Abstract

Title: EXPLORATION OF THE SECURITY AND USABILITY OF THE FIDO2 AUTHENTICATION PROTOCOL

Team PASS
Gemstone Honors College 2022

Directed by: Dr. John Baras
Department of Electrical and Computer Engineering

Fast IDentity Online (FIDO) is a passwordless authentication protocol for the web that leverages public key cryptography and trusted devices to avoid shared secrets on servers. The current version of FIDO, FIDO2, has become widespread and is directly integrated into popular systems such as Windows Hello and Android OS. This thesis details two contributions to the advancement of FIDO2. The first is a modification to the protocol which uses Trusted Execution Environments to resolve security vulnerabilities in the Client To Authenticator Protocol Version 2 (CTAP2), which is a component of FIDO2. It is formally demonstrated that this modification provides a stronger security assumption than CTAP2. The second contribution is an outline of procedures and resources for future researchers to carry out a study of the usability of FIDO2 authenticators via a within-subjects experiment. In the study, subjects register an account on a custom web app using both passwords and FIDO2 credentials. The web app collects metrics about user behavior such as timing information for authentication sessions. Over the course of a week, subjects log in to the same web app every day using both authentication methods. Subjects complete entrance and exit surveys based on the System Usability Scale (SUS) according to their experiences. The surveys and user metrics would then be analyzed to determine whether users perceive FIDO2 as more usable than passwords.

Exploration of the Security and Usability of the FIDO2 Authentication Protocol

by

Team PASS

Zachary Breit, Hunter Dean, Tai-Juan Generrette, Samuel Howard, Balaji Kodali, Jim Kong, Jonah Tash, Phillip Wang, and John Wu

Thesis submitted in partial fulfillment of the requirements of the Gemstone Honors Program, University of Maryland
2022

Mentor: Dr. John Baras

Advisory Committee:
Dr. Jonathan Katz
Dr. Michelle Mazurek
Dr. Michael Marsh
Dr. Dave Levin

Acknowledgements

First and foremost, we would like to thank Dr. John Baras, for his mentorship over the last four years. We would also like to acknowledge Asim Zoulkarni and Mahshid Noorani for sharing their expertise and providing invaluable contributions to our work. We would like to thank Dr. Kristen Skendall for her guidance on our project and assistance navigating the IRB process. We would like to acknowledge Dr. David Lovell and the Gemstone staff for all of their support. For their assistance serving on our thesis defense panel, we would like to thank Dr. Jonathan Katz, Dr. Michelle Mazurek, Dr. Michael Marsh, and Dr. Dave Levin. We would like to thank Suzy Wilson for serving as our team librarian.

Contents

1 Introduction

1.1 The Problems with Passwords

According to the World Bank, the percentage of American Internet users has risen from 43% of the United States' population in 2000 to 87% in 2017 [1]. Growing alongside this expanded Internet connectivity is a burgeoning market of online services such as social media, mobile banking, and online retailers, many of which require users to store sensitive information about themselves in order to facilitate payments, enable communication, and more. To protect this information and prevent unwanted account intrusions, online services require users to authenticate themselves before accessing personal data, typically using text-based passwords. Passwords have been the predominant method of user authentication in computer systems since the 1960s, and security experts have taken issue with them ever since [2, 3]. In an ideal world, an individual would use a unique, difficult-to-crack password for each service that requires one, but the proliferation of password usage motivates taking security-diminishing shortcuts for user convenience. Crucially, the security of passwords in general suffers from users' tendencies to create weak passwords and reuse the same strings across multiple services.

The most secure passwords are long, random, and composed of a diverse set of characters [4]. Weak passwords, meanwhile, are short and feature predictable strings, including common dictionary words, simple numerical sequences, and/or personal information such as dates and names [5]. Strong passwords are often associated with difficulty of use, so users tend to opt for more convenient

alternatives that sacrifice security, even when they are aware of good password habits [5]. The public proclivity for choosing weaker passwords may be reinforced by their complacency—that is, users believing that their accounts will simply not be compromised [5]. One is left with a web environment in which roughly 32% of users, left to their own devices, employ passwords that are easily compromised by standard cracking techniques [4]. This tendency for users to prefer easy-to-remember but weak passwords over lengthier but stronger ones is an inherent flaw with the human aspect of password authentication.

Even when users do create secure passwords, they frequently reuse the same string across multiple sites [6], motivated in part by an aversion to the recovery process should they forget their account's password. 63.9% of password users were found to be inconvenienced by the process of forgetting and retrieving passwords, fueling desires to reuse passwords and mitigate any burden on their memories [5]. In fact, roughly 19% of individuals reused nearly identical passwords on multiple sites and 26% of individuals reused passwords for online financial services [7].

The danger of password reuse stems from the sharing of vulnerability. If a user registers several accounts protected by the exact same string, a data breach affecting one online service can expose associated sensitive information from all other services that are defended by the same hacked password [7]. Any individual service may be able to enforce its own set of strict password criteria, but it has no way to prevent reuse of that password elsewhere.

1.2 Criteria for Evaluating Authentication Schemes

Although many alternatives to passwords have been proposed, experts in authentication often disagree on what factors are most important when developing a strong authentication scheme. Security experts design systems that are robust but complicated to use. Biometric experts, meanwhile, focus on performance metrics like false-positive/false-negative rates while ignoring implementation concerns like hardware size and cost. These considerations are important within their respective fields, but they divert attention from the greater perspective.

In order to create a viable replacement for passwords, one must consider the multifaceted uses

of authentication. Bonneau et al. studied 35 popular password replacement schemes and compiled a list of 25 properties that encompass the various benefits offered by each scheme [2]. These 25 properties are divided into three broad categories: usability, security, and deployability.

Usability versus security is a classic trade-off in the field of authentication. In general, solutions that require less user interaction sacrifice elements of security, and more robust solutions demand more interaction from users. The less heeded category, deployability, refers to factors such as scalability, compatibility with existing systems, and ease of implementation. This third category is crucial since the adoption of a scheme is contingent on its ability to integrate into present infrastructure.

The properties that fit into the three overarching categories can be used to compare existing authentication alternatives to passwords.

1.3 Existing Approaches for Replacing Passwords

Despite the aforementioned flaws, passwords remain ubiquitous on the Internet. How do the existing alternatives compare?

To begin, developers have tried to tackle the problem of password memorability with password managers that handle some or all of a user's passwords by storing them in an encrypted format on either the user's machine or the cloud [8]. These managers take advantage of the benefits of passwords while removing the issue of reuse by eliminating the need to remember a unique string for each service. They also improve convenience by entering passwords on the behalf of users. However, the tendency to lock the functionality of the manager behind a master password creates an all-or-nothing scenario with the same risks as any individual password. Furthermore, password managers suffer from availability issues: if one runs a local manager on their machine, then its services do not work across devices; if one runs a cloud-based manager, then its services do not work without a network connection. Finally, password managers place the burden of finding a security solution on the user, rather than the service, which not only means that users may have to pay for the use of a manager, but that services cannot enforce their use or any related standard.

Federated identity management (FIM), allows users to sign in to multiple online services using a single set of login credentials [9], achieved by creating federations of services built on mutual trust. A subset of those in the federation, known as identity providers, provide authentication information to other trusted services. When many applications are linked to a federation, users are able to authenticate once to gain access to multiple services, which is known as single sign-on (SSO) [9]. FIM systems that utilize passwords for their initial login create an all-or-nothing scenario like with password managers. This time, however, the onus is on the services to provide, for better or worse. Such systems rely on services exchanging user information with each other securely, which may not always be the case.

Graphical passwords offer an alternative to text entirely by harnessing users' ability to remember images more easily than text, coming in three main flavors: recognition, recall, and cued-recall [10]. The variance and novelty of these schemes, however beneficial they might be for memory, may suffer from an initial learning curve that harms usability of the systems that employ them [11].

In an attempt to shore up the weaknesses of individual authentication methods, there is multi-factor authentication (MFA), a security paradigm that requires users to employ more than one method to obtain access to a service. One of the most widespread forms of MFA is mobile-based two-factor authentication (2FA), in which users are first required to supply their credentials (i.e., username and password) to an online service and then confirm their identity through a call or text message that contains a one-time-use code [12]. Other common secondary factors include security questions, hardware tokens, PINs, and biometrics [12]. Ultimately, MFA serves as an extra inconvenience for all parties involved—some helpful, some harmful—in that it forces attackers to contend with each additional factor added to the login process, developers to implement and secure each additional channel, and users to log in to services using multiple steps.

1.4 Biometrics and FIDO2

Most notably absent from the previous discussion of alternatives to passwords are biometrics—i.e., fingerprint reading, iris/retinal scanning, facial recognition, voice recognition, and the

like. Usage of such physical characteristics shifts the authentication mode from knowledge to being, from tokens of memory to the inherent physical qualities of an individual [13]. This bears with it the significant advantage of having no human memory requirement. With no cognitive strain, there is no following temptation to trade security for convenience and settle for something weaker. In fact, there is no "something weaker:" one is either able to provide their fingerprint (or equivalent reading), or they are not.

