A Comparative Usability Study of Key Management in Secure Email

Scott Ruoti, University of Tennessee; **Jeff Andersen, Tyler Monson, Daniel Zappala, and Kent Seamons,** Brigham Young University

https://www.usenix.org/conference/soups2018/presentation/ruoti

This paper is included in the Proceedings of the Fourteenth Symposium on Usable Privacy and Security.

August 12–14, 2018 • Baltimore, MD, USA

ISBN 978-1-931971-45-4

Open access to the Proceedings of the Fourteenth Symposium on Usable Privacy and Security is sponsored by USENIX.

A Comparative Usability Study of Key Management in Secure Email

Scott Ruoti
University of Tennessee
ruoti@utk.edu

Jeff Andersen
Brigham Young University
andersen@isrl.byu.edu

Tyler Monson
Brigham Young University
monson@isrl.byu.edu

Daniel Zappala
Brigham Young University
zappala@cs.byu.edu

Kent Seamons
Brigham Young University
seamons@cs.byu.edu

ABSTRACT

We conducted a user study that compares three secure email tools that share a common user interface and differ only by key management scheme: passwords, public key directory (PKD), and identity-based encryption (IBE). Our work is the first comparative (i.e., A/B) usability evaluation of three different key management schemes and utilizes a standard quantitative metric for cross-system comparisons. We also share qualitative feedback from participants that provides valuable insights into user attitudes regarding each key management approach and secure email generally. The study serves as a model for future secure email research with A/B studies, standard metrics, and the two-person study methodology.

1. INTRODUCTION

The cryptography needed to deploy secure email is well studied and has been available for years, and a number of secure email systems have been deployed and promoted recently, including ProtonMail, Tutanota, Mailvelope, Virtru, Voltage, Encipher.it, etc. While some of these systems have millions of users, the vast majority of email users still do not use secure email [21]. The lack of adoption of secure email is often attributed to the significant gap between what the technology can offer and the ability of users to successfully use the technology to encrypt their emails.

Beginning with Whitten and Tygar [36], secure email usability studies have shown that key management is a significant hurdle for users. More recent usability studies (e.g., [1, 2, 23]) show signs that progress toward greater usability is being made, but limitations in each study make it difficult to draw conclusions regarding the impact key management has on secure email usability, other than the need for automation. We previously conducted studies [23, 24, 26, 27] that directly compared key management schemes from different families, but the systems implementing the various key management schemes were wildly different, introducing a significant confounding factor. Bai et al. [2] compared two key management schemes, but their study explored user mental models and trust, not usability generally.

Additionally, even though public key directories have recently received significant attention [19, 29], it is unclear how their usability compares to other key management schemes. Lerner et al. [18] studied a public key directory system but didn't use a standard metric, making it difficult to directly compare their results to past work. Atwater et al. [1] simulated a public key directory, but permitted a user to send an email to a recipient who had not yet generated a key pair. Normally, when a user attempts to send an email to a recipient who has not yet generated a key pair, they must wait until the user does so and uploads their public key to the key directory. Because this affected numerous participants in their study, it is unclear how this issue impacted their results.

Our work was motivated by the desire to build on these earlier studies and reduce the number of confounding factors in order to increase our confidence in the resulting usability measurements. In this paper, we describe a user study comparing three key management schemes, taken from different families, to better understand how key management impacts the usability of secure email during initial setup and first use of the system. Using the MessageGuard research platform, we built three secure email tools which differ only in the key management scheme they implement (passwords, public key directory [PKD], and identity-based encryption [a]), reducing potential confounding factors in the study. In our study design we used a standard metric, allowing comparison to results from past studies. Finally, we replicated our earlier paired participant study setup [23], allowing us to evaluate grass roots adoptability.

In total, 47 pairs of participants completed our study. All three systems received favorable ratings from users, with server-derived public keys being considered the most usable, followed by user-generated public keys, and finally shared secrets. Each system performed better than similar (i.e., same key management) systems previously studied in the literature. Users also provided valuable qualitative feedback helping identify pros and cons of each key management scheme.

The contributions of this paper include:

1. **First A/B evaluation of key management using standard metrics.** Our study was able to confirm Atwater et al.'s [1] findings that public key directories

Copyright is held by the author/owner. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee.
USENIX Symposium on Usable Privacy and Security (SOUPS) 2018.
August 12–14, 2018, Baltimore, MD, USA.

are usable. Additionally, we find evidence that the secure email design principles we identified in previous work [24] generalize beyond server-derived public keys.

2. **The MessageGuard platform.** To enable this work, we built MessageGuard, a research platform for building secure email and other end-to-end encryption prototypes. MessageGuard significantly simplifies the effort required to work in this space and provides a means whereby research results may be shared and replicated. MessageGuard has a pluggable architecture, making it easy to build prototype variants for use in A/B testing.

3. **Lessons learned and recommendations.** Our study elicits user attitudes regarding the three key management schemes we evaluate, including security and usability trade-offs identified by participants. For example, even after understanding that the user-generated public key scheme protects against a stronger threat model than server-derived public keys, many users indicate that they do not need that level of security and prefer server-derived public keys because they can immediately send email without waiting for the recipient to generate a public/private keypair. Based on our findings, we give recommendations for future work.

2. BACKGROUND

In this section, we first describe several key management schemes commonly used with end-to-end email encryption. Next, we provide a chronological review of usable secure email research.

2.1 Key Management

We study three families of key management schemes used in end-to-end encryption of email: shared secrets, user-generated public keys, and server-derived public keys. Each has different methods for creating, sharing, and linking cryptographic keys to email addresses. We describe each scheme briefly; a more complete treatment can be found in [9].

2.1.1 Shared Secrets

Users can encrypt their emails using symmetric keys derived from a secret shared between pairs of users. Most commonly, these secrets are in the form of simple passwords, which are more readily communicated and remembered by users than cryptographically secure random values. The security of this key management scheme is dependent on users' ability to satisfy the following requirements when they create and share passwords: (1) choose a unique password for each user they will communicate with, (2) choose passwords that will resist a brute-force attack, (3) communicate passwords over a secure channel, and (4) safely store passwords.

2.1.2 User-generated Public Keys

Before sending or receiving encrypted email, users must first generate a cryptographic key pair. A user's private key should never be shared with any other party and must be safely stored by the user. The user's public key, with relevant metadata, is then distributed to other users 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.

There are numerous ways to verify the authenticity of a public key (i.e., the binding of a public key to an email address), some of which include:

1. **Manual validation.** Users can directly communicate with each other and directly share their public key or compare key fingerprints[1]. Users are expected to know each other personally and thus be able to confirm the identity of those they are communicating with.

2. **Web of trust.** Users can have their public key signed by one or more other users, who are expected to only sign public keys that they have verified using manual key validation. When retrieving a public key, users check to see if it has been signed by a user they trust to have validated it properly. Users may choose to transitively trust public keys that are trusted by users they trust, forming a web of trust.

3. **Hierarchical validation.** Users can have their public key signed by an authoritative signer (e.g., a certificate authority), which will only sign a public key after verifying that the user who submitted it owns the associated email address. When retrieving a public key, its signature is validated to ensure that it was properly signed by an authoritative signer. This method of key validation is most commonly associated with S/MIME [8, 13].

4. **Public key directory.** Users can submit their public keys to a trusted key directory. This directory will only accept and disseminate public keys for which it has verified that the user who submitted the key owns the associated email address. Due to its trusted nature, keys retrieved from the directory are assumed to be authentic. The behavior of the key directory can be audited through the use of certificate transparency [29] or a CONIKS-like ledger [19].

Manual verification and the web of trust are commonly associated with PGP [12], though any of the above can be used with PGP.

The security of these schemes depends on the ability of users to protect their private keys, obtain necessary public keys, and faithfully validate these public keys. If users lose access to their private keys (e.g., disk failure with no backup), they will be unable to access their encrypted email.

2.1.3 Server-derived Public Keys

In this scheme, a user's public key is generated for them by a server they trust, which may also store their private key (called key escrow). This alleviates the problems associated with a user losing their private key, and is often used in corporate environments. A variant of this scheme is identity-based encryption (IBE) [31]. With IBE, a user's public key is generated mathematically based on their e-mail address and public parameters provided by an IBE key server. A user's private key is also generated by the IBE key server, which will only release that key to the user after the user verifies ownership of the associated email address. In any situation

[1] A public key's fingerprint is typically derived from a cryptographic hash of the public key.

when a user cannot trust a server with their private key (e.g., an activist in an oppressive regime, or a journalist that needs to protect sources) key escrow should not be used.

2.2 Usable Secure Email

Whitten and Tygar [36] conducted the first formal user study of a secure email system (PGP 5 with manual key validation), uncovering serious usability issues with key management and users' understanding of the underlying public key cryptography. The results of their study took the security community by surprise and helped shape modern usable security research.

Garfinkel and Miller [13] created a secure email system using S/MIME (hierarchical key validation) and demonstrated that automating key management provides significant usability gains. However, their study also revealed that the tool "was a little too transparent," leading to confusion and mistakes.

