PGP AUTH: USING PUBLIC KEY ENCRYPTION FOR AUTHENTICATION ON THE WEB

by

Derek Wueppelmann

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of
the requirements for the degree of

MASTER OF COMPUTER SCIENCE

Human Computer Interaction

at

CARLETON UNIVERSITY

Ottawa, Ontario
September, 2015

Abstract

The majority of authentication systems use text passwords, as they provide a flexible method of authenticating on a wide variety of devices. Unfortunately, having sufficiently strong passwords does not protect users against phishing or offline guessing attacks. In this thesis, we present a new authentication mechanism that uses PGP.

We iteratively designed PGP Auth, implemented it, and conducted user testing. Users rated the software highly and indicated that they would be very likely to use the software. They also liked the idea of having a single password to access their accounts and appreciated the security of using PGP as an authentication system.

We believe that with a refined user interface, PGP Auth is a viable authentication mechanism that addresses many of the security vulnerabilities of traditional text password authentication. We also provide recommendations to aid in the development of future versions of PGP Auth based on our results.

Acknowledgements

First, I would like to thank my supervisor Sonia Chiasson. Without her hard work an dedication this thesis would not have been possible. The advice given over the past three years has been an invaluable asset and has been greatly appreciated. Our meetings may have been short, but they were always informative and helpful.

I thank the members of my committee, Anil Somayaji and Anthony Whitehead for their feedback, guidance and perspectives. I would also like to thank Kasia Muldner for chairing the defense. To all of you, it was appreciated that you could make the time for my defense during the first weeks of the semester.

I would also like to thank my family. My wife Julie for being so patient with my long nights spent working on this thesis, especially nearing the end. My two daughters who were so patient and understanding when I responded “not right now, I’m doing school work.” to their requests. Without their patience, understanding, and support I do not know what I would have done.

Table of Contents

Abstract **ii**

Acknowledgements **iii**

List of Tables **viii**

List of Figures **ix**

Chapter 1 Introduction **1**

1.1 Motivation 1

1.2 Research Challenge 2

1.3 Contribution 2

1.4 Thesis Outline 3

Chapter 2 Background **4**

2.1 Introduction 4

2.2 Common Authentication Methods 5

2.2.1 Text Based Authentication 5

2.2.2 Other Knowledge Based Systems 10

2.2.3 Biometric Passwords 10

2.2.4 Token and Two-Factor Passwords 12

2.2.5 Summary of Common Authentication Methods 12

2.3 Public Key Encryption for Authentication 13

2.3.1 Public Key Encryption 13

2.3.2 Using PKE for Authentication 14

2.3.3 PGP 15

2.3.4 Summary of PKE for Encryption 17

2.4 Device Pairing 17

2.5 Methodological Background 18

2.5.1 Statistical Tests . . . 18
2.5.2 System Usability Scale . . . 18
2.6 Summary . . . 19

Chapter 3 PGP Auth - Prototype 1 **20**
3.1 Overview . . . 20
3.2 PGP Auth Accounts . . . 21
3.3 Protocol . . . 22
3.3.1 Prerequisites . . . 22
3.3.2 Additional HTTP Headers . . . 23
3.3.3 Data Transmission . . . 23
3.3.4 User Authentication . . . 24
3.4 Client UI Implementation . . . 28
3.4.1 Generating a PGP Auth Account . . . 29
3.4.2 Unlocking the Private PGP Key . . . 30
3.4.3 Creating a New Website Account . . . 31
3.4.4 Linking Devices Together . . . 32
3.5 Server Implementation . . . 33
3.5.1 Index Page . . . 33
3.5.2 Login Page . . . 34
3.5.3 Create New Account . . . 36
3.5.4 Protected Page . . . 36
3.6 Security . . . 37
3.7 Summary . . . 39

Chapter 4 User Study for Prototype 1 **40**
4.1 Study Design . . . 40
4.2 Setup . . . 40
4.3 Instrumentation . . . 41
4.4 Procedure . . . 42
4.5 Participants . . . 44

4.6 Results 45
4.6.1 Usage and Outcomes 46
4.6.2 Timing 47
4.6.3 Survey Results 49
4.6.4 Interviews 52
4.6.5 Users' Mental Models 53
4.7 Summary 56

Chapter 5 PGP Auth: Prototype 2 57

5.1 Wizard Interfaces 57
5.2 Modification: Linking Devices Together 58
5.3 Summary 60

Chapter 6 User Study for Prototype 2 61

6.1 Study Design 61
6.2 Participants 61
6.3 Results 62
6.3.1 Usage and Outcomes 63
6.3.2 Timing 64
6.3.3 Survey Results 65
6.3.4 Interviews 68
6.3.5 Users' Mental Models 70
6.4 Summary 71

Chapter 7 Discussion and Conclusion 72

7.1 Summary of Results 72
7.2 Implementation Recommendations 73
7.2.1 More Guidance 73
7.2.2 Instructions 74
7.2.3 PGP Key IDs are not User-Friendly 74
7.2.4 Server-Side Considerations 75

7.2.5 Recommendation Summary . . . 75
7.3 Discussion . . . 75
7.4 Limitations . . . 77
7.5 Future Work . . . 77
7.6 Conclusion . . . 78

Bibliography **80**

Appendix A PGP Auth - Information Sheet **86**
A.1 About PGP Auth . . . 86
A.1.1 Key Benefits . . . 87

Appendix B Survey Questions **88**
B.1 Demographic Information . . . 88
B.2 PGP Authentication . . . 90

Appendix C Interview Questions **93**

Appendix D Interview Diagram **94**

List of Tables

Table 3.1 Additional HTTP X headers created for PGP Auth 24
Table 3.2 PGP Auth process overview 25

Table 4.1 Outcome summary 47
Table 4.2 Time taken to complete the given tasks. 48
Table 4.3 Results from Likert questions 50
Table 4.4 User mental model accuracy 55

Table 6.1 Outcome summary 64
Table 6.2 Time taken to complete the given tasks. 65
Table 6.3 Results from Likert questions 67
Table 6.4 User mental model accuracy 70

List of Figures

Figure 2.1 Public key encryption 14

Figure 3.1 Successful login attempt 26
Figure 3.2 Successful new account creation 28
Figure 3.3 Generate new PGP Auth account interface 29
Figure 3.4 PGP Auth busy working overlay 30
Figure 3.5 PGP Auth passphrase entry 31
Figure 3.6 PGP Auth create new account prompt 31
Figure 3.7 Linking devices - step 1 33
Figure 3.8 Linking devices - step 2 34
Figure 3.9 Linking devices - step 3 35
Figure 3.10 Public index page 35
Figure 3.11 Authorization required page 36
Figure 3.12 Account protected page 37

Figure 4.1 Distribution of user competency responses 45
Figure 4.2 Distribution of user habits responses 45
Figure 4.3 Distribution of user responses 49
Figure 4.4 Distribution of user SUS responses 51
Figure 4.5 Correct mental model diagram 54
Figure 4.6 Low scoring mental model diagram 55
Figure 4.7 High scoring mental model diagram 56

Figure 5.1 Linking devices - modified step 1 58
Figure 5.2 Linking devices - modified step 2 58
Figure 5.3 Linking devices - modified step 3 59
Figure 5.4 Linking devices - modified step 4 59

Figure 6.1 Distribution of user competency responses 62
Figure 6.2 Distribution of user habits responses 62

Figure 6.3 Distribution of user responses 66
Figure 6.4 Distribution of user SUS responses 68

Chapter 1

Introduction

1.1 Motivation

The majority of authentication systems use text passwords. Password systems provide a flexible method of authenticating on a wide variety of devices that can provide a keyboard interface. These authentication systems are susceptible to common attacks, making passwords problematic. Recent attacks reported in news media have shown that password leaks and phishing are of particular concern [8, 42].

Users are commonly encouraged to generate strong passwords. However, this has been a difficult task for several reasons. One is that users have a difficult time determining what actually makes a secure password [60]. Another reason is that when sites employ poor password policies, they can actually reduce the security of the passwords used [63]. This is further compounded by the number of passwords a user is required to remember; users have on average 7 passwords and each one is reused on 6 different accounts [19].

Unfortunately, having sufficiently strong passwords does not protect users against phishing attacks [4, 48], nor against offline guessing attacks caused by password leaks [42]. Well-crafted phishing attacks can not only put an individual user at risk, but can even lead to a password leak scenario, as was experienced by RSA in 2011 [8]. In 2007, Gartner estimated that $3.2 billion was lost due to phishing attacks in the United States [43]. A later study by Gartner estimated that the average user who fell victim to a phishing attack lost $351, with banks and other financial institutions covering most of that cost [49].

While many systems encrypt stored passwords, an attacker with unrestricted access to the password hash can make repeated guesses on the password until a match is found. This leaves all accounts open to unauthorized access after a password leak regardless of the original password strength. In addition, since many user reuse passwords and usernames [19], a password leak on one system can compromise a user's accounts across multiple systems.

Alternative authentication mechanisms have been implemented to address these types of issues. Two factor authentication [2,24] increases security by requiring an attacker to breach two authentication mechanisms before gaining access. The usability cost is requiring the user to have access to additional information or a device any time they wish to access their accounts. Other authentication methods like biometrics, require specialized hardware, may not work in all circumstances, or have additional privacy concerns.

A new method of authentication is needed to address these issues. It needs to be secure, usable, and not require special hardware or have any privacy risks.

1.2 Research Challenge

The challenge addressed in this thesis is to design an authentication mechanism that is resistant to password leaks, phishing, bulk guessing attacks, and offline guessing attacks. We also want to reduce the memory burden on users and make the system easy to use. Our selected approach is to use PGP, a public key encryption system, as the underlying mechanism for an authentication system. Given this goal, we address the following specific questions:

- Can PGP be used as an authentication system?
- Can key-signing be used to enable devices to share accounts?
- Are users successfully able to use this system?

1.3 Contribution

This thesis contributes the following items to the literature:

- A new PGP Auth protocol that uses PGP as a mechanism for authentication. It uses public-key encryption to authenticate users. Using PGP's key-signing ability and web of trust, we allow multiple devices to share the same account on a server without directly involving the server in the linking process.

- A proof-of-concept implementation demonstrating the feasibility of the protocol.
- An evaluation showing that this type of interface is usable and that users like the concept. Improvements are still needed to address some remaining usability issues.

We accomplish this by building a prototype that implements our protocol and then performing user studies with the prototypes. The prototype went through one revision to improve the interface. User satisfaction was determined using post-study questionnaires. The usability of the interface was assessed using the industry standard SUS scale and measuring user performance with the prototype.

1.4 Thesis Outline

The rest of this thesis is organized in the following way: Chapter 2 outlines the background research with respect to passwords and other authentication systems. We also discuss public key encryption systems and how they have been used for authentication. We describe attacks against password systems and how public key encryption systems can provide an extra layer of protection against these attacks. In chapter 3, we describe our proposed PGP Auth protocol. We also outline the prototype client and server software created for the user study. Chapter 4 details the user study performed using the first prototype. We analyze the data gathered and evaluate our users' experience with the system. Based on these results, we updated the prototype as discussed in chapter 5. Chapter 6 provides the results from our second user study. We conclude with chapter 7, which includes a discussion of the results from both surveys, a description of future work, and a conclusion to the thesis.

Chapter 2

Background

2.1 Introduction

Since computers have had user accounts, it seems there has always been an issue with passwords. At least as far back as 1979, researchers were looking for ways to improve password use and security [46]. For Morris *et al.* it was simply trying to encrypt the list of passwords used on a system so that this information would not be given out inadvertently.

Authentication systems today consist of two components; user authentication and authorization. User authentication provides a mechanism to verify the user's identity to the system. Authorization grants access to features of a system. There are currently three basic types of authentication systems. Knowledge based systems require you to confirm something you know (e.g., text based passwords, graphical passwords, gesture based systems). Another variety relies on something you have with you (e.g., token keys, one-time passwords sent via SMS). There are also biometric passwords that rely on something you are or do (e.g., fingerprint, retinal scan, movements).

Each of these types of passwords have their own benefits and drawbacks. Research has assessed security concerns with all forms of authentication as well as proposals for possible solutions to known attacks. Many of the attacks that have been demonstrated are purely theoretical [34, 37, 52, 67]. Others have had actual implementations [8, 42, 62]. Solutions often offer alternatives to existing authentication systems [18, 24, 26, 32, 41] or propose additional tools to combat the potential security risks [44, 57, 67].

Advances in cryptography allow for the possibility of confirming a user's identity by what they are able to decrypt [5, 26]. PGP [12] is one implementation where a user generates two types of keys; one private key and one public. The public key can be made available to anybody; when used to encrypt a message, only the user that has access to the private key can decrypt it. This presents the possibility of using this as a method of user authentication and authorization.

2.2 Common Authentication Methods

There are several different kinds of authentication methods currently being used. The following sections outline some these methods.

2.2.1 Text Based Authentication

Most websites use a form of username and a text based password to grant access. Herley *et al.* [29] discussed many reasons why it is likely to stay that way for the foreseeable future as well. Although sites like Google [24] offer alternatives, they still also provide the traditional username and password interface.

Problems

Part of the problem with text based passwords is the fact that they are so common. In 2011, Hayashi *et al.* [27] estimated that an average user had about 11 accounts, with values ranging from 3 to 16. With this number of accounts, users tend to reuse their passwords. Florencio *et al.*'s 2007 study [19] found that users used an average of 7 passwords over a period of two months and that the average password was used for 6 separate accounts. The authors [19] also estimated that around 4.28% of Yahoo! users forgot their passwords during a two month period.

The average user enters a password about 75 times per day, 75% of those for logging into various web sites [27]. With so many interactions, having something that is memorable to the user is essential. However, having users select their own password may leave them open to security risks. Ur *et al.* [60] found that some users are unable to determine what makes a strong password. The passwords they chose as high security ones were actually easier to guess than those they thought offered lower security.

Many sites employ a password policy to aid users in selecting a more secure password. Komanduri *et al.* [35] explain that a well-crafted password policy will aid in increasing the entropy of a password. However, some policies actually hinder the process of choosing a better password [1]. This is further affected by some users' belief that a password is more secure just by adhering to any password policy [60].

With users having a large number of accounts, password reuse is common. In their sample, Florencio *et al.* found that on average a password was reused on 6 sites [19]. More secure passwords were reused less often, on an average of 4 sites. If users also used the same username on these sites, a potential attacker could gain access to multiple accounts at the same time if only one password was obtained. The attacker would still need to determine which sites the password was reused on.

Attacks

Florencio *et al.* [21] outlined the main attacks that face text based passwords. These were phishing, keylogging, brute-force attacks (offline and online), bulk guessing attacks, and special knowledge attacks.

Phishing: A phishing attack aims to solicit your password credentials or other details from you by earning your trust. These types of social engineering attacks usually consist of a carefully crafted email designed to look like a respected company requesting information from a user. The user trusts the contents of the email due to the perceived authenticity of the sender, then follows the instructions and provides the desired information to the phisher. In 2007, Florencio *et al.* [19] estimated that around 0.4% of users would fall victim to a phishing attack each year.

This can be a highly successful attack, especially when coupled with mining data from social media websites for targeted attacks. In 2011, RSA Security's internal staff were successfully phished leading to the master keys of the RSA SecureID tokens being stolen [8]. These types of attacks are particularly difficult to prevent since the user is deceived into providing information to a false authority. Many solutions rely on two-factor authentication to prevent phishing, discussed below in section 2.2.4. Badra *et al.* [4] described a way of using a Pre Shared Key (PSK) to lower the risk of phishing sites.

In a physical version of phishing, Orgill *et al.* [48] simulated an attack on user's accounts by using social engineering. In this scenario, the researcher posed as an auditor and asked employees questions, including asking for their username and password. The researcher was also able to gain access to the building after hours using similar techniques and find passwords written down at employees' desks.

Keylogging: Keyloggers capture a user's input to a system and then provide that data to the owner of the application at a later time. There are many different kinds of keyloggers, some are very sophisticated and capture screen-shots and other information in addition to keystrokes [62]. While legitimate keyloggers do exist[1], many are installed via viruses and malware to maliciously obtain user data. Herley *et al.* [28] suggests switching focus between the input field and a non-input field while typing random letters to make it difficult for a keylogger to obtain your password. However, this seems a bit impractical to perform as a standard operating practice.

Brute-Force: A brute force attack is performed by sequentially attempting all possible password combinations to gain access to an account. There are two types of brute force attacks. In an online attack, the passwords are tried to login to the actual account in real time. In an offline attack, the hashed version of the password is known. In this case, the attacker will compare the results of hashing their password guesses against the known value offline. If the two values match, then the password has been guessed.

Online brute force attacks should be infeasible. With a basic three-strikes policy that locks an account for a short period of time, an online attack would take 10 years to guess 1% of the password space of a 6-digit PIN [21]. This leaves protecting against unauthorized access to the stored passwords to prevent offline attacks. Hashing the passwords stored on the system is an important counter measure in case the passwords are accessed maliciously.

As early as 1979, research into passwords and security was being conducted. Morris *et al.* [46] looked at how long it would take to brute force passwords of various lengths on the existing hardware. For example. they found that it took 112 days for a 5-character password containing the 95 printable ASCII characters. In the same

[1]http://www.keylogger.org/ has a listing of keylogger software with feature breakdowns [33]

paper, Morris *et al.* also noted that the password file was stored as a plain text file, so any user with access to that file could gain access to any account they wished.

Bulk Guessing: In a bulk guessing attack, an attacker uses a large array of computers to make online login attempts to many accounts using the same password guess. Based on users' password habits, an attacker can craft the guess based on standard password generation rules and likely obtain access to a subset of accounts. Simple passwords are more susceptible to this type of attack. The more accounts attacked, the greater the chance that an attacker will obtain access to a subset of accounts.

After analyzing over 70 million passwords from Yahoo! Bonneau [10] concluded that an attacker could guess 1% of account passwords within 10 guesses in an online bulk guessing attack using high probability passwords. In the offline case, Bonneau's analysis indicated that to break half of the accounts, an attacker would only need to expend the effort required to brute-force a 20-bit password with an optimized guessing technique.

Special Knowledge: Special knowledge attacks are similar to bulk guessing attacks. An attacker reduces the number of password guesses by crafting their guesses to coincide with knowledge of a user. This could include phone numbers, relatives' names, birth dates, favorite animals, and many other types of information. While this may help to gain access to a specific user's account, Bonneau [10] suggests that crafting guesses to a group of users did not yield a much better success rate than using a generalized guessing strategy.

Solutions

A number of solutions have been proposed and implemented to prevent these types of attacks. With the exception of encrypting the passwords held on the server, these all concentrate on increasing password strength to reduce guess-ability and increasing the time required to perform an offline attack. These solutions can be grouped into server side and client side tools.

Server Tools: Of the server-side tools, password policies have been widely implemented by sites. However, Adams *et al.* [1] and Komanduri *et al.* [35] showed that some policies actually decrease the security of user passwords. Sometimes, policies can actually increase the guess-ability of the passwords used [63]. In addition, the more a site has a requirement to be usable, the less likely it is to enforce a strong password policy [20]. This leaves users with potentially weaker passwords on popular websites, putting them at risk for bulk guessing attacks.

Honeywords stores a set of false positive results for a given password [32]. When one of those false positives is used, a notification or alarm is triggered to prevent infiltration. This seems overly complex, and most system that want to protect against online guessing attacks employ either a delay between each password attempt, or a longer waiting period after a defined number of failed attempts, as suggested by Florencio [21].

Another method intended to improve password strength is strength meters. These rely on metrics to determine whether the password selected during password creation is sufficiently strong. Egelman *et al.* [17] performed a study which showed that when meters were present, users selected stronger passwords. Password meters can suffer the same problems as password creation policies [13].

Client Tools: Many security specialists encourage the use of a password manager. These are software tools that store all of a user's passwords and usually locks them with a single master password. The idea is that users can craft or generate hard to remember, but highly secure, passwords and then have the password manager remember them. The potential weak point is the password manager software which must be resistant to attack. For example, Zhao *et al.* [67] showed that many browser-based password managers are susceptible to theoretical exploits.

Password managers also suffer from portability issues. If the passwords are all saved on one device, you need access to that device in order to obtain your passwords. Some solutions, like LastPass [38], use a cloud based solution to store the passwords in a centrally accessible location. However, as was seen in June of 2015, the stored passwords may be vulnerable if the provider is compromised [42]. McCarney *et al.* [44] looked to solve this issue by using a cell phone as a digital wallet to store the passwords. This required the use of both a cell phone and a desktop to access a shared set of passwords. If either device was compromised, the passwords were still protected.

Other work has been done to improve users' abilities to select good passwords [36, 55]. Shay *et al.* worked on validating the wisdom that longer passwords without any other restrictions can be more secure. They found that while some of these policies do yield stronger passwords, cracking programs can also easily find common words and strings in longer passwords. Mnemonic phrase-based passwords were analyzed for their effectiveness by Kuo *et al.*. They showed that their user generated mnemonic passwords had better protection than a control group of passwords, but that cracking applications could be improved to target mnemonic passwords.

2.2.2 Other Knowledge Based Systems

The recent dominance of smartphone technology has made touch interfaces commonplace. As such, many phones now include touch-based authentication systems; Android's graphical password is just one example. While users generally like these types of interfaces [61], there are known and easy to perform attacks against them, in particular due to their small password space [3].