Biometric authentication has its own set of challenges that may limit its popularity. Concerning accessibility, physical complications may prevent certain people from being able to produce the verification needed. On the development side, biometrics are not as easy as passwords for services to enable. For users, providing a biometric reading requires additional hardware that may not be readily available. And as an extra inconvenience, biometrics cannot be shared like a password can, making it difficult for one to authorize others to access their account, if desired.

Enter FIDO2, a passwordless authentication protocol for the Internet standardized by the World Wide Web Consortium (W3C) into the Web Authentication API [14]. Fortunately, the growing adoption of FIDO2 helps to address most of the issues associated with biometrics. If one biometric reading is inaccessible to an individual, another type is probably accessible; FIDO2 supports a multitude of authentication options, making such options possible. Being a standardized protocol supported by most major Internet browsers and having various open implementations available eases the burden on developers to implement biometric-accepting capabilities. The acceptance of FIDO2 on major browsers also means that it is usable on mobile devices, and most modern mobile devices have the requisite hardware for biometric readings, such as fingerprint scanners, cameras, and microphones, already built-in, mitigating the hardware requirement's cost on users.

1.5 Present Research

As evident from the summary of password alternatives, there is not a one-size-fits-all solution to the issue of user authentication. However, we believe that the wider adoption of FIDO2, particularly in conjunction with physical biometrics, would be a suitable replacement for traditional text-based

passwords as a single-factor authentication method on the Internet. To verify the validity of this belief, our research is two-fold: investigate the protocol's theoretical security using formal analysis tools, and conduct an experiment to observe the protocol's usability.

A significant portion of our initial research's motivation originates from The Quest to Replace Passwords, published by Bonneau et al. in 2012. The researchers rated voice, iris, and fingerprint biometrics as having poor deployability across the board, which is less true today due to widespread mobile device ownership as well as the standardization and growing adoption of FIDO2. As such, our present research seeks to advance the state of the FIDO2 protocol and assess the usability of current FIDO2 implementations.

2 Literature Review

2.1 Background

Authentication, in the context of this research, is best defined as a process that uses some method of scrutiny in order to verify the identity of a user or entity [15]. For example, authentication may come in the form of passport verification, passwords, or fingerprinting. Most authentication methods can be categorized into three overarching categories: possession, knowledge, and characteristics; passport verification, passwords, and fingerprinting are examples of possession, knowledge, and characteristics respectively [16]. For this literature review, our reviewed authentication schemes are first grouped into these three categories, then considered alongside each scheme's strengths and weaknesses.

In order to analyze the trade-offs of each scheme, the criteria listed by Bonneau et al.'s "The Quest to Replace Passwords" are used [2]. These criteria, which characterize beneficial traits for a secure and convenient authentication method, can be categorized into the three aforementioned measures: usability, security, and deployability. Usability refers to how accessible and effortless an authentication method can be. Some of the criteria for a highly usable authentication scheme are being memorable (a user has to memorize a minimal amount of information to use this scheme) and time-efficient (an authentication scheme takes mere moments to properly verify a user's identity) [2]. Deployability refers to how easily an authentication method can be implemented on a larger scale. For instance, a deployable authentication scheme may be server compatible, which means that the

scheme integrates well with existing web server infrastructure for storing text-based passwords [2]. Finally, security describes how protected an authentication method is from intrusions by others. Thus, a secure authentication scheme would be resilient against brute-force, physical and internal observation, and phishing attacks [2]. These three categories of criteria are used in this literature review to provide a defined method to analyze an authentication scheme's benefits and drawbacks.

For the remainder of this review, we will explore research into many different authentication schemes, ranging from the common password to voice recognition to graphical authentication. These authentication schemes will be described and analyzed through the lens of Bonneau et al.'s criteria to determine the scheme's effectiveness.

2.2 Text-Based Passwords

The most widely used authentication method is the text-based password, a convenient combination of simplicity and inexpensive implementation. Its advantages were recognized even in ancient times: the Roman military passed "watchwords" from man to man and from division to division on a wooden tablet until every soldier was made aware of the word for the night [17]. It should be no surprise, then, that approximately two millennia later, passwords found a place within computers in the mid-1960s, when it was used by the Massachusetts Institute of Technology's Compatible Time-Sharing System as a means of protecting each of its users' private files [3]. The advent of the World Wide Web in the following decades, and in particular e-commerce sites, significantly increased the general population's usage of services that might require identity verification, with those services turning to passwords as their primary means of authentication [18]. Now, as online services have continued to expand and users find themselves with more passwords than ever, it is apt to consider whether password authentication is secure and user-friendly enough to justify its continued popularity.

The security of password authentication relies on the strength of individual passwords, where "strong" or "good" passwords take a long time for attackers to guess and "weak" or "bad" passwords take a short amount of time to crack. The time needed to guess passwords in a brute-force manner

depends on the password's length, the password's variance from standard dictionary words, and the size of the character sets used [19]. Using these properties as the basis for metrics, Ma et al. propose the usage of the pair (D, L) to encapsulate password strength, with D being the Levenshtein distance to the most similar dictionary word (i.e., the minimum number of single-character changes one needs to produce the password from a dictionary word) and L being the password's "effective length" (i.e., the length that the password would be, accounting for its character set diversity, if expressed only in digits). Ma et al. consider a password with D greater than or equal to three and L greater than or equal to 14 to be acceptable in strength. To achieve this level of security, they recommend making passwords "at least 8 characters long, with at least 3 special characters, plus other alphanumeric characters." The National Institute of Standards and Technology's 2019 revision to the 800-63B authentication guidelines concurs with this 8 character password minimum [20]. The security of password authentication appears to be a question of whether users follow the rules for creating strong passwords, such as the one proposed by Ma et al.

However, such rules are generally not followed by the public, as evident from Florencio and Herley's 2007 "Large Scale Study of Web Password Habits" [6]. After gathering password and URL data through a client that approximately half a million people voluntarily downloaded alongside Windows Live Toolbar, the researchers concluded that, though passwords play a significant role in the average Web user's life (having 25 password-requiring accounts and typing eight passwords per day), users often fail to create secure passwords. There were about 24,000 passwords categorized as "weak" of the approximate 150,000 total, and only about 7,200 "strong" ones. On top of these figures, it was discovered that the average user shares 6.5 passwords across 3.9 websites, with weak passwords tending to be reused more often than strong ones across more sites, compromising a greater number of user accounts.

Password reuse is a particularly concerning byproduct of the popularity of passwords. Beyond just increasing the chances that the repeated password is discovered by an attacker, having the same password across multiple accounts means that users' best-protected accounts are just as insecure as their least-protected ones. This weakness opens up an avenue for hackers to infiltrate secure systems by exploiting password reuse in less secure systems [18]. It is no wonder where the motivation to

reuse passwords comes from—as mentioned in Florencio and Herley's account, the average user needed to enter eight passwords over the course of a day in 2007 [6], a number predicted to only increase with time [21]. It was thought that most users are only capable of effectively using four to five passwords [18] and that the amount of passwords that users have to remember necessitates reuse. More recent research suggests that these poor security habits may not be the result of humans' incapable memories, but rather their perceptions of their memories as incapable [21]. Regardless of the cause, people are not utilizing strong, unique passwords for all of the accounts that today's Web-saturated society demands, leading to the pervasiveness of weak passwords that trade account security for user convenience.

2.3 Text-Based Password Creation Advice

One area of particular interest to researchers is determining how the advice that users receive when creating their passwords affects the security of their generated passwords. To study this, Yan et al. examined three sets of password creation instructions: no advice, advice to choose randomly generated passwords, and advice to choose passphrase/mnemonic passwords [4]. Each set of instructions was assigned to a different group of college students who used this advice to guide their password creation. Through their experiment, Yan et al. found that passwords based on mnemonic phrases are more secure than random passwords and that mnemonic phrases are easier to remember than naively selected passwords. This study's findings provide a potential solution for creating passwords that are strong and easy to remember while also being more secure through the use of mnemonic phrases as passwords. However, online services do not have a reliable way to compel users to follow such advice or to protect against new attacks that are evolving to target new forms of password creation.

Another potential method for improving password security is making an easy-to-remember password that is comparably secure to a password with random letters. To accomplish this, Abadi et al. discuss the three-way trade-off between user memory, security, and access time, as well as how to increase security while decreasing memory burdens in their research paper [22]. Most au-

thentication systems are made more secure by adding a randomly generated string to the end of the user-selected password to increase its strength, also known as a salt. By adding the salt, the password immediately becomes more secure as the password's length is increased. Also, when this salt is added before the password is hashed, the password becomes even harder for an attacker to guess as a single character can change the entire hash value assigned to the password. This salting method results in great security and fast access time, but it is still only as secure as the randomness and length of the password used by the user. Another drawback is that, regardless of how secure this process appears, the authentication system might not store the value of the salt in a secure location, resulting in little to no improvement over authentication systems that do not use a salt. To combat these issues, Abadi et al. propose appending randomly generated supplemental characters to ensure that all passwords are at least 20 characters, but then discard this supplement to make the client's computer brute force its way through the possible character combinations when a user logs in. This technique will improve security and decrease user memory requirements but significantly increase access time, leading to user dissatisfaction. Considering this downside, it is unlikely that commercial organizations would adopt this password padding technique, but it may be used by groups that value security over usability.

