PGP	1	47	9:36	7:13	−2:23	0.01
	2	45	4:20	2:54	−1:26	< 0.01
	Both	45	13:56	10:14	−3:42	0.01
IBE	1	46	4:47	2:49	−1:58	< 0.001
	2	44	8:01	4:48	−3:13	< 0.001
	Both	43	12:55	7:39	−5:16	< 0.001
Passwords	1	46	4:50	3:01	−1:49	< 0.001
	2	44	9:49	5:48	−4:01	< 0.001
	Both	43	14:48	8:50	−5:58	< 0.001

[†] Two-tailed student t-test, equal variance.

🟩 Result is statistically significant

Table 7.4: Time Taken to Complete Task (min:sec)

As shown in Table 7.4, the task for the first system tested took longer than the other two.[3] The effect was most extreme for Jane, who didn't know that she would be using secure email until receiving her first message from Johnny. Interestingly, system ordering had the smallest effect on PGP's task completion time; though, this could be tied to PGP having the highest overall times.

7.3 Mistakes

We define mistakes to be instances when users send sensitive information in normal email when it should have been encrypted. For Passwords, a user is also considered to have made a mistake if they send the email-encryption password in a plaintext email.[4]

In PGP and IBE there were a low number of mistakes, and each was made by Johnny (PGP–[n = 1; 2%], IBE–[2; 4%]). In all three cases, the participant transmitted the sensitive information in the unencrypted greeting of the encrypted message. This happened in spite of the fact that two participants watched the compose tutorial, which warned them that text in that field would not be encrypted. This problem area could likely be addressed by making users explicitly enable unencrypted greetings, instead of displaying it as a default field.

In Passwords, all mistakes were a result of users sending their password in plaintext email (Johnny–[9; 19%], Jane–[1; 2%]). For five of these mistakes (5; 11%), Johnny first sent the password over cellular text messaging, but for various reasons Jane never got this message. When Jane received her encrypted email, she didn't yet have the password and would email Johnny requesting the password, which he sent to her using email. It is unclear whether this represents users' lack of understanding regarding the security of email [15], a lack of concern for the safety of their sensitive information, an artifact of taking the study in a trusted environment [25], or a mixture of the three. Additionally, in four cases Johnny used Google Chat to send their password, giving Google access to both the secure email and

[3] There wasn't a strong difference in task timing between whether the system was second or third.

[4] Mistakes can also include revealing PGP or IBE private keys, though neither of our systems allowed users to make this mistake.

the password used to encrypt it. Still, we chose not to include this as a mistake as it is not as egregious as sending the password over email.

Regardless, we note that the mistake rate of our Passwords system is significantly lower than that for Tutanota. In Ruoti et al.'s evaluation of Tutanota, a system which allows users to password-encrypt email, they found that Johnny sent the password in the clear 68% of the time. This is 49% greater than our system's rate, and the difference is statistically significant ($N-1$ chi-squared test—$\chi^2[1, N = 72] = 16.65, p < 0.001$). This is likely due to the fact that during password input, our Passwords system instructed users to send their password using a non-email communication channel.

7.4 Understanding

In the post-study interview we asked participants to identify what an attacker would need to do to read their encrypted email. The goal of this question was to evaluate whether participants understood the security model of each system they had tested. Study coordinators would continue probing participants until they were confident regarding whether or not the participant had a proper understanding of the system in question. In five cases (Johnny–2, Jane–3), the study session ran late and participants had to leave without completing the post-study interview. As such, percentages in this Subsection are calculated off a different total number of participants (Johnny–45, Jane–44, Both–89).

Participants had a poor understanding of both PGP's (Johnny–[2; 4%], Jane–[2; 5%], Both–[4; 4%]) and IBE's (Johnny–[2; 4%], Jane–[3; 7%], Both–[5; 6%]) security models. Generally, participants believed that if an attacker could gain access to a user's email then they could decrypt that user's messages. Only a handful of participants recognized that signing up for an account prior to downloading the tool was meaningful. During the interviews, most participants indicated they saw no difference in the security of IBE and PGP.

In strong contrast, nearly all participants had a clear understanding of how password-based encryption protected their emails (Johnny–[41; 91%], Jane–[41; 93%], Both–[82; 92%]).

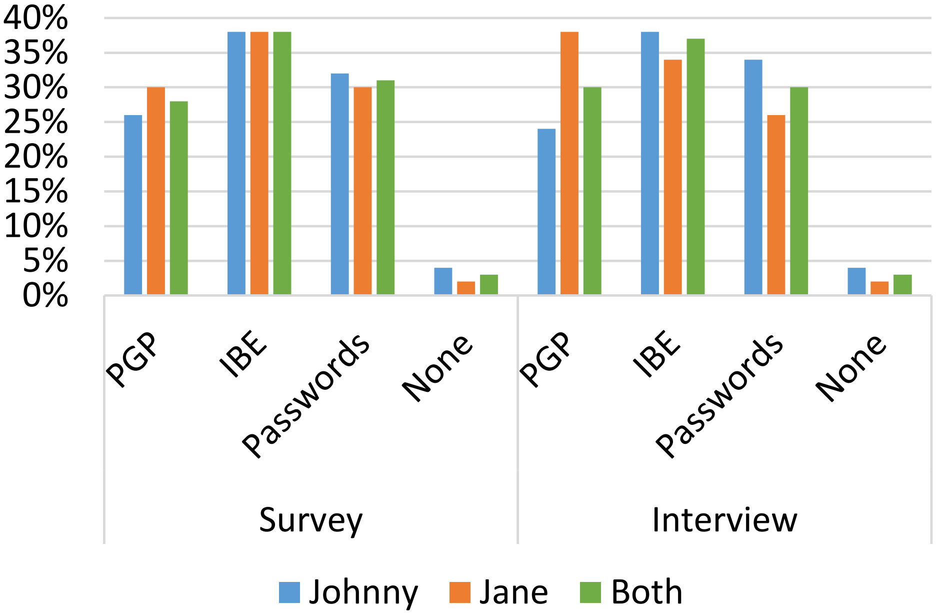


Figure 7.3: Participants' Favorite System

7.5 Favorite System

At the end of the study survey, participants were asked to indicate which system was their favorite and why. Later, during the post-study interview, participants were given descriptions of each system's security model and were invited to ask further clarifying questions as needed. After hearing these descriptions, participants were allowed to update which system they felt was their favorite. Participants' preferences, both pre- and post-survey, are summarized in Figure 7.3.