Traditional desktop systems have other methods of authenticating as well. Graphical passwords [7] offer one such method. Stobert *et al.* created a password manager that uses a graphical password to generate passwords for websites, bridging graphical passwords to be used in place of a text based password [57].

With bendable displays on the horizon, a new form of password entry will be possible. In her work, Maqsood [41] looked at the usability of bend passwords. After a bit of a learning curve, users we able to remember their bend passwords with about the same recall as PINs. This type of interface requires highly customized hardware at the moment.

2.2.3 Biometric Passwords

Biometric passwords use some physical characteristic to identify you to a system [31]. The system analyzes the aspect presented and compares it against a pattern stored in a database to determine if there is a match. As Jain *et al.* [31] describe, different biometric aspects can be used depending on the security level required. Items such as gait could be used for low security systems, whereas fingerprint or retinal scans could be used for higher-security applications.

Saevanee *et al.* [53] looked at using passive biometric analysis of users' interaction with their cell phones to be used a continuous authentication system. Users' SMS text messages were analyzed for their keystroke behaviour as well as the linguistic contents of the messages. Saevanee *et al.* concluded that their technique offered a reasonable method to authenticate a user with an overall low error rate. In order to authenticate a user, a large enough sample of data is required to compare against known patterns. This would make authenticating on demand potentially problematic.

As early as 1997, there has been concerns about using biometric data [65]. This revolves around giving up immutable information that personally identifies you to another party. One possible solution was put forward by Bhargav-Spantzel *et al.* [6]. The authors propose a technique called zero-knowledge proof to prove that the system holds a private key without actually providing the key. A biometric component is then used by the user to prove the possession of the private key. The iPhone fingerprint scanner behaves in much the same way [15].

When the iPhone 5s launched, it had a fingerprint scanner that could be used to unlock the device. Within days of the release Reiger [52] was able to successfully perform an attack on the device to gain access. In addition, there are cases where the fingerprint scanner may not be able to recognize the user, so a backup password or PIN entry method is provided. This leaves the device open to the same attacks that a PIN or password protected device has on its own [15].

Biometrics rely on the perceived immutability of the personal characteristics being used for identification. Research by Mehrotra *et al.* [45] suggests that, at least for irises over a 10 year period, this may not be the case. In addition, many biometric systems require additional hardware (e.g., fingerprint scanner, camera, motion sensors). This reduces the number of scenarios where biometrics could be used.

2.2.4 Token and Two-Factor Passwords

Another method of authentication requires the user to have something with them to aid in the authentication process. This can either come in the form of a token that generates a PIN number, as is the case with RSA Security's SecureID solutions [66], or a secondary device which stores, receives, or generates the PIN to be used.

One example used by Google [24] sends the user a text message with a PIN to be entered after they have performed the initial login process. This requires that the user setup two factor authentication ahead of time and has access to their device for each subsequent login attempt.

BeamAuth [2] also performs two-factor authentication by using a bookmark stored in the browser to aid in the authentication process. The bookmark uses the anchor tag component to store the second authentication factor and JavaScript is used to provide the anchor value. The site will only grant access if the username, password, and anchor value are all correct. Users are required to have access to the bookmark in order to access the site.

2.2.5 Summary of Common Authentication Methods

Today, text based passwords are the most commonly used method of providing user authentication. These are open to some common attacks; phishing, bulk-guessing, and password leaks. Solutions that have been provided aim to improve the strength of users passwords with varying degrees of success.

Other types of authentication exist and are being used today. While these forms of authentication can provide a higher level of security than passwords they may not be possible to use in all situations. Some of these require specialized hardware not available on all devices. Other forms of authentication require the user to have access to a separate component to complete the authentication process, which the user may not have access to at all times.

2.3 Public Key Encryption for Authentication

Encryption is a method of encoding messages so that it can be read only by entities that know how to decode it [23]. Without additional work, a third party would not be able to understand the contents of an intercepted message. In order to decode the message, a key is required. This key can come in many forms, but in all cases it provides the necessary information to take the encoded message and translate it back into the original message sent. While it can be possible to decode a message without the key, a strong encryption scheme will require a infeasible amount of computational resources to decode.

2.3.1 Public Key Encryption

In 1976, Diffie and Hellman [16] described public key encryption (PKE), also known as asymmetric encryption, as a method to securely communicate between two parties. After generating a sufficiently large random number K, encrypting and decrypting functions are generated using K as the basis. These two functions operate by acting as opposites to each other, if a message is encoded with one function it can only be decoded with the other. The decrypting function would then be stored as a private key and the encrypting function would be given as the public key. Figure 2.1 shows how the keys are used to transmit a message between two users.

Messages can be sent confidentially to a user using their public key. In order to decode the message, a user would need access to the private key. A third party who intercepted the message would not be able to read the message being transmitted.

In their paper, Diffie and Hellman [16] outlined how PKE can be used to authenticate a user. The user can send a message encrypted using their private key, then anyone can verify the sender by using the sender's public key to decrypt the data. The other way to authenticate a user would be to send them a message using their public key; if the message is decoded and reported successfully then you know the user has access to the private key.

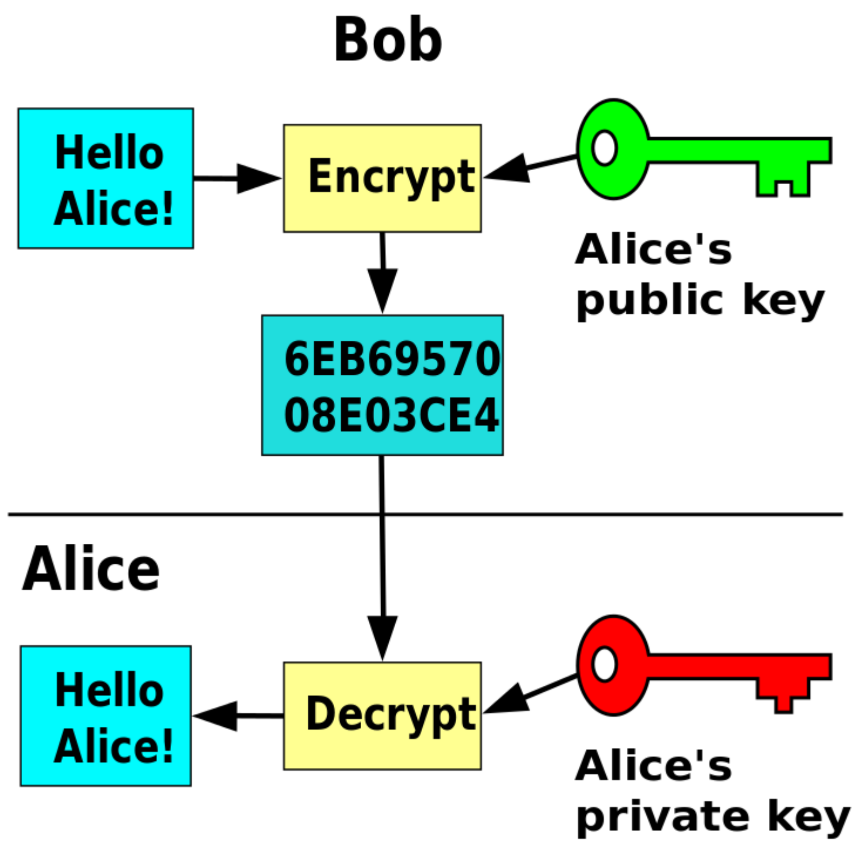


Figure 2.1: Diagram of how public key encryption works [25].

2.3.2 Using PKE for Authentication

Using PKE as an authentication system has been proposed by Halevi *et al.* [26]. In their system, the client encrypts their password using the server's public key before transmission. The server decrypts the password using the server's private key, then verifies the password before allowing access. They showed that an authentication attempt using public-key encryption would be resistant to server compromises and to offline guessing attacks (given a suitably strong encryption mechanism). An attacker would require access to both the set of passwords stored and the private key.

PKE systems also offer the ability to digitally sign messages instead of just encrypting them [23]. In this case, the message sent also includes an extra piece of information containing the digital signature. The signature is generated using the private key and the message. The signature can be verified by using the associated public key to authenticate the signature and the message it is attached to.

One implementation provided by the FIDO Alliance [5] called the Universal Authentication Framework (UAF) provides another method of authentication using PKE. In the UAF specification, users hold a set of private keys in a local key store that are used to communicate with a remote website. When a user accesses a new website that supports UAF, the user's local system generates a new key pair and stores the private key in the local key store. Once authenticated, the remote sever can verify the client by challenging them to return a message digitally signed with the private key. Users unlock their local key store with regular user authentication; the specification encourages authentication through biometrics, but a PIN or password could also be used. The UAF specification does not mention how accessing a single account from multiple devices would work and does not go into other uses of PKE.

Recently, Farrell *et al.* [18] proposed an experimental authentication protocol using PKE, as an alternative to existing authentication protocols performed over HTTP. In their scheme, the client generates a new key-pair for each website visited and digitally signs data provided by the server. If the authentication is successful, then the server continues with normal operation, asking for additional client credentials. Specifics on the client and server implementations are not offered.

2.3.3 PGP

PGP (Pretty Good Privacy) is a PKE scheme created in 1991 with the most recent OpenPGP RFC update occurring in 2007 [12]. PGP provides the ability to digitally sign other users' public keys; this is called key signing. Digital signatures enable a recipient to verify that the sender is legitimate and that the info has not be altered in route. PGP uses digital signatures to enable user A (i.e., the signer) to vouch for user B (i.e., the signee) by saying that they trust that B truly owns their key. Each digital signature is based on components of the signee's public key, including the key ID. This data is then encrypted using the signer's private key to form the digital signature. The digital signature is added to the signee's public key structure. A third user can validate a signee's key by using the signer's public key to decrypt the digital signature and validate the information against the signee's public key. When keys are signed by multiple users, this begins to form a web of trust [12]. This is a decentralized mechanism to validate key authenticity. A user can validate if a key is authentic if any of the signatures found on that key originate from keys that the user currently trusts.

When PGP generates a key, it produces a structure that holds several components [12]. These include: an identifier typically holding the owner's email address and name, the public key, the private key, a set of signatures, and a list of other users' public keys, also known as a key-chain. The set of signatures is initially empty, but can contain as many signatures as desired. Signatures are added through the process of key signing.

PGP is currently used for many purposes. In addition to encrypting messages sent between users, there are many different implementations of software that integrate PGP into email applications [40, 51]. Some instant messaging solutions also use PGP to encrypt their communications [22].

Usability

Historically, PGP has been difficult to use when used to encrypt, decrypt, and digitally sign email messages [56, 64]. In 1999, Whitten *et al.* [64] showed that users had a difficult time using the provided email interface for PGP and made several critical errors when trying to send messages which would compromise security. Sheng *et al.* [56] attempted a similar study 7 years later with similar result.

Security

PGP is among the most secure methods of transmitting data between two parties. However, the strength of PGP's security is related to the size of the key used. In PGP, the size of the key is measured in bits; the more bits used in the key, the larger the key size. In 2010, Kleinjung *et al.* [34] were successful in cracking a 768-bit key. This took the equivalent of 2000 years of computing time from 2.2GHz system. They concluded that a 1024-bit key would be about 1000 times harder to factor (i.e., determine the initial key used to generate the public and private keys). They recommended that

1024-bit keys should be phased out and replaced with higher bit keys because it may be possible to factor a key of this size by 2020.

Public key servers are systems that have been set up so that users can upload their public keys. These servers contain a large database of public keys and allow users to search for other users' keys through a number of criteria, including the key ID, email address, and name. A key ID is generated by taking a portion of the public key value as a 16-digit hexadecimal string. This acts as a unique identifier for the key, along with the user ID (typically the user's name and email address), key-length, and fingerprint (a hashed version of the public key and other data). Key IDs can be presented as a string of 8 or 16 hexadecimal digits. The 8-digit key ID is known as the short key and consist of the last 8-digits of the full 16-digit key ID. A known weakness exists, whereby an attacker could generate a key yielding an arbitrary 8-digit key [37]. This could allow an attacker to impersonate another user's key by having the same 8-digit key ID. It is therefore recommended that users use the full 16-digit key when performing searches.

2.3.4 Summary of PKE for Encryption

PKE systems like PGP provide a method of securely authenticating users without the transmitting passwords. Existing systems that use PKE as an authentication system rely on a single server key or a new key pair generated for each client-sever relationship. None of the existing methods make use of digital signatures to aid in the authentication process.

2.4 Device Pairing

When two previously unknown devices need to directly communicate with each other, a process is required to initiate the communication channel. Once this initialization has been completed, it can be remembered for future use. This process is known as pairing and is used, for example, by Bluetooth [9] devices. During pairing, one Bluetooth device presents the user with a code that needs to be verified by entering on the second device. Once correctly entered, the pairing process completes and the two devices can communicate in the future without user interaction.

A pairing system was also used in previous versions of FireFox Sync [58]. This used a server as an intermediary to communicate a secret that could be verified on the second device. Once the second device verified the secret presented in the first browser, the pairing was complete. Both browsers could then sync and share FireFox profile data across the two devices.

2.5 Methodological Background

When analyzing data obtained during a user study, various techniques can be employed. Statistical tests are used to determine if perceived differences in data sets are statistically relevant. The usability of an interface can be assessed through questionnaires, interviews, observation, and through performance measures collected by instrumented prototypes. We discuss the statistical tests used and the usability scale used in assessment.

2.5.1 Statistical Tests

The Wilcoxon RankSum test, also known as a Wilcoxon-Mann-Whitney test or a Mann-Whitney U test, is a non-parametric test used to evaluate if two groups of ranked data are statistically different [39]. In order to use this test, the two data sets should come from different user groups. This test also works well on data that is not normally distributed.

2.5.2 System Usability Scale

The System Usability Scale (SUS) was developed as a simple standardized method to assess the usability of a user interface [11]. It is comprised of ten 5-point Likert-scale questions ranging from strongly agree to strongly disagree. The results are tabulated so that the end score is a value between 1 and 100. It is strongly emphasized that the score is not a percentage; instead, the score is to be used to compare the interface against other interfaces. To aid in this, Sauro [54] took the results from 1000 interfaces, sorted the results and turns a SUS score into a percentile rank against his dataset. Using this scale, Sauro provides a method to convert the score into a letter grade. A score of 68 is considered to be average (50th percentile) and receives an ‘C’, a score of 80.3 or higher is in the top 10% of websites and receives a letter grade of ‘A’.

2.6 Summary

Authentication is part of using computers in everyday life. The most common form of authentication is text based passwords. These are pervasive not just because they have been used for so long, but also because they are the most easily implemented form of authentication. Users are obtaining more accounts that require passwords and so are faced with memorability and reuse issues related to having so many different passwords.

Attackers employ a number of different methods, such as Phishing, brute-force attacks, guessing attacks, and compromising servers. Simple text based passwords make many of these attacks easier to accomplish than more complex passwords.

Public-key encryption may offer a method that still maintains the flexibility of passwords while removing many of the risks. A public-key system would be resistant to compromises of the server, avoid issues with online and offline guessing attacks, be resistant to brute force attacks, prevent phishing attacks, and be resistant to key logging software. This is the goal of the FIDO Alliance with their UAF specification. However, their specification uses a password manager to hold all of the keys associated with the remote entities.

Key signing can also play a role in the authentication process. Signing a key places a stamp of approval on that key. If the approval was reciprocal, a system could trust that both parties agree to certain predefined actions, such as sharing accounts.

Chapter 3

PGP Auth - Prototype 1

3.1 Overview

We propose PGP Auth as a new method of authenticating to a website. This method uses PGP (Pretty Good Privacy) keys to act as a user account and verification system. This allows users to authorize and identify themselves to a remote website without having to provide a username or a password. PGP Auth has been designed to remove many of the complications found when using PGP.

With PGP Auth, each user's devices has its own privately-held authentication key as well as a publicly-available authentication key. The private key is kept secret on the device and is never transmitted or seen by any remote website. When a user attempts to login to a remote website, PGP Auth only provides the public key. The remote website will then validate the key against a publicly available version of this key. Using the device's public key, it will encode the information needed to complete the authentication process in such a way that only the device's privately held key can decode it. When a device successfully decodes the information, access is granted to the website.

For added protection, the device's private key is locked using a password of the user's choosing. This password is only used to unlock the private key and the password never leaves the device. This helps prevent against password theft and password guessing, since an attacker needs physical access to the device. Once unlocked, the device's private key will remain unlocked until the web browser is closed or the devices is turned off. This means that an attacker would not only need to know the password, but also have a copy of the private key in order to gain access to the user's website accounts.

PGP Auth allows users to link all their devices together so that they can seamlessly access all their website accounts regardless of the device they are using. Linking two devices requires a user to authorize each pair of devices against each other. For example, if a user wanted his or her phone and desktop computer to share the same set of accounts, the user would first authorize the phone from the computer, and then later, authorize the computer from the phone. While this authorization process may seem a little complicated, it is a one time process that ensures the user has control of both devices and prevents a third party impersonating the user. Subsequent devices would follow the same process with one of the previously authorized devices, forming a chain.

An explanation of how PGP Auth uses PGP to provide an account for users is provided in section 3.2. The general protocol devised for PGP Auth is explained in section 3.3. Section 3.4 describes the browser extension prototype that was created and used in the user study. Finally, section 3.5, gives a description of the server side implementation used for the user study. The implementations are examples of how to implement the protocol; other implementations are also possible.

3.2 PGP Auth Accounts

A separate PGP Auth account is generated for each of a user's devices. The account consists of three components stored in a single PGP key:

- Email address: Used as the PGP Auth user ID.
- Passphrase: Used to lock the private key.
- Key-pair: Generated as a standard PGP key-pair

Two PGP Auth accounts are linked together by a reciprocal signing of compatible keys. Compatible keys are keys that share the same user ID value. Each one will sign the other's public key and re-upload it to a common key server. Later, a linkage can be verified by obtaining both public keys from a key server and validating that each key has signed the other.

When performing authentication actions with a remote system, only the PGP public key needs to be transmitted. While a client application will always contain both the public and private keys, it may not have the most recent signing information.

For this reason, anytime linkages need to be checked, they should always be checked against keys pulled from a key server.

3.3 Protocol

Our proposed authentication method uses two components, a browser extension and a compliant website running a server side application. A website that supports PGP Auth provides additional HTTP headers that specify the resources required for authenticating using PGP Auth. The client side detects these headers and begins the authentication process. We considered security during our protocol design but PGP Auth has not been subjected to a thorough security analysis. Although beyond the scope of this thesis, security analysis should be done before deploying our protocol.

Instead of authenticating to the server using a traditional username and password, the authentication begins when the client provides their public key. The server validates that the key matches an existing account and encrypts the session cookie to be used when accessing the site using the given public key. The client decrypts the session cookie returned by the server and sets the local browser cookie to the decrypted value. Subsequent requests to the server will then be able to use this cookie value to validate the session so that additional authentication handshakes are not required.

Devices can be linked together so that they can share the same account information by using PGP key signing. When two keys are cross signed, each key signs the other, creating a link between the two accounts. The server can validate that the current key attempting to login has been linked with a key that already has an account in the system. Once the validation has occurred, the server can continue with the login process and grant access to the account associated with the linked key. This linking process allows for daisy chaining as well so that you do not need a master key to sign all other keys.

3.3.1 Prerequisites

In order for PGP Auth to be implemented correctly, the following prerequisites are required:

- Both the client (browser extension) and the server must have their own PGP key-pairs.
- Both the client and the server must be able to encrypt and decrypt data using their private keys as well as any given public key.
- The client must be able to intercept HTTP headers while a page is being loaded.
- The client must be able to redirect the browser to a different content page as needed based on the content headers received.
- The client has to be able to display prompts to the user while a page is being loaded in the browser.

3.3.2 Additional HTTP Headers

A client that supports PGP Auth will look for certain additional custom HTTP headers when a page is loaded. These headers are included as X headers, which are not part of an existing standard set of headers, but are allowed to be included as part of the HTTP protocol. Table 3.1 describes the content of these headers. When the correct headers are found, the client starts the PGP Auth process. Paths given in the headers are appended to the current domain name being access to create a complete URI.

3.3.3 Data Transmission

All encryption is done using PGP (Pretty Good Privacy). The client and server both generate their own public-private key-pair, which is used in encrypting and decrypting data. When data is sent either to or from the server, it will always be encrypted using the recipient's public key. Since only the associated private key can decrypt the data, the sender knows only the recipient can obtain the data and respond correctly. The transmission is done over standard HTTP or HTTPS protocols depending on the server's configuration.

HTTP Header	Description
X-PGP-AUTH-SERVER-PUB-KEY	The path to the server's PGP public key. (e.g. /path/to/private_key.pub)
X-PGP-AUTH-POST-PATH	The path where data should be posted using HTTP POST to complete the current action (login or new account creation). (e.g. /login)
X-PGP-AUTH-POST-VAR	The name of the post parameter used in the HTTP POST. (e.g. pub_key)
X-PGP-AUTH-CREATE-PATH	The path to access the account creation page. (e.g. /new_account)
X-PGP-AUTH-COOKIE-VALUE	The JSON object key name which holds the session cookie value (e.g. auth_session).
X-PGP-AUTH-COOKIE	The name of the cookie that stores the session key.

Table 3.1: Additional HTTP X headers created for PGP Auth

3.3.4 User Authentication

Table 3.2 provides an overview of the entire authentication process. An interaction diagram detailing a successful login attempt is shown in Figure 3.1. Figure 3.2 shows an interaction diagram detailing the successful account creation process.