2.4 Password Managers

Another potential solution to the password memorability problem is password managers, which allow users to secure their accounts with strong text passwords while minimizing the amount of information that the user needs to remember. Password managers accomplish this by generating strong, randomized passwords for each online service. Although different managers have varying password storage methods, almost all of them lock the manager behind a master password that can be used to access all of a user's credentials [23]. This authentication system makes password managers convenient for users because they only have to remember a single password. Modern password managers autofill the passwords whenever the user requests them, further increasing convenience.

There are two main types of password managers: local and cloud-based. Local password managers store user passwords in an encrypted database file on the user's machine. Despite the added layer of security from the encryption, Zhao et. al. demonstrated that many of these local password managers are still insecure. This was accomplished by highlighting how the encrypted data is often stored without using strong key derivation functions [8]. On the other hand, cloud-based password managers store user passwords on remote servers. These managers commonly come in the form of a paid service, such as LastPass, Dashlane, or 1Password. These services are beneficial because they can take advantage of (k,n) thresholding schemes that allow data to be secured by multiple vendors. For example, k = n = 15 means that even if 14 of the vendors storing information about your passwords are hacked, the attackers know nothing about your data until they hack all 15 [24]. Browser-based Password Managers (BPMs) are another widely used form of password managers; they save users' passwords to the browser and automatically fill in users' credentials on login pages [8]. BPMs rose in popularity in response to the domination of text-based passwords as the most popular form of online user authentication. Nowadays, all five of the most popular Web browsers provide password managers as a built-in feature. Password managers have been effectively deployed and only have a few variables that affect its accessibility. Local password managers require that the user has access to the device they are stored on and cloud password managers require connection to all of the servers storing the passwords. Local password managers are limited because they do not allow the user access to their passwords from multiple devices. On the other hand, cloud-based password managers require a connection to all of the servers used to store the passwords.

While password managers help improve the process of password creation and storage, they still have factors that limit their viability. One of the biggest issues that they face is garnering a user base. Users decide not to use password managers for a variety of reasons. One common reason users cite is that they do not believe they will be a target of an attack, so a service that creates strong passwords for them would be useless [5]. Also, individuals may choose to manage their passwords themselves rather than using an internet-based service, which usually translates to them writing their passwords on a sheet of paper and storing it somewhere. For example, in a recent survey of

information technology professionals, 40% reported writing down their important business network passwords on paper [5]. If users believe that their method of creating and storing passwords is already sufficient, they would also think that a password manager performs redundant, unnecessary work. Although this false sense of user security is a hurdle for all authentication methods, it may affect the implementation of password managers the most.

Online password managers are more secure than the methods the average individual would use for password creation and storage. Strong passwords that are randomly generated and difficult to remember have an increased chance of being written down and password managers eliminate that need [5]. While a written password can be stored in a safe location, doing so does not eliminate the risk of password mismanagement. Password managers can create extremely secure passwords for other sites, but a "master password" is often needed to access the manager itself. This master password can greatly diminish the overall security of password managers. If the user needs to remember the master password, they may end up mismanaging the master password, putting their other passwords at a higher risk of being stolen. Additionally, cracking a user's master password for their manager would grant a hacker access to all of a user's passwords, as opposed to just cracking the password for a single service. The all-or-nothing nature of a password manager breach may cause general users to doubt their security and feel uncomfortable using one.

In a study conducted by Tam et al, they investigated the motives behind password selection, whether users knew what constituted good password-management behavior, and whether their behavior varied based on the type of account [5]. The ultimate goal of the study was to determine what online services can do to encourage users to adopt good password-management practices. The researchers began their study by inviting a group of university students to answer a series of questions regarding password management behaviors and tracking their responses. Additionally, the researchers collected data from a group of 140 separate Internet users on how time-frames affected their password creation and selection. At the end of the study, the researchers found that individuals understood the difference between good and bad password management behaviors and the potential consequences of bad management techniques. The researchers found that the fear of forgetting passwords contributes to users using weak passwords and that 36.4% of participants

were willing to sacrifice security for convenience [5]. This supports the concept of people not using password managers because of the additional time it would take to log in. In addition, the fear of forgetting passwords shown in the study also supports why users may opt to not use a password manager, as the fear of forgetting the master password could be a major deterrent. However, the researchers found that participants were willing to sacrifice convenience for security if they only had to remember one password, so these participants would likely benefit from using a password manager. Their data also supported the idea that password management behavior depends on the account for which the password is used, such as a bank account versus a social media account. Typically, participants used stronger passwords for accounts that had more immediate consequences if breached. Even though there are still debates regarding the psychology of password management, studies like that conducted by Tam et al. may explain why password managers are more appealing to some individuals than others. Additionally, the patterns identified in the study, such as the high user desire for convenience, may highlight why other authentication methods are gaining popularity.

2.5 Federated Identity Management

Federated Identity Management (FIM) is an authentication scheme that enables users to log in to many different online services using a trusted set of identity providers (IdPs). These IdPs are responsible for storing user information and transmitting a user's authentication status to other websites that trust these IdPs [9]. Although FIM systems typically rely on text-based passwords to authenticate users, they avoid some of the flaws associated with conventional passwords. Most notably, FIM frees users from having to remember separate passwords for many websites [2]. One early implementation of FIM is Microsoft Passport. When signing into a website that implements Microsoft Passport, a user is directed to a Passport page where they input a username/password to log in [9]. Before returning to the Passport participating site, the user is given the option to share their email, full name, and/or all other personal information (e.g., date of birth, country, ZIP code) during that Passport session. For the duration of that session (i.e., before the user logs out), the user does not have to re-enter their login credentials or enter any of the information they opted to share

when visiting other Passport-participating sites [9]. Passport's ability to provide users entry into multiple websites while only logging in once is known as single sign-on (SSO). Although Microsoft Passport integrated well with other Microsoft products such as Hotmail and MSN, it suffered from a few critical flaws. Primarily, Microsoft was not a trusted authority for authenticating users for more consequential web services such as online banking or retail [9]. Another significant flaw was that Passport did not give precise control over which user attributes are shared with which sites, which posed a privacy threat to users.

A more modern FIM approach is Shibboleth, which provides a more convenient user experience for login with better control over what user information is shared with which participating service providers (SPs). Shibboleth also simplifies the process of forming arbitrary federations between SPs and IdPs and provides a better mechanism for allowing users to choose their desired IdP, as opposed to Microsoft's more centralized system. To further ensure user privacy, IdPs in Shibboleth only provide SPs in a federation with a random identifier for the user instead of their actual login credentials, helping to anonymize user accounts [9]. Most critically, Shibboleth protects user privacy by forcing service providers to request specific attributes from the IdP, which must be authorized by the user [9]. Explicit control over user attributes is one of Shibboleth's biggest strengths. Shibboleth is commonly employed in login systems for higher education [25].

More recently, FIM has seen mass adoption through the OAuth protocol. OAuth 1.0 was originally released in 2010 to create a standard mechanism for users to grant online services access to their accounts on separate domains and was later improved upon with OAuth 2.0 in 2012 [26]. Prior to OAuth, allowing Facebook to suggest friends based on a user's Gmail contacts, or some similar action, required users to provide their Google login credentials to Facebook [26]. This poses security risks by providing Facebook with unrestricted account access. To improve user privacy, OAuth instead uses access tokens to facilitate authentication. Say that service A and service B both correctly implement OAuth. If a user decides to interact with service A through service B, service B would first redirect the user to service A's login page, where the user can specify exactly which permissions to grant service B [26]. Afterward, service B receives an access token, which can be sent alongside any request to service A to perform an authorized action on behalf of the

user [26]. Although OAuth has a multitude of use cases, it is often used to implement SSO. OAuth is not without flaws: notably, if a service fails to follow proper security protocols in order to secure the redirect endpoints, it could leave its users vulnerable to an attack [26]. OAuth also suffers if a service is unable to convey the extent of the permissions that a user is granting a separate service through their authorization interface. One key finding of Sun and Beznosov is that the access tokens that grant a service permission to obtain and modify user data were often vulnerable to being stolen [27]. After examining 96 SPs that connect to Facebook's OAuth endpoint, Sun and Beznosov found that only 21% use a method of encrypting web requests called Secure Socket Layer (SSL) when issuing authentication request and that 91% could have their access tokens stolen if there was a single cross-site scripting (XSS) vulnerability on any page on their site [27].

Overall, FIM systems facilitate SSO and reduce the total amount of login information that users have to remember, but still come with a myriad of flaws. Notably, they require sites to exchange information between each other, and many sites fail to implement even the most basic security measures such as SSL during these data exchanges [28]. Another concern is centralization; if an IdP has a vulnerability that allows an attacker to steal login credentials for users, the attacker could gain entry to all sites that the user logs into using that IdP [28]. Because FIM often involves redirecting users to the IdP in order to log in, FIM systems are vulnerable to phishing attacks wherein a hacker is able to redirect the user to a malicious login page instead of the legitimate IdP [2]. Despite these flaws, FIM is still a widely used authentication scheme that alleviates the issue of credential reuse by enabling SSO.