Overall, participants were split on which system they preferred (During Survey—PGP–[26; 28%], IBE–[36; 38%], Passwords–[29; 31%]; After Interview—PGP–[29; 31%], IBE–[34; 36%], Passwords–[28; 30%]). While IBE was a slight favorite, the difference was not statistically significant (Chi-squared test—Survey–$\chi^2[2, N = 282] = 2.56, p = 0.28$, Interview–$\chi^2[2, N = 282] = 1.01, p = 0.60$). Of the three participants who did not select a favorite system (3; 3%), two indicated that they liked all three systems equally, and the third participant indicated that they disliked all three systems because he erroneously believed that the systems did not store his encrypted email in Gmail.

		Interview			
Participant	Survey	PGP	IBE	Passwords	Net Change
Johnny	PGP	–	3	0	−1
	IBE	2	–	1	+0
	Passwords	0	0	–	+1
Jane	PGP	–	1	0	+4
	IBE	4	–	1	−2
	Passwords	1	2	–	−2
Both	PGP	–	4	0	+3
	IBE	6	–	2	−2
	Passwords	1	2	–	−1

Table 7.5: Changes in Favorite System Between Survey and Interview

Approximately a sixth of participants (15; 16%) changed their favorite system after better understanding the security models of each system. These changes are detailed in Table 7.5. The differences were all small, with the only notable changes being Jane's overall preference for switching from IBE to PGP.

7.6 Other Results

We observed participants to determine how often they used various features in MessageGuard. We noted that Johnny frequently watched both the compose and read tutorials (Compose–[14; 87%], Read–[38; 81%]). Jane similarly watched the read tutorial (43; 91%), but rarely composed a new email and as such was rarely given a chance to view the compose tutorial[5] (6 out of 10 participants; 60%). We also found that Johnny was somewhat likely to include a plaintext greeting with his encrypted email (33; 70%). When Jane did send a new encrypted message, she included an unencrypted greeting a little under half of the time (4 of 10 participants; 40%).

[5] Encrypted replies do not contain plaintext greetings.

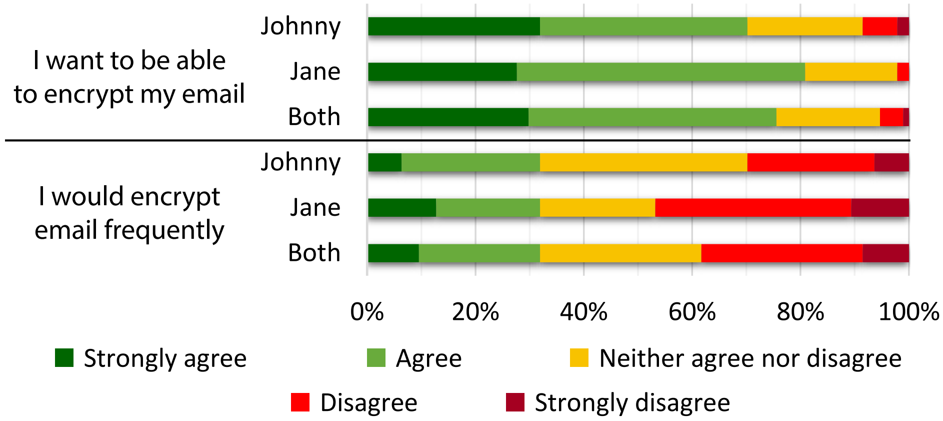


Figure 7.4: Participant Opinions Regarding Secure Email

We also recorded how Johnny sent the password he used to encrypt his email when using the Passwords system. Johnny largely preferred phone-based communication channels for communicating the password with Jane (cellular text messaging–23, phone call–11, email–9, Google Chat–4, in person–2, Facebook Chat–1).[6] Also, in three cases (phone call–2, email –1) Johnny did not actually transmit the password, but merely gave clues to Jane that were sufficient for her to figure it out.

At the end of the survey, participants were asked whether they wanted to be able to encrypt their email and whether they would frequently do so. Participant responses to these questions are summarized in Figure 7.4. Overall, participants were in strong agreement that email encryption is something they want (want–[71; 76%], unsure–[18; 19%], don't want–[5; 5%]). Still, participants were split on how often they would use secure email, with the plurality going to infrequent use (frequent use–[30; 32%], unsure–[28; 30%], infrequent use–[36; 39%]). This is in line with previous results regarding desired secure email usage [20].

[6] We note that these usage numbers do not sum to 47 as Johnny sometimes used multiple methods to communicate the password.

Chapter 8

Discussion

In this section we discuss participants' qualitative feedback and observations from the study coordinators. Within this section, participants have each been assigned a unique identifier R[1–47][A,B], where the final letter refers to the role of the participant (e.g., R1A and R1B were Johnny and Jane, respectively, in the first study session).

8.1 PGP

Our work succeeded at creating a highly usable PGP-based secure email system that was liked by participants. In general, participants described the PGP system as fast and easy-to-use.