Once the client has detected that the current web page supports PGP Auth, the authentication process begins automatically (Table 3.2, step 1 & 2). The initial step is for the client to obtain the public key of the server. It is obtained by accessing the path given in the X-PGP-AUTH-SERVER-PUB-KEY header on the current domain (Table 3.2, step 3). The client obtains an armored version of the key; this means it is in plain text, so the client will decode the key into its binary form. This avoids any character set transmission issues. The armored version of the key is 7-bit safe, meaning that it will be transmitted and received by the client without going through any character encoding or decoding.

With the server's public key, the client sends the server an encrypted copy of its armored PGP Auth public key (Table 3.2, step 4). The client posts the data to the path given in the X-PGP-AUTH-POST-PATH header using the post parameter name from the X-PGP-AUTH-POST-VAR header. To give the user a seamless experience, the client should use the HTTP POST method in an AJAX call to prevent added

Step	Client	Server
1.	Client requests page.	
2.		Server returns a page with additional PGP Auth Headers.
3.	Client sees headers, starts authentication process by downloading the server's PGP Public key.	
4.	Client encrypts client's PGP Auth public key using server's PGP public key and posts to server.	
5.		Server receives POSTed data and decodes encrypted public key using server's PGP private key.
6.		Server uses the public key ID to look up an existing user account on the system.
7.		If no account is found, server searches for cross signed keys. Server checks for associated accounts for all cross signed keys.
8.		Server prepares session data to be sent to client in JSON format if an account was found. Session data is encrypted using the client's public key. If no user was found, server returns a 403 Authorization Required status.
9.	Client receives status and content from server.	
10.	On success, client decrypts the content of the page and uses the information to set the browser cookie and redirect to the desired page. On a 403 failure, client shows a message about signing up for a new account. On other failures, client shows an error message.	

Table 3.2: PGP Auth process overview

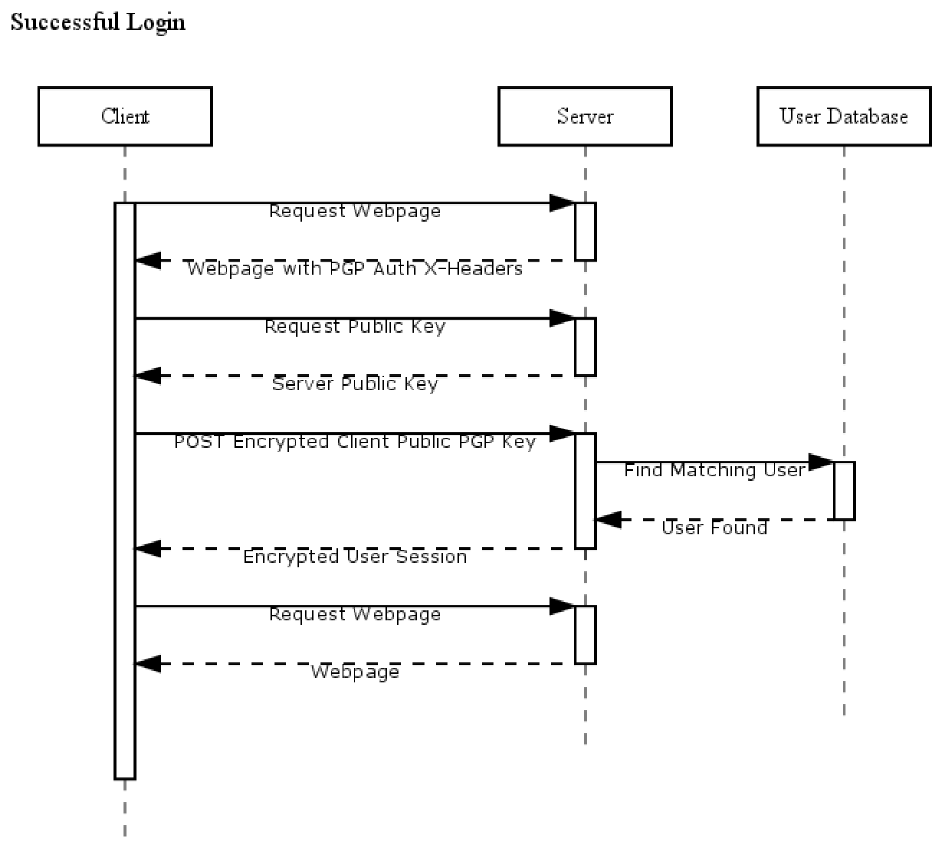


Figure 3.1: Interaction diagram showing a successful login attempt

pages from being displayed in the browser.

The server decrypts the encrypted POST parameter using the server's private PGP key (Table 3.2, step 5). The resulting key is decoded from its armored version into its binary format. The server then obtains the key ID from the given key. This key ID is used to look up an associated account in the server's list of users (Table 3.2, step 6). If a user is not found with a matching key ID, the server returns a 403 (Authorization Required) response and the authentication process stops (Table 3.2, step 8). If a successful match is made, then the user is granted access to the corresponding account.

Optionally, a server may allow cross linked keys to access accounts held on the system (Table 3.2, step 7). In this case, the server goes through the list of signatures for the given public key. For each signature found, the public key of that signature is downloaded from a public key server. The downloaded key's set of signatures is searched to see if it has been signed by the key given in the POST data. If a match is found, then the corresponding user is identified. A user is not likely to have more than a dozen cross-signed keys so this process should not take too long to process.

Identified cross signed keys can be kept for daisy chaining. The server repeats this process for every cross signed key. If any key along the chain matches a user account, then the server may grant access for the account.

When a matching user account is found, the server generates a new session for that user (Table 3.2, step 8). The server responds using a JSON formatted data structure with two components. The first is the session ID, accessed using the JSON attribute name found in the X-PGP-AUTH-COOKIE-VALUE header. The second component, accessed using the *redirect_url* JSON attribute name, holds the URL to where the client should redirect after setting the cookie. The entire JSON data structure is encrypted using the client's public key.

The client must interpret the server's response. (Table 3.2, step 9). If the client receives a 403 HTTP status code (authorization required), the client informs the user that an account was not found on the server and prompts the user to create one. To create an account, the client redirects the browser to the path found in the X-PGP-AUTH-CREATE-PATH header. The client is finished processing.

If the client receives a 200 HTTP status code (success) then it decrypts the contents of the page using the client's private PGP key to obtain the JSON encoded session information (Table 3.2, step 10). From this, the client obtains the session ID from the X-PGP-AUTH-COOKIE-VALUE header. The client sets a browser cookie using the name found in the X-PGP-AUTH-COOKIE header and the session ID as the value. Finally, the client redirects to the *redirect_url* from the decoded JSON object. The client is finished processing.

If the client receives an HTTP status code other than 403 or 200, it issues an error. This might occur if the key cannot be decoded or any unforeseen occurrence happens during the authentication process.

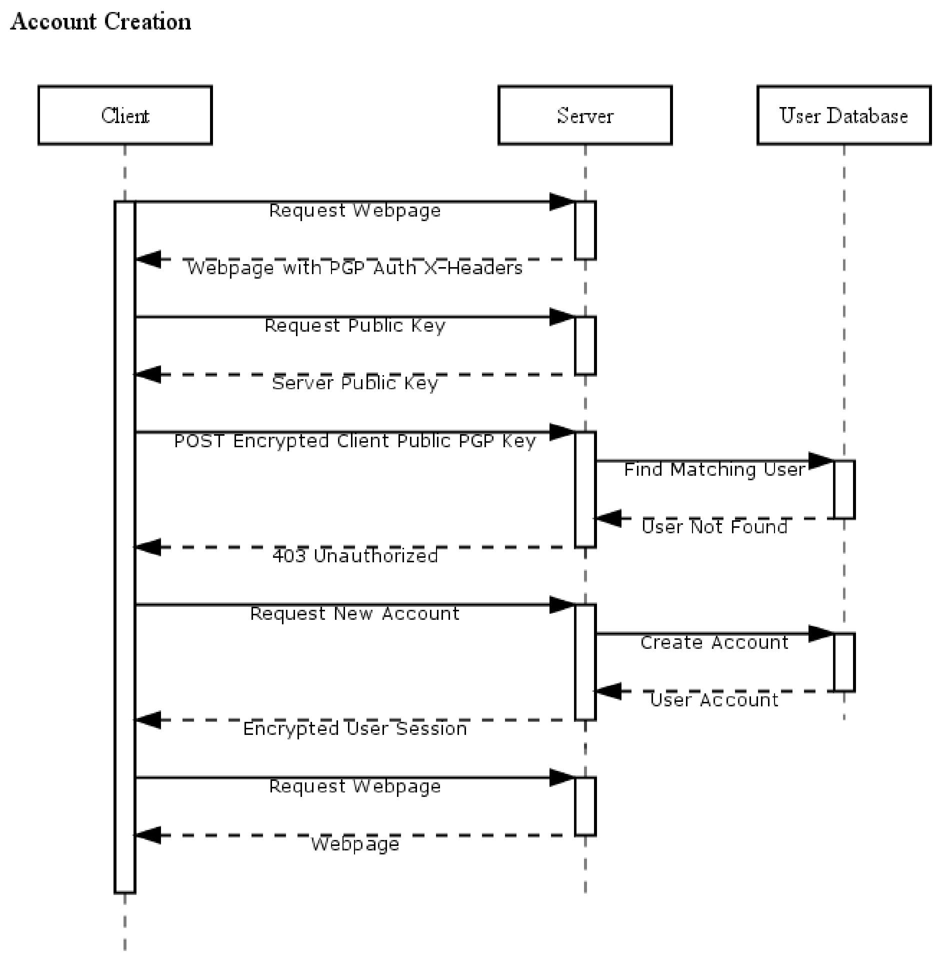


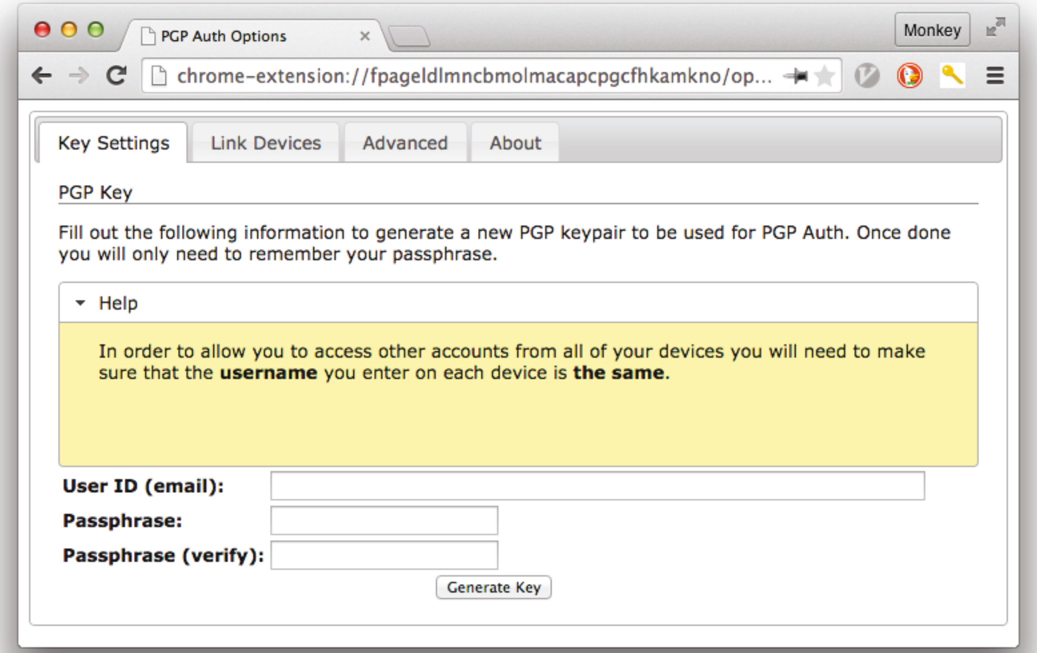
Figure 3.2: Interaction diagram showing a successful new account creation

3.4 Client UI Implementation

The client UI (User Interface) created for the user study provides all the basic functionality required to support PGP Auth. We built interface as a Chrome extension using JavaScript and HTML. The UI facilitates generating a new PGP Auth key and linking two devices together. These were designed with the intention that users could use the interface without prior knowledge of PGP Auth or PGP in general. Help documentation was provided. Figures 3.3 to 3.9 illustrate the features of the user interface.

3.4.1 Generating a PGP Auth Account

Generating a new PGP Auth key-pair is done by accessing the *Key Settings* tab (see Figure 3.3). Initially, the interface displays help documentation describing how to generate the key. Three input fields and a generate button are available. The input elements are the user ID (email address) and passphrase. The passphrase requires duplicate entry to validate that the user did not make a typo. In this implementation, any passphrase is allowed, including a single character, although stricter password rules could be enforced.



PGP Key

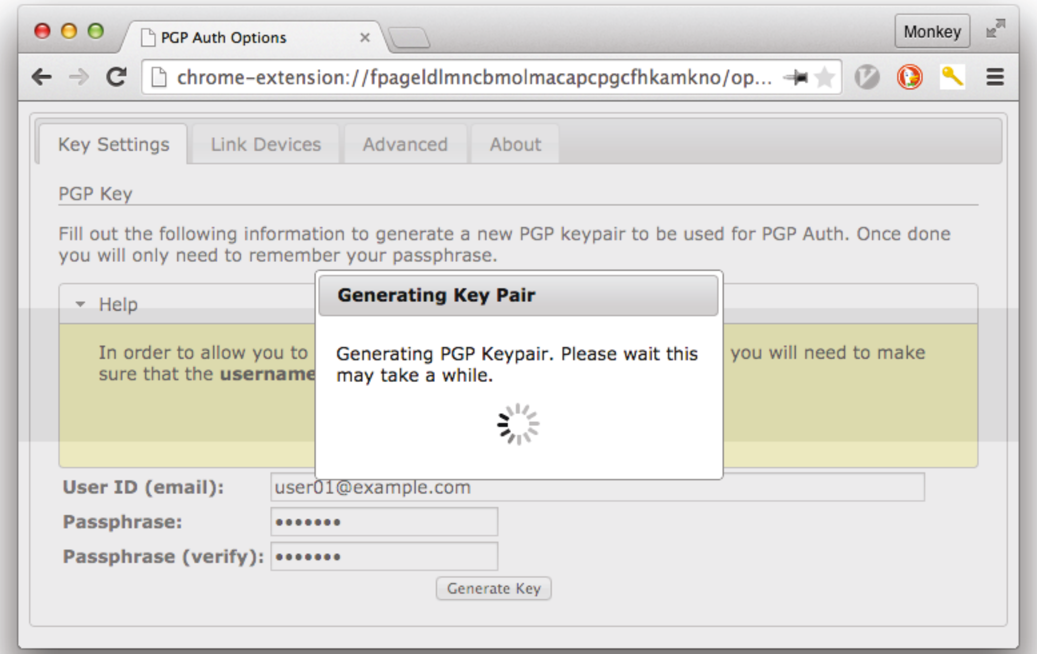
Fill out the following information to generate a new PGP keypair to be used for PGP Auth. Once done you will only need to remember your passphrase.

In order to allow you to access other accounts from all of your devices you will need to make sure that the **username** you enter on each device is **the same**.

User ID (email):
Passphrase:
Passphrase (verify):

Figure 3.3: The client interface to generate a new PGP Auth user account in the browser.

After a user clicks on the *Generate Key* button, a overlay is added to the screen indicating that it is generating the key (see Figure 3.4). This lets the user know that the system is busy processing the request as generating a key can take a minute on some slower systems.



PGP Key

Fill out the following information to generate a new PGP keypair to be used for PGP Auth. Once done you will only need to remember your passphrase.

In order to allow you to … sure that the **username** … you will need to make

Generating Key Pair

Generating PGP Keypair. Please wait this may take a while.

User ID (email): user01@example.com

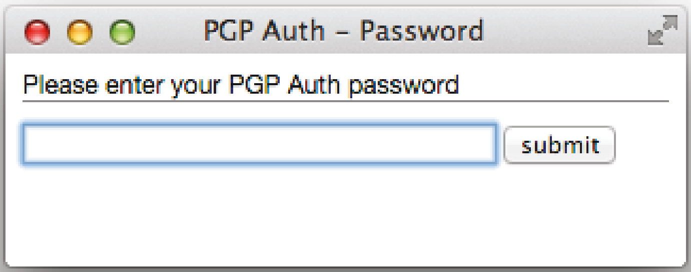
Passphrase: •••••••

Passphrase (verify): •••••••

Figure 3.4: When generating a key, the interface displays an overlay indicating that it is busy working. This includes a “spinner” graphic.

3.4.2 Unlocking the Private PGP Key

In order to authenticate to a website or link devices together, a user must unlock their private key using the password prompt (see Figure 3.5). If the passphrase is correct, the key is unlocked and stored for the remainder of the browser session. If a match was not found then the user is notified and asked to re-enter the password. A user is given three attempts to unlock their private key before the UI closes the passphrase window and issues an error. In our prototype, the user could try again immediately. However, a production implementation may introduce a waiting period, a lockout period, or a recovery process to prevent a brute force attack on the local system.

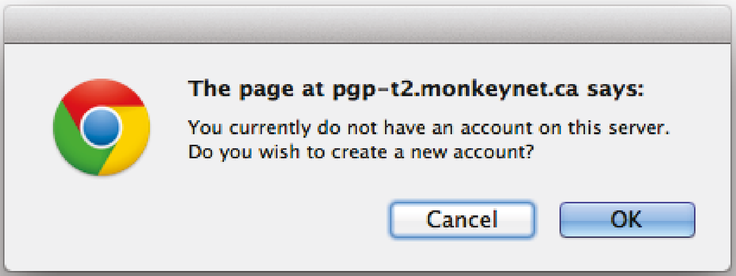


Please enter your PGP Auth password

Figure 3.5: The passphrase prompt appears when the PGP Auth client needs to unlock the user's private key

3.4.3 Creating a New Website Account

In the event that a user does not have an account for the website, the PGP Auth extension will prompt the user to create one (see Figure 3.6). A user can elect to create a new account by clicking on the *OK* button, or canceling the process. If a user elects to create an account, the client will direct the user's browser to the account creation page found in the HTTP header data. Otherwise, it stops processing and the original, non-protected page requested by the browser is displayed to the user.



The page at pgp-t2.monkeynet.ca says:

You currently do not have an account on this server.
Do you wish to create a new account?

Figure 3.6: The prompt shown to a user when they currently do not have an account on the website. Clicking on *OK* takes the user to the account creation page.

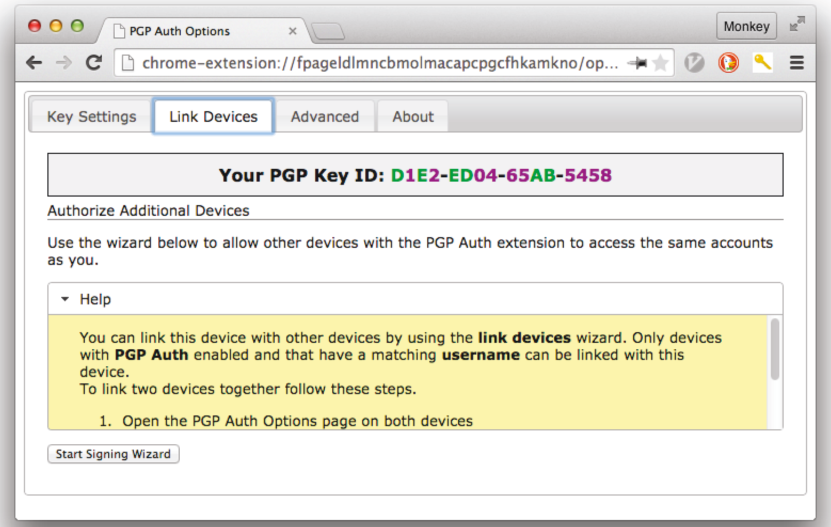
3.4.4 Linking Devices Together

To facilitate the linking of two devices, the PGP Auth extension has a separate tab interface for this process. It consists of a three step process that needs to be completed on both devices.

- **Step 1:** Access the *Link Devices* tab
  Figure 3.7 shows an example of this interface. Help instructions explain how the process works. To aid in key verification, the user's PGP key ID is colour coded and broken into 4 character segments at the top of this interface. The key ID is visible throughout the signing process. The *Start Signing Wizard* button is used to start the signing process.

- **Step 2:** Select the key to link with
  The system fetches all of the compatible keys from a public key server and presents them to the user (see Figure 3.8). The instructions inform the user that the key ID should first be verified against the ID value found on the other device before signing the keys. Keys that have already been signed are shown in a separate section. The user can select the keys they wish to sign from the list of unsigned keys, allowing multiple devices to be linked at the same time, provided the user has them all on hand to verify the matching key IDs. Users start the signing process by clicking on the *Sign Selected Keys* button. Each selected key is signed and uploaded to the public key server. An overlay indicates that the signing progress is underway.

- **Step 3:** Signing completed
  Once all keys are signed, the user is informed that signing has completed (see Figure 3.9). Instructions remind the user this process needs to be done on both devices to complete the link. The process can be repeated to sign additional keys by clicking the *Sign Additional Keys* button.



Your PGP Key ID: D1E2-ED04-65AB-5458

Authorize Additional Devices

Use the wizard below to allow other devices with the PGP Auth extension to access the same accounts as you.

