BluePass: A Secure Hand-Free Password Manager

Yue Li[1](✉), Haining Wang[2], and Kun Sun[3]

[1] College of William and Mary, Williamsburg, VA 23187, USA
yli@cs.wm.edu

[2] University of Delaware, Newark, DE 19716, USA
hnw@udel.edu

[3] George Mason University, Fairfax, VA 22030, USA
ksun3@gmu.edu

Abstract. With the growing number of online accounts a user possesses, managing passwords has been unprecedentedly challenging. Users are prone to sacrifice security for usability, leaving their accounts vulnerable to various attacks. While replacing text-based password with a new universally applicable authentication scheme still seems unlikely in the foreseeable future, password managers have emerged to help users managing their passwords. However, state-of-the-art cloud based password managers are vulnerable to data breach and a master password becomes a single point of failure. To address these security vulnerabilities, we propose BluePass, a password manager that stores the password vault (i.e., the set of all the encrypted site passwords of a user) locally in a mobile device and a decryption key to the vault in the user computer. BluePass partially inherits the security characteristics of 2-Factor authentication by requiring both a mobile device and a master password to retrieve and decrypt the site passwords. BluePass leverages short-range nature of Bluetooth to automatically retrieve site passwords and fill the login fields, providing a hand-free user experience. Thus, BluePass enhances both security and usability. We implement a BluePass prototype in Android and Google Chrome platforms and evaluate its efficacy in terms of security, usability, and overhead.

Keywords: Password manager · Two-factor authentication

1 Introduction

Text-based password still dominates online authentication despite that it has long been plagued by a well-known and long-standing problem: the wide use of weak password. Due to limited human memory, users tend to choose weak passwords [3,5]. However, weak passwords are easy to guess and thus are vulnerable to a variety of attacks [4,19,22,28,32]. Today's increasing number of accounts a user possesses even worsen the problem since the user poorly manage their passwords. For example, on average users may reuse one password for as many as 3.9 online accounts [11]. As such, instead of impractically expecting users to select a strong password for each account, password managers are developed as built-in or standalone gadgets to help users manage their credentials. A password manager includes a vault that stores all encrypted passwords of a user, and the user only needs to remember one master password, which is used to generate the decryption key to the vault, to access all the passwords in the vault. To support user authentication on different devices, password managers usually synchronize the vaults to their own servers and provide a downloading service to their users. However, a password manager has its own security and usability problem. For example, password managers usually synchronize the local vault to the remote server, which makes data breach possible [2]. Furthermore, to enhance usability, many browser built-in password managers do not necessarily need a master password, which makes it vulnerable to unauthorized use and meanwhile sacrifices portability. Even being used, a master password becomes a single point of failure. Usability issues of a password manager may even lead to reduced security, stemming from incomplete user mental models [7].

For critical online services, users may desire more secure authentication than merely password. Toward this end, two-factor authentication (2FA) is proposed to include another layer of protection to user accounts. Nowadays many leading service providers such as Google and Microsoft, have integrated 2FA into their online systems. However, 2FA suffers from limited adoption due to undesired extra burden on users. It is estimated that in 2015, only around 6.4% of Google users are using 2FA [24]. In order to improve usability, transparent 2FA has been proposed [8,23] by leveraging additional devices (mainly user smartphones) to automatically complete the enhanced authentication procedure without user involvement. However, these approaches are hard to deploy because of imperative modifications at both the web server and the client sides.

In this paper, we propose BluePass, an enhanced password manager that partially inherits the security benefit of 2FA to improve the security and usability of existing password managers. One of the key features of BluePass is to isolate the storage of the password vault from that of the decryption key. Here the password vault is the set of all the encrypted site passwords of a user. Specifically, the password vault is stored locally in a mobile device (e.g., a user's smartphone) and the decryption key is stored in the BluePass server, which can be accessed and downloaded only once to a computer after authentication through a master password. The mobile device communicates with the computer using Bluetooth in a transparent manner. When a user needs to log in a website, the computer will automatically request the site password from the mobile device. The encrypted site password will then be delivered through Bluetooth. Afterwards, the computer is able to decrypt the site password using the local decryption key and auto-fill the web forms for the user. BluePass relies on Bluetooth for communication rather than other channels, because Bluetooth can be both transparent to users and a subtle indicator of co-location of the user mobile device.

BluePass is secure since it does not store password vaults on a server and is not vulnerable to massive password breach. Furthermore, a server data breach is likely to leak both password vaults and hashed master passwords. By cracking the master password table offline, it is almost guaranteed that most master passwords can be craked out, given today's computing power and the weakness of user-selected passwords. Attackers are given direct access to password vaults under such a case, since the vault decryption key is generated from the master password. By contrast, in BluePass, the password vault and its decryption key are stored separately, and decryption key is not generated from master passwords, losing one of them will not practically leak any password.

While BluePass itself uses 2FA, it does not require any modifications on the website servers. Thus, the underlying password framework remains unaltered, i.e., logging into a website still only needs one site password. BluePass is also usable since it demands little effort to configure on the computer and no extra effort from a user to authenticate afterwards.

We implement a BluePass prototype in Android and Google Chrome and evaluate its efficacy in terms of security, overhead, and usability. First, we conduct a comprehensive security analysis to demonstrate that BluePass can defend against various attacks. Then we evaluate the auto-fill time latency of BluePass by recording the time between login forms being detected and the forms being automatically filled. We also run a series of experiments, in which we retrieve passwords under different frequencies, to measure the energy overhead of BluePass. Based on our experimental results, BluePass is energy efficient while automatically filling in the login forms with user-unperceived latency. Afterwards, we conduct a user study including 31 volunteers to examine the usability of BluePass. The results show the test subjects regard BluePass as both secure and usable. Moreover, the majority of testers report that they are willing to use BluePass to manage their passwords.

The remainder of the paper is organized as follows. Section 2 elaborates the system overview and threat model. Section 3 details the system architecture of BluePass and Sect. 4 conducts security analysis on BluePass. Section 5 illustrates the prototype implementation of BluePass. We evaluate BluePass in Sect. 6 and present a user study in Sect. 7. Section 8 discusses BluePass-related issues and its limitation. Section 9 surveys related work and finally, Sect. 10 concludes this paper.