The most common complaint regarding PGP was that recipients needed to install PGP before they could be sent encrypted messages. As stated by R1A, *"It's not great that sending someone an encrypted email means you have to ask them to download an extension."* Additionally, some participants felt that they were less likely to install the system if they didn't already have an encrypted message. For example, R9B described,

> *"I am more motivated (i.e., I can more readily see the need) to install the app if the encrypted message is already sitting there in my inbox. Also, the fewer emails I have to send/receive the better".*

A small number of participants also recognized that sending PGP messages to multiple participants could become cumbersome if most of the recipients had never used PGP before. In this regard, R7A said,

> *“[PGP] also didn’t let me send the email until they had done so [installed the system]. This would be annoying if I needed to send many emails for work or something similar. I would want to send the email and move onto the next task, without waiting for the other person to have the extension.”*

On the other hand, a small number of participants recognized that these extra steps were tied to PGP’s stronger security mode. In this vein, R19A stated,

> *“Easy to use and felt very secure. There were more steps to this version than the IBE version, but that made it seem more secure.”*

The annoyance of requiring recipients to first install PGP was somewhat alleviated by the fact that the system would automatically generate an invitation message explaining to the recipient how to install MessageGuard. This made it clear to participants what they needed to do next to send or receive an encrypted email. For example, participants R9A and R33A stated, respectively,

> *“I really like the idea of being able to send encrypted messages regularly. I also liked the automatic e-mail generated for having the recipients set up the system as well.”*

> *“I also liked that it was easy to send an email to someone who didn’t have MessageGuard–PGP and they could easily download it.”*

The most significant issue we discovered with our PGP system was that very few participants understood its security model (4; 4%), with most participants assuming that an attacker only needed access to the user’s email account to read their encrypted email. It is likely that because so much of PGP’s key management was automated (e.g., key generation, uploading and retrieval of of public keys) that participants had insufficient contextual clues to determine the system’s security model. While reducing the automation of the system could improve understanding, these changes would likely come at an unacceptably high usability cost [17, 24, 30]. Future work should examine how we can help users establish accurate mental models of PGP's security model in a way that does not significantly impact the usability of the system.

After explaining to participants PGP's actual security model, they felt much more confident in its security. Particularly, participants liked that it did not rely on any third parties. For example, after hearing about PGP's security model R47B enthusiastically changed her favorite system from Passwords to PGP and stated,

> *"Just because it had to be from your computer, it seems like, if they were to get the [encrypted contents], it'd be a little bit harder for them to get [the plaintext contents]."*

Participants' interest in PGP was tempered by the risk of losing all their encrypted email if something were to happen to the private key stored on their computer. In this regard, R18A expressed,

> *"I guess, depending on what you're doing, [PGP] could be helpful, but it could also be very frustrating... if you changed systems or something like that, it could be frustrating to realize that you couldn't decrypt previously sent messages."*

Future work could examine how to best backup and transfer private keys between devices in a usable and secure fashion.

8.2 IBE

Similar to prior results regarding IBE, participants found the system to be extremely usable. Additionally, task completion times show that IBE was faster than the other two systems.

Compared to prior version of secure email based on IBE, our version required that users sign up for an account before they could retrieve their IBE private key.[1] Having a

[1] Our PGP system also required users to establish a separate MessageGuard account.

separate account prevents a sometimes-malicious email provider from downloading the user's IBE private key.[2] While most users were fine with setting up an account, several participants indicated that they disliked the need to set up an account. For example, R3B and R5B respectively expressed,

> *"As a general comment, I think the password one was my favorite, since you didn't have to create an account for MessageGuard."*

> *"Easy to use, installed it, created an account (I liked that I had to create an account), worked well"*

During the user study, several participant pairs encountered an edge case for IBE—Jane had multiple email address aliases, and the message was encrypted for a different alias than Jane used when she set up her MessageGuard account. This resulted in Jane being unable to decrypt Johnny's message. This was especially confusing for Johnny and Jane as both of them had set up the secure email systems but had no indication of what they needed to do to resolve the current issue.

The difficulty of handling email aliases is not limited to IBE and affects PGP as well. As of now, it is unclear how best to solve this problem. MessageGuard's design anonymizes the identity of the recipients, which makes it difficult to tell users which email alias they need to register with their MessageGuard account. This an area that future work could examine, as it represents an important edge case for the adoption of secure email.

As with PGP, participants had a poor understanding of IBE's security model. In fact, nearly all participants assumed that PGP and IBE had the same security model. After instructing participants on IBE's security model, many participants were happy to hear it was more secure than they assumed. After understanding each system's security model, some participants who initially preferred IBE switched their preference to PGP; most remained with IBE, stating that it had good enough security. Additionally, these participants felt that

[2] As such, our IBE system has stronger security than the IBE systems previously developed by Ruoti et al. [16, 20].

IBE's ability to send an encrypted message to recipients without ensuring they had installed MessageGuard trumped the security drawbacks of IBE.

8.3 Passwords

While participants gave Passwords a lower SUS score than either PGP or IBE, they did feel that overall it was quite usable. Still, even though users rated Passwords as usable, a substantial number indicated that they liked PGP and IBE due to their not requiring a password to encrypt email. For example, R29A said, *"There was the benefit of not having to send a password by exterior means."* Also, R32A stated, *"It was simple to send a message and you didn't need to set up a password."*

The main problem with password-based encryption was the need to communicate the password to the recipient. As already discussed, many participants shared their password over plaintext email. In some cases, they recognized that this didn't seem secure, but still proceeded. For example, R32A said,

> *"The recipient of your email needs to know your password before they can open your email, so we ended up sharing the password through an non-encrypted email which seems counterproductive."*

Even when using an out-of-band channel, some participants questioned the security of those channels. R24B and R34B, respectively, described this concern:

> *"We also communicated the password through a text message. I'm not sure what that does for the security of the system if we are using an outside and unprotected means of communication in order to make it work."*

> *"We used a text to transport the password and I don't know that that is necessarily the safest way. If there was some way to have the person get the password safely, that would be great."*

Many participants also felt that communicating a password out-of-band negated the need to use secure email, as they could just communicate the sensitive information over the out-of-band channel. R39B indicated,

> *"It was way lame that I had to call him because I might as well have just given him the info that way. . . . If I'm gonna communicate with them through email, its because I want to do it through email, not through a phone call."*

