2016-07-01

Usable, Secure Content-Based Encryption on the Web

Scott Ruoti
Brigham Young University

Usable, Secure Content-Based Encryption on the Web

Scott Isaac Ruoti

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

Doctor of Philosophy

Kent Seamons, Chair
Daniel Zappala
Mike Goodrich
David Wingate
Bryan Morse

Department of Computer Science

Brigham Young University

July 2016

ABSTRACT

Usable, Secure Content-Based Encryption on the Web

Scott Isaac Ruoti
Department of Computer Science, BYU
Doctor of Philosophy

Users share private information on the web through a variety of applications, such as email, instant messaging, social media, and document sharing. Unfortunately, recent revelations have shown that not only is users' data at risk from hackers and malicious insiders, but also from government surveillance. This state of affairs motivates the need for users to be able to encrypt their online data.

In this dissertation, we explore how to help users encrypt their online data, with a special focus on securing email. First, we explore the design principles that are necessary to create usable, secure email. As part of this exploration, we conduct eight usability studies of eleven different secure email tools including a total of 347 participants. Second, we develop a novel, paired-participant methodology that allows us to test whether a given secure email system can be adopted in a grassroots fashion. Third, we apply our discovered design principles to PGP-based secure email, and demonstrate that these principles are sufficient to create the first PGP-based system that is usable by novices.

We have also begun applying the lessons learned from our secure email research more generally to content-based encryption on the web. As part of this effort, we develop MessageGuard, a platform for accelerating research into usable, content-based encryption. Using MessageGuard, we build and evaluate Private Facebook Chat (PFC), a secure instant messaging system that integrates with Facebook Chat. Results from our usability analysis of PFC provided initial evidence that our design principles are also important components to usable, content-based encryption on the Web.

Keywords: Security, HCI, Usable security, Content-based encryption, Secure email, Webmail, End-to-end encryption, user study

ACKNOWLEDGMENTS

Thanks first and foremost goes to my advisor Kent Seamons. His mentorship has been crucial in helping me develop the skills necessary to be a successful research scientist. I also want to thank Daniel Zappala for helping me significantly refine my writing and presentation skills. Additionally I am grateful for the mentoring provided by various researchers during my graduate education: Mike Goodrich, Charles Knutson, Jay McCarthy, Rich Shay, and Shabsi Walfish.

Thanks also goes to the students who have helped me conduct studies, gather data, and analyze results for this dissertation: Jeff Andersen, Ben Burgeon, Scott Heidbrink, Travis Hendershot, Nathan Kim, Tyler Monson, Mark ONeill, Chris Robison, Elham Vaziripour, and Justin Wu.

Finally, a special thanks goes to my wife, Emily Ruoti. She has always been a willing participant in my pilot studies and has provided copious amount of feedback on the systems I have built as well as the papers I have written. More importantly, she has helped raise our daughter Esther, picking up the slack when my dissertation required nearly all of my free time. Thank you Emily.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Users share private information on the web through a variety of applications, such as email, instant messaging, social media, and document sharing. TLS protects this information during transmission, but does not protect users' data while at rest. Additionally, middleboxes can weaken TLS connections by failing to properly implement TLS or adequately validate certificate chains [19]. Even if a website correctly employs TLS and encrypts user data while at rest, the user's data is still vulnerable to honest-but-curious data mining [24], third-party library misbehavior [34], website hacking, protocol attacks [10, 12, 18], and government subpoena.

This state of affairs motivates the need for *content-based* encryption of user data. In content-based encryption, users' sensitive data is encrypted at their own device and only decrypted once it reaches the intended recipient, remaining opaque to the websites that store or transmit this encrypted data.[1] The best known examples of content-based encryption are secure email (e.g., PGP, S/MIME) and secure instant messaging (e.g., OTR). In addition to protecting communication, content-based encryption can protect any data users store or distribute online; for example, files stored in the cloud (e.g., DropBox, Google Drive) or private postings to web-based message boards.

In this dissertation, most of our research focuses on a specific application of content-based encryption—secure email. We chose this focus for several reasons:

[1] In contrast, connection-level encryption (i.e., TLS) only protects data during transmission, and not while it is stored or handled by websites.

1. **Email is the dominant form of communication on the Internet.** There are 4.3 billion email accounts, owned by 2.5 billion users. Collectively, these users send over 205 billion email messages per day.[2] This dwarfs other platforms, including instant messaging clients such as WhatsApp (42 billion messages per day).[3]

2. **Sensitive information is sent over email.** While not necessarily an every-day occurrence, from time to time individuals need to use email to send highly sensitive information. For example, some businesses will ask that a job applicant send their social security number to the business in order to process a reimbursement.[4] Furthermore, unlike many other communication mediums, email is usually archived by the email server, increasing the vulnerability of sensitive information sent over email.[5]

3. **Advances in secure email are likely to be applicable to other content-based encryption systems.** While there are some features that are unique to secure email, many of the principles for designing securing email in a usable manner also apply to other content-based encryption systems. For example, in this dissertation we used principles learned from our secure email work to create a secure Facebook Chat system that was well received by users.

4. **Email is unlikely to disappear in the near future.** While new communication platforms continue to replace other applications, email has seen steady growth. This indicates that advances in secure email will have far-reaching and long-lasting benefits.

5. **Secure email obviates the need for secure storage depots.** Because email transmission is not secure, many organizations resort to the use of separate secure messaging websites for sensitive communication. When a user receives a message on these systems, they also receive an email that tells them to log into the secure messaging

[2] `http://www.radicati.com/wp/wp-content/uploads/2015/02/Email-Statistics-Report-2015-2019-Executive-Summary.pdf`

[3] `http://www.iphoneincanada.ca/news/whatsapp-1-billion/`

[4] I have personally experienced this as I was interviewing for jobs; multiple institutions requested that I send them my social security number to process my reimbursements.

[5] This also demonstrates why connection-level security (i.e., TLS) is insufficient, as it does nothing to protect data while at rest.

website to read their sensitive message. This is an annoying and cumbersome process that is largely disliked by users [24, 26]. Secure email would remove the need for these separate web applications, allowing sensitive information to be sent directly through email.

While the cryptographic primitives that enabled secure email have long existed [13], prior to the work in this dissertation it was an open problem whether secure email could be implemented in a usable fashion. In this dissertation, we explore what design principles are necessary to allow secure email to be sufficiently usable for the masses.

1.1 Email Security

When email was first designed in 1971[6] no meaningful attention was paid to security. As such, it is unsurprising that it was trivial for attackers to steal email and also to inject false messages (see Figure 1.1 for details of these vulnerabilities).[7] Since then, there have been attempts to patch security on top of email (e.g., STARTTLS, DKIM, DMARK, SPF), but even with these advances it is still possible for an attacker to steal and inject email messages (see Figure 1.2).

As such, email is an easy target for attackers. For example, Durumeric et al. found that in seven countries over 20% of inbound Gmail messages are being stolen [10]. Also, email can also be stolen while it is stored at the user's email server.[8] Additionally, the inability to authenticate the sender of an email increases the likelihood of email phishing, a multi-billion-dollar problem.[9]

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

[7] Injection of false messages refers to an attacker creating a message that claims to be from Alice, and then getting that message sent to Bob. This type of attack is highly useful as part of a spear-phishing attack.

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

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

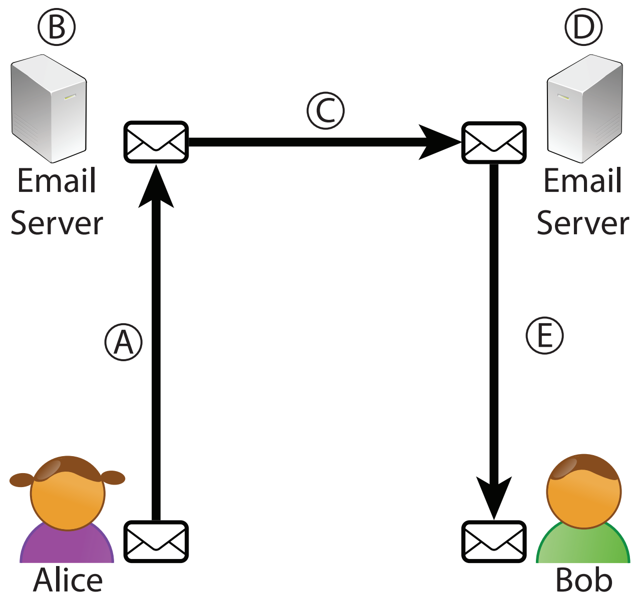
A better approach for protecting email is content-based encryption. In content-based encryption, email is secured (i.e., signed and encrypted) before it ever leaves the sender's machine. This means that regardless of the trustworthiness of the network over which the email message is sent, it is still safe from attackers (see Figure 1.3). We note that in much of the literature, content-based encryption is usually called end-to-end encryption. We use both terms interchangeably in this dissertation.

1.2 Unusable, Secure Email

One of the earliest attempts at providing content-based encryption for email was Pretty Good Privacy [13], better known by its acronym PGP. PGP was developed in 1991 by Phil Zimmerman, and allows users to encrypt and sign their email messages using public key cryptography. In PGP, keys are generally validated using the web of trust—i.e., user's verify and sign their associates' public keys; a user can check if they trust a key by seeing if they or one of their associates has signed that key. Public keys can be shared 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.

Eight years after the first release of PGP, Whitten and Tygar studied the usability of PGP 5.0 [38]. In their study individuals were brought into a lab, asked to pretend that they were part of a political campaign that required emails to be encrypted with PGP. In their study, 9 of the 12 users (75%) mistakenly sent messages unencrypted, and several even included their private key in their emails. Moreover, at the conclusion of the study, Whitten and Tygar found that participants were unclear as to how PGP protected their email.

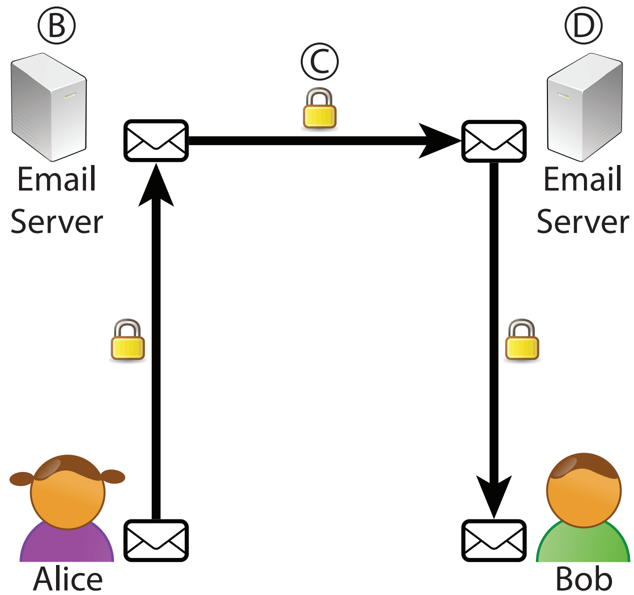
Sheng et al. [33] repeated the Whitten and Tygar user study with PGP 9.0. They noted that some egregious usability issues had been addressed since PGP 5.0, but also found that users were still confused regarding key management and digital signatures. As part of the work in this dissertation, we evaluated a modern PGP-based, secure email client—Mailvelope. In our study only one in ten participants was able to use Mailvelope to send encrypted email,



An attacker is able to steal email in five different locations: during transmission between the user and their email server (A, E), during transmission between email servers (C), while at rest on either user's email server (B, D).

An attacker is also able to inject false messages at any of the five locations (A, B, C, D, E).

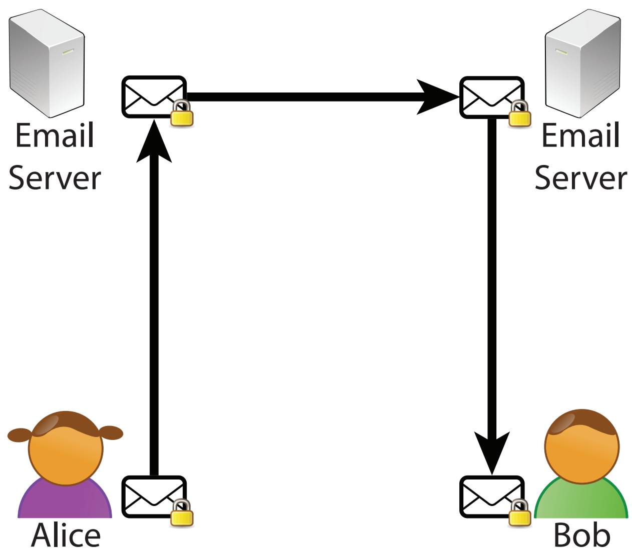
Figure 1.1: Basic Email Security



With TLS, an attacker is unable to steal messages during transmission between the user and their email server. Theoretically, an attacker should not be able to steal messages during transmission between email servers (C), but in practice this location is still vulnerable to attackers [10, 12, 18]. As TLS only protects data during transmission, an attacker can still steal a user's email while at rest at the user's email server (B, D).

Additionally, TLS does not protect against an attacker injecting false messages at any of the above locations (B, C, D). DKIM can partially protect against message injection during transmission of messages between email servers (C), but cannot prevent message injection at the user's email server (B, D).

Figure 1.2: Email Security with TLS



In end-to-end encryption, messages are encrypted at Alice's computer and are decrypted on Bob's computer. Even if an attacker were able to steal a user's email during transmission or storage at an email server, the attacker would be unable to obtain the plaintext content of the message.

In addition to encrypting the email she authors, Alice will also sign it. This allows Bob to verify that the email he received really was authored by Alice. This feature of end-to-end encryption prevents an attacker from injecting false messages.

Figure 1.3: Email Security with End-to-End Encryption

demonstrating that over a decade and half since Whitten and Tygar's original study, PGP remains unusable.

Another approach for content-based encryption of email is Secure/Multipurpose Internet Mail Extensions (S/MIME). Similar to PGP, S/MIME uses public key cryptography to encrypt and sign email. Unlike PGP's web of trust, S/MIME certificates are authenticated using the certificate authority (CA) system.[10,11]

Garfinkel and Miller studied the usability of S/MIME with automatic key management [15]. In their study, they repeated and expanded upon the original Johnny experiment. Their results showed that automating key management significantly increased the usability of secure email. Still, even with this improvement a large number of the participants in the study were unable to correctly send secure email.

Prior to the work in this dissertation, there was no evidence that secure email could be made sufficiently usable for adoption by the masses. This has led some to question whether secure email for the masses is a feasible goal.[12,13]

1.3 User Studies

A key component of this dissertation is the use of user studies to evaluate secure email systems, both those we created and systems from industry. In these studies, users will be brought into the lab or other spaces specifically prepared for the study and will complete several tasks using one or more secure email tools. After completing the assigned tasks with a given system, users then complete a survey regarding their experience with that system. Additionally, participants' screens are recorded and this recording is used to calculate task completion times and note mistakes made while using the system. Finally, we interview participants regarding their experience to gather additional qualitative data.

[10] The CA system is the same mechanism by which websites are authenticated in TLS.

[11] More recent versions of PGP also support certificates validated using the CA system, but in our experience this is not widely used feature for secure email.

[12] https://www.schneier.com/blog/archives/2015/11/testing_the_usa.html

[13] https://moxie.org/blog/gpg-and-me/

1)	I think that I would like to use this system frequently.
2)	I found the system unnecessarily complex.
3)	I thought the system was easy to use.
4)	I think that I would need the support of a technical person to be able to use this system.
5)	I found the various functions in this system were well integrated.
6)	I thought there was too much inconsistency in this system.
7)	I would imagine that most people would learn to use this system very quickly.
8)	I found the system very cumbersome to use.
9)	I felt very confident using the system.
10)	I needed to learn a lot of things before I could get going with this system.

Table 1.1: The Ten SUS Questions

	Questions 1,3,5,7,9	Questions 2,4,6,8,10
Strongly Agree	10	0
Agree	7.5	2.5
Neither Agree or Disagree	5	5
Disagree	2.5	7.5
Strongly Disagree	0	10

Table 1.2: SUS Score Card

1.3.1 System Usability Scale

A system's usability is measured using a standard usability metric—the System Usability Scale (SUS) [7, 8]. SUS has been used in hundreds of usability studies [5] and the original SUS paper [7] has been cited over 3,535 times.[14] In work not included in this dissertation, we have shown that a system's SUS score is consistent across different sets of users [25]. Moreover, Tullis and Stetson compared SUS to four other usability metrics (three standard metrics from the usability literature and their own proprietary measure) and determined that SUS gives the most reliable results [36].

The SUS metric is a single numeric score from 0, the least usable, to 100, the most usable, that provides a rough estimate of a system's overall usability. To calculate a system's SUS score, participants first interact with the system and then answer ten questions relating to their experience (see Table 1.1). Answers are given using a five-point Likert scale (*strongly agree* to *strongly disagree*). The questions alternate between positive and negative statements

[14]Citation count retrieved from Google Scholar on June 9, 2016.

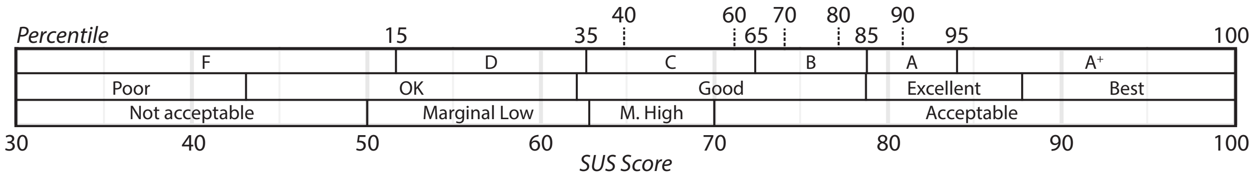


Figure 1.4: Adjective-based Ratings to Help Interpret SUS Scores

about the system being tested. Participants' answers are assigned a scalar value (see Table 1.2) and then summed to produce the overall SUS score, and the system with the highest average SUS score is the most usable.

SUS produces a numeric score for a non-numeric measure (i.e., usability), making it difficult to intuitively understand how usable a system is based solely on its SUS score. To give greater context to the meaning of a system's SUS score, we leverage the work of several researchers. Bangor et al. [5] analyzed 2,324 SUS surveys, and derived a set of acceptability ranges that describe whether a system with a given score is acceptable to users in terms of usability. Bangor et al. also associated specific SUS scores with adjective descriptions of the system's usability. Using this data, we generated ranges for these adjective ratings, such that a score is correlated with the adjective it is closest to in terms of standard deviations. Sauro et al. [31] also analyzed SUS scores from Bangor et al. [4], Tullis et al. [36], and their own data. They calculate the percentile values for SUS scores and assign letter grades based on percentile ranges. The above contextual clues are presented in Figure 1.4.

1.3.2 Mistakes

In this dissertation, we define a mistake as a user accidentally sending sensitive information in the clear when they intended to encrypt it. As part of our studies, users will be instructed to send several encrypted messages. Using the screen recordings, we will record how often users accidentally send these messages without encryption. After the study, we will interview users about these mistakes to try and determine what caused them and how the system could be designed to avoid these mistakes in the future.

1.4 Dissertation Overview

Chapter 2—Scott Ruoti, Nathan Kim, Ben Burgon, Timothy Van Der Horst, and Kent Seamons. *Confused Johnny: When Automatic Encryption Leads to Confusion and Mistakes.* Ninth Symposium on Usable Privacy and Security (SOUPS 2013), Newcastle, United Kingdom, 2013. ACM.

Chapter 3—Scott Ruoti, Jeff Andersen, Travis Hendershot, Daniel Zappala, and Kent Seamons. *Private Webmail 2.0: Simple and Easy-to-Use Secure Email.* Twenty-Ninth ACM User Interface Software and Technology Symposium (UIST 2016), Tokyo, Japan, 2016. ACM.

The first question we tackled in this dissertation was whether it is possible to create secure email that is usable by the majority of novice users. To this end, we explored different design principles that could increase the usability of secure email, and implemented these design principles in various prototypes. To maximize usability, our prototypes used identity-based encryption (IBE [32]) to encrypt and sign messages.[15] We then tested these prototypes and other systems across a range of user studies, analyzing the results of these studies to determine which design principles were most important for usable secure email.

The results of our studies demonstrated that the prototypes we built successfully allowed novice users to send and receive encrypted messages. Moreover, our work provided the first evidence that it was possible to create usable, secure email for the masses, as all previous results had been negative. Still, our early studies of secure email only demonstrated that users could use secure email when paired with an expert user (i.e., a study coordinator).[16]

[15] IBE is a form of public key cryptography. In IBE, a user's public key is their email address, meaning that all users of email already have a public key that can be used to encrypt and send them email. Private keys are generated by a trusted third-party key escrow server. To obtain their private key, a user will prove ownership of their email account, after which the key escrow server will provide the user with their private key.

[16] This limitation is present in all secure email research [2, 15, 33, 38].

Chapter 4—Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent Seamons. *A Usability Study of Four Secure Email Tools Using Paired Participants*, 2016. Under Submission.

Chapter 4 is an extended version of a paper published in CHI. Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent Seamons. *"We're on the Same Page": A Usability Study of Secure Email Using Pairs of Novice Users*. Thirty-Fourth Annual ACM Conference on Human Factors in Computing Systems (CHI 2016), pages 4298–4308, San Jose, CA, 2016. ACM.

To address this limitation, we developed a novel methodology where we tested the ability of pairs of novice users to begin using secure email between themselves. The results of our study using this novel methodology demonstrated that our secure email prototype could be adopted by users in a grassroots fashion. Furthermore, qualitative feedback from participants and observations by study coordinators demonstrated that our new methodology elicited more natural participant behavior and more fully explored the usability of the systems tested, as compared to prior study methodologies for secure email.

Chapter 5—Scott Ruoti, Jeff Andersen, Tyler Monson, Daniel Zappala, and Kent Seamons. *MessageGuard: A Browser-Based Platform for Usable End-to-End Encryption Research*. Under Submission, 2016.

In this dissertation, we also applied the lessons learned from our secure email research more generally to content-based encryption on the web. As part of this effort, we modified one of our early secure email prototypes, allowing it to support generic content-based encryption across the Web. Additionally, to assist other researchers in exploring this space we architected this new system so that it could be used as a platform for research into usable, secure content-based encryption on the Web. We named this platform MessageGuard.

Chapter 6—Chris Robison, Scott Ruoti, Timothy W. van der Horst, and Kent E. Seamons. *Private Facebook Chat*. Fifth ASE/IEEE International Conference on Social Computing (SocialCom 2012) and Fourth ASE/IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT 2012), pages 451–460, Amsterdam, The Netherlands, 2012. IEEE Computer Society.

MessageGuard was used to build most of the secure email prototypes we developed during our research. To verify our belief that MessageGuard could be used to create usable, secure prototypes for other applications (i.e., not email) of content-based encryption, we used MessageGuard to develop Private Facebook Chat (PFC), a system that adds content-based encryption to Facebook Chat. Results from a user study of PFC demonstrated that many of the design principles we were using to build usable, secure email could also be applied to instant messaging.[17] Moreover, our experiences building prototypes using MessageGuard demonstrated that MessageGuard as a platform significantly reduced the cost and difficulty of prototyping and evaluating content-based encryption solutions.

Chapter 7—Jeff Andersen, Tyler Monson, Scott Ruoti, Kent Seamons, and Daniel Zappala. *A Comparative Usability Study of Key Management in Secure Email*. Under Submission, 2016.

To conclude the work in this dissertation, we applied everything we had built and learned up to that point to try and create usable, secure email based on PGP.[18] As part of the evaluation of our PGP prototype, we used MessageGuard to build three secure email

[17] Based on quantitative results and participant feedback from our study, we feel confident that MessageGuard will also be beneficial to other types of content-based encryption (e.g., secure file storage).

[18] PGP-based secure email that can be used by the masses is the holy grail of this area.

variants, each of which used a different key management approach: PGP, IBE, and passwords. Using our paired-participant methodology, we then compared these systems to each other to determine their relative usability.[19]

Our results indicate that users find all three systems to be usable. Participants were evenly split regarding which system was their favorite (PGP–30%, IBE–37%, Passwords–30%), as participants had varying secure email preferences—for example, some participants liked that with the Passwords system they had full control of their security, whereas other users did not want to have to manage passwords for each of their contacts. Still, our results demonstrated that PGP can be made usable for novices, answering long-standing questions regarding its viability. Furthermore, by demonstrating that PGP can be made usable, we have opened the doors to future work within the field of usable, secure email.

[19] The three prototypes incorporated all the design principles we had identified as being necessary to make secure email usable, and only differed as required by each key management approach, eliminating potential confounding factors in our analysis.

Confused Johnny: When Automatic Encryption Leads to Confusion and Mistakes

Scott Ruoti, Nathan Kim, Ben Burgon, Timothy van der Horst, Kent Seamons
Internet Security Research Lab
Computer Science Department
Brigham Young University
Provo, Utah, USA
{srouti, nkim, bburgon, timv}@isrl.cs.byu.edu, seamons@cs.byu.edu

ABSTRACT

A common approach to designing usable security is to hide as many security details as possible from the user to reduce the amount of information and actions a user must encounter. This paper gives an overview of Pwm (Private Webmail), our secure webmail system that uses security overlays to integrate tightly with existing webmail services like Gmail. Pwm's security is mostly transparent, including automatic key management and automatic encryption. We describe a series of Pwm user studies indicating that while nearly all users can use the system without any prior training, the security details are so transparent that a small percentage of users mistakenly sent out unencrypted messages and some users are unsure whether they should trust Pwm. We then conducted user studies with an alternative prototype to Pwm that uses manual encryption. Surprisingly users were accepting of the extra steps of cutting and pasting ciphertext themselves. They avoided mistakes and had more trust in the system with manual encryption. Our results suggest that designers may want to reconsider manual encryption as a way to reduce transparency and foster greater trust.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation (e.g. HCI)**]: User Interfaces—*user-centered design, evaluation*

General Terms

Design, Human Factors, Security

Keywords

Usable security, secure email, manual encryption, security overlays

Symposium on Usable Privacy and Security (SOUPS) 2013, July 24–26, 2013, Newcastle, UK.

1. INTRODUCTION

Secure email solutions exist but have not been widely adopted. Research indicates that this is due in part to usability issues, especially in the areas of key management and portability [19, 15]. These issues are a significant impediment to secure webmail, as users expect high levels of usability and portability from their webmail systems. We believe that users will adopt a secure webmail system only if it is tightly integrated with their existing webmail systems in order to maintain the usability and convenience they are accustomed to. If secure webmail becomes a burden to users, they will reject it and choose instead to focus on their primary goal to send and receive email.

This paper presents results and lessons learned from usability studies of Pwm (Private WebMail, pronounced "Poem"), our solution to extend existing webmail systems (Gmail, Hotmail, Yahoo! Mail) with end-to-end encryption and message integrity. Pwm's security is mostly transparent; key management details are hidden and users are never exposed to ciphertext. Pwm was designed to maximize usability so that users would be willing to adopt it. Pwm integrates tightly with webmail providers' interfaces using security overlays, reducing the burden a user feels towards learning a new system.

The first research question addressed in this paper is how usable is Pwm's tight integration with existing webmail systems using security overlays and its transparent encryption. Pwm was designed to help everyday users send and received encrypted email with little or no training. We conducted IRB-approved user studies of Pwm where nearly all participants were able to decrypt secure messages sent to them without any prior notice or training.

However, these user studies revealed two concerns: First, some users did not trust that the system was secure because the security details (key management and encryption) were so transparent that they did not have a clear idea about how the system actually worked. Second, a small but consistent number of users accidentally sent plaintext when they intended to communicate a sensitive message. Since the steps a user takes to send a message are quite similar for both encrypted and unencrypted data, a user's "click-whirr" response makes them susceptible to sending sensitive messages without first enabling Pwm [5]. Several users realized their mistake immediately after they sent the message, but the damage was already done.

These problems caused us to reconsider hiding some of the security details. The second research question addressed in this paper is whether manual encryption in an application separate from the browser would prevent users from mistakenly sending out sensitive information without encryption, compete with Pwm in terms of usability, and reveal enough security details that users would have greater trust in the system. To test this hypothesis, we built Message Protector (MP), a mockup that included manual encryption in an application separate from the browser.

We conducted two more IRB-approved user studies using MP. We were surprised that users rated MP with manual encryption to be as usable as Pwm. Users also had more trust in MP with manual encryption and avoided the mistake of sending out sensitive messages unencrypted. However, more users preferred that security systems be tightly integrated with the browser. Thus, in the effort to balance security and usability, we argue that a combination of exposing some encryption details and tight integration will produce a system that users trust and help them to secure their data without making mistakes.

The remainder of this paper is organized as follows: Section 2 describes Pwm, and Section 3 presents the user studies of Pwm. Section 4 describes MP, and Section 5 presents the user studies of MP. Section 6 presents the limitations of our user studies and Section 7 discusses related work. Section 8 contains conclusions and future work.

2. PWM

Based on earlier research and our experience, we believe that there are three problems inhibiting the adoption of secure email by the masses:

1. Users are resistant to change. If secure email requires too much effort for the perceived benefits it will be rejected by users [11].

2. Users do not understand how to obtain, distribute, or use cryptographic keys [19, 15]. Additionally, PKI-based secure email has a chicken and egg problem, as most users will not perform key management until they have received an encrypted email, and users cannot receive an encrypted email until they perform key management.

3. Users are confused by the details of cryptography [19]. This leads users to omit or incorrectly use various cryptographic operations necessary for securing email.

We hypothesized that if these difficulties were overcome, users would be able to successfully use secure webmail and be willing to adopt it. Based on this hypothesis, we developed Pwm (Private WebMail, pronounced “Poem”). Pwm adds end-to-end encryption and message integrity to existing webmail systems (Gmail, Hotmail, Yahoo! Mail) and runs in all major browsers (Chrome, Firefox, Internet Explorer, Opera, and Safari). Pwm is designed to maximize usability and provide additional security to users who are already sending sensitive information over email. Pwm addresses the problems we identified as follows:

1. Pwm tightly integrates with existing webmail systems using *security overlays*, windows placed over the webmail providers interface that allow users to interact with secure content. Security overlays are functionally transparent to users helping avoid the frustration of learning a new system.

2. Key management is automatic and fully transparent to users. Keys are managed by a key escrow server that uses email-based identification and authentication (EBIA [8]) to authenticate users without their interaction.

3. All encryption is handled automatically by Pwm, and users are never directly presented with ciphertext or the details of encryption.

2.1 Security Overlays

Pwm uses security overlays to tightly integrate new security features into existing webmail interfaces. A security overlay is a window where users view and interact with secure content. It is positioned directly over the portions of the webmail provider’s interface that need to be secured. The user interacts with the security interface in lieu of the overlaid portion of the webmail provider’s interface. A security overlay is displayed using an iFrame and uses the browser’s same domain policies to protect its contents from access by the honest-but-curious webmail provider.

Security overlays are designed to be functionally transparent to users, matching the functionality that exists in the overlaid portion of the webmail provider’s interface. This functional transparency allows users to complete tasks in the way they are accustomed to, lowering the chance that users will disable the secure system to more readily complete their tasks. Security overlays are also designed to be visually distinctive from the webmail provider’s interface. This distinction assists users in determining whether they are using a security overlay or the webmail provider’s original interface and highlights features unique to the security overlay.

For example, Figure 1 is an encrypted Pwm email and Figure 2 is that same message after it has been decrypted. The security overlay has been positioned in the page where users expect to read email. Functionally, it is identical to reading any other message, but visually it is distinctive and allows users to quickly identify when they are reading encrypted emails. We avoid visual transparency as that would prevent users from determining when the system is in use and reduce trust in the system [7].

2.2 Key Management

A key escrow server handles key management. The key escrow server follows the principles of identity-based

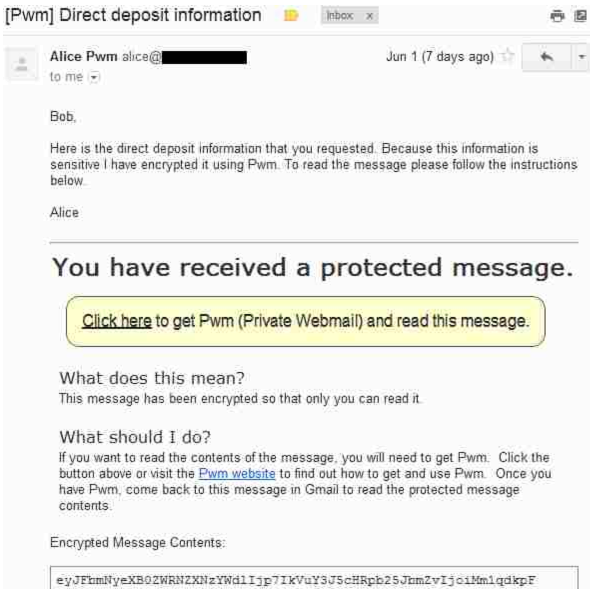


Figure 1: A sample email prior to decryption

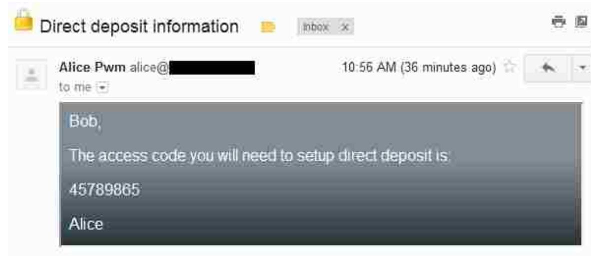


Figure 2: Decrypted email

cryptography (IBC) introduced by Shamir [14] in that cryptographic keys are generated based on users' identities (i.e., email address). This model allows users to send encrypted email to recipients who are currently outside the system. Unlike IBC, the key escrow system uses symmetric key cryptography and key derivation [4, 12] instead of public key cryptography. The advantages of key escrow are (1) key management can be fully automated, (2) users can never lose their encryption keys, and (3) keys can be automatically ported to new devices. The disadvantage of key escrow is that the key escrow server has access to users' keys, which is a recognized trade-off to get the other usability benefits [1].

Pwm interacts with the key escrow server using an invisible key management security overlay. This security overlay handles all key management operations (e.g., obtaining and storing keys, authentication). Authentication is handled by Simple Authentication for the Web (SAW [18]), a form of email-based identification and authentication (EBIA [8]). SAW generates an authentication token for the key server and then splits it in half. One half is returned to the requester and the other half is sent to email account that is being authenticated. Pwm runs inside the webmail provider and has access to this email, allowing Pwm to obtain the full authentication token without reliance on user input. The combination of key escrow and SAW allows for transparent and automatic key management, removing many of the difficulties users faced with traditional secure email solutions.

2.3 Automatic Encryption

Pwm hides almost all security details from users. Encrypted Pwm emails include ciphertext, but it is positioned so that it will be largely ignored by users (Figure 1). The most prominent portion of an encrypted email is the instructions for setting up Pwm. These instructions are designed to help first time users obtain the software needed to decrypt the email. Optionally, the sender of an encrypted email can add a personalized message explaining the nature of the encrypted message and the need to obtain Pwm to access it. This message can provide important context so the recipient can trust that the message is legitimate.

Once Pwm is installed and running, it automatically decrypts the email and displays the plaintext contents of the message to the user in a security overlay (Figure 2). If they had opened the email with Pwm already running, they would only see the decrypted message and not the encrypted email. Users can detect that they are reading a decrypted message because of the visual distinctiveness of security overlays and the addition of a lock icon to the subject of the message (Figure 2 and Figure 3)

When a user replies to an encrypted email, their response is automatically encrypted for them. Unlike replies, new messages are not encrypted by default. Instead, users are presented with an open lock icon next to the email compose form that must be clicked in order to activate the security overlay for composing encrypted email (Figure 4). When users click the send button on the security overlay, the message is encrypted and sent automatically without ever showing them ciphertext. We take this user-centric approach to maximize the impact encrypted emails have on users by reserving encryption for when it is needed [10]. Since new users must install software in order to read Pwm messages, it limits the number of new users required to install Pwm to those that need to access sensitive messages. It also helps to minimize the impact on webmail's ad-based revenue model. Encrypted email limits the webmail provider's ability to scan user's messages in order to serve targeted ads. If default encryption produced a surge of encrypted emails, this might cause a webmail provider to block Pwm traffic or actively limit Pwm's ability to tightly integrate with the webmail interface. For convenience, a user can turn on encryption permanently for a given recipient that they always desire to communicate with securely.

2.4 Setup

The prototype can be installed and run using either a browser extension or a bookmarklet. A browser ex-

Figure 3: New Pwm email in inbox

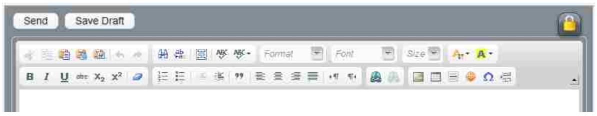


Figure 4: Compose interface

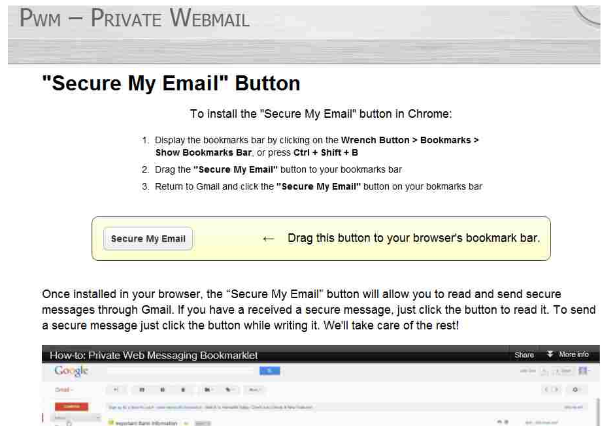


Figure 5: Pwm website

tension is a well-known method for adding functionality to the browser, but bookmarklets are a newer and increasingly popular method for doing the same thing. A bookmarklet is simply a browser bookmark that contains JavaScript instead of a URL. The bookmarklet and the browser extension both function by inserting the in-page services script tag onto the webmail provider's web page. The only difference between the two is that the browser extension is always running, while the bookmarklet must be clicked each time the user visits Gmail.

Bookmarklets have several advantages in comparison to browser extensions, the most important being ease of setup. On the prototype's website, the bookmarklet is represented by a large button with the text "Secure My Email". Installation is as simple as dragging this bookmarklet to the bookmarks bar. Bookmarklets are also quick and easy to use, whenever Bob wants to run the prototype he only needs to click the "Secure My Email" bookmarklet in his bookmarks bar.

As demonstrated by the success of Pinterest,[1] average users are able to set up and use bookmarklets with little difficulty. Since this does not qualify as installation in the traditional sense, Bob does not need administrative privileges to use it. Furthermore, the prototype can be set up and run on any computer where Bob accesses webmail.

3. PWM USER STUDIES

We conducted two IRB-approved user studies to evaluate the usability of Pwm. The goal of these studies was to test whether Pwm's design would lead to secure email that was both usable and desirable for users. This included determining whether new, untrained users could set up and use Pwm relying only on the directions provided in the plaintext portion of the encrypted email (Figure 1) and Pwm's website (Figure 5). We also wanted to discover what, if anything, caused users to fail when sending and receiving encrypted email. Finally, we wanted to determine users' opinions toward the tight integration provided by security overlays.

The participants for both studies were recruited at Brigham Young University using posters that invited students to participate in a Gmail usability study, but did not alert them that it was related to security. To minimize unfamiliarity with Gmail impacting our results, we stipulated that volunteers for the study should be active Gmail users. We also indicated that the study would take approximately thirty minutes and provide compensation of $10.

During this study, all participants used the same computer[2]. We provided Gmail accounts to participants to use in the study. This allowed us to preserve the privacy of the participants' personal accounts and furthered our ability to provide a uniform environment. Participants were required to complete the study using the Google Chrome Web browser.To match a fresh install of Google Chrome, we ensured that the bookmarks bar was not displayed initially. Before beginning, participants completed a demographic questionnaire (Appendix A.1). Users were not required to identify themselves, and we did not record the identity of any participant.

3.1 Bookmarklet Study

3.1.1 Setup

This study used the bookmarklet version of Pwm and was comprised of 25 students, representing 19 different majors, and with low to medium technical experience. Of the 25 participants, 19 (76%) had been Gmail users for over a year and only 3 (12%) had been Gmail users for less than 6 months. Twenty-three (92%) of the participants used Gmail on a daily basis.

We remotely monitored each user's actions in real-time using RealVNC and recorded their actions locally using CamStudio. Participants were presented with simple tasks to complete using Pwm (Appendix A.2). After

[1]Pinterest is a website that makes heavy use of bookmarklets. `http://pinterest.com/`

[2]3.0 GHz Intel Core 2 Quad CPU with 8 GB of RAM

completing the tasks, participants were presented with a short survey about their experience using Pwm (Appendix A.3). We then augmented this survey with a brief interview in which we asked each participant about difficulties or failures we had observed.

3.1.2 Tasks

Each participant was given three tasks to complete using Pwm. These tasks were designed to simulate what an individual would experience if they received an unsolicited Pwm email and began using Pwm.

In the first task, participants were told to check their inbox for an email containing instructions on how to proceed with the study. Unknown to them, this email had been encrypted using Pwm. Participants were given no explanation or help from the study conductor and were required to rely only on the directions provided by the encrypted Pwm email. Once decrypted, the email instructed participants to send an encrypted reply and return to the study instructions. The primary goal of this task was to observe whether untrained users could successfully set up and use Pwm with no outside assistance. Because we were in a lab environment where participants knew they would not be exposed to any real risks, we refrained from drawing any conclusions about participants' trust in bookmarklets.

For the second task, participants were asked to open a new Gmail session, send an encrypted email to the study conductor, and then wait for a reply with further instructions. If participants did not encrypt their email, they would then receive an unencrypted reply informing them that their email had not been encrypted and instructing them to try again. Once the participant successfully sent an encrypted email, they received an encrypted reply instructing them to close Gmail and return to the study instructions.

The third task required a new Gmail session be started. Since Pwm was no longer running, the participant would need to restart Pwm by clicking on the bookmarklet. The primary goal of this task was to determine whether participants would be able to correctly restart Pwm in order to compose an encrypted email.

3.1.3 Results

Overall, participants were highly successful in using Pwm. All but one of the 25 participants (96%, Confidence Interval (CI) at 95%, $\pm$7.68) successfully set up Pwm and decrypted the email received in the first task. The only participant who failed to decrypt the email had correctly set up Pwm but then moved on to the second task without trying to read the decrypted email. When asked why she did this, she said that it was because she assumed the task was complete once she had added the bookmarklet. This was a flaw in the task setup because we should have had information contained in the encrypted message that participants needed to report in order to continue on with the study. This would also have more closely resembled real world use cases.

Of the 24 that decrypted the email in the first task, 23 (96%, CI $\pm$7.84) successfully sent an encrypted reply. The only participant who failed to send the encrypted reply had correctly used Pwm but then clicked Gmail's "Compose" button rather than the "Send" button. He did not repeat this error on the second task. When asked about this, he said that he was accustomed to using Gmail on his iPod Touch where the send button is in the upper left-hand corner of the screen where the "Compose" button was in our test.

On the third task, 22 participants (88%, CI $\pm$12.74) successfully sent an encrypted email on their first try. Of the three who failed, one immediately recognized his mistake and correctly sent an encrypted email before receiving a reply. When asked about this, he reported that he knew it wasn't encrypted when he didn't see the security overlay's black background. The remaining two participants successfully sent an encrypted email after receiving the reply asking them to try again. One of the two stated that they had misread the instructions and didn't realize they were supposed to encrypt the email. The other reported that he didn't realize he needed to click the bookmarklet again and said that he wouldn't repeat that mistake again.

3.1.4 System Usability Scale

We used the System Usability Scale (SUS) [3], a usability evaluation metric developed at Digital Equipment Corp., to rate the usability of Pwm. SUS works by asking participants to respond to ten statements on a Likert scale. We included these statements as part of the survey we administered to participants. Based on the participants' responses we calculated a SUS score of 75.70 out of 100 (standard deviation (SD) of 13.61, CI $\pm$5.33) for Pwm.

As part of an empirical evaluation of SUS, Bangor et al. [2] reviewed SUS evaluations of 206 different systems and compared these against objective measurements of the various systems' success to derive adjective-based ratings for SUS scores (Appendix C, Figure 1). When compared against Bangor's findings on 273 SUS studies, our score of 75.70 falls in the third quartile (70.5–77.8) and above the mean score of 69.5. Pwm's score qualifies for an adjective rating of "Excellent" and is considered "acceptable" in Bangor's acceptability range.

3.1.5 Lessons Learned

Overall, this study was a success. Pwm succeeded in helping first-time users set up and use secure email. When asked in the survey what they liked about Pwm, 23 out of 25 (92%) stated that it was simple and easy to use. No participant indicated that they felt Pwm was difficult to use. Most participants stated that they would use Pwm if they needed to send secure email. Five of the participants (20%) even asked if Pwm was available for download because they wanted to begin using it immediately.

Participants were able to clearly tell the difference between Pwm's secure interface and the underlying interface. Some liked the distinct black background of the security overlay while others wished it looked more like Gmail's native interface. When asked, all participants indicated that it was easy to determine when email had been encrypted using Pwm. The three participants (12%) that initially failed the second task indicated that in the future they would not make the same mistake as they would ensure they could see the distinctive look of the security overlay before sending a sensitive message. While it is hard to know if this is correct, it is still encouraging that users were able to recognize the importance of the visual distinctiveness.

Bookmarklets also proved to be highly usable. Only five (20%) of the participants had used a bookmarklet before; nevertheless, all participants were able to set up and use Pwm. Many participants noted that they liked the fact that the Pwm bookmarklet did not require installation. The one complaint was that the instructions for how to install the bookmarklet should have been more prominent. No participant demonstrated pre-existing knowledge of how to enable the bookmarks bar and the instructions were crucial in helping them set up Pwm. The participants who read the instructions before attempting to add the bookmarklet set up Pwm far faster (average of 30 seconds) than those that tried to add the bookmarklet without first reading the instructions (average of 1.5 minutes).

We asked about half of the participants, including the three who struggled with the second task, how they would react to having email encrypted by default. We explained that this would ensure that they would not accidentally send email without encryption. The participants disliked this idea. In their minds, they saw encryption as something that they would only turn on for sensitive messages and thought it would be annoying to need to frequently turn off encryption. They recognized that decrypting messages adds work for the recipient, and wished to avoid this unless the message was important. Some participants rejected automatic encryption because they believed that only Gmail users could install Pwm and read messages they had sent. Also, it is possible that the short-term nature of the study unfairly biased participants against the ease of decrypting messages as a disproportionate amount of their time (in comparison to real use scenarios) was spent installing Pwm.

We were interested to discover that approximately one third of participants were interested in how their email was being encrypted. Although these participants lacked the technical background to fully understand the cryptography being used, they would still like to see these details published on Pwm's website. They indicated that this would make them feel more confident using Pwm. Even though Pwm users do not want to be concerned with cryptographic details (e.g., key management, signing) while operating Pwm, they still want this information available so that they can feel more confident that Pwm is securing their messages.

3.2 Voltage Comparison Study

In order to establish that Pwm provided usability benefits in comparison to existing depot-based secure email solutions, we conducted a user study comparing Pwm with Voltage SecureMail Cloud[3] (hereafter referred to as Voltage). Like Pwm, Voltage was designed to allow messages to be encrypted and sent to recipients who had not taken any preparatory action. In addition to comparing usability of similar features, comparing Pwm against Voltage also allowed us to compare users' reactions to secure email systems requiring software installation (Pwm) against systems requiring account creation and verification (Voltage).

In this study, Pwm was run using a browser extension. In the first user study, some participants suggested they would prefer to use an extension to a bookmarklet, and we wanted to see if using the extension would make any difference in the user's experience.

3.2.1 Setup

The second study was comprised of 32 students. Like the Pwm studies, participants were aware they were taking part in a usability study, but were unaware of its focus on security. All but one (97%) participant had been using Gmail for over six months, and 27 (84%) reported that they used Gmail on a daily basis. All participants had experience using Google Chrome. Two of the participants (8%) had encountered PGP in the past but had never used it for any significant period of time.

This study was a within-subjects study, where participants were given simple tasks to complete using both Pwm and Voltage. The order in which they used the systems was randomized so that half used Pwm first, and the other half Voltage. After completing the tasks for one system, they were given a survey rating their experiences (Pwm – Appendix A.4, Voltage – same questions as Pwm, but with "Voltage" replacing "Pwm"). Participants would then complete the tasks and associated survey for the other system. Participants were given a survey with questions about their online security behavior and preferences (Appendix A.5). Finally, participants were interviewed to gather more information about their experiences.

3.2.2 Pwm Tasks

The tasks for Pwm remained the same as the first Pwm study. The only difference was the instructions on the Pwm website were replaced with instructions for setting up and running the browser extension instead of the bookmarklet.