2.6 Graphical Passwords

Graphical passwords are a family of authentication schemes that rely on a user's ability to recognize, remember and recreate images. Graphical passwords have been the subject of academic study since 1996 [11]. Early graphical passwords required a user to touch predetermined areas on an image in a certain order [29]. However, different graphical schemes have been developed in recent years. Today, graphical passwords are commonly used on personal computers and online services

including online banking [30].

There are three main categories of graphical passwords: recognition, recall, and cued-recall-based schemes [11]. Recognition schemes require the user to identify images they have seen before. Recall schemes require the user to remember an image without aid such as requiring the user to draw an image purely from memory. Cued-recall schemes share characteristics of the two strategies above, requiring the user to remember details of their graphical password but with the aid of visual cues. An example of a cued-recall scheme is requiring the user to click certain points on a given image.

In "PassPoints: Design and longitudinal evaluation of a graphical password system," Wiedenbeck et al. introduced a cued-recall graphical password, PassPoints [11]. In the PassPoints scheme, a user creates their password by selecting an ordered sequence of points on an image of their choosing. A user logs in by simply clicking the correct points, within a certain tolerance, in the correct order. This study found that creating and remembering their graphical password was similar in ease and speed when compared to text-based passwords, but slower and more difficult when learning and entering the password. This study highlights the inherent difficulty of introducing a new authentication scheme to users as they are most likely highly experienced with traditional passwords and new to alternative schemes such as graphical passwords.

Picture Passdoodle, a cued-recall graphical password, allows a graphical password to be created via free-form drawing over a background image. The user-created drawing may consist of multiple strokes. A user logs in by recreating their drawing so that their strokes are within a set tolerance of the original drawing and drawn in the same order. Schwab et al. analyzed the security of this scheme by having users create, learn, and use their own Picture Passdoodles [10]. Users stated that Picture Passdoodles were faster, easier, and more secure than traditional passwords, despite having to learn how the service worked. This is in contrast to the findings for PassPoints described above. Specifically, this study found that incorporating a background image made the graphical passwords easier to create and remember.

Graphical passwords are a promising alternative to traditional text-based passwords, but they have their own strengths and weaknesses. The most prominent strength of graphical passwords is

their memorability. Images are easier to remember than text over extended periods of time [11], meaning that graphical passwords are generally easier to remember than text-based passwords. One of the largest security threats plaguing conventional passwords is simply that many users use weak passwords that are easy to remember, sacrificing security for the sake of recalling their passwords easily. Since graphical passwords utilize images, it is easier for users to construct and remember more secure passwords.

Although graphical passwords can be as secure as traditional passwords, they have unique security concerns. In particular, graphical passwords are vulnerable to certain types of attacks due to their visual nature, such as shoulder surfing and smudge attacks [10]. Shoulder surfing is a method of attack where the attacker simply observes a user input their password and is then able to mimic the user's password. Graphical passwords are especially vulnerable to this type of attack as they have a necessary screen presence, unlike text-based passwords. Smudge attacks, meanwhile, are possible on touchscreen devices where attackers utilize smudges on a screen to learn a screen's common pressure points, thus leaking authentication information.

2.7 Physical Biometrics

2.7.1 Fingerprints

Fingerprint recognition is a classic biometric that is based on a centuries-old forensic science concept [31]. The probability that two fingerprints are alike is approximately 1.9×10^{-15} [32]. This fact, combined with the general availability of one's fingerprints, has made this metric popular in consumer devices. The contemporary method of fingerprint scanning involves a three-piece structure. The first is a sensor that converts scanned fingerprint images into electrical signals; then, that data is then passed to the second piece, an interfacing chip that converts the data into a standardized format; and finally, the processed data sent to a standardized I/O interface [33]. These scanners have become a mainstay in biometric authentication and can be found in most contemporary mobile phones.

An important measure in fingerprint authentication schemes is fingerprint image quality. Fingerprint image quality is defined as a predictor of a matcher's performance [34]. The idea is that a fingerprint matching algorithm has a higher chance of correctly identifying matches or nonmatches when the scanned fingerprint image is of a higher quality. This metric allows systems to reduce matching errors and to determine when a rescan is required. Researchers from Michigan State proposed two indices for quantifying fingerprint image quality [35]. The first index extracts a quality score ranging from 0 to 1 by performing analysis on the energy concentration of the image as a whole. The second index generates a quality score by taking the weighted average of the quality of square image partitions. These two indices are evaluated on their ability to predict the performance of image enhancement, feature detection, and image matching. The global index outperformed the local index in predicting image enhancement, and the two performed comparably in the other two experiments. In the end, these two indices were shown as effective methods of quantifying a matching algorithm's performance.

Another important concern of any biometric scheme is secure storage. An attacker should not be able to recreate a biometric entry based on its representation in long-term storage. Tan and Lee proposed a fingerprint authentication system that encrypts fingerprint features using ring learning with errors (ring-LWE) [36]. A key feature of this scheme is that ring-LWE is an asymmetric algorithm with no known polynomial solution for quantum computers. In other words, ring-LWE has the potential to be "quantum-safe," unlike many popular encryption algorithms [37]. The scheme uses a remote server to store encrypted versions of fingerprint features. When a user requests access to a local device, called a Request-To-Authenticate (RTA), their fingerprint features are extracted on the local machine and encrypted using the ring-LWE public key. The encrypted features are then sent off to the remote server and decrypted using the server's private key. The server then decrypts the stored version of the features and compares the two copies. The server sends back an Accept-To-Authenticate (ATA) if the features match, otherwise it sends a rejection message. The researchers ran experiments in a controlled operating environment that demonstrated a reasonably fast time-to-authenticate of approximately 75ms. The fact that the private decryption key is stored on the same machine as the registered copies of fingerprint features is somewhat

concerning, however it is difficult to avoid this configuration and there are methods of mitigating harm if either the key or the encrypted entries are stolen. Regardless, this implementation is preferable because it allows public-key cryptography to be used for biometric authentication.

The major benefits of fingerprint recognition as a means of authentication are its maturity and the fact that it makes sense to use one's fingers for authentication since most devices that require authentication, namely mobile phones and computers, are already manipulated using one's hands. Therefore, using a fingerprint scanner is simple and direct in most applications of this scheme. This is directly related to one major downside of this scheme, which is its accessibility. Not all users have the requisite ability and dexterity to manipulate smaller fingerprint scanners, like those found on mobile phones. The second major disadvantage of fingerprint recognition is its reliance on hardware since not all devices have fingerprint scanners. The last major drawback of fingerprint scanning is that fingerprint sensors are susceptible to forgery attacks. In [38], Putte and Keuning describe how an attacker can use a fingerprint lifted off of a glass or fingerprint scanner to create a dummy silicone stamp that mimics a genuine fingerprint. This dummy fingerprint was accepted on the first or second try by many varieties of optical and solid-state fingerprint sensors [38]. In summary, the maturity of fingerprint biometric authentication makes it a helpful resource, however, several key flaws have prevented its universal adoption.

2.7.2 RFID Implants

Radio Frequency Identification (RFID) is an inexpensive technology that allows for contactless user identification and authentication. This technology has become very popular in the past few decades, as evidenced by its widespread use in building keycards and its appearance in larger applications such as the "EZ-Pass" toll booth system [39]. Furthermore, this technology has made its way into end-user authentication schemes. The basic design of a contemporary RFID system has two main components, an RFID tag, which is like a barcode, and an RFID reader, which is like a barcode scanner. RFID tags are small chips that can typically store 2kb of data. These tags receive requests from an RFID reader and are capable of responding to these requests using only the energy from the request signal itself, or ambient heat. This allows RFID tags to operate virtually

indefinitely. An RFID tag can easily be implemented into a hardware token authentication scheme, but a more recent scheme, RFID implants, has taken this usage one step further. The concept is to introduce RFID tags into the human body and to incorporate these tokens in a pseudo-biometric scheme.

An example of a publicly available RFID implant is the VeriChip microchip implant [40]. This scheme was specifically designed for patient identification in the medical industry. The implant features an FDA-approved tissue-bonding cap that holds the cap in place within the upper arm. The purpose of this product was the identification of patients, specifically "at-risk" individuals like those with Alzheimer's or diseases associated with memory loss. Since this implementation was only used for identification, the chips only held a 16-digit ID. What is notable about this product is that there have been no reported health complications associated with this product in the over 15 years since its release. Therefore, this implementation provides a promising case study for the physical aspects of implant technology.

Approved access is a major concern associated with this authentication scheme. Since RFID tags have minimal computational ability, it is difficult to prevent malicious actors from reading sensitive data. In many cases, attackers are able to read RFID data without victims even knowing that the theft occurred. Feldhofer et. al. proposed a symmetric challenge-response protocol for RFID tags that would overcome this risk [41]. The protocol makes use of the Advanced Encryption Standard (AES). This is advantageous since the algorithm has been demonstrated to be reliably secure. This study also demonstrates that a small tag could be implemented with the hardware required to complete AES computations. In all, this strategy is promising since the longevity of the security on RFID implants is a major concern. A major advantage of RFID implants is that they can provide a manufacturable, secure biometric. Cryptographic primitives can be built into their hardware, ensuring that they withstand common attack vectors. A major hurdle that this scheme faces is low social acceptance [42]. According to a 2002 study, 78.2% of respondents would be unwilling to put an RFID chip in their bodies. Another major concern is longevity, as it can be very costly and potentially harmful to remove an RFID implant. Therefore, it is very important that the methods used in the chip are secure and that the chip itself is not likely to degrade. In

summary, this scheme is a promising form of manufacturable biometrics that still requires further development before it can become a feasible replacement for passwords.