2 System Overview and Threat Model

Before presenting the BluePass system, we first introduce important BluePass notations for clarification purposes.

- *BluePass server:* a server that is mainly responsible for registering users and distributing keys to user computers.
- *Key pair (K_1, K_2):* a pair of RSA keys that are used by the mobile device and the computer to encrypt/decrypt site passwords. K_1 is only stored in the mobile phone while K_2 is stored in the BluePass server for re-distribution. We manage to use only one pair of keys to protect bi-directional communication, and the details can be found in Sect. 8.1.

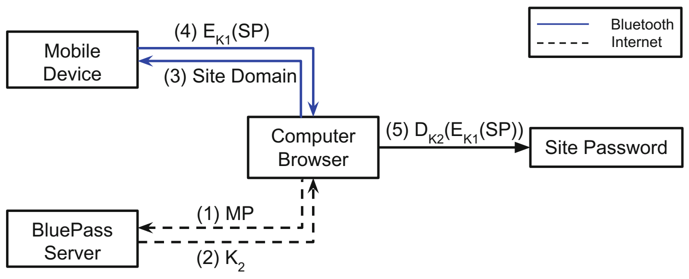


Fig. 1. BluePass authentication.

- *Master password (MP):* a user uses its master password to authenticate itself to the BluePass server and retrieve its own decryption key K_2. A master password is the only password a user needs to remember.
- *Site password (SP):* passwords to access online services, which will be encrypted by K_1 and then stored in the BluePass mobile application.
- *Trusted computer:* a computer that the user trusts, such as the user's personal computer. It stores the decryption key K_2 for a long term.
- *Untrusted computer:* a computer that the user does not trust, such as a library computer. The decryption key K_2 must be retrieved from the BluePass server every time a browser is opened in the untrusted computer. K_2 is only temporarily stored in a browser instance, and is removed when the browser instance is terminated.
- *Client-side (computer/browser) application:* the user installs it on the computer, which is in charge of detecting and auto-filling login forms, communicating with the mobile device, and decrypting the received site passwords.
- *Mobile application:* the user installs the app on its mobile device. The app stores the encrypted site passwords and delivers the encrypted site passwords to the user computer through Bluetooth.

2.1 System Overview

BluePass works on two premises. First, a site password can only be recovered by having both the encrypted site password $E_{K_1}(SP)$ that is only stored in the mobile device and the corresponding decryption key K_2 that is distributed through the BluePass server. Second, the encrypted site password $E_{K_1}(SP)$ can only be retrieved from the mobile device to the user computer through Bluetooth, which requires the proximity of the two devices. The flow chart of BluePass password authentication is shown in Fig. 1.

The working mechanism of BluePass mainly includes three phases, which are detailed as follows.

Phase 1: Registration is a once-in-a-lifecycle operation, in which a user needs to register for the BluePass service. The user installs the BluePass mobile app on its mobile device and uses its master password to log into the BluePass account. The mobile device is then initialized with an empty password vault.

Phase 2: Configuration is to install and configure the user devices. First, a client-side application needs to be installed on the user computer. Then, the user will log into the BluePass server and download the decryption key K_2 into the computer. The user will store the key either for a long term or temporarily, depending on whether the computer is trusted or untrusted. Note that the installation of the client-side application on a computer is also a one-time operation. The retrieval of K_2 from the BluePass server is needed each time opening a browser only when the user is on a untrusted computer.

Phase 3: Authentication is almost transparent to the user. In a trusted device, the user only needs to carry the registered mobile phone and wait for the passwords being automatically filled. In a untrusted device, the user needs to re-enter the master password every time a new browser instance is opened since the key K_2 is deleted when a browser instance is closed.

2.2 Threat Model

Attackers aim at stealing one or (preferably) all site passwords in the password vault. In the design of BluePass, all the site passwords of a user are encrypted and stored in the user's mobile device. We assume that the attacker cannot access the encrypted site passwords in the mobile device and knows the decryption key from the computer at the same time.

All attacks can be classified into two categories: *co-located attacks* and *remote attacks*. A co-located attack can only happen within the Bluetooth communication range of the user mobile device, while a remote attack can be launched from anywhere. In a co-located attack, since the attacker could access the encrypted site passwords through sniffing, we must prevent the decryption key from falling into the hand of the attacker. Therefore, both the BluePass server and the master password cannot be compromised. Moreover, the communications for key distribution must be protected. By contrast, in a remote attack, since the attacker cannot access the mobile device through Bluetooth, either the BluePass server or the master password could be compromised. Also, no secure communication is required for key distribution. As the Bluetooth reachability is very limited (33 feet for class 2 Bluetooth devices), a co-located attack is much more difficult to launch than a remote attack.

3 System Architecture

3.1 Core Functions

As mentioned in Sect. 2.1, BluePass mainly consists of three phases. The first two phases, registration and configuration of BluePass, are mostly one-time effort; however, the third phase, authentication, will be triggered each time a user needs to log in a website. Figure 2 illustrates BluePass architecture and the data flow of these three phases.