Help

You can link this device with other devices by using the **link devices** wizard. Only devices with **PGP Auth** enabled and that have a matching **username** can be linked with this device.
To link two devices together follow these steps.

1. Open the PGP Auth Options page on both devices

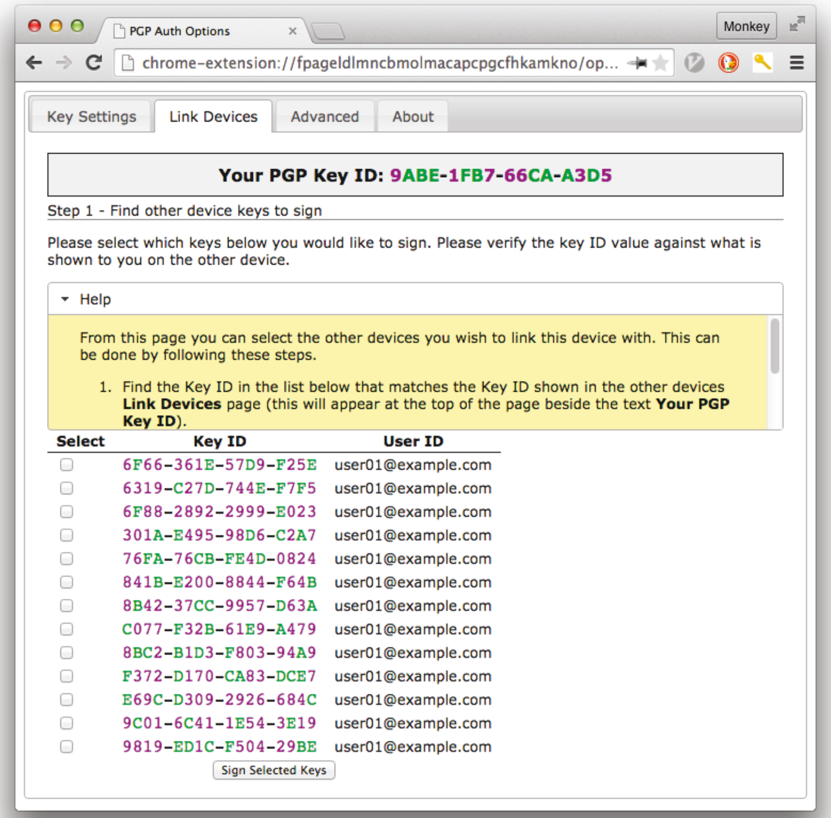
Figure 3.7: The first step of the linking devices process.

3.5 Server Implementation

A very basic website was created using Perl that supports all the functionality required for PGP Auth. Four pages represent the types of pages that would be found on a typical website with both public and account-restricted content. These consist of a public index page, a login page, a new account creation page, and an account-protected content page. These pages were created for the user study to demonstrate PGP Auth's functionality. A real implementation would include actual protected content.

3.5.1 Index Page

The index page is a publicly accessible page that presents the user with a link to log in to the site (see Figure 3.10). Users first visit this page as a means of entry into the protected content. As this page is publicly accessible and does not provide any PGP Auth functionality, it is accessible without the PGP Auth extension installed. Users access the protected content by clicking on the *Sign-In or Create New Account* link.



Your PGP Key ID: 9ABE-1FB7-66CA-A3D5

Step 1 - Find other device keys to sign

Please select which keys below you would like to sign. Please verify the key ID value against what is shown to you on the other device.

From this page you can select the other devices you wish to link this device with. This can be done by following these steps.

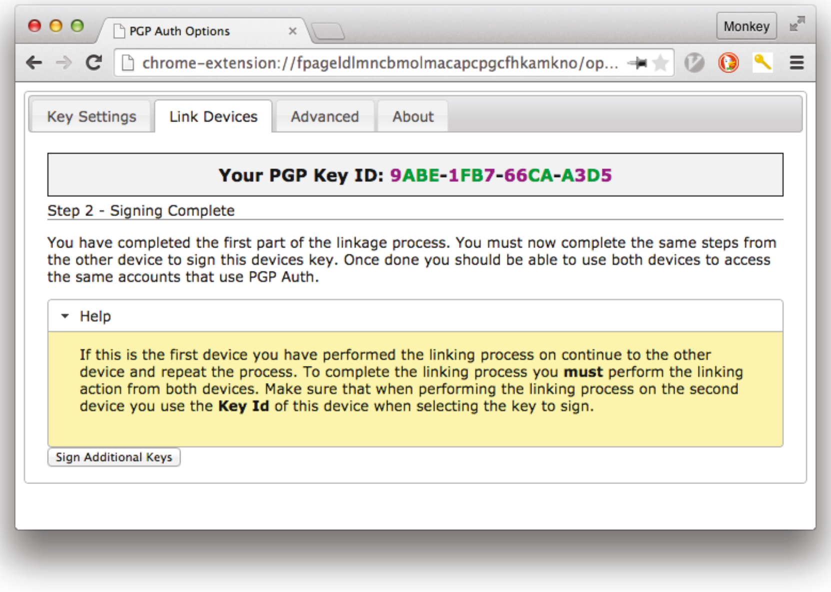
1. Find the Key ID in the list below that matches the Key ID shown in the other devices **Link Devices** page (this will appear at the top of the page beside the text **Your PGP Key ID**).

Select	Key ID	User ID
☐	6F66-361E-57D9-F25E	user01@example.com
☐	6319-C27D-744E-F7F5	user01@example.com
☐	6F88-2892-2999-E023	user01@example.com
☐	301A-E495-98D6-C2A7	user01@example.com
☐	76FA-76CB-FE4D-0824	user01@example.com
☐	841B-E200-8844-F64B	user01@example.com
☐	8B42-37CC-9957-D63A	user01@example.com
☐	C077-F32B-61E9-A479	user01@example.com
☐	8BC2-B1D3-F803-94A9	user01@example.com
☐	F372-D170-CA83-DCE7	user01@example.com
☐	E69C-D309-2926-684C	user01@example.com
☐	9C01-6C41-1E54-3E19	user01@example.com
☐	9819-ED1C-F504-29BE	user01@example.com

Figure 3.8: The second step of the linking devices process: The user selects which key IDs to sign.

3.5.2 Login Page

When a browser without a valid session attempts to access protected content, it is redirected to the login page. This page provides the additional HTTP headers that trigger the extension to begin the PGP Auth authentication process. To accommodate browsers without the PGP Auth extension, or users with trouble authenticating, the login page displays information about the PGP Auth login process (see Figure 3.11). During the login process, this page may be visible in the user's browser. When the user's PGP Auth key is unlocked and the system has negotiated a successful login, a valid session is created and the user will be redirected to the protected content automatically. If a user happens to access the login page directly or is redirected to



Your PGP Key ID: 9ABE-1FB7-66CA-A3D5

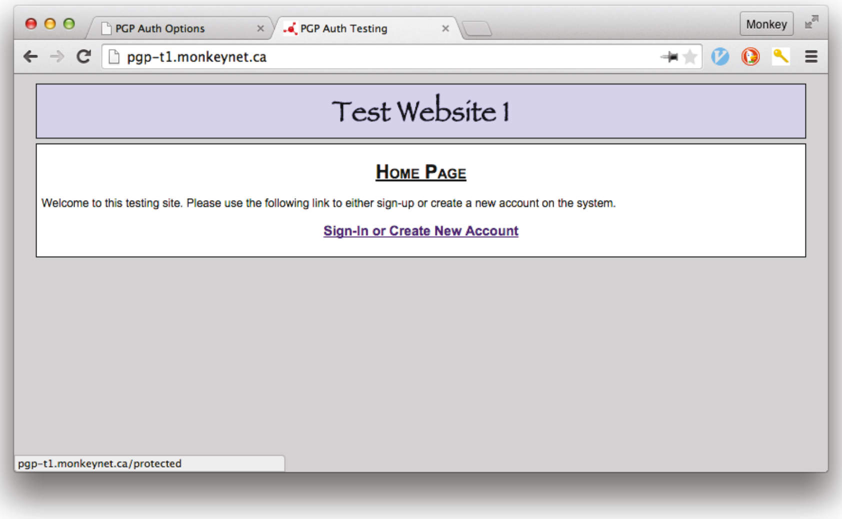
Step 2 - Signing Complete

You have completed the first part of the linkage process. You must now complete the same steps from the other device to sign this devices key. Once done you should be able to use both devices to access the same accounts that use PGP Auth.

Help

If this is the first device you have performed the linking process on continue to the other device and repeat the process. To complete the linking process you **must** perform the linking action from both devices. Make sure that when performing the linking process on the second device you use the **Key Id** of this device when selecting the key to sign.

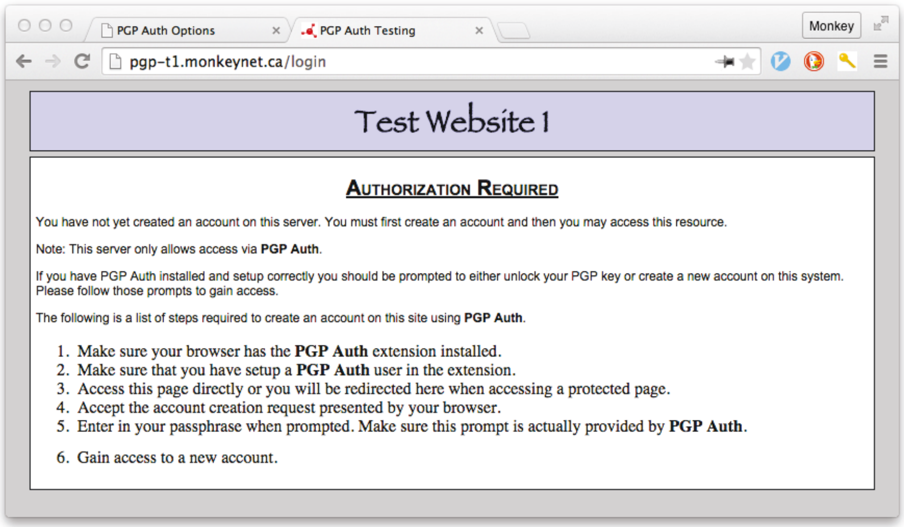
Figure 3.9: The third step of the linking devices process: The user is informed that the linking was successful and that they need to complete this process on the other device(s).



Welcome to this testing site. Please use the following link to either sign-up or create a new account on the system.

Figure 3.10: The publicly accessible index page of the PGP Auth test website.

the login page, as long as the browser has a valid session, it will be redirected to the originally requested content or a default protected content page.



Authorization Required

You have not yet created an account on this server. You must first create an account and then you may access this resource.

Note: This server only allows access via **PGP Auth**.

If you have PGP Auth installed and setup correctly you should be prompted to either unlock your PGP key or create a new account on this system. Please follow those prompts to gain access.

The following is a list of steps required to create an account on this site using **PGP Auth**.

1. Make sure your browser has the **PGP Auth** extension installed.
2. Make sure that you have setup a **PGP Auth** user in the extension.
3. Access this page directly or you will be redirected here when accessing a protected page.
4. Accept the account creation request presented by your browser.
5. Enter in your passphrase when prompted. Make sure this prompt is actually provided by **PGP Auth**.
6. Gain access to a new account.

Figure 3.11: The page displayed when authorization is required.

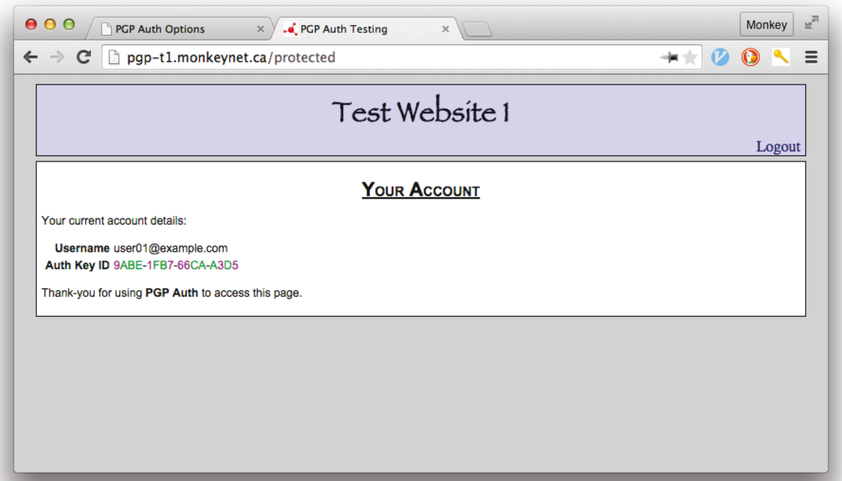
This website supports cross linked keys. When a user logs in with a PGP Auth key that does not directly map to an account, the linked keys will be checked for valid accounts. If a match is found, they are granted access to the cross signed account. This happens seamlessly on the server by looking up the key data on a public key server.

3.5.3 Create New Account

When a user creates a new account they are directed to the account creation page. The account creation page generates a new account entry in the server's database using the client's public PGP key and then redirects to the protected content page. The email address is also extracted from the public PGP key to be associated with the account. Other websites may want to capture additional user information before generating an account, however for this test website we included only the core requirements for PGP Auth.

3.5.4 Protected Page

A successfully authenticated user is redirected to the protected page (see Figure 3.12). This page gives some basic information about the current account, including



Your Account

Your current account details:

Username user01@example.com
Auth Key ID 9ABE-1FB7-66CA-A3D5

Thank-you for using **PGP Auth** to access this page.

Figure 3.12: The account protected page. This page shows some basic information about the currently logged in account.

the account's public PGP key ID. If a user is accessing this page using a cross signed key, the key ID of the device that created the account is shown. On a real deployed site, the protected content would be shown.

3.6 Security

PGP Auth is designed to prevent a user's password from being compromised on a remote website. This authentication method does not store a password remotely; a passphrase is instead associated with the locally stored private key of a device. Remote websites and scripts loaded by remote websites do not have access to the private key or the passphrase used.

The authentication occurs as a transaction using encryption. We chose to use PGP as it offers a very high level of encryption that is difficult to crack [50]. The effectiveness of this encryption is directly related to the length of the key used. Klienjung *et al.* [34] were able to successfully factor a 768-bit RSA key in 2009; this is where the private-key is derived using an encoded message and the public-key. This factoring effort used the equivalent of 2000 years of computing time for a standard 2009 desktop system. Based on the work by Klienjung *et al.*, we set the default key to be

2048-bit; this should prevent factoring attacks for the foreseeable future.

While the authentication token is a public item, the actual session information is sent via an encrypted channel. This would prevent a man-in-the-middle attack as the encoded contents would need to be decrypted using the intended device's private key. Both parties rely on the ability of only the recipient being able to decode the contents of the messages transmitted as the basis of the authentication and authorization.

Sharing accounts across devices relies on a reciprocal key signing from both devices. The server checks that both keys have signed one another and uses this information to grant access to a related key. This is open to social engineering attacks, where an attacker could get a user to inadvertently sign the key for their device. The aim of our interface was the reduce this risk so that users would not fall victim to this type of attack. A mechanism for revoking the key or signing should be implemented in the event that the user chose the wrong key, or no longer wants to have the devices linked.

There is a chance of a local attack if the client's browser is compromised and the decrypted session ID is sent to an attacker. This is possible if the communication between client and server is over HTTP and the browser transmits the cookie information in the clear, as opposed to using HTTPS. There is, however, no new risk here over standard session cookies. Due to this, it is strongly recommended to use HTTPS for the transmission protocol to reduce man-in-the-middle attacks. The possibility also exists to complete the entire data transaction using PGP encrypted packets instead of using HTTPS. The advantage of PGP packets is that a constant authentication system could be achieved; the downside is that a larger overhead is placed on performing the encryption. It is important to note that a local attack would need to gain access to both the private key and the passphrase to pose a threat.

Unlike other types of authentication, PGP Auth is not susceptible to many common attacks. Guessing attacks against a user's web accounts are not possible as the password is never sent and guessing the private-key is computationally hard. Phishing attacks would have to be reworked to get the user to sign a new key. Interfaces should make it clear that validating the keys is important and should only be done for devices the user controls. Shoulder surfing is not an issue as the attacker would need both the passphrase and the private-key; having the passphrase on its own does not provide the attacker any information to gain access to the user's accounts. If, however, an attacker were to gain access to the device, or the private-key and the passphrase, they would gain access to all of the user's accounts that use PGP Auth.

3.7 Summary

These two components, the browser extension and the server implementation, provide a working example of PGP Auth. It is possible to use the browser extension to create a new PGP Auth account on multiple devices and link them together. With these accounts setup, a user can access a website running the server implementation and create or log in to an account. With two devices linked together, a user can access accounts that were created using a different device. Authentication is done without having to provide a password to a remote website, while still allowing authorization and authentication checks to prevent unauthorized access.

Chapter 4

User Study for Prototype 1

4.1 Study Design

With the prototype built, we wanted to determine how users would interact with the software. Were users able to navigate the user interface successfully and perform all the actions required to create a new auth account, log into a remote websites, link two devices together, and use the linked devices to access their existing accounts? During all of these tasks, would users perform actions that compromise the security of their accounts? Would a user's mental model of the software aid in their usage of the software? Finally, would users like the software and want to use it in the future?

We obtained clearance from Carleton's Research Ethics board for a lab study using one-on-one sessions lasting 30 to 45 minutes. Each session consisted of four parts:

- Read a one page information sheet on PGP Auth.
- Complete a set of tasks using two laptops with the PGP Auth extension.
- Answer an online survey.
- Participate in a short interview session.

4.2 Setup

The two laptops used for this study were MacBook Pros. Laptop 1 had a 13-inch display and Laptop 2 had a 15-inch display. They both had version 43 of the Google Chrome web browser and the PGP Auth browser extension installed. When a user was presented with a laptop, it had two tabs open. One tab held a web page with links to the websites being used for the study, the other tab held the PGP Auth options page. The PGP Auth options page was active and the *Key Settings* tab was visible. Users began the first task on the 13-inch MacBook.

The one page PGP Auth description is provided in Appendix A. It was designed as a general information sheet to introduce users to the benefits of PGP Auth and give a high level overview of the system and how it works. Users did not need to remember everything on the page to use the system.

Prior to the study, we setup user IDs (email addresses). For each user ID, a PGP Auth account was created and stored on the private key server set up for the test. This extra key simulates an attacker trying to gain access to a user's accounts, since it was not generated by the participant. If a user selected this key to link with their device it would represent a critical error in the process. Each user was given a user ID to use for the duration of the study. The user ID also links the user's interview and online questionnaire responses to the timing data captured during their session.

Two testing websites were created for the study. Both websites ran the software described in section 3.5 and differed in the logos displayed on every page and the colour schemes used. Website 1 had the logo text of *"Test Website 1"* in black text on a light blue background. Website 2 had the logo text of *"Test Website 2"* in black text on a light green background. Each website had its own PGP key-pair and separate PostgreSQL databases to hold user accounts.

4.3 Instrumentation

The PGP Auth interface was instrumented to record timing information when users took certain actions. This allowed us to see how long it took users to enter passwords, log in to the website, create a new account, and link the two devices together, as well as how many password attempts were needed. At the end of each session, the timing data was stored server-side in a separate delimited file containing timestamps and actions. The file was linked to the user ID.

4.4 Procedure

Users were welcomed to the testing location and asked to complete the consent form. Once the consent form was completed, we explained how the rest of the study was going to proceed: read a one page information sheet on PGP Auth, complete a short set of tasks on the two laptop computers provided, complete an online survey questionnaire, and then complete a short five to ten minute interview session.

When users finished reading the PGP Auth information sheet, we gave them a user ID for the duration of the study. We told the users that passwords and passphrases would not be recorded and that they could use whatever password they wished. We asked users to complete the following 7 tasks:

- **Task 1:** Create PGP Auth Account on Laptop 1
  Using Laptop 1, we asked the user to create a new PGP Auth user account (See section 3.4.1 for details).

- **Task 2:** Create an Account on Website 1 Using Laptop 1
  We directed users to the tab that held the links to web pages used in the study and indicated which link they should click to access the testing website. Once there, users determined how to create the website account themselves (See section 3.4.3 for details).

- **Task 3:** Create an Account on Website 2 Using Laptop 1
  Users were directed to Website 2 and asked to create the second account. The second website account creation process did not require entering in their passphrase.

- **Task 4:** Create PGP Auth Account on Laptop 2
  Users were instructed to create a new PGP Auth account on the second laptop.

- **Task 5:** Link Laptop 1 and Laptop 2
  With the account created, we asked users to link both laptops together without guidance from the experimenter (See section 3.4.4 for details).

- **Task 6:** Access Account on Website 1 Using Laptop 2
  With the two laptops linked together, we asked users to log into Website 1 from Laptop 2. Passphrase entry would not be required during this task.

- **Task 7:** Access Account on Website 2 Using Laptop 2
  Finally we asked users to log in to Website 2 using Laptop 2. No password was required.

During the tasks, we monitored the user's actions and noted their success and any problems encountered. We decided on 5 different levels of success to be applied to each task. If a user was struggling with the task, we would aid them by providing small hints only after a few minutes of them attempting the task on their own. If, even after a few hints, the user was still unable to perform the task, we provided as much instruction as needed to get them on to the next task. Each time a hint or additional help was provided, this was noted. Each task had 5 possible outcomes:

- **Completed:** The user successfully completed the task without any help.

- **Completed With Help:** The user required a little bit of help. Small hints were given (e.g.,"You need to complete the process on the second laptop", when users thought they were finished linking the devices together.)