3.2.3 Voltage Tasks

To begin, participants opened an email that had been sent to the provided Gmail account using Voltage. This email, which was generated by Voltage, contained an HTML attachment that included a link to the Voltage website where they could read their encrypted email. At the Voltage website, participants were instructed to create a free Voltage account in order to view their message. Participants had been provided with fake credentials that they could use to fill in the account creation form. After submitting this information, participants were directed to return to their email to retrieve an account verification email from Voltage. After verifying their new Voltage account, participants were able to return to the Voltage website and read their encrypted message. This message instructed them to send a secure reply, which in Voltage only requires clicking "reply."

Unlike the Pwm tasks, participants were not required to send a new encrypted email through Voltage. The participants were using free Voltage accounts, which do

[3] `http://www.voltage.com/products/vsn.htm`

not allowing sending new email (only replying), and licensing fees made it impractical to give each participant a commercial account. This step is trivial in Voltage, as it is no different than sending an email in any depot system, and so we do not believe this omission affects the usability results.

3.2.4 Results

Pwm Results.

As with the first study, all participants successfully set up Pwm, but this time they did so without any mistakes or delays. This is likely due to the ease of installing browser extensions in Chrome (only requires two mouse clicks) as well as greater user familiarity with browser extensions (in the first study 5 participants (20%) has used bookmarklets before, where as in the second study 28 (87.5%) had used extensions previously).

Users did experience confusion about being required to refresh the Gmail page before the extension became active (a limitation of Chrome extensions). Several users needed to return to the Pwm website and re-read the instructions before they refreshed the Gmail webpage.

All of the participants (100%) successfully decrypted the email they received. They also all successfully sent an encrypted reply. In the third task, three participants (9%, CI ±11.22) sent their message without encryption. Two of the three recognized their mistake immediately after clicking "Send;" one recognized the mistake when he saw the lock icon on the compose form just after he clicked "Send" and the other when he realized he had not seen the dark background of a security overlay.

Voltage Results.

As with Pwm, all participants (100%) successfully read their encrypted message and replied to it. However, 14 of the 32 participants (44%, CI ±17.32) complained that the process for reading the initial email was extremely cumbersome. Two participants (6%, CI ±8.23) expressly stated that they did not want to leave Gmail to access encrypted email. Overall, only six participants (19%, CI ±13.59) indicated that they preferred Voltage. Of these participants, four preferred the look and feel of Voltage's website, one disliked installing any browser extensions, and one liked that there was a separate site for handling secure messages.

3.2.5 System Usability Scale

Pwm's extension implementation had a calculated SUS score of 70.70 (SD 12.28, CI ±4.26). Voltage had a calculated SUS score of 62.66 (SD 17.53, CI ±6.07). This is a statistically significant difference (paired two tailed t-test, $p = 0.0073$). This result matches opinions expressed by participants during the interview at the end of the study. According to Bangor's adjective ratings, both systems qualify for an adjective rating of "Good." Pwm was in the third quartile and above the mean of 69.5 while Voltage was in the second quartile and below the mean. According to Bangor's acceptability ranges, Pwm qualifies as "acceptable" while Voltage ranks as "low marginal."

The SUS score for extension-based Pwm is lower than bookmarklet-based Pwm (75.70) but this difference is not statistically significant (unequal variance two tailed t-test, $p = 0.1576$). We believe that this difference in perceived usability was due to a bug that caused a delay between when the page loaded and Pwm became fully functional. This delay caused visible confusion in 7 of the 32 participants (22%), four of whom later commented that the delay bothered them. This annoyance could have contributed to the lower scores Pwm received in this study.

3.2.6 Lessons Learned

In addition to the SUS results, the user surveys also showed that users largely preferred Pwm to Voltage. When asked why, they stated that it was because Pwm was integrated directly with Gmail. This supports our original hypothesis that users are resistant to systems that require changes to existing behavior, and that tight integration is able to overcome this concern. Also, like the first study, many participants expressed a desire to use Pwm if they needed to encrypt their personal email.

Several participants suggested ways to improve Pwm. First, while they liked the simplicity of the browser extension, they were also interested in having the option of using a bookmarklet to run Pwm, preferring the flexibility that having both options would provide. Second, participants suggested Pwm's website should look more professional and provide additional details on how Pwm functions. While this is not directly useful to most participants, they indicated that their confidence would be bolstered by the knowledge that Pwm's claims are open to scrutiny by security experts. These suggestions closely parallel suggestions from the first user study. Also, of the six users who preferred Voltage, only one preferred it for reasons inherent to depot-based secure email, while the other five would have preferred Pwm if these two issues were addressed.

4. MP – MANUAL ENCRYPTION

The results from our Pwm user studies along with results from a user study of Private Facebook Chat[13] (a companion system built using security overlays for Facebook Chat) were very positive, especially when compared to some earlier usability studies for secure email [19, 15]. However, several aspects of Pwm and PFC were concerning. First, each study had a small number of users (approximately 10%) that forgot to enable encryption before they sent secure messages. Second, follow-up interviews with participants revealed that some of them did not understand how Pwm works, believing that anyone with Pwm installed could decrypt their email if they stole it.

Initially we thought these issues would be simple to resolve. We considered modifying Pwm to better train users, including displaying video explaining how to encrypt message the first time they ran Pwm, but ultimately felt that this would make users think Pwm was either spam or a virus. We also considered turning on encryption for all messages, but decided against doing so because it places an undue burden on each recipient and

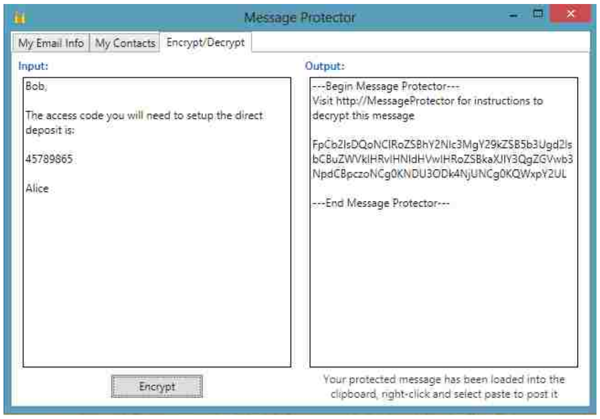


Figure 6: Interface for encrypting a message

potentially interferes with the revenue model of webmail providers; when asked about this option, participants in the user studies also rejected it as undesirable.

Ultimately, we concluded that Pwm's security details were too transparent. This transparency led some participants to author and click "Send" before realizing they had not enabled encryption (a "click-whirr" response [5]). It also caused some participants to not trust Pwm. Since the users did not see the ciphertext, they lacked confidence that encryption and decryption were taking place and didn't understand how the system worked.

We built a mockup of Message Protector (MP), a standalone application that allows users to manually encrypt and decrypt messages. We believed that a separate application with manual encryption could help users better understand what was happening and help them avoid mistakes.

MP is a very simple system. Following installation, MP prompts users to sign in with their email credentials. Users then select email contacts they would like to communicate with securely. These two steps would permit automatic key management and allow users to clearly determine who can read their encrypted messages.

Users are then able to encrypt and decrypt messages (Figure 6). To encrypt a message, they input their desired message into the "Input" text field and click "Encrypt". Their message is then encrypted (base-64 encoded in the mockup) and the ciphertext is placed in the "Output" field. Users then copy the ciphertext to whatever application they wish to use to store or transmit it.

To decrypt a message, users input the MP ciphertext into the "Input" field and then click the "Decrypt" button ("Encrypt" changes to "Decrypt" when ciphertext is detected in the "Input" field). The plaintext is placed in the "Output" field.

5. MP USER STUDIES

We conducted two IRB-approved user studies to evaluate MP. The goal of these studies was to compare MP to existing systems and determine if users would trust the system and find it usable, and if they would make fewer mistakes.

As with the Pwm studies, the participants for these two studies were recruited at Brigham Young University using posters that invited students to participate in either a 45 minute Gmail and Facebook study or a 30 minute Gmail study for the first and second studies respectively. The compensation for both studies was \$10.

All participants used the same computer and browser as in the previous user studies. Once again we provided participants with Gmail and Facebook accounts to use during the studies. Before beginning, participants completed a demographic questionnaire (Appendix B.1). Users were not required to identify themselves and we did not record the identity of any participant.

5.1 Encipher.it Comparison Study

In the first MP study we wanted to test MP against an existing secure data sharing tool with manual encryption. We selected Encipher.it[4] because it is a relatively well-known tool, which unlike most other similar tools is currently functional. Encipher.it is a generic bookmarklet-based secure data sharing tool that can encrypt text in any HTML textbox. To use it the user types a message in a textbox and clicks the Encipher.it bookmarklet. Encipher.it then prompts the user to supply a password which is used to encrypt their message. This password must be transmitted out-of-band to the recipient. Following encryption, Encipher.it displays ciphertext in place of the original plaintext message. When a recipient receives an encrypted message and clicks the Encipher.it bookmarklet, Encipher.it prompts the user for the sender's password. After the recipient supplies the password, the message is decrypted and displayed in place of the ciphertext on the webpage.

5.1.1 *Setup*

This study was comprised of 28 participants. Participants were told that this was a usability study but were not made aware of its security focus. Of the 28 participants, 25 (89%) used webmail daily and 27 (96%) used Facebook weekly. Sensitive information had previously been sent over webmail or Facebook by 24 (86%) of the participants, while only three (11%) of them had encrypted those messages. All (100%) participants reported that protecting the contents of sensitive information was important.

At the beginning of the study, participants were presented with a document that described the study (Appendix B.2). The study was a within-subjects study, where participants were given simple tasks to complete using both Encipher.it and MP (Appendix B.3). The order in which the systems were used was randomly chosen; 16 (57%) participants used MP first and 12 (43%) used Encipher.it first. After completing the tasks for one system, participants were then given a survey to rate their experiences (Appendix B.4). Participants would then complete the tasks and associated survey

[4] `https://encipher.it/`

for the other system. Participants were finally given a post-study survey asking them to state which system they preferred and why (Appendix B.5).

5.1.2 Tasks

Users were given step-by-step instructions on how to complete three tasks using both systems. Task 1 instructed users to install the given system.

Task 2 instructed participants to open Gmail and send an encrypted message containing the text "The last four digits of my SSN is 6789" to the study coordinator. Participants then received an encrypted response to this message and were instructed to decrypt it. To continue they had to input the decrypted message.

Task 3 instructed participants to open Facebook and send an encrypted message containing the text "My bank account password is cougars" to the account's friend named "Alice Jones." Participants then received an encrypted response to this message and were instructed to decrypt it.

5.1.3 Results

MP Results.

Of the participants, 25 (89%, CI ±11.45) correctly completed the Gmail tasks and 27 (96%, CI ±7.17) completed the Facebook tasks. The mistakes were split between not understanding how to use the tool and not understanding which portion of the ciphertext to submit to correctly complete the task.

Participants largely succeeded in understanding how MP worked. Twenty-five participants (89%, CI ±11.45) correctly identified who could read encrypted messages. Additionally, 25 participants (89%, CI ±11.45) were able to correctly identify how to decrypt a message using MP.

Encipher.it Results.

Many participants were not able to get Encipher.it to allow them to encrypt or decrypt messages. Only 16 (57%, CI ±18.34) participants were able to decrypt a message in Gmail and only 14 (50%, CI ±18.52) were able to send an encrypted email. Similar to MP, participants fared a little better using Encipher.it with Facebook, as 17 (61%, CI ±18.07) participants successfully decrypted a message and 23 (82%, CI ±14.23) participants successfully encrypted a message. Four participants (14%, CI ±12.85) failed the encryption tasks because they never communicated to the test coordinator the password they had used to encrypt the message.

Participants largely understood how Encipher.it worked, but not as clearly as they understood MP. Twenty-three (82%, CI ±14.23) correctly identified who could read encrypted messages, but only 20 (71%, CI ±16.81) understood how to decrypt a message.

5.1.4 System Usability Scale

MP had a calculated SUS score of 72.23 (SD 13.02, CI ±4.96). Encipher.it had a calculated SUS score of 61.25 (SD 20.11, CI ±7.65). This is a statistically significant difference (paired two tailed t-test, $p = 0.0176$). According to Bangor's adjective ratings, both systems qualify for an adjective rating of "Good." MP was in the third quartile and above the mean of 69.25, while Encipher.it was in the second quartile below the mean. According to Bangor's acceptability ranges, MP qualifies as "acceptable" while Encipher.it ranks as "low marginal"

5.1.5 Lessons Learned

MP was much better at helping the participants avoid making mistakes (paired two tailed t-test, Gmail Decryption – $p = 0.0171$, Gmail Encryption – $p = 0.0052$, Facebook decryption – $p = 0.0022$, Facebook encryption – $p = 0.1033$ [Not significant]). This is likely due to the higher usability marks received by MP, as users found it far easier to use.

MP also performed better at helping participants understand who could read encrypted messages (paired two tailed t-test, $p = 0.0114$) and also how to successfully decrypt messages (paired two tailed t-test, $p = 0.1610$), though the second result is not statistically significant.

Perhaps the greatest surprise was that MP's SUS score was as high as Pwm in our previous studies. We had not anticipated this outcome, as we felt that the extra effort of manual encryption would cause participants to reject the system. It is clear from participant responses that they felt more confident using MP precisely because it helped them understand what they were doing. This is reflected by the majority of participants who indicated that the usability of the system was important to them in deciding whether they would use it in their personal lives (MP – 24 [86%, CI ±12.85], Encipher.it – 22 [79%, CI ±15.09]), and more people found MP easy to understand (MP – 23 [82%, CI ±14.23], Encipher.it – 17 [54%, CI ±18.46]).

At the conclusion of the study, we asked participants which system they preferred and why (Appendix B.6, Table 1). First, most participants preferred integrating encryption with the browser and several participants preferred Encipher.it primarily for this reason. Participants that preferred MP also indicated that they wish it had been more integrated. Still, there was a small number of users who felt that MP being a separate application increased security. Second, we observe that users recognize the problem of distributing keys, and several disliked that this was a necessary step of Encipher.it.

5.2 Pwm Comparison Study

The results of the initial MP study were very positive and so we decided to compare it against Pwm. MP was a mockup to study manual encryption and lacks any functionality to help first time recipients of an MP message know how to proceed. For this reason, we selected to replicate the previous MP study (with Pwm replacing Encipher.it) instead of modeling this study after the original Pwm studies. The goal of the study was to see how well MP fosters user understanding when compared to Pwm, and also to explore users' attitudes when comparing the two systems.

5.2.1 Setup

This study was comprised of 28 participants. Participants were told that this was a usability study but were not made aware of its security focus. Of the 29 participants, 28 (97%) used webmail daily and Facebook weekly. Sensitive information had been sent over webmail or Facebook by 27 (93%) of the participants, while only one (3%) had encrypted those messages. Once again, all (100%) participants reported that protecting the contents of sensitive information was important.

The setup and tasks for this system were similar to the Encipher study, but did not include the Facebook tasks since Pwm does not support Facebook. The order in which the systems were used was randomly chosen; 15 (52%) participants used MP first and 14 (48%) used Pwm first.

5.2.2 Results

MP Results.

Of the participants, 27 (93%, CI ±9.29) correctly decrypted a message and 28 (97%, ±6.21) successfully encrypted a message. Comprehension was also high, as 27 (93%, CI ±9.29) correctly identified who would be able to read encrypted messages and all 29 (100%) participants correctly identified how to decrypt a message using MP.

Pwm Results.

Twenty-five (86%, CI ±12.63) participants were able to decrypt a message and 24 (83%, CI ±13.67) were able to send an encrypted email. This is lower than previous results for both Pwm and PFC, and this is possibly due to the study not as closely mimic real world conditions. Pwm is designed to help first time recipients of unsolicited encrypted messages, and so does not provide step-by-step documentation like Encipher.it or MP.

A number of participants fared poorly in understanding how Pwm was functioning. Only 22 (76%, CI ±15.54) of the participants correctly identified who could read a Pwm message, and only 21 (72%, CI ±16.34) knew the proper steps to decrypt a message. Perhaps even more interesting is that 6 (21%, CI ±14.82) participants stated they were unsure of who could read messages and 4 (14%, CI ±12.63) were unsure how to read an encrypted message, whereas no users (0%) reported being unsure of how to use MP in either category.

5.2.3 System Usability Scale

MP had a calculated SUS score of 73.96 (SD 14.23, CI ±5.42). Pwm had a calculated SUS score of 75.69 (SD 16.31, CI ±6.21). This was not a statistically significant difference (paired two tailed t-test, p = 0.61633).

In comparison to Bangor's findings both systems qualify for an adjective rating of "Excellent." Both were in the third quartile and above the mean of 69.25 and both qualifies as "acceptable" on Bangor's acceptability scale.

Extension-based Pwm scored higher in this study then in the second Pwm user study, but this difference was not statistically significant (unequal variance two tailed t-test, p = 0.1867). Extension-based Pwm in this study scored nearly identically to the bookmarklet-based version of Pwm from the first Pwm user study (unequal variance two-tailed t-test, p = 0.9980). In aggregate across all studies Pwm had a SUS score of 73.84 (SD 14.17, CI ±3.04) and MP had a SUS score of 73.14 (SD 13.56, CI ±3.60). This was not a statistically significant difference (unequal variance two tailed t-test, p = 0.7596).

5.2.4 Lessons Learned

MP's manual encryption and clear separation led to nearly all participants correctly understanding who could read messages (paired two tailed p-test, p = 0.0225) as well as how to decrypt a message (paired two tailed p-test, p = 0.0028). Since Pwm keeps more security details transparent to its users, they did not understand how Pwm works and were aware of their lack of understanding.

We were once again surprised that MP performed on par with Pwm in terms of usability. Contrary to our initial thinking, users are not opposed to manual encryption. Users preferred manual encryption because they felt it helped them understand, and thereby trust, the system. Even though MP is a mockup and Pwm is a working system, participants felt that MP was more secure based on its manual encryption.

At the end of the study, we again asked participants which system they preferred and why (Appendix B.6, Table 2). Their answers are helpful in understanding how these results should guide our research. First, users preferred the integration provided by Pwm. Even users who preferred MP were likely to state that they felt Pwm was more usable, but choose MP because they didn't feel they could trust Pwm. Second, participants felt that manual encryption was necessary to their understanding. Without seeing the ciphertext, they did not feel that Pwm was actually encrypting messages and so were unwilling to use it, and accordingly did not feel that Pwm's other usability benefits were enough to overcome this concern.

6. LIMITATIONS

There are three key limitations in our user studies:

1. The first MP study was an exploratory study designed to measure the potential benefits of manual encryption. MP was not designed to spread in a grass root fashion like Pwm, so the first MP user study assumed the user had already installed the necessary software and shared secret keys before they received an encrypted message. Once the results from the first study indicated the potential benefits of manual encryption, we decided to compare MP against Pwm. We modeled this comparison study after the first MP study for consistency. Thus, the second MP study assumed the user had already installed Pwm. The user tasks focused on encryption and decryption and not software installation. While this proved sufficient for comparing comprehension, the results of this study are not fully representative of all aspects of Pwm, and potentially biased participants by not allowing them to experience one of Pwm's key usability features.

2. Our user studies were short-term laboratory studies. Short-term studies have several inherent limitations: first, it is hard to accurately address trust in a laboratory setting [16], and second, it does not allow us to analyze whether participants would correctly use the prototype over an extended period of time. Perhaps MP's lack of integration would become a considerable issue after repeated use. Similarly, the perception of Pwm's usability could change over time. In the future, we plan to conduct long-term studies to address these issues.

3. The first SUS question reads "I think that I would like to use this system frequently." In the post-study interviews, we determined that users were giving Pwm lower scores for this question because they did not feel that they would send secure email very often, even though they were enthusiastic about using Pwm whenever they would send secure email. Thus, SUS scores may be negatively impacted by an important yet infrequent activity, even if the tool for performing that activity is highly usable.

7. RELATED WORK

Whitten and Tygar conducted a usability study of PGP 5.0 in their seminal paper on usable security [19]. It served as a wakeup call to the security community because a large percentage of users failed to complete basic tasks installing and using a state-of-the-art secure email tool. In their study, 3 of the 12 users (25%) mistakenly sent the secret message unencrypted. In our work, we demonstrated a secure webmail tool with very high success rates sending encrypted email, but we also observed a small percentage of users mistakenly sending out plaintext.

Sheng et al. [15] repeated the Whitten and Tygar user study with PGP 9.0. They noted some improvements due to automatic encryption, but they identified a number of problematic issues surrounding key management and digital signatures. One of their major findings was that encryption was so transparent that users were unsure whether it had occurred or not. The paper recommends that users be given the option to designate in advance whether an email is to be encrypted or not. We designed Pwm to follow this suggestion, but our own studies indicate that users can still mistakenly send out a sensitive message without encryption.

Garfinkel and Miller [9] created a secure email system that combined the idea of Key Continuity Management (KCM) with S/MIME. They introduced a tool to Outlook Express that would alert users through visual indicators if a sender that had previously sent them secure email was now sending an email that was not signed or was signed by a different key. They repeated the original Johnny experiment with some additional tasks to test how users reacted to attacks against the KCM system. Their work demonstrated that automatic key management provides significant usability compared to earlier studies that burdened users with key management tasks. They observed that their tool "was a little too transparent" in how well it integrated with Outlook Express, and sometimes users failed to read the instructions accompanying the visual indicators. Our work also illustrates the benefits of automatic key management, but we use a very different key management paradigm based on identity-based cryptography since we focus on making it easy for users to obtain our software after they have received an encrypted message and to start encrypting their webmail. We also observed some issues related to too much transparency. Our work is complimentary, and we could incorporate KCM to address the kinds of attacks they describe in their paper.

Clark et al. [6] analyze the P25 short-range wireless two-way communications protocol used for emergency and law enforcement personnel. They discovered that a small amount of sensitive traffic is inadvertently sent unencrypted due to individuals and groups being confused about when encryption is actually turned on. One contributing factor is the user interface design that enables encryption by rotating a knob to a specific position. They observed that users occasionally assume encryption is on and mistakenly communicate in the clear. We experienced a similar phenomenon with our software interface that lets users turn encryption on and off.

Fahl et al. [7] conducted usability studies for various design options for Facebook private messaging. They determined a strong user preference for automatic key management. They also selected automatic encryption, but there wasn't a significant preference for it compared to manual encryption. They suspected that making encryption details too transparent could fail to generate a feeling of trust among the users of a system, and they recommended that this issue be explored in more detail in the future. Our work provides evidence to confirm this suspicion. We had users report that they had more trust in a system that exposes more security details.

Sun et al. [17] examined the usability of OpenID, a promising Web single sign-on system. They identified concerns and misconceptions among users that inhibit the adoption of OpenID. They illustrate how the OpenID login flow promotes an inaccurate mental model to users. They describe an alternative to the OpenID login flow that assists users in forming a more accurate mental model and believe that this will help users be more likely to adopt OpenID. In our research, we observed that some users were wary about adopting our email system even though they found it easy to use because the security details were too transparent.

8. CONCLUSIONS AND FUTURE WORK

The contributions of this paper are:

- An overview of the design of Pwm, which layers encryption over existing webmail solutions using a novel approach of security overlays that are functionally transparent but visually distinctive. Pwm is specifically designed to spread in a grass roots fashion so that a user can send any other user an encrypted message before the recipient generates any cryptographic keys or installs any software.

- The results of a series of usability studies of Pwm that compare it to several existing secure email tools. The systems are compared using a standard usability metric, System Usability Score (SUS). Pwm is shown to be highly usable and compares favorably to the other tools used in the studies.

- Even though most Pwm users in the study encrypted and decrypted messages correctly, a few users mistakenly sent out secure messages in the clear. Users were unsure whether to trust the system because security details are too transparent. We compared Pwm to MP, a mockup that uses manual encryption. We were surprised that users rated the usability of MP on par with Pwm. They had more trust in MP and avoided mistakes. Our results suggest that designers may want to reconsider manual encryption as a way to reduce transparency and foster greater trust.

Usable secure webmail has been a long unsolved problem. Pwm (Private WebMail) is a system that adds end-to-end encryption and message integrity to existing webmail systems. Pwm addresses the problem of usability in secure email in three ways: First, Pwm integrates tightly with existing webmail interfaces, providing functionally transparent interfaces, to relieve the burden users feel when learning new systems. Second, Pwm features fully automatic key management that requires no interaction from users. Third, Pwm provides transparent and automatic encryption, whereby users can trivially encrypt and sign their messages. Overall, Pwm was designed to maximize usability while still providing good enough security and is advantageous to those already sending sensitive information over webmail.

To verify the usability of Pwm we conducted two IRB-approved user studies with 25 and 32 participants respectively. In a laboratory setting, the participants were sent an encrypted email that also contained plaintext instructions on how to set up and use Pwm in order to read the message. Every participant, except one who misunderstood the instructions, was able to decrypt and read the message. The second study included a comparison with Voltage, an existing depot-based email encryption system.

Even though participants gave Pwm high usability marks, a small but consistent percentage of participants (approximately 10%) forgot to enable encryption. Some users did not trust that the system was secure because the security details (key management and encryption) were so transparent that they did not have a clear idea about how the system actually worked. We speculated that a combination of manual encryption and a clear separation of duties would help users trust the system and avoid sending information without encryption. To test this hypothesis we built Message Protector (MP), a mockup that included manual encryption in an application separate from the browser.

Using MP, we conducted two more IRB-approved user studies with 28 and 29 participants respectively. In the first study we compared MP against an existing secure data sharing tool, Encipher.it. MP proved to be significantly more usable than Encipher.it and also helped users better understand what security was being provided. We then tested MP against Pwm and found evidence that manual encryption can foster greater trust and reduce user errors. The user studies also revealed that participants preferred that security systems be tightly integrated with the browser.

Thus, in the effort to balance security and usability, we argue that a combination of exposing some encryption details and tight integration will produce a system that users trust and help them to secure their data without making mistakes.

8.1 Future Work

We were surprised that users were accepting of the extra effort that manual encryption requires in MP compared to the transparent encryption in Pwm. Even more significant was the feeling of trust fostered by manual encryption in MP. We are not yet convinced that MP should replace Pwm because there was a strong preference for tight integration with the website from a number of users. Also, we believe MP would be unacceptable to users when sensitive information is exchanged so rapidly (e.g., secure chat) that it would require repeated switching between applications. Instead, we intend to combine the advantages of Pwm and MP into a hybrid system that leverages the strengths of MP in order to overcome weaknesses in Pwm.

Our experience demonstrates that automatic encryption hides so many details that users are confused about what precisely is occurring and can sometimes lead users to mistakenly disclose plaintext when the encryption option is too similar to no encryption. We plan to take these lessons learned and apply them to our next-generation secure Webmail tool. We plan to support manual encryption and explore varying degrees of separation in order to eliminate confusion and mistakes. For example, we plan to add manual encryption to Pwm's existing security overlays. Instead of treating encryption and transmission of email as one step, we will have the compose security overlay produce cipher text, and then require the users to separately click "Send" on the webmail interface. Additionally, we will require that users manually choose to decrypt Pwm email messages by clicking a button, instead of having this occur automatically.

Another potential way to incorporate manual encryption, but with more separation, would be to move encryption and decryption into a sidebar. This sidebar would be hosted in a security overlay and would be independent of the underlying website. This independence would allow it to support all webmail providers, as well as any other websites that allow users to share text. It could also work in conjunction with the more integrated portions of Pwm by becoming the fallback mechanism used when the more integrated Pwm is unable to parse a website.

We will experiment with these and similar ideas to strike a balance between manual encryption/separation and usability/integration. Hopefully this would raise the SUS score of Pwm to increase the probability that users will recommend it to their friends.[5] We will conduct further laboratory studies to verify how well our improved system is trusted and helps users avoid mistakes. Finally, we will then conduct a long-term usability study to determine if our results carry over into the real world where users would use Pwm to protect their sensitive data.

9. ACKNOWLEDGMENTS

The authors would like to thank Trevor Florence, Kimball Germane, Scott Robertson, Chris Robison, and Ryan Segeberg for their work in the Internet Security Research Lab (ISRL) at BYU to develop ideas and software related to this work. They also thank the anonymous reviewers and shepherd for their valuable feedback.

10. REFERENCES

[1] H. Abelson, R. Anderson, S.M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P.G. Neumann, R.L. Rivest, J.I. Schiller, et al. The risks of key recovery, key escrow, and trusted third-party encryption. *World Wide Web Journal*, 2(3):241–257, 1997.

[2] 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.

[3] J. Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189:194, 1996.

[4] O. Chevassut, P.A. Fouque, P. Gaudry, and D. Pointcheval. Key derivation and randomness extraction. In *In Crypto'05*. Citeseer, 2005.

[5] Robert B Cialdini. *Influence: Science and practice*, volume 4. Allyn and Bacon Boston, MA, 2001.

[6] Sandy Clark, Travis Goodspeed, Perry Metzger, Zachary Wasserman, Kevin Xu, and Matt Blaze. Why (special agent) johnny (still) can't encrypt: a security analysis of the apco project 25 two-way radio system. In *Proceedings of the 20th USENIX Conference on Security*. USENIX Association, 2011.

[7] Sascha Fahl, Marian Harbach, Thomas Muders, Matthew Smith, and Uwe Sander. Helping johnny 2.0 to encrypt his facebook conversations. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 11:1–11:17, Washington, D.C., 2012. ACM.

[8] Simson L. Garfinkel. Email-based identification and authentication: an alternative to PKI? *IEEE Security & Privacy*, pages 20–26, 2003.

[9] Simson L. Garfinkel and Robert C. Miller. Johnny 2: a user test of key continuity management with s/mime and outlook express. In *Proceedings of the 2005 symposium on Usable privacy and security*, SOUPS '05, pages 13–24, Pittsburgh, Pennsylvania, 2005. ACM.

[10] S. Gaw, E.W. Felten, and P. Fernandez-Kelly. Secrecy, flagging, and paranoia: adoption criteria in encrypted email. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 591–600. ACM, 2006.

[11] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*, pages 133–144. ACM, 2009.

[12] H. Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. *Advances in Cryptology–CRYPTO 2010*, pages 631–648, 2010.

[13] Chris Robison, Scott Ruoti, Timothy W van der Horst, and Kent E Seamons. Private facebook chat. In *2012 ASE/IEEE International Conference on Social Computing (SocialCom) and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT)*, pages 451–460. IEEE, 2012.

[14] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin / Heidelberg, 1985. 10.1007/3-540-39568-7_5.

[15] S. Sheng, L. Broderick, CA Koranda, and JJ Hyland. Why johnny still can't encrypt: evaluating the usability of email encryption software. In *2006 Symposium On Usable Privacy and Security - Poster Session*, 2006.

[16] Andreas Sotirakopoulos, Kirstie Hawkey, and Konstantin Beznosov. "i did it because i trusted you": Challenges with the study environment biasing participant behaviours. In *SOUPS Usable Security Experiment Reports (USER) Workshop*, 2010.

[17] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. What makes users refuse web single sign-on?: an empirical investigation of openid. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, SOUPS '11, pages 4:1–4:20, Pittsburgh, Pennsylvania, 2011. ACM.

[18] Timothy van der Horst and Kent Seamons. Simple authentication for the web. In *Security and Privacy in Communication Networks*, September 2007.

[19] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *8th USENIX Security Symposium*, 1999.

[5] `http://www.measuringusability.com/usability-loyalty.php`

APPENDIX

A. PWM USER STUDIES

A.1 Demographic Questionnaire

Are you a student?
What is your major?
What is your occupation?
What is your gender?
What is your approximate age?
How long have you been a Gmail user?
Approximately how often do you use Gmail?

A.2 Bookmarklet Tasks

Introduction

Thank you for your participation. During this study, you will be asked to perform certain tasks using Gmail and then provide feedback to help us improve our software. During the course of this study, all acts taking place on the screen will be recorded along with audio of anything we discuss. This will help us learn whether or not our software is easy to use. None of the video or audio content captured during the study will be released publicly or given to a third party. Before beginning the study, we will also ask you to provide some demographic information. None of the results published as part of this research will personally identify you as a participant.

You will have access to a temporary Gmail account for use in completing tasks during this study. You will not be asked to use your own Gmail login name or password at any time. Do not enter or access any of your own personal data during the study since everything on the screen will be recorded.

You will receive $10.00 as compensation for your participation in this study. The expected time commitment is 20-30 minutes. If you feel uncomfortable with any aspect of this study you may quit at any time.

Please advance to the next screen when ready.

Task 1

Please login to our test Gmail account with the login name and password shown below. Read the first message and follow the instructions given in the message. Close Gmail when you are finished and advance to the next page of instructions.

Click here to open Gmail
Username: pwmstudy@Gmail.com
Password: pwmusability

Task 2

Please log back into our test Gmail account with the user name and password shown below. Send a secure message to Gmailstudy@isrl.cs.byu.edu using Pwm. Include the ID number you were given in your message. Wait for a reply with further instructions.

Click here to open Gmail
Username: pwmstudy@Gmail.com
Password: pwmusability

A.3 Bookmarklet Post-study Survey

SUS Questions:

Please answer the following question about Pwm. 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.

Choose from 1 (strongly disagree) to 5 (strongly agree).

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system

7. I found the system very cumbersome to use
8. I would imagine that most people would learn to use this system very quickly
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

Remaining Questions:
Please give your response to the following general statements. 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.
Choose from 1 (strongly disagree) to 5 (strongly agree).

1. I trust Gmail with my sensitive email messages
2. I am concerned about Gmail scanning my messages
3. I feel safe sending important information through email
4. I worry that some messages aren't really from who they say they are from
5. I found the bookmarklet easy to use (The button you dragged to your toolbar is called a bookmarklet)

Have you used a bookmarklet before this study?
What did you like about Pwm?
What did you dislike about Pwm and how would you like it to be changed?
Have you ever been asked to send sensitive information you were not comfortable sending through email?
What type of sensitive information were you asked to send?
Did you send the requested information?
Have you ever received information you were not comfortable receiving through email?
What type of sensitive information did you receive?
If you started using Pwm on your own, would you prefer protection for new messages to be? *Choose one: Always on; Only on for the message that was open when you clicked "Secure my Email"; Off, unless I click a separate button on the Gmail page*

A.4 Extension Post-study Survey

Please answer the following question about Pwm. 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.
Choose from 1 (strongly disagree) to 5 (strongly agree).
Same SUS questions from A.3.

What did you like about Pwm?
What did you dislike about Pwm and how would you like it to be changed?
Other comments on Pwm

A.5 Additional Post-study Questions

Please answer the following general statements. 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.
Choose from 1 (strongly disagree) to 5 (strongly agree).

1. I trust Gmail with my sensitive email messages
2. I am concerned about Gmail scanning my messages
3. I worry that some messages aren't really from who they say they are from
4. I feel safe sending important information through email
5. I feel safe creating accounts with usernames and passwords on new sites
6. I feel safe installing browser extensions or plugins
7. Creating accounts for new websites is easy
8. Installing browser extensions is easy
9. I feel safe clicking on links in email messages
10. I feel safe clicking on links in email messages from people I know
11. I never click on links in email messages

Have you installed browser extensions, add-ons or plugins before today?
What has prevented you from installing browser extensions, add-ons or plugins in the past?
When deciding whether you will trust a browser extension, add-on or plugin, what influences your decision?
Have you ever been asked to send sensitive information you were not comfortable sending through email?
What type of sensitive information were you asked to send?
Did you send the requested information?
Have you ever received information you were not comfortable receiving through email?
What type of sensitive information did you receive?

B. MESSAGE PROTECTOR USER STUDIES

B.1 Demographic questions

What is your age?
What is your gender?
What is your major?
How do you rate your level of computer expertise?
How often do you use webmail?
How often do you use Facebook?
Have you ever sent private or sensitive information via Web email or Facebook?
How did you send that information *Select all that apply: Web email; Facebook private message; Facebook wall post; Instant message; Other (please specify below):*
How important is maintaining the privacy of your messages containing sensitive information? *Very important; Important; Neither important nor unimportant; Unimportant; Very unimportant*
Have you ever encrypted an email or Facebook message?

B.2 Study Introduction

Purpose

The purpose of this study is to compare two Internet encryption systems, Message Protector (MP) and Encipher.

What to Expect

In the study, you will attempt a set of tasks that Internet users regularly perform. You will do each set of tasks twice, once with MP and once with Encipher. Following the completion of each set, you will complete a survey about your experience with the application. During this study, all actions taking place on the screen will be recorded along with audio content of anything we discuss, however we will not record video of you. This will help us to analyze our software's usability. None of the video or audio content recorded during the study will be released publicly or given to third parties. Prior to the study beginning you will complete a short survey about yourself. None of the results published as part of this research will personally identify you as a participant.

Introduction

MP and Encipher are programs that allow Internet users to encrypt text that they communicate through websites. In this study you will perform common Internet tasks with MP and Encipher. This study will take about 45 minutes. Try to perform each task as quickly and accurately as you can. If you get stuck at any point, please call the proctor for assistance. You will receive $10.00 as compensation for your participation in this study. If you feel uncomfortable with any aspect of this study, you may quit at any time. Thank you for participating!

B.3 Message Protector Tasks

Message Protector Tasks

Message Protector (MP) is a computer program that allows users to protect Internet messages (e.g., email, Facebook private messages) via encryption. In this portion of the study, you will execute various tasks that comprise the primary functionality of MP and answer a few related questions.

Scenario 1: Installation

In this scenario, you will install MP on a computer. Please follow the instructions as closely as possible.

Scenario 1 Task 1: MP Installation

Access http://MessageProtector and follow the instructions in section 1 "Installing Message Protector."

MP requires an email address and the email account password to allow the user's contacts to be able to read their protected messages. For this study, we have created the following test account for you to use:

Email Address: userstudyMP@Gmail.com

Password: mpUserStud

Allow the following contacts to read your protected messages: randomFriend@hotmail.com, mom@familyWebsite.com, recipientMP@Gmail.com, stalwartStudent@byu.edu

Scenario 2: Gmail
In this scenario, you will encrypt and decrypt email messages with MP. Open Chrome and click the Gmail bookmark on the Favorites bar. A test account will already be logged in.

Scenario 2 Task 1: MP Email Encryption
Access http://MessageProtector and follow the instructions in section 2 “Encrypting Messages” to send an email to recipientMP@Gmail.com. Include the phrase “The last four digits of my SSN is 6789” in the message.

Scenario 2 Task 2: MP Email Decryption
After completing the previous task, you will receive a protected reply email from recipientMP@Gmail.com. Access http://MessageProtector and follow the instructions in section 3 “Decrypt Message” to decrypt the protected message.

Type the decrypted message below:

Scenario 3: Facebook Private Message
In this scenario, you will encrypt and decrypt Facebook private messages with MP. Open Chrome and click the Facebook bookmark on the Favorites bar. A test account will already be logged in.

Scenario 3 Task 1: MP Private Message Encryption
Access http://MessageProtector and follow the instructions in section 2 “Encrypting Messages” to send an encrypted Facebook private message to the user study account’s friend named “Alice Jones.” Include the phrase “My bank account password is cougars” in the message.

Scenario 3 Task 2: MP Private Message Decryption
After completing the previous task, you will receive a reply private message from “Alice Jones”. Access http://MessageProtector and follow the instructions in section 3 “Decrypting Messages” to decrypt the protected message.
Type the decrypted message below:

Finished
This concludes the Message Protector portion of the study. Please answer the questions below about your experience. Please record your immediate response to each question. If you feel that you cannot respond to a particular question, please mark the center point of the scale.

B.4 Message Protector Post-study Survey

Please answer the following question about Voltage. 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.
Choose from 1 (strongly disagree) to 5 (strongly agree).
Same SUS questions from A.3.
My level of understanding of MP directly affects whether I would use it to protect my email and Facebook messages.

Who can read messages that you protect with MP? *Choose one: Anyone that has MP installed, receives the message, and that I have selected to communicate with securely; Anyone who receives the message and who I have selected to communicate with securely; Anyone who receives the message; Anyone who has MP installed; I don’t know*
After MP is installed, what actions must recipients take to read MP protected messages? *Choose one: Access the MP website; Copy the message and paste to MP; Copy the message, paste to MP, click the Encrypt button; Copy the message, paste to MP, click the Decrypt button; I don’t know*
How often would you use MP to protect your email and Facebook messages? *Choose one: Always; Very Often; Occasionally; Rarely; Very Rarely; Never*
What did you like about MP?
How could MP be improved *Select all that apply: Provide better operating instructions; Provide more information about MP to the user; Provide an easier way to select trusted contacts; Provide a more intuitive user interface; Provide a less intrusive or cumbersome experience; Other (please specify below):*

B.5 Additional Post-study Survey Questions

Please answer the following question about Voltage. 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.

Choose from 1 (strongly disagree) to 5 (strongly agree).
I feel that it is important to encrypt my emails and Facebook messages that contain sensitive or private information
I would use a different Internet Encryption tool for every website that I store or share sensitive information
I trust Gmail employees to not disclose, misuse, or abuse my email and Facebook messages
I trust Facebook employees to not disclose, misuse, or abuse my email and Facebook messages
I would trust a company other than Facebook or Gmail (i.e., Encipher, MP) to protect my email and Facebook messages

Which system would you prefer to use? *Choose one: Message Protector; Encipher; Both; Neither*
Please explain your answer to the previous question:

B.6 Survey Responses

Table 1: Encipher.it Comparison Study Results

Preferred System	Reason
Message Protector	I had trouble decrypting the messages when using Encipher.
Encipher	I felt I understood how Encipher works more clearly. Also, I liked having the bookmark tab available.
Message Protector	The program was easy to use and did not require any kind of key.
Neither	I felt that anyone with Message Protector would decipher my emails so there is no point in encrypting them. / Encipher was a little clunky. I don't understand how I would set up a passcode and how the receiver would know what it was / Plus I don't send private information that frequently.
Neither	Both were too complicated. I probably wouldn't use either because it takes too much time and i would just take the risk of me getting my stuff stolen
Encipher	Encipher is easier to use even though the person receiving the message has to have the encryption key.
Message Protector	Encipher didn't work either time and was slow. I think it would be easier than message protector though as it is already integrated into the website and i dont have to leave the page I am on. So I like the idea of encipher better but since it didn't work for me, I am biased to MP
Message Protector	It felt safer. Encipher just felt like a pop-up that I should block and I wasn't sure why it was safe or how I would get the security key thing or give it to the people I would want to see my private message.
Message Protector	Message protector is most easy to use i like it!!!
Message Protector	IT is easy to use
Message Protector	I have a hard time remembering passwords, and I don't really understand how I could send a password privately to the recipient of an encrypted message via Encipher it. Message Protector was simpler for me to understand and use.
Message Protector	I liked that I didn't need to specify an encryption key in Message Protector and my secure contacts were already recognized by the computer.
Message Protector	Unlike Encipher, it doesn't require a new key each time you want to encrypt or decrypt a message.
Encipher	I didn't have to load my contacts into it. You also need a decryption key from the sender, so you can't just decrypt it if you have the program, like Message Protector allows you to do.
Encipher	Encipher is already integrated into the internet so it's much more convenient to use. I would use message protector if I wasn't in an HTPPS website.
Message Protector	Just was eaiser to use. No password needed, and I felt that it was more secure.
Encipher	Encipher is a quick and easy tab that requires everytime a new password that can be complex. That's it. you use the tab, you use a password. that simple. MP requires installing something and going from window to window, and someone can decipher your messages with your email and email password, or your contact's email or email password. Most people don't have that complex passwords and so I would be concerned a bit with Personal Information. Encipher uses a whole new level of protection.
Encipher	I think that Encipher is better because it has you pick your secure contact every time. With the MP, you could accidentally send sensitive info to one of your 'safe' contacts, but not the one that you wanted to see that info. Encipher is more user friendly.

Continued on next page

Table 1 – continued from previous page

Preferred System	Reason
Encipher	Even though Encipher requires a decryption key, it doesn't require pasting your message into a separate window.
Message Protector	
Message Protector	Easier and cleaner experience. It didn't take me even half the time to figure out how to use this as it did with Encipher.
Both	They both are nice. I like the toolbar aspect of Encipher, but did not like that the encryption key did not work for me on the first task.
Encipher	I can send protected messages to all my friends who have the key but not only with some limited contacts.
Both	If I have to send some private information over the internet, it's most likely that I would do so with my father who is not very good at using computer. Encipher would be easier for him to start with. If he gets used to it, I would switch to Message Protector. Also, I can use Encipher at any computer with the least effort-just adding to favorites. It's extremely convenient. However with other people who are good at computer, I would use Message Protector right away. And if I have to send a very important message, I would use Message Protector, since I feel like it's a more secured program.
Message Protector	I don't know if it was me, but I couldn't make the Encipher program decrypt the messages. However, I worked with Message Protector much better and I was able to decrypt the message. However, I feel Encipher has a better idea in just simply typing in a password instead of copying and pasting. If Encrypt would've worked for me I would've liked it more because of the simplicity.
Encipher	more protection. I could forget who I selected from my contacts if I use message protector.
Both	Enchipher was quicker and easier, but Message Protector was easier to do without contacting the sender/recipient for a password which I feel is a plus.
Both	With important sensitive information, such as my SSN or Bank account number and password, I would use MP because the fact alone that it requires a download and then a separate window from the conversation makes it feel more secure, although it probably has no actual security difference. With more routine information, such as the fact that I can't stand my untrustworthy boss, I would use Encipher because it is fast and conveniently located in the bookmarks bar, rather than requiring that I open a program I have saved somewhere on my computer. / / On top of this, if I were to have a long conversation, all of which I would like to have encrypted, I would use MP because it requires less time to function and to carry out operations. However for a quick encrypted message, I would use Encipher becasue it is more convenient than opening a separate program.

Table 2: Pwm Comparison Study Results

Preferred System	Reason
Both	I like the encryption for the Message Protector, but it is not as convenient as Pwm since it is right in Gmail.
Message Protector	The reason i would prefer Message Protector is because it seemed more reliable and safe. There seamed to be a more secure connection between you and the recipient due to the fact that you had to add the secure email. I'm not sure how it works and that not just any one can decrypt your message with the same software. I think that PWM is defiantly easier to use but doesn't seem as secure. If i am wanting to encrypt some content I would take a little but longer to make sure that it is safe
Both	They both seemed effective and useful. I would use the system that others are using.
Pwm	Pwm was integrated into Gmail which I use and many of my friends as well.
Pwm	I liked MP, but Pwm being directly in the email makes it more user friendly and less of a hassle.

Continued on next page

Table 2 – continued from previous page

Preferred System	Reason
Message Protector	Message Protector gives me a reason to believe that my message was actually encrypted. Pwm, on the other hand, was very easy to use, but seemed almost too easy. I can still read my message after encrypting, which makes me think that perhaps it wasn't actually encrypted.
Pwm	I would use Pwm simply because it's an extension easily integrated into Gmail. It required little customization, just a simple click of the button. However, I felt a little more confident that I was using MP correctly and that it was encrypting my messages. My choice is mostly out of design and convenience, trusting that both programs do the task effectively.
Pwm	Mainly because I don't have to have to separate windows to encrypt/decrypt stuff. It just seems less of a hassle when it's built into the Gmail system.
Message Protector	Although MP was a little (not by much) more difficult to use, I can be certain of who is able to view the encrypted data.
Pwm	I found it faster and easier for the program to encrypt and decrypt messages for me than copying and pasting it myself.
Pwm	It was simpler, and easier to use.
Pwm	Much more convenient. MP is too much of a hassle, although if it was more secure than Pwm, I might use it, but I would still use it much less than Pwm.
Pwm	Pwm was much easier to use and the instructions were easier to understand.
Both	Depending on who I was corresponding with, I may switch between encrypting programs. It seems useful, but it may require the other party to have some decrypting program installed, which is not very convenient for banks or places that I might be sending messages to.
Both	I would use both because I am not very educated on either of them. I would want to use both to see which would become more comfortable for me. When I understood more about them I could then decide which was more effective for me.
Pwm	I like Pwm's integration right into the browser. The only fault is that you only see the encryption after it has been sent, so if it fails, your information isn't encrypted.
Message Protector	I found message protector to be a little easier to understand
Message Protector	It was easier to use and had much better instructions.
Message Protector	See the reasons stated before for using MP - offline encryption, more intuitive, etc
Message Protector	Even though it is slightly more difficult to use, it seemed more secure. I also liked that you could choose what contacts could communicate with.
Both	Pwm is much easier to use.... but Message Protector seems like it might be safer in only letting certain people see it? DEfinitely prefer pwm if it's just as secure.
Pwm	I would prefer Pwm, because you don't have to copy and paste your message, then press the encrypt or decrypt button, as you need to do with MP. I think that it is easier just to have pwm enabled, which is a lot faster and smoother, and it just protects your message for you, without needing to encrypt the message manually.
Message Protector	It's much easier to use, and makes more sense than Pwm. I like that you can see all the steps of what you're doing, so you feel more in control of the process.
Message Protector	Pwm didn't work for me.
Message Protector	MP had a simpler design and was much more user friendly. (i felt like i was in a catch-22 with Pwm.)
Pwm	PWM requires fewer steps and was less complicated to me.
Message Protector	I feel that Pwm, as a Gmail extension would be easy for anyone to get. If I were to send sensitive information to the wrong address, to my understanding the could simply install the extension and view it. With MP, they not only need a 2nd party program installed, (one not as easily located) and I would need to have them on my selected contact list. It feels safer.
Message Protector	Though it was slightly more complex in the set up I personally found it more easy to use. I like that I could see that it was encrypted.
Pwm	It's easier than Message protector.