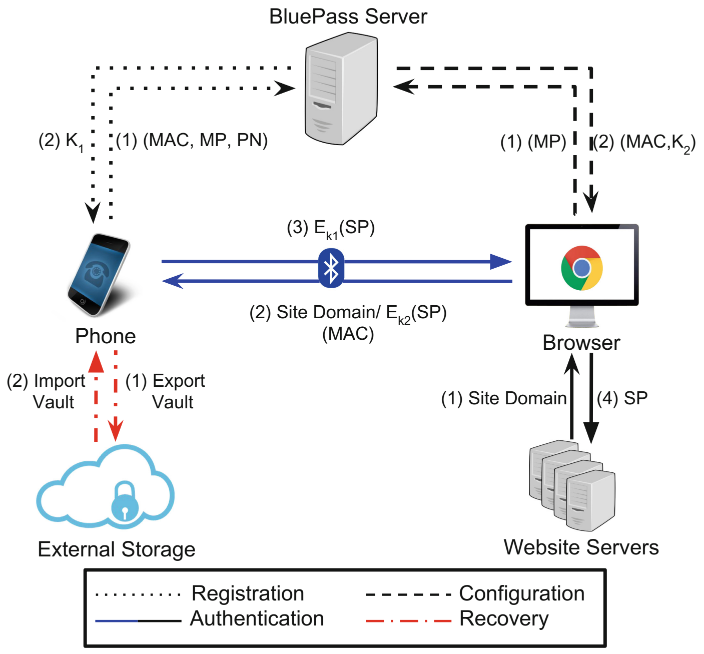


Fig. 2. BluePass architecture

Registration. The black dotted lines in Fig. 2 show the registration process. To register a BluePass service, the user only needs to download a BluePass application to the mobile phone and create a master account on the BluePass server. The creation of the master account is similar to the creation of an account in any website. Upon logging into the master account on the mobile app, the user can choose to bind the mobile device. The binding process should follow a traditional 2FA mechanism. Namely, the user re-authenticate herself with another authentication factor, for example, a sms. Afterward, the device information, specifically, the MAC address of the device Bluetooth, will be uploaded to the BluePass server. The MAC address is used for the client-side application to automatically locate the associated mobile device without user involvement. For a newly associated device, the BluePass Server generates a pair of asymmetric keys (K_1, K_2). It then distributes K_1 to the mobile phone and keeps only K_2 on the server side. We list the database of the BluePass server in Table 1 and that of the mobile device in Table 2 populated with made-up data. The registration should only be done once on the mobile device. After registration, the mobile device is initialized as a password vault. Note that the key pair of (K_1, K_2) is not used as a conventional public-key pair, where the public key is known to all and the private key is kept in secret. Instead, the key pair is used for a two-way communication channel and both of them should be kept in secret.

Table 1. Server side data

Username	Salt	$H(MP + Salt)$	K_2	Device MAC address
Alice	ifu92@fb	$a4f3b3c9e61b838f8cda07...$	$VDSnrzjqFBy9...$	BC:F5:AC:9D:9A:57
Bob	01dm.a<w	$daa4a403bfec911a3ef199...$	$yKhTC3dNAkE...$	BC:F5:AC:9D:9A:58

Table 2. Mobile device data

Domain	Username	K1	$E_{K_1}(Password)$
.yahoo.com/	aliceweb1	$AoGAKooOHMT...$	$Encrypted\_Password_1$
.yahoo.com/	aliceweb2	$VN9SdOeFbo4w...$	$Encrypted\_Password_2$
.google.com/	aliceweb2	$B1FUeDXiqv4j...$	$Encrypted\_Password_3$

After registration, the user has initialize a password vault in its own mobile device and associated the BluePass account with this device.

Configuration. The computer needs to be configured to run BluePass, which is shown in the dashed black lines in Fig. 2. The user installs and runs a client-side application, and then logs into the BluePass server to fetch the Bluetooth MAC address of the mobile device and K_2 generated during the registration. At this point, the user can choose whether the computer is trusted or not. If the computer is trusted, the Bluetooth MAC address and K_2 will be stored in the browser for a long term. Otherwise, they will be deleted after the user closes the current browser instance. Knowing the device Bluetooth MAC address enables the computer to pair with the device automatically by using RFCOMM insecure mode, in which the Bluetooth data is broadcasted and the target MAC address is specified in the data. BluePass does not rely on secure Bluetooth communication. Using RFCOMM insecure mode enhances usability while not degrading security.

Authentication. The authentication phase is the only phase that a user will constantly experience during use of Bluepass. The solid lines in Fig. 2 show the data flow of BluePass authentication process. First, the user directs the browser to a website it wants to login. The BluePass client-side application will examine the Document Object Model (DOM, which is a tree structure representing the webpages) of the returned page and check the existence of a login form. If a login form is present, the application requests the corresponding credentials from the mobile phone using Bluetooth. After receiving the request, the mobile application returns the encrypted credentials. If no related credential exists, BluePass will instead respond with a "NO_PASSWORD" flag. We realize that auto-filling in a non-HTTPS environment is vulnerable to JavaScript injection attacks [26], so we only do auto-filling for websites that are based on HTTPS. For other websites, BluePass will pop up a window for a user's consent before filling the login form. Note that none of the above steps require any user interactions. This fully automated authentication enables users to login a website in a hand-free manner. When there exists more than one account for a specific website, the browser will let the user choose an account to be decrypted and automatically filled in the forms since there is no way to predict which account will be used.

3.2 Account Management

Account management is essential to a password manager. Users should be able to add, edit, or delete the credentials in BluePass. These functions must be correctly designed to guarantee the security of BluePass.

The addition of an online account into BluePass can be done when a user has manually inputted the login credentials into a new website. BluePass adopts a similar approach just as current browser built-in password managers. If the "NO_PASSWORD" flag is sent back, the browser knows that no login credentials are associated to this particular website. If the user manually inputs the credentials, the browser will capture the value in the form before submission and prompt a non-intrusive dialog window, asking whether the user wishes to store the login information into BluePass. Specifically, there are three options: "yes", "not this time", and "never". If "yes" is chosen, the browser will encrypt the credentials using key K_2 and send it to the mobile device (see Fig. 2). The mobile device will decrypt the information using K_1 and encrypt it again using K_1. Mathematically the process is denoted as $E_{K_1}(D_{K_1}(C))$, in which $C = E_{K_2}(SP)$. Then the encrypted credentials are stored in the BluePass database.

The edition of an online account is similar to the addition process. The browser monitors if the user has modified the value in the login form when being submitted. If the password is changed, the browser will prompt a dialog that asks for user permission to update the login credentials in BluePass. Upon user consent, the browser will send the updated values in an encryption and decryption procedure similar to that of adding a new account. Note that the chosen option of "never" should also be recorded in the password vault, which prevents the dialog from prompting repeatedly. In this case, the password vault records the domain name and the username without storing a password. When an empty password is passed back, the application is acknowledged that the user does not wish to store the login credentials. The revocation of the "never" status can be done in the administration page in the mobile applications.

The deletion of login credentials can also be done on the mobile application's administration page. The mobile application shows a list of websites whose site passwords are stored in the mobile phone. The user can choose to delete one of the websites' login credentials. However, the user needs to manually input the website's URL and login credentials. Before the deletion is granted, the user must input the correct master password. This will prevent an attacker from manipulating the user's online accounts.