We previously created Private WebMail (Pwm) [27], a secure email system that tightly integrates with Gmail and uses identity-based encryption (IBE) to provide key management that is entirely transparent to users. User studies of Pwm demonstrate that it was viewed very positively by users, and significantly outperformed competing secure email systems.

Atwater et al. [1] compared the usability and trustworthiness of automatic versus manual encryption, finding that there were no significant differences between the two approaches. As part of this study, Atwater et al. developed two email clients—one integrated with Gmail and one standalone—both of which simulated the user experience of using a public key directory.

We also developed a novel two-person methodology [23] for studying the usability and grassroots adoptability of secure email. In particular, this study involved recruiting pairs of recipients (e.g., friends, spouses), who would then be responsible for sending secure email among themselves. Compared to single-participant studies, this methodology revealed differences between the experience of initiating others and being initiated by others into using secure email. Our study compared systems using three different families of key management: shared password, public key directory, and IBE; unfortunately, confounding factors in this study make it difficult to draw any conclusion on how key management affects secure email's usability.

Bai et al. explored user attitudes toward 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 manual key validation and one that used a public key directory. Users were provided with instructions on how to use each tool and given several tasks to complete. The results of this study showed that, overall, individuals recognized the security benefits of manual key validation, but preferred the public key directory and considered it to have sufficient security. While this study gathered data on user attitudes regarding two key management schemes, it did not evaluate their usability.

More recently, we further refined our Pwm system [24], identifying four design principles that increase the usability, correct behavior, and understanding of secure email: (1) having informative and personalized initiation messages that guide users through installing the secure email software and give them confidence that the email they received is not malicious; (2) adding an artificial delay during encryption to build trust in the system and show users who their message is being encrypted for; (3) incorporating inline, context-aware tutorials to assist users as they are sending and receiving their first encrypted emails; and (4) using a visually distinctive interface to clearly demarcate which content is encrypted/to-be-encrypted and helping users avoid accidentally sending sensitive information in the clear.

	Whitten and Tygar [36]	Garfinkel and Miller [13]	Ruoti et al. [27]	Atwater et al. [1]	Ruoti et al. [23, 26]	Bai et al. [2]	Ruoti et al. [24]	Lerner et al. [18]	This work
Comparative			✓		✓	✓	✓		✓
A/B Study						✓			✓
Standard metric			✓	✓	✓		✓		✓
Two-person					✓				✓

Table 1: Comparison of Usable Secure Email Research

Lerner et al. [18] built Confidante, a secure email tool that leverages Keybase, a public key directory, for key management. A user study of Confidante with lawyers and journalists demonstrated that these users could quickly and correctly use the system.

The significant differences between this earlier work and our current work are summarized in Table 1.

3. SYSTEMS

To limit confounding factors in our study, it was necessary to build several secure email tools that differed only in how key management was handled. To accomplish this we substantially modified our Private WebMail 2.0 (Pwm 2.0) system [24], leaving its UI unchanged, but otherwise completely rewriting its codebase to add support for a pluggable key management subsystem. This allowed us to rapidly develop three secure email prototypes that only differed in how they handled key management, while keeping the remaining system components consistent. We call this pluggable version of Pwm 2.0 *MessageGuard*.

We choose to extend Pwm 2.0 for several reasons. First, it is an existing system with established favorable reviews, saving us a significant amount of development time and helping avoid the possibility of designing a new secure email tool that was viewed unfavorably by users. Second, it had the highest usability score [24] of any secure email systems evaluated using the System Usability Metric (SUS) [6]. Third, this allowed us to test whether the secure email design principles proposed by Ruoti et al. and implemented in Pwm 2.0 (see Section 2.2) generalize beyond IBE-based systems.

In addition to adding a pluggable key management system to MessageGuard, we also added several other features to MessageGuard in order to allow other researchers to use it as a research platform for building end-to-end encryption

prototypes. First, MessageGuard supports a wide range of non-email sites (e.g., Facebook, Twitter, Blogger), automatically scanning these pages for user-editable content and allowing users to encrypt this content end-to-end. Second, the page scanning functionality is pluggable, allowing researchers to create finely-tuned, per-site end-to-end encryption plugins. Finally, MessageGuard includes pluggable user interface, encryption, and content packaging subsystems.

There are three key benefits to using MessageGuard as a research platform:

1. *Accelerates the creation of content-based encryption prototypes.* MessageGuard provides a fully functional content-based encryption system, including user interfaces, messaging protocols, and key management schemes. The modular design of MessageGuard allows researchers to easily modify only the portions of the system they wish to experiment with, while the remaining portions continue operating as intended. This simplifies development and allows researchers to focus on their areas of expertise—either usability or security.
2. *Provides a platform for sharing research results.* Researchers who create prototypes using MessageGuard can share their specialized interfaces, protocols, or key management schemes as one or more patches, allowing researchers to leverage and replicate each other's work. Additionally, research can be merged into MessageGuard's code base, allowing the community to benefit from these advances and reducing fragmentation of efforts.
3. *Simplifies the comparison of competing designs.* MessageGuard can be used to rapidly develop prototypes for use in A/B testing. Two prototypes built using MessageGuard will only differ in the areas that have been modified by researchers. This helps limit the confounding factors that have proven problematic in past comparisons of content-based encryption systems.

The source code for MessageGuard is available at `https://bitbucket.org/account/user/isrlemail/projects/MES`.

In the remainder of this section, we give a brief overview of MessageGuard. Additional details are available in Appendix A–C, and a complete description can be found in a technical report [25]. Next, we describe the workflow for the three secure email variants that we created using MessageGuard. We chose well-known instances of each key management scheme and explain the rationale for that choice: passwords, public key directory (PKD), and IBE. Other alternatives and hybrids of these approaches are possible.

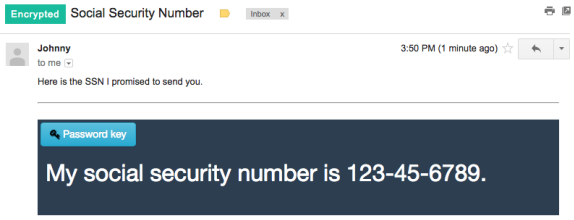
These systems can be downloaded and are available for testing at `https://{pgp,ibe,passwords}.messageguard.io`

3.1 MessageGuard

MessageGuard tightly integrates with existing web applications, in this case Gmail, using *security overlays*. 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 (Figure 1a and Figure 1b).

Figure 2 shows the MessageGuard architecture:

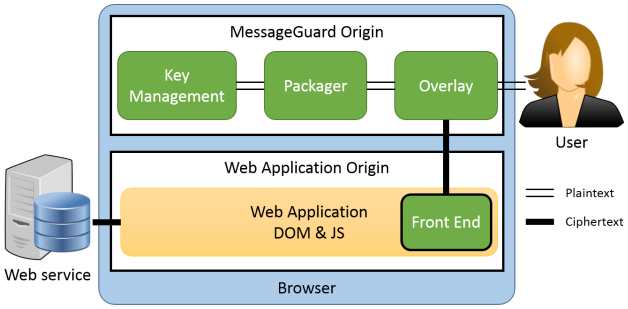
(a) Composition Overlay



(b) Read Overlay

Figure 1: MessageGuard Overlays

- The **front end** scans for encrypted payloads and data entry interfaces and replaces these items with a secure overlay. The front end is the only component that runs outside of MessageGuard's protected origin, and it can only communicate with overlays using the `window.postMessage` API. The overlay always encrypts user data before transmitting it to the front end component and sanitizes any data it receives from the front end. In addition, the front end also displays tutorials that instruct new users how to use MessageGuard. These are all context-sensitive, appearing as the user performs a given task for the first time.
- **overlays** use `iframes` and the browser's same-origin policy to keep plaintext from being exposed to the email server and its application. A read overlay displays sensitive information to the user, and a compose overlay allows users to encrypt sensitive information before sending it to the website. Overlays have a distinctive, dark color scheme that stands out from most websites, allowing users to easily identify secure overlays from insecure website interfaces.
- The **packager** encrypts/decrypts user data and encodes the encrypted data to make it suitable for transmission through web applications. The packager uses standard cryptographic primitives and techniques to encrypt/decrypt data (e.g., AES-GCM). Ciphertext is packaged with all information, save the key material, necessary for recipients of the message to decrypt it.
- The **key management** component enables a variety of key management schemes to be configured, without changing other aspects of MessageGuard such as the read or compose overlays.



A user's sensitive data is only accessible within the MessageGuard origin.

Figure 2: MessageGuard Architecture

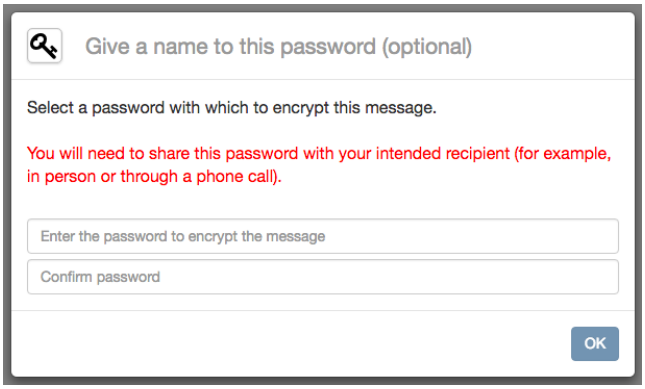


Figure 3: Dialog for Entering a New Password with Which to Encrypt Email.