2.7.3 Retinal Scans

Developed in the 1980s, retinal scans operate on the assumption that blood vessel patterns in the eye are unique to individuals, which may be thus used reliably for identification. Typically, retinal scanning implementations involve the use of a sophisticated infrared (IR) light emitter and sensor to identify blood vessel patterns in the eye. The emitters use IR light to illuminate and generate a high-resolution image of the retina, and software then performs analysis on the retinal image to find characteristic patterns. In practice, retinal scanning is reserved for highly confidential military situations where passcodes or other forms of biometrics may be insufficient. Most established retinal scanning procedures require users to stare at a lens, during which time the infrared emitter and sensor take a 360-degree circular scan of the retina and establish any characteristic patterns. Once patterns are established, they are digitized into a 96-byte template and stored in memory to be used for verification later. In general, the advantages of retinal scanning include reliability and robustness [43].

Retina scanning schemes, if used in the right situations, are very difficult to exploit or fool. The retina itself is located deep into the human eye, making any alterations to retinal patterns extremely unlikely. Additionally, the template matching algorithms used by retinal scanning software are very unlikely to produce false-positives. However, retinal scanners are very expensive and are typically used only when security is of paramount importance (i.e., in classified or confidential environments). Therefore, retinal scanning is too situational and expensive to adopt on a commercial scale. In the consumer market, other means of biometrics such as fingerprints provide much cheaper prices and only a slightly smaller degree of security, rendering the market demand for retinal scanning technology very low [43].

2.8 Behavioral Biometrics

2.8.1 Keystroke Dynamics

Keystroke dynamics is an authentication scheme where computers store the unique patterns and habits of a user's typing rhythm to determine whether a user's access is allowed or not. By examining the latencies between keystrokes, keystroke durations, and the force and placement of fingers on keys, computers can create a user's profile for identification [44]. As opposed to many other biometrics, keystroke dynamics do not reflect a user's personal attributes (such as fingerprint, facial, or other personal data of a user), ensuring the confidentiality of its users' identities. This concept, although it has been around since as early as 1980, has seen little usage in security at any time [44].

Researchers from Georgia Tech constructed an intelligent keyboard (IKB), aiming to construct a keyboard that supported keystroke dynamic authentication in conjunction with a password. By including vertically stacked, transparent film materials on each key, their program measured a user's finger pressure and location of their fingers on each key for every keystroke. In addition, they were able to create a keyboard with negligible keystroke lag (compared to a modern keyboard) that classified keys into groups based on button size and mapped pressure on groups of keys against time for each user, creating graphs (profiles) for users when calibrating the keyboard by typing the desired password multiple times. A profile of a user contained the password for the computer system along with a plethora of keystroke data. This data included mappings of finger pressure against time, typing speed, and pause lengths between individual keystrokes, among other various data sets. Profiles for each participant in the study were unique and their system was able to pick out the correct user among multiple "imposters" (people who input the password impersonating a user's typing pattern) and grant that user computer access. As a result, they were able to boast an extraordinarily low Equal Error Rate (EER) of 1.34% [35]. Although their implementation does not meet the standards of modern-day security, the IKB had promising data for being one of the first security-based intelligent keyboards.

Monrose and Rubin examined a keystroke dynamics implementation developed by Joyce and Gupta, which achieved a correct identification rate of 87.18% (63 subjects) using a weighted probabilistic classifier [44, 45], . Joyce and Gupta chose to identify users based on their habits while typing text freely (i.e., free-text). However, identification rates could be further increased by placing restrictions on the authentication text. For example, the computer could prompt the user to type a preset password, phrase, or sentence, which would allow the computer to analyze the user's keystrokes in a more standardized fashion.

Using keystroke dynamics as an authentication scheme is a promising way to increase the security of our systems. Compared to many other biometric schemes, keystroke dynamics is not intrusive, especially for computer access, since users will be typing at the computer regardless of the authentication type. Keystroke dynamics is far from being a secure authentication scheme for widespread usage, but it does have places where its application is valuable. For example, keystroke dynamics could be an effective way to secure a master server containing sensitive user information. Since there is usually no outside access to the server and the only entry point is via console login, if a user's keystrokes in conjunction with their username and password match their claimed identity, access can be granted with confidence [44].

2.8.2 Voice Recognition

Voice and aural (audio-based) devices such as Amazon's Alexa and Google Home are common in modern-day homes. These devices have significant security issues, such as the existence of a voice hack, in which a computer-generated "voice" is used. The voice is unintelligible by the human ear, but voice-operated systems pick up on it with significant accuracy [46]. However, voice recognition security systems are seldom used for reasons such as a user's voice changing over time during the day or during their lifetime [47].

Voice-recognition algorithms have different strategies within their implementation and development, but Muda et al. divide one of their systems into a training phase followed by a testing phase [48]. During the training phase, multiple samples of each subject's voice are taken, and a template model is built by the system. Afterward, subjects may input their voice again to ensure

it matches with a given template model, and the algorithm makes a decision on whether the subject is recognized or not. The use of Mel frequency cepstral coefficients (MFCCs) is within most implementations of voice recognition software, including the implementation by Muda et al. [48]. When building a template for a user, mathematical operations are performed on the raw voice input data to obtain usable and comparable data for future inputs. Their process uses the dynamic time warping algorithm [48], which measures the similarity between two time series which may vary in time and/or speed by warping data and a comparison.

With the increasing popularity and practical use of machine learning and deep learning, speech recognition is becoming more practical and powerful [49]. The study claims that voice recognition will experience increasing usage within the banking industry and various web applications. There are multiple factors that need to be taken into account for voice recognition software. When used on a busy street, a user's voice may overlap with another's voice, whereas if a user wishes to be recognized from another room, their voice may be muffled and distorted. The ability to filter out noise and highlight a user's voice is key and plays an important role in the success of a working program. Obtaining the training samples needed to cover all the cases may be difficult. Boles and Rad used a support vector machine, which is a machine learning system that classifies input data, to overcome these issues to a degree [49]. In a study done by Ahmad et al., researchers noted that with their implementation, in order to minimize false-positives below 5% (their ideal), their false rejection rate rose to 75% [50]. These data show that voice-recognition security systems are promising, but additional research must be done in order for the programs to be more reliable.

However, voice-recognition is not without its fair share of faults and potential security issues. As stated earlier, a person's voice may change throughout the day, potentially posing problems with how strict the security system may be. In addition, voices can be altered voluntarily, where impersonators are able to spoof systems [47]. Multiple voice-altering software programs have already been developed that can disguise a user's voice. As of now, a plethora of constraints is necessary for users when calibrating the security system and when attempting to gain recognition. For example, users must not attempt to distort their voice, voice data must be saved (posing a potential for a data leak), and that linguistic content must include words known by the system (not necessary but

increases accuracy massively) [47]. The same research document notes that as research progresses, fewer of these constraints may be necessary, leading to a more reliable, satisfactory performance for security systems. However, their paper concludes with a statement that claims that there is no process that distinguishes individuals with absolute certainty from voice.

2.8.3 Gait Analysis

Gait-based authentication uses a person's walking style to identify them and can be separated into three major categories: machine vision, ground sensors, and wearable sensors. Most current gait analysis schemes use some form of computer vision to extract patterns, such as long steps, stature, and maximum distance between legs from a person's walking style. Those patterns can then be matched up against a person's gait profile template to authenticate that individual. Gait analysis is currently more of a proof of concept than an actual form of authentication, as human gait recognition is dependent upon a multitude of factors such as walking surface, angle of view, shoe type, and objects carried. Because of this, gait analysis lacks the robustness of many other systems such as fingerprint scanners and retina scanning to be used at scale [51].

2.9 Multi-Factor Authentication

Multi-factor Authentication (MFA) is a form of authentication that requires more than one factor (a way to authenticate a user's identity) in order to authenticate a user's identity and gain access to the system. There are three main types of factors which can be used to authenticate an identity: Knowledge Factor, something that the user would know such as a password, Ownership Factor, something the user would have such as a phone, and Biometric Factor, something inherent to the user such as fingerprint or behavior. The purpose of MFA is to provide the user with greater security than provided by Single Factor Authentication and to protect computing devices and critical services from unauthorized access by requiring more than one method of authentication [12]. MFA could be any combination of more than one way of authenticating an identity, but in most practices, it is a combination of the multiple types of factors [12]. MFA is not just useful for identifying

users for online services and providing security for the users' respective accounts. Ometov et al. break the applications of MFA into three groups: governmental applications, forensic applications, and commercial applications [12]. MFA for governmental applications can be best viewed with the requirements to obtain photo identification from either a state government or the Federal Government. In order to obtain a state identification for the first time, an identity document such as a birth certificate or any other document that has the patron's full name on the document in addition to a social security card, and proof of residency are required. These different types of documents are needed for the first time in order to provide a cross-check to make sure that the person filing for the identification is actually the person in question. They are only required to obtain a new identification because, after the first authentication, the holder has been authenticated. Similar documents are required in order to obtain a federal photo ID. To obtain these ID cards, the state or federal government utilizes multiple forms of information to authenticate the identity of the person or persons applying for an identification card. In criminal investigations, MFA can be used, for example, to identify an alleged perpetrator or a corpse [12]. Concerning this project, we are more concerned with the commercial applications of MFA as there has not only been an increase in diverse risks in online environments but also an increase in the diversity of authentication methods [16].