3.3 Recovery

When using a cloud-based password manager, users can backup their password vaults on the server side. On the contrary, BluePass is de-centralized and stores local copies on mobile devices. Though users usually do not lose their mobile devices quite often, it is essential for BluePass to back up and recover the password vault when the mobile devices are lost, which is illustrated with red lines in Fig. 2.

Users can choose to back up their vaults to an external storage including a portable hard disk, a USB, or a cloud storage. If a user loses the mobile device, it can recover the vault from the external storage. Backing up the vault to a user-owned physical device may require the user to periodically back up and synchronize the password vault to the external storage device. Alternatively, BluePass allows users to synchronize their password vaults to a cloud drive provider. Nowadays many large drive providers, such as Google Drive or Dropbox, have published APIs to facilitate data synchronization. Note such design still ensures the 2FA design of BluePass – an attacker needs to breach both the BluePass Server and the cloud provider server to collect the two necessary pieces of secret.

4 Security Analysis

BluePass is secure in a sense that as long as a user does not lose two factors at the same time, the user's login information is safe. We conduct a security analysis on BluePass to verify the robustness of BluePass against various attack vectors.

4.1 Two-Factor Security

We have introduced that BluePass relies on the premise that two factors need to be possessed to derive a site password. The two factors are user mobile device and a master password. Now we discuss the security of BluePass when one of the factors is compromised.

Master Password. An attacker may be able to compromise the master password of a user, which can be done through different ways such as guessing, phishing, shoulder-surfing, etc. The compromisation of a trusted computer is also equivalent to losing the master password because the only purpose of having the master password is to retrieve K_2 from BluePass server, which can be directly extract K_2 from a trusted computer. In such scenarios, the attacker is able to obtain key K_2. However, if the attacker does not have the password vault of the user, K_2 is merely a meaningless token and the security of BluePass holds. Besides, the user is able to change the master password and re-generate a new key pair.

Mobile Device. If an attacker gains access to the mobile device by either compromising the device or stealing the device, it may be able to access the encrypted password vault and the encryption key K_1. However, without the decryption key K_2, the attacker cannot decrypt the site passwords from the encrypted password vault. Unlike cloud-based password managers, BluePass does not keep master password and the vault on the same storage, thus obtaining K_2 together with the password vault is not practical. Moreover, the mobile phone itself may have its own protection, such as an unlock code or fingerprint verification, and remote data erasal.

4.2 Data Breach and Brute-Force Attacks

A serious threat to a password manager is data breach. Under this scenario, the attacker may be able to mount a brute force attack against the master password of a user. In a normal password manager such as LastPass or 1Password, the loss of a master password also means the loss of an entire password vault, namely, when an attacker successfully mounts a brute force attack against the master password, it can also retrieve all the passwords from the password vault since the key used to encrypt the vault is derived from the master password. Again, BluePass does not centralize the password vault storage. Instead, the password vault of a user is stored locally in its own mobile device. A server data breach would at most leak the user master passwords and then further leak the decryption keys. However, as the password vault of each user is not stored at the BluePass server, a data breach at the BluePass server cannot break BluePass.

On the other hand, assuming that a password vault is lost from a user's mobile device, we believe that brute-force cracking such an encrypted password vault is impractical given the current computing power. We emphasize that the password vault is protected by K_1, which is 2048-bit long randomly generated RSA key. Cracking K_1 is much harder than cracking a master password, which is generated by a human user within limited and predictable password space.

4.3 Broken HTTPS or Bluetooth

If an attacker compromises the HTTPS communication, it will be able to steal the encryption/decryption key pair (K_1, K_2) of a user. However, K_1 and K_2 are only transmitted through the web when a user installs BluePass on its mobile device (K_1) or when the user log on BluePass from a new computer (K_2), which makes the attack strictly time sensitive. Even though, having the key pair does not help the attacker to identify any of the user's site password, unless the attacker can also eavesdrop on the Bluetooth connection (i.e., co-located attack) to capture the encrypted password in transmission. On the other hand, eavesdropping Bluetooth alone does not compromise BluePass either, since the content is encrypted.

To succeed, the attacker needs to compromise both HTTPS and Bluetooth communications to steal site passwords from users. However, such a successful attack is very difficult to launch, due to time (to steal the keys) and location (to eavesdrop the Bluetooth) constraints. Furthermore, a large scale attack is infeasible since Bluetooth signals can only be sniffed within a short range.

5 Implementation

BluePass consists of three major components that cooperate with each other on user authentication, namely, a BluePass server for user registration and key distribution, a BluePass client application on the laptop for detecting and auto-filling the website login forms, and a BluePass app on the mobile phone serving as the password vault and administration console. We build the BluePass client application in a Macbook Air running OS X 10.10.4 and Chrome 46.0.2490.80. We implement the BluePass app on a Nexus 5 running Android version 4.4.2.

5.1 BluePass Server

We implement a BluePass server using Cherrypy [13], a python web framework. We use self-signed certificate in https to protect communication. The key pair (K_1, K_2) is generated using Pycrypto[1] on the server side. Sqlite database is used to store user data (see Table 1 for detail). When registering to the service, we do not use the standard 2FA to verify the phone number since it is not necessary for evaluation and user study. After registration, the user needs to log in BluePass on both mobile application to upload mobile phone Bluetooth MAC address and download K_1 and client-side application to download K_2 and Mobile phone Bluetooth MAC address.

5.2 BluePass Client-Side Application

We build the BluePass client-side application on Chrome platform, which consists of 2 modules: one Chrome application for Bluetooth communication and one Chrome extension for password auto-filling. We use two modules because currently Chrome extension does not support Bluetooth API while Chrome application does. However, only Chrome extensions allow reading and modifying the DOM of web pages, which unavoidably makes us separate client-side application functionality into 2 modules. Chrome application is more like a native application, but it is built on Chrome platform to deliver content in HTML, CSS and Javascript (e.g., Google Doc, Google Drive). It uses the *chrome.Bluetooth* API to connect to the Bluetooth device and then communicate with the smart phone through Bluetooth. The Chrome extension is responsible for detecting the authentication form and automatically fill the form after decrypting the site password from the mobile application.