C. BANGOR'S SUS SCALE

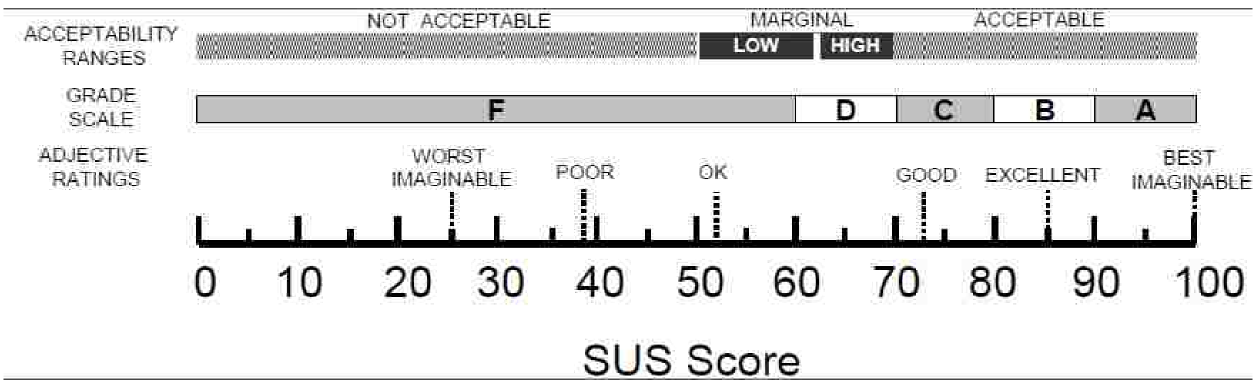


Figure 1: An adjective oriented interpretation of SUS scores [2]

Private Webmail 2.0: Simple and Easy-to-Use Secure Email

Scott Ruoti[†*], **Jeff Andersen**[†], **Travis Hendershot**[†],
Daniel Zappala[†], **Kent Seamons**[†]
Brigham Young University[†], Sandia National Laboratories[*]
{ruoti, andersen, hendershot} @ isrl.byu.edu, {zappala, seamons} @ cs.byu.edu

ABSTRACT

Private Webmail 2.0 (Pwm 2.0) improves upon the current state of the art by increasing the usability and practical security of secure email for ordinary users. More users are able to send and receive encrypted emails without mistakenly revealing sensitive information. In this paper we describe user interface traits that positively affect the usability and security of Pwm 2.0: (1) an artificial delay to encryption that enhances user confidence in Pwm 2.0 while simultaneously instructing users on who can read their encrypted messages; (2) a modified composition interface that helps protect users from mistakenly sending sensitive information in the clear; (3) an annotated secure email composition interface that instructs users on how to correctly use secure email; and (4) inline, context-sensitive tutorials, which improved view rates for tutorials from less than 10% in earlier systems to over 90% for Pwm 2.0. In a user study involving 51 participants we validate these interface modifications, and also show that the use of manual encryption has no effect on usability or security.

Author Keywords

Security, usability, secure email, encryption

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces—*user-centered design*; H.1.2. Models and Principles: User/Machine Systems—*human factors*

[*]Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000

INTRODUCTION

Recent measurements demonstrate that email is largely insecure [8, 6, 11]. Although adoption of connection-based encryption and authentication is increasing (i.e., STARTTLS, SPF, DKIM, DMARC), these technologies are often configured incorrectly and have weaknesses that render them susceptible to attacks. However, even if transmission of email was properly secured, email at rest is generally unencrypted, leaving email open to scanning by an honest-but-curious email provider or by a government that coerces an email provider into allowing access. End-to-end encryption provides email with much stronger protection. With end-to-end encryption, the sender encrypts email messages before transferring them to the email provider, ensuring that no one but the intended recipients is able to read the messages.

Unfortunately, usable secure email is still an open problem more than 15 years after Whitten and Tygar's seminal paper, *Why Johnny Can't Encrypt* [26].[1] Arguably the most secure form of end-to-end encryption is PGP, but PGP has continually been shown to be unusable [26, 21, 15]. More recently, research has shown that by relaxing the security requirements of PGP, it is possible to create usable, secure email [10, 16, 1]. While these newer systems have lower *theoretical* security than PGP, they have higher *practical* security—i.e., using these new tools, ordinary users are able to encrypt their email, something that they are largely unable to do with PGP-based tools.

Of these usable, secure email systems, Private Webmail (Pwm) is specifically targeted at helping the masses encrypt their existing webmail accounts, such as those provided by Google, Yahoo, and Microsoft. A set of user studies of Pwm demonstrated that it had significantly higher usability than similar secure webmail systems (Encipher.it and Voltage Mail [16]). Still, in this evaluation of Pwm about a tenth of participants accidentally sent email in the clear when they meant to encrypt it, and many

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

more users were unsure of the security provided by Pwm.

In this paper, we describe an improved interface design for Pwm that addresses issues raised in the original studies. First, we added an artificial delay to encryption that enhances user's confidence in the strength of the message encryption while simultaneously instructing users on who can read encrypted messages. Second, we modified the email composition interfaces to better ensure that users understand which emails are encrypted and help them avoid mistakenly sending plaintext in the clear. Third, we annotated the encrypted email composition interface to help users understand how to correctly use secure email. Finally, we implemented inline, context-sensitive tutorials, which improved view rates for tutorials from less than 10% for Pwm to over 90% in some cases for Pwm 2.0. Through a user study involving 51 participants, we demonstrate that our new design, referred to as Pwm 2.0, is significantly more usable and secure than Pwm.

Our main contributions are as follows:

1. We describe the interface modifications we used to build Pwm 2.0. Our user study demonstrates that these modifications had a significant impact on the usability and security of Pwm 2.0, as compared to the original Pwm system. Moreover, we discuss user reactions to these interface modifications and lessons learned from our study.

2. Pwm 2.0 scores an 80.0 on the System Usability Scale (SUS), rating in the "excellent" category for usability and receiving an "A" grade. This score is in the 87^{th} percentile for system usability [18] and is the highest SUS score for secure email to date. Moreover, over 80% of users wanted to immediately begin using Pwm 2.0, and over 90% felt that their friends and family could use Pwm 2.0 with relative ease.

3. We show that with Pwm 2.0 users rarely send plaintext messages when they meant to encrypt the message. Out of 306 tasks, only six mistakes were made using Pwm 2.0 (2%), a significant improvement to Pwm's rate of about 10%. In addition, the majority of users understood the confidentiality, authenticity, and integrity Pwm 2.0 provided their messages (84%, 63%, 76%, respectively).

4. As part of our study, we split users into two groups: one that used an automatic-encryption version of Pwm 2.0, and one that used a manual-encryption version of Pwm 2.0. Our results demonstrate that, contrary to the findings of the original Pwm study [16], requiring a user to view their encrypted message before sending it (manual encryption) has no benefit over encrypting and sending the message in one step (automatic encryption).

BACKGROUND

In this section, we review related work for secure email and automatic vs. manual encryption. We then describe the threat model that motivates our work. Finally, we describe the original Private Webmail system.

Related Work

Whitten and Tygar [26] conducted the first formal user study of a secure email system (i.e., PGP 5). They found serious usability issues with key management and users' understanding of the underlying public key cryptography. Subsequent studies of newer PGP-based clients have shown that PGP tools are still unusable for the masses [21, 15].

Garfinkel et al. [10] conducted a usability study of secure email based on S/MIME, and observed that hiding encryption details can impact usability. For example, they found that because the integration of encryption into Outlook Express "was a little too transparent," users were often unsure whether a given email was encrypted. Additionally, they found that some users failed to read the instructions associated with various visual indicators, leading to difficulties in understanding the interface. Still, they found that automating key management increased the overall usability of secure email.

Fahl et al. [7] explored manual and automatic encryption in the design of a Facebook Chat system. Here, automatic encryption refers to encrypting and sending a message without explicitly demonstrating that encryption has occurred, whereas manual encryption shows the user ciphertext and requires them to manually send this ciphertext. Their user study demonstrated that users reported preferring automatic key management, but found no significant difference between automatic and manual encryption. Nevertheless, they raised the issue that automatic encryption could impact users' feelings of trust and recommended the issue be addressed in future work.

Ruoti et al. [16] conducted a series of user studies with Private Webmail (Pwm), a secure email prototype that tightly integrates with Gmail. Even though results showed the system to be quite usable, they found that some users made mistakes and were hesitant to trust the system since the automatic encryption was too transparent. They also conducted a study comparing Pwm to a desktop application (MP) that supported manual encryption. They found that participants using MP's manual encryption made fewer mistakes and answered more questions correctly on a quiz that tested their understanding of the system. However, there are significant differences between Pwm and MP, which are confounding factors in Ruoti et al.'s results. In this paper, we also examine the differences between automatic and manual encryption, but do so using two variants of our Pwm 2.0 system that only differ in regards to the use of automatic and manual encryption.

Atwater et al. [1] have also examined the question of automatic and manual encryption brought up by Ruoti et al.'s study. They created two version of a tool based on Mailvelope, and these versions only differed in their usage of manual and automatic encryption. They found that manual encryption had no statistically significant effect on participants' trust of their secure email system. Their work differs from ours in that they examined the question of trust whereas we are looking at understanding and mistakes, the two pivotal points from the original Pwm study. In addition, though Atwater et al. demonstrate usability with a PGP-based client, the system they simulated does not include several practical key management steps that users would be required to perform, limiting the applicability of its results.

Threat Model

In our threat model there are four entities:

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

2. **Webmail provider.** We consider the webmail provider as an honest-but-curious party.[2] The webmail provider has access to the users' encrypted messages, but not to the keys that encrypt those messages.

3. **IBE Key server.** Key management in Pwm uses Identity-Based Encryption (IBE) [20]. In IBE, a user's public key can be computed using public parameters from the IBE key server and the user's identity (e.g., email address). To retrieve the private key for a given identity, a user proves ownership of the appropriate identity, and then the key server will generate and send the private key to the user.[3]

   We consider the IBE key server as an honest-but-curious party. While the key server is responsible for generating a user's private key, it does not have access to the messages encrypted with those keys.

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

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

To steal the user's sensitive data, the adversary must obtain both the encrypted email and the key material needed to decrypt the email. The former can be accomplished by either compromising the webmail provider, or intercepting an encrypted email that is not transmitted using TLS. The latter can be accomplished by compromising the IBE key server. Just as the adversary must collect the data from both the webmail provider and key server, neither of these parties alone has enough information to unilaterally steal the user's sensitive data. While as honest-but-curious parties, these two entities will not collude, a government could subpoena both organizations and compromise a user's sensitive data. If this is a significant concern, it is possible to apply a thresholding scheme to the IBE key server and place the various key servers in different localities that for political reasons will not cooperate.[5]

While our threat model is more permissive than the traditional PGP threat model, it is nonetheless useful in a variety of situations. For example, it satisfies the typical case where a small business or a university has outsourced its email to a third-party service provider such as Google, but does not trust Google with sensitive data. The business or university does, however, trust a browser extension that it has vetted and a key server that it operates. Alternatively, our threat model allows every-day users

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

[3] Private keys are not stored on the key server, but rather generated on demand based on a master secret. For added security, this secret can be stored inside of a crypto-card.

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

[5] Alternatively, the IBE key server could roll over its keys on a regular (e.g., daily) basis, limiting the amount of time an adversary or a government has to steal the user's sensitive data.

Figure 1. The read interface for an encrypted email.

to easily encrypt sensitive email, preventing their webmail providers from storing, scanning, or accidentally leaking sensitive information. Regardless, the model provides greater assurance than currently available through vanilla SMTP [6].

Private Webmail (Pwm)

Pwm implements secure email through tight integration with Gmail and leverages automatic key management. When users read or compose secure email, Pwm replaces portions of Gmail's interface with Pwm's own secure interface. Pwm's interfaces are styled differently than Gmail's, providing a clear demarcation of which information is being protected by Pwm. Pwm's interfaces are implemented using *security overlays* [25]. Security overlays allow Pwm's interfaces to be visually integrated within Gmail's interface, while still protecting the content of Pwm's overlays from Gmail.[6]

All secure email sent using Pwm includes instructions on how to setup Pwm for first-time users. The setup instructions direct new users to the Pwm website, where they are able to add a bookmarklet to their browser's bookmark storage.[7] The new user then returns to Gmail and clicks on the Pwm bookmarklet to run Pwm. The benefits of using a bookmarklet to run Pwm include not requiring installation permission on the machine and also avoiding any user worries related to installing extensions [16]. The drawback is that users are required to click on the Pwm bookmarklet each time they reopen Gmail.

When Pwm is running, secure email messages are automatically decrypted for users. The decrypted contents of the message are displayed in place of the instructions and ciphertext that were in the email message's body (Figure 1).[8] Encrypted email messages in the user's inbox are marked as *Encrypted* (Figure 2). Pwm allows users to add encryption on a per-message basis by clicking a lock icon inserted next to the "Send" button at the bottom of the compose interface.

In Pwm, users are authenticated to the IBE key server using Simple Authentication for the Web (SAW) [24], a form of email-based authentication [9]. Since Pwm runs inside the user's Gmail session, it is able to complete authentication automatically and does not need to prompt the user for input. It should be noted that because Pwm authenticates to the IBE key server using email, if an adversary were to compromise the user's webmail account they would also be able to authenticate to the key server. This means that messages encrypted with Pwm are only as secure as the user's webmail account. Ruoti et al. took this approach in order to maximize usability and because it still provides greater security than currently available to webmail users [8, 6, 11]. If greater security is needed, the IBE key server could be modified to authenticate users by some other means (e.g., separate password, two-factor authentication).

USABILITY IMPROVEMENTS

We used an iterative design methodology to address problems with Pwm's original interface. For each problem, we brainstormed several potential solutions and evaluated each solution using cognitive walkthroughs and heuristic evaluations. The most promising solutions were then implemented in prototypes, and these prototypes were evaluated with pilot usability studies conducted with a convenience sample from friends, family, and students. Those solutions that were rated as helpful in our pilot usability studies were then implemented in our Pwm 2.0 system. The source for Pwm 2.0 can be found at `https://bitbucket.org/isrlemail/pwm`.

In the remainder of this section, we describe the most relevant solutions that were added to Pwm 2.0.

Delayed Encryption

One concern expressed by a significant portion of participants in the original Pwm study [16] was that Pwm encrypted email so quickly that participants were unsure if Pwm had really done anything. Also, participants indicated the encryption process was so invisible that they were unsure who could read their encrypted email message.

To address both of these problems we added an artificial delay after users click the *Send encrypted* or *Encrypt* buttons. For each email recipient, users are shown a message lasting 1.5 seconds that states the email is being encrypted for that user (e.g., "Encrypting for bob@gmail.com"). This helps users understand who will be able to read the encrypted

[6]This prevents Gmail from being curious and scanning the contents of encrypted emails.

[7]A bookmarklet is a browser bookmark that contains executable JavaScript. This JavaScript is run on the page that users are viewing when the bookmark is clicked.

[8]Screenshots in this section are of Pwm 2.0.

Figure 2. An encrypted email in the inbox

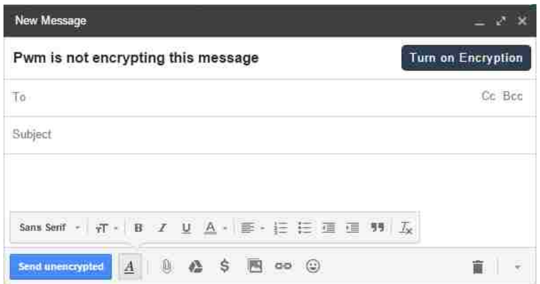


Figure 3. Gmail compose interface before enabling encryption

email and also lets them feel that something substantial has happened during the encryption process.[9] Because most email messages have a small number of recipients, this artificial delay does not significantly impact the overall email experience.

Compose Interface

In the original studies, about 10% of users accidentally sent a sensitive email without first encrypting it. In each case, the user recognized that they had made a mistake immediately after hitting the send button, but unfortunately their sensitive information had already been transmitted in the clear. Moreover, many users initially composed their sensitive emails in plaintext, only encrypting them right before sending the message, allowing Gmail to scan their message as it was being typed.

To address both of these issues, we revised Pwm 2.0's compose interface to better conform to the flow of composing email messages in Gmail (i.e., moving from top to bottom). As part of this process, we added informative text at the top of the compose interface (before the data entry sections) that indicates to users whether their message is being encrypted, then allows them to enable encryption before they begin composing their message (Figure 3). When encryption is enabled, we modify the informative text to make it clear that the message is now being encrypted (Figure 4). Furthermore, we modified Gmail's *Send* button to read *Send unencrypted*, to make it clear to users that by default their messages are sent without encryption.

To help users better understand how Pwm 2.0 protects their emails, we modified the *placeholder* text for the `Recipients` and `Subject` fields, explaining how Pwm 2.0 uses these two fields (Figure 4). Also, we added functionality allowing users to compose

[9]Future work could examine the effect of removing this delay for experienced users who already have a correct understanding of the system's functionality.

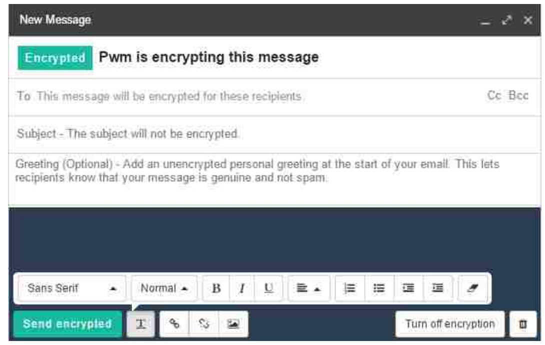


Figure 4. Pwm's compose interface

Figure 5. Style of the tutorial window

plaintext greetings that are included with the encrypted email (Figure 1 and Figure 4).[10] The greeting can help engender trust in a new user that receives an encrypted email, giving them confidence to setup and use Pwm 2.0. It also provides context for encrypted email messages before they are decrypted.

Tutorials

Several of the problems encountered by participants in the original studies could have been avoided if they had viewed the included video tutorial. As participants were clearly uninterested in reading the instructions or watching a video on the Pwm website, we replaced these with integrated, context-sensitive tutorials in Pwm 2.0. These tutorials provide step-by-step instructions on how to use Pwm 2.0, are displayed the first time a user performs a new action, and are shown directly to the side of the interface the user is currently using. Each step in the tutorials uses simple language and instructs the user about a single feature of Pwm 2.0 (Figure 5). Some tutorial steps require explicit action from users before they can move on to the next step, helping users internalize correct behavior (Figure 6). The tutorials are as follows:

1. **Introduction.** This tutorial is shown to users the first time they run Pwm 2.0. It informs users that Pwm 2.0 will help protect their email and

[10]This feature was first suggested by Ruoti et al. [16], but was not actually implemented.

Figure 6. Tutorial waiting for action from the user

tells them how to identify whether Pwm 2.0 is running.

2. **Read.** The first time users read an encrypted email message, they are shown this tutorial. The tutorial shows users how to identify an encrypted email, explains the plaintext greeting, and identifies which portions of the email message were encrypted. Additionally, it informs them that email messages encrypted with Pwm 2.0 are provided authenticity and confidentiality (even from Gmail).

3. **Compose.** The first time users compose an email message while Pwm 2.0 is running, they are given the option to watch a tutorial describing how to compose encrypted email. This tutorial teaches users the correct order of operations for composing an encrypted email. It also informs them about who can read their message, the purpose of the optional greeting, and where to type sensitive information.

Look and Feel

We designed a new website for helping users install Pwm 2.0.[11] This website has a more professional look and feel than the original Pwm website. This change was made in reaction to user comments that they decide how much they trust a tool based on how professional the tool's website looks. To make Pwm 2.0 also feel more professional, we standardized its look and feel to use the same colors and styles as the website.

Automatic and Manual Encryption

The original version of Pwm automatically encrypts and sends email as soon as the user clicks *Send encrypted* (Figure 4). We created a manual encryption version that splits this operation into two distinct steps. In the first step, instead of clicking the *Send encrypted* button, the user instead clicks the *Encrypt* button. Upon doing so, the user's email message is encrypted, and both the instructions for decrypting the email message and the encrypted ciphertext are shown to the user inside Gmail's compose interface (Figure 7). This allows the user to confirm that encryption has actually taken place. In the second step, the user then clicks Gmail's *Send* button to send the encrypted email.

METHODOLOGY

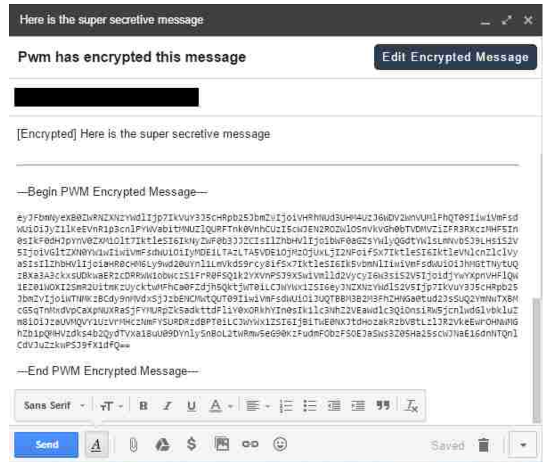


Figure 7. Ciphertext shown to users after manual encryption

This section gives an overview of our IRB-approved user study that evaluated Pwm 2.0. This study was also used to run a between-subjects A/B test comparing automatic and manual encryption. The data from this study is available at `https://uist2016.isrl.byu.edu`.

We were careful to design the study to test a generic secure email system, rather than specifically testing Pwm 2.0. This approach helps us to feel more confident that participants were interacting with Pwm 2.0 in a realistic fashion. To help with this goal, we included researchers who are not involved in our secure email research in designing the scenarios and tasks. After creating scenarios and tasks that were acceptable to all parties, we then conducted a pilot study with three participants. We did not identify any significant flaws during the pilot study.

Study Setup

The study ran for two weeks (February 17, 2015 through March 3, 2015) and included 52 participants that were randomly assigned to test either the automatic or manual encryption version of Pwm 2.0.[12] Participants took 30 to 60 minutes to complete the study and were compensated $10 USD for their efforts.

Studies were conducted in a room dedicated for this study. When participants first entered the room, they were read a brief introduction to the study. For the remainder of the study, all instructions

[11] `https://pwm.byu.edu`

[12] Our power analysis ($\alpha = 0.05$, $\beta = 0.80$) indicated that we would need 23 participants in each treatment group to reasonably detect differences in SUS scores and task completion time. In contrast, our power analysis revealed that detecting even medium sized (10%) differences in our proportional measures would require 387 participants in each treatment group (far beyond our group's ability to recruit participants). As such, the proportional measures should only be evaluated as trends, and not as statistically significant.

were written and provided to them, either via a printed information sheet or via email. Participants completed all tasks on a virtual machine, ensuring that the computer started in the exact same state for all participants. We also recorded participants' screens.

Two study coordinators were involved in the study. One coordinator sat in the same room as the participant, within the participant's peripheral vision. This coordinator was instructed to avoid prompting participants and to only assist participants if they had not made any progress within five minutes. The second coordinator sat in another room and corresponded with the participant over email as part of the study tasks.

Quality Control

One participant was unable to use Pwm 2.0 with their Gmail account. This participant had a special Gmail configuration that was not compatible with our tool. While this participant did complete the study using a temporary email account we provided them, we were worried that their increased interactions with the study coordinator might bias their responses, and so we discarded their results. For the remainder of this work, we report results based on the 51 remaining participants.[13]

Participant Demographics

We recruited Gmail users for our study at a local university. Participants were evenly split between genders: male (25; 49%), female (26; 51%). Participants skewed young: 18–24 years old (45; 88%), 25–34 years old (3; 6%), 35–44 years old (2; 4%), 55 years or older (1; 2%).

We distributed posters broadly across campus to avoid biasing our results by any particular major. All participants were affiliated with the university,[14] with the overwhelming majority being undergraduate students: undergraduates (44; 86%), graduates (2; 4%), faculty and staff (5; 10%). Participants had a variety of majors, including both technical and non-technical majors. No major was represented by more than three participants, with the vast majority having only a single participant.

Most participants indicated they logged into Gmail on the web to check their email frequently: many times a day (39; 76%), once a day (3; 6%), 2–3 times a week (2; 4%), once a week (2; 4%), 2–3 times a month (2, 4%).

Scenario and Task Design

During the course of the study, participants were given two scenarios to complete: (1) being hired for a new job, and (2) sending credit card information to a spouse. Prior to beginning each scenario, participants were provided with a written description of the scenario. This description included information that participants should send in place of their own personal information.[15] Participants were asked to treat this sensitive information with the same care as they would treat their own information.

For each scenario we designed realistic tasks that used as many email features as possible. Additionally, we designed the tasks without considering Pwm 2.0's feature set, to avoid biasing the tasks to reflect favorably on Pwm 2.0. Participants completed tasks in the order shown below. Tasks were completed entirely using email, and participants used their own email accounts. If participants accidentally sent sensitive information without encryption they were notified of their mistake in an email and asked to resend the information using encryption.

Being hired for a new job

Participants were told that they had recently returned from an interview at National Citadel, a fictitious company created for this study.

Task 1. Participants received an email from National Citadel containing instructions on how to be reimbursed for expenses from their recent interview. Participants were told to send their Social Security number and a picture of their receipts. They were informed that this information must be encrypted as per National Citadel's policies. This email also instructed users to set up and use Pwm 2.0 to encrypt their email messages.

This task was designed to test whether participants were able to set up and use Pwm 2.0 using only instructions that might be reasonably expected from a company requesting information to be encrypted.[16]

Task 2. Participants were first asked to close their browser and then reopen Gmail. They were informed that this simulated several weeks passing after the completion of task 1. Participants were then sent an encrypted email that contained an offer letter. They were asked to reply with their acceptance. They were also asked to *CC* their new manager.

This task was designed to test whether participants would remember how to enable Pwm 2.0. It also tested whether they could use Pwm 2.0 to *CC* a new recipient.

[13]After analyzing the remaining data, we reviewed the removed participant's results and found that they were in line with the results we analyzed.

[14]We did not require this affiliation.

[15]We took this approach to avoid situations where sensitive information might have been leaked to Gmail.

[16]This task also shows that we designed the tasks around normal email usage and not Pwm 2.0. Pwm 2.0 does not support attachments and only allows for images to be inserted in line with the email body. This task could potentially be confusing to users as they cannot use their normal work-flow for attaching an image.

Task 3. Participants were sent an email instructing them to email information to a background check company. They were instructed to encrypt the information.

This task was designed to test whether users could enable encryption, either by forwarding the request for information or by composing a new email message.

Task 4. Participants were instructed to send bank account information to National Citadel's payroll department. They were *not* reminded to encrypt this information.

This task was designed to test whether users would remember to encrypt information if they were not explicitly prompted to do so. Unlike the preceding tasks, if they sent information without encryption, we still considered this task complete.

Sending credit card information to a spouse

Participants were told that their spouse had texted them asking for login information to a credit card website.

Task 5. Participants were instructed to send login information to their "spouse" using encrypted email. Participants were told that their spouse had never before used secure email, so they should take whatever steps they felt necessary to ensure their spouse could read the encrypted email.

This task was designed to see how participants would induct a new person into using secure email. Regardless of what instructions were sent, we considered this task complete when the information had been sent.[17]

Task 6. Participants received another email from their spouse asking them for additional credit card information. This request was *not* encrypted and did *not* instruct the participant to send the additional credit card information encrypted.

This task was designed to test whether users would remember to encrypt information if they were not explicitly prompted to do so. Unlike most of the preceding tasks, if they sent information without encryption, we still considered this task complete.

Using the video recording of participants' screens, we measured how long they took completing each task and how often they sent sensitive email without encryption (i.e., made a mistake).

Study Questionnaire

We administered our study using the Qualtrics web-based survey software. Before beginning the survey, participants were read an introduction by the study coordinator and asked to answer a set of demographic questions. Participants then completed the six study tasks, following which they were asked to complete a questionnaire regarding their experience.

To measure perceived usability, we had participants complete the ten System Usability Scale (SUS) questions [4, 5]. Answers to these questions are used to derive each version's SUS score, a single numeric score from 0, the least usable, to 100, the most usable, that provides a rough estimate of the version's overall usability. Recent research has shown that SUS scores are effective for comparing systems across different study populations and that SUS gives more reliable results than other usability metrics [23, 19, 17].

After completing the SUS questions, participants were asked several questions regarding whether they would want to use Pwm 2.0 in their day-to-day lives. We then asked participants questions to examine how well they understood the cryptographic properties of Pwm 2.0. To test understanding of confidentiality, they were asked to indicate which parties could read a message encrypted with Pwm 2.0. Similarly, participants were asked whether Pwm 2.0 provided authenticity and integrity for secure email messages composed with Pwm 2.0. Each question was asked in language that would be approachable to users and did not require a technical background. Participants were given the option to indicate that they were unsure whether a given property was provided by Pwm 2.0.

Limitations

While our study included students with a diverse set of majors and technical expertise, it may not be representative of non-student populations. Likewise, Gmail users may also not be representative of the general population's preferences regarding secure email. Our study is short-term and is not necessarily representative of how participants would use secure email over a longer period of time. Further studies could address these limitations.

Our study is a lab study and has limitations common to all studies run in a trusted environment [13, 22]. While there are indications that some participants treated the provided sensitive information as they would their own (e.g., refusing to email the provided social security number), there is no guarantee that participants' reactions mimic their real life behaviors. Additionally, our studies did not test participants' ability to resist attacks.

RESULTS

[17] Pwm 2.0 includes instructions by default, and since they were sufficient to help the participant start using Pwm 2.0, it was reasonable to assume that the participant believed they would be sufficient to help their spouse start using Pwm 2.0.

	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)
Automatic	27	79.1	9.6	±3.69
Manual	24	81.1	9.0	±3.60
Overall	51	80.0	9.2	±2.52
Original Pwm Study	72	74.2	13.7	±3.16

Table 1. SUS scores

In this section we report the quantitative results from our user study.

Usability

We evaluated Pwm 2.0 using the System Usability Scale (SUS). The automatic encryption version of Pwm 2.0 had a SUS score of 79.1 and the manual encryption version had a SUS score of 81.1. Further breakdown of the SUS scores, along with SUS scores from the original Pwm study [16], can be found in Table 1. The difference between manual encryption was not statistically significant (two tailed student t-test, unequal variance—$p = 0.43$). The difference between Pwm 2.0's overall SUS score (80.0) and Pwm's SUS score (74.2) is statistically significant (two tailed student t-test, unequal variance—$p < 0.01$).

To give greater context for Pwm 2.0's SUS score, we leverage the work of several researchers. Bangor et al. [3] analyzed 2,324 SUS surveys, and derived a set of acceptability ranges that describe whether a system with a given score is acceptable to users in terms of usability. Bangor et al. also associated specific SUS scores with adjective descriptions of the system's usability. Using this data, we generated ranges for these adjective ratings, such that a score is correlated with the adjective it is closest to in terms of standard deviations. Sauro et al. [18] also analyzed SUS scores from Bangor et al. [2], Tullis et al. [23], and their own data. They calculate the percentile values for SUS scores and assign letter grades based on percentile ranges.

Using these contextual clues, Pwm 2.0's SUS score of 80.0 is rated as having "excellent" usability and given an "A" grade. It is in the 87$^{\text{th}}$ percentile for system usability.

Task Completion Times

For each task, we calculated the average time between when the user could start a task and when they finished that task. These are shown in Table 2. For Tasks 2–6, and for the study as a whole, any differences between manual and automatic encryption were negligible. Task 1 had a statistically

	Count	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Total
Automatic	27	4:06	4:00	3:01	1:20	4:01	0:54	17:22
Manual	24	5:10	3:48	3:03	1:08	3:44	0:50	17:43
Overall	51	4:37	3:54	3:02	1:14	3:52	0:52	17:32

Table 2. Average task completion times (min:sec)

significant difference (two tailed student t-test, unequal variance—$p = 0.04$), but we caution against overemphasizing this result. First, while the difference is significant, the 95% confidence interval indicates that the actual effect size might be small (64 ± 60 seconds). Second, any differences in task completion times are amortized over the remaining tasks. Finally, a single statistically significant difference could arguably be the result of alpha inflation [19].

Understanding

We asked three questions to determine whether each participant understood the confidentiality, authenticity, and integrity properties provided by Pwm 2.0. The responses indicated whether each participant correctly understood the principle, had some misunderstanding, or was unsure. Overall, our data suggests that over 50% of users would likely understand the security provided by Pwm 2.0 (Adjusted-Wald binomial confidence interval [19], $\alpha = 0.05$—Confidentiality–83% ± 10%, Authenticity–62% ± 13%, Integrity–75% ± 12%). In all cases, the differences between manual and automatic encryption were not statistically significant.

The proportion of users who understood how Pwm 2.0 was protecting their email was much higher than the rate reported in the original Pwm study.[18] Still, it is interesting that even with tutorials that explicitly instruct participants on these three cryptographic properties, there are still a small number of participants who were unsure how their messages were protected. This indicates that more work needs to be done to determine how to effectively teach users about these properties.

Avoiding Mistakes

During the study, we recorded all instances of a participant taking an action that deviated from the study parameters. Using this data, we identified several instances where a participant's actions leaked sensitive information. These results are reported by task in Table 3.

[18] We do not calculate statistical significance for these two proportions as they were derived from different questions. Note that the understanding questions for Pwm 2.0 were stricter than those used in the original Pwm study, emphasizing how much better understanding in Pwm 2.0 is likely to be.

	Count	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Total
Automatic	27	0	0	1	0	0	0	1
Manual	24	1	0	0	1	0	0	2
Overall	51	3	0	2	1	0	0	6

Table 3. Number of participants who sent sensitive information without encryption

During the study, two participants sent sensitive information without ever running Pwm 2.0, and another refreshed the Gmail page but did not restart Pwm 2.0 before sending sensitive information. In all three cases, the mistakes were caused by the misuse of the tool and occurred before participants had ever seen any differences in the two versions of Pwm 2.0. For this reason, they are reported in the overall number of mistakes, but not under automatic or manual encryption. The differences between mistakes related to manual and automatic encryption are negligible.

Overall the rate of mistakes was extremely low (2%, 6 tasks out of 306). This is lower than the mistake rate reported in the original Pwm study of 10%, and the difference is statistically significant ($N-1$ two-proportion test for comparing two independent proportions–$p = 0.001$).

Tutorials

We recorded the number of participants that completed each tutorial. The video recording for one participant's session was corrupted, and we were unable to determine if they had completed the tutorials. The data from the remaining 50 participants indicate that participants completed a significant number of tutorials: completed introductory tutorial (46; 92%), completed tutorial on reading secure email (46; 92%), completed the tutorial on composing secure email (27; 54%). This is in stark contrast to the original Pwm study where nearly all participants ignored both the setup instructions and a video tutorial included on the Pwm website [16]. This result indicates that participants are willing to watch tutorials, though two criteria seem to be crucial: they appear in-page as participants need them and they contain simple and direct wording.[19]

Acceptability of Pwm 2.0

Overall, participants responded very positively to the prospect of using Pwm 2.0 in their own lives: 82% of participants agreed with the statement "I want to start using Pwm"[20] (81% $\pm$ 11%).[21] 73% of participants agree that "I would use Pwm with my friends and family" (71% $\pm$ 12%). 92% of participants agreed that "My friends and family could easily start using Pwm" (90% $\pm$ 8%). There were no significant differences between participants who used the automatic or manual versions of Pwm 2.0.

Due to the nature of the study, it is likely that participant responses were overly positive, and that the proportions are somewhat high. Nevertheless, this data suggests that if introduced to Pwm 2.0, a non-negligible number of participants would want to continue using it, and would feel comfortable using it to send encrypted email to their friends and family.

DISCUSSION

In this section we discuss noteworthy items, including participant experiences, opinions, and preferences regarding secure email and lessons learned from improving Pwm. We refer to participants here as R1–R52, with the number corresponding to the order in which they participated in our study.

Automatic and Manual Encryption

Our results indicate no significant differences between automatic and manual encryption. As such, we hypothesize that the effect observed in the original Pwm study was due to confounding factors between the two systems being studied (i.e., automatic encryption–Pwm, manual encryption–MessageProtector).[22] Alternatively, it is possible that manual encryption would have benefited Pwm's original implementation, but that the modifications we made while creating Pwm 2.0 were sufficient to provide those same benefits.

Pwm 2.0's High Usability

Pwm 2.0's SUS score of 80 falls in the 87th percentile for system usability [19], and is the highest score for secure email systems [16, 14]—Mailvelope (35, 4th percentile), Tutanota (52, ~15th), Encipher.it (61, ~30th), Voltage (62, ~32nd), Virtru (72, 63rd), original Pwm (74, 70th).[23]

Additionally, most participants understood how Pwm 2.0 was protecting their messages, only rarely made mistakes, and were interested in using Pwm 2.0 with their friends and family. These positive

[19] We believe fewer participants watched the compose tutorial, which appeared the first time participants clicked on Gmail's "Compose" button, because it was not clear they needed to see it at this point and because the tutorial drew attention to the option that they could skip it. Better tutorial design could possibly address this issue.

[20] In the study, Pwm 2.0 was referred to as just Pwm.

[21] The confidence intervals reported here were calculated using the Adjusted-Wald binomial confidence interval, $\alpha = 0.05$.

[22] This hypothesis is supported by similar results in Atwater et al.'s work [1].

[23] Neither Message Protector [16] or Atwater et al.'s system [1] had functioning key management, so their scores are not reported here.

opinions were also reflected in numerous positive qualitative responses. For example, R26, R39, and R42 expressed, respectively,

> *"Pwm was very simple, but definitely carried a [sic] aura of confidence in actually protecting your information. Now, how well it does is something that I cannot determine due to my lack of knowledge, but it was very well put together. It was very concise and user friendly and did not require esoteric knowledge to operate. I would definitely feel more comfortable sending sensitive information over email if I were using Pwm versus just sending it via an email provider."*

> *"I liked how I could encrypt sensitive information like bank account information, credit card, and other things. It wasn't that hard to use. I didn't have to download anything; all I had to do was just save a bookmark and then click on it. It was really easy to use."*

> *"I liked how it made encrypting important information so easy. The tutorial was fast and easy. I like that it is easy and convenient to use with day to day emails. I liked that the background was blue so I knew when it was encrypting."*

Delayed Encryption
While we did not specifically collect data regarding users' opinions toward our delayed encryption mechanism, we note that not a single participant complained about encryption being too fast. This is especially significant when compared to the original Pwm study, where over 33% of participants self-reported that encryption was too fast and therefore untrustworthy. The difference between these two proportions is statistically significant ($N-1$ two-proportion test for comparing two independent proportions–$p < 0.001$). While this may not mean that users' concerns regarding the speed of encryption were eliminated, it is strongly suggestive that they were reduced enough not to be a distraction.

Mixed mode email
The optional plaintext greeting that can accompany email encrypted by Pwm 2.0 is intended to help email senders give confidence to recipients who have never used Pwm 2.0 that the email is authentic and not spam. Surprisingly, during our study we noted that a small, but significant number of users would include a plaintext greeting in many of their encrypted email messages. For example, in Task 4 eight participants (8; 16%) including a greeting stating that they were sending their direct deposit information. Interestingly, this was actually a feature that seven participants (7; 14%) listed as one of their favorite features of Pwm 2.0. R12 and R51 stated, respectively,

> *"It wasn't rigid, I could write part of a message and have the other parts encrypted if I wanted. It was very clear what was encrypted and what wasn't [...]"*

> *"That it lets me chose when to encrypt and when not to. It's also nice to have the option of writing a message before the encrypted part so others know it's not spam."*

Look and Feel
Participants indicated that having a color-scheme that was distinct from Gmail helped them more easily understand what information was encrypted and what information was in the clear. This intuitive understanding potentially helped participants avoid mistakenly entering sensitive information where it would not be protected. Additionally, participants mentioned that it helped them feel more confident in the system. Thus it is clear that when designing tightly integrated systems, it is important to have a distinct look and feel.

Instructing New Users
Even though Pwm 2.0 automatically inserts instructions on how to setup and use Pwm 2.0 in every encrypted message, most participants were unaware of this. During Task 5, participants would often spend several minutes trying to open an old Pwm 2.0 message to grab the instructions, only to have the message immediately decrypted and the instructions disappear. It would be helpful to make it more clear that these instructions are always included or to add an option that allows users to explicitly add these instructions.

CONCLUSION
Our modifications to Pwm's interface have significantly increased its usability and security. In our user study, Pwm 2.0 is rated with an 80.0 SUS score, the highest reported SUS score for secure email systems. Over 80% of participants expressed a desire to begin using Pwm, and over 90% of participants believed that their friends and family could easily start using Pwm. Pwm 2.0 also helped participants avoid mistakenly sending their sensitive data in the clear.

This work represents an important step toward providing usable, secure email encryption. While Pwm 2.0 has lower theoretical security than PGP-based systems, it provides higher practical security for an ordinary user. Whereas in studies of PGP-based systems most users were unable to encrypt their emails [26, 21, 15], in Pwm 2.0 all users are able to encrypt their emails. While there is little evidence that PGP-based systems will ever be sufficiently usable for the masses [12], techniques from Pwm 2.0 could be applied to PGP-based systems to raise their relative usability.

ACKNOWLEDGMENT
We would like to thank Mike Jones and Dan Olsen for providing feedback on this paper.

REFERENCES

1. Atwater, E., Bocovich, C., Hengartner, U., Lank, E., and Goldberg, I. Leading Johnny to water: Designing for usability and trust. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, USENIX Association (Montreal, Canada, 2015), 69–88.
2. Bangor, A., Kortum, P., and Miller, J. An empirical evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction 24*, 6 (2008), 574–594.
3. Bangor, A., Kortum, P., and Miller, J. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies 4*, 3 (2009), 114–123.
4. Brooke, J. SUS—a quick and dirty usability scale. In *Usability Evaluation in Industry*. CRC Press, Boca Raton, FL, 1996.
5. Brooke, J. SUS: A retrospective. *Journal of Usability Studies 8*, 2 (2013), 29–40.
6. Durumeric, Z., Adrian, D., Mirian, A., Kasten, J., Bursztein, E., Lidzborski, N., Thomas, K., Eranti, V., Bailey, M., and Halderman, J. A. Neither snow nor rain nor MITM. . . : An empirical analysis of email delivery security. In *Fifteenth ACM Internet Measurement Conference (IMC 2015)*, ACM (Tokyo, Japan, 2015), 27–39.
7. Fahl, S., Harbach, M., Muders, T., Smith, M., and Sander, U. Helping Johnny 2.0 to encrypt his Facebook conversations. In *Eighth Symposium on Usable Privacy and Security (SOUPS 2012)*, ACM (Washington, D.C., 2012), 11.
8. Foster, I. D., Larson, J., Masich, M., Snoeren, A. C., Savage, S., and Levchenko, K. Security by any other name: On the effectiveness of provider based email security. In *Twenty-Second ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, ACM (Denver, CO, 2015), 450–464.
9. Garfinkel, S. L. Email-based identification and authentication: An alternative to PKI? In *Twenty-Fourth IEEE Symposium on Security and Privacy (S&P 2003)*, IEEE Computer Society (Oakland, CA, 2003), 20–26.
10. Garfinkel, S. L., and Miller, R. C. 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)*, ACM (Pitsburg, PA, 2005), 13–24.
11. Holz, R., Amann, J., Mehani, O., Wachs, M., and Kaafar, M. A. TLS in the wild: An internet-wide analysis of TLS-based protocols for electronic communication. In *Twenty-Fourth Network and Distributed System Security Symposium (NDSS 2016)*, The Internet Society (San Diego, CA, 2016).
12. Marlinspike, M. Gpg and me, 2015. Accessed Sep 25, 2015. `http://www.thoughtcrime.org/blog/gpg-and-me/`.
13. Milgram, S., and Van den Haag, E. *Obedience to Authority*. Ziff–Davis Publishing Company, New York, NY, 1978.
14. Ruoti, S., Andersen, J., Heidbrink, S., ONeill, M., Vaziripour, E., Wu, J., Zappala, D., and Seamons, K. “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)*, ACM (San Jose, CA, 2016), 4298–4308.
15. Ruoti, S., Andersen, J., Zappala, D., and Seamons, K. Why Johnny still, still can’t encrypt: Evaluating the usability of a modern PGP client, 2015. arXiv preprint arXiv:1510.08555.
16. Ruoti, S., Kim, N., Burgon, B., Van Der Horst, T., and Seamons, K. Confused Johnny: When automatic encryption leads to confusion and mistakes. In *Ninth Symposium on Usable Privacy and Security (SOUPS 2013)*, ACM (Newcastle, United Kingdom, 2013).
17. Ruoti, S., Roberts, B., and Seamons, K. Authentication melee: A usability analysis of seven web authentication systems. In *Twenty-fourth International Conference on World Wide Web (WWW 2015)*, ACM (Florence, Italy, 2015), 916–926.
18. Sauro, J. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC, Denver, CO, 2011.
19. Sauro, J., and Lewis, J. R. *Quantifying the User Experience: Practical Statistics for User Research*. Elsevier, Amsterdam, The Netherlands, 2012.
20. Shamir, A. Identity-based cryptosystems and signature schemes. In *Fourteenth International Cryptology Conference (Crypto 1984)*, Springer (Santa Barbara, CA, 1984), 47–53.

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

22. Sotirakopoulos, A., Hawkey, K., and Beznosov, K. "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).

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

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

25. van der Horst, T. W., and Seamons, K. E. Encrypted email based upon trusted overlays, March 2009. US Patent 8,521,821.

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

A Usability Study of Four Secure Email Tools Using Paired Participants

SCOTT RUOTI, Brigham Young University
JEFF ANDERSEN, Brigham Young University
SCOTT HEIDBRINK, Brigham Young University
MARK O'NEILL, Brigham Young University
ELHAM VAZIRIPOUR, Brigham Young University
JUSTIN WU, Brigham Young University
DANIEL ZAPPALA, Brigham Young University
KENT SEAMONS, Brigham Young University

Secure email is increasingly being touted as usable by novice users, with a push for adoption based on recent concerns about government surveillance. To determine whether secure email is ready for grassroots adoption, we employ a laboratory user study that recruits pairs of novice users to install and use several of the latest systems to exchange secure messages. We present both quantitative and qualitative results from 25 pairs of novices as they use Pwm, Tutanota, and Virtru and 10 pairs of novices as they use Mailvelope. Participants report being more at ease with this type of study and better able to cope with mistakes since both participants are "on the same page." We find that users prefer integrated solutions over depot-based solutions, and that tutorials are important in helping first-time users. Additionally, hiding the details of how a secure email system provides security can lead to a lack of trust in the system. Finally, our results demonstrate that PGP is still unusable for novice users, with 9 out of 10 participant pairs failing to complete the study.

CCS Concepts: •**Security and privacy** → *Domain-specific security and privacy architectures;* **Usability in security and privacy;**•**Human-centered computing** → **User studies; Empirical studies in HCI;** *Laboratory experiments;*

General Terms: Design, Experimentation, Human factors, Security

Additional Key Words and Phrases: Grassroots adoption, paired-participants, PGP, secure email

ACM Reference Format:
Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, Kent Seamons, 2016. A Usability Study of Four Secure Email Tools Using Paired-Participants *ACM Trans. Priv. Secur.* V, N, Article A (January YYYY), 26 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In recent years there has been an increase in the promotion of secure email, with tools such as Tutanota, Virtru, Mailvelope, ProtonMail, StartMail, Hushmail and others being pitched for everyday use by novice users. This interest is likely spurred by concern over government surveillance of email, particularly when third-party services such as Gmail and Hotmail store email in plaintext on their servers. The Electronic Frontier Foundation has heavily promoted secure communication and has released a security scorecard of secure messaging systems that includes several email tools [Foundation 2015].

While TextSecure, Signal, WhatsApp, and other secure instant messaging platforms are becoming popular, it is unclear whether efforts to encourage users to likewise switch to secure email will succeed, given that usable, secure email is still an unsolved problem more than fifteen years after it was first formally studied [Whitten and Tygar 1999]. Moreover, widespread use of secure email partly depends on whether it could be adopted in a grassroots fashion, where both parties of an email conversation are novice users. All prior laboratory usability studies of secure email bring one novice user at a time

This work was completed while Scott Ruoti, Mark O'Neill, and Scott Heidbrink were interns at Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

DOI: http://dx.doi.org/10.1145/0000000.0000000

into the lab and have him or her communicate with a study coordinator using a secure email system. While this helps researchers understand how well a novice can start using secure email when paired with an expert user, it does not shed light on whether a pair of novices can start using the system independently.

In this work, we describe a novel paired-participant methodology for the study of secure email, in which pairs of novice users were brought into the lab and asked to exchange secure email between themselves. We asked participants to bring a friend with them, ensuring the participants already knew each other, in the hope that participants would behave more naturally. Participants then used one or more secure email systems without any specific training or instructions on how to use the system other than what the system itself provided. The main difference between this type of study and a traditional single-user study are that the participants played different roles (initiating contact versus being introduced to secure email) and that they interacted with another novice user and not a study coordinator.