Several participants also noted that if they had to securely communicate with multiple people, it would be annoying to manage separate passwords for each contact. In this regard, R9A expressed,

> *"I may want to use [Passwords] often in sending regular messages to many people. If I had to share a password each time, it may make the process cumbersome."*

After using the Passwords system, participants had several suggestions for how it could be improved. First, participants felt that within an email thread the password used to encrypt the email should be static and unchanging. While users could reuse the same password to encrypt replies, many participants became confused about this and created a new password, necessitating an additional round of password exchange. Second, some participants felt it would be helpful to have a built-in password complexity meter or random password generator when creating passwords. As stated by R25A and R18B, respectively,

> *"I would want it to tell me if my password is complex or even provide random passwords that are complex. I would want this to make myself feel more confident that what I was sending could not be hacked or decrypted easily be someone else that I did not intent."*

> *"If you don't have a random password generator, then people will just end up using familiar passwords, which is actually more of a problem than if there were no passwords at all."*

Unlike PGP and IBE, the security model for the Passwords system was well-understood by participants. This is likely due to the fact that users have an ingrained sense of how passwords protect their data. Understanding the security model of passwords helped users trust the system's security. As stated by R3A and R23A, respectively,

> *"It was nice to be able to create a password that only myself and the sender know. It felt more secure. . . . "*

> *"Though it is the most time consuming (and some would say the most hassle as well), it is obviously the most secure. I wouldn't use it for every email, but I would certainly use it for sending private information."*

8.4 Key Management Trade-offs

As partially discussed above, we identify usability and security trade-offs for each key management approach. These trade-offs were identified based on both our quantitative data and participants' qualitative responses. Overall, there were two clear trade-offs.

First, hiding cryptographic details increases usability, but inhibits understanding of a system's security model. For example, both IBE and PGP hid key management from the user, leading to high usability scores. However, in the post-study interview it was clear that participants did not understand the security model of either system. In contrast, the Passwords system required users to manually manage their keys (i.e., passwords). This led to lower usability scores for Passwords, but nearly all users understood its security model.

Second, tools that rely on third-party key servers sacrifice security, but significantly reduce the burden of adopting the system. For example, evaluations of PGP systems that use manual key exchange have consistently found these systems to be unusable [17, 24, 30]. In our PGP system, we employed a third-party key directory, which significantly improved its usability at the expense of trusting a third-party. Similarly, IBE fully trusts its third-party server with private keys, making it trivial to send any recipient an encrypted message. Even though participants recognized the lower security of IBE, many indicated that it still had "good enough" security for their needs.

8.5 User Attitudes Regarding the Design of Secure Email

We asked participants whether they would be interested in MessageGuard including a master password.[3] Overall, participants were interested in this feature (Johnny–[33; 70%], Jane–[35; 74%], Both–[72; 77%]). Participants felt that this would provide an important security property when multiple users shared a single computer. R6A and R18A expressed, respectively,

> *"I let others use my computer enough it would be nice to now my email was relatively safe when I was not in control."*

> *"I like the idea of having more control over what is visible to other users, especially in cases of using a shared computer."*

The participants that were not interested in a master password indicated that they had sole access to their computer, and that a master would add a hassle for no real security gain. As described by R3B and R9B, respectively,

> *"I wouldn't like the inconvenience of having to enter in an additional password when opening my browser."*

> *"My computer is password protected. While someone could theoretically get on and obtain sensitive information, I watch my computer like I watch my 1-year old: very closely. Not anticipating needing to protect my data from non-authorized users of my devices."*

[3] With a master password, MessageGuard would not encrypt or decrypt email until this password was entered. Moreover, cryptographic keys would be encrypted using the master password before being stored to disk.

Since a majority of participants were interested in a master password, future work should explore how to best implement a master password, with special attention paid to its potential usability impact.

We note that the split in participants' opinions regarding master passwords demonstrates two important principles of usable, secure email. First, the potential users of secure email have very different usage scenarios (e.g., shared computer vs. private computer). Second, no one set of features satisfies all users. While these principles may seem obvious, the sad reality is that they are not being respected by any of the industrial secure email systems we have previously tested. Our experience has shown that secure email tools are largely built for a single user group, and are not sufficiently customizable for groups that have different usability and security criteria.

Participants also expressed a strong desire to better understand how the secure email systems worked. They felt that this would help them verify that the system was properly protecting their data. Additionally, several participants stated that they would not feel comfortable using a "random" tool from the Internet. Instead, they looked for tools that were verified by security experts or were distributed and endorsed by a well-known brand (e.g., Google). R40B and R4A shared, respectively,

> *"I'm wary of downloading unfamiliar things because of viruses. But I don't know a lot about viruses.... I think to use it I would have to know it was really legit form a legit company, or approved by someone. I don't know if I could trust a random program with my personal info."*

> *"I need very strong reasons to believe that it's not just a way that the workers of MessageGuard can access my personal information. Like knowing that it's not a spam or can be broken into."*

Interestingly, a white paper on MessageGuard's design is available to view on the MessageGuard website, but only two users actually looked at it. In conjunction with participants' responses, this behavior suggests that users are not actually interested in personally reading about MessageGuard's security, but rather want these details available for critique by security-conscious friends and experts. Users would then base their trust in the system on these individuals' recommendations.

Several participants were especially delighted with MessageGuard's ability to verify the identity of the sender. For example, R16A stated, *"I could verify the sender and make sure the email wasn't from someone or something sketchy."* Similarly, R38 said, *"I like that it encrypts and that it tells you whether the name of the sender is their real name."* This indicates that users are aware of the ability for sender addresses to be spoofed, and that they are interested in secure email's ability to prevent this type of attack.