3.2 Passwords

We choose to evaluate passwords as they are a scheme that should be familiar to users. The workflow for our password system is as follows:

1. The user visits the MessageGuard website. They are prompted to download the system.
2. After installation, the system is immediately ready for use.
3. When the user attempts to send an encrypted email, they are informed that they need to create a password for encrypting the email (see Figure 3). After creating the password, the user can 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.

3.3 Public Key Directory (PKD)

We choose to evaluate public key directories because they have received significant attention lately [2, 18, 19, 29]. The workflow for our public key directory system is as follows:

1. The user visits the MessageGuard website. They are instructed to create an account with their email address.[2] Their address is verified by having the user click a link in an email sent to them. They are then able to download the system.
2. After installation, the user is told that the system will generate a 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 that the recipient hasn't yet installed the system.[3] They are then prompted to send their recipient an email inviting them to install the system. This email message is auto-generated by MessageGuard, with the system able to add a custom introduction message if desired.
4. Once the recipient has installed the system, which generates and publishes their public key, they inform the sender that they are ready to proceed. The sender can now send their encrypted email.

3.4 Identity-based Encryption (IBE)

We choose to evaluate IBE because it is the key management scheme that has been shown to be most usable in past studies, providing a good baseline for this work. The workflow for our IBE-based system is as follows:

1. The user visits the MessageGuard website. They are instructed to create an account with their email address.[2] Their address is then verified by having the user click a link in an email sent to them. They are then able to download the system.
2. After installation, the user is informed that the system will retrieve their IBE key from the key server. This happens automatically because the user is already authenticated to the key server from the previous step.
3. The user can send encrypted email to any address.
4. The recipient, upon receiving the encrypted email, is prompted to visit the MessageGuard website and create an account. After their address is verified and their private key is downloaded from the key server, they can read the encrypted message.

4. METHODOLOGY

We conducted a within-subjects, IRB-approved lab study wherein pairs of participants used three secure email systems to communicate sensitive information to each other (study materials are found in Appendix D). Our study methodology is patterned after our previous paired participant methodology [23], allowing us to examine usability in the context of

[2]We chose to require a MessageGuard account in order to prevent a compromised email provider from being able to transparently upload (PKD) or download (IBE) cryptographic keys from the MessageGuard key server, which would be possible if these operations were only protected by email-based authentication.

[3]The recipient must install the system and use it to upload a public key before the sender can encrypt email for the recipient.

two novice users, without potential bias or other behaviors introduced by direct involvement with a study coordinator.

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).

4.1 Study Setup

Participants took 50–60 minutes to complete the study, and each participant was compensated $15 USD in cash. Participants were required to be accompanied by a friend, who served as their counterpart for the study, and were instructed to use their own Gmail accounts.[4]

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 need to use email to share some sensitive information with each other. They were told that they were free to communicate with each other however they normally would, with the caveat that the sensitive information they were provided must be transmitted over email. 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. Finally, participants were told that a study coordinator would be with them at all times and could answer questions related to the study but were not allowed to provide instructions on how to use any of the systems being tested.

4.2 Study Tasks

Using a coin flip, one participant was randomly assigned as Participant A and the other as Participant B (referred to as "Johnny" and "Jane", respectively, throughout the paper). The participants were then led to separate rooms to begin the study. The participants were then guided through the study by following a Qualtrics survey, which included both instructions and then questions regarding their experience.

After answering demographic questions, participants were asked to complete a multi-stage task three times, once for each of the secure email systems being tested. The order in which the participants used the systems was randomized. To complete this task, 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 he could download at a URL we gave him. Jane was told that she would receive some information regarding taxes from Johnny but was not informed that the information would be encrypted.

The tasks they were asked to perform were:

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.
3. After Johnny received this information, he would inform Jane that he had received the necessary information, and then the task would end. This confirmation step is added to ensure that Johnny could decrypt Jane's message. We did not require the confirmation message to be encrypted.

During each stage, participants were provided with worksheets containing instructions regarding the task and space for participants to record the sensitive information they received. These instructions did not include directions on how to use any of the systems. 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 tasks for each of the three systems being tested.

Immediately upon completing the tasks 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) [6, 7]. Multiple studies have shown that SUS is a good indicator of perceived usability [34], is consistent across populations [28], and has been used in the past to rate secure email systems [1, 23, 24, 27]. Next, participants were asked to describe what they liked about each system, what they would change, and why they would change it.

After completing the tasks and questions for all three secure email systems, participants were asked to select which of the 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.

4.3 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

[4]Using their own accounts increases ecological validity, but has privacy implications. To help mitigate these concerns we have destroyed the screen recordings for this study. Though not used, we did prepare study accounts for any participants who were not comfortable using their own account.

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.

4.4 Quality Control

We excluded responses from eight pairs of participants.[5] First, three pairs were removed because the secure email tools became inoperative during the study, making it impossible for participants to complete the study.[6] Second, two pairs were removed because the participants did not speak or read English well enough to understand the study instructions and study coordinators. Third, we removed three participant pairs that were not paying attention to the study survey and filled in nonsense answers.

4.5 Demographics

We recruited Gmail users for our study at a local university, as well as through Craigslist. We distributed posters across campus to avoid biasing our participants toward any particular major. 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%). 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 technical and non-technical majors.

4.6 Limitations

Our study involved each 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 examine other usage scenarios.

Our study also has several common limitations. First, our population is not representative of all groups, and future research could broaden the population (e.g., non-students, non-Gmail users). While we did use Craigslist to try and gather a more diverse population, these efforts were largely unsuccessful. Second, our study was a short-term study, and future research should look at these issues in a longer-term longitudinal study. Third, since our study was run in a trusted lab environment, participants may not have behaved the same as they would in the real world [20, 33].

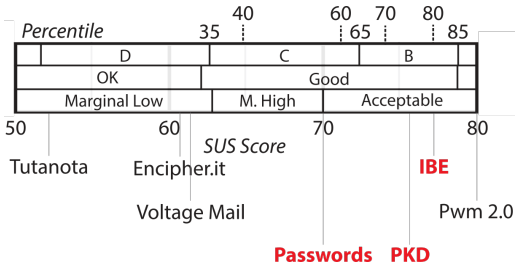
[5]When we excluded a participant's results, we also excluded their partner's results.

[6]These errors were not related to the usability of the system. For example, in one case, the Chrome Webstore went down, making it impossible for users to download the necessary extensions.

	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)	Range	Percentile
Passwords	94	70.0	15.0	±3.0	67.0–73.0	56%
PKD	94	75.7	14.9	±3.0	72.7–78.7	76%
IBE	94	77.3	13.5	±2.7	74.6–80.0	81%

Percentiles are calculated by looking up the SUS score in a table [30]. When a SUS score is not in the table we estimate the percentile based on the available data.

Table 2: SUS Scores



The red items are systems evaluated in our study. The black items are systems evaluated in previous work that share key management schemes with the systems we tested: Encipher.it uses passwords, Tutanoa uses a public key directory, Pwm 2.0 and Voltage Mail use IBE.

Figure 4: Adjective-based Interpretation of SUS Scores

5. RESULTS

This section contains the quantitative results from our study: the SUS score for each system, task completion times, mistakes made by participants, participant understanding of each system's security model, rankings for the favorite system, and several other minor results. For brevity, we refer to the three variants tested as Passwords, PKD (public key directory), and IBE (identity-based encryption). The data for this study can be downloaded at `https://isrl.byu.edu/data/soups2018/`.

In several situations, we performed multiple statistical comparisons on the same data. In these cases, we use the Bonferroni correction to adjust our α value appropriately. Where a correction is not needed, we used the standard value $\alpha = 0.05$.

5.1 System Usability Scale

The System Usability Scale (SUS) score for each system is listed in Table 2. 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, 30, 34]. The descriptions are presented in Figure 4.

Passwords' score of 70.0 is rated as having "Good" usability, receives a "C" grade, and reaches the 56$^{\text{th}}$ percentile. PKD's SUS score of 75.7 is rated as having "Good" usability, receives a "B" grade, and falls in the 76$^{\text{th}}$ 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 81$^{\text{st}}$ percentile.

A one-way repeated measures ANOVA comparing the effect of system on SUS scores revealed a statistically significant omnibus ($F(2, 186) = 13.43$, $p < .001$). The difference between Passwords' and PKD's scores are statistically significant (Tukey's HSD test—$p < 0.01$) as is the difference between Passwords' and IBE's SUS scores (Tukey's HSD test—$p < 0.01$). In both cases, the differences in means represent a significant improvement (20 and 25 percentile difference, respectively). In contrast, the difference between PKD's and IBE's SUS scores are not statistically significant. We also tested to see whether there was a difference between the SUS score ratings of Johnny and Jane, but the difference was not statistically significant (two-tailed student t-test, matched pairs—$p = 0.29$, $\alpha = 0.0125$).