The communication between the Chrome application and the chrome extension is implemented through Chrome External Messaging[2]. Specifically, this extension specifies the Application ID, which is a unique identifier for the Chrome application. After Chrome extension delivers the data to the application that is binded to the ID and has a pre-added listener, the listener can extract the data. The communication from Chrome application to Chrome extension works similarly.

Our prototype implements the BluePass client on the Chrome platform to simplify the communications among different modules; however, the framework of BluePass can be widely deployed on more platforms as long as both the computer and the mobile device have Bluetooth support and the browser extension

[1] https://www.dlitz.net/software/pycrypto/.
[2] https://developer.chrome.com/extensions/messaging.

is able to communicate with local applications on the computer. First, Bluetooth has become a standard device on modern computers and smartphones. Second, communication between browser extensions and native applications has been supported by most modern browsers, including Internet Explorer, Chrome, Firefox, Safari, Opera, etc.

5.3 BluePass Mobile Application

The BluePass mobile application starts a BluePass service, which runs in the background of Android and has a dedicated thread to listen to the incoming Bluetooth connection, which helps transparently authenticate a user to a registered website. The BluePass service inherits from the *Service* class in Android and keeps running until the user explicitly stops the service.

BluePass mobile application has a simple and clear user interface, which shows the status of the background BluePass service, either "running" or "suspended". The user can easily change the service status by clicking "Start BluePass Service" or "Stop BluePass Service" buttons. When the service status is running, the Bluetooth listener starts listening and remains active even the mobile device turns off the screen and goes to sleep. Whenever users would like to stop the service, they just need to open the application and click the "Stop BluePass Service" button.

We use RFCOMM Bluetooth protocol to establish communication between the mobile phone and the computer, since RFCOMM is widely supported and provides public APIs in most modern operating systems. Android supports two modes of RFCOMM connections, *secure mode* and *insecure mode*. The secure mode requires successful pairing before any RFCOMM channel can be established while the insecure mode allows connection without pairing two devices. Secure mode RFCOMM adds another layer of encryption. However, as BluePass communication is secured by (K_1, K_2) so that it does not rely on Bluetooth security. While insecure mode may fit better since it saves a pairing step from the user, Chrome application does not support insecure RFCOMM communication due to security concern. Therefore, we use the secure RFCOMM connection mode. Consequently, in the registration phase, the user also needs to pair the mobile phone and the computer first if they have never been paird before. Note that pairing only needs to be done once in a computer unless the user manually deletes paird devices on the mobile phone or computer.

6 Evaluation

6.1 Comparative Evaluation Framework

We use the comparative authentication scheme evaluation framework [6] to compare BluePass with other related authentication schemes. The results are summarized in Table 3. We can see that BluePass is *physically-effortless* since the entire authentication process is transparent to the user and *Quasi-Nothing-to-Carry* since users still need to carry their mobile phones though they carry

Table 3. BluePass scheme evaluation

Scheme	Usability								Deployability						Security										
	Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Physically-Effortless	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	Resilient-to-Physical-Observation	Resilient-to-Targeted-Impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Unthrottled-Guessing	Resilient-to-Internal-Observation	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable
Password			●		●	●	❍	●	●	●	●	●	●	●		❍						●	●	●	●
Firefox (with MP)	❍	●	❍	❍	●	●	●		●	●	●		●	●	❍	❍					●	●	●	●	●
LastPass	❍	●	❍	❍	●	●	●	❍	●	❍	●		●		❍	❍	❍	❍		❍	●	●		●	●
Tapas	●	●	❍	❍	●	❍	●		❍	●	●			●	●	❍					●	●	●	●	●
BluePass	❍	●	❍	●	●	●	●	❍	●	●	●			●	●	❍	❍	❍		❍	●	●	●	●	●
Sound-Proof			❍		●	●	❍	❍	●	●		●		●	❍		●	●		●	●	●	●	●	

● indicates that the scheme fully carry the characteristic and ❍ indicates that the scheme partially carry the characteristic (the Quasi prefix). We take rows 1-3 from [6], row 4 from [20], and row 6 from [23].

them anyway. BluePass is accessible since it does not require the cellphone to have signal or cellular data. BluePass is *Quasi-Resilient-to-Throttled-Guessing* and *Quasi-Resilient-to-Unthrottled-Guessing*. Although BluePass itself does not enhance the security of the underlying password mechanisms, it can help defend throttled and unthrottled guessing by generating long random passwords for users and motivating users to use more secure passwords since they do not need to remember the passwords.

Bonneau et al. [6] points out that the framework does not describe all possible properties of an authentication scheme. Besides these factors, BluePass also keeps a simple and clean user mental model, which is highly suggested since wrong mental models easily make user passwords weaker [7]. Furthermore, BluePass strengthen usability by not requiring users to delete their password traces after use on a untrusted computer as other password manager (e.g., log out master account or delete local password vault).

6.2 Password Auto-Fill Latency

For a usable password manager, the time required to fill the password field should be short. We record the delay between the time that the password input form is detected and the time that the form is automatically filled (denoted as T_{bp}).

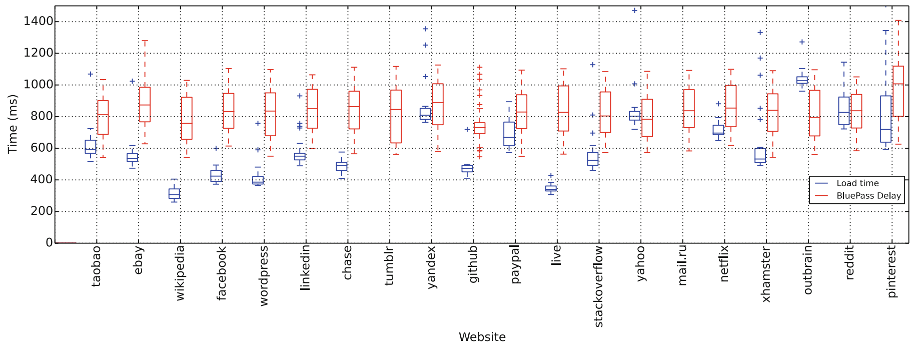


Fig. 3. BluePass latency

Table 4. Delay statistics

	Median	Mean	SD	Skewness
T_{bp}	778.0	814.6	158.3	1.6
T_{load}	599.5	837.6	691.3	2.6
T_{bp} (removed)	775.0	812.5	155.2	1.6
T_{load} (removed)	570.0	631.4	259.8	2.6

Since the delays on different websites may be different due to the specific website design, we choose 20 major providers from Alexa Top 100 website [30]. For each site, we make up a username/password pair and test the pair of credentials for at least 50 times. The password of each site is a randomly generated 16 byte string composed of all 4 characters types (Uppercase character, lower case character, digit, and special character).