Finally, we note that users are interested in being able to toggle encryption for individual email messages in an email thread. In MessageGuard once an email is encrypted, future replies in that thread are always encrypted. While this helps better protect encrypted email, participants indicated that they would prefer to be able to turn off encryption for specific emails in a thread. For example, R45A indicated,

> *"When I reply to an e-mail chain, I would like to have the ability to turn off the encryption, instead of not having the option."*

8.6 Validation of Prior Research

Our research validates prior secure email research. First, our research confirms that usability modifications suggested by Atwater et al. [1]—obviating the use of master passwords, auto-generating invitational emails, using a key directory—do indeed increase the usability of PGP-based secure email. This confirmation is important as errors in Atwater et al.'s study made it unclear if their results were valid.

Third, our results demonstrate that the design principles for usable, secure email discovered by Ruoti et al. [16, 20] generalize beyond IBE, and are also applicable to PGP- and password-based systems. Moreover, we demonstrate that these principles are sufficient to make

PGP—a previously unusable type of secure email [17, 24, 30]—highly usable and preferred to other approaches by a third of our participants. Many participant responses demonstrated the importance of Ruoti et al.'s design principles (e.g., tight-integration; context-sensitive, inline tutorials; unencrypted greetings) in their high estimation of our systems' usability. For example, R7A, R9A, R11A, R26B, and R34B all commented on these various design principles:

> *"I really like the integration into Gmail, so that I can safely send information without having to use an entirely new system."*

> *"The tutorial was very helpful. I also found the icons to be helpful in using the tool. I was surprised at how easily the program integrated into my e-mail. There was never any confusion as to what I needed to do or as to what was going on."*

> *"The tutorials were great. I really felt like I didn't need to know much to be able to use it."*

> *"I like... that the subject/top of the email are not encrypted to help others realize that this is not spam."*

> *"Cause that [auto-generated PGP invitation email], where it was just this, 'Hey, I'm sending you an encrypted message', that felt very fake, kind of like those Skype messages when it's like, 'Hey I wanna be your friend', and you're like, I don't know you so I'm gonna delete it.... I feel like it [the greeting field] helped me realize it was him because it had that personality behind it."*

Finally, we gathered further evidence that paired-participant usability studies [19] are helpful in assessing the usability of secure email systems. When asked, participants indicated that they enjoyed working with a friend and that they felt it was more natural than it would be with a study coordinator; this was especially true for our Passwords system, where they indicated that calling their friend was natural, but not something they would feel comfortable doing with a coordinator. R7B stated,

> *"I think it was easier, just 'cause, the familiarity, I send her emails all the time, we message all the time, and so it was just like, it wasn't like, 'Am I allowed to do this, am I supposed to do this, like what kind of communication can I have?' Like, I know exactly how to talk to my wife, so it was really [easy]."*

Additionally, we note that in both our quantitative and qualitative data, we found several strong differences between Johnny and Jane. This indicates that there is value in gathering information regarding the usability of secure email for both roles.

Chapter 9

Conclusion

In our work, we compared the usability of three different key management approaches to secure email: PGP, IBE, and Passwords. To test these approaches, we used MessageGuard to build three secure email systems which each adopted one of these three key management schemes. The systems were built using state-of-the-art design principles for usable, secure email [1, 2, 16, 20]. We implemented these systems using a modular key management framework that we designed and built. This allowed each of our systems to share as much of their user interface as possible. We validated our design decisions through the use of a cognitive walkthrough. We then evaluated the usability of each system using a paired-participant study methodology [19]. This evaluation was the first formal A/B evaluation of the usability of different key management schemes. It is also the largest secure email study to date, including twice as many participants as previous studies [19].

The results of our study demonstrate that each of these three key management approaches can be used to create usable, secure email; though each has their own trade-offs. As such, our research represents the first time that PGP-based email encryption has been shown to be usable by novices. Additionally, participants' qualitative feedback provides valuable insights into the usability trade-offs of each key management approach, as well as several general principles of usable, secure email. Finally, our work provides evidence that validates both the design principles used in our systems as well as the study methodology.

Chapter 10

Future Work

The work described in this thesis invites several areas of investigation in the future. We found that the security model of password-based secure email is far more accessible and intuitive to users than that of PGP- or IBE-based secure email. This is an open problem in the field of usable security; there are currently no effective, vetted methods for educating users on the use of public and private keys, and the trust and security implications that accompany them. Now that usable secure email has been shown to be feasible, educating users on the nature of public key encryption is an even more important area of research. Users should understand the security assurances provided by their messaging platforms, and the precautions they should take to avoid security vulnerabilities or usability missteps. The greater degree of understanding users have in a system, the more educated decisions users can make with regard to what security and usability tradeoffs they are comfortable with.

Now that we have an effective platform for performing usability research into key management, there are several options for future research and development efforts. Instead of restricting MessageGuard users into one of several key management approaches, we can allow users to "upgrade" their security over time, escalating the security provided by the system as their proficiency and confidence in the system grows. For example, new users could get onboarded into MessageGuard through IBE. Once users wish to further improve their security by removing the reliance on a third-party key escrow server, they could be given the option to seamlessy transition to PGP encryption.

Password-based secure email was implemented using ephemeral keys; when users close their browser window, any passwords previously entered for encryption or decryption are lost. This is beneficial security-wise, but can have negative usability implications. We did not evaluate this during this study; our task did not involve forcing participants to close their browsers and be exposed to this behavior. Future work could A/B test password ephemerality and quantify the associated usability tradeoff.

As part of the post-study questionnaire, we asked participants about their opinions on master passwords, which could be used to protect encryption keys from unauthorized use. A majority of participants indicated that they would prefer having a master password, and future work could analyze its usability implications.

There are options for new key management approaches that might be studied. One intriguing option would be a fusion of IBE- and password-based secure email. This fusion would present a stronger security model than either approach on its own. Neither leakage of the password, nor compromise of the key escrow server, would be sufficient on their own to expose users' messages to attack. The usability implications of this approach bear further investigation.

The security assurances provided by PGP with a trusted third-party key directory are highly similar to those provided by S/MIME. In the future, MessageGuard could be extended to support S/MIME-based secure email. This would allow the usability of S/MIME to be evaluated and compared against the usability metrics gathered in this work.