Next, we compared the SUS scores for our variants against SUS scores of publicly available systems that used the same key management schemes. In each case our secure email variants outperformed these publicly available systems. We compared Encipher.it [27] against our Password variant, which scored 8.75 points higher (~25 percentile difference), Tutanota [23] against our our PKD variant, which scored 23.5 points higher (~60 percentile difference), and Voltage Mail against our IBE variant [27], which scored 14.64 points higher (~45 percentile difference).

Finally, we explored whether the order in which systems were tested had an effect on their SUS scores, finding three orderings with a non-negligible effect size: (1) Passwords scored 9.5 points higher when tested immediately after PKD, (2) PKD scores 9.5 points lower when it is tested after Passwords, (3) IBE scores 14.1 points lower when the system ordering is Passwords->IBE->PKD. All three of these differences are statistically significant (two-tailed student t-test, equal variance—$p < 0.001$, $p = 0.002$, $p < 0.001$, respectively, $\alpha = 0.0125$).

5.2 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 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 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. We took this approach because stage one is clearly not finished, but Jane is also able to start making progress on completing stage two.

	Stage	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)	Range
Passwords	1	46	3:31	1:25	±0:25	03:06–03:56
	2	44	6:54	3:34	±1:03	05:51–07:57
	1 + 2	43	10:22	4:00	±1:12	09:10–11:34
PKD	1	47	8:02	3:06	±0:53	07:09–08:55
	2	45	3:24	1:28	±0:26	02:58–03:50
	1 + 2	45	11:33	3:53	±1:08	10:25–12:41
IBE	1	46	3:30	1:30	±0:26	03:04–03:56
	2	44	5:58	2:36	±0:46	05:12–06:44
	1 + 2	43	9:30	3:50	±1:09	08:21–10:39

Table 3: Time Taken to Complete Task (min:sec)

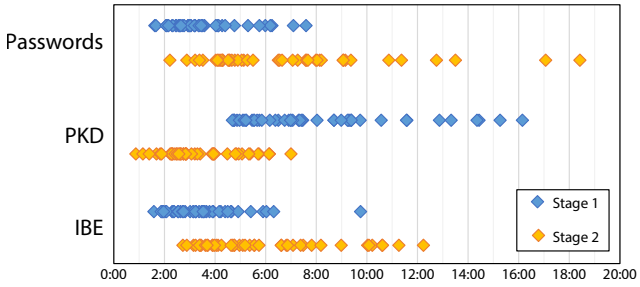


Figure 5: Individual Participant Task Completion Times

Timings were calculated using the video recordings of each participant's screen. We had missing or corrupted video in four cases. Task completion time data from the remaining recordings is given in Table 3 and Figure 5.

A two-way repeated measures ANOVA comparing the effect of system and stage on stage completion time fails to find a statistically significant overall difference between systems, but does reveal a statistically significant interaction effect (System—$F(2, 82) = 2.60$, $p = .08$, Stage—$F(1, 41) = 1.936$, $p < .0.17$, Interaction—$F(2, 82) = 82.52$, $p < .001$). By design, PKD shifts a significant portion of user effort from Stage 2 to Stage 1—Jane installs PKD in Stage 1 instead of Stage 2—resulting in a statistically significant difference in stage completion times (Tukey's HSD test—in all cases $p < 0.001$) with a large effect size (Stage 1—+4:30, Stage 2—−3:00). The difference between Passwords and IBE was not statistically significant for either Stage 1 or Stage 2.

We also explored whether system ordering had an effect on task completion times. As shown in Table 4, if a system was the first system tested, its task took considerably longer to complete than if it was not the first system tested. This difference is statistically significant for all three systems (two-tailed student t-test, equal variance—in all cases $p < 0.001$, $\alpha = 0.016$).

5.3 Mistakes

We define mistakes to be instances when users send sensitive information in normal email when it should have been

	Stage	Count	Mean When First	Mean When Not First	Effect Size
Passwords	1	46	4:50	3:01	−1:49 (−38%)
	2	44	9:49	5:48	−4:01 (−41%)
	Both	43	14:48	8:50	−5:58 (−40%)
PKD	1	47	9:36	7:13	−2:23 (−25%)
	2	45	4:20	2:54	−1:26 (−33%)
	Both	45	13:56	10:14	−3:42 (−27%)
IBE	1	46	4:47	2:49	−1:58 (−41%)
	2	44	8:01	4:48	−3:13 (−40%)
	Both	43	12:55	7:39	−5:16 (−41%)

Table 4: Time Taken to Complete Task as a Function of Whether it Was Tested First (min:sec)

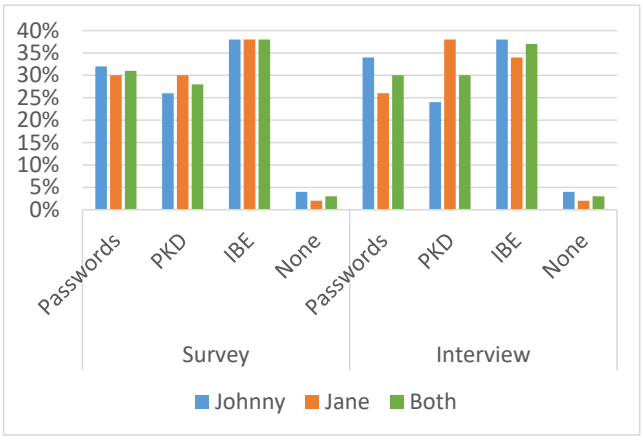


Figure 6: Participants' Favorite System

encrypted. For Passwords, a user is also considered to have made a mistake if they send the encryption password in a plaintext email.[7]

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. Additionally, in four cases Johnny used Google Chat to send their password, giving Google access to both the secure email and 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.

In PKD and IBE there were a low number of mistakes, and each was made by Johnny (PKD–[n = 1; 2%], IBE–[2; 4%]). In all three cases, the participant transmitted the sensitive information in the unencrypted greeting[8] of the encrypted message. This happened in spite of the fact that two of these participants watched the compose tutorial, which warned them that text in that field would not be encrypted.[9]

5.4 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 asked follow-up questions until they were confident that they could judge whether the participant had a correct understanding.

[7]Mistakes could conceivably also include revealing PKD or IBE private keys, but neither of our systems allowed users to make this mistake.

[8]The MessageGuard front end provides an unencrypted greeting field, which senders can populate with text readable by recipients who have not installed MessageGuard, aiding in the onboarding process.

[9]This problem could potentially be addressed by making users explicitly enable unencrypted greetings, instead of displaying it as a default field.

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).

Few participants had a correct understanding of PKD'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 was meaningful. During the interviews, most participants indicated they saw no difference in the security of IBE and PKD.

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%]).

5.5 Favorite System

At the end of the study survey, participants were asked to indicate their favorite system, 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 6.

Overall, participants were split on which system they preferred (During Survey—PKD–[26; 28%], IBE–[36; 38%], Passwords–[29; 31%]; After Interview—PKD–[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 he disliked all three systems because he erroneously believed that the systems caused his encrypted email to not be stored by Gmail.

Approximately a sixth of participants (15; 16%) changed their favorite system after better understanding the security

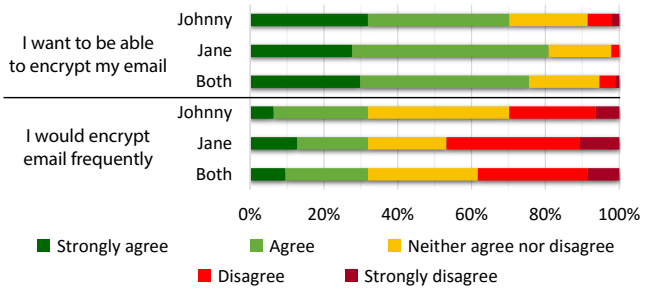


Figure 7: Participant Opinions Regarding Secure Email

models of each system: one from Passwords to PKD, two from passwords to IBE, four from PKD to IBE, six from IBE to PKD, and two from IBE to Passwords. In total, Passwords lost one vote, PKD gained three votes, and IBE lost two votes.

5.6 Other Results

We also recorded how often participants used various features in MessageGuard. We noted that Johnny frequently watched both the compose and read tutorials (Compose–[41; 87%], Read–[38; 81%]). Jane similarly watched the read tutorial (43; 91%), with a slightly lower rate of watching the compose tutorial (6 out of 10 participants; 60%).[10] We found that Johnny was 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%).[11]

We noted that Johnny used a variety of methods to transmit the password used to encrypt his email, overall preferring phone-based communication channels (cellular text messaging–23, phone call–11, email–9, Google Chat–4, in person–2, Facebook Chat–1).[12] In three cases (phone call–2, email–1) Johnny did not 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. 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 [24].

6. QUALITATIVE RESULTS

In this section we discuss participants' qualitative feedback and observations from the study coordinators. We refer to participants using a unique identifier R[1–47][A,B], where A refers to the Johnny role and B refers to the Jane role.