Besides the Bluetooth communication latency, we also measure the loading time (denoted as T_{load}) for a website since page rendering (bottleneck to load a page) and Bluetooth communication tasks are running in parallel, indicating that the actual latency a user is experiencing is roughly $T_{bp} - T_{load}$, which is the time difference between BluePass running time and page loading time. T_{load} is measured by injecting a piece of javascript code, which measures the time when all javascripts on the webpage that need to run immediately are being executed subtracting the time that the browser is ready to send the HTTP request.

The results for all 20 sites are shown in Fig. 3. Figure 3 does not show the T_{load} results for two web services, Tumblr and mail.ru, that have much higher T_{load} (averaged 2,700–2,800 ms). Generally the BluePass delay time (T_{bp}) is slightly higher than the page loading time (T_{load}). To illustrate the extent of the time gap, we show statistical analysis in Table 4. In the last two rows, we do not include Tumblr and mail.ru in our analysis since they have significantly higher T_{load} that are not representative for normal cases. With the two sites excluded, the average T_{bp} is 814.6 ms, which is short enough to be acceptable by most users. Furthermore, the actual delay that a user experiences is $T_{bp} - T_{load}$, which is only 181.1 ms in average. The standard deviation for T_{load} is higher than T_{bp}. The loading time T_{load} could be different under various factors, such as network condition, website implementation, etc. Since T_{load} highly depends on the website implementation, heavy javascript use in a site could largely contribute to a high T_{load}.

On contrast, T_{bp} is relatively stable since Bluetooth communication and mobile device computing are almost the same in each login attempt. Since the delay caused by BluePass is bounded by $T_{bp} - T_{load}$, BluePass imposes a very low latency on the password auto-filling process. According to our user study, users can hardly notice the latency.

6.3 Power Consumption

One major concern of BluePass usage is the power consumption overhead on the mobile device, since BluePass requires the mobile device serve as a Bluetooth server that keeps listening to incoming connections. We measure the extra power consumption imposed by BluePass through monitoring the power levels of the mobile device when running BluePass password retrieval process in different frequencies. For comparison, we also record the power level of the device when Bluetooth is turned off (we call it a clean state).

To monitor the current battery level of the mobile device, we register a broadcast receiver in a simple battery monitoring application on the mobile device to listen to battery level changing event, upon which the current battery level and the timestamp are recorded. We tune the login frequency in the browser side (by refreshing a webpage in different frequency) to evaluate different use cases.

Except for the login frequency and BluePass on/off status, we keep all other settings exactly the same, such as installed and running application on the device as well as the network status (e.g., Wifi connection is turned off). We use a Nexus 5 mobile phone for evaluation, which has 2100 mAh battery capacity. As it takes a long time to use up the battery that has been fully charged, we run each experiment for 10 h before charing the phone and running the next experiment. Though the granularity of battery usage broadcasting is in percentage level that may not be highly accurate, it is sufficient to evaluate the power efficiency of BluePass in a 10-h test period.

Figure 4 illustrates the battery level dynamics through time under different experiment setups. "On" means the Bluetooth is turned on and "off" means the Bluetooth is turned off. Other lines represent the Bluepass log-in frequency. A reasonable frequency of login attempted by a normal user should not exceed 100 times a day, which means that the login frequency should lie around 0–10 times per hour. With 10 logins per hour, the power consumption is only 1% more than a clean state. We believe it is an unnoticeable overhead for users, given that almost 90% of users charge their phone more frequently than once per 2 days [27]. Besides, we can see that a significant power overhead is only incurred when the user tries to log in very frequently (17% when trying to log in every 2 s). However, normal users would not try logging in at such a high frequency.

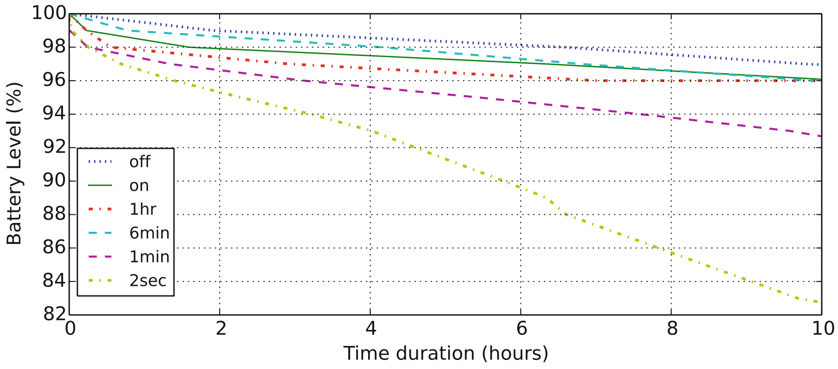
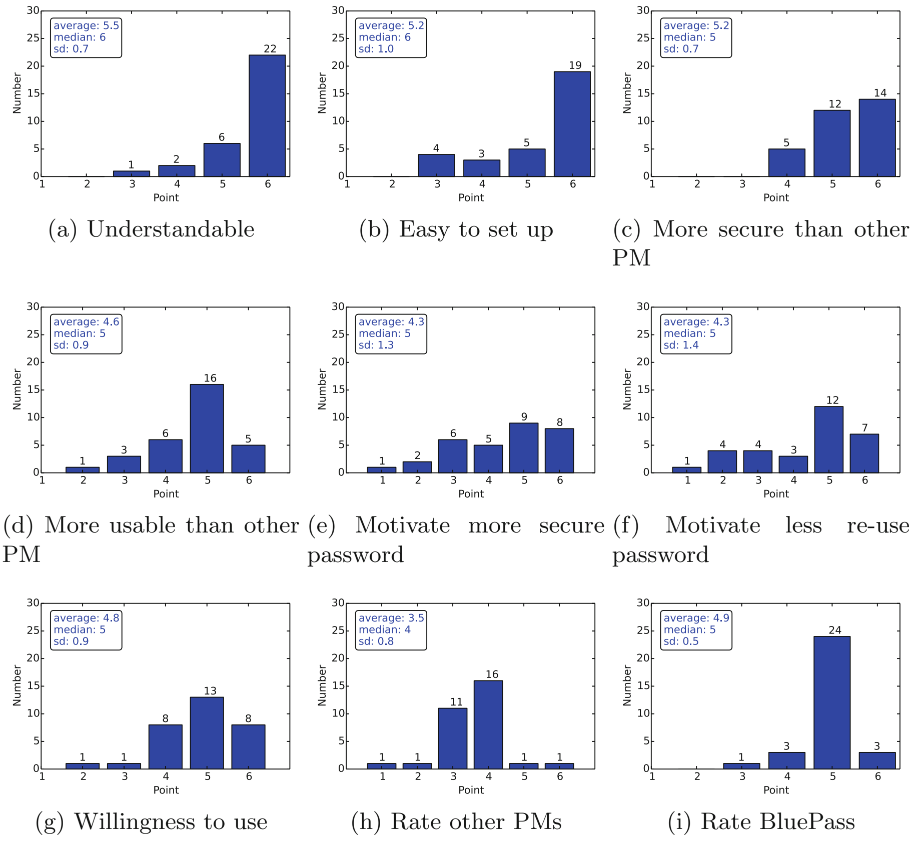