In this paper we describe two different studies using this methodology. In our first study, 25 pairs of participants tested three different secure email systems: Pwm, Tutanota, and Virtru. Each of these systems represents a different philosophy related to the integration of secure email with existing email systems. Pwm integrates secure email with users' existing Gmail accounts, allowing them to compose and receive secure email with a familiar interface. In contrast, Tutanota is a secure email depot that requires users to log into Tutanota's website to interact with their secure messages. Virtru is a hybrid of these two approaches, allowing users who install the Virtru plugin to use secure email that is integrated with Gmail, but also allowing non-Virtru users to receive encrypted email through a depot-based system on Virtru's website.

In our second study, ten pairs of participants tested Mailvelope, a modern PGP tool that was designed with usability in mind. Mailvelope is a browser extension that integrates with users' webmail systems, and is the only such tool that appears on the EFF's secure messaging scorecard[1].

Our results and participant comments lead to the following contributions:

(1) **Using pairs of novice users for an email usability study has several benefits.** Having participants play different roles allowed us to gather data about different types of first-use cases (i.e., sending a secure email first vs receiving a secure email first). In addition, participants exhibited more natural behaviors and indicated that they felt "more at ease", that they and their friend were "on the same page" or at the same level of technical inexpertise, and that they did not feel discomfort from being "under the microscope."
(2) **Hiding the details of how a secure messaging system provides security can lead to a lack of trust in the system.** This phenomenon was first noted in some of our earlier work [Ruoti et al. 2013], but those results are affected by multiple confounding factors [Atwater et al. 2015; Ruoti et al. 2016]. This paper provides further evidence that when security details are hidden from users, users are less likely to trust the system. For example, although Pwm and Virtru utilize the same authentication method, Pwm completes authentication without user interaction, causing several users to doubt Pwm's security. Similarly, participants like that Tutanota requires that an email be encrypted with a password, since this makes it clear that the message was protected, unlike other systems that manage and use keys behind the scenes.
(3) **Users prefer integrated solutions over depot-based solutions.** While to some it may be intuitive that users would prefer to continue using their existing email accounts, a number of depot-based systems have appeared recently (e.g., Tutanota, ProtonMail, StartMail). Our results demonstrate that most everyday users strongly dislike using separate websites such as secure email depots to read their email.
(4) **Tutorials are very important for users of secure email.** When asked what they liked about Pwm and Virtru, participants often reported that it was the tutorials presented alongside these systems. The efficacy of these tutorials is shown by the fact that while using Pwm and Virtru, participants were able to quickly complete the study task, whereas while using Tutanota—which lacks a tutorial—participants took on average 72% longer to complete the study tasks, often making mistakes as they did so.

[1] https://www.eff.org/secure-messaging-scorecard

(5) **Users want the ability to use secure email but are unsure about when they would use it.** Three-quarters of the participants in our study indicated that they wanted to be able to encrypt their email, but only one-quarter indicated that they would want to do so frequently. Furthermore, when asked to describe how they would use encrypted email in practice, most participants were unsure, giving only vague references to how secure email might be useful. This demonstrates a need for future research to establish whether the true problem facing the adoption of secure email is usability or that day-to-day users have no regular need to send sensitive data via email.
(6) **PGP still appears to be unusable for the masses.** In our second study, nine out of the ten participant pairs were unable to complete the study, and had not even made significant progress in the hour allotted for the study. The only pair that completed the study took slightly longer than the allotted hour, and reported that they were only successful at using PGP because one of them had learned about public key cryptography in a college course. Moreover, all participants found key establishment and sharing difficult.
(7) **PGP-based secure email systems can be improved.** Based on our observations during both studies and participants' feedback, we have identified several suggestions that would significantly improve the chances that novice users could adopt PGP. First, integrated tutorials would be helpful in assisting first-time users in knowing what they should be doing at any given point in time. Second, an approachable description of public key cryptography could help users correctly manage their own keys. Third, in line with previous work by Atwater et al. [Atwater et al. 2015], we find that PGP-based tools would be well served by offering automatically generated emails for unknown recipients asking them to install the PGP software, generate a public key, and share it with the sender. Finally, the PGP block itself could be enhanced to help non-PGP users who receive an encrypted email know how to work with their friend to get an encrypted message they will be able to read.

2. BACKGROUND

In this section we provide background on the area of secure email. First, we describe the current state of email security. We then describe approaches for securing email with end-to-end encryption. Finally, we discuss related work; i.e., usability studies of secure email.

2.1. Email Security

When email was first designed in 1971[2] no meaningful attention was paid to security. As such, it was originally trivial for an attacker to steal email during transit or to send messages with falsified sender information. In recent years, there have been attempts to patch security into email. For example, TLS is now used to protect email during transmission, and DKIM and DMARC are used to authenticate the sender of an email. However, the deployment of these technologies is limited and they are often misconfigured.

In an analysis of email delivery security (i.e., TLS, DKIM, DMARC, SPF), Durumeric et al. found that a majority of email is still vulnerable to attack [Durumeric et al. 2015]. They showed that only 35% of SMTP servers are configured to use TLS, and even when TLS is enabled it is often vulnerable to a downgrade attack. Similarly, they demonstrated that the adoption of DKIM and DMARC were so low that they provide no practical benefits. These results were further confirmed by concurrent work by both Foster et al. [Foster et al. 2015] and Holz et al. [Holz et al. 2016].

As such, email is still an easy target for attackers. For example, Durumeric et al. found that in seven countries over 20% of inbound Gmail messages are being stolen [Durumeric et al. 2015]. Additionally, the inability to authenticate the sender of an email increases the likelihood of email phishing, a multi-billion-dollar problem.[3] Perhaps most troubling, even if TLS, DKIM, and DMARC were to be widely adopted and configured correctly, it would do nothing to protect email when at rest.[4]

[2]http://openmap.bbn.com/~tomlinso/ray/firstemailframe.html
[3]http://krebsonsecurity.com/2016/04/fbi-2-3-billion-lost-to-ceo-email-scams/
[4]At rest, email can be stolen as the result of a breach (https://www.washingtonpost.com/world/national-security/chinese-hackers-who-breached-google-gained-access-to-sensitive-data-us-officials-say/, a malicious insider (http://gawker.com/5637234/gcreep-google-engineer-stalked-teens-spied-on-chats, or a subpoena (https://en.wikipedia.org/wiki/PRISM_(surveillance_program), http://www.law360.com/articles/488725/post-snowden-google-report-shows-data-requests-growing).

2.2. End-to-end Email Encryption

The problems afflicting secure email could be solved through the use of end-to-end encryption. A well-known approach for providing end-to-end encryption is Pretty Good Privacy (PGP) [Garfinkel 1995]. It was developed in 1991 by Phil Zimmerman, and allows users to encrypt and sign their email messages using public key cryptography. In PGP, keys are generally validated using the web of trust—i.e., users verify and sign their associates' public keys; a user can check if they trust a key by seeing if they or one of their associates has signed that key. Public keys can be shared 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.

Another end-to-end approach is Secure/Multipurpose Internet Mail Extensions (S/MIME [Ramsdell and Turner 2010]). Similar to PGP, S/MIME uses public key cryptography to encrypt and sign email. Unlike PGP's web of trust, S/MIME certificates are authenticated using the certificate authority (CA) system.[5,6] Users still need to share public keys just as they do in PGP.

A third approach is identity-based encryption (IBE [Shamir 1984]). IBE is a public key system where a user's public key is simply their email address. Private keys are generated by a trusted third-party server, which authenticates the identity of the user before providing them with their private key. Unlike PGP and S/MIME, users do not need to install secure email software, generate a key pair, and share their private key before they can receive their first encrypted message.

In addition to public key cryptography, it is also possible to use symmetric key cryptography for end-to-end encryption of email. One approach is for a user to select and share a password which will be used to encrypt their email. Another approach is to have a trusted third-party key escrow server that can generate and distribute symmetric keys for users.

In this paper, we analyze the usability of systems that represent a variety of end-to-end encryption approaches: PGP, IBE, passwords, and a custom key escrow scheme.

2.3. User Attitudes Towards Secure Email

Gaw et al. [Gaw et al. 2006] interviewed users at a political activist organization that use secure email and noted that adoption was driven by the organization deciding encryption was necessary due to secrecy concerns. These interviews also showed that having IT staff set up the secure email software was necessary to enable the successful adoption of secure email. Even with this support, there were users who did not intend to use the software regularly, due to usability concerns and social factors. In this work, we tested systems that help users begin using secure email without the support of IT staff.

Renaud et al. conducted a series of 21 interviews regarding users' understanding of email [Renaud et al. 2014]. They found that most users did not have an accurate mental model regarding how email functioned, including how an attacker might steal their email. Renaud et al. suggested that this lack of understanding made it difficult for users to understand the value of secure email, potentially explaining why users have not adopted it. Their research suggests that more work needs to be done to build comprehensive mental models of email and secure email for end-users.

Tong et al. created a set of metaphors and descriptions that attempted to describe public key-based secure email to users [Tong et al. 2014]. They found that even though these new metaphors and description only had a marginal effect on users' mental models, when these metaphors and descriptions were implemented into a mock secure email interface participants were able to more intuitively understand how the interface would function.

Bai et al. explored user attitudes toward different models for obtaining a recipient's public key in PGP [Bai et al. 2016]. In their study, they built two PGP-based secure email systems, one that used the traditional key-exchange model[7] and one that used a key directory-based registration model.[8] Users were provided with instructions on how to use each tool and given several tasks to complete. Afterwards, participants shared their opinions regarding the key exchange models. The results of this

---

[5]The CA system is the same mechanism by which websites are authenticated in TLS.

[6]More recent versions of PGP also support certificates validated using the CA system, but in our experience this feature is not widely used for secure email.

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

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

study showed that, overall, individuals preferred the key-directory-based registration model, though they were not averse to the traditional key-exchange model either.

2.4. Related Work

Whitten and Tygar [Whitten and Tygar 1999] conducted the first formal user study of a secure email system (i.e., PGP 5), which uncovered serious usability issues with key management and users' understanding of the underlying public key cryptography. They found that a majority of users were unable to successfully send encrypted email in the context of a hypothetical political campaign scenario. The results of the study served as a wake-up call to the security community and helped shape modern usable security research.

Sheng at al. demonstrated that despite improvements made to PGP in the seven years since Whitten and Tygar's original publication (i.e., PGP 9), key management was still a challenge for users [Sheng et al. 2006]. Furthermore, they showed that in the new version of PGP, encryption and decryption had become so transparent that users were unsure if a message they received had actually been encrypted.

Atwater et al. evaluated the usability of PGP using a mock-up of a secure email tool that automatically generates key pairs for users, shares the generated public key with a key server, and retrieves the recipient's public key as needed. Their results showed that with these modifications, users could successfully use PGP to send and receive secure email. Unfortunately, their mock-up did not correctly simulate PGP's key management, making their system unsuitable for inclusion in our own study and calling into question the validity of their results.

Garfinkel and Miller created a secure email system using S/MIME and used this system to replicate Whitten and Tygar's earlier study [Garfinkel and Miller 2005]. Their work demonstrated that automating key management provides significant usability compared to earlier studies that burdened users with key management tasks. Still, they observed that their tool "was a little too transparent" in how well it integrated with Outlook Express, and sometimes users failed to read the instructions accompanying the visual indicators.

Ruoti et al. developed Private WebMail (Pwm), a secure email tool that uses IBE to encrypt users' messages [Ruoti et al. 2013]. Across three usability studies of Pwm, they demonstrated that all participants could use Pwm to send and receive encrypted email. However, they also found that some users made mistakes and were hesitant to trust the system due to the transparency of its automatic encryption. They later revised Pwm to address these issues, and demonstrated that their modified system receives the highest usability ratings of any secure email system tested in the literature [Ruoti et al. 2016].

Ruoti et al. also developed MessageProtector, a stand-alone, IBE-based encryption tool [Ruoti et al. 2013]. Unlike other secure email systems, MessageProtector is not integrated with an email tool, but instead requires users to compose their messages in MessageProtector and then copy the resulting ciphertext into their mail program. Two usability studies demonstrated that users approved of MessageProtector, though they wished that it could be implemented as part of the browser. Song created a modified version of MessageProtector that was integrated into the browser, and found that it was perceived as being more usable than the original version [Song 2014].

In Ruoti et al.'s work they also evaluated Voltage Mail, an IBE-based, secure email depot and Encipher.it, a password-based encryption tool [Ruoti et al. 2013]. In a usability study of both systems, Ruoti et al. showed that while users could successfully send and receive encrypted email using these systems, they also strongly disliked both systems. The primary reason for this dislike was that neither tool was integrated with the user's webmail system.

The methodology in this paper is unique in that it tests whether *two* novice users can collaboratively begin using secure email. In contrast, all past studies involved a single participant interacting with an expert user of secure email (i.e., the study coordinator). As discussed in this paper, using pairs of novice participants has significant advantages over having a study coordinator simulate a user.

3. SECURE EMAIL

Two and a half decades after the invention of PGP, secure email still remains sparsely used. While some businesses require the use of secure email by their employees, there is little use of secure email by the population at large. While it is possible that secure email will eventually diffuse from the workplace, it may be that if secure email is to flourish, it will do so because of grassroots adoption (i.e., if participants are able to discover secure email on their own and easily begin using it with their acquaintances).

To date, no secure email studies have tested the ability of two novices to begin using secure email; instead, these studies have tested a single novice interacting with an expert user (i.e., a study coordinator). Both Whitten and Tygar's study [Whitten and Tygar 1999] as well as Garfinkel and Miller's study [Garfinkel and Miller 2005] used a simulated political campaign, where the study participant was the only individual in the campaign who did not already know how to use PGP. Similarly, studies by Sheng et al. [Sheng et al. 2006], Ruoti et al. [Ruoti et al. 2013], Song [Song 2014], Atwater et al. [Atwater et al. 2015], and Bai et al. [Bai et al. 2016] involved participants sending email to study coordinators, none of whom were instructed to simulate a novice user.

Even if the study coordinators had attempted to simulate a novice user, there are difficulties with this approach. First, study coordinators are unlikely to make mistakes while using the encryption software, which is atypical of a true novice. Even if study coordinators make use of scripted mistakes, there is a strong risk that these mistakes might be seen as artificial by participants, thereby breaking immersion for the participant. Second, in many tasks there is a high level of variability possible in participant actions, making it difficult to script for all possible situations, and unscripted responses from coordinators are likely to be biased by their experience with the system. Third, participants are likely to attribute any problems they encounter to their own mistakes, and not to the coordinator, whereas when interacting with a friend, participants are just as likely to attribute the mistake to their friend as to themselves.[9]

To avoid these difficulties, our study uses two novice participants. This is the first study to test whether two novice participants, who know each other beforehand, can successfully use secure email in a grassroots fashion. Our observations, as discussed later in this paper, show that this approach produces more natural behavior than when participants email a study coordinator. Moreover, this approach allows us to examine how users perform when they are introduced to secure email in different ways (i.e., installing and then sending an email vs. receiving an email and then installing).

To select which systems to test, we surveyed existing secure email systems, including those listed on the EFF's scorecard [Foundation 2015], and filtered them according to two criteria. First, we focused on browser-based solutions, as previous work has shown that this approach is preferred by users [Ruoti et al. 2013; Atwater et al. 2015]. Second, we required the systems to use automatic key management, as research has shown that users are highly amenable to this approach [Garfinkel and Miller 2005; Ruoti et al. 2013]. Of the systems that matched these criteria, we found that they could be grouped into three types of secure email systems: integrated, depot-based, and a hybrid of integrated and depot-based systems. For each of these groups, we tested the systems in the group, and selected the system that we felt had the best usability and included that system in our study. In addition to the above systems, we also included a secure email system based on PGP, because this approach is viewed as highly secure by the research community and a recent system claimed to significantly improve its usability.

The remainder of this section describes the types of secure email that were tested, as well as the representative system for each. Screenshots of each system can be found in Appendix A.

3.1. Integrated Secure Email (Pwm)

"Integrated secure email" refers to secure email systems that integrate with users' existing email systems. In this model, users do not need to create new accounts and are able to encrypt messages within the email interfaces they are already accustomed to [Ruoti et al. 2013].

Private WebMail (Pwm)[10] is the representative system for this type of secure email. Pwm was developed as part of our research [Ruoti et al. 2013; Ruoti et al. 2016] and has the highest usability[11] of any secure email system tested in the literature [Ruoti et al. 2016]. In addition, because Pwm has previously been the subject of several formal user studies, it provides a good baseline for comparing the results of the other systems tested in this study.

Pwm is a browser extension that tightly integrates with Gmail's web interface to provide secure email. Users are never exposed to any cryptographic operation, including the verification of the user's identity, which are completed without user interaction. Pwm modifies the color scheme of Gmail for en-

[9]In some ongoing work, we have attempted to simulate a novice user and encountered these difficulties in practice [Ruoti et al. 2016]

[10]https://pwm.byu.edu/

[11]Based on the System Usability Scale [Brooke 1996].

crypted emails in order to help users identify which messages have been encrypted. Pwm also includes inline tutorials that instruct users on how to operate Pwm.

Pwm's threat model is focused on protecting email from individuals who do not have access to the sender's or recipient's email account. While this does not protect email against attackers who compromise the user's email account, it does provide security during transmission and storage of the email. Pwm is susceptible to a malicious email service provider.

3.2. Depot-Based Secure Email (Tutanota)

"Depot-based secure email" refers to secure email systems that use a separate website from users' existing email systems. In this model, users have a separate account with the depot where they can send and receive secure emails. When a user receives a new message in their depot account, many depot-based systems will send an email to the user's standard email address, informing them that they have a new email to check in the depot system. Often, these systems do not allow users to send email to individuals not already using the depot. Depot-based systems are commonly deployed by companies and organizations for secure communication.

There are many depot-based systems to choose from. We chose Tutanota[12] because it was the most usable of the depot systems we tested, available for free, available to new users, and receiving positive publicity on Twitter. Other popular systems charged an annual fee (e.g., Hushmail and StartMail) or were currently not offering email addresses to new users (e.g., ProtonMail).

While there are many depot-based systems to choose from, most are either costly (e.g., Hushmail and StartMail) or are currently not offering email addresses to new users (e.g., ProtonMail). We chose Tutanota[13] because it was the most usable of the depot systems we tested, is free, is currently available to new users, and is receiving publicity on Twitter.

Tutanota assigns users an email address ending in `@tutanota.com`. Users can send and receive email from this address as they normally would. During account creation, Tutanota generates a public/private key pair for the user. These keys are stored on Tutanota's servers, with the private key being encrypted with the user's Tutanota account password. When Tutanota users send messages to other Tutanota users, the messages are automatically encrypted and signed with the appropriate keys. When a Tutanota user sends a message to a non-Tutanota user, they have the option of encrypting it with a shared secret (i.e., password). When the non-Tutanota user receives the encrypted email, they are redirected to Tutanota's website, where they can enter the shared secret and decrypt the message. Tutanota's interface also allows the non-Tutanota user to respond to the message, and will encrypt the message using the same shared secret.

The threat model for Tutanota is similar to Pwm, except that instead of having normal and secure email stored in the same email accounts, they are stored in separate accounts. This means that if a user's normal email account is broken into, their sensitive messages are still secure. Users are still susceptible to a malicious email service provider, or to having their secure email account password guessed/stolen.

3.3. Hybrid Secure Email (Virtru)

Virtru[14] is a hybrid of integrated and depot-based secure email. Once Virtru's browser plugin is installed, it functions much the same as Pwm, including automatic key management and integration with the webmail provider. If a Virtru user sends an email to a non-Virtru user, the sender still does so through the webmail provider, but the recipient will receive an email informing them that they need to log into Virtru's website to view their message. At this point Virtru is similar to Tutanota in its management of new users, except that instead of providing a password, non-Virtru users are asked to prove that they own their email address. As such, the threat model for Virtru is identical to Pwm.

3.4. PGP (Mailvelope)

We chose Mailvelope[15], a modern PGP-based tool as our representative system. Mailvelope was our preferred choice for several reasons: First, it is the only PGP-based secure email tool promoted by the

[12]https://tutanota.com/
[13]https://tutanota.com/
[14]https://www.virtru.com/
[15]https://www.mailvelope.com/

EFF's secure messaging scorecard[16] that is browser-based. Second, Mailvelope is highly rated on the Chrome Web Store, with 242 users collectively giving it 4.6 out of 5 stars. Third, Mailvelope claims to be "easy-to-use"[17] and focused on helping novice users begin sending encrypted email.[18] Finally, in our evaluation of other PGP-based secure email tools, we found Mailvelope to be at least as usable as the alternatives (i.e., GPG Tools, Enigmail, Google's End-to-End Encryption).

Like Pwm, Mailvelope integrates with the user's webmail provider. Upon installation, users need to generate a PGP key pair and select a password to encrypt their private key. To encrypt a message, the user takes the following steps: (1) click on a button that opens Mailvelope's compose interface in a new window; (2) compose their message, click encrypt, and select the intended recipients; (3) click the transfer button, which then sends their PGP-encrypted message back to the webmail provider's compose interface, where the user can then send the encrypted message. Upon receipt of an encrypted email, the Mailvelope user take the following steps: (1) click a lock icon that is displayed over the encrypted text; (2) enter the password for their private key.

Mailvelope has a stricter threat model than the other systems. In order to compromise a PGP encrypted message, the attacker must accomplish three things: (1) steal the user's email; (2) steal the user's private key; (3) steal the password for the user's private key. An attacker who gains access to the user's email account could attempt to convince the user's contacts to encrypt messages with the attacker's public key instead of the user's true public key. Still, this does not compromise the security of messages previously encrypted with the correct public key.

4. FIRST STUDY–METHODOLOGY

We conducted two IRB-approved user studies wherein pairs of participants used secure email to communicate sensitive information to each other. Both studies ran concurrently, with participant pairs assigned randomly to one of the studies upon arrival. For clarity's sake, we first describe the methodology and results of our first study, and then the methodology of our second study. A full copy of the surveys and recruitment poster can be found at https://tops2016.isrl.byu.edu.

This section gives an overview of our first study and describes the scenario, task, study questionnaire, and post-study interview. In addition, we discuss the development and limitations of the study.

4.1. Study Setup

The study ran for two weeks—beginning Tuesday, September 8, 2015 and ending Friday, September 18, 2015. In total, 25 pairs of participants (50 total participants) completed the study. Participants took between forty and sixty minutes to complete the study, and each participant was compensated $15 USD for their participation. Participants were required to be accompanied by a friend, who served as their counterpart for the study. For standardization and requirements of the systems tested in the study, both participants were required to have Gmail accounts.

When participants arrived, they were read a brief introduction detailing the study and their rights as participants. Participants were informed that they would be in separate rooms during the study and would use email to communicate with each other.[19] Participants were also informed that a study coordinator would be with them at all times and could answer any questions they might have.

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

During the study, participants were asked to complete a multi-stage task three times, once for each of the secure email systems being tested: Pwm, Tutanota, and Virtru. The order in which the participants used the systems was randomized. For each system, participants installed any necessary software and were then given fifteen minutes to complete the task. If they were unable to complete the task in the

[16] https://www.eff.org/secure-messaging-scorecard
[17] https://www.mailvelope.com/faq
[18] https://github.com/mailvelope/mailvelope/issues/14#issuecomment-11419791
[19] The study coordinators ensured that the participants knew each other's email addresses.

time limit, the study coordinators helped them move to the next system. In practice, this only occurred a single time.

4.2. Demographics

We recruited Gmail users for our study at a local university. Participants were two-thirds female: female (33; 66%), male (17; 34%). Participants skewed young: 18 to 24 years old (44; 88%), 25 to 34 years old (6; 12%).

We distributed posters across campus to avoid biasing our results to any particular major. All participants were university students,[20] with the majority being undergraduate students: undergraduate students (40; 80%), graduate students (10; 20%). Participants were enrolled in a variety of majors, including both technical and non-technical majors. No major was represented by more than four participants, with the vast majority only having one or two participants.

4.3. Scenario Design

During the study, participants were asked to role-play a scenario about completing taxes. Each participant was shown the following text, respectively.

— **Johnny.** 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.
— **Jane.** 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.

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.

4.4. Task Design

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

(1) Johnny would encrypt and send his SSN and last year's tax PIN to Jane.
(2) Jane would reply to this sensitive information with a confirmation code and this year's tax PIN. This information would also be encrypted.
(3) Johnny would reply and let Jane know he had received the confirmation code and this year's tax PIN.

The instructions guiding the participants through the three stages are as follows:

— **Johnny.** In this task, you'll be using {Pwm, Virtru, or Mailvelope}. The system can be found at the following website: {Appropriate website}. Please encrypt and send the following information to your friend using {Pwm, Virtru, or Mailvelope}: SSN: {Generated SSN}. PIN: {Generated PIN}.
Once you have received the confirmation code and PIN from your friend, send an email to your friend letting them know you have received this information. After you have sent this confirmation email, let the study coordinator know you have finished this task.
— **Jane–Sheet 1.** Please wait for your friend's email with their last year's tax PIN and SSN. 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.
— **Jane–Sheet 2.** 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 {Pwm, Virtru, or Tutanota} to send sensitive information to you, please also use {Pwm, Virtru, or Tutanota} to send them the confirmation code and PIN. Confirmation Code: {Generated code}. PIN: {Generated PIN}.
Once you have sent the confirmation code and PIN to your friend, wait for them to reply to you and confirm they received the information. Once you have received this confirmation, let the study coordinator know you have finished this task.

The instructions for Johnny and Sheet 1 of the instructions for Jane were given at the start of the task. Sheet 2 for Jane was given once Jane had received and decrypted the sensitive information sent

[20]We did not require this.

by Johnny in Stage 1. Participants completed this task once for each of the three systems being tested. Each time, the instructions only included information relevant to the system being tested.

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

Study coordinators were allowed to answer questions related to the study but were not allowed to provide instructions on how to use any of the systems being tested. If participants became stuck and asked for help, they were told that they could take whatever steps they normally would to solve a similar problem. Additionally, when asked for help, if the study coordinator believed communication between the two parties could help, he could remind participants that they were free to communicate with their friend and that only the sensitive information was required to be transmitted over secure email.

4.5. Study Questionnaire

We administered our study using the Qualtrics web-based survey software. Before beginning the survey, participants answered a set of demographic questions. Participants then completed the study task for each of the three secure email systems.

Immediately upon completing the study task for a given secure email system, participants were asked several questions related to their experience with that system. First, participants completed the ten questions from the System Usability Scale (SUS) [Brooke 1996; 2013]. Multiple studies have shown that SUS is a good indicator of perceived usability [Tullis and Stetson 2004], is consistent across populations [Ruoti et al. 2015], and has been used in the past to rate secure email systems [Ruoti et al. 2013; Atwater et al. 2015]. After providing a SUS score, participants were asked to describe what they liked about each system, what they would change, and why they would change it.

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

4.6. Post-Study Interview

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

After the participants completed their individual post-study interviews, they were brought together for a final post-study interview. First, participants were once again asked which system was their favorite and why. This question was intended to observe how participants’ preferences might change when they could discuss their favorite system with each other. Second, participants were asked to describe their ideal secure email system. While participants are not system designers, our experience has shown that participants often reveal preferences that otherwise remain unspoken. Finally, participants were asked to share their opinions related to doing a study with a friend. They were informed that it was the first time that we had conducted such a study. This question was designed to learn possible benefits and limitations of conducting such a two-person study.

4.7. Pilot Study

We conducted a pilot study with three pairs of highly-technical participants (six participants total). The lessons learned during the pilot study motivated two minor changes to the study. First, in addition to the three systems included in this study, the pilot study also included Mailvelope. While task completion times for Pwm, Tutanota, and Virtru averaged around five minutes, it took participants fifteen to thirty minutes to complete the assigned task with Mailvelope. This made it clear that it was unlikely that participants would be able to complete tasks for all the systems in the assigned hour. Additionally, we also estimated that participants would likely need the whole hour to complete the Mailvelope task, and so we split Mailvelope into its own study, detailed later in this paper. Second, in

Table I. SUS Scores

	Participant	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)	Range
Pwm	Johnny	25	76.3	15.3	±6.3	70.0–82.6
Pwm	Jane	25	69.1	17.2	±7.1	62.0–76.2
Pwm	Both	50	72.7	16.5	±4.7	68.0–77.4
Virtru	Johnny	25	73.1	14.7	±6.1	67.0–79.2
Virtru	Jane	25	71.4	12.7	±5.2	66.2–76.6
Virtru	Both	50	72.3	13.7	±3.9	68.4–76.2
Tutanota	Johnny	25	50.0	18.2	±7.5	42.5–57.5
Tutanota	Jane	25	54.3	17.4	±7.2	47.1–61.5
Tutanota	Both	50	52.2	17.8	±5.1	47.1–57.3

the pilot study, participants were shown all instructions within the Qualtrics survey. After the pilot, we printed out the task instructions and gave these to users for easier reference.

4.8. Limitations

During the study, a bug in the Qualtrics software led to an uneven distribution of treatments (i.e., order the systems were used). Due to this problem, treatments where Virtru was the first system were used in two-thirds (68%, n=17) of the study sessions. Other than this abnormality, treatment distribution was as expected.

We examined our data to determine what effect this abnormality had on our quantitative results, but found nothing that was statistically significant. In one case, we did discover an observable, but statistically insignificant difference in task completion times. This difference is mentioned in the Results section.

Our study also has limitations common to all existing secure email studies. First, our populations are not representative of all groups, and future research could broaden the population (e.g., non-students, non-Gmail users). Second, our study was a short-term study, and future research should look at these issues in a longer-term longitudinal study. Third, our study is a lab study and has limitations common to all studies run in a trusted environment [Milgram and Van den Haag 1978; Sotirakopoulos et al. 2010].

Our study only examines the case where a single user sends email to one other user. While this is the most likely scenario for secure email among the masses, future work could also explore alternative scenarios (e.g., sending to multiple users, mailing lists).

5. FIRST STUDY–RESULTS

In this section, we report the quantitative results from our user study. First, we report on the usability scores for each system. Next, we give the time taken to complete the task for each system as well as the number of mistakes encountered while using each system. Finally, we report which system participants indicated was their favorite.

The data from both studies is available at https://tops2016.isrl.byu.edu.

5.1. System Usability Scale

We evaluated each system using the System Usability Scale (SUS). A breakdown of the SUS score for each system and type of participant (i.e., Participant A—Johnny, Participant B—Jane, or both) is given in Table I. The mean value is used as the SUS score [Brooke 1996]. When evaluating whether a participant's role as Johnny or Jane affected the SUS score, we found the differences were small and were not statistically significant (two-tailed student t-test, equal variance—Pwm–$p = 0.12$, Virtru–$p = 0.66$, Tutanota–$p = 0.40$).

To give greater context to the meaning of each system's SUS score, we leveraged the work of several researchers. Bangor et al. [Bangor et al. 2009] analyzed 2,324 SUS surveys and derived a set of acceptability ranges that describe whether a system with a given score is acceptable to users in terms of usability. Bangor et al. also associated specific SUS scores with adjective descriptions of the system's usability. Using this data, we generated ranges for these adjective ratings, such that a score is corre-

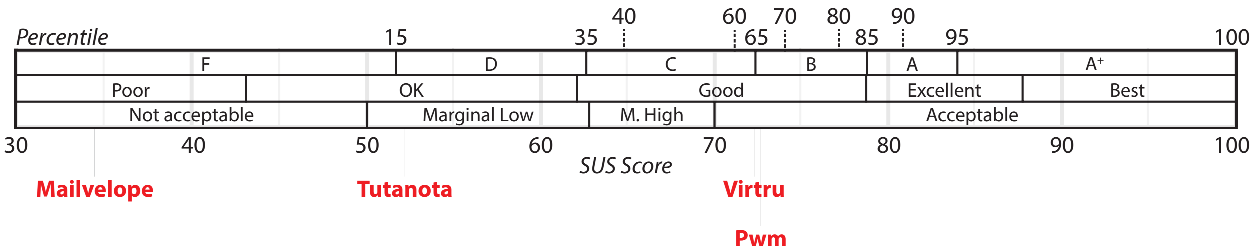


Fig. 1. Adjective-based Ratings to Help Interpret SUS Scores

lated with the adjective it is closest to in terms of standard deviations. Sauro et al. [Sauro 2011] also analyzed SUS scores from Bangor et al. [Bangor et al. 2008], Tullis et al. [Tullis and Stetson 2004], and their own data. They calculated the percentile values for SUS scores and assigned letter grades based on percentile ranges. These contextual clues are presented in Figure 1.

Pwm and Virtru's SUS scores of 72.7 and 72.3, respectively, are rated as having "Good" usability. Both systems fall right at the 65th percentile and on the line between a "B" and "C" grade. The difference between these two systems is not statistically significant (two-tailed student t-test, matched pairs—$p = 0.86$). The scores for Pwm are roughly consistent with those seen in prior work [Ruoti et al. 2013; Ruoti et al. 2016], though our results exhibited more low outliers than prior studies. Whether this difference is due to negative experience related to using these systems with a friend as compared to a study coordinator or whether it is due to differences in the study populations is unclear.

Tutanota's score of 52.2 is rated as having "OK" usability. It falls in just about the 15th percentile and at just about the base of the "D" grade. The difference between Tutanota and the other systems (i.e., Pwm and Virtru) is statistically significant (two-tailed student t-test, matched pairs—$p < 0.001$)

Finally, we note that Pwm's SUS score in this study was significantly lower than in a prior study of the same version of Pwm [Ruoti et al. 2016]. In the prior study, Pwm received a SUS score of 80.0 and had no low scoring outliers (there were six such outliers in this study). We note that the difference between the two system's SUS scores are statistically significant (two-tailed student t-test, unequal variance—$p = 0.01$). We are unsure why such a large difference manifested itself between two studies, especially when prior work has shown SUS is largely consistent between studies [Ruoti et al. 2013; Ruoti et al. 2015]. Still, we conjecture that the paired-participants study might have revealed pain points related to having two novice users communicate using Pwm, something which was only simulated in the prior study. Alternatively, it is possible that the earlier study, which was an in-depth, hour-long study of Pwm, allowed users to become more acclimated to Pwm, thereby raising their opinions of it.

5.2. Time

We recorded the time it took each participant to finish the task. Completion times are split into two stages:

(1) Timing for this stage started when Johnny clicked the "Install" (Pwm, Virtru) or "Sign Up" button. Timing ended when Johnny had successfully sent an encrypted email with his SSN and last year's tax PIN.
(2) Timing for this stage started when Jane opened the encrypted email sent in the previous stage. Timing ended after Jane had successfully sent an encrypted email with the confirmation code and this year's tax PIN. This stage included the time Jane spent determining how to decrypt the initial message. In the case of Tutanota, this included obtaining the shared secret from Johnny.

Timings were calculated using the recorded video. There were two sessions with abnormalities in the recording. First, the Virtru portion of a Jane session had become corrupted. Second, an entire Jane session was also corrupted. This second session was of special note because this participant failed to successfully complete Stage 2 using Tutanota. The remaining times are reported in Table II and are graphically shown in Figure 2.

In line with the SUS scores, both Pwm and Virtru have completion times that are roughly the same, with any differences failing to be statistically significant (two-tailed student t-test, matched pairs—

Table II. Time Taken to Complete Task (min:sec)

	Stage	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)	Range
Pwm	1	25	2:29	0:49	±0:19	2:10–2:48
Pwm	2	24	2:59	0:52	±0:21	2:38–3:20
Pwm	Both	24	5:28	1:22	±0:35	4:53–6:03
Virtru	1	25	2:39	1:09	±0:27	2:12–3:06
Virtru	2	23	3:06	1:53	±0:46	2:20–3:52
Virtru	Both	23	5:48	2:55	±1:12	4:36–7:00
Tutanota	1	25	3:49	1:04	±0:25	3:24–4:14
Tutanota	2	24	5:49	3:38	±1:27	4:22–7:16
Tutanota	Both	24	9:41	3:54	±1:34	8:07–11:15

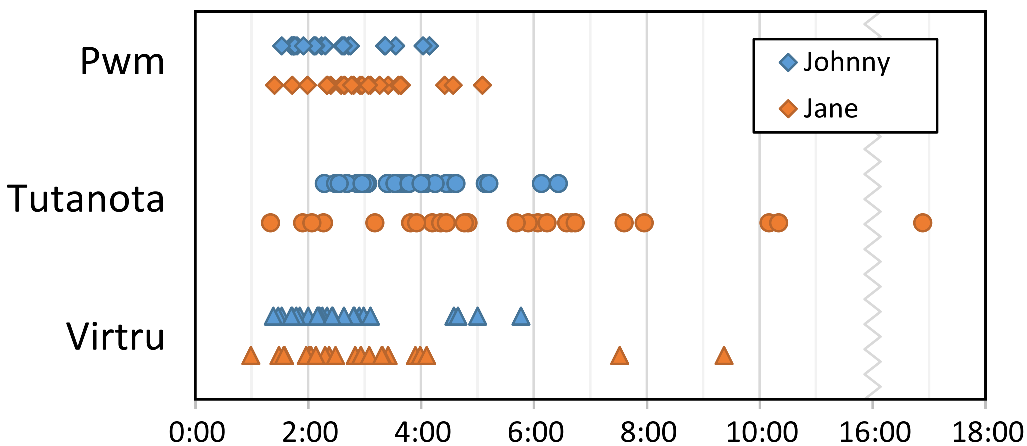


Fig. 2. Individual Participant Task Completion Times

Stage 1–$p = 0.58$, Stage 2–$p = 0.85$, Total–$p = 0.61$).[21] Participants using Tutanota took roughly one minute longer to complete Stage 1 and almost three minutes longer to complete Stage 2. The differences between Tutanota and Pwm/Virtru in Stage 1, Stage 2, and the combined times are all statistically significant (two-tailed student t-test, matched pairs—each case–$p < 0.001$)

5.3. Mistakes

We defined mistakes as any situation in which sensitive information was sent in plaintext or was sent encrypted along with the key to decrypt the sensitive information (i.e., the Tutanota shared secret was sent as plaintext in email). Using Pwm, no participants sent their sensitive data in the clear. With Virtru, only a single participant sent their sensitive information in the clear. In this case, the participant had entered the sensitive information into an unencrypted greeting field that Virtru allows participants to include with an encrypted email.

In contrast, participants were much more likely to make mistakes with Tutanota. Two-thirds of the participant pairs (68%, n=17) communicated the Tutanota shared secret over clear text in email. Additionally, half of the participant pairs (48%, n=12) selected shared secrets that had low entropy and could be quickly guessed by a password-cracking system.

While Tutanota clearly performed worse than Pwm and Virtru, care should be taken in analyzing this result. In the post-study interview several participants indicated that while they had transmitted

[21]After adjusting the data for the higher number of treatments in which Virtru was the first system tested, we found that on average Johnny's task completion time dropped by 15 seconds for Virtru and increased by 10 seconds for Pwm. These differences are not statistically significant from the non-adjusted data, and the difference between Virtru and Pwm in the adjusted data is not statistically significant.

Table III. Participants' Favorite Systems

	Johnny	Jane	Total
Pwm	48%	60%	54%
Virtru	44%	28%	36%
Tutanota	4%	12%	8%
Disliked All	4%	0%	2%

their password over email in the study, they stated that in the real world they would be more likely to send the data over a different channel.

5.4. Favorite System

At the end of the study, participants were asked which of the three systems was their favorite. Their responses are summarized in Table III. Pwm was most frequently rated as the best system, with Virtru also rated highly. Tutanota was rarely selected as the best system, and one participant indicated that they disliked all of the systems. These results roughly correlate with the SUS score of each system.

Interestingly, we do see a difference in the choice of favorite system based on what role the participant played. While Pwm and Virtru are rated as the favorite system about equally by Johnny, Pwm was most often selected as the favorite system by Jane. Based on participant responses, this disparity is due to the fact that unlike Johnny, Jane had to leave Gmail to interact with Virtru messages, a process that was frequently described negatively.

Similarly, Tutanota was more highly rated by Jane than by Johnny. Participant responses reveal that this is likely due to the fact that Jane did not have to go through the Tutanota account setup (which required a long, complex password) and selection of a shared secret for the email (which caused nearly all participants to struggle).

5.5. Differences Based on Treatment

In Atwater et al.'s work [Atwater et al. 2015], they noticed that the order in which users tested systems strongly affected the SUS scores for those systems. We analyzed our results to determine if we saw a similar affect, but found that the order in which the systems were tested had no effect on SUS scores. This is in line with our prior experience using SUS to evaluate secure email systems [Ruoti et al. 2013].

We also evaluated the data to see if the order in which users tested systems affected task completion time, number of mistakes, or participants' favorite systems. The only measurable difference was in task completion times for Pwm and Virtru: whichever system was tested first would have a longer-than-average task completion time, and whichever system was tested second would have a shorter-than-average task completion time. This difference in times was not statistically significant, but this is likely because the sample population was too small.

6. SECOND STUDY–METHODOLOGY

Using the same methodology as the first study, we conducted a second IRB-approved user study wherein pairs of participants used Mailvelope to transmit sensitive information to each other.

6.1. Study Setup

The study ran for two weeks, beginning Tuesday, September 8, 2015 and ending Friday, September 18, 2015. In total, 10 pairs of participants (20 total participants) completed the study. Unlike the first study, participants only used a single system, Mailvelope. Participants were allocated sixty minutes to complete the study, with about 35-40 minutes spent using Mailvelope.

6.2. Demographics

We recruited Gmail users for our study at a local university. Participants were two-thirds male: male (13; 65%), female (7; 35%). Participants skewed young: 18–24 years old (18; 90%), 25–34 years old (2; 10%).

We distributed posters broadly across campus to avoid biasing our results to any particular major. All participants were university students,[22] with the majority being undergraduate students: under-

[22]We did not require this.

Table IV. Mailvelope SUS Scores

	Count	Mean	Standard Deviation	Confidence Interval ($\alpha = 0.05$)	Range
Johnny	10	30.5	16.6	± 10.3	20.2–40.8
Jane	6	41.3	10.9	± 8.7	32.6–50.0
Both	16	34.5	15.3	± 7.5	27.0–42.0

graduate students (17; 85%), graduate students (3; 15%). Participants were enrolled in a variety of majors, including both technical and non-technical majors. No major was represented by more than four participants, with the vast majority only having one or two participants.

6.3. Limitations

Our study only included twenty participants, all of whom were students. While this was enough to show difficulties associated with Mailvelope it is not indicative of all possible outcomes. It would be especially interesting to rerun this study using different populations (e.g., technical professionals, computer scientists, security professionals).

7. SECOND STUDY–RESULTS

Overall, participants were unable to use Mailvelope to send encrypted email, with only one in ten participant pairs completing the assigned task. This is in stark contrast to the results of our first study in which all participant pairs completed the assigned task. As such, it is clear that Mailvelope is not suitable for helping novices send secure email between themselves.

In the remainder of this section, we detail Mailvelope's failure rate, report on its SUS scores, and list mistakes made by participants.

7.1. Failures

The study lasted an hour, with roughly 45 minutes allocated to complete the assigned task. If, after installing Mailvelope, Johnny made absolutely no progress for 30 minutes, study coordinators were instructed to end the task and continue to the post-study interview. If instead participants were making some progress, study coordinators would allow them to continue until there were ten minutes left in the hour, reserving the last ten minutes for the post-study interview.

Of the ten participant pairs, nine were unable to successfully complete the task. In two of those nine pairs, Johnny never figured out how to use Mailvelope to send any message.[23] In another two pairs, Jane was completely mystified by the encrypted PGP message and was unaware that she needed to install Mailvelope to read it. Only one of the nine pairs actually traded public keys, though this pair was still confused about what to do after sharing their public keys.

The one pair that completed the task actually required more than the full forty-five minutes to do so. At their request, we allowed 10 extra minutes to complete the task. We did so because this was the only pair that appeared close to finishing the task and we were interested in observing a successful trial. Interestingly, in this pair, Jane indicated having had previously learned about public key cryptography in a college class and attributed their success to this prior knowledge. Based on our observation of this pair, we agree that Jane's knowledge of public key cryptography was instrumental to the pair's success at completing the assigned task.

7.2. System Usability Scale

We evaluated Mailvelope using the System Usability Scale (SUS). While Johnny always completed the SUS evaluation for Mailvelope, in four instances Jane never progressed far enough in the assigned task to install the system, and so we did not have her complete the SUS questions. A breakdown of the SUS score for each type of participant is given in Table IV. Similar to the first study, the difference between Johnny and Jane's SUS scores were not statistically significant (two-tailed student t-test, equal variance—$p = 0.18$).

[23] In these two instances the study was stopped after 30 minutes as no progress had been made by Johnny. The other eight pairs were given the full 45 minutes to complete the task.

Mailvelope's SUS score of 34.5 is rated as having "Poor" usability. It falls in the 4^{th} percentile, is given a letter grade of "F" and is labeled as "Not acceptable." The differences between Mailvelope and the three systems tested in the first study are all statistically significant (two-tailed student t-test, equal variance—$p < 0.001$)

7.3. Mistakes

All participant pairs made mistakes. The most common mistake was encrypting a message with the sender's public key. This occurred for seven of the participant pairs, including the participant pair that was eventually successful. Three of the participant pairs generated a key pair with their friend's information, and then tried to use that public key to encrypt their message. One participant modified the encrypted message after encryption (while still in the PGP compose window), adding their sensitive information to the area before the PGP block. Finally, one participant eventually exported his private key and sent it along with his key ring password to his friend so that his friend could decrypt the message he had received. In this case, even though the participants had transmitted the required information, they were informed that they needed to try some more and accomplish the task without sending the private key.

8. DISCUSSION

In this section we discuss themes that we noticed across both studies, especially the qualitative feedback provided by participants on the study survey and in the post-study interview. Participants in the first study have been assigned a unique identifier R[1–25][A,B], while participants in the second study have been assigned a unique identifier M[1–10][A,B]. The final letter refers to which role the participant played during the study, and participants with the same number were paired with each other (e.g., R1A and R1B were Johnny and Jane, respectively, in the same study session).

8.1. Paired-participant Methodology

During the studies, we noticed several clear benefits of our paired-participant methodology.

First, by having participants play different roles, we were able to gather data about users' experiences both when they are introduced to secure email and when someone else is introducing them. For example, in Tutanota, messages need to have a shared secret to be encrypted. Johnny's experiences revealed the difficulty in discovering that a password is required and that it needs to be communicated to the recipient, Jane. Similarly, Jane's experience showed the aversion participants felt to leaving their current email system to view a sensitive message. While these same experiences might have been elicited by running two different studies, it was convenient to obtain them in a single study, and it was helpful to be able to correlate the experiences of participant pairs. Furthermore, showing that a participant can successfully use a new secure email system when inducted by another novice user is stronger than only showing that a new user can be inducted by an expert.

Second, our study design led to more natural behaviors by participants. In past studies, we observed that participants expected study coordinators to immediately respond to emails. Even after being informed that a response would take a couple of minutes, participants would constantly refresh their inbox to see if a message had arrived, and if a response took longer than fifteen to thirty seconds to arrive participants would often complain. In contrast, participants in our studies were content to wait to receive their email and did not appear agitated when their friends took a long time to respond. Also, instead of constantly refreshing their inbox, participants would browse the Web or check their phones, which is likely more representative of how they use email in practice.

In addition to the observations by study coordinators, participants also noted that they felt more natural interacting with a friend than with a study coordinator. For example, participants R24B and R25A stated, respectively,

> *"I was more at ease probably than I would've been if it was someone random on the other end.. . . It would've felt more mechanical, robotic, whereas I know [her] and I was calling my wife, 'Hi wife! What's the password?' It felt a lot more personable for me I think."*

> *"It was good in that you saw the troubles, like the third system [Tutanota], I didn't even know how it worked, so I ended up sending an email to myself on Gmail so then I could see what was happening on her end, to know like how it works on the other end. So I think it's good to*

have two people on each end that don't know what's going on, because if it weren't I'd assume the person on the other side had done it before."

Third, participants indicated that because they were working with their friends, they felt more relaxed. For example, participants R11B and R14B indicated, respectively,

"I thought it was good, I dunno, might've taken the pressure off too, where it's like, 'Okay, he's figuring this out too', so I can just, y'know, I don't have to feel as 'under-the-microscope' in the study."

"I felt like neither of us knew what we were doing, but if I knew that someone else knew what was going on, I'd be like, 'K, hopefully I'm not doing it wrong,' so it's kind of like, 'K, we're on the same page, neither of us know what we're doing.'"

Fourth, we were pleased to note that requiring participants to bring a friend with them resulted in a much lower missed-appointment rate than we have seen in the past.

Based on our observation of participants' behavior and the participants' qualitative feedback, we feel that there is significant value in conducting two-person studies. Still, future research should examine in greater depth the differences between one- and two-person studies. For example, an A/B study comparing these two methodologies could be conducted which compares differences in system metrics (e.g., SUS, task completion time) as well measures differences in users' agitation during the study (e.g., heart rate, eye tracking). Similarly, research could compare how participant experiences differ when both roles are filled by a novice, as opposed to having one simulated by a coordinator.