[10] Jane only saw the compose tutorial if she started a new email chain.

[11] Encrypted replies do not contain plaintext greetings.

[12] These usage numbers do not sum to 47 as Johnny sometimes used multiple methods to communicate the password.

6.1 Passwords

Participants gave Passwords a lower SUS score than both PKD and IBE, but overall indicated it was quite usable. Even though users rated Passwords as usable, a substantial number indicated they preferred PKD and IBE due to these systems not requiring a password to encrypt email.

Communicating the password to the recipient was the main problem with password-based encryption. As already discussed, many participants shared their password over plaintext email. In some cases, they recognized this didn't seem secure, but still proceeded. Some participants questioned the security of using out-of-band channels to send the password.

> *"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."* [R24B]

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, it's because I want to do it through email, not through a phone call."*

Several participants noted it would be annoying to manage separate passwords while communicating securely with multiple people. 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."*

Participants had several suggestions to improve Passwords. First, participants proposed allowing only a single password to protect an email thread. Users could reuse passwords to encrypt replies, but many participants became confused and created new passwords, necessitating more password exchanges. Second, some participants felt that it would be helpful to have a built-in password complexity meter or random password generator when creating passwords.

> *"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."* [R18B]

Unlike PKD and IBE, the security model for the Passwords system was well-understood by participants. Understanding the security model of passwords helped users trust the system's security.

> *"It was nice to be able to create a password that only myself and the sender know. It felt more secure. . . . "* [R3A]

6.2 PKD

In general, participants described the PKD system as fast and easy-to-use. The most common complaint about PKD

was that recipients needed to install PKD 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 they were less likely to install the system if they didn't already have an encrypted message.

> *"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."* [R9B]

The most significant issue we discovered with our PKD system was that very few participants understood its security model (4; 4%), with most participants assuming an attacker only needed access to the user's email account to read their encrypted email. After explaining PKD's security model to participants, 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 PKD's security model R47B enthusiastically changed her favorite system from Passwords to PKD 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 PKD was tempered by the risk of losing all their encrypted email if something were to happen to the private key stored on their computer.

> *"I guess, depending on what you're doing, [PKD] 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."* [R18A]

6.3 IBE

Similar to previous studies [23, 24, 27], participants found IBE to be extremely usable. Task completion times show that IBE was faster than the other two systems.

Prior implementations of IBE relied on automatic email authentication to deliver private keys [24, 27]. Our implementation has users create a username and password on the key server for authenticating a request to retrieve a private key.[13] This prevents the email provider from being able to access the user's private key. This added security can impact usability. While most users did not mind setting up an account, several participants disliked this aspect.

> *"As a general comment, I think the password one was my favorite, since you didn't have to create an account for MessageGuard."* [R3B]

As with PKD, participants had a poor understanding of IBE's security model. Nearly all participants thought PKD and IBE had poor security, incorrectly believing that anyone who broke into their Gmail account could read all encrypted emails. After receiving instructions on IBE's security model, some participants who initially preferred IBE switched their preference to PKD; most remained with IBE, stating it had adequate security. Additionally, these participants felt that the ability to send an IBE-encrypted message to a recipient without waiting for them to first install MessageGuard trumped the security drawbacks of IBE.

[13] Our PKD system also required users to create an account.

6.4 User Attitudes

We asked participants if they would be interested in MessageGuard including a master password. 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.[14] Overall, participants were interested in this feature (Johnny–[33; 70%], Jane–[35; 74%], Both–[72; 77%]). Participants felt this would provide an important security property when multiple users shared a single computer. The participants not interested in a master password indicated they had sole access to their computer, and a master would add a hassle for no real security gain.

Participants also expressed a strong desire to better understand how the secure email systems worked. They felt this would help them verify the system was properly protecting their data. Additionally, several participants stated 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).

7. DISCUSSION

We discuss lessons learned, usability and security trade-offs, and validation of prior work.

7.1 Lessons Learned

It is unclear whether the mistake of sending the password via email represents users' lack of understanding regarding the security of email [22], a lack of concern for the safety of their sensitive information during the role play, an artifact of taking the study in a trusted environment [33], or a mixture of the three.

With so much of PKD's key management automated (e.g., key generation, uploading and retrieval of public keys), it is likely participants had insufficient contextual clues showing 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 [23, 26, 32, 36]. Future work should examine ways the system could conform to users' existing mental models.

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 because they had no indication of what they needed to do to resolve the issue. MessageGuard's design anonymizes the identity of the recipients, so the system could not inform Jane which email alias she needed to register with her MessageGuard

[14] The master password differs from the MessageGuard account password in that the former is used only locally to protect access to cryptographic keys stored on the local device, whereas the latter is used to protect against an adversary uploading (PKD) or downloading (IBE) cryptographic keys to/from the MessageGuard key server. Users could choose to use the same password for both use cases.

account in order to read the message. The difficulty of handling email aliases is not limited to IBE. It affects PKD as well. It is unclear how best to solve this problem, and this is an area for future work.

7.2 Usability and Security Trade-offs

Hiding cryptographic details increases usability, but inhibits understanding of a system's security model.[15] For example, both IBE and PKD hid key management from the user, leading to high usability scores. However, post-study interviews revealed participants did not understand the security model of either system. In contrast, the Passwords system required users to manually manage their keys (using passwords). This led to lower usability scores for Passwords, but nearly all users understood its security model.

Tools relying on third-party key servers sacrifice security but significantly reduce the burden of adopting the system. For example, evaluations of PKD systems using manual key exchange have consistently found these systems to be unusable [26, 32, 36]. Our PKD system significantly improved its usability at the expense of trusting a third-party by employing a public key directory. 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 had "good enough" security for their needs.

7.3 Validation of Prior Research

Our results demonstrate that the design principles we identified in previous work [24, 27] generalize beyond IBE, and are also applicable to PKD and password-based systems. Many favorable participant responses demonstrated the importance of tight-integration; context-sensitive, inline tutorials; and unencrypted greetings (R7A, R9A, R26B, respectively):

> *"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."*

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

We also gathered further evidence showing paired-participant usability studies [23] are helpful in assessing the usability of secure email systems. Both the quantitative and qualitative data revealed strong differences between Johnny and Jane, indicating that there is value in gathering information for both roles. When asked, participants indicated they enjoyed working with a friend and felt it was more natural than working with a study coordinator. This was especially true for our Passwords system, where they indicated calling their friend was natural, but not something they would feel comfortable doing with a coordinator.

[15] Understanding a system's security model is important as it allows users to understand what actions are safe and what put them at risk.

8. CONCLUSION

The paper compared the usability of three different key management approaches to secure email: passwords, public key directory, and IBE. The systems were built using state-of-the-art design principles for usable, secure email [1, 2, 24, 27] and were evaluated using standard metrics and a paired-participant study methodology [23]. This evaluation was the first A/B evaluation of key management schemes in which participants were allowed to self-discover how the system worked. It is also the largest secure email study to date (94 participants), which is twice as large as previous studies [23].

Our research demonstrates that each key management approach has the potential to be successfully used in secure email. 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 prior work on the design principles [24] used in our systems as well as the study methodology [23].

While our results are very positive, they are focused on helping users begin using secure email. Further research is needed regarding how secure email systems, including MessageGuard, perform when used on a day-to-day basis. Based on our experience, we make the following recommendations for this future research:

- The public key directory scheme requires that users store and backup their private keys securely and reliably. They also need to transfer them between devices. Future work should explore users' ability to do so, as this could be a potential usability impediment that would also greatly reduce security.
- Future work needs to examine how to design encrypted email systems that support key email functionality, including spam filtering and search.
- Given the promising results for the various key management schemes in a laboratory setting, the next step is to design and conduct longitudinal studies to see if the results hold over an extended period in real-world scenarios.
- Participants in our study struggled to understand the threat model of the public key directory and IBE schemes. This is problematic inasmuch as users overestimate the security of the system and send sensitive data they would not if they properly understood the system's threat model. Future work should examine how tutorials can be constructed to address this issue. Particular care should be taken to validate that tutorials will not be ignored by users when completing secure email tasks.
- Future email studies should compare features of interest using A/B tests, standard metrics, and a two-person methodology to increase the confidence in results from these studies and also help situate new results clearly within the existing body of work.

Acknowledgment

We thank the anonymous reviewers and our shepherd, Marian Harbach, for their suggestions that helped improve the paper. This work was supported in part by the National Science Foundation under Grant No. CNS-1528022.

References

[1] E. Atwater, C. Bocovich, U. Hengartner, E. Lank, and I. 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. USENIX Association, 2015.

[2] W. Bai, M. Namara, Y. Qian, P. G. Kelley, M. L. Mazurek, and D. 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. USENIX Association, 2016.

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

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

[5] C. Bravo-Lillo, L. Cranor, J. Downs, S. Komanduri, S. Schechter, and M. Sleeper. Operating system framed in case of mistaken identity: measuring the success of web-based spoofing attacks on OS password-entry dialogs. In *Nineteenth ACM SIGSAC Conference on Computer and Communications Security (CCS 2012)*, pages 365–377, Raleigh, NC. ACM, 2012.