ATMs are an example of a commercial application of MFA. To properly use an ATM, the user needs both their bank card and PIN to access the desired account [12]. This is an example of MFA as the bank card is something that is owned and the PIN is something that is known. As an added layer of authentication, biometric authentication methods have been added to ATMs [52]. The inclusion of biometric authentication means that all three categories are utilized to authenticate the identity of the person using the ATM. Biometrics can be used to greatly improve identity proving through pairing with the other two factors and thus making it more difficult for an imposter to be authenticated as the actual user [12]. According to Sunehra in "Fingerprint Based Biometric ATM Authentication System," the fingerprint scanner system could be augmented by adding a Global System for Mobile Communication module which could then contact the proper authorities when the fingerprint does not match with the bank card and PIN [52].

The combination of something a user knows, has, or is, gives MFA its strength, as it is more difficult for an imposter to spoof the authentication system due to the different requirements for authentication. Despite the added security to the entire authentication system, MFA still faces security and authentication challenges inherent in each authentication scheme due to exploitable weaknesses. For example, the security of the known factor only lasts for as long as no malicious entities know the known factor, such as the user's password or PIN, as one of the modes of authentication is no longer secure. A common application of the possession factor is the communication of an authentication code through another communication channel such as a text or email [53]. However, this method is only secure if the user's mobile phone is neither lost nor stolen and malicious entities do not have access to the user's email account as otherwise this particular factor can be spoofed [53]. In order to authenticate with a biometric, a digital system requires a variety of components vulnerable to attacks at several different levels [12]. In order to prevent security breaches, only the proper user should be able to access the biometric and have his or her data processed. The digital system needs to be designed in a specific way to prevent imposters from being able to analyze either the physical system or the electronic patterns within it in order to prevent these entities from spoofing the system [12]. In addition, the digital system needs to protect the user's data from being stolen while it is in transit from the sensor to the processing/storage unit [12]. Furthermore, the digital system for a biometric should be able to handle a decent load, as the system is not feasible if it cannot handle the necessary throughput [12].

Although MFA is generally perceived as a secure authentication scheme, it ultimately suffers in its usability. In an ideal world, multiple authentication factors would not be necessary to fully protect an account, as the layered steps to log in are often inconvenient to users. Thus, despite acknowledging the security strengths of MFA, it is preferable to pursue a single-factor authentication method.

2.10 Literature Review Conclusions

Considering all of the existing research into alternative authentication schemes, it was surprising to find that very few recently-published papers explore how biometric authentication could be implemented on the Web. While a system for Web-based biometric authentication has been proposed and researched starting in the late 1990s, passwords still dominate online authentication [54]. Since the 1990s, the prevalence of biometric authentication has increased in tandem with the rising number of smartphones, which now commonly feature biometric sensors for on-device authentication. For example, Apple iPhones allow users to authorize purchases on iTunes, perform transactions using Apple Pay, and even sign into apps using Touch or FaceID [55]. Even though these examples rely on biometrics for authentication, the biometric is only used to authenticate on the device and the biometric template is not transmitted to the service provider. This configuration requires that a user's device is already trusted by the particular service before that user can log in using their biometric. In 2015, the Fast IDentity Online Alliance (FIDO), put forward a technical specification for a similar system, which relies on biometric sensors on trusted devices to authenticate users [14, 56].

3 TECTAP: Securing FIDO With Trusted Execution Environments

3.1 Abstract

FIDO2 is a passwordless authentication protocol for the web that leverages public key cryptography and trusted devices to avoid shared secrets on servers. It was recently standardized by the World Wide Web Consortium (W3C) into the Web Authentication API. The API integrates with many popular authenticators such as Windows Hello, YubiKey, and Apple TouchID/FaceID. In this paper, we summarize recent efforts to formally analyze FIDO2's security using symbolic and computational models. After exploring these findings, we present a formal specification of a modification to the FIDO protocol called TECTAP that leverages Trusted Executions Environment (TEE) technology to resolve security vulnerabilities associated with the current FIDO2 protocol.

3.2 Introduction

The FIDO protocol promises to "move the world beyond passwords" by providing users with a secure authentication method that generates unique credentials for each website that a user visits [57]. However, it is important to ask the question: does FIDO reach its stated security goals? There have been many attempts to formally analyze the security claims of FIDO1 and

FIDO2 in recent literature, which have uncovered numerous attacks on the protocol [58] [59] [60] [61]. Modifications made in the second version of the protocol (FIDO2) resolve some of these vulnerabilities [59]. However, as shown in [60], there are still vulnerabilities in FIDO2. Namely, the Client to Authenticator Protocol Version 2 (CTAP2) is vulnerable to a MitM attack because it uses unauthenticated Diffie Hellman to establish a shared symmetric key between a client and an authenticator [60].

To address this vulnerability, Barbosa et al. propose a replacement for CTAP2, called the Strong Pin-based Access Control for (sPACA) protocol [60]. sPACA replaces unauthenticated Diffie-Hellman in CTAP2 with a Password Authenticated Key Exchange (PAKE) to prevent MitM attacks. They prove that the sPACA protocol is Strongly Unforgeable (SUF) under the Bellare-Rogaway model, i.e., it is not vulnerable to MitM attacks [60]. sPACA represents a significant advancement in the security of the CTAP2 protocol. One problem that sPACA does not address is the requirement for users to enter a memorized PIN code into the client every time an authenticator connects to a new client [60]. Both CTAP2 and sPACA require users to resupply their PIN to the client for each session because (potentially compromised) clients have no way of maintaining a trusted state.

Therefore, one can further improve the security and usability of FIDO2 by introducing another protocol that uses PAKE with the requirement of a user-memorized PIN. To remove this requirement, we propose to use a Trusted Execution Environment (TEE) for executing a small portion of the client code. This approach was inspired by Fidelius, which is a technology that secures website forms on devices with compromised browsers and operating systems [62]. TEEs are tamper-resistant environments for executing code that provide isolation, privacy, and remote attestation capabilities [63]. TEEs rely on specialized hardware and software to protect against digital and physical attacks against program memory [63]. Furthermore, TEEs allow processes to store private, incorruptible state through a process called **sealing**, which our protocol uses to store a shared symmetric key for communication between a client and an authenticator. We call this new protocol **Trusted Execution CTAP (TECTAP)**. Under the Bellare-Rogaway model, we will prove that TECTAP has SUF if sPACA has SUF.

3.3 Background

3.3.1 FIDO Overview

FIDO2 is composed of two distinct parts: the Web Authentication (WebAuthn) protocol and the Client to Authenticator Protocol (CTAP) [57]. WebAuthn is a challenge/response protocol for authenticating a user to a web server, often referred to as a relying party. The user interacts with a trusted authenticator (e.g., a fingerprint reader) and a potentially untrustworthy client (e.g., a web browser) to communicate with the website on their behalf [60]. When a user registers for a web service using FIDO2, their authenticator generates a scoped private/public key pair that is only valid for that service. The key will be used in all future authentication sessions to verify the identity of the user and their authenticator. Below, we describe a simplified view of a WebAuthn authentication session between a user Alice and the website `https://example.com` using a biometric authenticator:

1. Alice requests to log in to `https://example.com`.
2. `https://example.com` sends Alice a random challenge.
3. Alice signs the challenge using her trusted authenticator. The authenticator uses an embedded private attestation key and a scoped public key for that service to sign the message.
4. Alice sends back her response.
5. `https://example.com` verifies Alice's response using the public key credential that they have on file for that particular authenticator.
6. Alice is granted access to the web service.