8.2. Hidden Details and Trust

Providing further evidence to prior work [Fahl et al. 2012; Ruoti et al. 2013], participants' experiences demonstrated that when security details were hidden from them, they were less likely to trust the system. This was most clearly demonstrated when examining Pwm and Virtru. Both systems use email-based identification and authentication [Garfinkel 2003] to verify the user's identity to a key escrow server (i.e., they send an email to the user with a link to click on to verify their identity). The difference is that while Virtru requires users to manually open this email and click on the link, Pwm performs this task automatically for users. While this difference might seem small, it was cause for concern for several participants. For example, participants R6B and R10B expressed, respectively,

"I liked the way that one [Virtru] and the last one [Tutanota] both had ways to confirm that it was you and no one else could see the information."

*"(Interviewer: But you didn't think that [Pwm] was as secure [Virtru]?) [Pwm] **said** that it was, but I liked how the other ones had additional 'send-you-an-email' verification or a password between you and the other person in the email. Just an added measure to feel like there really is something different. 'Cause Pwm for all I know, like, I'm just taking their word for it. There's not really anything extra that shows that it really is secure."*

In contrast, the shared secret used by Tutanota made it clear that only the recipient who had the password would be able to read the message. This made a large number of participants feel that Tutanota was the most secure system, even if usability issues prevented it from being their favorite system. R17B's response demonstrates this principle:

*"Like the order of the programs was interesting 'cause I thought the Virtru one was great, like until I saw this [Tutanota], 'Oh, this one [Tutanota] requires a password—why did I think that one [Virtru] was great?' And I wish, it **would** have required a password because anybody that has your email password can just see [everything]."*

The sentiment regarding passwords was so strong that several participants stated that they wished Pwm and Virtru would also allow them to password-encrypt messages. For example, R17B, R10A, and R10B expressed, respectively,

"I like that [Pwm] encrypted the info so that Gmail couldn't read it. I think Pwm would be the best one if it required passwords."

*"R10B: I would say exactly the second one [Pwm], just with a password (R10A: 'Yeah') per conversation is what I would do. Just because it's so simple, right there. R10A: Stay on Gmail, but then have a password to get to **that** encrypted email. (R10B: 'Yeah')"*

Still, not all participants were enamored with using a shared password, seeing it as an added memory burden or hassle. As stated by participants R10A, R5A, and R25A, respectively,

"I will never remember my crazy password."

"I don't know if I loved the password idea, just because if I was sending a secure password over something, then why didn't I just send the information over that anyways?"

"How do you send a password safely if your encrypting program requires a password?"

Some participants were concerned that it was impossible to verify if any of the systems were truly encrypting their data. This likely stems from two facts: first, that participants are not security experts and lack the means to truly verify the security of a tool, and second, that the tools themselves—once working—never show the user any indication that they are actually receiving encrypted email. While results from Atwater et al. suggest that showing ciphertext does not address this issue [Atwater et al. 2015], the fact that participants are concerned indicates that this problem needs more research. For example, participants R14A and R17B stated, respectively,

"It would be kind of cool to see what it would look like as an encrypted message.... Seems kind of weird. Like 'it's encrypted now, trust us.'"

"I would like to know exactly how the encryption happens—I understand that it is encrypting it, but how do I know it's completely safe?... There are too many programs that are not what they seem, and I would not want this to be one of those."

8.3. Integrated vs Depot

Participants overwhelmingly preferred secure email to be integrated into their existing email systems and not require a second account (i.e., a depot). This preference was expressed through the low SUS scores of Tutanota and the fact that only four participants rated it as their favorite system. Additionally, participant comments made it clear that they were not interested in using depot-based secure email. For example, participants R10A, R25A, and R16A respectively stated,

"I hate having so many emails. Gmail is enough for me."

"No one wants another email system."

"It is just not my type. I don't want to set up another account and send a password to my friend."

However, several participants felt that Tutanota was more secure than other systems, precisely because it required the creation of an account separate from Gmail. Participants noted that in Pwm and Virtru, access to the user's Gmail account was all that was required to decrypt sensitive email. For example, participants R4B, R25A, and R25B stated, respectively,

"[I like it,] I dunno, just because I leave my email up a lot, someone could just go on to my email and look at it. I don't sign out of my email."

*"I just kinda feel like anybody could go into your email and look at those secure ones if it **is** inside your email."*

"How strong is your Gmail password, you know? If you can get in there, then it defeats these other encryption. So, really, you're just trying to hide your stuff from Google, which, they already know everything, so."

Users' preference for integrated secure email is also shown in participant interactions with Virtru. When users have Virtru installed on their machine, they can read and compose messages within Gmail. In contrast, when non-Virtru users receive a Virtru-encrypted message, the message does not prompt recipients to download and install the Virtru plugin but instead takes them to an external webpage with message-depot functionality, bypassing the Gmail integration that participants were so fond of.

This was disliked by several participants, with participant R13B stating, *"I don't like being taken to another website to send a message in Gmail. I would prefer to just stay in Gmail."*

8.4. Tutorials

Tutorials were a significant factor in participants' experiences. Pwm was rated by participants as having the best tutorials, with a fourth of participants (24%, n=12) bringing up tutorials when asked what they liked about Pwm. Participants largely liked the style of the tutorials as well as their content. For example, participant 8B expressed, *"I also really liked the tutorial. It was similar to tutorials Apple or Google/Gmail give you to learn things."*

Virtru also has tutorials, but praise for these tutorials was not as common as it was with Pwm, with Jane participants criticizing the tutorials more than Johnny participants. This result can likely be attributed to the fact that the Virtru plugin walks new users through a tutorial upon installation, but someone who receives a Virtru-encrypted message without the plugin is simply presented with a blue button labeled "Unlock message" without additional instruction beyond what the sender of the email has personally and manually added. This is in contrast to Pwm, which prefaces incoming encrypted email with instructions on what encrypted email is and how the recipient should go about decrypting the message.

Tutanota had no tutorials, and this clearly led to confusion. Nearly all participants failed to notice that they needed to set a password to encrypt their email, and just as many didn't realize that they needed to communicate this password to the other participant. Additionally, some participants didn't understand that they couldn't just use Tutanota to communicate the password Many of these problems could have been alleviated by a simple tutorial.

8.5. Reasons to Use Encrypted Email

The majority of participants (72%) agreed with the survey statement, "I want to be able to encrypt my email," although only a much smaller fraction (20%) agreed with the idea that they would "encrypt email frequently." Still, when asked to describe how they would use encrypted email in practice, many participants were unsure. The range of opinions are summarized in responses from participants R22A, R24A, R20A, and R23B, respectively, on how they would use secure email in practice:

> *"Um, I've never really used it before because I didn't know it was so accessible through Gmail so now that I know I can, I will use it more often. (Interviewer: Do you actually plan to use it?) Yeah, I will."*

> *"Like, just the other day I needed [my husband's] social security card number for something and then didn't feel like there was any way I could ask him and if I had known about this, I would have done that."*

> *"Well, I'm trying to think when I would need to. It would be nice to have it, in case, but I don't know if there's anyone I would need to send that information to."*

> *"Knowing that I could encrypt email I probably could find uses for it. . . "*

These responses indicate that more research needs to be done to discover under what circumstances users would employ secure email.

8.6. Guidelines for Usable PGP

Mailvelope clearly failed to help the majority of participants encrypt their email. All participants expressed frustration with Mailvelope, with the most comical expression of this frustration coming from M3A: *"Imagine the stupidest software you would ever use, and that was what I was doing."* The difficulty also led several participants to indicate that in the real world they would have given up trying to use Mailvelope long before they did during the study. For example, M3A also said, *"After five minutes, I would have just given up and called."*

While our study's results are specific to Mailvelope, our experimentation with other PGP-based tools suggest that Mailvelope is at least as usable as those other tools. As such, it is still an open question whether PGP-based secure email can be made sufficiently usable for the masses. Still, there are clearly some areas where existing PGP-based secure email tools could be improved. Based on our analysis of all four systems studied and participant responses, we have derived the following guidelines that we believe will help PGP-based secure email tools be significantly more usable for novice users.

8.6.1. Integrated Tutorials. When using Mailvelope, participants were constantly flipping between Mailvelope's website and Gmail, looking for instructions on what to do next. At no stage was it intuitive how they should proceed based on Mailvelope's UI. Nearly all participants indicated that they wished Mailvelope had provided instructions that were integrated with the Mailvelope software, and would walk them through, step-by-step, in setting up Mailvelope and sending their first encrypted email. As seen for Pwm and Virtru, tutorials likely could greatly assist first-time users in acclimating to PGP.

Important steps that could be addressed by tutorials are: (1) helping participants generate their PGP key pair, (2) discussing how to share public keys, (3) inviting their friends to set up a secure email tool, (4) importing their friend's public keys, (5) sending their first encrypted email, and (6) decrypting their first encrypted email.

8.6.2. Explanation of Public Key Cryptography. The only participant pair that successfully completed the study task likely did so because one of the participants in the pair had previous knowledge related to public key cryptography. Additionally, the only other pair that made progress did so because they realized that they needed each other's public keys, but even that pair did not know how to then use those shared public keys. For the remaining eight participant pairs, the post-study interview made it clear that they did not understand how public and private keys were used.

To help address this, a simple explanation of PGP needs to be developed that is accessible to the masses. Several participants indicated that they would prefer these concepts to be displayed in a simple video. Ideally, whatever form the description takes, it would be integrated with the tutorials, allowing new users to be introduced to public key cryptography as a natural extension of their usage of the secure email tool.

8.6.3. Automatic Key Generation. Johnny participants struggled to generate their own PGP key pair, with much of the confusion tied to their lack of understanding regarding public and private keys. Often participants were unsure whose information they should input into the key generation dialog, their own or that of the intended participants.[24] An easy way to address this problem would be to automatically generate a user's private key, retrieving necessary information from the webmail provider (i.e., name and email address), and during installation only prompt the user for the private key's encryption password.

8.6.4. Better Text to Accompany PGP Block. During Johnny's attempts to send Jane an encrypted email, Johnny often encrypted a message for himself, and then sent that encrypted message to Jane. Upon seeing the PGP ciphertext block, Jane was unclear what she was supposed to do with it. One participant noted that she thought it was an image that had gotten garbled during email transmission.

While Johnny had obviously made a mistake, it also represented an opportunity for Jane to recognize that Johnny was trying to use secure email. To take better use of this opportunity, the PGP ciphertext block could be modified to include plaintext instructions detailing the nature of the encrypted email, how to obtain a PGP-based secure email tool, and how to start sending encrypted email [Ruoti et al. 2013]. While this wouldn't allow Jane to read the email from Johnny, it would allow her to better collaborate with Johnny in discovering how to use secure email. An indication that this approach could be successful is given by participant M9B, the only Jane participant who finished the study task. In referring to the PGP ciphertext block, she said, *"It was like a puzzle, I only got a link to Mailvelope. I then had to go there and explore."*

8.6.5. Automatic Key Discovery and Email Invites. Similarly, Johnny participants were confused about what Jane needed to do in order to receive an encrypted email. Much of this confusion was centered around how to share keys. To address this, we recommend the use of automated key discovery—for example, the key lookup service provided by keybase.io or other public key directories. While there are security implications to this approach, based on our studies we feel that this would addresses a significant hurdle for adoption of PGP-based secure email. This is corroborated in recent work by Bai et al. [Bai

[24]This was only compounded by the fact that Mailvelope showed users their own public key in the list of "recipients" the message was encrypted for.

et al. 2016], which found that users consider automatic key discovery more usable than manually exchanging keys.

We also recommend that PGP clients detect when recipients don't have a public key, and help the sender take the appropriate steps to resolve this problem. For example, a PGP client could generate an email for the recipient stating that the sender wants to communicate with them using PGP. This generated email could also include instructions on how to set up the PGP client. This technique was first explored by Atwater et al. [Atwater et al. 2015], and our experience leads us to strongly recommend it.

9. CONCLUSION

In this work, we conducted the first two-person study of secure email where two novice users are brought into the lab together and asked to exchange secure email between themselves. Our study analyzed Pwm, Tutanota, Virtru, and Mailvelope. Using a two-person study enabled us to see participants under different first-use experiences. In addition, participants exhibited more natural behaviors, seemed less agitated, and indicated that they felt less like they were "under the microscope."

Our results indicate several observations about secure email systems. First, we found further evidence that hiding security details can lead to a lack of trust in secure email systems. This gives further credence to similar results from earlier work [Ruoti et al. 2013]. Second, we found that participants largely rejected depot-based secure email systems. Third, participant success in using a system without mistakes is heavily influenced by the presence of well-designed tutorials. Lastly, while participants are interested in using secure email, few express a desire to use it regularly and most are unsure of when or how they would use it in practice.

Our results also demonstrate that after two and half decades, PGP-based secure email is still unsuitable for novice users. Still, in comparing results from the successful secure email systems (Pwm and Virtru) with results from Mailvelope, we derived several guidelines to help make PGP tools more suitable for novice users.

REFERENCES

Erinn Atwater, Cecylia Bocovich, Urs Hengartner, Ed Lank, and Ian Goldberg. 2015. Leading Johnny to Water: Designing for Usability and Trust. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*. USENIX Association, Montreal, Canada, 69–88.

Wei Bai, Moses Namara, Yichen Qian, Patrick Gage Kelley, Michelle L. Mazurek, and Doowon Kim. 2016. 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)*. USENIX Association, Denver, CO, 113–130.

Aaron Bangor, Philip Kortum, and James Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction* 24, 6 (2008), 574–594.

Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining What Individual SUS Scores Mean: Adding An Adjective Rating Scale. *Journal of Usability Studies* 4, 3 (2009), 114–123.

John Brooke. 1996. SUS—A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*. CRC Press, Boca Raton, FL.

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

Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J. Alex Halderman. 2015. Neither Snow Nor Rain Nor MITM…: An Empirical Analysis of Email Delivery Security. In *Fifteenth ACM Internet Measurement Conference (IMC 2015)*. ACM, Tokyo, Japan, 27–39.

Sascha Fahl, Marian Harbach, Thomas Muders, Matthew Smith, and Uwe Sander. 2012. Helping Johnny 2.0 to Encrypt His Facebook Conversations. In *Eighth Symposium on Usable Privacy and Security (SOUPS 2012)*. ACM, Washington, D.C., 11.

Ian D. Foster, Jon Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko. 2015. Security by Any Other Name: On the Effectiveness of Provider Based Email Security. In *Twenty-Second ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*. ACM, Denver, CO, 450–464.

Electronic Frontier Foundation. 2015. Secure Messaging Scorecard. (2015). Accessed Sep 25, 2015. https://www.eff.org/secure-messaging-scorecard.

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

Simson L. Garfinkel. 2003. Email-based Identification and Authentication: An Alternative to PKI?. In *Twenty-Fourth IEEE Symposium on Security and Privacy (S&P 2003)*. IEEE Computer Society, Oakland, CA, 20–26.

Simson L. Garfinkel and Robert C. Miller. 2005. 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)*. ACM, Pitsburg, PA, 13–24.

Shirley Gaw, Edward W Felten, and Patricia Fernandez-Kelly. 2006. Secrecy, Flagging, and Paranoia: Adoption Criteria in Encrypted Email. In *SIGCHI Conference on Human Factors in Computing Systems (CHI 2006)*. ACM, Montreal, Canada, 591–600.

Ralph Holz, Johanna Amann, Olivier Mehani, Matthias Wachs, and Mohamed Ali Kaafar. 2016. TLS in the Wild: An Internet-wide Analysis of TLS-based Protocols for Electronic Communication. In *Twenty-Fourth Network and Distributed System Security Symposium (NDSS 2016)*. The Internet Society, San Diego, CA.

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

B. Ramsdell and S. Turner. 2010. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751 (Proposed Standard). (January 2010). http://www.ietf.org/rfc/rfc5751.txt.

Karen Renaud, Melanie Volkamer, and Arne Renkema-Padmos. 2014. Why Doesn't Jane Protect Her Privacy?. In *Fourteenth Privacy Enhancing Technologies Symposium (PETS 2014)*. Springer, Philadelphia, PA, 244–262.

Scott Ruoti, Jeff Andersen, Travis Hendershot, Daniel Zappala, and Kent Seamons. 2016. Private Webmail 2.0: Simple and Easy-to-Use Secure Email. In *Twenty-Ninth ACM User Interface Software and Technology Symposium (UIST 2016)*. ACM, Tokyo, Japan.

Scott Ruoti, Nathan Kim, Ben Burgon, Timothy Van Der Horst, and Kent Seamons. 2013. Confused Johnny: When Automatic Encryption Leads to Confusion and Mistakes. In *Ninth Symposium on Usable Privacy and Security (SOUPS 2013)*. ACM, Newcastle, United Kingdom.

Scott Ruoti, Brent Roberts, and Kent Seamons. 2015. Authentication Melee: A Usability Analysis of Seven Web Authentication Systems. In *Twenty-fourth International Conference on World Wide Web (WWW 2015)*. ACM, Florence, Italy, 916–926.

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

Adi Shamir. 1984. Identity-based Cryptosystems and Signature Schemes. In *Fourteenth International Cryptology Conference (Crypto 1984)*. Springer, Santa Barbara, CA, 47–53.

S. Sheng, L. Broderick, C.A. Koranda, and J.J. Hyland. 2006. Why Johnny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software. In *Poster Session at the Symposium On Usable Privacy and Security*. Pitsburg, PA.

Yuanzheng Song. 2014. *Browser-based Manual Encryption*. Master's thesis. Brigham Young University, Provo, UT.

Andreas Sotirakopoulos, Kirstie Hawkey, and Konstantin Beznosov. 2010. "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.

Wenley Tong, Sebastian Gold, Samuel Gichohi, Mihai Roman, and Jonathan Frankle. 2014. Why King George III can encrypt. (2014). Unpublished.

Thomas S. Tullis and Jacqueline N. Stetson. 2004. A Comparison of Questionnaires for Assessing Website Usability. In *Usability Professional Association Conference*. Usability Professionals Association, Minneapolis, MN, 1–12.

Alma Whitten and J.D. Tygar. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Eighth USENIX Security Symposium (USENIX Security 1999)*. USENIX Association, Washington, D.C., 14–28.

A. SYSTEM SCREENSHOTS

Full resolution screenshots can be found at https://tops2016.isrl.byu.edu.

A.1. Pwm

Fig. 3. Pwm's Integrated Tutorial

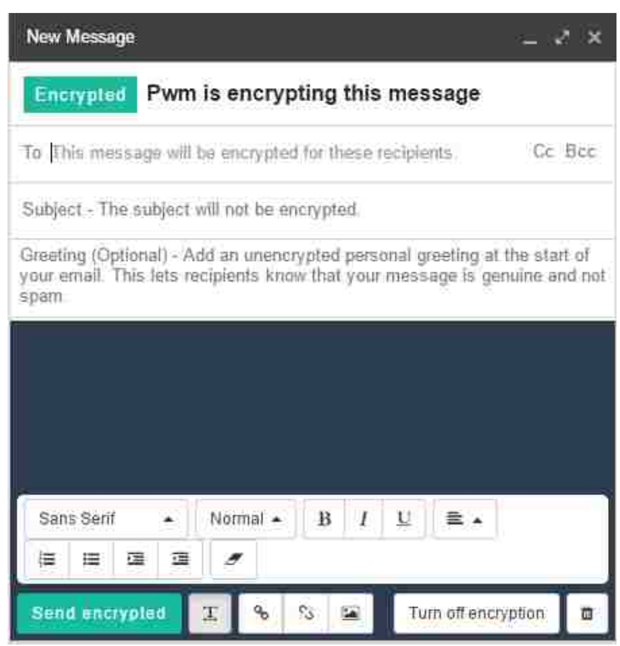


Fig. 4. Pwm's Secure Composition Interface

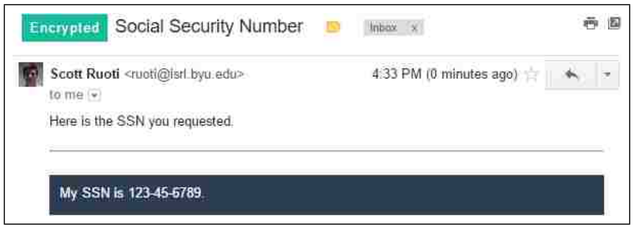


Fig. 5. Pwm's Secure Read Interface

A.2. Tutanota

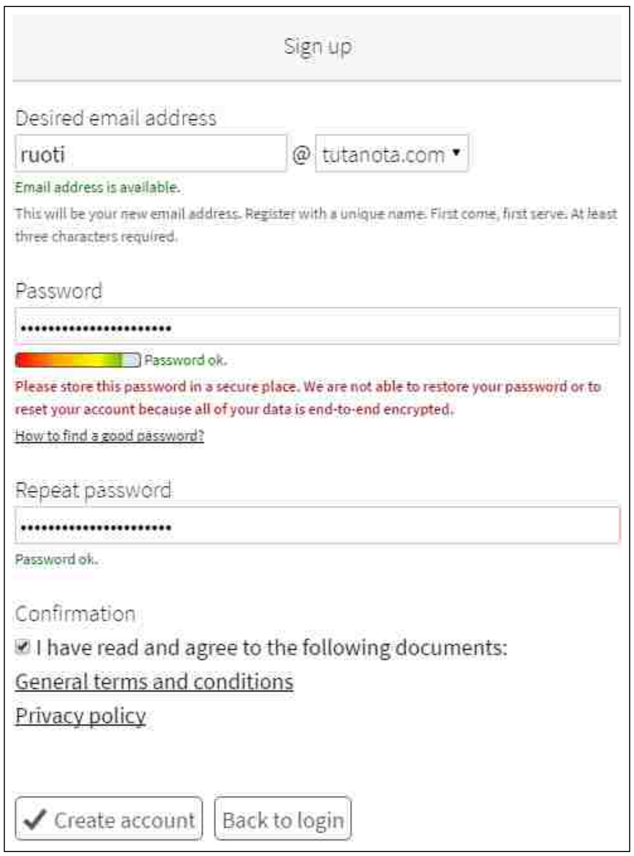


Fig. 6. Tutanota's Sign Up Page

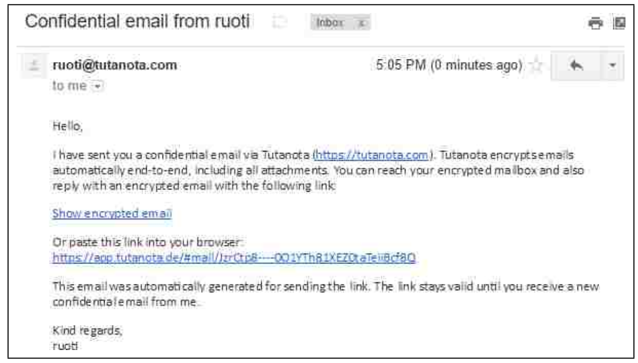


Fig. 8. Tutanota's Password-Encrypted Email

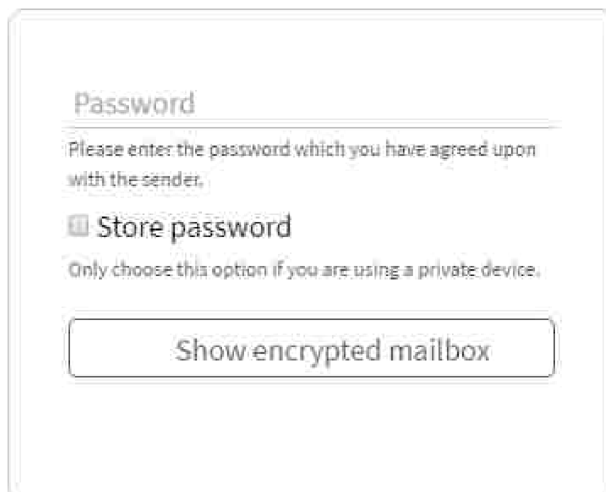


Fig. 9. Tutanota's Password-Encrypted Email Password Entry Interface

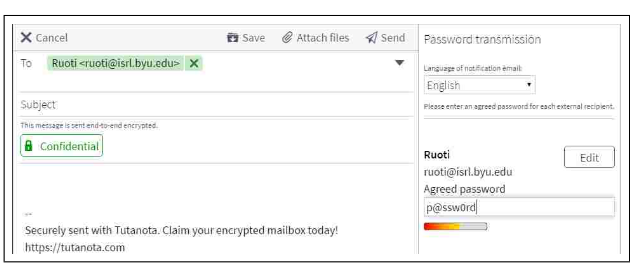


Fig. 7. Tutanota's Secure Compose Interface

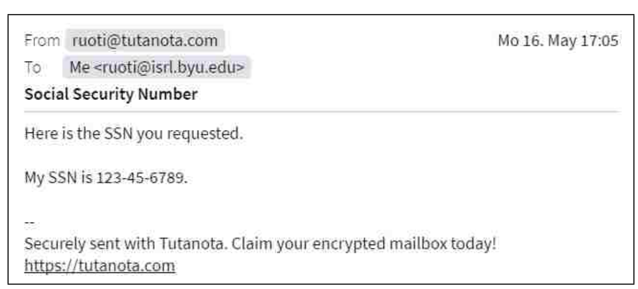


Fig. 10. Tutanota's Secure Read Interface

A.3. Virtru

Fig. 11. Virtru's Integrated Tutorial

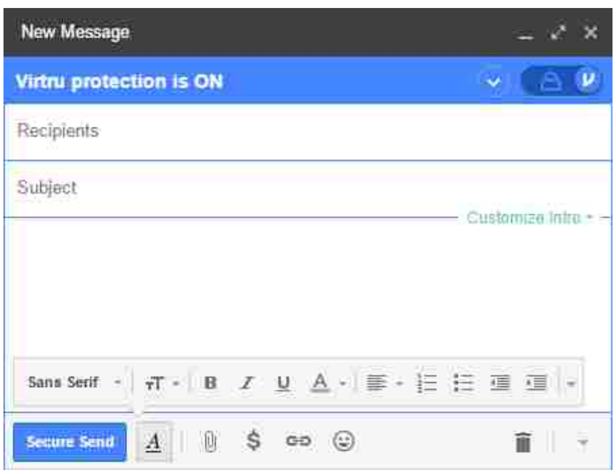


Fig. 12. Virtru's Secure Composition Interface

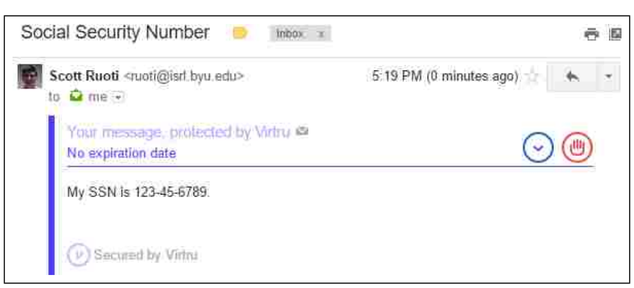


Fig. 13. Virtru's Integrated Secure Read Interface

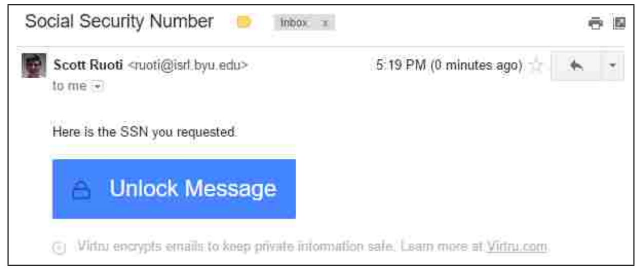


Fig. 14. Virtru's Depot-based Encrypted Email

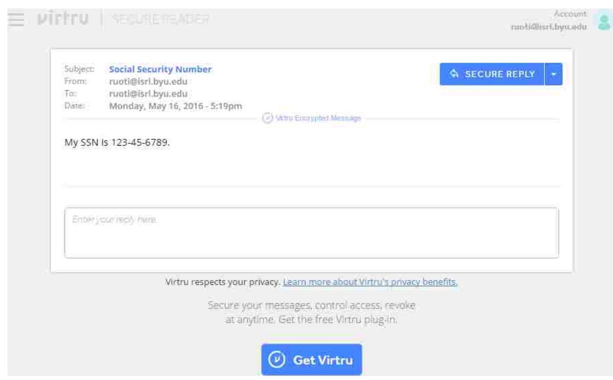


Fig. 15. Virtru's Depot-based Secure Read Interface

A.4. Mailvelope

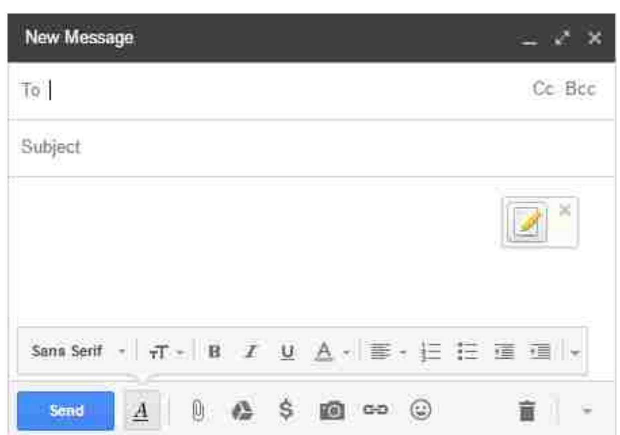


Fig. 16. Mailvelope's Enable Encryption Button

Fig. 17. Mailvelope's Secure Composition Interface

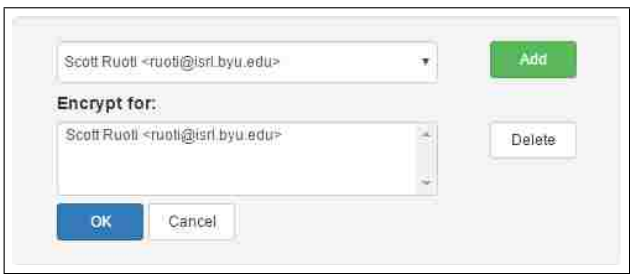


Fig. 18. Mailvelope's Public Key Selection Interface

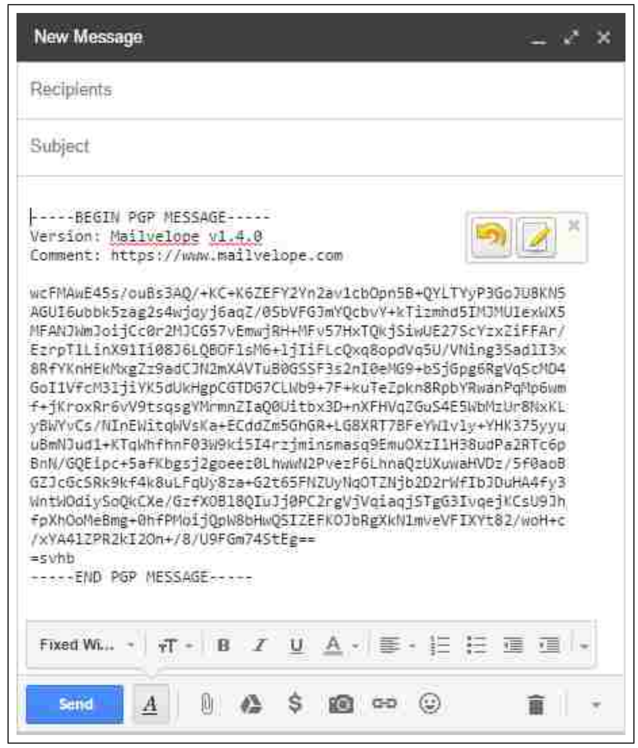


Fig. 20. Mailvelope's Encrypted Message in the Webmail Provider's Compose Interface

Fig. 19. Mailvelope's Transfer Encrypted Message Interface

Fig. 21. Mailvelope's Encrypted Message in the Webmail Provider's Read Interface

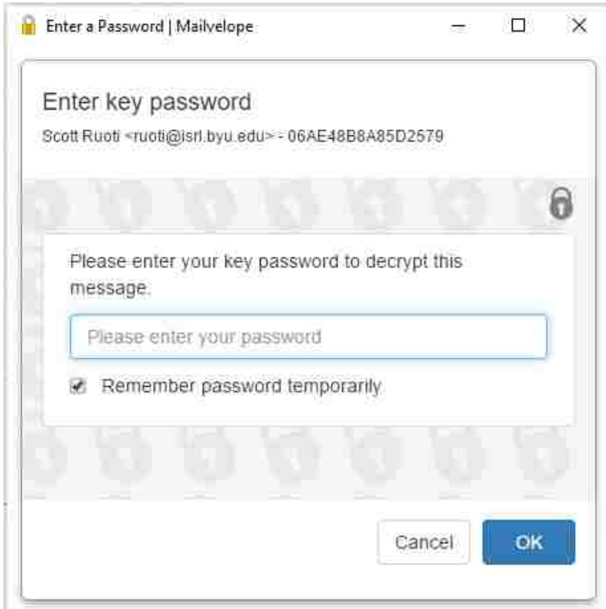


Fig. 22. Mailvelope's Private Key Password Entry Interface

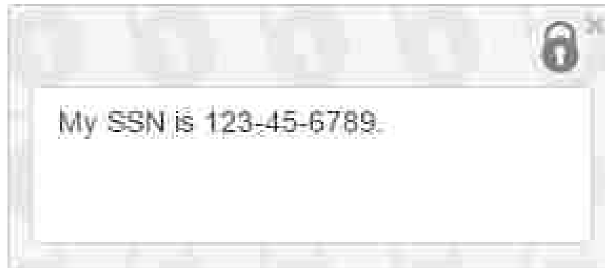


Fig. 23. Mailvelope's Secure Read Interface

MessageGuard: A Browser-Based Platform for Usable, Content-Based Encryption Research

Scott Ruoti*†, Jeff Andersen*, Tyler Monson*, Daniel Zappala*, Kent Seamons*
Brigham Young University, Sandia National Laboratories†*
{*ruoti, andersen, monson*}*@isrl.byu.edu*, {*zappala, seamons*}*@cs.byu.edu*

***Abstract*—This paper describes MessageGuard, a browser-based platform for research into usable content-based encryption. MessageGuard is designed to enable collaboration between security and usability researchers on long-standing research questions in this area. It significantly simplifies the effort required to work in this space and provides a place for research results to be shared, replicated, and compared with minimal confounding factors. MessageGuard provides ubiquitous encryption and secure cryptographic operations, enabling research on any existing web application, with realistic usability studies on a secure platform. We validate MessageGuard's compatibility and performance, and we illustrate its utility with case studies for Gmail and Facebook Chat.**

1. Introduction

Users share private information on the web through a variety of applications, such as email, instant messaging, social media, and document sharing. HTTPS protects this information during transmission, but does not protect users' data while at rest. Additionally, middleboxes can weaken HTTPS connections by failing to properly implement TLS or adequately validate certificate chains [1]. Even if a website correctly employs HTTPS and encrypts user data while at rest, the user's data is still vulnerable to honest-but-curious data mining [2], third-party library misbehavior [3], website hacking, protocol attacks [4], [5], [6], and government subpoena.

This state of affairs motivates the need for *content-based* encryption of user data. In content-based encryption, users' sensitive data is encrypted at their own computer and only decrypted once it reaches the intended recipient, remaining opaque to the websites that store or transmit this encrypted data. The best known examples of content-based encryption are secure email (e.g., PGP, S/MIME) and secure instant messaging (e.g., OTR). In addition to protecting communication, content-based encryption can protect any data users store or distribute online; for example, files stored in the cloud (e.g., DropBox, Google Drive) or tasks in a web-based to-do list.

Unfortunately, while there is a large body of work on messaging protocols and key management schemes for content-based encryption [7], too little of it has been subjected to formal usability evaluation. This is problematic as experience has shown that many proposals with strong theoretical foundations are either unfeasible in real-world situations [8] or have problems that are only apparent when studied empirically [9], [10], [11]. As a result, there are many open HCI and security questions regarding content-based encryption [12].

1.1. MessageGuard

To address these concerns, we have developed MessageGuard, a platform for usable security research focused on content-based encryption. Currently, participating in research of this area has been a costly affair, requiring researchers to either build a system from scratch [13], [14] or to make substantial modifications to one of the existing open source tools [15].[1] MessageGuard is designed to greatly simplify the effort required to develop usable content-based encryption Ultimately, we hope that MessageGuard will enable collaboration between applied cryptography and usability researchers, allowing them to solve the long-standing open questions in this field, such as the creation of usable and secure key management.

The benefits of MessageGuard include:

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

1. Substantial modifications are needed because existing open source content-based encryption tools are unusable [16].

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,[2], allowing other researchers to leverage and replicate their work. Additionally, successful research that would benefit all MessageGuard prototypes 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 [17]. 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 [2], [15], [18]. The usability studies reported in this paper provide a baseline for comparison against future work built on MessageGuard.
4) *Retrofits existing web applications with content-based encryption.* Because MessageGuard works with all websites, in all browsers, and on both desktop and mobile platforms, it enables researchers to design usable content-based encryption for a wide variety of applications. Researchers do not need to cooperate with application developers or service providers, allowing them to easily work on systems that have large, installed user bases. This removes a confounding factor when conducting user studies, since users will be familiar with the application they are using. It also enables long-term usability studies, with users interact with encryption as part of their daily habits.
5) *Provides secure cryptographic operations for all applications.* MessageGuard uses security overlays [2] to isolate a user's sensitive content from web applications, ensuring that only an encrypted copy of the user's data is available to those web applications. Thus MessageGuard enables HCI researchers to easily test their ideas with applications that are actually secure, rather than relying on mock-systems, which could run the risk of invalidating their results.

In this paper we describe the MessageGuard platform, including our threat model, system goals, and implementation details. Our contributions include the following items. (1) A description of the MessageGuard platform and a guide for how researchers can use it to develop usable, content-based encryption for existing web applications. (2) Development of the first content-based encryption system that is designed to work with all websites, in all browsers, and on both desktop and mobile platforms. (3) A validation of MessageGuard's compatibility and performance, showing that it supports all of the Alexa Top 50 (e.g., Facebook, Instagram, Twitter) websites with little impact on page load times. (4) Two case studies that illustrate how MessageGuard meets its design objectives, including six usability studies with 203 participants demonstrating that users find the prototypes we built using MessageGuard to be highly usable.

2. Specifically, we are referring to a diff-based patch of MessageGuard's code base.

1.2. Areas of Potential Research

MessageGuard enables researchers to examine diverse applications of content-based encryption in web applications. For example, MessageGuard can be used to improve existing forms of content-based encryption such as secure email or secure chat. Alternatively, researchers could use MessageGuard to evaluate the feasibility of adding content-based encryption to web applications in novel contexts, such as signing Tweets for highly-targeted Twitter accounts or securing cloud storage.

Researchers can use MessageGuard to implement new interfaces, protocols, algorithms, and key management schemes while leveraging MessageGuard's existing usable interfaces and security features. Prototypes built with these new research features can be evaluated empirically, measuring the effect that they have on the user experience.

Potential areas of exploration and collaboration for security and HCI researchers include, but are not limited to: (a) designing content-based encryption interfaces that are resilient to spoofing; (b) exploring messaging protocols and key management schemes for content-based encryption; for example, certificate revocation, certificate transparency, key ratcheting, puncturable encryption; (c) developing instructive interfaces that help users build correct mental models of content-based encryption; (d) investigating key escalation, which starts users on an easy-to-use, but less secure form of key management (e.g., passwords) and then migrates them to a more secure key management scheme (e.g., PGP) as they gain expertise; (e) creating interfaces that help users avoid mistakenly sending sensitive data in the clear; (f) supporting easy-to-use key discovery for traditional public key cryptography (e.g., PGP); (g) notifying users of potential insecurities regarding their encrypted content; and (h) assisting users in migrating their encryption keys between devices they own.

2. Related Work

Unger et al. [7] provide a comprehensive overview of secure messaging protocols, which use content-based encryption, and discuss a variety of key management schemes. Unger et al.'s work demonstrates that there is strong interest in content-based encryption within the security community. Still, examining the systems highlighted by Unger et al.'s survey makes it clear that few proposals have ever undergone user studies, making clear the need for a platform to easily prototype and test these proposals.

2.1. Security Overlays

There have been several systems that have used security overlays [2], [19] to enhance existing web applications with content-based encryption: Fahl et al. [13] describe Confidentiality-as-a-Service (CaaS), a system designed to make it easy for users to encrypt their sensitive data stored in the cloud. CaaS uses Greasemonkey, a Mozilla Firefox extension, to add encryption capabilities to existing web pages. The paper describes proof-of-concept implementations of CaaS for Dropbox, Facebook, and email.

He et al. [14] proposed ShadowCrypt, a Google Chrome extension that sits between the user and their web services and allows the users to create and consume encrypted content without revealing the content to the web service. ShadowCrypt displays encrypted contents using the Shadow DOM, an upcoming `HTML5` standard.[3] Unfortunately, as discussed in Appendix §1, several flaws compromise the security of this approach.

Lau et al. [20] present Mimesis Aegis (M-Aegis), a privacy-preserving system for mobile platforms. M-Aegis places a transparent window on top of the application GUI in order to intercept and encrypt data before it reaches the native application. M-Aegis uses a novel design that utilizes features of the accessibility layer of the operating system in order to overlay the interface of any application. To our knowledge, M-Aegis is the first system that attempts to provide ubiquitous and integrated encryption outside of the browser. A prototype implementation of M-Aegis overlaying Gmail was part of a study with 15 participants that showed most of the participants did not report any noticeable difference between the original app and the app with M-Aegis enabled.

In comparison to these systems, MessageGuard is the first system that is capable of supporting content-based encryption using security overlays across all desktop and mobile platforms. A more in-depth evaluation of these systems, their strengths and weaknesses can be found in Appendix A.

3. Only blink-based browsers support the Shadow DOM.

2.2. Other approaches

In contrast to using security overlays, there are systems that attempt to provide content-based encryption through a combination of modifying the browser and modifying existing web applications.

In this first category, COWL modifies the JavaScript runtime to provide confinement between different scripts, enforcing mandatory access control [3]. For example, this allows untrusted JavaScript, such as a third-party library, to compute over sensitive data but not to transmit that data to an untrusted server. This provides powerful capabilities to the browser, enabling content-based encryption to be widely supported, but requires modifications to JavaScript and cooperation from the application provider. Hails provides a similar confinement system, written in Haskell [21]. Approaches that require a new runtime are still largely theoretical and may struggle to cope with complex web applications.

In this second category, Content Cloaking provides a browser extension recognizes and intercepts AJAX requests that the web application makes to the content provider. The extension then encrypts data as it leaves the browser and is sent to a content-provider, and then decrypts data when it arrives from the provider and before it is displayed by the browser. This must be customized for each web application, and was only demonstrated for Google Docs. Similarly, Beeswax requires tight cooperation between application developers and the security platform [22]. Developers must indicate which DOM elements should be kept private and which users can share the contents of those elements, and then the platform provides cryptographic operations and key management.

In comparison to these systems, MessageGuard does not require cooperation from web applications. This is an important distinction for two reasons: (1) modifying individual applications does not scale with the explosive growth of web applications, and (2) requiring cooperation from web developers severely limits the ability of researchers to explore content-based encryption in the applications that most interest them. For these reasons, in this paper we focus on adding content-based encryption using secure overlays.

3. Ubiquitous, Content-Based Encryption

In this section, we give the threat model that motivates our work. Next, we describe how security overlays can be used 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.

3.1. 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,[4] 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[5] are all considered part of the trusted computing base in our threat model.

Our threat model is concerned with ensuring the confidentiality, integrity, and authenticity 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 learn basic information about the encrypted package (e.g., approximate length of message, number of recipients).[6]

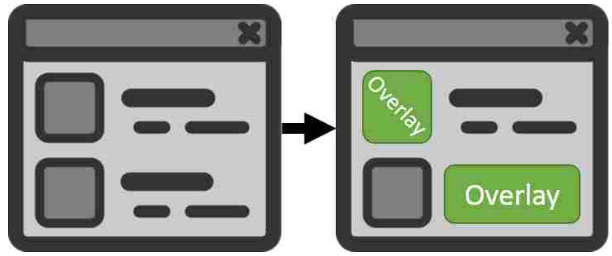
While our threat model is necessarily strict to support the wide range of web applications that researchers may wish to investigate, we note that research prototypes built using the MessageGuard platform are free to adopt a weaker threat model that may be more appropriate for that particular research.

3.2. Security Overlays

To encrypt user data before it reaches web applications, we leverage a technique known as **overlaying** [2], [19]. In this technique, MessageGuard replaces portions of the web application's interface with secure interfaces known as **overlays** (see Figure 1).[7] While a security overlay appears to be a part of the web application, the security overlay itself is inaccessible to the web application.

There are several approaches for implementing overlays: `iFrames` [2], [19], the `ShadowDOM` [14], Greasemonkey [13], and the operating system's accessibility framework [20]. An in-depth analysis of each of these approaches can be found in Appendix §1. 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 restrict 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 researcher 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., [2], [13], [14], [15]) and professional (e.g., Virtru, Mailvelope, Encipher.it).



Portions of the web application (shown on the left) have been overlayed with secure interfaces (shown on the right).

Figure 1. Overlaying a web application

3.3. Platform Goals

We examined the existing work on content-based encryption (e.g., [7], [9], [23], [24]) in order to establish a set of design goals for MessageGuard. These goals are centered around enabling to researcher to conduct research into usable, content-based encryption.

3.3.1. 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 web

4. Many websites are best described as honest-but-curious.

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

6. This type of leakage also occurs in HTTPS.

7. The overlayed elements are not actually removed, but visually occluded by the secure overlays.

application which MessageGuard has secured. 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 [2], [17]. 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 [10], [25].

3.3.2. 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 [7]. Furthermore, complicated systems foster user errors, decreasing system security [2], [9]. While some systems need to expose users to complex security choices, basic functionality (e.g., sending or receiving an encrypted email) should be approachable to 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 [2], [15]. 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 [26].

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 sizes. 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.

3.3.3. 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.

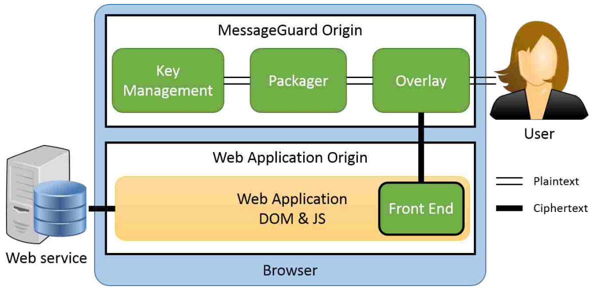
3.3.4. Extensible. MessageGuard should be easily extensible and contribute to the rapid development of content-based encryption prototypes.

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.

3.3.5. 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.



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

Figure 2. Overview of MessageGuard architecture.

Modifying the cryptographic primitives should have minimal effect of MessageGuard's usability. As above, MessageGuard should separate its UI and security functionality into separate components. This will allow security researchers to modify the cryptographic primitives without worrying about how they will affect MessageGuard's usability. The one caveat to this 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.

4. MessageGuard

Based on our design goals and using `iFrame`-based security overlays we created MessageGuard. Figure 2 shows an overview of MessageGuard's architecture.

All source code related to MessageGuard can be found at https://bitbucket.org/isrlemail/messageguard. An example secure email prototype built using MessageGuard can be found at https://messageguard.io.[8]

In the remainder of this section, we first describe MessageGuard's workflow. We then detail the design of MessageGuard's core components, and describe MessageGuard's default functionality.[9] We then discuss several web service (i.e., key server, encrypted file server) we have created to further aid researchers in building solutions using MessageGuard. Finally, we give a brief overview of MessageGuard's implementation.

4.1. Workflow

MessageGuard's workflow is as follows:

8. This prototype secures email sent and received through Gmail and uses PGP-based key management. Similar prototypes that secure email using passwords or identity-based encryption are also available upon request.

9. Per-application customization is discussed in §5.

Figure 3. Generic Read overlay.

Figure 4. 4
Generic Compose overlay.

1) MessageGuard injects the **front end** component into the website's front end. MessageGuard's front end component scans for encrypted payloads and data entry interfaces. When found, it replaces these items with an **overlay**. After this initial scan, changes to the page are tracked and only elements that have been modified are scanned.
2) Read overlays are used to display sensitive information to the user, and a compose overlay allows users to encrypt sensitive information before sending it to the website. Each overlay is displayed within an `iframe` and uses the browser's same-origin policy to protect its contents from the website. Overlays use the **packager** component to handle the actual encryption/decryption and packaging of data.
3) In conjunction with the **key management** component, the packager encrypts and decrypts data. The packager also encodes encrypted data, making it suitable for transmission to and from the website's front end.

4.2. Front End

MessageGuard's front end component is injected into every website. The front end has three responsibilities:

1) **Identify encrypted payloads.** The front end identifies encrypted payloads, overlays these payloads with a read overlay, and sends the payloads to the overlay so that they can be decrypted and displayed to the user (Figure 3).
2) **Identify data entry interfaces.** The front end identifies interfaces where the user may wish to enter encrypted data (e.g., `[contentEditable]` and `textarea` elements). Each of these interfaces are then modified to display a button that users can click

Figure 5. Generic read overlay in constrained space.