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

[7] J. Brooke. SUS: a retrospective. *Journal of Usability Studies*, 8(2):29–40, 2013.

[8] R. Chandramouli, S. L. Garfinkel, S. J. Nightingale, and S. W. Rose. Trustworthy email. *Special Publication (NIST SP) 800-177*, 2016.

[9] J. Clark, P. C. van Oorschot, S. Ruoti, K. Seamons, and D. Zappala. Securing Email. *ArXiv e-prints*, Apr. 2018. arXiv: `1804.07706 [cs.CR]`.

[10] R. Dhamija and J. D. Tygar. The battle against phishing: Dynamic Security Skins. In *First Symposium on Usable Privacy and Security (SOUPS 2005)*, pages 77–88, Pittsburgh, PA. ACM, 2005.

[11] S. Fahl, M. Harbach, T. Muders, and M. Smith. Confidentiality as a service–usable security for the cloud. In *Eleventh International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2012)*, pages 153–162, Liverpool, England. IEEE Computer Society, 2012.

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

[13] S. L. Garfinkel and R. 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, Pittsburgh, PA. ACM, 2005.

[14] W. He, D. Akhawe, S. Jain, E. Shi, and D. Song. ShadowCrypt: encrypted web applications for everyone. In *Twenty-First ACM SIGSAC Conference on Computer and Communications Security (CCS 2014)*, pages 1028–1039, Scottsdale, AZ. ACM, 2014.

[15] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Seventeenth New Security Paradigms Workshop (NSPW 2009)*, pages 133–144, Oxford, England. ACM, 2009.

[16] T. W. v. d. Horst and K. E. Seamons. Encrypted email based upon trusted overlays, 2009. US Patent 8,521,821.

[17] B. Lau, S. Chung, C. Song, Y. Jang, W. Lee, and A. Boldyreva. Mimesis aegis: a mimicry privacy shield—a system's approach to data privacy on public cloud. In *Twenty-Third USENIX Security Symposium (USENIX Security 2014)*, pages 33–48, San Diego, CA. USENIX Association, 2014.

[18] A. Lerner, E. Zeng, and F. Roesner. Confidante: usable encrypted email: a case study with lawyers and journalists. In *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*, pages 385–400. IEEE, 2017.

[19] M. S. Melara, A. Blankstein, J. Bonneau, M. J. Freedman, and E. 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. USENIX Association, 2015.

[20] S. Milgram and E. V. d. Haag. *Obedience to Authority*. Ziff–Davis Publishing Company, New York, NY, 1978.

[21] H. Orman. *Encrypted Email: The History and Technology of Message Privacy*. Springer, 2015.

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

[23] S. Ruoti, J. Andersen, S. Heidbrink, M. O'Neill, E. Vaziripour, J. Wu, D. Zappala, and K. 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. ACM, 2016.

[24] S. Ruoti, J. Andersen, T. Hendershot, D. Zappala, and K. 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. ACM, 2016.

[25] S. Ruoti, J. Andersen, T. Monson, D. Zappala, and K. Seamons. MessageGuard: a browser-based platform for usable, content-based encryption research, 2016. arXiv preprint arXiv:1510.08943.

[26] S. Ruoti, J. Andersen, D. Zappala, and K. Seamons. Why Johnny still, still can't encrypt: evaluating the usability of a modern PGP client, 2015. arXiv preprint arXiv:1510.08555.

[27] S. Ruoti, N. Kim, B. Burgon, T. Van Der Horst, and K. Seamons. Confused Johnny: when automatic encryption leads to confusion and mistakes. In *Ninth Symposium on Usable Privacy and Security (SOUPS 2013)*, Newcastle, United Kingdom. ACM, 2013.

[28] S. Ruoti, B. Roberts, and K. 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. ACM, 2015.

[29] M. D. Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *Twenty-Second Network and Distributed System Security Symposium (NDSS 2014)*, San Diego, CA. The Internet Society, 2014.

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

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

[32] S. Sheng, L. Broderick, C. Koranda, and 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*, Pittsburgh, PA, 2006.

[33] A. Sotirakopoulos, K. Hawkey, and K. 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.

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

[35] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. SoK: secure messaging. In *Thirty-Sixth IEEE Symposium on Security and Privacy (S&P 2015)*, pages 232–249, San Jose, CA. IEEE Computer Society, 2015.

[36] A. Whitten and J. 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. USENIX Association, 1999.

APPENDIX

A. MESSAGEGUARD'S DESIGN GOALS

In this section, we give the threat model that motivates our work. Next, we describe how to implement security overlays in order to enhance existing web applications with content-based encryption. Finally, we discuss our goals for MessageGuard, that are necessary to support research of content-based encryption in a usable, secure, and extensible manner.

A.1 Threat Model

In content-based encryption, sensitive content is only accessible to the author of that data and the intended recipient. In contrast to transport-level encryption (e.g., TLS), which only protects data during transit, content-based encryption protects data both during transit and while it is at rest. In our threat model, we consider web applications, middleboxes (e.g. CDNs), and the content they serve to be within the control of the adversary. The adversary wins if she is able to use these resources to access the user's encrypted data. While it is true that most websites are not malicious, in order to support ubiquitous, content-based encryption, it is necessary to protect against cases where websites are actively trying to steal user content. Users' computers, operating systems, software, and content-based encryption software[16] are all considered part of the trusted computing base in our threat model.

Our threat model is concerned with ensuring the confidentiality and integrity of encrypted data, but does allow for the leakage of meta-data necessary for the encrypted data to be transmitted and/or stored by the underlying web application. For example, in order to transmit an encrypted email message, the webmail system must have access to the unencrypted email addresses of the message's recipient. Additionally, the webmail provider will be able to inspect the encrypted package and gain learn basic information about the encrypted package (e.g., approximate length of message, number of recipients).[17]

While our threat model is necessarily strict to support the wide range of web applications that researchers may wish to investigate, we note that research prototypes built using the MessageGuard platform are free to adopt a weaker threat model that may be more appropriate for that particular research.

A.2 Security Overlays

There are several approaches for implementing overlays: `iframes` [16, 27], the `ShadowDOM` [14], user script engines such as Greasemonkey [11], and the operating system's accessibility framework [17]. Based on our analysis of each of these approaches, `iframes` are the implementation strategy best suited to work across all operating systems and browsers (including mobile). Additionally, `iframe`-based security overlays have security and usability that are greater than or equal to that of other approaches. As such, we designed MessageGuard using security overlays based on `iframes`.

Relying on `iframes` largely restricts MessageGuard to supporting only web applications deployed in the browser. Still the browser is an ideal location for studying content-based encryption: (1) There are a large number of high-usage browser-based web applications (e.g., webmail, Google Docs). (2) Traditional desktop and mobile application development increasingly mimics web development, allowing lessons learned in browser-based research to also apply to these other platforms. (3) There is already a substantial amount of research into adding content-based encryption to web applications, both academic (e.g., [1, 11, 14, 27]) and professional (e.g., Virtru, Mailvelope, Encipher.it).

[16] This includes the software's website and any web services the software relies upon (e.g., a key server).

[17] This type of leakage also occurs in HTTPS.

A.3 Platform Goals

We examined the existing work on content-based encryption (e.g., [13, 32, 35, 36]) in order to establish a set of design goals for MessageGuard. These goals are centered around enabling a researcher to investigate usable, content-based encryption.

A.3.1 Secure

MessageGuard should secure users' sensitive content from web applications and network adversaries.

MessageGuard should protect data in its overlays from being accessed by the web application. Sensitive data that is being created or consumed using MessageGuard should be inaccessible to the underlying web application. A corollary to this rule is that no entities that observe the transmission of data encrypted by MessageGuard should be able to decipher that data unless they are the intended recipients.

MessageGuard's interfaces should be clearly distinguishable from the web application's interfaces. In addition to protecting content-based messages from websites, it is important that systems clearly delineate which interfaces belong to the website and which belong to the content-based encryption software. This helps users to feel assured that their data is being protected and assists them in avoiding mistakes [24, 27]. Additionally, visual indicators should be included that can help protect against an adversary that attempts to social engineer a user into believing they are entering text into a secure interface when in reality they are entering text directly into the adversary's interface [5, 10].

A.3.2 Usable

MessageGuard should provide a usable base for future research efforts.

MessageGuard should be approachable to novice users. Easy-to-use systems are more likely to be adopted by the public at large [35]. Furthermore, complicated systems foster user errors, decreasing system security [27, 36]. While some systems need to expose users to complex security choices, basic functionality (e.g., sending or receiving an encrypted email) should be approachable for new users. At a minimum this includes building intuitive interfaces, providing integrated, context-sensitive tutorials, and helping first-time recipients of encrypted messages understand what they need to do in order to decrypt their message.

MessageGuard should integrate with existing web applications. Users enjoy the web services and applications they are currently using and are disinclined to adopt a new system solely because it offers greater security. Instead, users prefer that content-based encryption be integrated into their existing applications [1, 27]. Equally important, content-based encryption should have a minimal effect on the application's user experience; if encryption gets in the way of users completing tasks it is more likely that they will turn off content-based encryption [15].