- **Completed With Errors:** The user made an error during the task that required the experimenter to intervene in order to reverse the mistake. This never occurred during our study.

- **Completed With Critical Errors:** The user made an error which would expose them to a security problem (e.g., if the user were to sign all of the keys presented during the linking process, this would be considered a critical error).

- **Failed:** The user was unable to complete the task, even with help from the experimenter. This never occurred during our study.

After completing the tasks, users filled out an online questionnaire (see appendix B). If users had questions about the questionnaire, the experimenter provided clarification. The questionnaire contained four sections. First, we asked some general demographic information questions. This was followed by questions pertaining to the user's password usage habits; these consisted of simple numeric entry, multiple choice, or 5-point Likert-scale questions. We then asked users to rate their experience with PGP Auth, using 5-point Likert-scale questions. Finally, we included the System Usability Scale (SUS) questions [11, 54]. The SUS questions are a quick and reliable tool used to determine the usability of interfaces.

We concluded with a short interview session. The interview session consisted of 7 main questions and one optional question (see appendix C). This included asking users to draw a picture showing their understanding of how the system worked. The session was audio recorded, and a video of the users' hands was taken to see the drawings as users explained their mental model of the system. We also wanted to get feedback from the user with suggested improvements to the system or problems they encountered during the study. The optional question was asked if users made a critical error during the study so that additional information could be obtained on how to make the interface better for future users.

At the end of the interview, users were thanked for their time and compensated with $10. The experimenter answered any remaining questions the user had.

4.5 Participants

We recruited a total of 12 participants for the user study using the first prototype. Participants were recruited on campus using a combination of posters and existing mailing lists. Our participants ages ranged from 20 to 63 years of age with an average age of 29. There was an even split of men and woman (6 men, 6 women) with the overall majority (75%) of participants being students. Most of our participants indicated they had 11-15 online accounts requiring passwords (7 users). Another 3 indicated they had between 5 and 10 accounts. One user responded that they had more than 21 accounts. Participants reported spending on average of 6.17 hours a day on a computer, with a standard deviation of 3.1. In addition, 75% of users reported using a single password for all accounts, 17% had a couple of passwords, and one user indicated they had one password per account.

Participants were asked a number of questions pertaining to their self described

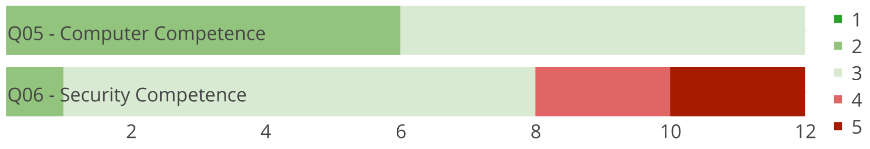


Figure 4.1: Distribution of user responses to competency questions. 1 = expert, 5 = no knowledge.

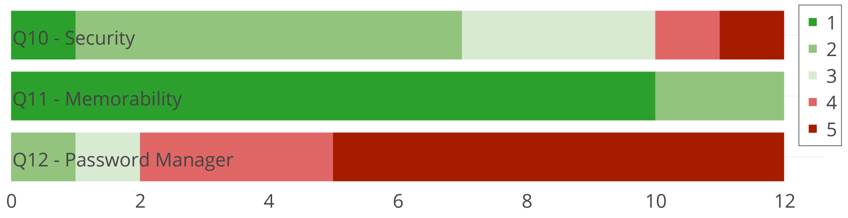


Figure 4.2: Distribution of user responses to password habits questions. 1 = always, 5 = never

competency with computers and security habits, Figure 4.1 summarizes their responses. Users rated themselves somewhat competent with computers (Q05), choosing a score of either 2 or 3. When asked about their technical competence (Q06) with security most users chose a score of 3. For both questions, 1 indicates more expertise.

We asked how often users thought about security and memorability when choosing their passwords; responses are available in Figure 4.2. For security (Q10), the average score across users was 2.58 (SD = 1.04). For memorability (Q11), 10 users chose 1 (always). We asked how often they use a password manager (Q12), and using the same scale, most users indicated never (7 users).

4.6 Results

We analyzed the data gathered from our study to determine users' satisfaction with the software. We also evaluated users' ability to successfully accomplish the tasks, the software's usability based on user feedback, and the amount of time required by users to accomplish tasks. In addition, we looked at whether users' mental models of how the software worked affected their ability to use the software successfully. Finally, we collected suggestions from users about how the software could be improved for a future prototype.

Partway through the study, logins during Task 2 started exhibiting a bug. The system would unlock the key, but the server would not return the correct encrypted session data, denying access to the system. When this occurred, the experimenter manually got the user past this point and to the successful login screen. Despite debugging, the problem could not be traced.

To accommodate this in the timing data, we eliminated usage data between when the user successfully unlocked their key until we received a valid session from the server (which should happen almost instantly in a functioning system). In our results, we report both sets of data: the actual unmodified value and the adjusted login times. Note that this only occurred occasionally for Task 2 and nowhere else.

4.6.1 Usage and Outcomes

For each task, we noted the outcome according to the criteria described in section 4.4. We analyzed the notes taken during the study to see if there were any common elements for the types of errors seen.

Of the 12 users, we found 6 users made a critical error on Task 5 when performing the requested actions. There was one additional user that needed a prompt to complete the linking process on the second laptop (Task 5) in order to continue. None of the users required significant help to use the system. Those that required prompting only needed guidance once or twice, and those instances are noted.

During the study some users required help in order to proceed. Two users asked some basic questions such as "What can my password be?" and "Can I use the same password on both laptops?". In such cases, the experimenter answered the questions as simply as possible so the user could continue. In two other cases, users came across minor bugs that required the experimenter to intervene to bypass the issue and get the user back on track. Finally, one user needed to be prompted to complete the linking process on the second laptop in order to proceed (Task 5).

Even though all users completed all the tasks, there were 6 instances where users

Task	Item	Completed	with Help	with Errors	with Critical Errors	Failed
1	Gen. PGP Auth Acct. 1	10	2	0	0	0
2	Create Website Account 1	12	0	0	0	0
3	Create Website Account 2	12	0	0	0	0
4	Gen. PGP Auth Acct. 2	11	1	0	0	0
5	Link PGP Auth Accounts	5	1	0	6	0
6	Login Website Account 1	12	0	0	0	0
7	Login Website Account 2	12	0	0	0	0

Table 4.1: Summary of outcomes for each task for 12 participants.

completed a task with critical errors. Table 4.1 shows a summary of each task and the outcomes as described in section 4.4. In five of the critical errors cases, the user selected both keys presented during the signing process instead of the specific one for the other laptop. Four of those users made the same critical error on both laptops. The remaining user signed both keys anyways, but did so one key at a time. The sixth critical error occurred when a user sequentially signed both keys presented on Laptop 1, but completed the process on Laptop 2 without any errors.

The number of participants that signed both keys rather than one in Task 5 was higher than expected. We explore why this happened in the interview and survey portions of the study to gain insight into how to modify the interface.

4.6.2 Timing

We collected the timing information from each laptop for each participant and then analyzed the data. We calculated the amount of time it took each participant to complete the tasks as well as several sub components to each task. The results of these calculations can be seen in Table 4.2.

We found that while it took users on average the same amount of time to generate

Task	**Item**	**Mean**	**Median**	**Std. Dev.**
1	Gen. PGP Auth Acct. 1	11.29s	11.93s	3.41s
2	a. Unlock Key Laptop 1	13.52s	7.37s	15.38s
	b. Create Website Account 1 (unmodified)	29.13s	22.87s	16.20s
	c. Create Website Account 1 (adjusted)	20.53s	15.97s	18.05s
3	Create Website Account 2	4.41s	3.92s	2.17s
4	Gen. PGP Auth Acct. 2	10.65s	9.38s	4.86s
5	a. Sign Key Laptop 1	45.55s	37.26s	21.75s
	b. Unlock Key Laptop 2	8.74s	6.83s	4.54s
	c. Sign Key Laptop 2	21.72s	9.36s	40.47s
	d. Total Signing Process	69.55s	67.12s	21.46s
6	Login Website Account 1	1.94s	1.47s	1.23s
7	Login Website Account 2	0.66s	0.61s	0.20s

Table 4.2: Time taken to complete the given tasks.

a PGP Auth account on both laptops (Task 1 and 4), there appears to be a learning curve for users when it comes to other features. For example, users took twice as long to perform the *Sign Key Laptop 1* (Task 5a) process (M = 45.55s), than performing the same steps on Laptop 2 (Task 5c) (M = 21.72s).

When creating an account from Laptop 1 on Website 1 (Task 2), users were required to unlock their password first before the account would be created and access granted. The entire account creation process took users an average of 20.53s to complete, considering the adjusted value due to the bug described in section 4.6. When users created their account on Website 2 (Task 3), it took them an average of 4.41s. This is the expected time a user would take to create an account in real world usage, once they had setup PGP Auth and unlocked their key for the browser session.

The login process from Laptop 2 (Task 6 and 7) was done after the linking process was completed (Task 5). This left the private key unlocked so the user was not required to enter a password to access either the first or second test websites. These login times reflect real world usage of accessing an account, once PGP Auth is setup and the user entered in their passphrase once for the browser session. Login times were quick, averaging 1.94s on Website 1 (Task 6) and 0.66s on Website 2 (Task 7).

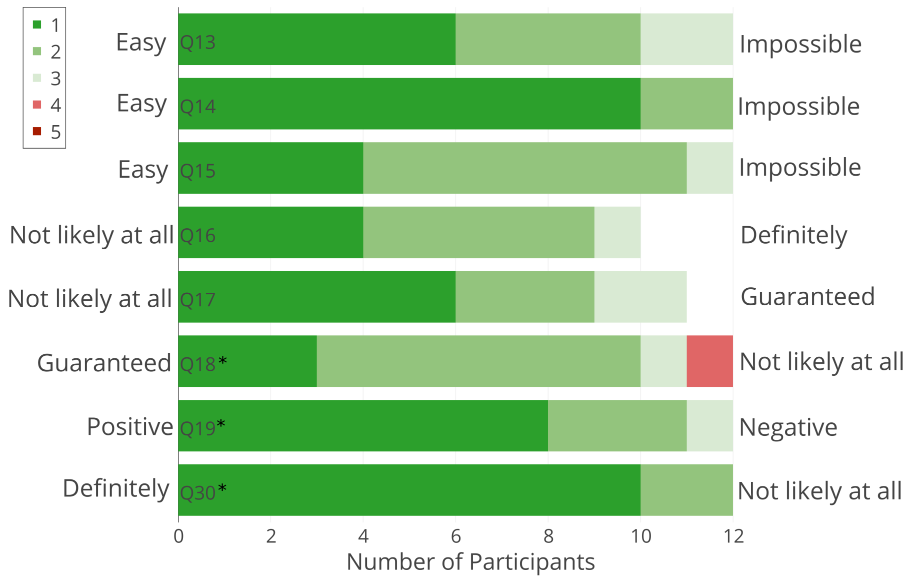


Figure 4.3: Distribution of user responses to Likert questions. Questions marked with an * indicate the Likert values were inverted for reporting purposes. 1 = most positive, 5 = most negative.

4.6.3 Survey Results

After performing the required tasks, users filled out a post-test questionnaire. We first present the results of users' perceptions of the software. We then describe and present the evaluation of the SUS questions. The average Likert scores for the perception and SUS questions are shown in Table 4.3. Two participants did not respond to question 16 and one did not respond to question 17. The questionnaire can be found in Appendix B. In all cases, 1 is most positive and 5 is most negative. Responses to negatively worded questions are inverted for reporting.

Figure 4.3 shows the distribution of responses to the perception questions. Users reported that using the PGP Auth system was quite easy; they almost universally found that creating a website account while using PGP Auth was easy (Q14), with 10 users rating it a 1. Even with the difficulties encountered with the linking process, many users still found this process relatively easy (Q15), with 7 users rating it a 2.

Users also felt that the system was secure. When asked how likely an attacker could gain access to their accounts (Q16), an average score of 1.7 was given. They also scored the likelihood that their password was shared with a remote website (Q17)

Category	**#**	**Question**	**Mean**	**Std. Dev.**
Competency	5*	Regarding computers	2.50	0.50
	6*	Regarding security	3.42	0.86
Security	10	Think about security	2.58	1.04
	11	Think about memorability	1.17	0.37
	12	Use a password manager	4.33	0.94
	16	Attacker gain access	1.70	0.64
	17	Password shared	1.64	0.77
Usability	13	Easy to setup	1.67	0.75
	14	Easy to create	1.17	0.37
	15	Easy to link	1.75	0.60
	18*	Likely to use	2.00	0.82
	19*	Overall experience	1.42	0.64
	30*	Choose stronger password	1.17	0.37
System Usability Scale	20	Use frequently	1.92	0.64
	21*	Unnecessarily complex	1.83	1.14
	22	Easy to use	1.58	0.64
	23*	Need support	1.83	0.90
	24	Well integrated	1.75	0.83
	25*	Too much inconsistency	1.25	0.43
	26	People would learn quickly	1.75	0.83
	27*	Cumbersome to use	1.50	0.65
	28	Confident using	1.67	0.75
	29*	Learn a lot to use	1.75	0.83

Table 4.3: Results from 5 point Likert questions. 1 = most positive, 5 = most negative. Items marked with an * indicate the Likert scales were inverted for reporting purposes.

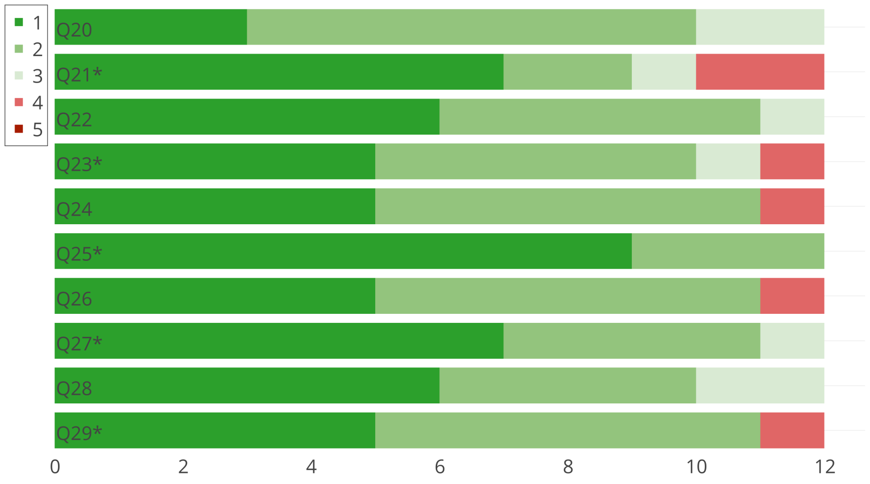


Figure 4.4: Distribution of user responses to SUS questions. Questions marked with an * indicate the Likert values were inverted for reporting purposes.

a value of 1.64.

When asked about their overall satisfaction with the software, users also reported favorable results. Users scored the likelihood of using the software (Q18) an average of 2. Users rated their overall experience (Q19) with the system an average of 1.42.

The users were also asked several Likert-scale questions to obtain the SUS ranking (see Figure 4.4). The SUS ranking required computing a score based on answers to questions 20-29. We calculated the SUS score using the method described by Brooke [11]. The score was subtracted from 5 for even numbered questions (positively worded questions) and 1 was subtracted from questions with odd numbers (negatively worded questions). The computed values were tabulated and multiplied by 4, yielding a number between 1 and 100. The ranking value is not a percentage, but can be used to analyze the usability of an interface relative to other interfaces [11].

When compiled, PGP Auth received an average rating of 82.82 (SD = 14.06). Sauro [54] determined that a SUS score above 80.3 yields an equivalent letter grade of an A, meaning that the software is among the top 10% of software. Based on PGP Auth' score, users perceived the software very usable and would be very likely to use it in the future.

4.6.4 Interviews

During the interview process, we assessed users' answers to see their level of understanding of the system. We compared their responses about how PGP Auth worked with the actual workings of the system. We then compiled the feedback about what worked well and what they felt could have been improved. Their responses were classified based on categories that emerged organically from the interview sessions. We used these results to come up with modifications that could be made to the prototype to improve on it. Users' responses to interview questions were also used to aid in understand the mental model diagrams produced.

We found several misconceptions held by users. Two users believed that the two laptops communicated with each other in order to allow access to the websites. One user indicated that Website 1 communicated to Website 2 to determine if the user had access to Website 1. Two users mistakenly believed that the user's passwords were sent to the remote websites. Many users expressed a general confusion about the relationship between keys and passwords.

As reported in the survey section, users were generally pleased with the software. When asked what they liked most about the system six users liked the idea of needing to remember only one password. Four users thought the system was quick and easy to use. Finally, two users liked the overall security of the system.

When we asked what users disliked about the system, we got a variety of answers addressing both usability and security. The only common answer was to have the UI look more polished, or have a "friendlier" interface. Some users gave specific items to work on, such as using Yes/No buttons instead of Cancel/OK when asking to create an account, including clearer messaging when the linking process is complete, and avoiding complicated and technical terms for security. One user was concerned about having their phone linked to their account: *"It would be easier to access your account if your devices was stolen and [someone] knew your password. So, for that reason I might choose to have my phone unlinked."* (User14). Similar sentiments were raised by another user about leaving a computer unlocked.

It was clear that many users were confused about how the linking process worked.

When asked, we got several confused responses: *"I didn't understand from the beginning"* (User04), *"I still don't understand"* (User08), and *"I guess I don't really understand how it links the two."* (User10). When we asked what they found confusing about the software directly, User13 indicated that simpler terms were needed and that diagrams to explain things might help.

In preparation for a second prototype, we asked users for suggestions on what could be done to improve the software. The main suggestions were directed towards the linking process: make it similar to BlueTooth linking, create a wizard (step-by-step process) for linking, and have a one-step linking process instead of having to link from both devices. One user wanted an option to disable it after having had a previous bad experience with another password manager extension.

Of these suggestions, the BlueTooth and step-by-step process options are similar. This could be achieved by adding a better guided process with clearer instructions at each step. It would, however, require removing the ability to sign multiple keys at the same time. A one-step process is not possible since keys need to be signed on both devices in order to create the appropriate relationship between the devices. By default, all extensions in chrome can be disabled by the user; therefore disabling PGP Auth is already possible through of the browser interface.

For those users with critical errors, we explained the error and then asked what the software could do to help prevent the issue in the future. One user again mentioned making the linking process more like the BlueTooth process where you obtain a value from one device and verify it on the second. Two users indicated that during the linking process, it would help to label the displayed keys. UI improvements were suggested by one user to highlight the important items. One user blamed themselves for the issues because they went through the UI too quickly without reading anything.

4.6.5 Users' Mental Models

Users were asked to explain their understanding of how PGP Auth authenticated to websites and how it linked two devices. After answering these questions, the user was asked to draw a diagram of how the system works on a sheet of paper. The sheet had printed blocks for Laptop 1, Laptop 2, Website 1, and Website 2 (see Appendix D).

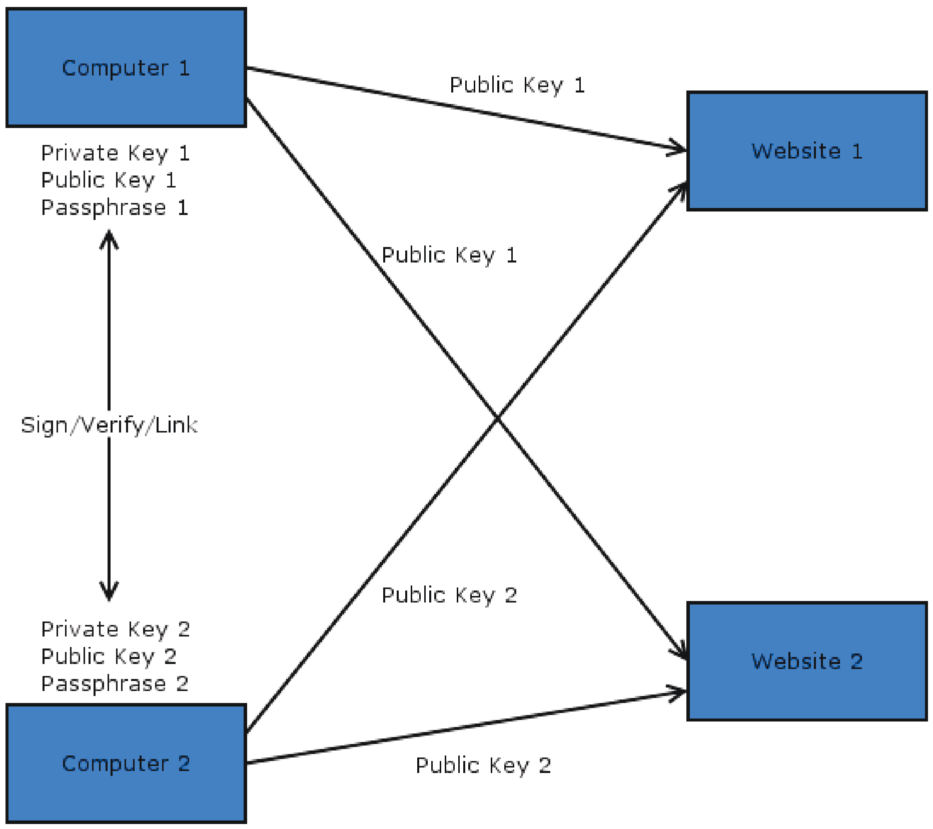


Figure 4.5: The correct, simplified mental model diagram used for comparison. Items in blue were provided on the user's sheet.