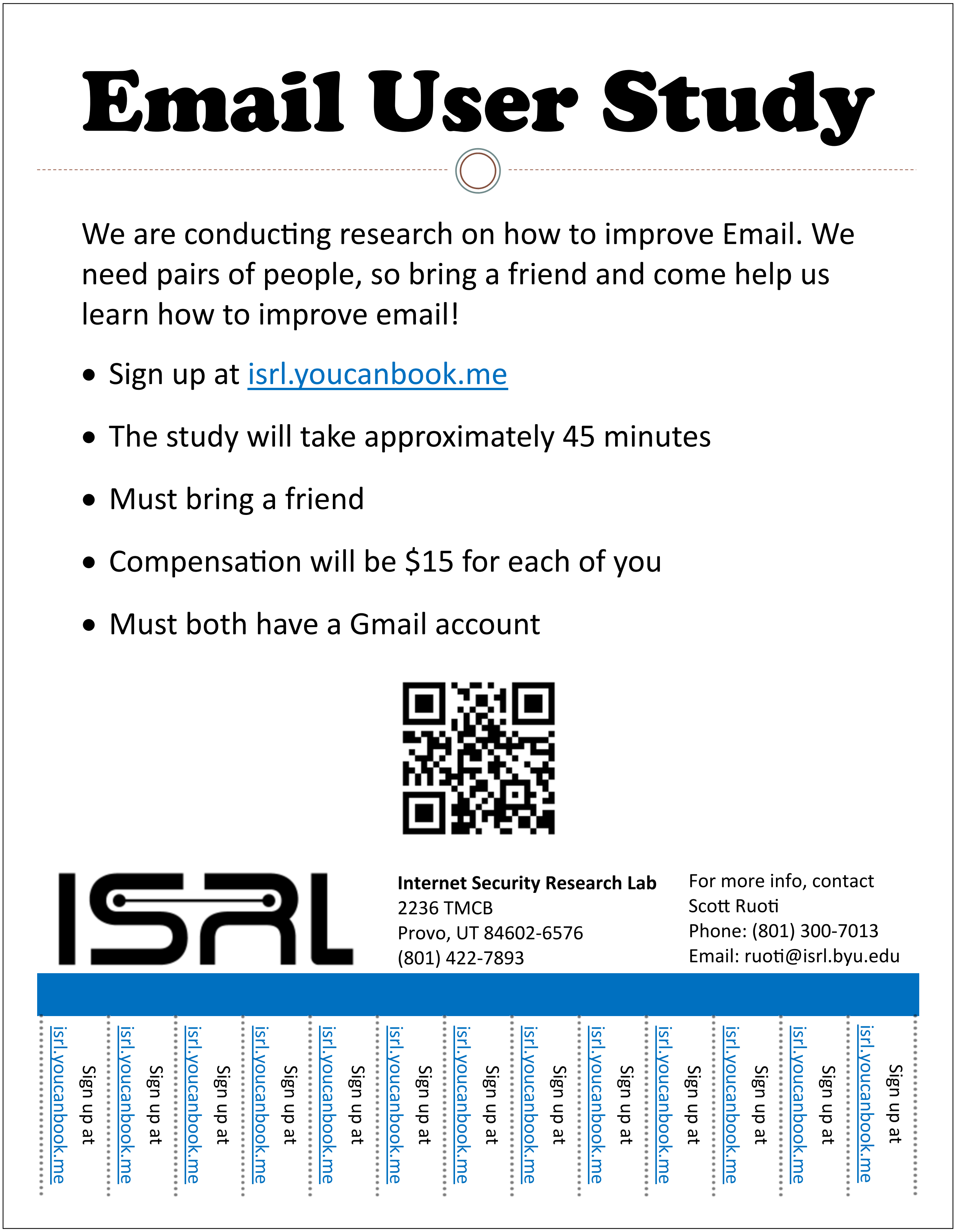
Finally, future work could involve longitudinal studies of MessageGuard users. Our study provided a brief introduction to MessageGuard and short-term observation. By observing long-term use of MessageGuard, valuable information could be gained with regard to how secure email might be used in the real world.

Appendices

Appendix A

Study Documents

A.1 Recruitment Poster



Email User Study

We are conducting research on how to improve Email. We need pairs of people, so bring a friend and come help us learn how to improve email!

- Sign up at isrl.youcanbook.me
- The study will take approximately 45 minutes
- Must bring a friend
- Compensation will be $15 for each of you
- Must both have a Gmail account

ISRL

Internet Security Research Lab
2236 TMCB
Provo, UT 84602-6576
(801) 422-7893

For more info, contact
Scott Ruoti
Phone: (801) 300-7013
Email: ruoti@isrl.byu.edu

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

Sign up at isrl.youcanbook.me

A.2 Study Coordinator Instructions

1. Have each participant sign two copies of the consent form. Give one copy to the participant to keep.
2. Use a coin flip to determine who is Johnny.
3. Johnny will remain in this room and Jane will go next door. (Door code: xxxxx)
   (a) Ask the participant to sit down. Invite them to adjust the chair if they wish.
   (b) Tell them, **"You and your friend are in different rooms, and will need to work together to complete a task. During this task, we will provide you with some information that needs to be sent over email. Other than this information, you can feel free to communicate with your friend however you normally would. While you are waiting for email from your friend, feel free to relax and use your phone or the Internet."**
4. Do the following:
   (a) Start the audio recorder.
   (b) Open "Open Broadcasting Software". Start recording.
   (c) On the desktop, click the "Start Survey" icon.
5. Before using each system, the survey will instruct the participant to tell you they are ready to begin the next task. When they do so, complete the following steps:
   (a) (Johnny) Look at which system the participant will be using, and provide Johnny with the appropriate information sheet.
   (b) (Jane) Provide Jane with the generic information sheet.
   (c) Start the VM software and resume the snapshot [VM→snapshot→Two Person Study (2016)].