MessageGuard's interfaces should be usable at any size. Current web interfaces allowing users to consume or create content come in a wide variety of sizes (i.e., height and width). When MessageGuard integrates with these web services, it is important that MessageGuard's interfaces work at these same dimensions. To support the widest range of sizes, MessageGuard's interfaces should react to the space available, providing as much functionality as is possible at that display size.

A.3.3 Ubiquitous

MessageGuard should support most websites and platforms.

MessageGuard should work with most websites MessageGuard should make it easy for researchers to explore adding end-to-end encryption into whichever web applications they are interested in. While it may be impossible to fully support all web applications (e.g., Flash applications or applications drawn using an HTML canvas), most standard web applications should work out-of-the-box. For those applications which don't work out-of-the-box, MessageGuard should allow researchers to create customized prototypes that handle these edge cases.

MessageGuard should function in all major desktop and mobile browsers. Prototypes built with MessageGuard should function both on desktop and mobile browsers, allowing researchers to experiment with both of these form factors. Furthermore, MessageGuard should work on all major browsers, allowing users to work with the web browser they are most familiar with, obviating the need to restrict study recruitment to users of a specific browser.

A.3.4 Extensible

MessageGuard should be easily extensible and contribute to the rapid development of content-based encryption prototypes.

MessageGuard should be modular. MessageGuard's functionality should be split into a variety of modules, with each module taking care of a specific function. Researchers should also be free to only change the modules that relate to their research and have the system continue to function as expected. Similarly, MessageGuard's modules should be extensible, allowing researchers to create new custom modules with a minimal amount of effort.

MessageGuard should provide reference functionality. As a base for other researchers' work, MessageGuard should include a reference implementation of the various modules that adds content-based encryption to a wide range of web applications. This reference implementation should be able to be easily modified and extended to allow researchers to rapidly implement their own ideas.

A.3.5 Reliable

The usability and security of MessageGuard should be reliable, protecting researchers from unintentionally compromising MessageGuard's security or usability.

Reducing the security of MessageGuard should require deliberate intent. HCI researchers should feel comfortable customizing MessageGuard's interface without needing to worry that they are compromising security. To facilitate this, MessageGuard should separate UI and security functionality into separate components. As long as researchers limit themselves to changing only UI components, there should be no effect on security.

Modifying the cryptographic primitives should have minimal effect on MessageGuard's usability. As above, MessageGuard should separate its UI and security functionality into separate

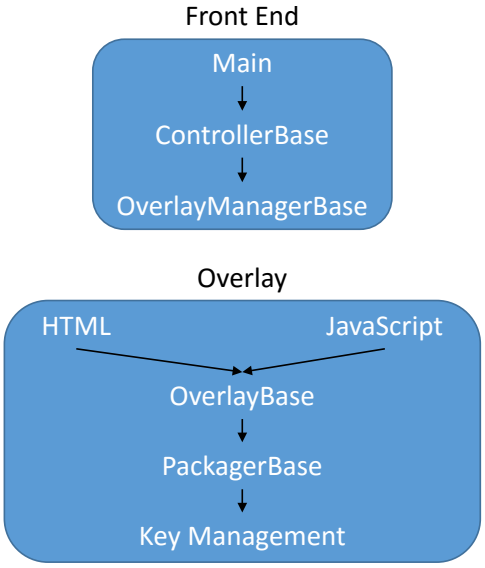


Figure 8: MessageGuard's customizable framework.

components. This will allow security researchers to modify the cryptographic primitives without worrying about how they will affect MessageGuard's usability. One caveat is if a new key management scheme requires a user interface that MessageGuard does not already make available. In this case, researchers will need to provide this key management scheme's interface, which could affect usability, but other interfaces should remain unaffected.

B. MESSAGEGUARD AS A RESEARCH PLATFORM

In this section, we describe the ways researchers can employ MessageGuard as a platform for their own research. In addition to the details described in this section, we invite researchers to download MessageGuard's source code. To help researchers quickly familiarize themselves with MessageGuard's code base, we have included instructive comments throughout the code and have provided a reference implementation that supports most websites that researchers can refer to as they build their own systems.

MessageGuard was designed to minimize the amount of code that must be changed in order for researchers to build new prototypes. The customizable classes enabling this rapid prototyping are shown in Figure 8. MessageGuard includes a default instantiation for each of the base classes (e.g. `ControllerBase`) seen in the figure. To change the global functionality of MessageGuard, researchers need to change the aforementioned default implementations. If researchers desire to implement new functionality (e.g., create a new overlay, support a new application), they can instead subclass these base classes. All classes, both base classes and default implementations, can be extended, but only allow researchers to override the methods that are unique to their functionality.

B.1 Frontend

The main class is responsible for parsing the URL and instantiating the appropriate controller (i.e., classes extending `ControllerBase`). Frontend controllers are responsible for the actual operations of the frontend, including detecting when overlays are needed and placing those overlays. Every overlay is created by and coupled to an overlay manager, which is responsible for handling communication between the overlay and MessageGuard's frontend. Currently, MessageGuard provides overlay managers for both reading and composing encrypted content.

The simplest way to modify the frontend is to change the elements that it will overlay. This can be done by changing the CSS selector that is passed to `ControllerBase`'s constructor.[18] The controller can also be configured to support additional types of overlays (i.e., creating a unified read and compose overlay for instant messaging clients). In this case, it will also be necessary to create an overlay manager to communicate with the new overlay.

Using these base classes, MessageGuard's default functionality was implemented using less than 200 lines of JavaScript.

B.2 Overlays

Overlays are composed of both HTML interfaces and JavaScript code. Researchers can either modify the existing overlays (read and compose) or create their own overlays. The steps for creating a new overlay modifying overlays on a per-application basis are as follows:

1. Create a new HTML file for each overlay. This will define the visual appearance of the overlay.
2. Create a custom read, compose, or entirely new overlay (e.g., file upload) by extending either the `OverlayBase` class or one the reference overlays (read and compose). These parent classes provide basic functionality (e.g., positioning, communication with the frontend).
3. Connect the overlay's HTML interface to its controlling code by referencing this new JavaScript class in the new HTML.
4. Create a new overlay manager to work with the new overlay. You can extend any of the existing overlay managers, or create a new one by extending `OverlayManagerBase`.
5. Add any custom communication code to both the new overlay and overlay manager.

MessageGuard's default read overlay required 70 lines of HTML and 150 lines of JavaScript to implement. The default compose overlay needed 190 lines of HTML and 670 lines of JavaScript, most of which was responsible for setting up the HTML5 rich-text interface and allowing users to select a specific key for encryption.

B.3 Packager

By overriding PackagerBase, it is possible to create custom message packages, allowing MessageGuard to support a wide range of content-based encryption protocols. This functionality can be used to allow prototypes developed with MessageGuard to inter-operate with existing cryptographic systems (e.g., using the PGP package syntax in order to be compatible with existing PGP clients). It could also be used

[18] Though unlikely to be necessary, it is also possible to modify the controller to do more complex selection that does not rely on CSS selection.

to experiment with advanced cryptographic features, such as key ratcheting [35].

B.4 Key Management

One key goal of MessageGuard is to allow existing proposals for key management to be implemented in a real system, and then compared against alternative schemes. As such, we took special care to ensure that MessageGuard would be compatible with all key management schemes we are currently aware of. In order to create a new key management scheme, the following two classes must be implemented:

KeyScheme. The KeyScheme is responsible for handling scheme-specific UI functionality for the key manager (e.g., importing public/private keys, authenticating to a key server). The KeyScheme methods are:

- **getUI** Retrieves a scheme-specific UI that will be included with the KeyUIManager's generic UI. This method is provided with the KeySystem being created/updated and a callback which notifies the KeyUIManager that the KeySystem is ready to be saved.
- **handleError** Modifies an existing KeySystem's UI to allow it to address an error. This method is provided with details about the error, the KeySystem UI to modify, and a callback which notifies the KeyUIManager that the error has been resolved. Examples of errors include not having a necessary key or expired authentication credentials.
- **create** Creates a KeySystem from the scheme-specific UI provided to this method.
- **update** Updates a KeySystem from the scheme-specific UI provided to this method.

KeySystem. A KeySystem is an instantiation of a key management scheme that allows the users to decrypt/sign data for a single identity and encrypt/verify data for any number of identities.[19] A KeySystem is responsible for performing cryptographic operations with the keys it manages. Every KeySystem has a fingerprint that uniquely identifies it. The KeySystem methods are:

- **serialize/deserialize** Prepares data that is not a part of the KeyAttributes type for storage by the KeyStorage class.
- **encrypt** Encrypts data for the provided identity. Returns the encrypted data along with the fingerprint of the KeySystem that can decrypt it.
- **decrypt** Decrypts the provided data.
- **sign** Signs the provided data.
- **verify** Verifies that the provided signature is valid for the provided data.

By default, MessageGuard will allow users to use all available key management schemes, though this can be overridden on a per-prototype basis.