Some users had keen insights into how PGP Auth worked; one user responded *"Like one door to enter all the other rooms."* (User07), when describing the password used to lock the private key. Many users found it difficult to draw their understanding of how PGP Auth works; *"I never knew that this would be so hard."* (User02), and *"Essentially there is going to be arrows everywhere"* (User12).

To assess users' mental models, we generated our own simplified version of the diagram (see Figure 4.5). Expecting users to understand the interaction between the extension and websites and a key server seemed unreasonable, so we did not include that in diagram. We added 11 components to our diagram from the original (see Appendix D). A scoring scheme was devised: 1 point added for each correct item, 0.5 points subtracted for incorrect items, and 1 point added for mentioning interaction with a key server. Scores below 3.5 points were considered to have a low accuracy (see Figure 4.6 for an example), scores between 3 and 7 were considered medium accuracy,

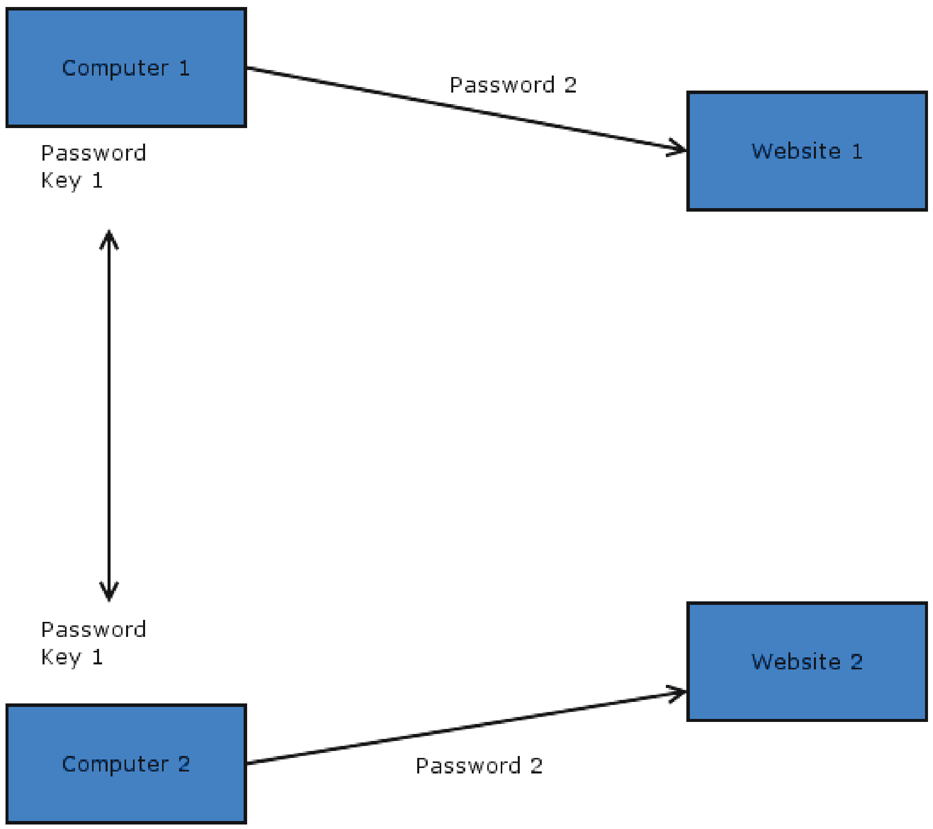


Figure 4.6: An example of a low scoring mental model, receiving a score of 1.5.

and scores 7 and above were considered to have a high degree of accuracy (see Figure 4.7 for an example). Users were encouraged to think aloud while they were drawing their diagram, any additional items that they mentioned during this process, but did not write down, were also taken into account and added or subtracted from scores as if written down. Verbal descriptions were used in place of drawn pictures in cases of conflicts. Table 4.4 has the results of this analysis.

We also informally explored whether users who made critical errors during task

Accuracy	**Total**	**Critical Errors**
High	4	1
Medium	5	4
Low	3	1

Table 4.4: Distribution of users' mental model accuracy scores. The number of users that had critical errors is also provided.

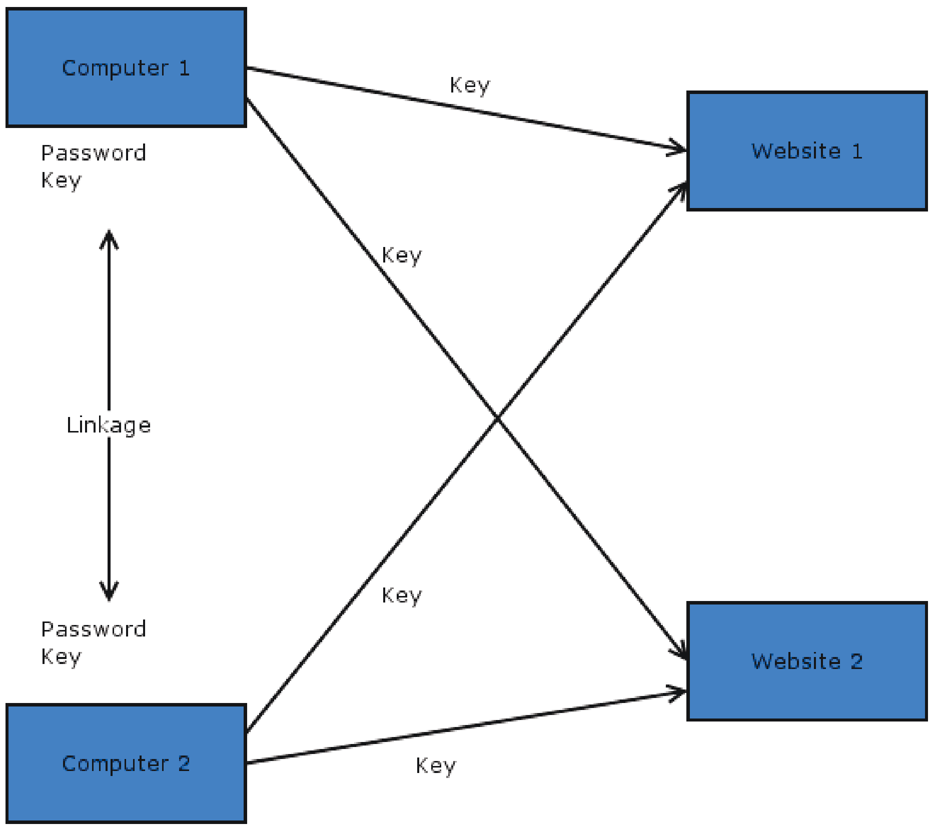


Figure 4.7: An example of a high scoring mental model, receiving a score of 9.

completion had lower understanding of how PGP Auth Works. Table 4.4 summarized the number of users making critical errors with each scoring category. Instead, we see that most critical errors were made by users with medium accuracy scores.

4.7 Summary

Users found the PGP Auth system easy to use and would likely use it in the future. More users committed a critical error during the signing process than anticipated. After analyzing the performance data and users' comments, we saw improved performance after first use, with the second key signing being much faster than the first. A guided approach to signing keys would likely solve many of the critical errors observed.

Chapter 5

PGP Auth: Prototype 2

While conducting our initial user study we found that a significant number of participants were making the same critical error. Users were selecting all of the keys from the list provided and proceeding to sign those keys. This would leave them open to an attacker gaining access to their accounts.

Users understood the mistake made when they were informed that they had signed an incorrect key. Many users, when prompted for possible solutions, offered up a similar idea. Instead of letting users select all of the keys they wanted to sign at the same time, the interface should guide them through the signing process for one key at a time.

Some users also were unsure of when the signing process was complete. They either asked if they were done, or paused as they tried to determine the system's status. Without a clear indication of completion, users may continue to sign additional keys.

Given the obvious user interface problems, we decided to stop the user study after 12 participants and modify the interface to better aid the user in the signing process. We completed a second user study to determine if the changes to the interface aided in the proper use of the system.

5.1 Wizard Interfaces

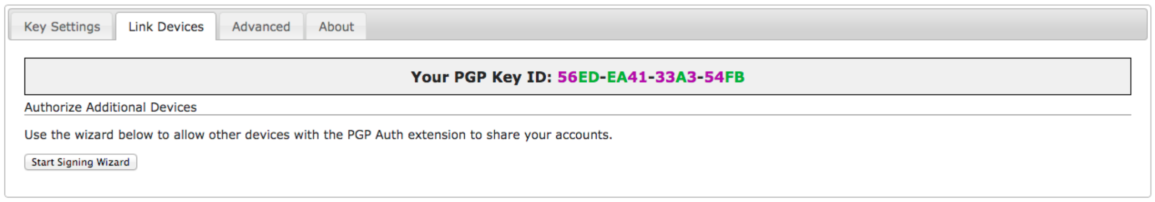
A wizard interface is designed to guide a user through a set of steps to achieve a goal. These are used when there are a large number of steps that need to be followed, or the task could be complicated for users to accomplish [59]. Users are provided short simple tasks one at a time that when completed together in the prescribed order achieve their goal. This allows users to accomplish tasks that would otherwise require them to have a much deeper understanding of the software.

We chose to a wizard interface as a replacement to our original linking process.

We felt that providing users with only one action at a time and providing instructions at every step of the process would reduce the number of critical errors.

5.2 Modification: Linking Devices Together

The entire *Link Devices* tab was modified to provide a different work-flow for signing keys. The new process consists of four steps which need to be completed on both devices. A wizard-like interface style was chosen to help guide users into selecting the correct key.

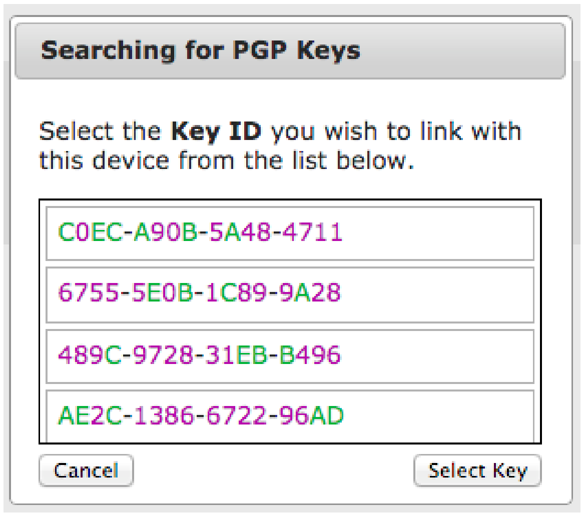


Your PGP Key ID: 56ED-EA41-33A3-54FB

Authorize Additional Devices

Use the wizard below to allow other devices with the PGP Auth extension to share your accounts.

Figure 5.1: The first step of the modified linking devices process.



Searching for PGP Keys

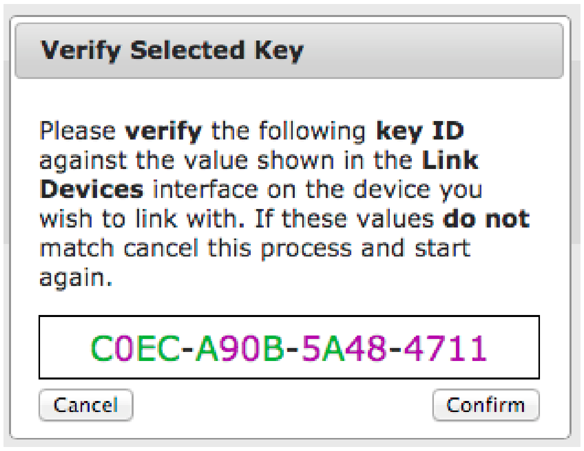
Select the **Key ID** you wish to link with this device from the list below.

C0EC-A90B-5A48-4711
6755-5E0B-1C89-9A28
489C-9728-31EB-B496
AE2C-1386-6722-96AD

Figure 5.2: The second step of the modified linking devices process: The user selects which key ID to sign.

- **Step 1:** Access the *Link Devices* tab

  The updated content is shown in Figure 5.1. The help instructions were removed and the user is informed to start the signing process by clicking on the *Start Signing Wizard* button. The local Key ID remains visible at the top of the tab.

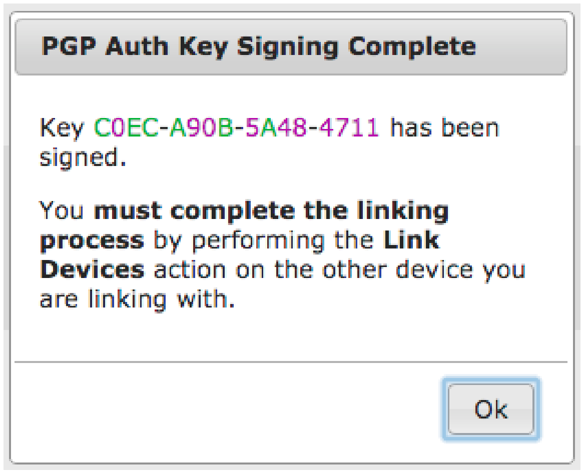


Verify Selected Key

Please **verify** the following **key ID** against the value shown in the **Link Devices** interface on the device you wish to link with. If these values **do not** match cancel this process and start again.

C0EC-A90B-5A48-4711

Figure 5.3: The third step of the modified linking devices process: A visual confirmation is made of the selected key ID to sign.



PGP Auth Key Signing Complete

Key C0EC-A90B-5A48-4711 has been signed.

You **must complete the linking process** by performing the **Link Devices** action on the other device you are linking with.

Figure 5.4: The fourth step of the modified linking devices process: Once signing has completed on this device, the user is informed that the process needs to be completed on the other device.

- **Step 2:** Select the key to link with

  A list of all keys that have not been signed are presented to the user in an overlay (see Figure 5.2). The user selects the key they wish to sign and clicks the *Select Key* button to proceed. Only one key can be chosen at a time.

- **Step 3:** Verify the selected key

  The selected key ID is shown to the user with instructions to verify this key against the key on the other device (see Figure 5.3). If the keys do not match, the user can cancel the process. Otherwise, the user confirms and signs the key using the *Confirm* button.

- **Step 4:** Signing complete
  The user is explicitly informed that the key has been signed. If the user still needs to complete the signing process from the second device, the confirmation text indicates this, as shown in Figure 5.4. Alternatively if both devices are now linked together, the user is informed that the linking process is complete.

5.3 Summary

These changes were made to aid the user navigate the two main issues seen in the first round of participants to our user study: signing incorrect keys, and not knowing when the signing process was complete. This modified process helps users find the correct key to sign. The system also now provides more feedback about the system status and progress towards successful completion of the signing process on both devices.

Chapter 6

User Study for Prototype 2

6.1 Study Design

After the updates were made to the prototype, we conducted a second study. We used the same procedure in order to determine if the changes made affected the outcomes. The prototype still contained the same instrumentation.

6.2 Participants

We recruited a total of 16 participants for the user study of the second prototype. Participants were recruited on campus using a combination of posters and existing mailing lists. Our participants' ages ranged from 20 to 42 years with an average age of 26.2 years. We had more women (62%) participants than men and 69% of participants were students.

Seven of our participants indicated they had 5-10 online accounts requiring passwords. Another 4 indicated they had between 11 and 15 accounts. One user had more than 21 accounts. Participants reported spending on average of 8.25 hours a day on a computer, with a standard deviation of 3.17. In addition, 81% of users reported using a single password for all accounts. No users used a unique password for each account.

Participants were asked a number of questions pertaining to their self described competency with computers and security habits; responses are summarized in Figure 6.1. Users rated their technical competence with computers (Q05) an average score of 2.06 (Md = 2, SD = 0.75). Users rated their technical competence with security (Q06) an average score of 3.19 (Md = 3, SD = 0.63). In both cases, 1 indicates higher expertise.

We asked how often users thought about security and memorability when choosing

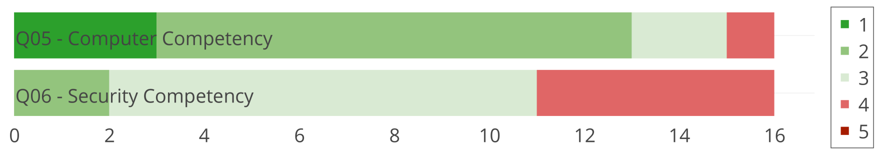


Figure 6.1: Distribution of user responses to competency questions. 1 = expert, 5 = no knowledge.

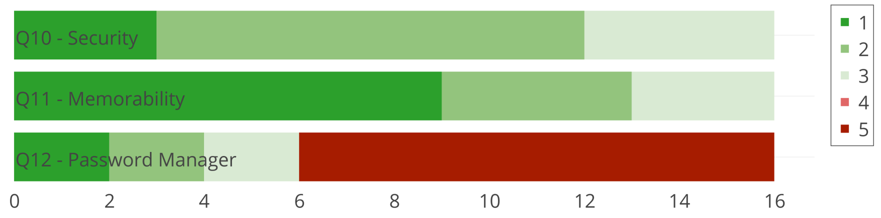


Figure 6.2: Distribution of user responses to password habits questions. 1 = always, 5 = never

their passwords. Figure 6.2 summarizes their responses. For security (Q10), the average score was 2.06 (Md = 2, SD = 0.66). For memorability (Q11), the average value was 1.63 (Md = 1, SD = 0.78). When asked how often they use a password manager (Q12), 10 indicated a value of 5 (never), with the average being 3.88 (Md = 5, SD = 1.54).

6.3 Results

We used the same evaluation strategies employed during the first user study to analyze the data gathered to look at the user's satisfaction with the software. This includes looking at the user's ability to accomplish the tasks, the user's perceptions of the software's usability, the amount of time to accomplish the set tasks, the user's mental models and how it affected their ability to use the software. We also collected more details from users about possible improvements to the software.

During Task 2, we saw the same persistent bug, requiring the experimenter to intervene. In addition, we had several users experience an unusually long delay in gaining access to the first testing website when accessing it from Laptop 2 (Task 6). This was not consistent and despite debugging, a source of the issue could not be found.

We accommodated this in the timing data as best we could. For Task 2, we eliminated usage data between when the user successfully unlocked their key until we received a valid session from the server (which should happen almost instantly in a functioning system). In our results, we report both sets of data: the actual unmodified value and the anticipated login times. We were unable to find a solution to the long delay in accessing Website 1 from Laptop 2.

6.3.1 Usage and Outcomes

For each task, we noted the outcome according to the criteria described in section 4.4. We analyzed the notes taken during the study to see if there were any common elements for the types of errors seen.

Of the 16 users, we found 5 users made a critical error on Task 5 when performing the requested actions. Four users needed a prompt to complete the linking process on the second laptop (Task 5) in order to continue. With the exception of one user, those that required prompting only needed guidance once or twice, and those instances are noted. One user required being walked through a portion of Task 5 in order to complete the study. Another 3 users asked some simple questions such as "What can my password be?" and "Can I use the same passwords on both laptops?". The experimenter answered these questions as simply as possible and allowed the user to continue.

During this study only 5 users completed the tasks with critical errors, and the remaining 11 users were successful. While this is an improvement over the first study, we were expecting better performance. A Wilcoxon RankSum test shows no statistically significant ($p < 0.05$) difference between the success rates in our two studies. We explored why this happened in the interview and survey portions of the study to gain additional insight into how to modify the interface.

Table 6.1 shows a summary of each task and the outcomes as described in section

Task	Item	Completed	with Help	with Errors	with Critical Errors	Failed
1	Gen. PGP Auth Acct. 1	16	0	0	0	0
2	Create Website Account 1	16	0	0	0	0
3	Create Website Account 2	16	0	0	0	0
4	Gen. PGP Auth Acct. 2	16	0	0	0	0
5	Link PGP Auth Accounts	8	3	0	5	0
6	Login Website Account 1	16	0	0	0	0
7	Login Website Account 2	16	0	0	0	0

Table 6.1: Summary of outcomes for each task for 16 participants.

4.4. In three of the critical errors cases, the user did not verify the key when performing Task 5 on Laptop 1. One of those users realized after they had committed the error and would likely have reverted this action, had they been able to. One user signed all the keys available on both devices sequentially during Task 5. Finally, one user incorrectly verified the keys during Task 5 and signed the incorrect one. This problem was due to two of the keys having the same first few characters.

6.3.2 Timing

As in the first study, we collected the timing information from each laptop for each participant and then analyzed the data. The results of these calculations are in Table 6.2.

It took users on average similar amounts of time to generate a PGP Auth account on both laptops (Task 1 and 4), 12.68s on Laptop 1 versus 9.97s on Laptop 2. A learning curve still was apparent for users when it comes to linking devices. For example, users took almost 50% longer to perform the linking process on Laptop 1 (Task 5a) (M = 45.93s), than performing the same steps on Laptop 2 (M = 32.54s), and that includes users needing to first unlock the key on Laptop 2.

Task	Item	Mean	Median	Std. Dev.
1	Gen. PGP Auth Acct. 1	12.68s	13.19s	5.43s
	a. Unlock Key Laptop 1	11.30s	8.55s	7.69s
2	b. Create Website Account 1 (unmodified)	77.09s	60.26s	27.41s
	c. Create Website Account 1 (adjusted)	10.93s	9.34s	7.57s
3	Create Website Account 2	3.87s	3.22s	1.72s
4	Gen. PGP Auth Acct. 2	9.97s	9.89s	3.75s
	a. Sign Key Laptop 1	45.93s	22.01s	65.83s
5	b. Unlock Key Laptop 2	13.37s	6.45s	18.26s
	c. Sign Key Laptop 2	32.54s	10.75s	50.81s
6	Login Website Account 1	23.42s	23.65s	19.63s
7	Login Website Account 2	0.78s	0.74s	0.11s

Table 6.2: Time taken to complete the given tasks.