(d) Change the view to full screen-exclusive mode.

   (View→Full Screen; View→Exclusive).

(e) Notify the other coordinator which system will be used.

(f) Record in the notes the order the systems are used.

6. During the course of the task pay attention to the following items:

   (a) (Jane) When Jane decrypts her email, give her the appropriate information sheet for her to complete the task.

   (b) Make notes of anything interesting you see.

   (c) If the participant sends sensitive information in the clear, make a note of this, then instruct them that they need to use the secure email system to send that information.

   (d) **Note how participants transmit passwords (e.g., phone call, text, email).**

   (e) During the study, participants may have questions for you. Answer any questions regarding the study task, but do not instruct participants on how to use the systems being tested. Instead, encourage them to continue trying.

   (f) In case users wrote their codes down incorrectly, we have included them at the end of this document.

7. When the task is complete, the participants will be instructed to tell you they have finished the task. When they do so, complete the following steps:

   (a) Ensure that the participants have correctly completed the task.

   (b) Exit exclusive mode by pressing **`right ctrl + right alt`**.

   (c) Restore the snapshot [VM→snapshot→Two Person Study (2016)].

   (d) Switch to the survey and have the participant continue the survey.

8. **When the survey is finished, ask the participant about their experience.**

(a) Ask the participants about any problems they encountered during the study and how they dealt with them. Try and understand what the user was thinking. Also ask the participant if something in MessageGuard could be changed to address this issue.

(b) Ask them about anything you felt was unusual or unique in their experience.

(c) For each key management scheme (**follow the order they used the systems in**):

   i. Ask participants who can read their messages. If unclear, ask them what would an attacker need to do to steal their secure email.

   ii. **Record whether the user correctly understood the scheme in the notes.**

(d) For each key management scheme (not concurrent with previous bullet, **follow the order they used the systems in**):

   i. **"I will now describe to you what an attacker would need to do in order to read your encrypted email. If you have any questions about my descriptions or how the systems work, feel free to ask."**

   ii. Explain to the users the security provided by each scheme. **Descriptions are provided on the next page.**

   iii. Ask the participant if, based on this information, their opinion on any system changes.

   iv. Ask the participant which system they would prefer to use in the real-world with their friends.

   v. **Record this information in the notes.**

9. Close out the individual portion of the study.

   (a) Stop the video recording.

    (b) (Jane) Stop the audio recording, and bring your participant back to the main room.

10. Now that the participants are together, ask the participants about their experience.

    (a) "How would your ideal email encryption system function? If you would like to, feel free to use the whiteboard to sketch ideas."

    (b) "What did you think about doing a study with a friend?"

11. Close out the study.

    (a) (Johnny) Stop the audio recording.

    (b) Clean the whiteboard if needed.

    (c) Thank the participants for their time.

    (d) Help them fill out the compensation form, and direct them to the CS office.

A.3 Participant Worksheets

A.3.1 Participant A Worksheet

In this task, you'll be using MessageGuard - {PGP, IBE, Passwords}.

Go to https://messageguard.io/{pgp, ibe, passwords} and get the tool.

Please encrypt and send the following information to your friend using MessageGuard - {PGP, IBE, Passwords}:

- SSN: XXX-XX-XXXX
- PIN: XXXX

Enter the confirmation code provided by your friend:

Enter the PIN provided by your friend:

Once you have received the confirmation code and PIN from your friend, send an email to your friend letting them know you received this information. After you have sent this confirmation email, let the study coordinator know you have finished this task.

A.3.2 Participant B General Worksheet

Please wait for your friend's email with their last year's tax PIN and SSN.

Enter your friend's SSN. Include dashes.

Enter your friend's PIN.

Once you have written down your friends SSN and PIN, let the study coordinator know that you are ready to reply to your friend with their confirmation code and PIN.

A.3.3 Participant B System-Specific Worksheet

You have completed your friends taxes and need to send them the confirmation code and this years tax PIN from their tax submission.

Since your friend used MessageGuard - {PGP, IBE, Passwords} to send sensitive information to you, please also use MessageGuard - {PGP, IBE, Passwords} to send them the confirmation code and PIN.

- Confirmation code: XXXXXXXX
- PIN: XXXX

Once you have sent the confirmation code and PIN to your friend, wait for them to reply to you and confirm they got the information. Once you have gotten this confirmation, let the study coordinator know you have finished this task.

A.4 Descriptions of Key Management Schemes

These descriptions were read to study participants following the study questionnaire.

A.4.1 PGP

In the {first, second, third} system you tested, your email was secured using PGP. In PGP, when you installed the system, a lock and key were created. The lock was stored on the MessageGuard website, allowing anyone to download it and use it to encrypt email for you. The key is kept on your own computer and is needed to decrypt your email. To read your encrypted email, an attacker would need to break into your computer and steal this key.

In PGP, your recipients need to install the system and generate their lock and key before you can encrypt and send email to them. If you lose or delete your key, email encrypted with your lock will be inaccessible.

A.4.2 IBE

In the {first, second, third} system you tested, your email was secured using IBE. In IBE, anyone can encrypt email for you, and the key to decrypt that email is stored on the MessageGuard website. To read your email, an attacker would need to break into the MessageGuard account you created during the study, and steal your key. Because the MessageGuard website does not have access to your email, it cannot decrypt it.

A.4.3 Passwords

In the {first, second, third} system you tested, your email was secured using a password you or your friend chose. To read your email, an attacker would need to steal or guess that password.

References

[1] Erinn Atwater, Cecylia Bocovich, Urs Hengartner, Ed Lank, and Ian Goldberg. Leading Johnny to water: Designing for usability and trust. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 69–88, Montreal, Canada, 2015. USENIX Association.

[2] Wei Bai, Moses Namara, Yichen Qian, Patrick Gage Kelley, Michelle L. Mazurek, and Doowon Kim. An inconvenient trust: User attitudes toward security and usability tradeoffs for key-directory encryption systems. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, pages 113–130, Denver, CO, 2016. USENIX Association.