[19] Key systems which don't support recipients set `canHaveRecipients` to `false` and ignore the identity parameters.

Stage	Static			Dynamic		
n	100	500	1000	100	500	1000
Chrome[1]	1.14	0.84	0.95	3.17	6.49	11.0
Firefox[1]	1.06	0.99	0.96	2.26	3.15	4.45
Safari[1]	0.45	0.63	0.53	3.73	12.8	25.5
Chrome[2]	4.27	4.39	4.60	12.9	30.2	51.1
Chrome[3]	5.68	5.97	5.94	12.4	32.0	61.2
Safari[3]	2.57	2.46	1.79	15.1	25.2	39.5

[1] MacBook Air (OSX 10.10.3, 1.7GHz Core i7, 8GB RAM). Chrome—42.0.2311.135, Firefox—37.0.2, Safari—8.0.5.
[2] OnePlus One (CyanogenMod 12S, AOSP 5.1, 64GB). Chrome—42.0.2311.47.
[3] iPad Air (iOS 8.3, 1st gen, 64GB). Chrome—42.0.2311.47, Safari—8.0.

Table 5: Average time to overlay an element (ms)

C. VALIDATION OF MESSAGEGUARD

We evaluated MessageGuard ability to support usable, content-based encryption research on a wide range of platforms. Additionally, we measured the performance overhead that MessageGuard creates. Our results indicate that MessageGuard is compatible with most web applications and has minimal performance overhead.

C.1 Ubiquity

We tested MessageGuard on major browsers and it worked in all cases: Desktop—Chrome, Firefox, Internet Explorer, Opera, and Safari. Android—Chrome, Firefox, Opera. iOS—Chrome, Mercury, Safari.

We tested MessageGuard on the Alexa top 50 web sites. One of the sites is not a web application (`t.co`) and another requires a Chinese phone number in order to use it (`weibo.com`). MessageGuard was able to encrypt data in 47 of the 48 remaining web applications. The one site that failed (youtube.com) did so because the application removed the comments field when it lost focus, which happens when focus switched to MessageGuard's compose overlay. We were able to address this problem with a customized frontend that required only five lines of code to implement.

These results indicate that researchers should be able to use MessageGuard to research content-based encryption for the web applications of their choice with little difficulty.

C.2 Performance

We profiled MessageGuard on several popular web applications and analyzed MessageGuard's impact on load times. In each case, we started the profiler, reloaded the page, and stopped profiling once the page was loaded. Our results show that MessageGuard has little impact on page load times and does not degrade the user's experience as they surf the Web: Facebook—0.93%, Gmail—2.92%, Disqus—0.54%, Twitter—1.98%.

Since MessageGuard is intended to work with all websites, we created a synthetic web app that allowed us to test MessageGuard's performance in extreme situations. This app measures MessageGuard's performance when overlaying static content present at page load (Stage 1) and when overlaying dynamic content that is added to the page after load (Stage

2). The application takes as input n, the number elements that will be overlayed in each stage. Half of these elements will require read overlays and half will require compose overlays.

Using this synthetic web application, we tested MessageGuard with six browsers and three values of n. We averaged measurements over ten runs and report our findings in Table 5. Performance for overlaying static content does not significantly vary based on the number of overlays created. In contrast, performance for overlaying dynamic content for most browsers seems to grow polynomial in the number of overlays added. Still, performance in the Firefox desktop browser demonstrates that this is not an inherent limitation of MessageGuard. Finally, we note that even in extreme cases (dynamic—$n = 1000$) overlaying occurs quickly (max 61 ms).

MessageGuard's low performance overhead indicates it is suitable for building responsive prototypes for testing by users. Moreover, if performance problems arise, researchers can be reasonably sure that the problems are in their changes to MessageGuard.

D. USER STUDY MATERIALS

This section of the appendix contains instructions and surveys from the user study that will allow others to replicate this research. The following items are included: A) instructions to the study coordinators that supervise Johnny and Jane; B) demographic questions; C) initial instructions to Johnny and Jane describing the user study scenario; D) instructions to Johnny and Jane regarding the tasks they must complete for each MessageGuard variant; E) survey questions Johnny and Jane answer after using each MessageGuard variant; F) post-study questions; G) and descriptions of the security of each key management scheme.

D.1 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.
   (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 {Screen recording software}. Start recording.
   (c) {Open the survey}
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.
   (d) Change the view to full screen-exclusive mode.
   (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.
   (c) Restore the snapshot.
   (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.
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.

D.2 Demographic Questions

In our study, Johnny was shown these questions at the end of the survey, while Jane was shown them at the beginning of the survey. This was done to let Johnny get started working on the first task right away and to give Jane something to do while waiting for the first email.

What is your gender?

- *Male*
- *Female*
- *I prefer not to answer*

What is your age?

- *18–24 years old*
- *25–34 years old*
- *35–44 years old*
- *45–54 years old*
- *55 years or older*
- *I prefer not to answer*

What is the highest degree or level of school you have completed?

- *Some school, no high school diploma*
- *High school graduate, diploma or the equivalent (for example: GED)*
- *Some college or university credit, no degree*
- *College or university degree*
- *Post-Secondary Education*
- *I prefer not to answer*

What is your occupation or major?

How would you rate your level of computer expertise?

- *Beginner*
- *Intermediate*
- *Advanced*

D.3 Scenario Instructions

D.3.1 Johnny Scenario

In this study, you will be role playing the following scenario:

Your friend graduated in accounting and you have asked their help in preparing your taxes. They told you that they needed you to email them your last year's tax PIN and your social security number. Since this information is sensitive, you want to protect (encrypt) this information when you send it over email.

You will be asked to send this information using three different secure email systems. In each task, you'll be told which system to use and assigned a new PIN and SSN. After correctly sending the information, your friend will reply to you with a confirmation code that can be used to continue with the study.

D.3.2 Jane Scenario

In this study, you will be role playing the following scenario:

You graduated in accounting and have agreed to help a friend prepare their taxes. You have asked them to email you their last year's tax PIN and their social security number.

As part of the study, your friend will send you this information three different times. Each time, after receiving their PIN and SSN, you will be provided with a confirmation code and a PIN number to send to your friend so that both of you can continue with the study.

D.4 Task Instructions

D.4.1 Johnny's Task

Johnny repeats the following for each MessageGuard variant.

Tell the study coordinator that you are ready to begin this task.

System: **MessageGuard—{Insert encryption scheme}**

In this task, you'll be using **MessageGuard—{Insert encryption scheme}**. The system can be found at the following website: {Insert url}

Please encrypt and send the following information to your friend using MessageGuard—{Insert encryption scheme}:

SSN: {Task SSN}
PIN: {Task PIN}

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.

D.4.2 Jane Task

Jane repeats the following for each MessageGuard variant.

Tell the study coordinator that you are ready to begin this task.

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 friend's SSN and PIN, let the study coordinator know that you are ready to reply to your friend with their confirmation code and PIN.

You have completed your friend's taxes and need to send them the confirmation code and this year's tax PIN from their tax submission.

Since your friend used MessageGuard—{System name} to send sensitive information to you, please also use MessageGuard—{System name} to send them the confirmation code and PIN.

- Confirmation code: {Task SSN}
- PIN: {Task PIN}

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.

D.5 Survey

Johnny and Jane complete the following survey after each MessageGuard variant.

You will now be asked several questions concerning your experience with **MessageGuard—{Insert encryption scheme}**.

Please answer the following questions about {Insert encryption scheme}. Try to give your immediate reaction to each statement without pausing to think for a long time. Mark the middle column if you don't have a response to a particular statement.

<SUS Questions>

What did you like most about using MessageGuard—{Insert encryption scheme}?

What would you change about MessageGuard—{Insert encryption scheme}?

Please explain why.

D.6 Post-study questions

You have finished all the tasks for this study. Please answer the following questions about your experience.

Which system was your favorite? ***(Ask the coordinator if you are unclear which system is which.)***

- *First system: MessageGuard—{First system name}*
- *Second system: MessageGuard—{Second system name}*
- *Third system: MessageGuard—{Third system name}*
- *I don't like any of the systems I used*

Please explain why.

Please answer the following questions. Try to give your immediate reaction to each statement without pausing to think for a long time. Mark the middle column if you don't have a response to a particular statement.

I want to be able to encrypt my email.
<Likert scale>

I would encrypt email frequently.
<Likert scale>

In the password-based version of MessageGuard, the passwords you entered would be deleted when you exited Chrome. This meant that others using your computer would not be able to read your encrypted email.

In contrast, the PKD and IBE versions save your encryption keys, and anyone logged into Gmail on your computer can read your encrypted email. This could be changed by adding a **master password** to MessageGuard. You would select your master password when you install MessageGuard.

From then on, whenever you open your browser, MessageGuard would require you to enter your master password before functioning. This would protect your IBE- and PKD-encrypted emails from others who use your computer.

Would you prefer MessageGuard to use a master password?

- *Yes*
- *No*

Please explain why.

D.7 Key Management Descriptions

PKD: "In the {first, second, third} system you tested, your email was secured using PKD. In PKD, 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 PKD, 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."

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."

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."