Figure 6. Generic compose overlay in constrained space.

to replace the interface with a compose overlay (Figure 4).The front end also creates an overlay manager for each overlay that handles passing data between the the compose overlay and the website (e.g., sending the overlay the encrypted payload, saving drafts).

3) **Displaying context-sensitive tutorials.** The front end 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.

The front end is the only MessageGuard component that runs outside of MessageGuard's protected origin. For this reason, MessageGuard is designed to treat the front end as untrusted. Since the front end component does not exist as part of MessageGuard's protected origin, it cannot directly access the packager, key management, or overlay components. Instead, it is limited to communicating with the overlay component using the web messaging API [27]. Additionally, the overlay always encrypts user data before transmitting it to the front end component and sanitizes any data it receives from the front end.

4.2.1. Default Functionality. MessageGuard's default front end modifies all web applications to allow for content-based encryption. The default front end will create a read overlay for encrypted payloads found anywhere on the page. It will also allow encryption in all larger textual entry interface elements (i.e., `[contentEditable]` and `textarea` elements). The front end also contains generic tutorials, which are shown when users first encounter new functionality in MessageGuard.

4.3. Overlays

MessageGuard's overlays are designed to mimic the placement and dimensions of the content they overlay and to be visually appealing and intuitive. Overlays have a distinctive, dark color scheme that stands out from most websites, allowing users to easily identify secure overlays from insecure website interfaces. Finally, overlays sanitize the plaintext contents of encrypted messages in order to prevent malicious messages from compromising the overlay.

4.3.1. Default Functionality. MessageGuard includes two standard overlays: a read overlay (Figure 3) and a compose overlay (Figure 4).[10] Read overlays are responsible for decrypting and displaying sensitive information to users. Compose overlays allow users to compose rich text messages and encrypt these messages before sending them to the website. Both overlays are reactive and modify their layout based on available space. For example, at small sizes, the compose overlay no longer shows tools for formatting text, but still allows their use through keyboard shortcuts (see Figure 5 and Figure 6).

4.4. Packager

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 necessary for recipients of the message to decrypt it.

4.4.1. Default Functionality. MessageGuard's default packager is based on the Cryptographic Message Syntax (CMS) [28] used in S/MIME. It differs from CMS in that all non-essential attributes (e.g., versioning information) are removed. This was done to reduce the package size, which is necessary for MessageGuard to support web applications that constrain the size of packages (e.g., chat applications). The one downside to this approach is that messages encrypted with this default packager cannot be read by existing secure email clients; though, customized packagers could be developed to support existing secure email standards (e.g., PGP, S/MIME).

4.5. Key Management

MessageGuard's key management component provides a UI (displayed in MessageGuard's options page) that lists the keys currently available to the user and allows the users to create, register, and delete keys. The key management component also manages storage for the various key management schemes. It includes encrypted storage for sensitive information (e.g., private keys), that is protected by a master password set by the user when first running MessageGuard. Finally, we note that while all the prototypes we have built with MessageGuard have only used a single key management scheme at a time, MessageGuard supports prototypes that permit users

10. Support for a file upload overlay is also nearing completion.

Figure 7. Interface for creating keys using a shared password.

to pick and choose which key management schemes they want to use for which messages.

4.5.1. Default Functionality. We have created three reference key management schemes for use in MessageGuard:

- **PGP.** We have created a standard-compliant implementation of PGP. This implementation generates 2048-bit keys for users, and publishes these keys for discovery on the MessageGuard's key server web service. The PGP private key pair is stored in the user's browser, but is encrypted using the key manager's master password.
- **Identity-based encryption (IBE).** We have implemented the Boneh-Boyen IBE scheme [29] for use in MessageGuard. The key server web service is responsible for storing the system parameters and master secret. Clients automatically generate public keys based on recipient identifiers. The private key is retrieved from the key server and stored in the user's browser, but is encrypted using the key manager's master password.
- **Shared Password.** Finally, we have created a key management scheme where emails are encrypted using passwords shared between users. These passwords are transformed into key material through the use of the PBKDF2 function. This key management scheme does not use the key server, and users are responsible for sharing their passwords out of band. The derived key material can either be stored in the user's browser and encrypted by the key manager's master password, or users can re-enter their shared passwords each time they want to encrypt or decrypt a message.[11]

Figure 7 is an example of the interfaces users see when adding a key to MessageGuard.

4.6. Web Services

To help researchers in creating systems using MessageGuard, we have created two web services that inter-operate with MessageGuard: a key server and a file server. The code for both servers is in MessageGuard's code repository, and researchers are welcome to have their prototypes point at our key server and file server, or deploy their own.

4.6.1. Key Server. To facilitate key management schemes that require discovery of public keys (e.g., PGP, S/MIME) or require key escrow (e.g., IBE) we have created a key server. The key server requires that users create an account with a username and a password. After account creation, users prove their ownership of other accounts (e.g., email, Twitter, Facebook). Once a user has established ownership of an account, ownership cannot be transferred to another owner unless first released by the original owner.

This key server exposes two sets of REST endpoints: public and private. The public endpoints are used to retrieve public data (e.g., public keys, IBE system parameters). The private endpoints are limited to users that have proved ownership of the appropriate account (e.g., setting a public key for a given email address, retrieving private IBE keys, etc.).

Both the PGP and IBE key management schemes we have included with MessageGuard make use of the key server. For the PGP scheme, users upload their public keys to the key server (via private endpoint), and other users download those public keys as needed (via public endpoint). For IBE, users can query the key server for the system parameters (via public endpoint), and also retrieve their private keys (via private endpoint). We welcome other researchers to use our publicly available key server, but they can also deploy their own as needed with the code hosted in the repository.

4.6.2. File Server. We have created a file server that allows for capability-based storage of files. Currently, the file server allows anyone to upload a file, after which they are given a capability (i.e., UID) that can be used to access that file. While we currently do not require authentication to the key server, that is something that could easily be added.

In practice, we use the file server to enable attachments in encrypted email. In our work using MessageGuard to build a secure integrated email prototype, we were not able to use the underlying email application's native file attachment functionality, and instead relied upon the file server to store encrypted attachments. This was done by encrypting the attachment with a random key, and storing the encrypted file sans key on the file server. The encryption key, a cryptographic hash of the attachment, and the capability for the attachment are stored in the encrypted email. As such, only individuals who can decrypt the email message are able to access and decrypt the attachment.

11. We have created one key management scheme for each behavior.

4.7. Implementation

We implemented MessageGuard so that it would run on all major desktop browsers (i.e., Chrome, Firefox, Internet Explorer, Opera, Safari) and mobile browsers (i.e., Android, iOS, Windows Phone). We employed only standard JavaScript functions that were confirmed to work on all major browsers.[12] We avoided injecting polyfills into the web application and polluting the `window` object in an effort to ensure that MessageGuard did not break existing web applications. MessageGuard is implemented as both a browser extension and as a bookmarklet (i.e., user script).

MessageGuard has a single codebase, with only a small portion of code that is used to address differences in various browsers ($< 1\%$). This codebase is implemented in JavaScript, Sass, and HTML. The code is compiled into browser extensions and a bookmarklet using NodeJS and Gulp. The code itself is split into a number of JavaScript modules which are bundled together at compile time using Browserify. Browserify also allows MessageGuard to leverage a large number of quality NodeJS modules (e.g., CryptoJS, jQuery, Bootstrap). The build system also generates source maps to facilitate debugging, runs style checking on the code, and generates documentation using JSDoc. Instructions for setting up and building MessageGuard are available in MessageGuard's repository.

Further details regarding the technologies used to implement MessageGuard can be found in Appendix §2.

5. 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.[13] 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.

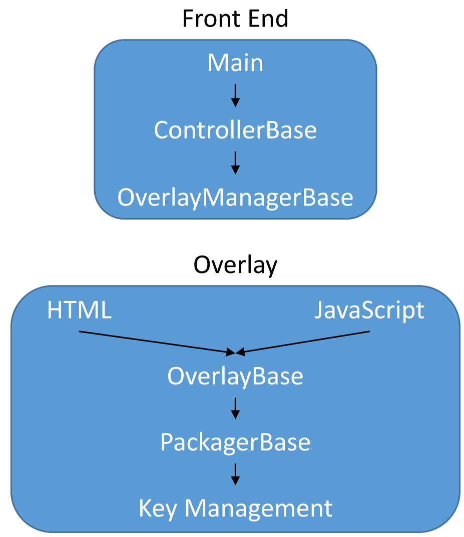


Figure 8. MessageGuard's customizable framework.

5.1. 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.[14] 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.

12. http://caniuse.com/.

13. The code base also includes an implementation of secure email as an additional reference.

14. 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.

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

5.2. 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 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.

5.3. 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 to experiment with advanced cryptographic features, such as key ratcheting [7].

5.4. Key Management

Key management is one of the most poorly understood areas of usable, content-based encryption. While there are many advocates of particular key management schemes (e.g., PGP), there has been little work to actually analyze the empirical usability of these schemes. 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.[15] 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.

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

6. Validation

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.

6.1. 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 front end 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.

6.2. 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 under extreme load. 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

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 1. AVERAGE TIME TO OVERLAY AN ELEMENT (MS)

n. We averaged measurements over ten runs and report our findings in Table 1. These results show that the time taken to overlay static content does not significantly vary based on the number of overlays created. In contrast, the time taken to overlay dynamic content is superlinear in the number of overlays created.[16] Regardless, even in extreme cases (dynamic - $n = 1000$) overlaying occurs quickly (max 61 ms).

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

7. Case Studies

While developing MessageGuard, we developed two content-based encryption prototypes using MessageGuard: a secure Facebook Chat prototype and a secure email prototype. In this section we describe these prototypes, the effort taken to build them, and IRB-approved usability studies of these prototypes. These case studies demonstrate that MessageGuard enables building high quality research prototypes that are well-liked by users.

7.1. Private Facebook Chat

Using an early version of MessageGuard, we created Private Facebook Chat (PFC), a system that adds content-based encryption to chat on Facebook [30]. PFC leverages identity-based encryption (IBE) in order to transparently manage encryption keys, removing the need for users to establish shared secrets or obtain public keys in before sending chat messages.

PFC uses a custom controller that detects the Facebook Chat interface. Instead of having separate

16. The time taken to overlay dynamic content is linear for Firefox. This indicates that our approach is not inherently superlinear, and will scale linearly as browsers become more optimized.

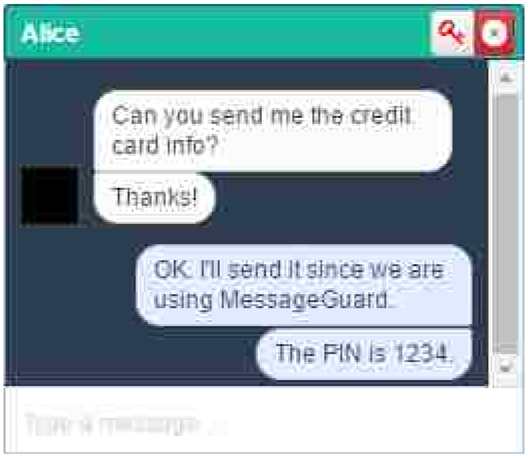


Figure 9. A unified overlay for Facebook Chat.

read and compose overlays, PFC uses a single unified overlay that handles displaying and composing encrypted messages (see Figure 9). Using a single overlay was advantageous because it was more space efficient and better mimicked a typical instant messaging interface.[17]

We conducted an IRB-approved user study of PFC involving 17 participants. Almost all users were able to use PFC to encrypt their chat sessions, except two extremely novice users that were unable to complete any tasks. Additionally, users indicated that they were generally satisfied with PFC and that they would be interested in using it in practice.

The Facebook Chat prototype took a single student working part-time one month to complete (about 80 hours). Most of this time was spent designing the interface for the unified overlay and modifying MessageGuard's IBE key management scheme to support authentication through Facebook Connect. In total, PFC's implementation required 50 lines of HTML and 350 lines of JavaScript.

7.2. Private Webmail (Pwm)

The design of MessageGuard was guided by our development of a series of secure email prototypes for Gmail [2], [16], [17] that we used to study usable, secure email for the masses. These prototypes all go by the name Private Webmail (Pwm). Pwm uses identity-based encryption (IBE) to allow users to send encrypted email to recipients who have not yet installed Pwm, something that is not possible with PGP or S/MIME. In addition, Pwm focuses on helping first time users understand how to decrypt their first secure email and how to use Pwm to encrypt messages.

17. In general, we recommend this approach for instant messaging clients.

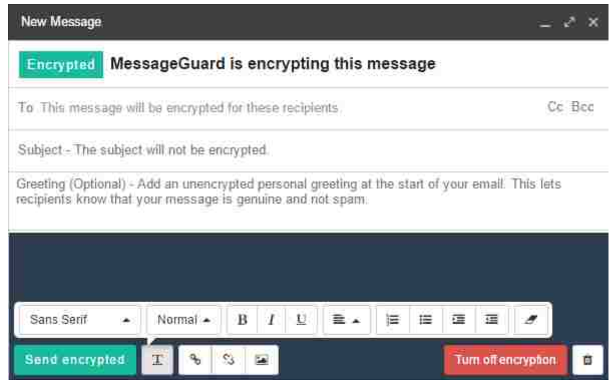


Figure 10. Customized MessageGuard front end for email.

In order to maximize Pwm's integration with Gmail we created a customized front end that annotates Gmail's interface. Our specialized front annotates Gmail's interface to help users understand better how their email is protected (see Figure 10). We also added the ability for users to include a plaintext greeting with their encrypted emails, helping recipients feel safe when receiving encrypted email. Additionally, we added context-sensitive, inline tutorials that instructed users on how to use Pwm and what protections it affords their email.

We have evaluated Pwm across five different IRB-approved usability studies, including a total of 186 participants. Each of these studies utilized a standard usability metric, the System Usability Scale (SUS) [31], [32], which generates a single score between 0 and 100 that is an indication of a system's usability. SUS consists of ten discriminating questions that users answer after completing tasks using MessageGuard.

The first three user studies all involved the same version of Pwm, with the first study (25 participants) having participants install Pwm using a bookmarklet, and the second (32 participants) and third study (28 participants) having participants install Pwm using an extension [2]. In each of the three studies, recipients were told to wait for an email containing information needed to continue the study, though they were not informed that this email would be encrypted. They were then sent the encrypted email, and were required to decrypt this email, before continuing to send and reply to several more encrypted emails. Overall, Pwm averaged a SUS score of 73.8, putting it in the 70$^{\text{th}}$ percentile of systems tested with SUS — first study (75.7), second study (70.7), third study (70.7).

Qualitative feedback from these three studies revealed that we had made the encryption and key management details too transparent and users were unsure whether or not to trust the system. This caused us to modify our design to make some encryption details more apparent in the interface [17]. Using this modified version of Pwm, we conducted our fourth usability study. It involved 51 participants that experimented with the bookmarklet version of MessageGuard and incorporated changes based on the results of prior studies. The results of this test was a SUS score of 80.0, falling in the 88th percentile of systems tested with SUS. More importantly, our modifications to Pwm (which were also incorporated into MessageGuard's reference implementation) succeeded in addressing user concerns. Nearly all participants (92%) believed that their friends and family could easily start using it.

Our fifth study examined whether Pwm could be adopted in a grassroots fashion [18]. To test this, we brought in pairs of novice users, and instructed the first participant to send an encrypted email to the second participant. The first participant was only given a link to Pwm, while the second user was not informed that they would be using encrypted email. In this study, Pwm received a SUS score of 72.3, falling in the 63rd percentile of systems tested with SUS.[18] Furthermore, all users successfully used Pwm to begin transmitting encrypted emails between themselves, and most users praised Pwm's tutorials and intuitive interfaces.

As Pwm was developed in lock-step with MessageGuard, it is difficult to know exactly how much time was spent specifically on Pwm specific functionality, though we estimate around 150 hours. Most of this time was spent reverse engineering Gmail's interface, which is difficult because is dynamically generated and minimizes both HTML and JavaScript. We also spent significant time creating and refining Pwm's tutorials. In total, Pwm includes approximately 1300 news lines of JavaScript.

7.3. Lessons Learned

Both case studies demonstrate that it is possible to rapidly build content-based encryption prototypes using MessageGuard. While these prototypes still required some time to implement (80 and 150 hours), this pales in comparison to the nearly 2000 hours that have already gone into the development of MessageGuard. Our work with Pwm especially helped us see the value of comparing competing systems; lab usability studies have been particularly helpful in helping us to differentiate between the usability of several different secure email systems. Finally, these case studies demonstrate that MessageGuard can be used to build prototypes that are rated by users as being highly usable. This is especially important in the case of Pwm, as Pwm's interfaces are highly similar to the interfaces used in MessageGuard's default functionality.

18. In this study we also examined other systems, all of which received uncharacteristically low scores, suggesting that Pwm's true SUS score is much closer to the 80 found in our fourth study.

8. Conclusion

We described MessageGuard, a platform for usable security research focused on content-based encryption. MessageGuard is designed to encourage collaboration between the security and usability research communities to solve longstanding usability problems in this space. It simplifies development of prototypes and comparison of competing designs, while also providing a platform for sharing and replicating research results. MessageGuard retrofits existing applications, providing broad utility to developers, and includes secure cryptographic operations so that usability is tested on functioning systems. We validated the performance and deployability of MessageGuard and shared several case studies demonstrating its utility for the field. Our hope is that MessageGuard will help security and usability researchers to cooperate in solving numerous pressing problems and speed the adoption of usable, content-based encryption.

9. Acknowledgment

This work was partially funded by Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. We would like to thank Jay McCarthy, Paul van Oorschot, and Dan Olsen for providing feedback on this paper.

References

[1] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu, "When HTTPS meets CDN: A case of authentication in delegated service," in *Thirty-Fifth IEEE Symposium on Security and Privacy (S&P 2014)*. San Jose, CA: IEEE Computer Society, 2014, pp. 67–82.

[2] 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.

[3] D. Stefan, E. Z. Yang, P. Marchenko, A. Russo, D. Herman, B. Karp, and D. Mazieres, "Protecting users by confining JavaScript with COWL," in *Eleventh USENIX Symposium on Operating Systems Design and Implementation (OSDI 2014)*. Broomfield, CO: USENIX Association, 2014, pp. 131–146.

[4] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko, "Security by any other name: On the effectiveness of provider based email security," in *Twenty-Second ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*. Denver, CO: ACM, 2015, pp. 450–464.

[5] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman, "Neither snow nor rain nor MITM…: An empirical analysis of email delivery security," in *Fifteenth ACM Internet Measurement Conference (IMC 2015)*. Tokyo, Japan: ACM, 2015, pp. 27–39.

[6] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar, "TLS in the wild: An internet-wide analysis of TLS-based protocols for electronic communication," in *Twenty-Fourth Network and Distributed System Security Symposium (NDSS 2016)*. San Diego, CA: The Internet Society, 2016.

[7] 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)*. San Jose, CA: IEEE Computer Society, 2015, pp. 232–249.

[8] W. C. Garrison III, A. Shull, S. Myers, and A. J. Lee, "On the practicality of cryptographically enforcing dynamic access control policies in the cloud," in *Thirty-Seventh IEEE Symposium on Security and Privacy (S&P 2016)*. San Jose, CA: IEEE Computer Society, 2016.

[9] 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)*. Washington, D.C.: USENIX Association, 1999, pp. 14–28.

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

[11] 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)*. Florence, Italy: ACM, 2015, pp. 916–926.

[12] S. Garfinkel and H. R. Lipford, "Usable security: History, themes, and challenges," *Synthesis Lectures on Information Security, Privacy, and Trust*, vol. 5, no. 2, pp. 1–124, 2014.

[13] 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)*. Liverpool, England: IEEE Computer Society, 2012, pp. 153–162.

[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)*. Scottsdale, AZ: ACM, 2014, pp. 1028–1039.

[15] 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)*. Montreal, Canada: USENIX Association, 2015, pp. 69–88.

[16] 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.

[17] 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.

[18] S. Ruoti, J. Andersen, S. Heidbrink, M. ONeill, 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)*. San Jose, CA: ACM, 2016, pp. 4298–4308.

[19] T. W. van der Horst and K. E. Seamons, "Encrypted email based upon trusted overlays," March 2009, uS Patent 8,521,821.

[20] 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)*. San Diego, CA: USENIX Association, 2014, pp. 33–48.

[21] D. B. Giffin, A. Levy, D. Stefan, D. Terei, D. Mazieres, J. C. Mitchell, and A. Russo, "Hails: Protecting data privacy in untrusted web applications," in *Ninth USENIX Symposium on Operating Systems Design and Implementation (OSDI 2012)*. Bellevue, WA: USENIX Association, 2012, pp. 47–60.

[22] J.-S. Légaré, R. Sumi, and W. Aiello, "Beeswax: A platform for private web apps," in *Twelfth Privacy Enhancing Technologies Symposium (PETS 2016)*. Darmstadt, Germany: Springer, 2016, pp. 24–40.

[23] 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)*. Pitsburg, PA: ACM, 2005, pp. 13–24.

[24] 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*, Pitsburg, PA, 2006.

[25] 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)*. Raleigh, NC: ACM, 2012, pp. 365–377.

[26] 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)*. Oxford, England: ACM, 2009, pp. 133–144.

[27] W3C, "Web messaging," accessed May 07, 2015. http://www.w3.org/TR/webmessaging.

[28] R. Housley, "Cryptographic message syntax (cms)," RFC 5652 (Standard), Internet Engineering Task Force, September 2009, http://www.ietf.org/rfc/rfc5652.txt.

[29] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Twenty-Third International Conference on the Theory and Applications of Cryptographic Techniques (EuroCrypt 2004)*. Interlaken, Switzerland: Springer, 2004, pp. 223–238.

[30] C. Robison, S. Ruoti, T. W. van der Horst, and K. E. Seamons, "Private Facebook Chat," in *Fifth ASE/IEEE International Conference on Social Computing (SocialCom 2012) and Fourth ASE/IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT 2012)*. Amsterdam, The Netherlands: IEEE Computer Society, 2012, pp. 451–460.

[31] A. Bangor, P. Kortum, and J. Miller, "An empirical evaluation of the System Usability Scale," *International Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.

[32] ——, "Determining what individual SUS scores mean: Adding an adjective rating scale," *Journal of Usability Studies*, vol. 4, no. 3, pp. 114–123, 2009.

[33] Chromium, "Sandbox," accessed October 13, 2015. http://www.chromium.org/developers/design-documents/sandbox.

[34] Wikipedia, "Assistive technology service provider interface," accessed October 13, 2015. https://en.wikipedia.org/wiki/Assistive_Technology_Service_Provider_Interface.

[35] ——, "Pwm2own," accessed October 13, 2015. https://en.wikipedia.org/wiki/Pwn2Own.

[36] A. Barth, "The web origin concept," RFC 6454 (Proposed Standard), Internet Engineering Task Force, December 2011, http://www.ietf.org/rfc/rfc6454.txt.

[37] Google, “Browser security handbook, part 2,” accessed October 13, 2015. https://code.google.com/p/browsersec/wiki/Part2.

[38] P. Stone, “Pixel perfect timing attacks with HTML5,” 2013, accessed October 13, 2015. http://www.contextis.com/documents/2/Browser_Timing_Attacks.pdf.

[39] N. Group, “Javascript cryptography considered harmful,” 2011, accessed October 13, 2015, https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2011/august/javascript-cryptography-considered-harmful/.

[40] E. Stark, M. Hamburg, and D. Boneh, “Symmetric cryptography in javascript,” in *Twenty-Fourth Annual Computer Security Applications Conference (ACSAC 2009)*. Honolulu, HI: IEEE Computer Society, 2009, pp. 373–381.

[41] W3C, “Content security policy,” accessed May 07, 2015. http://www.w3.org/TR/CSP.

Appendix

1. Implementation Strategies

To determine the appropriate implementation strategy for MessageGuard, we examined approaches for enhancing web applications with content-based encryption. In this section, analyze these approaches with respect to their deployability, security, and usability. We group the implementation strategies into two categories: integrated strategies and non-integrated strategies. We also propose two new hybrid strategies that address limitations in the existing strategies and are better suited to meet our system goals.

1.1. Integrated Strategies. Integrated strategies attempt to add content-based encryption directly to the interfaces of existing websites. In a process known as **overlaying** [19], the browser can replace portions of the website's interface with secure interfaces known as **overlays**.These overlays are displayed as if they were a part of the web application, but their contents are inaccessible to the web application. Often, there are different overlays for different functions, such as composing textual content, uploading files, and decrypting content.

iFrame. The oldest integrated strategy is to create overlays using HTML iFrames [2], [19]. Since iFrames are a part of the browser's DOM, it is trivial to integrate them with websites. Further, iFrames are protected by the browser's same-origin policy. There are three methods for overlaying websites with iFrames:

1) **Browser extension.** Extensions are desktop only, except in the case of Firefox on Android.
2) **Bookmarklet.** Bookmarklets are user scripts that are stored as browser bookmarks. When a user clicks a bookmarklet, the associated script is executed on the current page. Bookmarklets are supported on all platforms, both desktop and mobile, and do not require users to install any software.
   When using bookmarklets, the iFrame's `src` attribute will need to reference a trusted remote domain. To maximize security, this domain should only host the contents needed by the bookmarklet's user script. This limited functionality makes it easier to lock down the domain's servers in order to prevent an intrusion by the adversary.
   Bookmarklets are affected by the browser's Content Security Policy (CSP), which can be used to limit the source of frames and scripts. Still, this limitation has been marked as a bug in Chromium[19] and Firefox,[20] and it is possible that in the future this limitation will be removed. In practice, only a small number of websites use CSP.
3) **Proxy application.** A proxy application can be used to augment the bookmarklet approach. First, it can modify the CSP settings of a website to allow iFrame's to reference the system's trusted domain. Second, a proxy can automatically inject the bookmarklet script into a webpage, obviating the need for the user to click the bookmarklet. Third, a proxy can allow the iFrame strategy to work with non-browser applications that display HTML interface elements retrieved from the web. A proxy application can work on any platform, except for Windows on mobile devices. Still, there are two significant drawbacks to this approach: (1) for non-rooted phones, the proxy must be implemented using the phone's VPN service and (2) it must proxy HTTPS connections in order to modify their contents.

In prior work, we used iFrames to develop Pwm, a secure email client for the masses [2]. Pwm tightly integrated with Gmail and was designed to maximize usability. Pwm focused on helping first time recipients of an encrypted email understand how to install Pwm and decrypt their message. Pwm was rated as highly usable by participants in several usability studies. Since our work on Pwm, additional systems in both research [15] and industry (e.g., Virtru, Mailvelope) have also used the iFrame strategy to secure email.

Shadow DOM. Shadow DOM is a new feature proposed in the HTML5 specification, and currently has a partial implementation in Blink-based browsers (e.g., Chrome). The Shadow DOM allows for the creation of a “shadow” `DocumentFragment` (i.e., `ShadowRoot`) that will be rendered in place of another fragment (i.e., host) on the website. Elements contained in the `ShadowRoot` are invisible to the rest of the DOM and must be accessed through their parent `ShadowRoot`. For example, `document.getElementById` cannot find children of a `ShadowRoot`.

19. https://code.google.com/p/chromium/issues/detail?id=233903
20. https://bugzilla.mozilla.org/show_bug.cgi?id=866522

Overlays can be implemented using a `ShadowRoot`. However, while `ShadowRoot` objects and their contents should only be accessible to the entity that created the `ShadowRoot` object, in practice this is not the case:

1) The Shadow DOM allows for an element to have multiple `ShadowRoots`, with the newest `ShadowRoot` having access to older `ShadowRoot` objects through their `olderShadowRoot` property.
2) There are two CSS selectors that “pierce” the Shadow DOM. First the “>>>” selector can be used to grab any element that matches the selectors to the right of “>>>”, regardless of whether that element is inside a `ShadowRoot`. Similarly, the “::shadow” pseudo-selector can be used to select the `ShadowRoot` attached to any element.
3) The `Element.prototype.createShadowRoot` method can be replaced by the web application with a version that saves references to the created `ShadowRoots`, allowing the web application to access the contents of these `ShadowRoots`.

In order to implement overlays using a `ShadowRoot`, it is necessary to modify the JavaScript environment to address these access methods [14]. Because of this limitation, ShadowDOM cannot be implemented using a bookmarklet, which cannot guarantee that the JavaScript environment is modified before the website has the ability to store pointers to these access methods. Instead, a Shadow DOM strategy must be implemented using either an extension or a proxy application.

He et al. used the Shadow DOM to implement ShadowCrypt, a Chrome extension that attempted to add secure communication to arbitrary websites [14]. In our experience, usability issues keep ShadowCrypt from working with most websites. Moreover, flaws in ShadowCrypt’s implementation allow the website to retrieve the user’s sensitive data.[21] This demonstrates the danger of relying on non-standard security models created by developers.

Greasemonkey. Greasemonkey is a Firefox plugin that allows for websites to be modified with custom interface elements defined using XUL. XUL interface elements can be used for overlays, and the contents of these interfaces are protected by the browser’s sandboxing of websites [33].

Fahl et al. used Greasemonkey to create a Firefox extension that added content-based encryption to Dropbox, Facebook, and email [13]. Their plugin used Confidentiality-as-a-Service (CaaS) to automate key management. Usability studies demonstrate that their system was highly usable.

Accessibility. Most operating systems have an accessibility layer that is used by accessibility tools to modify apps to make them usable by individuals with disabilities. Lau et al. proposed using the accessibility layer to implement secure overlays [20]. In this strategy, the content of secure overlays is protected by the browser’s sandbox. While our work is concerned with encrypting content within the browser, the accessibility layer is interesting in that it has the potential to secure non-browser applications. While iFrames implemented using a proxy application can also support non-browser applications, the accessibility layer has the potential to be more universal and reliable.

Lau et al. used this strategy to develop Mimesis Aegis (M-Aegis), a system for enhancing Android apps with content-based encryption. Their system secured several communication apps (e.g., Gmail) and also provided a method for support to be added for additional apps.[22] A usability study of the Android Gmail app secured with M-Aegis showed that most participants did not report any noticeable difference between the original Gmail app and the Gmail app with M-Aegis enabled.

1.2. Non-integrated Strategies. In contrast to integrated strategies, non-integrated strategies use separate applications to provide encryption, relying on the user to copy-and-paste encrypted content in and out of the original application. Because these applications are not integrated with websites, they are free to make UI design choices that maximize usability, without any concern that these choices will clash with a website’s UI.

Standalone website. A standalone website can be used to handle encryption and decryption. While it is not integrated with websites, it is integrated with the browser. Since a browser is used to run the website, no installation is required. Security is provided by the browser’s same-origin policy. This approach is not common, but there have been a few standalone websites that provide PGP encryption.[23]

Standalone Application. A standalone application is the traditional strategy for implementing content-based encryption. Security is provided by the operating system’s app separation policies.

The earliest example of a standalone content-based encryption application is Pretty Good Privacy (PGP). PGP was created in 1991 by Phil Zimmerman and is used to protect a wide range of data. Since then, many similar programs have been created.

21. A detailed description of this problem is given later in the appendix.

22. This method is not generic and requires custom logic for each supported application.

23. For example, https://www.igolder.com/pgp/encryption/ and http://www.hanewin.net/encrypt/PGcrypt.htm.

Strategy	Browsers: Chrome	Firefox	IE	Opera	Safari	Others	Desktop OS: Linux	Mac OS	Windows	Mobile OS: Android	iOS	Windows	Usability: Integration	Non-standard Interfaces	No install	Security Model
iFrame	●	●	●	●	●	●	●	●	●	◐	◐	◐[5]	●	◐	◐[5]	Same-origin
Shadow DOM	◐[2]	○	○	◐[2]	○	○	●	●	●	◐	◐	○	●	◐	○	None
Greasemonkey[1]	○	●	○	○	○	○	●	●	●	◐	○	○	●	◐	○	Browser Sandbox
Accessibility	●	●	●	●	●	◐[3]	○[4]	◐	◐	◐	○	○	●	◐	○	Browser Sandbox
Standalone website	●	●	●	●	●	●	●	●	●	●	●	●	○[6]	●	●	Same-origin
Standalone App	●	●	●	●	●	●	◐	◐	◐	◐	◐	◐	○	●	○	Browser Sandbox

● Full support, ◐ Partial support, ○ No support

[1] Scheduled for deprecation.
[2] ShadowDOM is not fully supported in Chrome.
[3] Depends on whether the browser is accessible.
[4] In development – AT-SPI [34].
[5] Potentially limited by CSP.
[6] Integrated with the browser, but not the website.

TABLE 2. Comparison of implementation strategies

1.3. Comparison. We have analyzed and compared each of the above strategies based on their deployability, usability, and security. The results are summarized in Table 2.

Deployability. There are three key areas of deployability: which browsers are supported, which desktop operating systems are supported, and which mobile devices are supported. Both the Greasemonkey and Shadow DOM strategies are limited to a single type of browser, Firefox and Blink-based, respectively. On mobile, Greasemonkey only works with Firefox on Android and the Shadow DOM requires the use of a proxy application, which as mentioned earlier is less than ideal (marked as "Partial support" in Table 2). Greasemonkey uses XUL, which is slated for deprecation. On the other hand, Shadow DOM is part of the HTML5 specification and there is a good chance that sometime in the future it will become standard in more browsers.

The deployment challenge faced by both the accessibility and standalone app strategy is that every platform, both desktop and mobile, requires its own implementation. For standalone apps, this burden could be reduced through the use of a cross-platform framework (e.g., Java, Mono), though these frameworks commonly lead to systems with a poor look-and-feel. Unfortunately, there is not a cross-platform accessibility layer and each platform does require a unique implementation. These limitations have been marked in Table 2 using the "Partial support" symbol.

Both iFrames and standalone websites are built on commonly deployed approaches, and work on all platforms and browsers. iFrames on mobile can be implemented using either bookmarklets or an application proxy. In the case of bookmarklets, iFrames are limited by websites' CSP policies, and in the case of a proxy application on mobile devices, there is the degradation of user experience (marked as "Partial support" in Table 2).

Usability. The standalone strategies suffer from a lack of integration, which is disliked by users [2], [15]. On the other hand, because the standalone strategies are not tied to the website's interface, the standalone strategies support non-standard interfaces (e.g., input field drawn using an HTML canvas). While the integrated strategies could contain logic to handle website-specific interfaces, this approach is a significant engineering effort (marked as "Partial support" in Table 2).

The standalone website strategy does not require installation. When the iFrame strategy is implemented using bookmarklets, there is also no installation.

Security. The Greasemonkey, accessibility, and standalone app strategies all rely on the browser's sandbox for security [33]. The sandbox prevents websites from being able to access any resources outside of the websites DOM, including the browser chrome (Greasemonkey) and other applications (accessibility, standalone app). While the browser's sandbox has been compromised in the past, such attacks are quickly patched [35].

The iFrame and standalone website strategies rely on the browser's same-origin policy for security [36], [37]. As part of this policy, browsers ensure that code executing on an arbitrary website is unable to access or modify content hosted on a different domain (i.e., iFrame, standalone website). This is an important model that provides security for much of the web [3].

Similar to compromises of the browser sandbox, from time to time the same-origin policy is partially broken. For example, in 2013 Paul Stone described an attack that used CSS and measurements of render times to leak the contents of an iFrame [38]. After disclosure of this flaw, the browser vendors worked with Stone and quickly addressed the issue. This is

```
var elements = document.querySelectorAll('[contentEditable]');
for (var i = 0, len = elements.length; i < len; i++) {
  var newShadowRoot = elements[i].createShadowRoot();
  newShadowRoot.innerHTML = '<shadow></shadow>';
  if (newShadowRoot.olderShadowRoot)
    console.log(newShadowRoot.olderShadowRoot.querySelector('textarea').value);
}
```

Attack 1. Steals content from ShadowCrypt elements by creating a new `ShadowRoot` and using it to access ShadowCrypt's `ShadowRoot`. ShadowCrypt stores content in a `TextArea` and so we extract it from there. This attack is possible because ShadowCrypt deletes the `Document.createShadowRoot` function instead of the `Document.prototype.createShadowRoot` function.

```
var textArea = document.querySelectorAll('*::shadow textarea');
for (var i = 0, len = textArea.length; i < len; i++) {
  console.log(textArea[i].value);
}
```

Attack 2. Steals content from ShadowCrypt elements by using the `::shadow` pseudo-selector. This attack is possible because ShadowCrypt does not filter the "::shadow" pseudo-selector. Due to frequent changes in the Shadow DOM specification, it is likely this selector did not exist when ShadowCrypt was implemented.

Figure 11. Attacks on ShadowCrypt

the advantage of relying on a security model actively supported by browsers: problems are quickly fixed.

Shadow DOM is the one approach that does not rely on a standard security model, but instead requires developers to modify the JavaScript environment to prevent websites from accessing the contents of a `ShadowRoot`. The dangers of this approach can be seen through a security analysis we conducted of ShadowCrypt. We downloaded the latest version of ShadowCrypt[24] and found two attacks that allowed web applications to read plaintext data stored in the secure overlays. The JavaScript to run these attacks and explanation of why they work are given in Figure 11

24. Version 0.3.3, released February 4, 2015.

These attacks demonstrate two problems with relying on developers to add security to the Shadow DOM. First, it is difficult for developers to correctly identify all attack vectors and correctly close them (Figure 11 – Attack 1). Second, as browsers are updated, it is possible that new attack vectors are added, and developers must be constantly vigilant (Figure 11 – Attack 2).

JavaScript-based Cryptography. The iFrame, Shadow DOM, Greasemonkey, and standalone website strategies all rely upon cryptographic primitives implemented in JavaScript. There have been concerns that in certain cases JavaScript-based cryptography is untrustworthy [39]. These arguments reduce to two different concerns. First, if cryptography is being used because TLS is not trusted by the website, then the website cannot guarantee that the JavaScript is delivered to the user's browser unmodified. Second, if cryptography is being used to encrypt the data so that it is opaque to the website, then you cannot trust the website to send you JavaScript that will encrypt this data.

Both of these are valid concerns, but are orthogonal to the strategies that use JavaScript-based cryptography. First, the strategies *do* trust TLS to correctly deliver the content-based encryption software. Second, the strategies *do not* trust the website, and that is precisely why the encryption software is separate from the website. Finally, except in the case of iFrames implemented using Bookmarklets, the cryptographic JavaScript code is only downloaded once, at installation (iFrame, Shadow DOM, Greasemonkey) or on first run (standalone website).

1.4. Hybrid Strategies. The comparison of implementation strategies shows that the iFrames and accessibility strategies best match our goals of retrofitting the web with content-based encryption. Still, both of these approaches struggle with non-standard interfaces. To address this, we suggest that each of these strategies be modified to fall back to a standalone strategy when users encounter a non-standard interface that the system was unable to overlay. We pair strategies that have the same security model: i.e., iFrame + standalone website (same-origin) and Accessibility + standalone app (browser sandbox). The capabilities of these new hybrid approaches are summarized in Table 3.

Using the hybrid strategies, a bookmarklet implementation of iFrames can protect websites that don't employ CSP, and for those websites that do use CSP, the implementation can fall back to the standalone website. This standalone website could be displayed in either a new window or tab. Alternatively, when a user encounters a non-standard interface that

Strategy	Browsers: Chrome	Firefox	IE	Opera	Safari	Others	Desktop OS: Linux	Mac OS	Windows	Mobile OS: Android	iOS	Windows	Usability: Integration	Non-standard Interfaces	No install	Security Model
iFrame + Standalone website	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	Same-origin
Accessibility + Standalone App	●	●	●	●	●	◐	○	◐	◐	◐	○	○	●	●	●	App Separation

TABLE 3. COMPARISON OF HYBRID STRATEGIES

is not overlayed, they could click a button to launch the standalone website. This allows iFrames to fully support mobile platforms without using a proxy application. Additionally, this means that the no-install implementation of iFrames (i.e., bookmarklets) is available for all browsers and platforms.

1.5. Best Strategy. Based on our goal of retrofitting the web with content-based encryption, the hybrid strategy **iFrames + standalone website** is the best strategy. It works with all current browsers and platforms, both mobile and desktop. Furthermore, in most situations it provides tight integration with websites, but can fall back to a standalone website when non-standard interfaces are encountered. Moreover, using bookmarklets, this strategy can be run on systems where the user does not have install permissions.

2. Implementation

In this section, we describe in greater depth the technical implementation of MessageGuard.

2.1. Front End. When initialized, the front end immediately scans the page using the documents `querySelectorAll` and a `TreeWalker`. After this initial scan, changes to the page are tracked using a single `MutationObserver` and only elements that have been modified are scanned. This process allows MessageGuard to have a minimal effect on page load times and application execution.

Where possible, the front end uses the Shadow DOM to position overlays (i.e., styling, not security). When the Shadow DOM is unavailable, as is the case in most browsers, we instead set the overlay's style to match the position and size of the overlayed element, and then set the overlayed element's `display` style to `none`. This alternative approach has some potential to interfere with the underlying application (e.g., `:nth-child()`) and is not as desirable as using the Shadow DOM.

2.2. Cryptography. Where ever possible stark2009symmetric used Node.js's Crypto library. For functionality not provided by this library, we used the Stanford Javascript Crypto Library [40] (SJCL). The only cryptographic primitives that we had to develop ourselves were for the implementation of Boneh-Boyen IBE. This was necessary as there no publicly available implementations IBE that were suitable for inclusion in MessageGuard.

2.3. Browser Extension. We developed the MessageGuard browser extension using Kango, a cross-browser extension framework. Kango packages MessageGuard and makes it available as a browser extension for Chrome, Firefox, Opera, and Safari. Within a browser extension, the front end is injected into a web application before the web application is loaded. The overlays, packager, and key management components operate within the extension's trusted origin, protecting them from the web application.

In Safari, browser extensions are subject to the Content Security Policy (CSP), which prevents MessageGuard from functioning on sites that set a `frame-src` CSP attribute. Until March 10 of this year, the CSP specification explicitly disallowed applying CSP policies to extensions and bookmarklets, but the language has since been weakened to allow browsers to choose whether CSP protections apply to extensions and bookmarklets [41]. Since Safari's broken functionality existed before March 10 and because all other browsers exempt extensions from CSP, we believe it is likely Safari will eventually exempt extensions as well.

2.4. Bookmarklet. Bookmarklets are user scripts that are stored as browser bookmarks. When a user clicks a bookmarklet, the associated script is executed on the current page. We have implemented MessageGuard so that it can be executed from a bookmarklet. This is helpful since mobile browsers do not currently support extensions but do support bookmarklets.

The browser extension and bookmarklets share the same codebase and are nearly identical. The only significant difference is that MessageGuard's components are hosted from a standard web origin (e.g., https://messageguard.com) instead of the local extension origin. If MessageGuard's origin was compromised, it would be possible for an attacker to inject malicious scripts into user's browsers. To help mitigate this, we recommend that MessageGuard's bookmarklet be hosted on an origin with no other responsibilities, allowing this origin to be significantly locked down.

Bookmarklets are not currently exempted from CSP protections, but this has already been marked as a bug in Chromium[25] and Firefox.[26]

25. https://code.google.com/p/chromium/issues/detail?id=233903
26. https://bugzilla.mozilla.org/show_bug.cgi?id=866522

Private Facebook Chat

Chris Robison, Scott Ruoti, Timothy W. van der Horst, Kent E. Seamons
Computer Science Department, Brigham Young University
Provo, Utah, USA
seamons@cs.byu.edu

***Abstract*—The number of instant messages sent per year now exceeds that of email. Recently users have been moving away from traditional instant messaging applications and instead using social networks as their primary communications platform. To discover attitudes related to instant messaging and its security, we have conducted a user survey. This paper also presents the design of PFC (Private Facebook Chat), a system providing convenient, secure instant messaging within Facebook Chat. PFC offers end-to-end encryption in order to thwart any eavesdropper, including Facebook itself. Finally, we have conducted a usability study of a PFC prototype.**

***Keywords*-instant messaging, Facebook, privacy, useable security**

I. Introduction

Instant messaging is an increasingly popular form of synchronous communication over the Internet. Every year the instant messaging user base grows by 200 million people, and the number of instant messages sent per year now exceeds that of email. A recent report shows that users are moving away from traditional instant messaging applications and are instead using social networks as their primary communications platform.[1]

Facebook Chat was introduced in 2008 and already has a large user base. Unfortunately, Facebook Chat is not secure. For example, Facebook can read all messages sent through the system. Even if we trust Facebook with our messages, Facebook does not transmit data over HTTPS by default. This means eavesdroppers on the network have potential access to Facebook Chat conversations. Additionally, Facebook Chat is susceptible to session hijacking attacks as demonstrated by Firesheep [1]. Facebook users need to explicitly turn on HTTPS in Facebook's account settings for the browser to communicate with Facebook over an encrypted channel. Even if a user secures their own connection to Facebook, there is no way to guarantee that the other party in a chat session also has HTTPS turned on.

This paper first presents the results of a survey concerning the awareness and attitudes of users regarding the security and privacy of instant messaging. The paper then presents PFC (Private Facebook Chat), a system that enables convenient, secure instant messaging within Facebook Chat.

The system prevents Facebook from accessing the plaintext of a chat session. The design and implementation of PFC represents the first system that provides end-to-end security in a browser-based instant messaging service. PFC uses a *security overlay* that is placed over the current Facebook Chat interface to allow users to easily secure chat sessions that contain sensitive information. PFC also uses an automated key escrow system to transparently manage encryption keys, removing the need for users to establish shared secrets or obtain public keys in advance. The paper includes the results of a usability study to determine whether users could easily use the system to accomplish specific tasks.

Facebook manages the storage and transmission of chat messages while the key server manages and distributes encryption keys. Assuming these parties do not collude with each other, they each have too little information during normal operation to be able to access the contents of an encrypted chat message. There is also a server that provides the client-side software necessary to encrypt and decrypt chat messages. This server is in a more powerful position to unilaterally compromise the system. Since the software it provides runs on the user's machine, it can be audited and monitored to detect attacks.

The remainder of this paper is organized as follows: Section 2 contains the results of a user survey regarding attitudes and opinions about secure chat. Section 3 describes the design and implementation of PFC. Section 4 contains the results of a usability study of the PFC prototype. Section 5 discusses related work. And Section 6 provides conclusions and future work.

II. User Survey

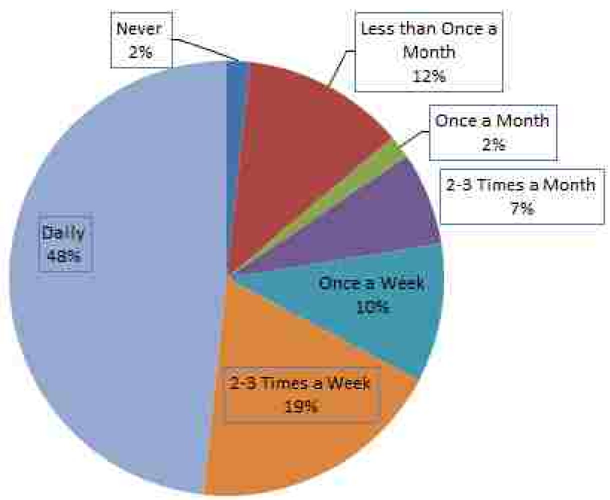
We conducted a three-part survey to determine user awareness and attitudes regarding the privacy of instant messaging. Part one gathers information about the chat systems users are currently using. Part two seeks to determine how users feel about transmitting sensitive information over chat. Finally, part three gathers information about opinions on the privacy of chat. The survey

[1]Email Statistics Report, 2009-2013, http://www.radicati.com/?p=3229

TABLE I
CHAT SYSTEM USAGE

System	Percent of Users
Google Talk	67%
Facebook Chat	50%
Skype	41%
Windows Live Messenger	19%
Yahoo Instant Messenger	19%
AIM	12%
IRC	3%
ICQ	2%
Others	19%

Fig. 1. Frequency of use



was distributed via word-of-mouth, email, and various social networks and resulted in 65 responses.

A. Chat Systems

Table I lists chat systems users reported to have used regularly. The survey allowed for users to select more than one option. Most users reported that their personal preference in which chat system to use is based on how easy it is to learn and use. Multiple users also stated that they use a particular system because their work mandates it or it integrates with other online services they use regularly, such as email or social networks. Google Talk, Facebook Chat, and Skype are the predominant chat platforms used by the survey participants. Google Talk and Facebook Chat are used primarily in the browser. The users that selected the *Others* option used commercial products regularly (e.g., Microsoft Lync, Microsoft Communicator, and IBM Sametime).

Figure 1 shows how frequently respondents use chat to communicate. Seventy-seven percent of respondents indicated they use chat at least once a week, with the majority of them using it multiple times a week. These frequent users predominantly used Google Talk, Skype, Facebook Chat, and other commercial products.