Fig. 4. BluePass power consumption



(a) Understandable (b) Easy to set up (c) More secure than other PM

(d) More usable than other PM (e) Motivate more secure password (f) Motivate less re-use password

(g) Willingness to use (h) Rate other PMs (i) Rate BluePass

Fig. 5. Survey results

In our experiments, the mobile phone is in a state that does not receive cellular or wifi signal, so the battery drains very slowly. When the mobile phone is in normal daily usage, the battery usage becomes much higher. However, the BluePass power consumption remains the level of 1% of total power with 10 h use.

7 User Study

To verify how real users rate the security and usability of BluePass, we conduct a user study to gather feedback and comments from normal users. Upon approval of IRB of our institution, we recruit 31 volunteers to use and comment on BluePass. The volunteers include 16 males and 15 females. As the study is only in a school scale, most of them age 20–30 years old. Besides, most of them have a bachelor degree. In order to spread our study of different computer expertise, we deliberately recruit volunteers from 10 fields of study.

We ask each of the volunteer to finish a series of tasks. They are (1) register to BluePass server and configure BluePass, (2) create a new account in our self-deployed test site, (3) log in the test site (Migrate password), (4) try using BluePass to log in again (Log in from a primary computer), (5) change the current password and try using BluePass to log in (Change Password), (6) configure BluePass in another computer and log in (Log in from another computer), (7) turn off BluePass and try logging in, and (8) turn on BluePass and try logging in. We also create a test website that has only login and changing password functions for the volunteers to operate on.

After finishing the tasks, the testers take a post-study questionnaire. The questionnaire mainly uses 6-point scale rating where 1 point means strongly disagree and 6 point means strongly agree. The results are shown in Fig. 5. Testers generally think the concept of BluePass is understandable and it is fairly easy to set up. 87% of testers (27 out of 31) agree that BluePass is more usable than any other password manager they have used before.

BluePass motivates the testers to increase password security. More than 70% (22 out of 31) of the testers state they are motivated to choose more secure passwords and less likely to re-use existing passwords, thus making their passwords stronger. However, though the testers report they are motivated to use more secure passwords, we notice that only 4 testers have tried using random passwords generated by BluePass to create/change their passwords, which may result from the fact that users feel “unsafe” to use a non-memorable password.

The majority of testers (94%) expresses willingness to use BluePass to manager their passwords. We also ask the testers to compare BluePass to other favorite password managers they have used, and testers show large preference to BluePass over existing password managers. To summarize, BluePass is generally considered more secure and usable than existing password managers by the testers. Most of them show preference to BluePass and willingness to use it. Thus, it is reasonable to conclude that BluePass does help users secure their passwords.

8 Discussion

8.1 RSA Key Pair

BluePass can use only one RSA key pair (K_1, K_2) to achieve bi-directional communication between the mobile phone and the computer. We must guarantee that the compromise of K_1 will not lead to the compromise of K_2, and vice versa. We know that all public key cryptography algorithms ensure that it is hard to derive the private key from the public key, but not vice versa. For instance, given an ECC private key, it is easy to derive the ECC public key, since $public\_key = private\_key * G$. However, for RSA, in theory, it is hard to derive either e or d from knowing the other one. Therefore, we can use only one pair of RSA keys with careful parameter settings.

There are two minor things to notice in the detailed RSA implementation. First, in practice e is usually chosen a small/fixed number, but this should be avoided. Second, RSA private keys are often stored in their "Chinese Remainder Theorem" form, which includes the two secret numbers often denoted p and q, from which the totient is computed. With totient and the private exponent, the public exponent is quickly computed. Therefore, BluePass cannot use the Chinese Reminder Theorem to speed up the calculation.

8.2 BluePass Limitations

BluePass has several limitations. First, a user has to carry a powered-on mobile phone to make BluePass work; otherwise, BluePass falls back to conventional ways that users remember and input passwords. Second, BluePass cannot work well when the mobile device or the computer does not support Bluetooth communication. In those cases, the hand-free benefit cannot be offered by BluePass. Instead, the users have to use their phones to display their site passwords after inputting their master passwords.

9 Related Work

Password is criticized to be insecure along its survival [17,21,22,31]. It is generally believed that there exists a general trade-off between security and memorability [32].

Whereas numerous evidences show that "easy" passwords are insecure, users generally do not follow advices from security experts and are inclined to choose weak passwords or reuse passwords [1,9,11]. Given a plethora of attack vectors, following security advice that specifically aims to defend against just one or few types of attacks becomes unrealistic for users. Therefore, it is crucial for a website to carefully manage its security policies, even allowing slight security sacrifice [12].