[3] Aaron Bangor, Philip Kortum, and James Miller. An empirical evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[4] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123, 2009.

[5] John Brooke. SUS—a quick and dirty usability scale. In *Usability Evaluation in Industry.* CRC Press, Boca Raton, FL, 1996.

[6] John Brooke. SUS: A retrospective. *Journal of Usability Studies*, 8(2):29–40, 2013.

[7] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J. Alex Halderman. Neither snow nor rain nor MITM. . . : An empirical analysis of email delivery security. In *Fifteenth ACM Internet Measurement Conference (IMC 2015)*, pages 27–39, Tokyo, Japan, 2015. ACM.

[8] Ian D. Foster, Jon Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko. Security by any other name: On the effectiveness of provider based email security. In *Twenty-Second ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 450–464, Denver, CO, 2015. ACM.

[9] Simson Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly Media, Inc., Sebastopol, CA, 1995.

[10] Simson L. Garfinkel and Robert C. Miller. Johnny 2: A user test of key continuity management with S/MIME and Outlook Express. In *First Symposium on Usable Privacy and Security (SOUPS 2005)*, pages 13–24, Pitsburg, PA, 2005. ACM.

[11] Ralph Holz, Johanna Amann, Olivier Mehani, Matthias Wachs, and Mohamed Ali Kaafar. TLS in the wild: An internet-wide analysis of TLS-based protocols for electronic communication. In *Twenty-Fourth Network and Distributed System Security Symposium (NDSS 2016)*, San Diego, CA, 2016. The Internet Society.

[12] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Secure key issuing in ID-based cryptography. In *Second Workshop on Australasian Information Security (AISW 2004)*, pages 69–74, Darlinghurst, Australia, 2004. Australian Computer Society, Inc.

[13] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Michael J. Freedman, and Edward W. Felten. CONIKS: A privacy-preserving consistent key service for secure end-to-end communication. In *Twenty-Fourth USENIX Security Symposium (USENIX Security 2015)*, pages 383–398, Washington, D.C., 2015. USENIX Association.

[14] Stanley Milgram and Ernest Van den Haag. *Obedience to Authority.* Ziff–Davis Publishing Company, New York, NY, 1978.

[15] Karen Renaud, Melanie Volkamer, and Arne Renkema-Padmos. Why doesn't Jane protect her privacy? In *Fourteenth Privacy Enhancing Technologies Symposium (PETS 2014)*, pages 244–262, Philadelphia, PA, 2014. Springer.

[16] Scott Ruoti, Nathan Kim, Ben Burgon, Timothy Van Der Horst, and Kent Seamons. Confused Johnny: When automatic encryption leads to confusion and mistakes. In *Ninth Symposium on Usable Privacy and Security (SOUPS 2013)*, Newcastle, United Kingdom, 2013. ACM.

[17] Scott Ruoti, Jeff Andersen, Daniel Zappala, and Kent Seamons. Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client, 2015. arXiv preprint arXiv:1510.08555.

[18] Scott Ruoti, Brent Roberts, and Kent Seamons. Authentication melee: A usability analysis of seven web authentication systems. In *Twenty-fourth International Conference on World Wide Web (WWW 2015)*, pages 916–926, Florence, Italy, 2015. ACM.

[19] Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark ONeill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent Seamons. "We're on the same page": A usability study of secure email using pairs of novice users. In *Thirty-Fourth ACM Conference on Human Factors and Computing Systems (CHI 2016)*, pages 4298–4308, San Jose, CA, 2016. ACM.

[20] Scott Ruoti, Jeff Andersen, Travis Hendershot, Daniel Zappala, and Kent Seamons. Private Webmail 2.0: Simple and easy-to-use secure email. In *Twenty-Ninth ACM User Interface Software and Technology Symposium (UIST 2016)*, Tokyo, Japan, 2016. ACM.

[21] Scott Ruoti, Jeff Andersen, Tyler Monson, Daniel Zappala, and Kent Seamons. MessageGuard: A browser-based platform for usable, content-based encryption research, 2016. arXiv preprint arXiv:1510.08943.

[22] Jeff Sauro. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC, Denver, CO, 2011.

[23] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Fourteenth International Cryptology Conference (Crypto 1984)*, pages 47–53, Santa Barbara, CA, 1984. Springer.

[24] S. Sheng, L. Broderick, C.A. Koranda, and J.J. Hyland. Why Johnny still can't encrypt: Evaluating the usability of email encryption software. In *Poster Session at the Symposium On Usable Privacy and Security*, Pitsburg, PA, 2006.

[25] Andreas Sotirakopoulos, Kirstie Hawkey, and Konstantin Beznosov. "I did it because I trusted you": Challenges with the study environment biasing participant behaviours. In *Usable Security Experiment Reports Workshop at the Symposium On Usable Privacy and Security*, Redmond, WA, 2010.

[26] Thomas S. Tullis and Jacqueline N. Stetson. A comparison of questionnaires for assessing website usability. In *Usability Professional Association Conference*, pages 1–12, Minneapolis, MN, 2004. Usability Professionals Association.

[27] Timothy W. Van Der Horst and Kent E. Seamons. Simple authentication for the web. In *Third International Conference on Security and Privacy in Communications Networks (SecureComm 2007)*, pages 473–482, Nice, France, 2007. IEEE Computer Society.

[28] Timothy W. van der Horst and Kent Eldon Seamons. Encrypted email based upon trusted overlays, March 2009. US Patent 8,521,821.

[29] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. The cognitive walkthrough method: A practitioner's guide. In *Usability Inspection Methods*, pages 105–140, New York, NY, USA, 1994. John Wiley & Sons, Inc.

[30] Alma Whitten and J.D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Eighth USENIX Security Symposium (USENIX Security 1999)*, pages 14–28, Washington, D.C., 1999. USENIX Association.