CTAP is a local protocol for ensuring that the client can only access the authenticator when given explicit authorization from the user [64]. The user grants authorization by gesturing to the authenticator. For example, the user could press a button on a hardware 2FA token or scan a fingerprint biometric. CTAP establishes a secure communication channel from the client to the user's trusted authenticator as follows:

1. **Authenticator Setup**: First, the user embeds a PIN inside of their authenticator. For

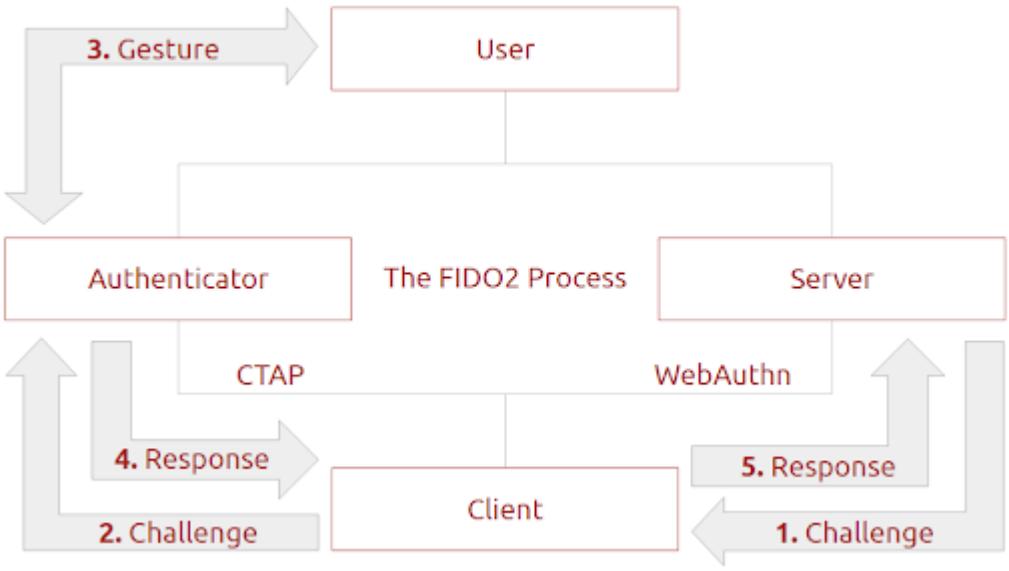

Figure 3.1: Graphical overview of the FIDO2 protocol

example, the user could register a new thumbprint on their fingerprint authenticator.

2. **Binding Phase**: Next, the user needs to give their authenticator permission to "trust" the given client. To do so, the user re-enters their PIN, which allows the client to "bind" to the authenticator. The client and authenticator both derive their own "bound" state during this interaction, which can be used in future interactions.
3. **Access Channel Phase**: Using its bound state, the client can securely communicate with the authenticator over a one-way channel. The other direction is not necessarily secure. Typically, the client will send some authorized message derived from its bound state, and the authenticator verifies that message with its own bound state.

3.3.2 FIDO1 Proven Security Claims

Preliminary formal models have demonstrated that the FIDO protocol satisfies many important security and privacy guarantees but still suffers from possible attacks. For example, a Man in the Middle (MitM) attack against the now outdated FIDO1.X protocol was discovered by Pereira et. al. [58] using an applied pi-calculus model and the ProVerif proof-checking tool.

The attack described in [58] targeted an optional step in the FIDO1.X specification for verifying that a relying party's appID (i.e., URL) matched the origin of the party requesting a user's credentials. If this check was disabled, a malicious phishing website could request user credentials for legitimate services. These requests would be treated as coming from the genuine service, and the malicious website could forward the user's credentials to the genuine service and log in on behalf of the user [58]. This vulnerability was patched in FIDO2 by requiring a user verification step [59].

Another attack, described in [65], targets the transaction confirmation feature of the FIDO UAF protocol. This feature prompts a user to confirm information about a transaction by opening a display. The protocol does not ensure that this display is secure, meaning that an attacker can display falsified transaction information to the user and create illegitimate transactions. Zhang et al. proposed a modification to the FIDO UAF protocol in which a secure display is created by utilizing ARM TrustZone, security functionality found on most modern mobile devices [65]. This modification adds a digital signature of the transaction content which is verified in the trusted execution environment created by TrustZone, resolving the vulnerability concern [65].

Four more attacks against FIDO UAF were uncovered in [61] using a symbolic model for the protocol developed using ProVerif. Their first attack was a "rebinding attack," in which an attacker sits in the middle of a registration session and waits for the user to receive a challenge from the relying party. If there is a malicious client, authenticator, or authenticator middleware module installed on the user's device, it could be used to forward the registration challenge remotely to the attacker. Then, the attacker could register on the user's behalf, and the relying party would link the attacker's authenticator to the user's identity [61]. The researchers found implementation vulnerabilities that enabled this attack against the China Mobile Pay and Jingdong Finance mobile apps, which had 214,424,508 and 1,043,164,617 downloads respectively as of October 2020 [61]. They responsibly disclosed these vulnerabilities to the China National Vulnerability Database of Information Security, which resulted in the following vulnerability ID: CNNVD-202005-1219. The other three attacks discovered in [61] require malware to be installed on the user's device (e.g., a malicious client or authenticator middleware module).

3.3.3 FIDO2 Proven Security Claims

A more up-to-date analysis of FIDO2's WebAuthn protocol by Guirat and Halpin [66] proved that it was resistant to phishing and man-in-the-middle attacks, again using ProVerif. However, the same researchers also showed that WebAuthn fails to satisfy its privacy goal of unlinkability between user credentials across different online services. Barbosa et al. [60] further verified the security of WebAuthn using the more robust Bellare Rogoway model, which captures the computational security of FIDO2. More concretely, WebAuthn is secure if the hash function used in the protocol is collision-resistant and the signature scheme used by the webserver and the authenticator to verify the other's identity is unforgeable.

Barbosa et al. were the first to analyze the security of CTAP2. To measure its security, they defined the syntax for **Pin-based Access Control for Authenticators (PACA)** protocols. A PACA protocol is considered **unforgeable (UF)** if probabilistic polynomial-time (PPT) adversaries are unable to generate fresh authenticated messages [60]. In their model, Barbosa et al. give adversaries the power to modify messages in transit, compromise any client that the authenticator is not currently bound to and read the binding state, and corrupt users that have not set up their authenticator to reveal their secret PINs.

The researchers proceed to define **strong unforgeability (SUF)**, which is a stronger security notion than UF security. It captures PACA protocols that are unforgeable even when adversaries are also able to compromise clients without a secure communication channel to the authenticator. They also define weaker security assumptions: **UF-t** and **SUF-t**, which capture attackers that cannot conduct (MitM) attacks during the binding phase of the PACA [60]. In terms of strength, we know that SUF $>$ SUF-t $\overset{?}{>}$ UF $>$ UF-t. That is (a) SUF is stronger than SUF-t, (b) SUF-t and UF are incomparable, (c) SUF is stronger than UF, and (d) UF is stronger than UF-t.

A key finding of Barbosa et al. is that the current definition of CTAP2 is only secure under UF-t, the weakest security assumption [60]. This weakness arises from the use of unauthenticated Diffie-Hellman during the binding phase of the protocol, which opens up CTAP2 to MitM attacks. To combat this vulnerability, Barbosa et al. describe the strong PACA (sPACA) protocol that replaces

Diffie-Hellman with a Password Authenticated Key Exchange (PAKE). In their report, Barbosa et al. demonstrated the unforgeability of sPACA and the composed security of Web Authentication with sPACA.

3.3.4 PACA Protocol

In order to analyze the security of CTAP2, we rely on Barbosa et al.'s syntax for PACA protocols [60]. Because TECTAP is modeled as a PACA protocol and derives its hardness assumptions from PACA syntax, we describe [60]'s work in considerable detail.

A PACA protocol has three parties: a user, a client, and an authenticator. The authenticator has two types of storage: static and volatile. The static storage holds onto a private key and a public retries counter, which is used to limit the number of failed attacks. The volatile storage stores the power-up state and the binding state. The client may also have its own volatile binding state.

Clients and authenticators communicate in the PACA protocols using a suite of five functions:

- **Reboot**: Resets the volatile storage of the authenticator and client. **Reboot()** should be executed before running any other protocol in the PACA suite.
- **Setup**: The user enters a PIN in the client, and the authenticator supplies its volatile storage state. On success, the authenticator establishes its static storage and the user confirms to the client that the protocol was successful. It is assumed that **Setup()** is performed over a secure channel since there are no authentication parameters established beforehand.
- **Bind**: Establishes a communication channel between the authenticator and the client. The authenticator supplies its power-up and static states and the user inputs a PIN through the client. Upon success, the binding states and session identifiers are set in the client and authenticator's volatile storage. Regardless of success, the authenticator increments its retry count.
- **Authorize**: Allows the client to send an authorized command M to run on the authenticator. The client also needs to send its binding state (for verification purposes). The output is an

authorized command (M, t).

- **Validate**: Verifies an authorized command. Takes in an authorized command (M, t), a user decision/gesture (input into the authenticator), and the authenticator's binding state.

The security model for PACAs includes an adversary that can:

- Passively observe honest executions of **Setup()**, **Bind()**, **Authorize()**, and **Validate()**
- Perform active attacks against an authenticator. This allows the adversary to learn a PIN by corrupting an authenticator. In stronger security definitions, the attacker is also able to perform active attacks against client oracles.
- In some security definitions, the adversary can perform active attacks against a client by revealing their binding state or by launching a MiTM attack during the Bind() protocol.

Within the PACA protocol, **Unforgeability (UF)** is the probability that an authenticator oracle accepts an authorized command (M, t) from an adversary given that the user did *not* authorize the command, or the authorized command was **not** output by **one of** the authenticator's valid partners. Unforgeability is *negligible* with the PACA protocol, meaning the probability is equivalent to or less than guessing an arbitrary PIN from the set of all PINs. It must also be the case that the PIN used for the authenticator was not *corrupted* before the authorized command was accepted. Here, *corrupted* means that the adversary is given the PIN used to set up the authenticator, and that pin is marked as corrupted. The client must also not be *compromised* after the latest reboot and before the command was authorized. Here, compromise is a function that the adversary can call which returns the client's binding state for a given authenticator and marks the client as compromised. UF security protects against attacks in which the adversary steals an authenticator and attempts to forge authorized messages on the user's behalf without corrupting the PIN or compromising any of the authenticator's bound clients. **UF-t** is the same as UF with the added restriction that the attacker is unable to perform active attacks against clients, such as MitM attacks.