Our result for users creating an account on Website 1 from Laptop 1 (Task 2) was 10.93s. This is based on the adjusted value after factoring out extra time to get past the bug. Creating a second account, once the private key was already unlocked, took a significantly shorter time, on average 3.87s (Task 3). These results are similar to that found during the first study.

When logging in from Laptop 2, the private key was already unlocked due to the completion of the linking process (Task 5). These login times reflect real world usage of accessing an account, after the user entered in their passphrase once for the browser session. Some users experienced an unusual delay during Task 6. This was traced to a delay in the server's response. As such, access times reported for Website 2 (Task 7) are the expected real world values, once they had setup PGP auth and unlocked their key for the browser session. We saw an average login time of 0.78s when accessing Website 2.

6.3.3 Survey Results

As in study 1, users completed a post-test questionnaire. We first present the results of users' perceptions of the software. We then describe and present the evaluation of the SUS questions. The average Likert scores given by users are shown in Table 6.3. One participant did not respond to questions 18, 19, 20, 24 and 25, therefore, their

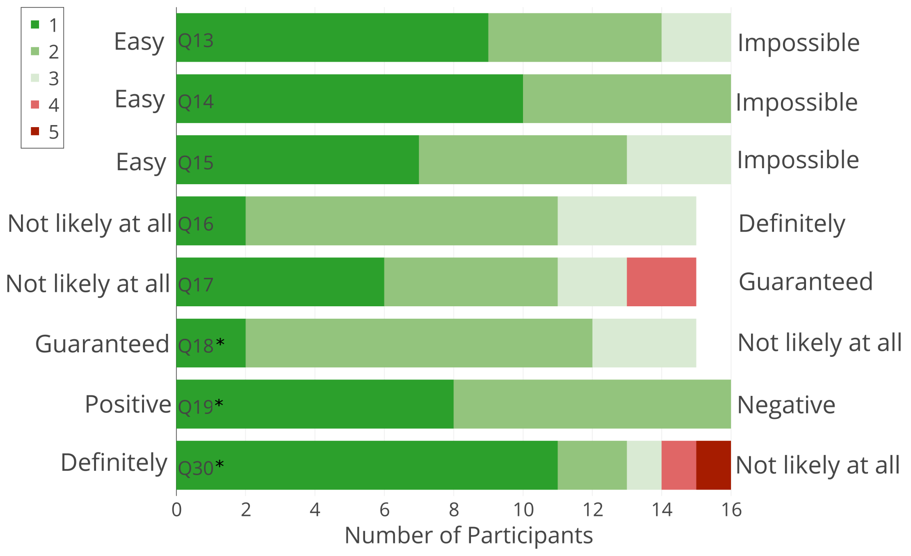


Figure 6.3: Distribution of user responses to Likert questions. Questions marked with an * indicate the Likert values were inverted for reporting purposes. 1 = most positive, 5 = most negative.

responses were not included in the SUS calculations. The questionnaire can be found in Appendix B. In all cases, 1 is most positive and 5 is most negative. Responses to negatively worded questions are inverted for reporting.

Figure 6.3 shows the distribution of responses to the perception questions. Users almost universally found that creating a website account while using PGP Auth was easy (Q14). Users found the new linking process relatively easy to use (Q15), with 7 users rating it a 1 and an additional 6 users rating it a 2.

When asked about their overall satisfaction with the software, users also reported favorable results. Ten users scored the likelihood of using the software (Q18) a 4. Rating their overall experience (Q19) with the system, users were split evenly between a score of 4 or 5 (5 = positive).

The users were also asked several Likert-scale questions to obtain the SUS ranking (see Figure 6.4). See section 4.6.3 for details on SUS scores.

We calculated the SUS score for the new interface using the same rules as mentioned in section 4.6.3. When compiled, the software received an average rating of 75.78 (SD = 14.49). Based on Sauro's work [54], getting a SUS score of 74% would

Category	**#**	**Question**	**Mean**	**Std. Dev.**
Competency	5*	Regarding computers	2.06	0.75
	6*	Regarding security	3.19	0.63
Security	10	Think about security	2.06	0.66
	11	Think about memorability	1.63	0.78
	12	Use a password manager	3.88	1.54
	16	Attacker gain access	2.13	0.62
	17	Password shared	2.00	1.03
Usability	13	Easy to setup	1.56	0.70
	14	Easy to create	1.38	0.48
	15	Easy to link	1.75	0.75
	18*	Likely to use	2.07	0.57
	19*	Overall experience	1.50	0.50
	30*	Choose stronger password	1.69	1.21
System Usability Scale	20	Use frequently	2.06	0.75
	21*	Unnecessarily complex	2.13	1.11
	22	Easy to use	1.63	0.48
	23*	Need support	2.19	1.24
	24	Well integrated	1.93	0.77
	25*	Too much inconsistency	1.93	1.00
	26	People would learn quickly	1.81	0.63
	27*	Cumbersome to use	1.81	0.95
	28	Confident using	2.00	0.71
	29*	Learn a lot to use	1.81	0.95

Table 6.3: Results from 5 point Likert questions. 1 = most positive, 5 = most negative. Items marked with an * indicate the Likert scales were inverted for reporting purposes.

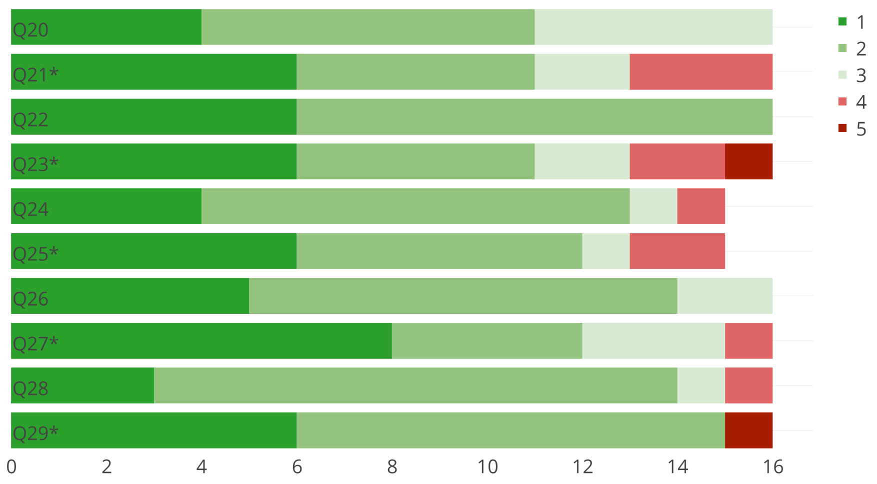


Figure 6.4: Distribution of user responses to SUS questions. Questions marked with an * indicate the Likert values were inverted for reporting purposes. 1 = most positive, 5 = most negative.

yield an equivalent letter grade of a B-, meaning that the software would be perceived to be more usable than 74% of other software. Interestingly, the SUS scores were lower than the perception scores. It is unclear why this happened.

We ran all of the perception and SUS results through Wilcoxon RankSum tests to determine if there were differences between the original prototype and the newer version. In all instances, we found no statistically significant differences.

6.3.4 Interviews

During the interview process, we once again compared users' understanding of how PGP Auth worked to the actual workings of the system. We then looked at the feedback given by users about what worked well and what they felt could have been improved.

As with the previous study, we found several misconceptions held by users. Three users indicated that Website 1 communicated directly with Website 2 to determine if the user had access to Website 1. One user mistakenly believed that the user's passwords were sent to the remote websites. Another user expressed confusion about the relationship between keys and passwords. Finally, one user believed that the passwords were shared across the two computers.

As reported in the survey section, users were generally pleased with the software. Eight users liked the idea of needing to remember only one password; one user commented: *"I find myself doing the forgot password thing more often than I should."* (User15). Another user said that having one password would let them choose a more secure password. Seven participants liked the simplicity of the software, with one user mentioning: *"I was thinking like the fingerprint stuff, or gestures .. Seem a lot more straight forward, without needing a body part."* (User23). Five users also liked the security of the software. One user liked the ability to link their devices together to share accounts. At the end of the session, one participant remarked: *"This was pretty cool."* (User24).

We asked users to share their dislikes with the software as well. Two users indicated that PGP Auth was complicated, *"Too complicated. If something happens in the middle and I fail, I'd call a tech guy."* (User 19). An additional 3 users found it hard to follow, with one commenting: *"I'm new to this setup, but I would say you would be given a code and I wasn't sure if I was supposed to memorize that code to log into the second computer."* (User18). One user noted that while having a single password would be good, it would also be a bad thing due to the possibility of an attacker gaining access to all of their accounts should they figure out the password.

The linking process was still the main point of confusion for users. One user admitted that they just did not understand how it all worked. Two others found the keys confusing and were not sure what keys were or how they were used. Another two users found the system easy enough for them, but expressed concerns that other "average" users might not understand the linking process. Finally, two other users found the whole linking process to be a foreign concept.

To aid in further enhancements to the interface, we asked users for suggestions as to what improvements could be made. Four users indicated that more and/or clearer instructions would be good. One user thought that a "friendlier" interface would help. We also had one user indicate that a diagram in the one-page information sheet would have helped a lot.

We feel that further written instructions in the interface may not alleviate the

Accuracy	Total	Critical Errors
High	2	1
Medium	10	2
Low	4	2

Table 6.4: Scores for users' mental models and the number of users that had critical errors.

errors that we were seeing. Instead, it may be better to provide an instructional video to walk the user through the process of setting up and using PGP Auth. A guided tour document might also achieve the same result. In both cases, the user could gain familiarity with the interface before using it.

We asked users with critical errors what could be adjusted in the interface to help prevent those errors in the future. We only had one participant offer a suggestion: adding labels to the keys presented in the interface. This may be possible using the comment field in the PGP Key, however, it would be easily spoofed by an attacker as well. So, this may not be a useful addition as it may offer a false sense of security and open a social engineering vulnerability in the system.

6.3.5 Users' Mental Models

Users were again asked to explain their understanding of how PGP Auth authenticated to websites and how it linked two devices. Users drew a diagram of how the system works on paper. The sheet had printed blocks for Laptop 1, Laptop 2, Website 1, and Website 2 (see Appendix D).

Even with the new interface, users still had difficulty drawing the diagram. One user commented: *"This is going to be a bit tough."* (User21). Others had difficulty understanding specific aspects of the system: *"The linking, I'm not sure I know what was going on"* (User27) and *"I'd hope it doesn't pass the encrypted password."* (User16).

We used the same simplified correct diagram to assess users' mental models (see Figure 4.5). We also used the same scoring mechanism as described in section 4.6.5. Table 6.4 has the results of this analysis.

After scoring users' diagrams, we found that 2 received a high accuracy score, 10 medium, and 4 low. There does not seem to be any relation between the accuracy of the mental model and making critical errors. Of the users that had critical errors, 1 had a high scoring mental model, 2 a medium, and 2 a low.

6.4 Summary

After making improvements to the PGP Auth interface, we found that the number of critical errors made by users did decrease, but the difference was not statistically significant. Although the small sample size may have affected the statistical results, it is likely that we will need to iterate on the design of the interface to further reduce the likelihood of critical errors.

Of those users that made critical errors, some recognized the problem and may have undone the step had the interface provided a method to do so. A further improvement to the interface could allow users to unlink devices and reset passwords to aid in error recovery.

Users enjoyed the system and liked the concept. There was a desire by many users to have only a single password, or fewer passwords than they currently have. Users also appreciated the security of the system and felt that it was more secure than traditional text based authentication systems.

While only a moderate understanding of how the system worked was found, this does not seem to affect most users' abilities to use the software. We found, between this and the previous study, that users with a higher level of understanding were not better able to avoid making critical errors.

Chapter 7

Discussion and Conclusion

In this thesis, we set out to develop an authentication system that would not be susceptible to most common types of attacks. Our system needed to protect against phishing, bulk guessing, brute-force guessing, password leaks, and shoulder surfing attacks. We looked at using Public-Key Encryption as an authentication method to accomplish these goals. As an enhancement to existing proposed methods, we included the ability to cross-sign keys as a method to share account access across devices to aid users that use multiple devices to access their accounts.

Historically, systems that used PGP have been difficult for users to understand and use. We wanted to determine the usability of our prototype, so we ran a user study. We found several users making the same critical errors so we modified the prototype to address these. A second user study showed some improvement but revealed other potential changes that might improve users' abilities to use the software.

In this chapter, we provide a summary of the results from our studies, and then present recommendations for future implementations. We follow with a section about future work to further extend the use of PGP as an authentication system.

7.1 Summary of Results

We wanted to replicate in our lab a set of steps that would closely mirror normal use of the authentication software in the real world. We devised a set of 7 tasks that included all of the functionality of the software. Users would interact with two separate websites that were running the PGP Auth server software and link two different laptops together to share the accounts.

In our first study, we found that users were making critical errors in signing keys. Many users would sign both the decoy key as well as the correct key in the signing process. It was apparent that the interface was not aiding in preventing this error.

A second prototype was developed to address some of the problems found. A second user study with additional participants was conducted with mixed results. Eleven of 16 users were successful.

We feel that additional user training and instruction would dramatically reduce the number of critical errors being committed. In our second study, 3 users with critical errors figured out how to correctly verify the keys by the end of the task. They either mistakenly selected the incorrect key or realized that they were supposed to verify the key after the fact. A second attempt may have yielded significantly fewer critical errors.

Users rated the software highly and indicated that they would be very likely to use the software. Users liked the idea of having a single password to access their accounts and appreciated the security of using PGP as an authentication system. Future implementations of the software should consider how to further simplify the linking process.

Unlike with other PGP software applications, users' understanding of how the system worked did not affect their ability to use the software. Even users with a medium to low understanding of the software were able to accomplish the tasks with relative ease. Users successfully linked accounts together without understanding the technical details of the infrastructure.

7.2 Implementation Recommendations

In order to have a successful implementation, the possible critical errors users can make must be minimized. We found that the linking process was the only point where users made critical errors. We suggest that the interface carefully guides users through this process.

7.2.1 More Guidance

According to Herzog *et al.* [30], a wizard interface can be used to guide a user through a complex set of tasks where users can easily make mistakes. The PGP Auth linking process seems well suited to this type of interface. In our second prototype, we implemented a wizard-like interface to walk users through the linking process. However, users still had trouble successfully completing the process. Our wizard did not include the ability to go back a single step, or to review the changes prior to final application. This, along with the non-traditional look of our wizard may have contributed to the poor performance of the interface.

7.2.2 Instructions

Although users recommended more instructions during the linking process, we found that users did not read many of the instructions already provided. This is similar to results that were found by Carroll *et al.* [14]. Novick *et al.* [47] also found that users did not generally read online manuals, so any help displayed to users should be minimal and part of the interface used to perform the current action.

7.2.3 PGP Key IDs are not User-Friendly

We found that some users had difficulty distinguishing between the 16-digit hex keys that were presented. Even though they were delineated and colour-coded, some users were randomly presented with similar looking keys. The process of validating keys is a relatively complex cognitive task. Users are likely to take shortcuts to simplify such tasks; for example, one user confirmed a key after looking at just the first few digits. To address this, alternative methods of validating the key would need to be investigated. Some possibilities could include QR codes, images generated based on the key ID, or changing the way the key ID is displayed. The effectiveness of these would have to be explored further.

Our initial design goal was to make PGP Auth as decentralized as possible and avoid relying on specific authorities. However, given users' difficulties with the signing process, it maybe worthwhile to explore whether a rendezvous system such as was implemented in Firefox Sync might be a useful. A user could connect to the intermediary rendezvous system with each device and instruct the rendezvous system to link the two devices.

7.2.4 Server-Side Considerations

Our server, while very basic, allowed for daisy chaining of cross-signing keys. This would allow for a device at position N in the daisy chain to have the same level of access to the server as the original device. This may not be the best implementation model, especially if a device becomes compromised. The server software could have much more flexibility in terms of granting access to different devices. A server could implement distance rules, whereby different levels of access are assigned based on the number links between this key and the original account. Another possibility could be to have user-defined levels of access; where a user could grant access to different devices.

A new system being built could easily implement PGP Auth as an authentication system. If a current system were to add PGP Auth, additional work would be required to transition. The system would need a way to map the key IDs to existing accounts and provide the additional HTTP headers. The amount of work should be minimal to get a functioning system; extra work would depend on the additional features that were desired.

7.2.5 Recommendation Summary

Based on the above discussion, the following is a list of recommendations to improve user success rates with the PGP Auth system:

- Add a summary step in the linking process for users to verify the linkage.
- Include a help icon that would display help documentation specific to the current action when clicked.
- Provide the user with additional visual information based on the key ID to aid in matching keys during the linking process.

7.3 Discussion

The users in Whitten *et al.*'s study [64] had difficulties with many aspects of the PGP system, including encrypting, publishing their public key, and signing keys.

Our interface was designed to minimize the user's interaction with the underlying PGP system. Our users still experienced problems when it came to signing keys, but the encryption and publishing of keys was handled for them automatically, therefore eliminating these sources of error.

In the UAF scheme [5], users are able to log into a remote site without having to provide their passwords. The system generates a new key-pair for each entity with which the user wishes to communicate. If a user wishes to access their accounts from a new device, a new key-pair is generated and the user must authenticate the device directly with the remote system. In PGP Auth, users can connect their devices together without involving the remote system. In addition, when a user links their devices, all of their web accounts are linked at the same time, instead of having to link each one separately.

In PGP Auth, users choose a password to use to lock their private key. Unlike in traditional text password systems, this password never leaves the device. Instead, we use the PGP encryption scheme as the authentication mechanism. A user's password strength no longer affects the ability of an attacker to access a user's web account, unless they have access to the user's physical device. Users' web accounts are now protected with strong encryption that is infeasible to crack. Once a user has unlocked their key, they no longer have to enter in credentials when accessing websites from this device. This makes the login process a seamless action that "just happens" when a user accesses a website, taking little time to complete.

In our implementation of PGP Auth, we used text passwords as the mechanism to unlock the private key. It would be possible to use other authentication mechanisms instead. For example, a gesture based mechanism could be used on a smartphone.

The PGP Auth system could be used in other situations as well. The limitation is that a client-server architecture is needed. For instance, no benefit would be gained by using it as a login method to a standalone desktop system. This system could be employed to authenticate a user to a mail server.

7.4 Limitations

Our user studies had small sample sizes (12 and 16 participants). With a larger sample size, some of the results may have been different. More participants would be required in each study to determine if there any statistical differences between the two interfaces. In addition, our participants were largely university students. Running the same study on a more diverse cross-section of the population may have different results.

We conducted our user studies in the lab during 30 minute sessions. During this time users were introduced to the software and asked to use it. In a real-world scenario, users may have more time and resources to learn about the software and use it. Users were also presented with prototype software which experienced some problems during the study; this may have affected their perception of the software. Users' opinions about the software were based on their experience in the lab, this may differ from their opinions about using the software in their day-to-day life.

7.5 Future Work

Our work forms the basis of an authentication system based on PGP. Many other aspects of PGP could be used to enhance the software. We used key signing to allow users to share accounts across devices, however, this can also be done to verify the server. Users could sign the server key to provide their approval of the site. When accessing the site, the client software looks for their signature on the server's public key to confirm that it is the same site.

PGP Auth could be extended on the server-side to support multiple user-level access controls based on keys. The initial key associated with a website could be granted full access to the website. Users could then selectively assign the type of access to subsequent devices based on their PGP Auth key. For instance, a user may want to grant their phone restricted access to their bank account so that it can only check account balances, but give their desktop full control. At setup, the server could assign a default level of access based on how many keys needed to be retrieved in order to find the original account. For example, a user has three devices A, B, and

C; A and B are linked together, and B and C are linked together. If the user creates an account on a website using device A and then later access the same website from device C, the system would need to load 3 keys to find the original account.

If the server supports granting different levels of access based on the key, it may be possible to extend this to allow users to grant access to keys belonging to other users. A user could grant access to a friend for a subset of actions. Users could set the permissions inside the application and the granting of access could be done just by signing a different user's key. For example, a user may sign a friends' key and set permissions to view photos. This type of access control would not necessarily require a reciprocal signing.

For privacy or other reasons, users may want to have multiple profiles that they use to access various websites to maintain distinct identities. PGP Auth could provide an interface on the client side to select which profile to use when logging into a website. This could be done on a per site basis or per session basis.

Longer term usability studies should also be performed. In our studies, we looked at participants' experience over a 30 minute period. Longer usage may affect users' perceptions. Users may also gain a better understanding of how the software worked and had a chance to develop a more complete mental model.

An ecologically valid real-world application study should also be done. This would require a website that users regularly use to implement the protocol and test extended usage over weeks or months.

7.6 Conclusion

In summary, we developed a method of authentication that used PGP. We iteratively designed PGP Auth, implemented it, and conducted user testing. We were able to successfully address our research questions:

- Can PGP be used as an authentication system?
  We successfully implemented an authentication system that used PGP encryption as the authentication mechanism. Our user studies showed that users generally liked the system and found the software usable.

- Can key-signing be used to enable devices to share accounts?
  PGP Auth provided a method to share accounts across devices by cross-signing keys. Users were able to perform the key-signing process and access their accounts from multiple devices.