TABLE II
TYPE OF INFORMATION SENT OVER CHAT

Group	Type of information	Percent
Group 1 (59%)	Non-sensitive personal info	84%
	Moderately sensitive personal info	44%
	Highly sensitive personal info	15%
	Non-sensitive business info	56%
	Moderately sensitive business info	24%
Group 2 (24%)	Non-sensitive personal info	93%
	Moderately sensitive personal info	36%
	Any business info	<15%
Group 3 (17%)	Non-sensitive personal info	90%
	Moderately sensitive personal info	40%
	Non-sensitive business info	50%
	Moderately sensitive business info	30%

B. Trust

In this portion of the survey we asked questions to gauge how safe users feel sending information through a chat system. We then compared those answers with what types of information respondents send via chat. We provided the users with a list of sensitivities ranging from non-sensitive to highly sensitive and asked them to select the options that describe the kinds of information they have ever sent via chat in both a personal and business setting. Fifty-nine percent (Group 1) of respondents reported they feel safe or very safe when using chat, 24% (Group 2) admitted they have never thought about how safe they felt and were uncertain, and 17% (Group 3) reported they feel unsafe when they chat. Table II breaks down what percentage of each group have sent what type of information.

Of those in Group 1, 15% say they send highly sensitive personal information (e.g., social security number, credit card, bank information). Those who send highly sensitive personal information report they use Google Talk, Skype, and Facebook Chat to do so. This is of particular interest because Google Talk and Facebook Chat do not enforce secure connections to communicate via their chat systems. With Google Talk, secure connections are available when connecting through Gmail, Google+, or third party software, but that does not guarantee the other party is connected securely. Facebook defaults to unencrypted HTTP connections for most of their online services. There is a greater risk of an eavesdropper seeing sensitive information sent through the Google Talk or Facebook Chat networks.

Those in groups 2 and 3 reported to have never sent highly sensitive information of any kind over chat. It is interesting that even though those in group 3 indicated that they felt unsafe or very unsafe about the security of

Fig. 2. More likely to send sensitive information to trusted friend or family member

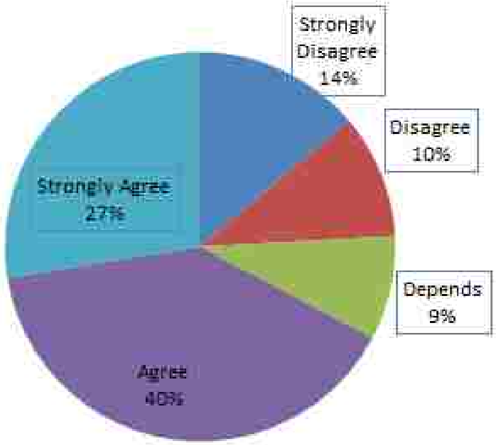
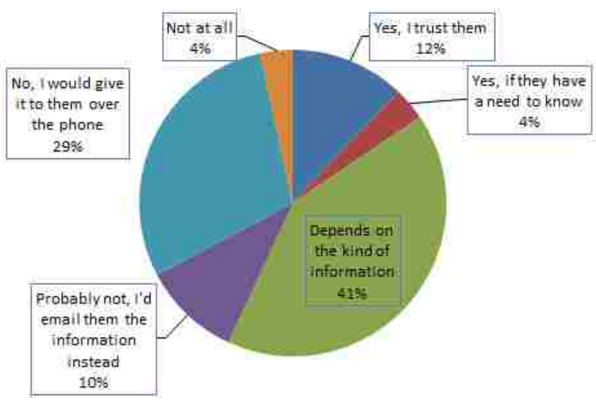


Fig. 3. Would reply with sensitive information to trusted friend or family member



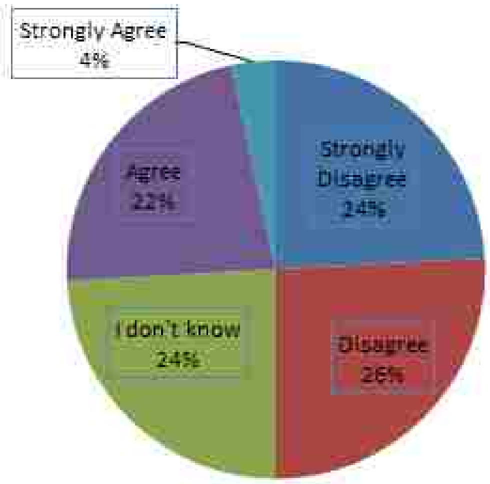
chat, most of them still reported using chat systems on a daily basis.

The feeling of safety that a user has is not only affected by the system they are using but also the context in which it is being used. We asked a set of questions to ascertain how users would react to communicating sensitive information to their trusted friends or family members.

When asked if they were more likely to send sensitive information to a trusted friend or family member (see Figure 2), 68% answered in the affirmative, 24% answered negatively, and 9% answered that it would depend on the kind of information. Most of the 9% who answered *Depends* identified information relevance and security of the communication medium as their deciding factors. However, one respondent was more concerned with the speed of the medium rather than the security of it.

When asked if they would reply over chat with sensitive information requested by a trusted friend of family

Fig. 4. Question: I'm confident that my chat conversations are private



member (see Figure 3), 41% answered that it would depend on the kind of information, 29% answered that they would use the phone instead, 12% answered yes, 10% answered they would use email, 3% answered yes if there was a need to know, and the remaining 5% responded no.

Users' responses in this part of the survey have been positive towards chat systems. Most feel safe chatting over their service of choice. This could be because their feeling of safety has never been challenged. It is apparent that users seem to have preconceived notions about which communication mediums are "safe" and which are not. The respondents mentioned email and phone systems as more secure alternatives to chat. However, email suffers from many of the same vulnerabilities as chat and the phone system has a long history of compromises [2].

C. Privacy

Some services, such as Google Talk, offer the ability for users to connect to the service using an HTTPS connection. HTTPS allows data to be transmitted in encrypted form, preventing eavesdroppers from reading the data. While this transport security is important, it does not offer full privacy because each communique is stored unencrypted and mined for data by Google. In addition, an HTTPS connection with Google does not guarantee that the other chat party is connected to Google with an HTTPS connection. We asked a set of questions to gauge the respondent's awareness of privacy and security issues.

There was a lack of agreement among users when asked if they were confident that their chat conversations were private (see Figure 4). Approximately half disagreed or strongly disagreed that chat conversations were private, approximately a quarter did not know and the remaining quarter agreed or strongly agreed. These results suggest that many users lack an understanding of privacy. Those who agreed or strongly agreed were also in the group that indicated they felt safe or very safe when using a chat system. Additionally, 71% of those that answered *I don't know* indicated they felt safe or very safe when using a chat system. We further tried to assess respondents' inclination toward verifying the identity of the person with whom they are chatting beyond the facilities provided by the chat provider. Only 5% of the respondents did not feel confident in the identity of the opposite party. This suggests a possible entry point of attack where someone malicious could assume the identity of another trusted person.

We then asked two questions to ascertain the respondents' awareness and concern about what chat providers do with their conversations after having sent and stored them. Fifty-seven percent of respondents showed concern that chat providers may mine the text of their chat conversations to provide better targeted advertising. The remaining 43% were either indifferent or not concerned. Forty-nine percent of respondents showed concern that chat providers permanently store their messages while 51% were indifferent or not concerned.

The final question attempted to assess how responsive users might be if their standard way of chatting was found to be vulnerable. We specifically asked about the chat client. Fifty-five percent answered affirmatively that they would be inclined to move to another client, and 29% preferred not to move but rather that the client be fixed. The remaining 16% were indifferent.

III. Design and Implementation

A. Design Goals

The primary design goal is ease of use. Usability issues have been a barrier in previous secure communication systems and have prevented wide spread adoption [3], [4]. PFC extends Facebook Chat so that users can continue to use their existing chat system that they are already familiar with instead of switching to a new secure chat system. Users will read and compose messages with the added ability to designate when a chat session should be secure. The interface will clearly illustrate the difference between a standard chat session and a secure chat session. The low-level details of how the secure chat is implemented (e.g., key management, encryption algorithms) are handled transparently.

PFC is designed to be adopted in a grass roots fashion. The system spreads incrementally as users engage in secure chat sessions. In order to facilitate a simple installation procedure, the PFC client is implemented as a bookmarklet. A bookmarklet is a browser bookmark that runs JavaScript in the browser window rather than navigating to a webpage. Because the bookmarklet is just JavaScript, we can reach a greater audience because all major browsers support JavaScript. In addition, the user does not have to be an administrator to install a bookmarklet since it is just a bookmark. It is also easy and highly usable because the "install" is the same as adding any other bookmark or favorite in the browser.

The threat model that PFC is designed to address is to prevent an eavesdropper from accessing Facebook chat messages. PFC uses end-to-end encryption to prevent network eavesdroppers and Facebook itself from from reading chat messages. PFC uses a key escrow system to handle all key management so that users don't need manage their own keys. The key server uses Facebook's authentication mechanism to reliably hand out keys to the proper Facebook user. This approach means that users don't need another account password in order to use PFC.

B. Prototype Implementation

We created a prototype implementation of PFC. This section describes the significant innovations of the prototype and describes the client interface that users experience while using PFC.

1) Security Overlays: PFC uses security overlays. An overlay is a frame that rests directly on top of another part of the page. Its purpose is to hide parts of the original page to prevent the user from interacting with it. The user interacts with the overlay while the overlay interacts with the obscured parts of the original page on behalf of the user. An overlay provides security features that the original page does not. Since the content in the secure overlay is served from a domain that differs from the original page, the browser's same origin security policies prevents the original page from accessing content in the overlay. PFC overlays Facebook Chat windows with an overlay frame where users read and compose chat messages while the normal Facebook Chat window only sees encrypted content.

2) Usage Scenario: Suppose Alice wants to chat securely with Bob and already has the PFC bookmarklet installed in her browser. She opens a Facebook Chat dialog to Bob and clicks on the bookmarklet to enable the secure chat feature. This executes the Javascript associated with the bookmarklet and activates PFC for the duration of her Facebook session.

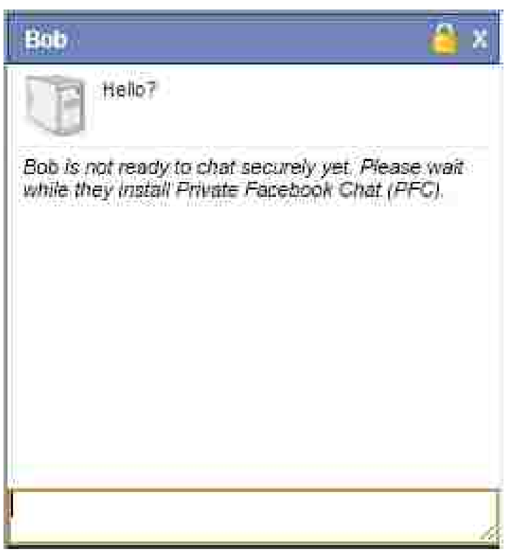
PFC overlays the Facebook Chat dialog with a security overlay and displays a lock icon to inform Alice that the chat session is secure (see Figure 5).

Alice now waits for Bob to enter secure chat before she can send a message to Bob. If Alice tries to send a message before Bob is ready, the overlay system will

Fig. 5. Secure chat window with closed lock

inform her that Bob is not yet ready to receive secure messages (see Figure 6).

Fig. 6. Pending setup notification



The PFC client on Alice's machine sends Bob a chat message from Alice stating that she would like to chat securely (see Figure 7).

The message directs Bob to a website containing instructions on how he can install the bookmarklet and chat securely with Alice (see Figure 8).

The website contains instructions and a short PFC video tutorial that directs Bob to install the secure chat bookmarklet by dragging it to his browser's bookmark bar. He then goes back to Facebook where the chat session with Alice is pending and clicks on the bookmarklet

Fig. 7. Standard greeting

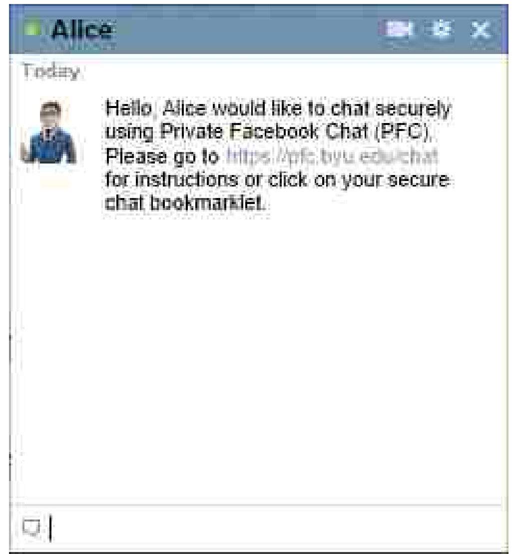
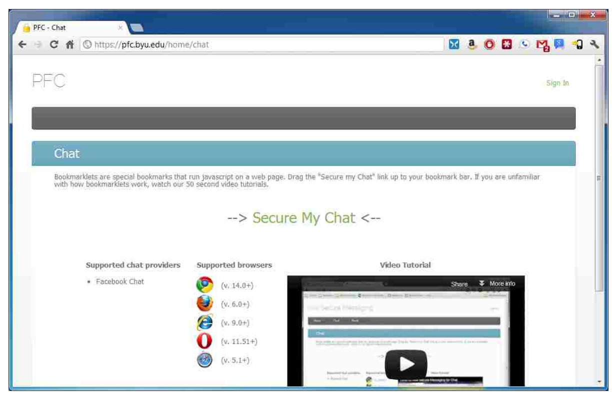


Fig. 8. PFC website



to secure the chat session in his browser. Alice and Bob are then both presented with the Facebook authentication dialog (Figure 9) that requests they authenticate with Facebook.

When they click on the *Login with Facebook* button, they are presented with a dialog from Facebook asking them if it is okay that the PFC Secure Messaging Facebook application access basic user information from their Facebook user profile (see Figure 10).

Upon choosing *Allow*, both Alice and Bob will receive a ready signal. Alice and Bob can now begin chatting securely via Facebook Chat (See Figure 11).

IV. Usability Study

A usability study of PFC was conducted involving 17 experienced Facebook Chat users (59% male, 41% female). The participants included BYU students and

Fig. 9. Authentication window

Fig. 10. Service provider authentication

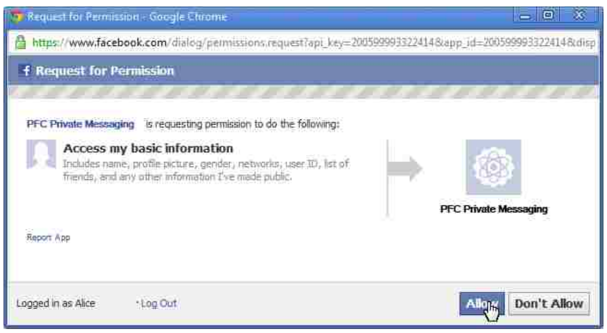


Fig. 11. Secure chat window after bookmarklet is used

employees from a local software company. Eight participants (47%) were students and had no technical background. Of the nine participants from the local software company, three of them (33%) had no technical background.

The study consisted of 5 tasks designed to exercise a specific system feature along with some survey questions. The users had no advance notice that the study would focus on the security of Facebook Chat. We created 4 dummy Facebook accounts for participants to use during the study to help segregate tasks and to pose no privacy risks to the users. Each account had a friend named Steve that the participants would chat with for testing purposes. The average completion for each user was approximately 25 minutes.

A. *Task #1*

The purpose of this task is to the determine the usability of the bootstrapping process — can a participant who receives a request to chat securely successfully obtain the software and launch a secure chat session. Each participant was instructed to log in to Facebook and wait for Steve to contact them and send them some passwords. We purposefully avoided mentioning explicitly that this information should be exchanged securely because we wanted to measure the effectiveness of our bootstrapping prompts.

Each user received the PFC standard greeting (see Figure 7) inviting them to the PFC website to learn how to chat securely. All the users went to the website except one that said they never click on links unless they are sent by a friend they know extremely well.

Overall, users were successful in completing the task. 14 of the 17 of the participants (82%) found the standard greeting and the PFC website helpful in getting them started. In addition, these 14 participants found the bookmarklet very easy to install. The three participants who struggled with installing the bookmarklet provided feedback on how to improve the bootstrapping process. This feedback included how to order the instructions on the PFC website, areas where additional feedback can be given to the user, and portions of the process that could be further simplified. The overall success of this task shows that most users can bootstrap into the system with just a short greeting, provided the link in the standard greeting is trusted.

The PFC website was generally successful in helping the participants learn to install and use the bookmarklet. We found that participants' attention was drawn to the bookmarklet link before they read the short instructions or watched the video tutorial. Slightly less than half of the participants tried installing or using the bookmarklet without instruction. 6 participants (35%) attempted to directly click the link to enable secure chat. Others tried dragging the link into the address bar. The term *bookmarklet* seemed to confuse a few participants. We observed them being unsure how to create the browser bookmark. Eventually these participants abandoned their efforts and proceeded to read the provided instructions and watch the video tutorial.

Since PFC uses Facebook authentication, we had to create a Facebook application. Users were prompted to allow the application access to their personal information. 9 participants (53%) were very wary of trusting the application. Some participants tried to continue the task without allowing that application. Most users reported that they normally do not allow any Facebook applications.

B. Task #2

For task 2, the users were instructed to securely send a checking account number to set up direct deposit (we provided a dummy account number) to Steve using Facebook Chat. The purpose of this task was to determine whether a user that had already installed the PFC bookmarklet would be able to easily initiate a new secure chat. 15 out of 17 participants (88%) completed the task and reported that they found the bookmarklet to be intuitive and easy to use.

Six participants (35%) said that clicking on the lock icon was an unnecessary step, that the bookmarklet should automatically default to a secure chat rather than require a two-click process. One participant reported they would never send a bank account number in chat, but would prefer to use the phone. Finally, 3 participants sent their checking numbers insecurely because they forgot the bookmarklet was there and needed reminding. This highlights a weakness of the bookmarklet approach since bookmarklets are unable to execute without direct interaction from the user. A browser plug-in approach would be able to better intervene and remind users of the need to use secure chat.

C. Task #3

The goal of this task is to get the reaction of the participants when they see cipher text in their chat histories and measure the usability of resuming a secure chat session. The task asks the user to re-assume the identity in task #1 and securely chat with Steve again. By resuming a conversation, the chat history in Facebook will contain the cipher text of the previous conversation. We posit that the cipher text might be a point of confusion for participants unfamiliar with it.

5 participants (29%) did not notice the cipher text during the task. Most of these participants did not complete the previous tasks that were needed for cipher text to appear in this task. Another 5 participants (29%) said that they understood the significance of the cipher text and it did not bother them. The remaining 7 participants (42%) who saw the cipher text had mixed reactions. One of the side effects of sending text that contains a valid URL is that Facebook converts them into links so that they are clickable. The ciphertext package contains a URL to the key server that provided keys to encrypt the message. Because of the link in the standard greeting, some participants, when they saw this new link in the cipher text, assumed they were supposed to click on it, which took them to an error page.

Other participants tried clicking on the link in the standard greeting again. Some participants reported that they were confused when they saw it or thought that the secure chat system was broken. Overall, 14 participants (82%) eventually used the bookmarklet and successfully completed the task. These participants agreed that resuming a secure chat conversation was easy. More participants reported this task as easier than the previous task. We hypothesize that this is because the process of resuming a secure chat only requires a single click of the bookmarklet whereas the previous task also required the participants to click the lock icon.

D. Task #4

The goal of this task is to highlight the feature that helps a user maintain a secure chat session with the opposite party. The task instructs the participant to have a secure conversation with Steve, but that he is having problems with his computer. The instructions warn that Steve might jump in and out of secure communication. The participant is supposed to do all that they can to maintain a secure communication with Steve.

During this task, 15 of 17 participants (88%) agreed that it was easy to distinguish the difference between secure and insecure messages. These participants successfully completed the task. When interacting with the option to invite the opposite party to continue secure chat, some participants were expecting a *Ready!* notification when the opposite party rejoined. Overall, the participants were able to easily maintain a secure chat session with the other party.

E. Task #5

The goal of this task is to have participants switch in and out of a secure chat session during a conversation. The task instructs the participant to conduct a casual, insecure conversation with Steve. At some point during the conversation the participant must enter a secure chat session to send a bank account number. After sending that number, the participant must exit secure chat and finish the casual conversation they were having before. At this point in the study, the participants should be familiar with all aspects of PFC.

One of the features of the secure chat history is that messages are displayed as users would have received them. For example, if an insecure conversation is transitioned to a secure conversation, all previous insecure messages sent from the opposite party will be displayed as though they were received insecurely in secure mode. Eight participants (47%) clicked on the action items presented in the insecure messages. These past messages seemed to cause a moment's confusion. However, as soon as the other party started chatting securely, all participants ignored the previous messages and continued with the task.

Fifteen participants (88%) reported that distinguishing the difference between secure and non-secure mode was easy. These participants referenced the lock icon as their primary means of demarcation. Fifteen participants (88%) also said that using the lock icon to transition between secure and non-secure mode was easy and intuitive. Six participants (35%) commented that clicking the lock icon was an unnecessary step to enter secure mode because by clicking on the bookmarklet, they are already signaling their intention of chatting securely. Nine participants (53%) reported that once in secure chat they would continue in secure chat rather than leaving it, even if the topic of conversation becomes non-sensitive.

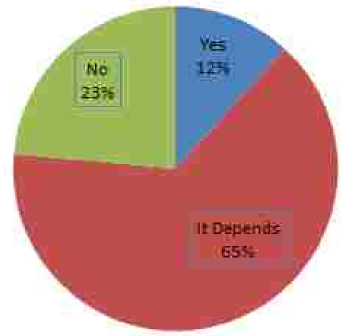
F. Survey

Following completion of the tasks, the participants answered some follow-up questions including several questions from the initial user survey. We also include questions that help us assess what the participants learned from the usability study. We try to assess the participants' inclination toward using PFC and their understanding of why using it is important.

Twelve participants (71%) said that after using PFC they were less inclined to send sensitive information over normal Facebook chat. The remaining 5 participants (29%) reported that they already do not send sensitive information through chat. Nine participants (53%) stated that they would be more inclined to send sensitive information via chat if PFC were available. Most of the other participants wanted more information about the underlying system before they would make a commitment to use it.

The success of the bootstrapping mechanism depends on users following the link sent in the standard greeting. We asked the participants if they trust links sent in chat, and Figure 12 shows their responses.

Fig. 12. Question: Do you trust links sent to you over chat?



Those who answered *It Depends*, stated that their trust of the link was dependent on their trust of the party who sent it. If trust of the standard greeting link is an issue for most users, the alternatives would be to change the standard greeting or rely on users to introduce the system to their contacts by giving them the link out-of-band.

Thirteen participants (76%) agreed they would be likely to start using this system with friends, family, or acquaintances if it were available. The remaining 4 participants (24%) were undecided. A previous task has shown that some users forgot the bookmarklet was present. This lack of decision could be due to the lack of need or reminders to use this kind of secure system. Most participants reported that if they did use this system in the real world, they would, at the very least, enable it for sensitive topics of conversation. Eight of the participants (47%) said their awareness about the security of chat systems changed as a result of their participation in this study.

G. Lessons Learned

Overall, PFC is usable. Everyone except two extremely novice users were able to successfully engage in secure chat sessions without training or human assistance. Two older, non-technical participants failed to complete any task. They use Facebook and Facebook Chat solely to stay in touch with family and close friends. They are only comfortable learning new technology with human guidance, so the current system was unable to meet their needs.

Users may be reluctant to bootstrap into the system by following the link contained in the standard greeting. In practice, this means a user wanting to initiate a secure chat session with someone unfamiliar with PFC may first have to chat insecurely and introduce them to the process.

The user study uncovered ways to improve the PFC bootstrapping web page to be more informative and easier to follow. During the study we observed that many participants were drawn immediately to the bookmarklet link before watching the video tutorial. As a result, many assumed they were supposed to click on the link, whereas if they had first watched the tutorial, they would have created a bookmark instead. Some users ignore the video and attempt to use the system immediately. A short list of steps included on the web page that summarizes the video content will be helpful. For instance, this information can alert the users that they will need to trust the associated Facebook application in order to use PFC.

Participants quickly picked up the bookmarklet during the course of the study and recognized a way to streamline the process by having the one-click of the bookmarklet automatically opt the user into a secure chat session. Finally, the user study illustrated that the chat history is confusing when it contains links. Some participants assumed they were supposed to click on them. This can be improved by presenting the actions items in a dialog instead so that the actions do not display in the chat history.

V. Threat Analysis

This section contains a threat analysis of the PFC prototype. First we list the passive attacks against the system, and then give several active attacks that can be attempted against PFC.

A. Passive Attacks

PFC thwarts passive eavesdropping by Facebook. It also thwarts other eavesdroppers on the network. PFC chat sessions are encrypted end-to-end, and protection from eavesdropping is effective even if some of the transmission links are not protected with HTTPS.

B. Key Compromise

One way to attack the system is to compromise the encryption keys. This could be done by attacking the key escrow server directly or by stealing encryption keys from the users. Attackers could attempt to steal secret keys or key material from the key server. This attack can be mitigated by storing these values in secure hardware. Alternatively, the attacker could attempt to compromise the key server to hand out encryption keys online, but this is much easier to detect and shut down. Another way to mitigate the risk of stolen encryption keys is to limit the lifetime of a key. Our implementation derives keys that are only valid for the month in which they were derived.

An important factor in the system design is that compromising the key server or a specific user's keys is not enough for an attacker to recover plaintext messages. If an attacker is unable to eavesdrop on encrypted content sent across a network, the attacker must compromise the Facebook accounts of either the sender or receiver to acquire the encrypted content. This second layer of defense helps to mitigate the threat of key compromise.

C. Impersonation

PFC relies on Facebook's OAuth system to authenticate a user and provide a unique Facebook identifier for use in deriving encryption keys. This means Facebook can impersonate a user to the key server to obtain that user's encryption keys. Users trust Facebook not to impersonate them, an attack that Facebook could already launch in it's Facebook Connect system. If Facebook were to launch such an attack in PFC, they would risk detection and a loss of reputation. The risk of this attack can be mitigated by adopting a multi-factor authentication mechanism at the key server, such as adding a user-specific password in addition to Facebook's authentication. This would prevent Facebook from easily impersonating users, at the expense of making the system less useable. Our initial PFC design favored usability over strong authentication. These properties can be adjusted to tailor the system to specific user's needs.

D. Social Engineering

During the user study, users were hesitant to click on the link provided in the standard greeting. However, once they accepted that the study required them to click on links, they assumed any link presented by PFC was trustworthy. In practice, an attacker can exploit users who trust PFC messages. For instance, a malicious service provider or an active attacker on an insecure link could inject a malicious link into the standard greeting (or any other PFC message) in order to exploit an unsuspecting user that already trusts PFC messages.

The system is designed so that users who already have a trust relationship in Facebook can chat privately. We don't make any assumptions about user's likelihood to click on PFC links because research has shown that participants in user studies may be more likely to trust experimental software in a user study compared to actual use [5].

VI. Related Work

Mannan and van Oorschot [6] surveyed the security features and threats to instant messaging protocols in an effort to spark future security improvements. They observe that the greatest threat is insecure connections, and almost a decade later this is still the case because Facebook uses HTTP by default. This threat was a primary motivation for developing PFC.

Jennings et al. [7] discuss three popular chat protocols from 2006: AOL Instant Messenger, Yahoo! Messenger, and Microsoft Messenger. Although these systems provide modest improvements beyond plaintext passwords, their security features are limited and make no effort to provide confidentiality.

Significant research has been focused on instant messaging protocols with advanced security features. Off-the-Record Messaging (OTR) allows private conversations over instant messaging by providing encryption, authentication, deniability, and perfect forward secrecy. Borisov et al. [8] introduces an Off-the-Record Messaging protocol for secure instant messaging. The protocol uses the Diffie-Hellman key exchange protocol to establish short-term keys that are impossible to rederive from the long-term key material. These keys are then discarded after a period of use, making any past messages permanently unrecoverable. The messages in this protocol are not digitally signed. It is thus impossible to prove who sent a message. Because of the frequent key exchanges necessary for secure communication, it is vulnerable to replay attacks that allow an attacker to impersonate the sender to any other party in the system. Now that PFC has proven its usability, researchers can explore ways to incorporate stronger security properties while still maintaining usability.

Raimondo et al. [9] analyzes the key features presented in Borisov et al. [8] and examines security vulnerabilities. They propose a series of change recommendations for Off-the-Record Messaging in an attempt to fix the vulnerabilities. Their recommendations include replacing the authenticated key exchange protocol with stronger exchange protocols.

Other research in Goldberg [10] and Jiang [11] addresses Off-the-Record group conversations, such as public chat rooms or other multi-party scenarios. Alexander [12] applies the socialist millionaire's problem to OTR to improve user authentication. OTR has also become publicly available as a Pidgin plug-in. Stedman et al. [13] conducted a usability study of this Pidgin plug-in to determine if it is easy to use and successful at hiding computer security details from the user. They discuss flaws in the user interface that cause confusion and decreased security. They also discuss possible solutions to these errors.

Kikuchi et al. [14] present a secure chat protocol that extends the Diffie-Hellman key exchange in order to thwart a malicious administrator. PFC could also be implemented using a Diffie-Hellman key exchange. We chose a key server approach instead to prevent an active man-in-the-middle attack on the basic Diffie-Hellman protocol.

Mannan and van Oorschot [15] present the Instant Message Key Exchange (IMKE) protocol, which is a password authentication and key exchange protocol. IMKE allows for strong authentication and secure communication in IM systems. It provides authentication (via memorable passwords), confidentiality, and message integrity with repudiation. IMKE cannot be layered on top of existing IM systems without modifications to both client and server technologies. We designed PFC so that it could be deployed without any server involvement in order to enhance the security of a popular chat platform.

In 2005, Google launched its Google Talk platform. Google Talk has an OTR mode that promises to not store the contents of an IM session. Users must trust Google to follow through on this promise since there is no cryptographic assurance that the policy is enforced. PFC could be adopted to work with Google Talk. An OTR mode could be supported by having PFC turn on the OTR flag in Google Talk so that the encrypted PFC messages are not retained by Google.

VII. Conclusions

We conducted a survey of 65 users to assess their awareness and attitudes concerning the privacy of instant messaging. 59% of the users surveyed feel safe or very safe while communicating via instant messaging, and 15% of that group admit to sending highly sensitive information in a chat session. Users are more likely to share sensitive information with a close friend or family member. Some users trust email or the phone more than they trust chat. A number of users are wary of chat service providers; 57% are concerned that providers may scan their messages for directed advertising purposes, and 49% have concerns with their messages being stored permanently.

This paper presents PFC (Private Facebook Chat), a system that provides end-to-end encryption for Facebook Chat sessions so that eavesdroppers (including Facebook itself) cannot access chat messages. The system is designed with good-enough security to thwart eavesdroppers. Security overlays provide a distinct interface on top of the existing Facebook interface so that the plaintext of a chat conversation is not available to Facebook or anyone who could modify a Facebook page during transmission. The primary design focus is on making the system easy to use.

We report on a user study that demonstrates the system is usable by current Facebook users except for the most novice computer user. The user study revealed several issues with the system. Including a link in the bootstrapping messages in order to install the system may be problematic because some users do not trust links in a chat message. To overcome this issue, the person initiating the secure chat may need to provide preliminary information that encourages the recipient being willing to click on the link. Users reported a increased awareness of the privacy issues of using Facebook Chat as a result of participating in the user study.

Our future work includes exploring ways to support secure messaging in other areas of Facebook where the communication is asynchronous, such as wall postings and status updates. We would also like to explore ways to make the ciphertext less intrusive when displayed to unauthorized parties. For instance, can we adapt steganography so that the encrypted message is stored in an image that is visible to unauthorized users, while the actual message is displayed to those authorized to see it. We also plan to propose a standardized API that service providers could support to make it easier for third parties to offer security services based on security overlays.

ACKNOWLEDGMENT

The authors would like to thank Ben Burgon, Trevor Florence, Kimball Germane, Scott Robertson, and Ryan Segeberg for their work in the Internet Security Research Lab (ISRL) at BYU to develop ideas and software related to this work.

REFERENCES

[1] Wikipedia, "Firesheep," 2011, [Online; accessed 18-May-2012]. [Online]. Available: http://en.wikipedia.org/wiki/Firesheep

[2] ——, "Phreaking," 2011, [Online; accessed 12-Nov-2011]. [Online]. Available: http://en.wikipedia.org/wiki/Phreaking

[3] S. Sheng, L. Broderick, J. Hyland, and C. Koranda, "Why johnny still can't encrypt: Evaluating the usability of email encryption software," in *Symposium On Usable Privacy and Security*, 2006.

[4] A. Whitten and J. Tygar, "Why johnny can't encrypt: A usability evaluation of pgp 5.0," in *Proceedings of the 8th USENIX Security Symposium*, vol. 99, 1999.

[5] A. Sotirakopoulos, K. Hawkey, and K. Beznosov, "I did it because i trusted you: Challenges with the study environment biasing participant behaviours," in *SOUPS Usable Security Experiment Reports (USER) Workshop*, 2010.

[6] M. Mannan and P. C. V. Oorschot, "Secure public instant messaging: A survey," in *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust*, 2004, pp. 69–77.

[7] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, S. Zon-Yin, and C. Waters, "A study of internet instant messaging and chat protocols," *Network, IEEE*, vol. 20, no. 4, pp. 16–21, 2006.

[8] N. Borisov, I. Goldberg, and E. Brewer, "Off-the-record communication, or, why not to use pgp," in *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, ser. WPES '04. New York, NY, USA: ACM, 2004, pp. 77–84. [Online]. Available: http://doi.acm.org/10.1145/1029179.1029200

[9] M. Di Raimondo, R. Gennaro, and H. Krawczyk, "Secure off-the-record messaging," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, ser. WPES '05. New York, NY, USA: ACM, 2005, pp. 81–89. [Online]. Available: http://doi.acm.org/10.1145/1102199.1102216

[10] I. Goldberg, B. Ustaoğlu, M. D. Van Gundy, and H. Chen, "Multiparty off-the-record messaging," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 358–368. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653705

[11] B. Jiang, R. Seker, and U. Topaloglu, "Off-the-record instant messaging for group conversation," in *IEEE International Conference on Information Reuse and Integration*, 2007, pp. 79–84.

[12] C. Alexander and I. Goldberg, "Improved user authentication in off-the-record messaging," in *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, ser. WPES '07. New York, NY, USA: ACM, 2007, pp. 41–47. [Online]. Available: http://doi.acm.org/10.1145/1314333.1314340

[13] R. Stedman, K. Yoshida, and I. Goldberg, "A user study of off-the-record messaging," in *Symposium on Usable Privacy and Security*, 2008, pp. 95–104. [Online]. Available: http://doi.acm.org/10.1145/1408664.1408678

[14] H. Kikuchi, M. Tada, and S. Nakanishi, "Secure instant messaging protocol preserving confidentiality against administrator," in *18th International Conference on Advanced Information Networking and Applications*, vol. 2, 2004, pp. 27–30.

[15] M. Mannan and P. van Oorschot, "A protocol for secure public instant messaging," *Financial Cryptography and Data Security*, pp. 20–35, 2006.

A Comparative Usability Study of Key Management in Secure Email

Jeff Andersen,* Tyler Monson,* Scott Ruoti,*† Kent Seamons,* and Daniel Zappala*
Brigham Young University*, Sandia National Labs†
{andersen, monson, ruoti}@isrl.byu.edu, {seamons, zappala}@cs.byu.edu

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

I. Introduction

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

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

One explanation for this poor adoption is secure email's long history of usability issues [29], [24], [20]. Recent research has begun making progress towards usable, secure email for the masses. Ruoti et al. evaluated design principles for creating a highly-usable version of IBE-based secure email [20], [17].[3] Similarly, Atwater et al. [1] explored how PGP-based secure email could be made more usable.

In this paper, we build upon this prior research and explore how different key management approaches affect the usability of secure email. To this end, we build and evaluate three secure email systems, each based on a different key management approach: PGP, IBE, and passwords. The systems largely have identical functionality and interfaces, different only as required by their particular key management scheme. Moreover, these systems incorporate all prior research into usable, secure email, giving each system its best chance at succeeding. To evaluate the systems, we conduct a within subjects, paired-participant [16] A/B user study that involved 47 participant pairs (94 total participants) using all three systems, the largest study of secure email to date.

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

The contributions of this paper are,

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

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

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

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

be unusable [29], [24]. More recently, research has made progress towards usable, PGP-based secure email, but the studies of these systems did not correctly simulate the experience of a novice user. In this paper, we use a fully-implemented system and a formal paired-participant methodology [16] to accurately evaluate the ability of novices to use PGP-based secure email. Our results demonstrate that participants viewed our PGP-based system as highly usable, with nearly a third of participants preferring it over our IBE- and password-based secure email systems.

3) **User attitudes regarding secure email.** Our study elicits user attitudes regarding the three key management approaches we evaluate. This includes security and usability trade-offs identified by participants. For example, even after understanding that PGP provides more security than IBE, many users indicate that they do not need that level of security and prefer IBE because they don't have to wait for the recipient to first install the system. Participant responses also reveal attitudes regarding secure email generally, such as the fact that many participants will only feel comfortable installing a secure email system if they know it has been verified by security-conscious individuals.
4) **Validation of prior research.** Recent research has proposed several design principles for making secure email usable for novices. In this paper we implement principles described by Atwater et al. [1] and Ruoti et al. [20], [17]. The positive results of our user studies provide validation of these design principles. More particularly, our work demonstrates that the design principles described by Ruoti et al. are generally applicable, and not just limited to IBE-based secure email. Finally, we replicate Ruoti et al.'s paired-participant methodology and provide further evidence that it has significant benefits over traditional methodologies where a study coordinator simulates one end of an email conversation.

II. BACKGROUND

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

A. Email Security

When email was first designed in 1971[4] no meaningful attention was paid to security. As such, it was originally trivial for an attacker to steal email during transit or to send messages with falsified sender information. In recent years, there have been attempts to patch security into email. For example, TLS is now used to protect email during transmission, and DKIM and DMARC are used to authenticate the sender of an email. However, the deployment of these technologies is limited and they are often misconfigured.

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

As such, email is still an easy target for attackers. For example, Durumeric et al. found that in seven countries over 20% of inbound Gmail messages are being stolen [7]. Additionally, the inability to authenticate the sender of an email increases the likelihood of email phishing, a multi-billion-dollar problem.[5] Perhaps most troubling, even if TLS, DKIM, and DMARC were to be widely adopted and configured correctly, these technologies do nothing to protect email when at rest.[6]

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

B. Threat Model

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

1) **User** — The user's computer, operating system, and secure email software are considered part of the trusted computing base.
2) **Email provider** — The email provider can be treated as either fully-malicious, honest-but-curious,[7] or *sometimes-malicious*. We define a

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

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

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

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

sometimes-malicious entity as one that is for the most part honest-but-curious, but from time to time can also take malicious action and collude with other sometimes-malicious entities. The sometime-malicious model is helpful for two reasons. First, this is a relatively accurate model of the largest email providers, which are unlikely to attack users' security unless forced to do so by an outside entity (e.g., court order). Second, this allows us to more accurately analyze systems that only transiently rely on the email provider, such as using email to verify a user's identity before they are allowed to post a public key to a key directory.

3) **Key server (optional)** — Some secure email schemes rely on the use of a trusted third-party key server, which—similar to the email provider—can be treated as either fully-malicious, sometimes-malicious, or honest-but-curious. The key server can be responsible for the generation and storage of key pairs (i.e., key escrow), or it can act as a lookup directory for individuals to find public keys (i.e., key directory). In either case, the reliance on this trusted third-party reduces the overall security of the secure email system. Still, we do note that there are methods for reducing potential harm from a sometimes-malicious third-party key server (e.g., thresholded-IBE [12], CONIKS [13]).
4) **Adversary** — The adversary is free to eavesdrop on any communication between users, email providers, and key servers.[8] Additionally, the adversary can attempt to compromise the email provider or key server. The adversary wins if she is able to use these resources to access the plaintext contents of the encrypted email body.

We do not consider attacks directly against the user or trusted computing base (i.e., phishing credentials, installing malicious software). Similarly, we do not consider an attacker who can compromise fundamental networking primitives (i.e., TLS, DNS). While these are valid concerns, if the attacker can accomplish these types of attacks, they can already do far more damage than they could by breaking the secure email system.[9] We also note that data needed by the email provider to transmit email (e.g., recipient addresses) cannot be encrypted, and may be available to the adversary (e.g., this information may be passed over an unencrypted channel). Our threat model instead focuses on ensuring that the data in the encrypted body is safe from an attacker.

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

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

C. Related Work

Whitten and Tygar [29] conducted the first formal user study of a secure email system (i.e., PGP 5), which uncovered serious usability issues with key management and users' understanding of the underlying public key cryptography. They found that a majority of users were unable to successfully send encrypted email in the context of a hypothetical political campaign scenario. The results of their study took the security community by surprise and helped shape modern usable security research.

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

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

Ruoti et al. used IBE to explore the design principles

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

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

necessary to create usable, secure email. In their first study, they demonstrated that users strongly preferred that secure email be tightly integrated with their existing email systems [20]. In a continuation of this work, Ruoti et al. demonstrated that usability could be further enhanced by adding context-sensitive inline tutorials, artificial delays on encryption while users are instructed regarding the security of their messages, and contextual clues inserted into the underlying webmail interfaces [17]. In our systems, we adopt the features discussed by Ruoti et al. and evaluate whether they are beneficial to non-IBE-based secure email (i.e., PGP- and password-based).

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

Bai et al. explored user attitudes towards different models for obtaining a recipient's public key in PGP [2]. In their study, they built two PGP-based secure email systems, one that used the traditional key exchange model,[10] and one that used a registration model based on a key directory.[11] Users were provided with instructions on how to use each tool and given several tasks to complete. Afterwards, participants shared their opinions regarding the key exchange models. The results of this study showed that, overall, individuals preferred the key directory-based registration model, though they were not averse to the traditional key exchange model either. In our study we adopt the key directory model, which was found to be preferable by Bai et al. Unlike our work, Bai et al.'s study only gathered data on user attitudes regarding key management, and did not evaluate their usability.

Ruoti et al. developed a novel paired-participant methodology for evaluating the usability of secure email [16]. Unlike other methodologies that have participants interact with study coordinators, this methodology brought in pairs of participants and observed whether these users could collaboratively begin using secure email. Each pair of participants were required to know each other before the study, better simulating how grassroots adoption of secure email would likely progress. Their results showed that this methodology was preferable to past methodologies for several reasons: first, users acted more naturally during the study; second, it identified additional pain points in the systems tested, that would not have surfaced in a traditional study; third, the methodology allowed researchers to observe both a) participants who are introducing their friends to secure email, and b) participants who are being introduced to secure email. In our study, we use this methodology to evaluate the systems we built.

III. Secure Email Systems

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

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

A. The MessageGuard Platform

The MessageGuard platform [18][12] is designed to allow researchers to rapidly prototype secure email systems.[13] These prototypes can then be deployed as browser extensions in all major browsers except IE. By building systems using MessageGuard, it is easy to ensure that the systems have overall identical interfaces and functionality, an important factor in A/B studies. MessageGuard also has support for pluggable key management, simplifying our job in developing the three secure email systems. In the remainder of this subsection

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

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

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

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

Fig. 1. Composition Overlay in MessageGuard.

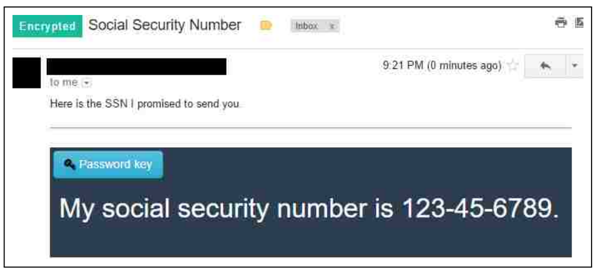


Fig. 2. Read Overlay in MessageGuard.

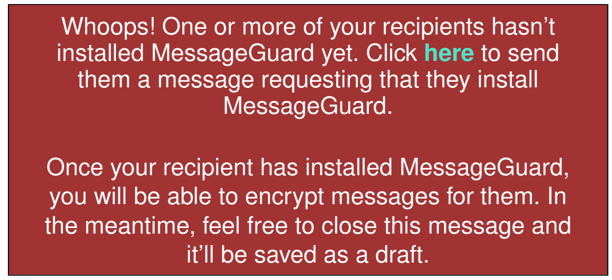
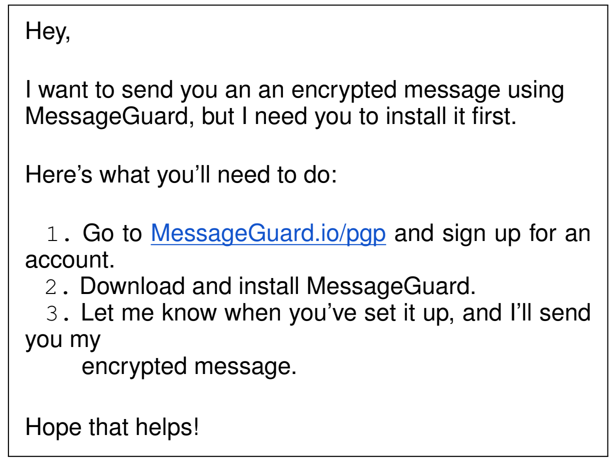


Fig. 3. Error Shown to PGP Users when Encrypting Message for Recipient Without Public Key Available in the Key Directory.


Hey,

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

Here's what you'll need to do:

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

Hope that helps!

Fig. 4. Invitation Email Generated to Invite Recipient to Install PGP.

we give a high level description of MessageGuard; for further details regarding its security we invite readers to refer to the MessageGuard paper [18].

MessageGuard tightly integrates with existing web applications—in this case Gmail—using *security overlays* [28]. Security overlays function by replacing portions of Gmail's interface with secure interfaces that are inaccessible to Gmail. Users then interact with these secure overlays to create and read encrypted email (composition–Figure 1, read–Figure 2). The overlays themselves use a distinctive color scheme to help users identify them as the interfaces to use when encrypting their email.

In addition to the distinctive interface, MessageGuard incorporates the design principles identified by Ruoti et al. as being necessary for secure email to be usable [20], [17]. First, when a message is encrypted, an artificial delay is added to this process; during the delay participants are shown informative text helping them understand how their message is being protected. Second, MessageGuard includes context-sensitive, inline tutorials that appear the first time a user initiates a specific task; research has shown that tutorials employing this style are more effective at getting users to read and understand them [17], [16]. Third, MessageGuard uses automatic encryption, but briefly shows users ciphertext as part of encryption and decryption, helping users feel confident that their messages have been encrypted. Fourth, users can include an unencrypted greeting with their secure email, helping their friends have confidence to install MessageGuard and decrypt the email. Fifth, encrypted emails contain information that help non-MessageGuard-users know how to set up and get started with MessageGuard.

B. PGP

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

PGP actually has a variety of valid configurations—for example, public keys can be verified using a web-of-trust, the certificate authority system, or by retrieval from a trusted key directory. In line with work by

Atwater et al. and Bai et al., we chose a configuration that maximized usability, while still maintaining the most important security features: First, keys are shared using a key directory [2]; users verify that they have permission to upload a key by creating an account at the key directory and verifying their email address using email-based identification and authentication [27]. Second, a user's private key is only stored on their local computer, but it is not password encrypted [1]. Third, if a user attempts to send an email to a recipient who hasn't yet uploaded a public key to the key directory, the user is prompted to send an email to the recipient with instructions on how to set up MessageGuard [1] (see Figure 3 and Figure 4). The design of our PGP system satisfies our threat model when third-party entities are sometimes-malicious, though we do note that there would be a need to monitor the key directory (e.g., with CONIKS [13]).

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

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

C. IBE

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

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

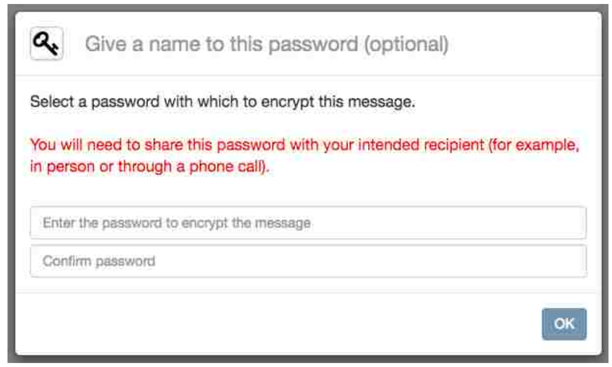


Fig. 5. Dialog for Entering a New Password with Which to Encrypt Email.

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

D. Passwords

Password-based encryption refers to allowing a user to select a password that will be used to encrypt their email.[14] This approach satisfies our threat model when third-parties are fully-malicious, and has the highest theoretical security of the three systems we built. Practically, the security of password-based encryption is limited by the strength of the user-chosen passwords and security of the out-of-band channels use to communicate those password. For example, if users choose easy-to-guess passwords, then an attacker could obtain an encrypted message and then brute force the password.

The *MessageGuard–Passwords* workflow is as follows:

1) The user visits the MessageGuard website. They are prompted to download MessageGuard–Passwords.
2) After installation, the system is immediately ready to work.
3) When the user attempts to send an encrypted email, they are informed that they need to create a password with which to encrypt the email (see Figure 5). After creating the password, the user can then send their encrypted email.

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

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.