Due to various drawbacks of password authentication, many alternative schemes has been proposed to replace passwords [10,15,16]. However, Bonneau et al. [6] evaluated all mainstream alternative schemes and concludes that none of them is able to replace the dominating status of password authentication.

Facing the dilemma of not being able to replace passwords, many works focus on helping users manage and remember their passwords, which indirectly enhance password strength due to decreased memorability requirement. In consequence, password manager earns its prosperity. Despite ubiquitous "memorize and fetch" type of password managers such as browser built-in password managers or LastPass, researchers also proposed password managers that can enhance password security in addition to usability [14,20,25,29].

Password manager significantly reduce the memory burden on users. However, it has its own usability and security problems [18]. Severe security issues may also be introduced due to the fact that users failed to capture the correct mental model [7]. Silver et al. [26] demonstrated that careless auto-filling policy on non-https websites could make passwords be extracted directly from the web form by an attacker.

10 Conclusion

This paper introduces a hand-free password manager called BluePass for achieving both strong security and high usability. BluePass attains the security level of two-factor authentication by storing password vaults in a mobile device and the decryption key in the user computer separately. Exploiting the automatic bluetooth communication between the mobile device and the computer, BluePass enables a hand-free password retrieval process for users. BluePass also places the decryption keys to remote servers to support password portability while decentralizing the storage of password vaults to prevent a single point of failure. We implement a BluePass prototype on Android and Google Chrome platforms. Through system evaluation, we show that the password retrieval latency a user experiences is less than 200 milliseconds on average, and BluePass only consumes a negligible 1% battery power with 10 h normal use on a mobile device. Through a user study comprising of 31 testers, we demonstrate that BluePass does motivate users to choose stronger passwords and less likely to reuse existing passwords.

References

1. Adams, A., Sasse, M.A.: Users are not the enemy. Commun. ACM **42**(12), 40–46 (1999)
2. Lastpass suffers data breach again (2016). http://www.csoonline.com/article/2936105/data-breach/lastpass-suffers-data-breach-again.html
3. Beautement, A., Sasse, M.A., Wonham, M.: The compliance budget: managing security behaviour in organisations. In: NSPW. ACM (2008)
4. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: IEEE Security & Privacy (2012)
5. Bonneau, J., Herley, C., van Oorschot, P.C., Stajano, F.: Passwords and the evolution of imperfect authentication. Commun. ACM **58**(7), 78–87 (2015)

6. Bonneau, J., Herley, C., Van Oorschot, P.C., Stajano, F.: The quest to replace passwords: a framework for comparative evaluation of web authentication schemes. In: IEEE Security & Privacy (2012)
7. Chiasson, S., van Oorschot, P.C., Biddle, R.: A usability study and critique of two password managers. In: USENIX Security (2006)
8. Czeskis, A., Dietz, M., Kohno, T., Wallach, D., Balfanz, D.: Strengthening user authentication through opportunistic cryptographic identity assertions. In: ACM CCS (2012)
9. Das, A., Bonneau, J., Caesar, M., Borisov, N., Wang, X.: The tangled web of password reuse. In: NDSS (2014)
10. Davis, D., Monrose, F., Reiter, M.K.: On user choice in graphical password schemes. In: USENIX Security (2004)
11. Florencio, D., Herley, C.: A large-scale study of web password habits. In: ACM WWW (2007)
12. Florêncio, D., Herley, C.: Where do security policies come from? In: SOUPS. ACM (2010)
13. CherryPy - A Minimalist Python Web Framework (2016). http://www.cherrypy.org/
14. Halderman, J.A., Waters, B., Felten, E.W.: A convenient method for securely managing passwords. In: WWW. ACM (2005)
15. Jain, A.K., Ross, A., Pankanti, S.: Biometrics: a tool for information security. IEEE Trans. Inf. Forensics Secur. **1**(2), 125–143 (2006)
16. Jermyn, I., Mayer, A.J., Monrose, F., Reiter, M.K., Rubin, A.D., et al.: The design and analysis of graphical passwords. In: USENIX Security (1999)
17. Li, Y., Wang, H., Sun, K.: A study of personal information in human-chosen passwords and its security implications. In: IEEE INFOCOM (2016)
18. Li, Z., He, W., Akhawe, D., Song, D.: The emperor's new password manager: security analysis of web-based password managers. In: USENIX Security (2014)
19. Malone, D., Maher, K.: Investigating the distribution of password choices. In: ACM WWW (2012)
20. McCarney, D., Barrera, D., Clark, J., Chiasson, S., van Oorschot, P.C.: Tapas: design, implementation, and usability evaluation of a password manager. In: ACSAC. ACM (2012)
21. Morris, R., Thompson, K.: Password security: a case history. Commun. ACM **22**(11), 594–597 (1979)
22. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: ACM CCS (2005)
23. Karapanos, N., Marforio, C., Soriente, C., Capkun, S.: Sound-proof: usable two-factor authentication based on ambient sound. In: Proceedings of USENIX Security (2015)
24. Petsas, T., Tsirantonakis, G., Athanasopoulos, E., Ioannidis, S.: Two-factor authentication: is the world ready?: quantifying 2FA adoption. In: Proceedings of the Eighth European Workshop on System Security, p. 4. ACM (2015)
25. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J.C.: Stronger password authentication using browser extensions. In: USENIX Security (2005)
26. Silver, D., Jana, S., Chen, E., Jackson, C., Boneh, D.: Password managers: attacks and defenses. In: USENIX Security (2014)
27. How Often Do You Charge Your Smartphone? (2016). http://lifehacker.com/how-often-do-you-need-to-charge-your-smartphone-1441051270
28. Veras, R., Thorpe, J., Collins, C.: Visualizing semantics in passwords: the role of dates. In: IEEE VizSec (2012)

29. Wang, L., Li, Y., Sun, K.: Amnesia: a bilateral generative password manager. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 313–322. IEEE (2016)
30. The top 500 sites on the web (2016). http://www.alexa.com/topsites
31. Weir, M., Aggarwal, S., De Medeiros, B., Glodek, B.: Password cracking using probabilistic context-free grammars. In: IEEE Security & Privacy (2009)
32. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password memorability and security: empirical results. IEEE Secur. Priv. Mag. **2**(5), 25–31 (2004)