- Are users successfully able to use this system?
  While users were successful at completing the tasks, some committed critical errors when it came to signing the keys. Additional work is needed to reduce chances of critical errors.

We offer recommendations for addressing the remaining usability issues, and believe that with a refined user interface, PGP Auth is a viable authentication mechanism that addresses many of the security vulnerabilities of traditional text password authentication.

Bibliography

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[2] Ben Adida. Beamauth: two-factor web authentication with a bookmark. In *Proceedings of the 14th Conference on Computer and Communications Security*, pages 48–57. ACM, 2007.

[3] Adam J Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th Conference on Offensive Technologies*, pages 1–7. USENIX Association, 2010.

[4] Mohamad Badra, Samer El-Sawda, and Ibrahim Hajjeh. Phishing attacks and solutions. In *Proceedings of the 3rd International Conference on Mobile Multimedia Communications*, page 42. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[5] Dirk Balfanz. UAF protocol specification. `https://fidoalliance.org/specs/fido-uaf-protocol-v1.0-rd-20140209.pdf`, 2014. Accessed August 11th, 2015.

[6] Abhilasha Bhargav-Spantzel, Anna Squicciarini, and Elisa Bertino. Privacy preserving multi-factor authentication with biometrics. In *Proceedings of the 2nd Workshop on Digital Identity Management*, pages 63–72. ACM, 2006.

[7] Robert Biddle, Sonia Chiasson, and P. C. van Oorschot. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys*, 44(4):19:1–19:41, 2012.

[8] blogs.rsa.com. Anatomy of an Attack. `https://blogs.rsa.com/anatomy-of-an-attack/?lang=en`, 2011. Accessed August 5th, 2015.

[9] bluetooth.org. Specification on the Bluetooth System. `https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439`, 2014. Accessed September 14th, 2015.

[10] Joseph Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Symposium on Security and Privacy (SP)*, pages 538–552. IEEE, 2012.

[11] John Brooke. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194):4–7, 1996. London.

[12] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. `http://www.rfc-editor.org/rfc/rfc4880`, 2007. Accessed August 5th, 2015.

[13] Xavier De Carné De Carnavalet and Mohammad Mannan. A large-scale evaluation of high-impact password strength meters. *ACM Transactions on Information and System Security (TISSEC)*, 18(1):1, 2015.

[14] John M. Carroll, Penny L. Smith-Kerker, James R. Ford, and Sandra A. Mazur-Rimetz. The minimal manual. *HumanComputer Interaction*, 3(2):123–153, 1987.

[15] Ivan Cherapau, Ildar Muslukhov, Nalin Asanka, and Konstantin Beznosov. On the impact of touch id on iphone passcodes. In *11th Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 257–276, Ottawa, July 2015. USENIX Association.

[16] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Transactions on Information Theory*, 22(6):644–654, 1976.

[17] Serge Egelman, Andreas Sotirakopoulos, Ildar Muslukhov, Konstantin Beznosov, and Cormac Herley. Does my password go up to eleven?: The impact of password meters on password selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2379–2388. ACM, 2013.

[18] S. Farrell, P. Hoffman, and M. Thomas. "HTTP Origin-Bound Authentication (HOBA)". `http://www.rfc-editor.org/rfc/rfc7486`, 2015. Accessed August 5th, 2015.

[19] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

[20] Dinei Florêncio and Cormac Herley. Where do security policies come from? In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, page 10. ACM, 2010.

[21] Dinei Florêncio, Cormac Herley, and Baris Coskun. Do strong web passwords accomplish anything? *HotSec*, 7:6, 2007.

[22] Gabber. Gabber: The GNOME Jabber Client. `http://gabber.sourceforge.net/`, 2015. Accessed August 20th, 2015.

[23] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2004.

[24] google.com. Google 2-Step Verification. `http://www.google.com/landing/2step/`, 2015. Accessed August 5th, 2015.

[25] David Gthberg. Public key encryption.svg. `https://commons.wikimedia.org/wiki/File:Public_key_encryption.svg`, 2006. Accessed August 20th, 2015.

[26] Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and System Security (TISSEC)*, 2(3):230–268, 1999.

[27] Eiji Hayashi and Jason Hong. A diary study of password usage in daily life. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2627–2630. ACM, 2011.

[28] Cormac Herley and Dinei Florencio. How to login from an internet café without worrying about keyloggers. In *Symposium on Usable Privacy and Security*, 2006.

[29] Cormac Herley, Paul C van Oorschot, and Andrew S Patrick. Passwords: If were so smart, why are we still using them? In *Financial Cryptography and Data Security*, pages 230–237. Springer, 2009.

[30] Almut Herzog and Nahid Shahmehri. User help techniques for usable security. In *Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology*, CHIMIT '07, New York, NY, USA, 2007. ACM.

[31] Anil K Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004.

[32] Ari Juels and Ronald L Rivest. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 SIGSAC Conference on Computer and Communications Security*, pages 145–160. ACM, 2013.

[33] keylogger.org. Keylogger.org. `http://www.keylogger.org/`, 2015. Accessed August 5th, 2015.

[34] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K Lenstra, Emmanuel Thomé, Joppe W Bos, Pierrick Gaudry, Alexander Kruppa, Peter L Montgomery, Dag Arne Osvik, et al. Factorization of a 768-bit rsa modulus. In *Advances in Cryptology–CRYPTO 2010*, pages 333–350. Springer, 2010.

[35] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2595–2604. ACM, 2011.

[36] Cynthia Kuo, Sasha Romanosky, and Lorrie Faith Cranor. Human selection of mnemonic phrase-based passwords. In *Proceedings of the 2nd Symposium on Usable Privacy and Security*, pages 67–78. ACM, 2006.

[37] Asheesh Laroia. Short key IDs are bad news (with OpenPGP and GNU Privacy Guard). `http://www.asheesh.org/note/debian/short-key-ids-are-bad-news.html`, 2011. Accessed August 5th, 2015.

[38] lastpass.com. How It Works — LastPass. `https://lastpass.com/how-it-works/`, 2015. Accessed August 11th, 2015.

[39] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. John Wiley & Sons, 2010.

[40] Claws Mail. Claws Mail - The user friendly, light-weight, and fast email client. `http://www.claws-mail.org/`, 2015. Access August 20th, 2015.

[41] Sana Maqsood. *Bend Passwords: Using Gestures to Authenticate on Flexible Devices*. PhD thesis, Carleton University Ottawa, 2014.

[42] Neil McAllister. LastPass got hacked: Change your master password NOW. `http://www.theregister.co.uk/2015/06/15/lastpass_data_breach/`, 2015. Accessed August 6th, 2015.

[43] Tom McCall. Gartner survey shows phishing attacks escalated in 2007; more than $3 billion lost to these attacks. `http://www.gartner.com/newsroom/id/565125`, 2007. Accessed August 22nd, 2015.

[44] Daniel McCarney, David Barrera, Jeremy Clark, Sonia Chiasson, and Paul C van Oorschot. Tapas: design, implementation, and usability evaluation of a password manager. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 89–98. ACM, 2012.

[45] Hunny Mehrotra, Mayank Vatsa, Richa Singh, and Banshidhar Majhi. Does iris change over time? `http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0078333`, 2013. Accessed August 6th, 2015.

[46] Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.

[47] David G. Novick and Karen Ward. Why don't people read the manual? In *Proceedings of the 24th Annual ACM International Conference on Design of Communication*, SIGDOC '06, pages 11–18, New York, NY, USA, 2006. ACM.

[48] Gregory L Orgill, Gordon W Romney, Michael G Bailey, and Paul M Orgill. The urgency for effective user privacy-education to counter social engineering attacks on secure computer systems. In *Proceedings of the 5th Conference on Information Technology Education*, pages 177–181. ACM, 2004.

[49] C Pettey and R Meulen. Gartner says number of phishing attacks on us consumers increased 40 percent in 2008. `http://www.gartner.com/newsroom/id/936913`, 2009. Accessed August 22nd, 2015.

[50] PGP.net. Security Questions. `http://www.pgp.net/pgpnet/pgp-faq/pgp-faq-security-questions.html`, 2015. Accessed August 5th, 2015.

[51] The Enigmail Project. Enigmail: A Simple Interface for OpenPGP Email Security. `https://www.enigmail.net/home/index.php`, 2015. Access August 20th, 2015.

[52] Frank Reiger. Chaos computer club breaks apple touchid. `http://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid`, 2013. Access August 6th, 2015.

[53] Hataichanok Saevanee, Nathan L Clarke, and Steven M Furnell. Multi-modal behavioural biometric authentication for mobile devices. In *Information Security and Privacy Research*, pages 465–474. Springer, 2012.

[54] Jeff Sauro. Measuring Usability with the System Usability Scale (SUS). `http://www.measuringusability.com/sus.php`, 2011. Accessed August 5th, 2015.

[55] Richard Shay, Saranga Komanduri, Adam L Durity, Phillip Seyoung Huh, Michelle L Mazurek, Sean M Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Can long passwords be secure and usable? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2927–2936. ACM, 2014.

[56] Steve Sheng, Levi Broderick, Colleen Alison Koranda, and Jeremy J Hyland. Why johnny still cant encrypt: evaluating the usability of email encryption software. In *Symposium On Usable Privacy and Security*, 2006.

[57] Elizabeth Stobert and Robert Biddle. A password manager that doesn't remember passwords. In *Proceedings of the 2014 workshop on New Security Paradigms Workshop*, pages 39–52. ACM, 2014.

[58] The Firefox Sync Team. Setting up Firefox Sync Just Got a Lot Easier. `https://blog.mozilla.org/services/2010/12/22/easy-setup-for-firefox-sync/`, 2010. Accessed September 14th, 2015.

[59] Jenifer Tidwell. *Designing interfaces.* " O'Reilly Media, Inc.", 2010.

[60] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. "i added '!' at the end to make it secure": Observing password creation in the lab. In *11th Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 123–140, Ottawa, July 2015. USENIX Association.

[61] Emanuel von Zezschwitz, Anton Koslow, Alexander De Luca, and Heinrich Hussmann. Making graphic-based authentication secure against smudge attacks. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, pages 277–286. ACM, 2013.

[62] G. Waller. Cybercrime Statistics Are Staggering, and Growing. `http://www.infosectoday.com/Articles/Cybercrime_Statistics.htm`, 2012. Accessed August 5th, 2015.

[63] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th Conference on Computer and Communications Security*, pages 162–175. ACM, 2010.

[64] Alma Whitten and J Doug Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Usenix Security*, volume 1999, 1999.

[65] John D Woodward. Biometrics: Privacy's foe or privacy's friend? *Proceedings of the IEEE*, 85(9):1480–1492, 1997.

[66] www.emc.com. RSA SecureID — Two-Factor Authentication — EMC. `http://www.emc.com/security/rsa-securid/index.htm`, 2015. Accessed August 5th, 2015.

[67] Rui Zhao and Chuan Yue. All your browser-saved passwords could belong to us: a security analysis and a cloud-based new design. In *Proceedings of the 3rd Conference on Data and Application Security and Privacy*, pages 333–340. ACM, 2013.

Appendix A

PGP Auth - Information Sheet

A.1 About PGP Auth

What if you only needed a single password for all your website accounts? What if websites never needed to know what that password was? What if you could access all your accounts from all of your devices without having to login separately to each one?

PGP Auth makes this possible!

PGP Auth uses PGP (Pretty Good Privacy) keys to act as your user account and verification system. While PGP can get pretty complicated the most important thing to know is that it allows you to identify and authorize yourself to a website without having to provide a username or a password.

With PGP Auth each of your devices has its own privately-held authentication key as well as a publicly-available authentication key. The private key is kept secret on your device and is never transmitted or seen by any remote website. When you attempt to login to a remote website, PGP Auth only provides your public key. The remote website will then validate the key against a publicly available version of this key. Using your public key, it will encode the information needed to complete the authentication process in such a way that only your privately held key can decode it. When your device successfully decodes the information you gain access to the website.

For added protection, your private key is locked using a password of your choosing. This password is only used to unlock your private key and the password never leaves your device. This prevents anybody else from knowing or even figuring out what your password is. Once unlocked, your private key will remain unlocked until the web browser (Internet Explorer, Chrome, Firefox, etc) is closed or you turn off your device. This means that an attacker would not only need to know your password, but also have a copy of your private key in order to gain access to your website accounts.

PGP Auth allows you to link all your devices together so that you can seamlessly access all your website accounts regardless of the device you are using. Linking two devices requires you to authorize each pair of devices against each other. For example, if you wanted your phone and desktop computer to share the same set of accounts, you would first authorize your phone from your computer, and then later, authorize your computer from your phone. While this authorization process may seem a little complicated, it ensures that you have control of both devices and prevents someone from impersonating you.

A.1.1 Key Benefits

Some of the key benefits of PGP Auth are listed below:

- Your password is never stored on remote websites, protecting you from password leaks.
- You can access all your website accounts using a single password for each device.
- Multiple devices (laptop, desktop, phone, tablet, etc) can all be linked together to access the same website accounts.
- Having a single password for each device allows you to use a more secure or complicated password.
- An attacker would need your password and have access to your private key, which is only stored on your device.

Appendix B

Survey Questions

B.1 Demographic Information

In this part of the questionnaire we collect some demographic information. You can always decline to answer should you feel uncomfortable with a question.

1. What is your gender? (Drop-down)
    - female
    - male
    - decline to answer
2. What is your age in years? (Text-field)
   you can decline to answer by leaving this field blank
3. What is the highest level of education you have completed?
    - no schooling completed
    - some high school
    - high school degree
    - some college
    - Bachelor's degree
    - Master's degree
    - Doctorate degree
    - trade or other technical school degree
    - other (specify below)
    - decline to answer

4. What is your current occupation? (Drop-down)
   - Administrative Support (e.g.,secretary, assistant)
   - Art, Writing, Journalism (e.g., author, reporter, sculptor)
   - Business, Management and Financial (e.g., manager, accountant, banker)
   - Education (e.g., teacher, professor)
   - Legal (e.g., lawyer, law clerk)
   - Medical (e.g., doctor, nurse, dentist)
   - Science, Engineering, and IT professional (e.g., researcher, programmer, IT consultant)
   - Service (e.g., retail clerk, server)
   - Skilled Labor (e.g., electrician, plumber, carpenter)
   - Student
   - If selected which program?
   - Other Professional
   - Unemployed
   - Retired
   - other (specify below) (Text-field)
   - decline to answer
5. How would you rate your technical competence with regards to computers? (5-Point Likert-Scale from 'No Knowledge' to 'Expert')
6. How would you rate your technical competence with regards to security? (5-Point Likert-Scale from 'No Knowledge' to 'Expert')
7. About how many accounts do you have that require passwords? (Drop-down)
   - fewer than 5
   - 5-10

- 11-15
- 16-20
- 21 or more

8. How many hours do you spend on a computer per day? (Text field)
9. How would you describe your password usage? (Drop-down)
   - One unique password for each account. Usually a unique password for each account.
   - A couple of passwords used for all my accounts.
   - One password used for all my accounts.
10. How often do you think about security when selecting your passwords?
    (5-Point Likert-Scale from 'Always' to 'Never')
11. How often do you think about memorability when selecting your passwords?
    (5-Point Likert-Scale from 'Always' to 'Never')
12. How often do you use password managers?
    (5-Point Likert-Scale from 'Always' to 'Never')

B.2 PGP Authentication

In this part of the questionnaire we are asking for your opinions based on your experience with the PGP Authentication system. You can always decline to answer a question if you feel uncomfortable with a question.

13. How easy was it to setup PGP Authentication?
    (5-Point Likert-Scale from Easy to Impossible)
14. How easy was it to create a new account on a website using PGP Authentication?
    (5-Point Likert-Scale from Easy to Impossible)

15. How easy was it to link two devices together using PGP Authentication?
    (5-Point Likert-Scale from Easy to Impossible)

16. How likely is it that an attacker could gain access to your accounts that use PGP Authentication?
    (5-Point Likert-Scale from Not likely at all to Definitely)

17. How likely is it that the password your used in PGP Authentication was shared with another system?
    (5-Point Likert-Scale from Not likely at all to Guaranteed)

18. How likely would you be to use PGP Authentication to log in to websites?
    (5-Point Likert-Scale from Not likely at all to Guaranteed)

19. Overall how was your experience with PGP Authentication?
    (5-Point Likert-Scale from Negative to Positive)

20. I think that I would like to use this system frequently.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

21. I found the system unnecessarily complex.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

22. I thought the system was easy to use.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

23. I think that I would need the support of a technical person to be able to use this system.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

24. I found the various functions in this system were well integrated.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

25. I thought there was too much inconsistency in this system.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

26. I would imagine that most people would learn to use this system very quickly.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

27. I found the system very cumbersome to use.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

28. I felt very confident using the system.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

29. I needed to learn a lot of things before I could get going with this system.
    (5-Point Likert-Scale from Strongly Agree to Strongly Disagree)

30. Would having a single password for all of your accounts make you more likely to choose a stronger password?
    (5-Point Likert-Scale from Not likely at all to Definitely)

Appendix C

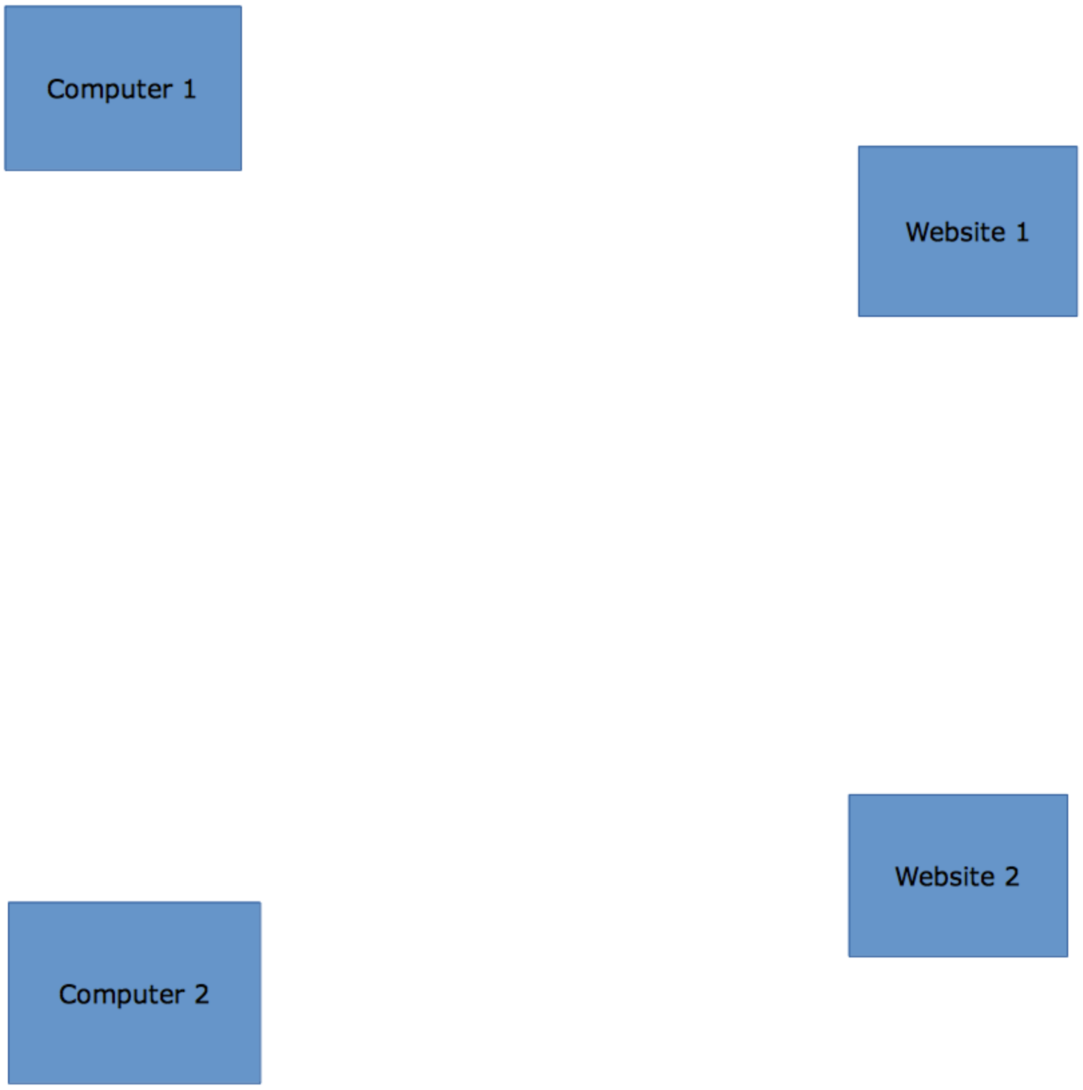
Interview Questions

- Describe how PGP Auth works for authenticating you on a web site.
- Describe how PGP Auth works to allow two devices to share the same account on a webs site?
- Can you draw a diagram representing how PGP Auth works and the way it interacts with the web browser and remote web site?
- What did you like about the system?
- What did you not like about the system?
- Any suggestions or improvements?
- Did you find anything confusing?
- (Optional) What could have been done with the interface to make the process of linking devices together clearer?

Appendix D

Interview Diagram

Q: Can you draw a diagram representing how PGP Authentication works and the way it interacts with the web browser and remote web site? Where are the keys and passwords stored and how are they used?


Computer 1
Website 1
Website 2
Computer 2