IV. Methodology

We conducted an IRB-approved user study wherein pairs of participants used secure email to communicate sensitive information to each other. Our study methodology is taken from work by Ruoti et al. [16]. We use this methodology for its various benefits and because it allows us to compare the results from our systems with the results produced by Ruoti et al.

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

A. Study Setup

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

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

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

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

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

B. Demographics

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

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

C. Scenario Design

During the study, participants were asked to role-play a scenario about completing taxes. Johnny was told that his friend, Jane, had graduated in accounting and was going to help Johnny prepare his taxes. To do so, Johnny needed to send her his social security number and his last year’s tax PIN. Johnny was told that because this information was sensitive, he should encrypt it using a secure email system we gave him.[15] Jane was told that she would receive some information regarding taxes from Johnny, but was not informed that the information would be encrypted.

D. Task Design

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

1) Johnny would encrypt and send his SSN and last year’s tax PIN to Jane.
2) Jane would decrypt this information, then reply to Johnny with a confirmation code and this year’s tax PIN. The reply was required to be encrypted. After Johnny received this information, he would inform Jane that he had received the necessary information, and then the task would end.[16].

During each stage, participants were provided with worksheets containing instructions regarding the task

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

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

and space for participants to record the sensitive information they received.[17] Both participants were provided with the information they would send (e.g., SSN and PIN), but were told to treat this information as they would their own sensitive information. Participants completed the same task for each of the three systems being tested.

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

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

E. Study Questionnaire

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

Immediately upon completing the study task for a given secure email system, participants were asked several questions related to their experience with that system. First, participants completed the ten questions from the System Usability Scale (SUS) [5], [6]. Multiple studies have shown that SUS is a good indicator of perceived usability [26], is consistent across populations [21], and has been used in the past to rate secure email systems [20], [1], [16]. After providing a SUS score, participants were asked to describe what they liked about each system, what they would change, and why they would change it.

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

Finally, the survey told participants that MessageGuard could be enhanced with a master password, which they would be required to enter before MessageGuard would function. This would help protect their sensitive messages from other individuals who might also use the same computer. After reading the description about adding a master password to MessageGuard, users were asked to describe whether they would want this feature and why they felt that way.

F. Post-Study Interview

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

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

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

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

G. Quality Control

We excluded responses from eight pairs of participants.[18] First, three pairs were removed because the secure email tools became inoperative during the study, making it impossible for participants to complete the study. Second, two pairs were removed because the participants were non-native English speakers, making it difficult for them to understand the instructions they were given.

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

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

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

H. Limitations

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

Our study also has limitations common to all existing secure email studies. First, our populations are not representative of all groups, and future research could broaden the population (e.g., non-students, non-Gmail users). Second, our study was a short-term study, and future research should look at these issues in a longer-term longitudinal study. Third, our study is a lab study and has limitations common to all studies run in a trusted environment [14], [25].

V. RESULTS

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

A. System Usability Scale

We evaluated each system using the System Usability Scale (SUS). A breakdown of the scores are given in Table I. To give context to these scores, we leverage the work of several researchers that correlated SUS scores with more intuitive descriptions of usability [3], [4], [22], [26]. The descriptions are presented in Figure 6.

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

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

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

TABLE I. SUS SCORES

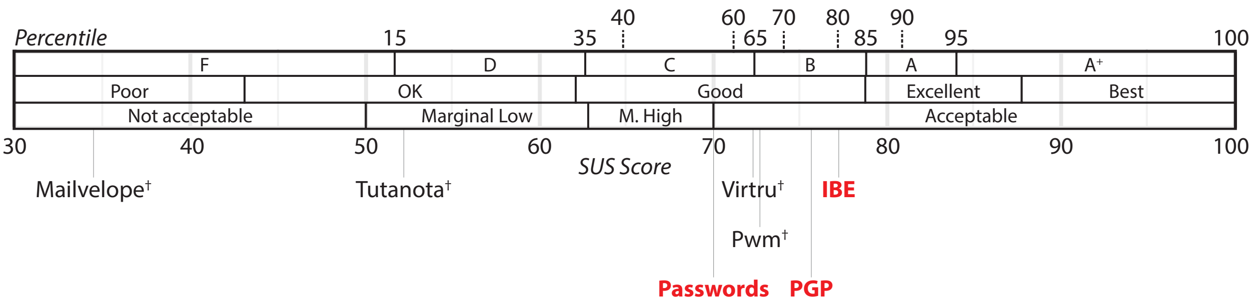
The difference between PGP's and IBE's SUS scores are not statistically significant (two-tailed student t-test, matched pairs—Johnny–$p = 0.12$, Jane–$p = 0.83$, Both–$p = 0.21$). In contrast, the difference between Jane's PGP and Passwords scores are significant (two-tailed student t-test, matched pairs—Johnny–$p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

We also compared the SUS scores for our systems against the SUS scores for other systems previously tested with the same methodology [16]. Specifically, we compared systems that had the same key management approach: PGP to Mailvelope, IBE to Pwm, IBE to Virtru,[19] and Passwords to Tutanota. In each case, our system out-performed these other systems (two-tailed student t-test, unequal variance—PGP and Mailvelope–$p < 0.001$, IBE and Pwm–$p < 0.09$, IBE and Virtru–$p = 0.04$, Passwords and Tutanota–$p < 0.001$). This gives strong evidence that the design principles integrated into MessageGuard [17], [18] lead to strong usability in secure email systems, regardless of the key management approach used.

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

We also explored whether the order in which systems were tested had an effect on their SUS scores.

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



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

Fig. 6. Adjective-based Ratings to Help Interpret SUS Scores

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

†Two-tailed student t-test, equal variance.
Result is statistically significant

TABLE II. SUS Score By Ordering

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

TABLE III. Time Taken to Complete Task (min:sec)

The results of this analysis are summarized in Table II. Overall, Johnny's experience was significantly affected by system ordering. Jane's experience was also affected, but the effect was never statistically significant.

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

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

B. Time

We recorded the time it took each participant to finish the assigned task with each system. For timing purposes the tasks were split into two stages. The first stage started when Johnny first visited the MessageGuard website and ended when he had successfully sent an encrypted email with his SSN and last year's tax PIN. The second stage started when Jane received her first encrypted email and ended when she had decrypted it, replied with the appropriate information, and received the confirmation email from Johnny. It is possible for stage one and 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.[20]

Timings were calculated using the video recordings of the participants' screen. In one instance the video file

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

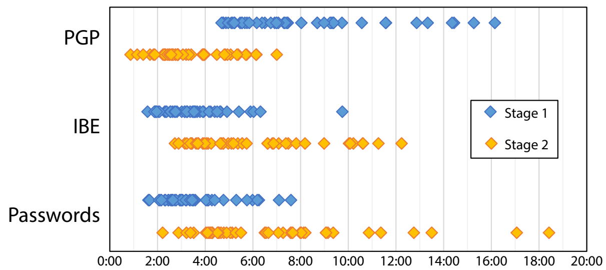


Fig. 7. Individual Participant Task Completion Times

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

†Two-tailed student t-test, equal variance.
Result is statistically significant

TABLE IV. Time Taken to Complete Task (min:sec)

of the recording was corrupted, making it impossible to gather task completion times. In three other instances the study coordinators forgot to start the video recording, causing either a complete (one instance) or partial (two instances) loss of data. Task completion time data from the remaining recordings is given in Table III and Figure 7.

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

As shown in Table IV, the task for the first system tested took longer than the other two.[21] The effect was most extreme for Jane, who didn't know that she would be using secure email until receiving her first message from Johnny. Interestingly, system ordering had the smallest effect on PGP's task completion time; though, this could be tied to PGP having the highest overall times.

C. Mistakes

We define mistakes to be instances when users send sensitive information in normal email when it should have been encrypted. For Passwords, a user is also considered to have made a mistake if they send the email-encryption password in a plaintext email.[22]

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

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

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

D. 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

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

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

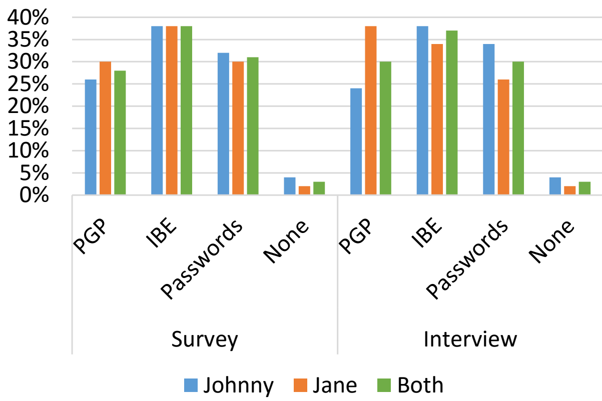


Fig. 8. Participants' Favorite System

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

TABLE V. CHANGES IN FAVORITE SYSTEM BETWEEN SURVEY AND INTERVIEW

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

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

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

E. Favorite System

At the end of the study survey, participants were asked to indicate which system was their favorite and why. Later, during the post-study interview, participants were given descriptions of each system's security model and were invited to ask further clarifying questions as needed. After hearing these descriptions, participants were allowed to update which system they felt was their favorite. Participants' preferences, both pre- and post-survey, are summarized in Figure 8.

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

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

F. Other Results

We observed participants to determine how often they used various features in MessageGuard. We noted that Johnny frequently watched both the compose and read tutorials (Compose–[14; 87%], Read–[38; 81%]). Jane similarly watched the read tutorial (43; 91%), but rarely composed a new email and as such was rarely given a chance to view the compose tutorial[23] (6 out of 10 participants; 60%). We also found that Johnny was somewhat likely to include a plaintext greeting with his encrypted email (33; 70%). When Jane did send a new encrypted message, she included an unencrypted greeting a little under half of the time (4 of 10 participants; 40%).

We also recorded how Johnny sent the password he used to encrypt his email when using the Passwords system. Johnny largely preferred phone-based communication channels for communicating the password with Jane (cellular text messaging–23, phone call–11, email–9, Google Chat–4, in person–2, Facebook Chat–1).[24] Also, in three cases (phone call–2, email –1) Johnny

[23] Encrypted replies do not contain plaintext greetings.

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

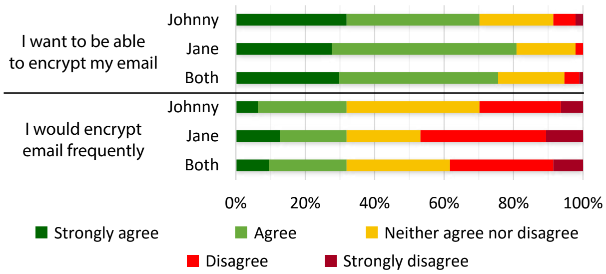


Fig. 9. Participant Opinions Regarding Secure Email

did not actually transmit the password, but merely gave clues to Jane that were sufficient for her to figure it out.

At the end of the survey, participants were asked whether they wanted to be able to encrypt their email and whether they would frequently do so. Participant responses to these questions are summarized in Figure 9. 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 [17].

VI. Discussion

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

A. PGP

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

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

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

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

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

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

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

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

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

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

The most significant issue we discovered with our PGP system was that very few participants understood its security model (4; 4%), with most participants assuming that an attacker only needed access to the user's email account to read their encrypted email. It is likely that because so much of PGP's key management was automated (e.g., key generation, uploading and retrieval of of public keys) that participants had insufficient contextual clues to determine the system's security model. While reducing the automation of the system could improve understanding, these changes would likely come at an unacceptably high usability cost [29], [24], [19]. Future work should examine how we can help users establish accurate mental models of PGP's security model in a way that does not significantly impact the usability of the system.

After explaining to participants PGP's actual security model, they felt much more confident in its security. Particularly, participants liked that it did not rely on any third parties. For example, after hearing about PGP's security model R47B enthusiastically changed her favorite system from Passwords to PGP and stated, *"Just because it had to be from your computer, it seems like, if they were to get the [encrypted contents], it'd be a little bit harder for them to get [the plaintext contents]."*

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

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

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

B. IBE

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

Compared to prior version of secure email based on IBE, our version required that users sign up for an account before they could retrieve their IBE private key.[25] Having a separate account prevents a sometimes-malicious email provider from downloading the user's IBE private key.[26] While most users were fine with setting up an account, several participants indicated that they disliked the need to set up an account. For example, R3B and R5B respectively expressed,

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

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

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

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

As with PGP, participants had a poor understanding of IBE's security model. In fact, nearly all participants assumed that PGP and IBE had the same security model. After instructing participants on IBE's security model, many participants were happy to hear it was more secure than they assumed. After understanding each system's security model, some participants who initially preferred IBE switched their preference to PGP; most remained with IBE, stating that it had good enough security. Additionally, these participants felt that IBE's ability to send an encrypted message to recipients without ensuring they had installed MessageGuard trumped the security drawbacks of IBE.

C. Passwords

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

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

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

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

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

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

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

[26] As such, our IBE system has stronger security than the IBE systems previously developed by Ruoti et al. [20], [17].

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

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

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

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

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

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

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

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

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

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

D. Key Management Trade-offs

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

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

Second, tools that rely on third-party key servers sacrifice security, but significantly reduce the burden of adopting the system. For example, evaluations of PGP systems that use manual key exchange have consistently found these systems to be unusable [29], [24], [19]. In our PGP system, we employed a third-party key directory, which significantly improved its usability at the expense of trusting a third-party. Similarly, IBE fully trusts its third-party server with private keys, making it trivial to send any recipient an encrypted message. Even though participants recognized the lower security of IBE, many indicated that it still had "good enough" security for their needs.

E. User Attitudes Regarding the Design of Secure Email

We asked participants whether they would be interested in MessageGuard including a master password.[27] Overall, participants were interested in this feature (Johnny–[33; 70%], Jane–[35; 74%], Both–[72; 77%]). Participants felt that this would provide an important security property when multiple users shared a single computer. R6A and R18A expressed, respectively,

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

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

[27] 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.

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

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

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

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

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

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

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

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

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

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

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

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

F. Validation of Prior Research

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

Third, our results demonstrate that the design principles for usable, secure email discovered by Ruoti et al. [20], [17] generalize beyond IBE, and are also applicable to PGP- and password-based systems. Moreover, we demonstrate that these principles are sufficient to make PGP—a previously unusable type of secure email [29], [24], [19]—highly usable and preferred to other approaches by a third of our participants. Many participant responses demonstrated the importance of Ruoti et al.'s design principles (e.g., tight-integration; context-sensitive, inline tutorials; unencrypted greetings) in their high estimation of our systems' usability. For example, R7A, R9A, R11A, R26B, and R34B all commented on these various design principles:

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

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

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

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

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

Finally, we gathered further evidence that paired-participant usability studies [16] are helpful in assessing the usability of secure email systems. When asked, participants indicated that they enjoyed working with a friend and that they felt it was more natural than it would be with a study coordinator; this was especially true for our Passwords system, where they indicated that calling their friend was natural, but not something they would feel comfortable doing with a coordinator. R7B stated,

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

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

VII. Conclusion

In our work, we compared the usability of three different key management approaches to secure email: PGP, IBE, and Passwords. To test these approaches, we used MessageGuard to build three secure email systems which each adopted one of these three key management schemes. The systems were built using state-of-the-art design principles for usable, secure email [20], [17], [1], [2]. We then evaluated the usability of each system using a paired-participant study methodology [16]. This evaluation was the first formal A/B evaluation of the usability of different key management schemes. It is also the largest secure email study to date, including twice as many participants as previous studies [16].

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

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)*. Montreal, Canada: USENIX Association, 2015, pp. 69–88.

[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)*. Denver, CO: USENIX Association, 2016, pp. 113–130.

[3] A. Bangor, P. Kortum, and J. Miller, "An empirical evaluation of the System Usability Scale," *International Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.

[4] ——, "Determining what individual SUS scores mean: Adding an adjective rating scale," *Journal of Usability Studies*, vol. 4, no. 3, pp. 114–123, 2009.

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

[6] ——, "SUS: A retrospective," *Journal of Usability Studies*, vol. 8, no. 2, pp. 29–40, 2013.

[7] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman, "Neither snow nor rain nor MITM…: An empirical analysis of email delivery security," in *Fifteenth ACM Internet Measurement Conference (IMC 2015)*. Tokyo, Japan: ACM, 2015, pp. 27–39.

[8] I. D. Foster, J. Larson, M. Masich, A. C. Snoeren, S. Savage, and K. Levchenko, "Security by any other name: On the effectiveness of provider based email security," in *Twenty-Second ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*. Denver, CO: ACM, 2015, pp. 450–464.

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

[10] 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)*. Pitsburg, PA: ACM, 2005, pp. 13–24.

[11] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar, "TLS in the wild: An internet-wide analysis of TLS-based protocols for electronic communication," in *Twenty-Fourth Network and Distributed System Security Symposium (NDSS 2016)*. San Diego, CA: The Internet Society, 2016.

[12] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo, “Secure key issuing in ID-based cryptography,” in *Second Workshop on Australasian Information Security (AISW 2004)*. Darlinghurst, Australia: Australian Computer Society, Inc., 2004, pp. 69–74.

[13] 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)*. Washington, D.C.: USENIX Association, 2015, pp. 383–398.

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

[15] K. Renaud, M. Volkamer, and A. Renkema-Padmos, “Why doesn’t Jane protect her privacy?” in *Fourteenth Privacy Enhancing Technologies Symposium (PETS 2014)*. Philadelphia, PA: Springer, 2014, pp. 244–262.

[16] S. Ruoti, J. Andersen, S. Heidbrink, M. ONeill, 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)*. San Jose, CA: ACM, 2016, pp. 4298–4308.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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)*. Florence, Italy: ACM, 2015, pp. 916–926.

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

[23] A. Shamir, “Identity-based cryptosystems and signature schemes,” in *Fourteenth International Cryptology Conference (Crypto 1984)*. Santa Barbara, CA: Springer, 1984, pp. 47–53.

[24] 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*, Pitsburg, PA, 2006.

[25] 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.

[26] T. S. Tullis and J. N. Stetson, “A comparison of questionnaires for assessing website usability,” in *Usability Professional Association Conference*. Minneapolis, MN: Usability Professionals Association, 2004, pp. 1–12.

[27] T. W. Van Der Horst and K. E. Seamons, “Simple authentication for the web,” in *Third International Conference on Security and Privacy in Communications Networks (SecureComm 2007)*. Nice, France: IEEE Computer Society, 2007, pp. 473–482.

[28] T. W. van der Horst and K. E. Seamons, “Encrypted email based upon trusted overlays,” March 2009, uS Patent 8,521,821.

[29] 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)*. Washington, D.C.: USENIX Association, 1999, pp. 14–28.

Chapter 8

Conclusion

In this dissertation we broadly researched usable, content-based encryption on the web. More specifically, we focused on discovering whether secure email could be made usable for the masses. Initially, this research focused on discovering what design principles would allow secure email to be more usable. To evaluate various design principles, we built secure email prototypes and tested them along with other secure email systems in a series of usability studies [24, 26, 27]. To better test the ability of systems to be adopted in a grassroots fashion, we developed a novel, paired-participant methodology for testing secure email [26, 28]. Our results demonstrated that we had succeeded in creating a usable, secure email system that could be adopted in a grassroots fashion by the novice users.

Building upon our success in creating secure email, we developed MessageGuard, a system which adds content-based encryption to most applications on the web [29]. MessageGuard incorporates all of the lessons that we learned from the work in this dissertation. Using MessageGuard, we added content-based encryption to Facebook Chat, demonstrating that many of the design principles that made secure email usable were also applicable to secure instant messaging [23]. We also architected MessageGuard so that it could act as a platform for research into usable, content-based encryption.

As the capstone of this dissertation, we used the MessageGuard platform to develop secure email variants, each of which used a different key management approach: PGP, IBE, and passwords. The three variants all used the MessageGuard user interface, and only differed from one another as needed for the unique operations of each key management approach. We then conducted an A/B study of the three variants using the paired-participant methodology we had previously developed [28]. The results of this study further demonstrated that the design principles we had discovered were sufficient to create a PGP-based secure email tool that was usable by novices, something that has long-eluded researchers and developers. Moreover, this research revealed intrinsic trade-offs between the key management approaches we tested.

The research contributions of this dissertation are as follows:

1. **Design principles for usable, secure email.** In this dissertation, we studied multiple secure email prototypes we developed (Pwm, Pwm 2.0, MessageGuard) as well as several industrial systems (Voltage Mail, Encipher.it, Tutanota, Virtru, Mailvelope). By examining quantitative results and qualitative feedback from these studies, we were able to distill the design principles that are essential for secure email to be considered usable by novice users. We also demonstrated that these principles were sufficient to create usable, secure instant messaging. Additionally, these principles were able to take PGP and make it usable for the masses, a goal which has long-eluded researchers.

2. **A novel, paired-participant evaluation methodology.** Past studies of secure email have generally involved evaluating whether a novice user could begin using secure email with recipients who are already expert users of the systems (i.e., study coordinators) [15, 33, 38]. While this is a valid usage scenario, it does not evaluate whether pairs of novice users (e.g., friends, spouses) could organically begin using secure email between themselves in a grassroots fashion. To evaluate this aspect of adoption, we developed a novel two-person methodology where pairs of participants were asked to self-discover how to send secure email between themselves using a system we provided them. Additionally, our new methodology also has the benefit of eliciting more natural and authentic behavior from study participants.

3. **A platform for usable, content-based encryption research.** While there is a large body of work on content-based encryption (i.e., end-to-end secure messaging),

little work has been done to empirically validate this research. This is problematic as experience has shown that many proposals with strong theoretical foundations are either unfeasible in real-world situations [16] or have problems that are only apparent when studied empirically [9, 25, 38]. Still, our own experience has demonstrated that building content-based encryption prototypes is a time-consuming process.

To encourage more empirical analysis of content-based encryption, we developed MessageGuard, a platform for research into usable, content-based encryption. MessageGuard helps amortize the cost of developing content-based encryption prototypes, making it easier for researchers to implement and test their ideas. Additionally, prototypes built using MessageGuard can be directly compared with minimal confounding factors, allowing researchers to more accurately compare their proposed solutions against others.

4. **Trade-offs of PGP, IBE, and password-based key management.** Past examinations of various key management approaches have all been confounded by the fact that the systems that implement each key management approach have drastically different interfaces and functionality. Leveraging MessageGuard, we were able to build three secure email variants, each of which used a different key management scheme, but otherwise had identical interfaces and functionality. Through a study of these three variants, we were able to reveal interesting trade-offs between each key management system. These trade-offs help inform researchers on the applicability of different key management approaches in various situations.

As part of this research, we conducted eight usability studies of eleven different secure email systems. In total, these studies incorporated 345 participants. Details about the studies and their results are summarized in Table 8.1, Table 8.2, and Figure 8.1.

In addition to the above research contributions, there are several more practical contributions. First, this research resulted in the first secure email system which was shown to be usable by the masses. This research also resulted in the first PGP-based system that

Citation	Systems Tested	Participant Count	Study Design
[24]	Pwm	25	Single System
	Pwm, Voltage Mail	32	Within Subjects
	MessageProtector	28	Single System
	Pwm, MessageProtector	28	Within Subjects
[23]	Private Facebook Chat	17	Single System
[27]	Pwm 2.0	51	Between Subjects
[26]	Pwm 2.0, Virtru, Tutanota	50	Within Subjects†
	Mailvelope	20	Single System†
[28]	MessageGuard—PGP, IBE, Passwords	94	Within Subjects†
	Total Participants	345	

† Uses are novel paired participant methodology.

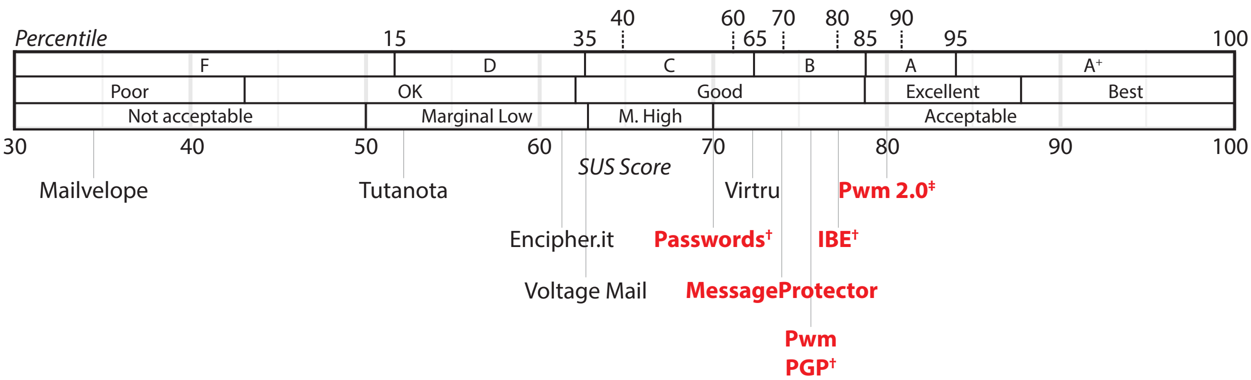
Table 8.1: User Studies in this Dissertation

		SUS		Contextual Clues		
Citation	System	Score	Confidence Interval ($\alpha = 0.05$)	Percentile†	Letter Grade	Adjective
[24]‡	Pwm	75.7	±5.3	75%	B	Good
	Voltage Mail	62.7	±6.1	35%	C−	Good
	Encipher.it	61.3	±7.7	32%	D	OK
	MessageProtector	74.0	±6.2	70%	B−	Good
[27]	Pwm 2.0	80.0	±2.5	88%	A−	Excellent
[26]	Pwm 2.0	72.7	±4.7	65%	B−	Good
	Virtru	72.3	±3.9	64%	C+	Good
	Tutanota	52.2	±5.1	16%	D	OK
	Mailvelope	34.5	±7.5	06%	F	Poor
[28]	MessageGuard					
	PGP	75.7	±3.0	76%	B	Good
	IBE	77.3	±2.7	81%	B+	Good
	Passwords	70.0	±3.0	56%	C	Good

† Percentiles are calculated by looking up the SUS score in a table [31]. When a SUS score is not in the table we estimate the percentile based on the available data.

‡ In this collection of studies, Pwm was tested three times and MessageProtector twice. In this table we list the highest score each system received.

Table 8.2: SUS Scores for Secure Email Systems



System's built as part of the work in this dissertation are shown in red.

† These are the MessageGuard systems.

‡ For readability we only included Pwm 2.0's highest score.

Figure 8.1: SUS Scores for Secure Email Systems

was usable by the masses, a significant first in this field. Second, our work has been a part of the revitalization of research into secure email. Prior to our first paper [24], it had been seven years since the last paper examined secure email, but since the publication of our paper there has been a small but steady stream of secure email papers. Our work represents the largest single contribution to the area,[1] with the majority of secure email work citing our papers [2, 3, 6, 14, 22, 30, 37]

Third, in addition to being a platform for research, MessageGuard is also a functioning system that adds content-based encryption to the majority of web applications. While other research has attempted to secure some services and types of data, MessageGuard is the first system that works across all web applications without requiring the cooperation of the web application provider.

In the remainder of this chapter, we first discuss in greater depth the research contributions in this dissertation. We then discuss potential future work that has been enabled by the research contained in this dissertation.

[1] As measured by number of papers published by any given researcher.

8.1 Design Principles of Usable, Secure Email

Throughout the various usability studies in this dissertation, we have tested a wide range of systems with a very diverse set of design features. By analyzing the quantitative results, qualitative feedback, and video recordings of user studies we were able to determine which design principles are most important for creating usable and secure email.

8.1.1 Tight Integration

Users overwhelmingly prefer that secure email systems tightly integrate with the email systems that they already use [2, 24, 26]. In the studies where integrated and non-integrated systems were directly compared, participants always preferred the integrated systems, indicating that they did not like leaving their webmail program to use the non-integrated systems. In the one case where a non-integrated system was well received (MessageProtector), participants indicated that they liked the system in spite of its lack of integration, and suggested that futures versions would strongly benefit from tight integration with webmail.

While most users preferred integrated solutions, there was a small but consistent number of participants (~5%) who preferred non-integrated solutions. For these users, having a separate encryption application gave them greater confidence that their messages were being properly encrypted. This is likely related to the design principle of hidden security details, discussed next.

8.1.2 Hidden Security Details

One of the most straightforward and intuitive ways to increase the usability of a secure system is to hide complex security details from end users [2]. While this approach does have merit, we have discovered that it can also lead to confusion and loss of user trust in the system [24, 26, 27].

Delayed Encryption

Users perceive encryption as being complex and taking a significant amount of work to complete [24]. While users are correct that encryption is complex, modern processors have made encryption of small- to medium-size messages nearly instantaneous. When presented with a secure email system that instantly encrypts their email (as normally happens), users are hesitant to trust that the system correctly and sufficiently protected their data. In our studies, users had very visceral reactions to the speed of encryption, self-reporting their discomfort without prompting from the study coordinator.[2]

To address this concern, we evaluated adding an artificial delay to message encryption [27]. While the added delay is short (1.5 seconds), our studies showed that it was sufficient to quiet users' fears regarding the speed of encryption. Additionally, during the artificial delay we were able to add contextual information to the interface that helped users better understand how their messages were being encrypted [26, 28]. For example, we added a dialog which showed who the message was being encrypted for, helping users avoid the notion that anyone with the software installed could read their messages.[3]

Automatic and Manual Encryption

Automatic encryption refers to taking a user's email, encrypting it, and automatically sending the encrypted email without ever showing the user ciphertext. In manual encryption, after the email is encrypted, users are allowed to see ciphertext before the message is sent.

In a survey of users, Fahl et al. showed that users slightly preferred automatic encryption [11]. In contrast, they observed that participants might be more trusting of a system that used manual encryption. In our own study, we observed a similar effect for

[2]In the systems tested, participants were briefly shown a dialog informing them that their message had been encrypted. Still, users reported that due to the speed of encryption they were unable to believe that encryption had really occurred.

[3]Future work could examine the effect of removing this delay for experienced users who already have a correct understanding of the system's functionality.

manual encryption, though the effect was more pronounced than observed in Fahl et al.'s study [24].[4]

To validate these earlier results, we conducted a second study where we ran an A/B comparison of manual and automatic encryption [27]. In line with previous results, we found that there was no significant difference in the usability of automatic and manual encryption. In contrast to prior results, we did not find any effect on user trust from manual and automatic encryption. The findings in our later study were simultaneously confirmed by Atwater et al. [2].

Authentication

When users do not observe some behavior they expect to see, they are likely to believe that behavior did not occur [21]. Two systems that we tested, Pwm 2.0 and Virtru, both use a trusted third-party key escrow server to manage users' encryption keys. To obtain the key necessary to decrypt their messages, a user proves their ownership of the appropriate email address by clicking a link in an email that is sent to them. While both Pwm 2.0 and Virtru used this authentication system, Pwm 2.0 automatically opens the authentication email and clicks the link for the user, whereas in Virtru users had to manually click the link. This difference was cited by multiple users in describing why they felt that Virtru was more secure than Pwm 2.0 [26], even though technically both have the exact same level of security. This makes it clear that users need to be clearly shown when and how they are authenticating so that they can feel confident that the system is functioning as intended.

8.1.3 Tutorials

While it is not surprising that tutorials would assist users in understanding how to use secure email, the question is how to convince users to watch and complete the tutorials. In initial prototypes we created textual and video tutorials that were available on the site where users

[4] This effect correlates with the well known HCI principle that users need to be able to clearly observe the system's state [21].

downloaded the secure email system. Unfortunately, our studies demonstrated that users did not view these tutorials, even when they needed help [24].

An alternative approach to static tutorials is to integrate the tutorials directly into the webmail system, and show them as they become relevant (i.e., context sensitive). We implemented this into later prototypes and found that most users watched all relevant tutorials [24, 26, 28]. More importantly, we found that after implementing these new tutorials, users better understood how their email was protected and made fewer mistakes when using the system.[5]

In designing the text for the inline, context-sensitive tutorials we found that it was best to keep the text short and to the point. Even though longer, descriptive messages might provide more potential benefits, users are more likely to skip reading these messages, negating any potential benefits. If more information is needed, it would be better to inform the users that the in-depth information exists and provide them a link to learn more [20].

8.1.4 Distinct Interface Design

When using secure email, it is easy to accidentally send sensitive information in the clear, even when a user intends to encrypt it. In our early studies, we found a consistent mistake rate (i.e., accidentally sending sensitive information in the clear) of around 10% [24]. While the nature of the studies probably leads to inflated mistake rate,[6] this is still likely to be an issue in the real world.

One approach to address this problem is to design the secure email interface so that it is visually striking and distinctive from the underlying webmail application. Our results show that this helps users clearly distinguish what information is being encrypted and what text is sent in the clear [24]. Still, our studies revealed that while this design was helpful,

[5] Here a mistake refers to sending sensitive information without encryption.

[6] In studies users are not using their own personal, sensitive information, potentially reducing the effort they expend ensuring they don't make mistakes. Additionally, some participants in our studies try to hurry through tasks without fully reading instructions, never realizing that we are asking them to encrypt information.

it was insufficient to fully stop mistakes, as many users only realized they hadn't seen the encryption interface until after they had already sent their sensitive information in the clear.

After making a mistake, users immediately recognized their error, noting that they had not seen our distinctive interface; regardless, this realization came too late. The reason for this mistake is likely related to muscle-memory, namely that users are habituated to click on "Send" immediately after finishing the composition of their email message. In addition to these mistakes, we noted that many users only enabled encryption when they were ready to send the message, which allows the webmail provider to read their sensitive information while their message is being drafted.

To further reduce mistakes, we found that it was helpful to add distinct contextual clues to the underlying webmail interface, indicating when a message was not being encrypted. For example, we added a header to email composition indicating when a message was not encrypted and changed the "Send" button to read "Send Encrypted."[7] In conjunction with the distinctive interface design, these contextual clues were able to eliminate nearly all mistakes by users [26–28]

8.2 Novel, Paired-Participant Evaluation Methodology

Other than the work described in this dissertation, all other studies of secure email have involved participants sending email messages to an expert user (i.e., study coordinator). While this is helpful in giving a general idea of the usability of a secure email system, it does not test whether two novice users could begin using secure email between themselves without outside intervention.

To address this issue, we developed a novel, paired-participant methodology that allowed us to test whether a given secure email system could be adopted in a grassroots fashion. In our methodology, pairs of participants are brought into the lab and asked to exchange

[7]We also moved the enable-encryption button to the top of the compose form. This prompts users to encrypt their messages before they begin drafting them, preventing the webmail provider from accessing sensitive information in draft form.

sensitive tax information over email using a secure email tool we provide them. When a participant signs up for this study, they are required to a bring a friend with them, ensuring that the participant pairs already know each other. During the study, each participant plays a different role, with the participant initiating communication referred to as Johnny and the other participant referred to as Jane.[8]

In addition to testing for the ability to support grassroots adoption, we found that this study methodology had several other benefits. First, by having participants play different roles, we were able to gather data about users' experiences both when they learn about secure email through self-discovery and when someone else is introducing it to them. While these same experiences might have been elicited by running two different studies, it was convenient to obtain them in a single study, and it was helpful to be able to correlate the experiences of participant pairs. Furthermore, showing that a participant can successfully use a new secure email system when inducted by another novice user is stronger than only showing that a new user can be inducted by an expert.

Second, observations by the study coordinators suggest that this study design led to more natural behavior by participants. In past studies, we observed that participants expected study coordinators to immediately respond to emails; even after being informed that a response would take a couple of minutes, participants would constantly refresh their inbox to see if a message had arrived, and if a response took longer than fifteen to thirty seconds to arrive, participants would often complain. In contrast, in the studies using our new design, participant pairs were content to wait to receive their email and did not appear agitated when their friends took a long time to respond. Also, instead of constantly refreshing their inbox, participants would browse the web or check their phones, which is likely more representative of how they use email in practice.

In addition to these observations by study coordinators, participants also noted that they felt more natural interacting with a friend than with a study coordinator. For example,

[8]There is no correlation between the roles and participant gender.

participants stated that "I don't have to feel as 'under-the-microscope' in the study," and that working with a simulated partner would have "felt more mechanical [and] robotic."

8.3 Platform for Usable, Content-based Encryption Research

Early in our study of secure email, we recognized that the design principles we were discovering could also be applied to other forms of content-based encryption. To this end, we created MessageGuard, a system which retrofits most web applications with content-based encryption capabilities [29]. As part of its design, MessageGuard leverages all of the lessons learned from our research on the design principles of usable, secure email. Furthermore, we designed MessageGuard so that it could act as a platform for other researchers to use to kick-start their own empirical research of usable, content-based encryption.

8.3.1 MessageGuard as a platform

While there is a significant body of content-based encryption research, little of it has been evaluated empirically [37]. To address this problem, we architected MessageGuard as a platform that helps enables empirical research of usable, content-based encryption. The benefits of MessageGuard as a platform include:

1. *Simplifies 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. In our own experience, building prototypes using MessageGuard took a small fraction (~10%) of the time needed to build similar prototype without MessageGuard.
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,[9] allowing other researchers to leverage and replicate their work. Additionally, successful research that would benefit all MessageGuard prototypes 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 [27]. 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 [2, 24, 26].

4. *Retrofits existing web applications with content-based encryption.* Because MessageGuard works with most websites, in all common browsers, and on both desktop and mobile platforms, it enables researchers to design usable content-based encryption for a wide variety of applications. Researchers do not need to cooperate with application developers or service providers, allowing them to easily work on systems that have a large user base. This removes a confounding factor when conducting user studies, since users will be familiar with the underlying application they are interacting with. It also enables long-term usability studies, with users interacting with encryption as part of their daily habits.

5. *Provides secure cryptographic operations for all applications.* MessageGuard uses security overlays [24] to isolate a user's sensitive content from web applications, ensuring that only an encrypted copy of the user's data is available to those web applications. Thus MessageGuard enables HCI researchers to easily test their ideas with applications that are actually secure, rather than relying on mock systems, which could run the risk of invalidating their results.

[9] A diff-based patch of MessageGuard's code base.

8.3.2 Case Studies

Most of the secure email prototypes we built as part of our research were built using MessageGuard. Other than secure email, we also used MessageGuard to build Private Facebook Chat (PFC), a system which adds content-based encryption to Facebook Chat [23]. Because PFC is built using MessageGuard, it incorporates most of the design principles discussed earlier in this chapter. Our usability study of PFC demonstrated that these principles were also sufficient to secure instant messaging in a usable manner.

In our experience, MessageGuard significantly reduced the time needed to develop content-based encryption prototypes. For example, PFC was implemented in approximately 80 hours, requiring only 350 lines of JavaScript. This is a significant reduction from the hundreds of hours and tens of thousands of lines of code needed to build Pwm, our first secure email system.

8.4 Trade-offs of PGP, IBE, and Password-based Key Management

In the final study in this dissertation, we compared three different key management approaches for secure email: PGP, IBE, and password-based encryption. In this study, users completed a task with three different secure email systems, one for each of the key management approaches. We held the interface and functionality of these systems identical–except for features and interfaces unique to each key management approach–in order to better examine inherent differences in each key management approach. Our results revealed several interesting trade-offs for each key management approach.

The most clear trade-off between the key management approaches was related to participants' understanding of each approach's security model. In the case of PGP and IBE, participants incorrectly assumed that anyone that could gain access to their email accounts could also gain access to the contents of their encrypted messages. In sharp contrast, nearly all users understood how password-based encryption protected their email. Interestingly, even though users better understood the security of password-based encryption, they rated it as less usable then either PGP or IBE.

Additionally, we found that users wanted to be immediately able to send encrypted email, and didn't want to wait for their friend to first install a secure email system. This led many participants to prefer IBE over PGP. After we explained to participants the security model for each system, several users who previously preferred IBE indicated that they would now prefer PGP, but most maintained their preference. Participants indicated that while IBE is less secure, it was secure enough for their needs and they were willing to sacrifice some security for the greater convenience provided by IBE.

The attitude of accepting good-enough security also applied to participants' opinions regarding secure email generally. For example, when asked if they wanted our systems to employ a master password,[10] participants were split. Two-thirds of participants indicated that multiple individuals used their computer and a master password would be essential to ensure the safety of their email. The remaining third indicated that their computers were already password-protected, and that the added security of a master password was not worth the increased inconvenience of having to enter yet another password.

8.5 Future Work

In this dissertation we demonstrated that secure email, and particularly PGP, could be designed such that it is usable by novice users. These results open the door to a range of interesting and compelling future work.

8.5.1 User Understanding of Secure Email

While the work in this dissertation succeeded in helping users understand how to send and receive encrypted email, there is still significant room for aiding users in building correct mental models for secure email [22]. Obviously education will play a part in this endeavour [1, 35],

[10]

but it is unclear how to ensure that users will participate in training. In businesses, users could be required to participate in training seminars, but in the wild, users are less willing to sit through instructional material [24, 26–28].

While the work in this dissertation focused on textual training, another promising avenue is graphical instruction (e.g., pictures, diagrams, animations, videos). Initial research in this direction has had moderate success [3, 35], but it is unclear whether users would be willing to watch this content if it was integrated into a real tool. MessageGuard could be used to implement this content into a real system, usability studies could then be conducted to measure the effectiveness of graphical training.

8.5.2 Anti-Phishing Interfaces

Content-based encryption does not address the risk of an attacker creating malicious interfaces that spoof legitimate interfaces, tricking users into typing their sensitive messages into non-secure interfaces, thereby leaking it to the attacker. While this is unlikely on many popular sites (e.g., Gmail, Facebook), it is more likely on small websites and certain on phishing websites. This problem is compounded by the fact that if the MessageGuard system were to become popular, people could become accustomed to typing sensitive information across the web, trusting that MessageGuard would protect this data, potentially increasing the chance of their falling victim to this type of phishing attack.

The creation of anti-phishing interfaces is a long-standing open problem in the field of security. One of the greatest hurdles to solving this problem is that promising ideas are rarely tested empirically, leaving it questionable whether the proposed techniques are actually viable [9]. Using MessageGuard, researchers could implement these anti-phishing ideas into MessageGuard's interface, and then test their effectiveness empirically.

8.5.3 New Encryption Protocols

Unger et al. [37] provide a comprehensive overview of secure messaging protocols that use content-based encryption, and discuss a variety of key management schemes. While most of the work described in their overview focuses on instant messaging, it would be interesting to determine which principles could be applied to secure email. For example, key ratcheting could be added to secure email to allow for perfect forward secrecy after the first message. Additionally, puncturable encryption could be used to add perfect forward secrecy to the first message as well [17].

While these new encryption protocols are very compelling, we have been unable to find any work that empirically evaluates the usability impact of these novel protocols. MessageGuard could be used to implement these protocols within instant messaging applications (e.g., Google Hangouts, Facebook Chat) or secure email. The resultant prototypes could then be used in a series of studies which evaluate users' opinions towards the usability of these protocols.

8.5.4 Key Management

In this dissertation, we explored trade-offs between PGP, IBE, and passwords. Additional research could be done to explore even more types of key management and their trade-offs. For example, a key escrow server could generate encryption keys for a sender on demand, and then delete those keys when retrieved by the recipient. This would allow users to send ephemeral email messages that are no longer accessible after the recipient has opened them.

Another interesting avenue is an idea we have named *key escalation*. In key escalation, users are introduced to encrypting their communication using a simple and easy-to-use key management scheme. As they become more comfortable with the system, it can progressively help them move to more secure practices, including entirely different key management schemes (e.g., IBE). This could allow users to start with the level of security they are comfortable with now, but continue upgrading their security as they become more accustomed to the system.

Future research could explore what schemes and features best fit together for key escalation, as well as determining how users at different security assurance levels could communicate with each other.

8.5.5 Usability Studies

In this dissertation, we have focused on building secure email that is sufficiently usable for the masses. For this reason, when we gathered study participants we purposely avoided recruiting computer scientists. Future work could broaden our demographics, and look at how various secure email systems support the needs of different populations. For example, an everyday user, a medical technician, and a political dissident all have different technical backgrounds and information security needs. It would be interesting to explore how various groups of people react to different secure email paradigms.

Future work should also conduct longitudinal studies of secure email. When we started this research, it wasn't practical to conduct longitudinal studies, as participants were unable to use secure email even in a short laboratory experiment [15, 33, 38]. Now that we have overcome these initial hurdles, it would be interesting to explore whether new security or usability problems appear when secure email is tested over a long period of time. Telemetric data could also be gathered to determine whether users attempt to get their friends and acquaintances to use secure email, and what adoption of secure email looks like in the wild.

8.5.6 New Applications for Content-based Encryption

In this dissertation, we explored content-based encryption of email and instant messaging. Still, the MessageGuard platform is not limited to supporting only these two use cases, and could support a range of other interesting applications:

1. **Document storage.** Users could encrypt the documents they store in the cloud, allowing them to leverage the cloud infrastructure without sacrificing data privacy.

This would be especially helpful to government agencies who want to use the cloud but currently can't due to security concerns.

2. **Online message boards.** Online message boards provide a means for people to share their opinions and have group discussions. Sometimes these posts are on sensitive topics that could have negative repercussions if discovered by the wrong parties.[11] Encryption could help protect these sensitive messages from unexpected eyes.

3. **Images.** Individuals are increasingly sharing images online, which later causes problems when unintended parties view them.[12] Encryption of images, especially schemes with revocable access, could help address this issue and give users more control of their online privacy.

In each of these cases, work will also need to be done to explore which key management schemes are most appropriate. For example, in email a PGP-based approach might be ideal, but when encrypting documents within an organization, IBE might be preferable.

[11] http://www.cnn.com/2016/02/04/asia/china-dissident-crackdown-goes-global/

[12] http://www.forbes.com/sites/kashmirhill/2012/03/06/what-employers-are-thinking-when-they-look-at-your-facebook-page/

References

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

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

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

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

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

[6] Zinaida Benenson, Gabriele Lenzini, Daniela Oliveira, Simon Parkin, and Sven Uebelacker. Maybe poor Johnny really cannot encrypt: The case for a complexity theory for usable security. In *Twenty-Third New Security Paradigms Workshop (NSPW 2015)*, pages 85–99, Twente, The Netherlands, 2015. ACM.

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

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

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

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

[11] Sascha Fahl, Marian Harbach, Thomas Muders, Matthew Smith, and Uwe Sander. Helping Johnny 2.0 to encrypt his Facebook conversations. In *Eighth Symposium on Usable Privacy and Security (SOUPS 2012)*, page 11, Washington, D.C., 2012. ACM.

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

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

[14] Simson Garfinkel and Heather Richter Lipford. Usable security: History, themes, and challenges. *Synthesis Lectures on Information Security, Privacy, and Trust*, 5(2):1–124, 2014.

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

[16] William C Garrison III, Adam Shull, Steven Myers, and Adam J Lee. On the practicality of cryptographically enforcing dynamic access control policies in the cloud. In *Thirty-Seventh IEEE Symposium on Security and Privacy (S&P 2016)*, San Jose, CA, 2016. IEEE Computer Society.

[17] Matthew D Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *Thirty-Sixth IEEE Symposium on Security and Privacy (S&P 2015)*, pages 305–320, San Jose, CA, 2015. IEEE Computer Society.

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

[19] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, Tao Wan, and Jianping Wu. When HTTPS meets CDN: A case of authentication in delegated service. In *Thirty-Fifth IEEE Symposium on Security and Privacy (S&P 2014)*, pages 67–82, San Jose, CA, 2014. IEEE Computer Society.

[20] Theresa Neil and Rich Malley. Rethinking mobile tutorials: Which patterns really work?, 2014. Accessed June 07, 2016. `https://www.smashingmagazine.com/2014/04/rethinking-mobile-tutorials-which-patterns-really-work/`.

[21] Jakob Nielsen. Enhancing the explanatory power of usability heuristics. In *Twelth ACM Conference on Human Factors and Computing Systems (CHI 1994)*, pages 152–158, Boston, MA, 1994. ACM.

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

[23] Chris Robison, Scott Ruoti, Timothy W. van der Horst, and Kent E. Seamons. Private Facebook Chat. In *Fifth ASE/IEEE International Conference on Social Computing (SocialCom 2012) and Fourth ASE/IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT 2012)*, pages 451–460, Amsterdam, The Netherlands, 2012. IEEE Computer Society.

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

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

[26] Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent Seamons. A usability study of four secure email tools using paired participants, 2016. Under Submission.

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

[28] Scott Ruoti, Jeff Andersen, Tyler Monson, Daniel Zappala, and Kent Seamons. A comparative usability study of key management in secure email, 2016. Under Submission.

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

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

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

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

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

[34] Deian Stefan, Edward Z. Yang, Petr Marchenko, Alejandro Russo, Dave Herman, Brad Karp, and David Mazieres. Protecting users by confining JavaScript with COWL. In *Eleventh USENIX Symposium on Operating Systems Design and Implementation (OSDI 2014)*, pages 131–146, Broomfield, CO, 2014. USENIX Association.

[35] Wenley Tong, Sebastian Gold, Samuel Gichohi, Mihai Roman, and Jonathan Frankle. Why king George III can encrypt. Unpublished, 2014.

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

[37] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure messaging. In *Thirty-Sixth IEEE Symposium on Security and Privacy (S&P 2015)*, pages 232–249, San Jose, CA, 2015. IEEE Computer Society.

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