Strong Unforgeability (SUF) is unforgeability with further restrictions. One is that the authenticator that validates the authorized command must be the *unique* valid partner of the client

that sent the command. This means that even if an attacker can compromise other access channels, they cannot compromise the access channel in question. The other added restriction is that the attacker can corrupt the user's PIN immediately after the bind phase, meaning the access channels have forward secrecy (i.e., remain secure even if the authenticator's secret PIN is later revealed). **SUF-t** is the same as SUF, except that the attacker is unable to perform active attacks against clients.

Schemes that feature UF-t or SUF-t can benefit from a user confirmation stage during the bind phase, which will help prevent online dictionary attacks. This added requirement will be a minimal burden for users who already have to type in a PIN for the bind phase and will entirely eliminate the possibility of online attacks against unstolen authenticators.

3.3.5 sPACA Protocol

The Strong Pin-Based Access Control for Authenticators (sPACA) protocol was developed by Barbosa et al. in [60] as an alternative to the CTAP2 protocol. As mentioned earlier, CTAP2 is insecure because it uses unauthenticated Diffie-Hellman during the bind phase of the protocol. This design choice leaves CTAP2 vulnerable to a MitM attack in which an adversary initiates binding with a client and then impersonates a token by sending Elliptic-Curve Diffie-Hellman Key Generation (ECKG) parameters to the client. As a response, the client will reveal part of the authenticator's secret binding state to the adversary [60].

Replacing unauthenticated Diffie-Hellman with an authenticated key exchange method is advantageous for numerous reasons. Firstly, unauthenticated Diffie-Hellman key exchange is vulnerable to MitM attacks in which an adversary intercepts communication destined for an uncompromised client. Another benefit of authenticated protocols is the ability to create independent keys [60]. The sPACA protocol introduces a PAKE to replace unauthenticated Diffie-Hellman. The password used to generate a shared random session token is a PIN that is stored in the authenticator during the setup phase of the protocol and subsequently entered by the user into a client during the binding phase of the protocol. The involvement of a PIN enables the use of an authenticated key exchange protocol; however, it also introduces usability concerns and additional attack surfaces.

The sPACA protocol introduces two additional user gestures: a one-time gesture to register a PIN and the gesture of entering a PIN upon client-authenticator binding. These actions have the potential to negatively impact the usability of the overall FIDO2 protocol since they increase the number of required user interactions, and the purpose of these interactions may be unclear to many users. Furthermore, the use of a PIN or password re-introduces concerns associated with password misuse. One of the key benefits of FIDO2 is the ability to use biometrics for single-factor authentication, which eliminates vulnerabilities specific to passwords such as password reuse and dictionary-based passwords. Therefore, the introduction of a PIN in sPACA, although confined to a local scope, diminishes some of the advantages of the FIDO2 protocol.

3.3.6 TEEs

Trusted Execution Environments (TEEs) are a popular subject of current research, and many hardware manufacturers use the term in marketing materials for their products [63]. TEEs typically use a combination of specialized hardware and low-level software to create an isolated execution environment with certain security guarantees. The isolated partitions used by TEEs adhere to certain properties which in turn provide the specific security guarantees of a TEE. Based on the survey of the state-of-the-art conducted by Sabt et. al., the most important of these properties are data separation, sanitization, control of information flow, and fault isolation [63]. Data separation refers to the property that data in one partition can not be read or written to by another partition. Sanitization is the property that the use of shared resources can not leak information out of a partition. Control of information flow refers to the property that the transfer of data in and out of a partition can only result from explicit functionality. This property is typically achieved using primitives known as OCALLs and ECALLs [62]. Fault isolation refers to the property that a security breach of one partition can not be used to compromise another partition. These four properties in tandem allow a TEE architecture to create a tamper-resistant execution platform for a Trusted Code Base (TCB) [63].

The work conducted on TEEs in this thesis uses Intel Software Guard Extensions (SGX) which are a set of extensions to Intel processors that enable trusted computing capabilities on Intel proces-

sors. In order to achieve the necessary TEE security properties, SGX reserves a region of memory for the Enclave Page Cache (EPC) [11]. This region of memory contains fixed-sized pages that are allocated to TEE partitions known as enclaves. SGX uses a structure called the Enclave Page Cache Map (EPCM) to keep track of the assignment of EPC pages to unique isolated enclaves. These enclaves are identified by the memory address of their SGX Enclave Control Structure (SECS) which is a data structure responsible for storing all per-enclave metadata. This memory architecture provides expandable, virtualized memory regions to code executing with an enclave. SGX allows for the provisioning of specialized Thread Control Structures (TCSs) which enable the concurrent execution of enclave code on multiple logical processors.

3.3.7 PAKE Protocol

Password Authenticated Key Exchange (PAKE), originally described by [67], is an interactive key exchange protocol where the client authenticates themselves to a server over an insecure channel using a short, memorized password. At the end of a successful PAKE execution, the server and client establish a shared key. If PAKE fails, the server and client learn no information other than that the client's password differed from the server's expected value [ThomasWu]. For practical use cases, PAKE keeps the client's password secret to outside eavesdroppers. Earlier key exchange protocols were prone to attacks, such as man-in-the-middle attacks, since they were not authenticated. PAKE's authentication of the client is what distinguishes it from other key exchanges.

One of the earliest instances of PAKE is the Secure Remote Password (SRP) protocol, discussed in [68] from 1997. SRP is an Asymmetric Key Exchange (AKE), while the previously discussed PAKE implementation by [67] is an Encrypted Key Exchange (EKE) [68]. EKE protocols allow two parties to determine a shared key via one party sending the other party an encrypted temporary public key [68]. AKE protocols do not use encryption. Instead, each party applies a one-way function to a secret and share the result of that function, called a verifier [68]. Since SRP does not require any protocol flows to be encrypted, there is no reliance on an encryption algorithm [68]. SRP requires only the client to keep a secret and compute its verifier, which the server then uses to authenticate the client [68]. It is important to note that this means the password is never seen by

the server. The lack of encryption and password sharing means AKE is a simpler version of PAKE than EKE. SRP's specific implementation of AKE is perhaps one of the most broadly used, due to its adoption by Apple and OpenSSL.

A far more recent instance of PAKE is AuCpace, discussed in [69] from 2019. This protocol is designed specifically for the Industrial Internet of Things (IIoT), referring to devices used in industrial applications that are interconnected. [69] argues that IIoT requires a PAKE protocol of its own, due to its unique qualities, such as requiring far more security certificates and having access to fewer computational resources. AuCpace provides a usable implementation in "AuCPace2551," designed to resolve issues that arise with other protocols in IIoT settings [69]. In particular, AuCpace strives for server-side and memory efficiency, minimizing the cost of a single client interacting with multiple servers [69].

3.4 Methods

3.4.1 Formal Specification

This section provides a formal specification of a modification of FIDO2 which introduces a new CTAP protocol called TECTAP and a reformulation of WebAuthn to accommodate the additional party required for TECTAP. Central to TECTAP is a new party known as the Trusted Client (TC), which is a portion of the Client code that runs in a TEE. The TC is responsible for establishing a secure channel with the Authenticator and exposing a limited interface through which an untrusted client can forward attestation challenges. Using the trust assumptions provided by TEE models, this new protocol design produces a system that is more secure and usable than sPACA and CTAP2 by removing the attack surface associated with a user-memorized PIN.

TECTAP consists of the same four phases used by sPACA: **Reboot()**, **Setup()**, **Bind()**, and **Verify()**. Throughout these phases, there are two categories of storage structures that persist data. The first is volatile storage, denoted $A.vs$ and $TC.vs$ for the volatile storage of the Authenticator and TC respectively. Data stored in these structures persists from phase to phase, however, it

is reset when the execution of either party ceases (i.e., when the Authenticator calls Reboot). The second storage category is long-term storage, denoted *A.lts* and *TC.lts* for the Authenticator and TC respectively. This data persists from phase to phase and to subsequent instances of the Authenticator and TC. The Authenticator and TC must be the only entities that have access to their respective long-term storage structures—otherwise, adversaries could trivially forge authenticated messages. This property is achieved by the Authenticator since it is assumed to be incorruptible and therefore it can safely read and write to any long-term storage device included in its hardware. The TEE satisfies this property if it supports confidential memory sealing.

The Authenticator and the TC both store a shared key **sk** in the long-term storage fields *A.lts.sk* and *TC.lts.sk*, which is the result of ECDH during the setup phase. Additionally, the authenticator stores a public pinRetries counter in the long-term storage field *A.lts.pinRetries*. The retries counter is decremented after every unsuccessful **Bind()** session between an authenticator and a TC. When the retries counter reaches zero, the authenticator becomes locked and its shared key must be re-initialized with a new call to **Setup()**. This counter is used to limit brute-force attacks against the authenticator.

The Authenticator and the TC store a shared bound session key **(bsk)** in the voltage storage fields *A.vs.bsk* and **TC.vs.bsk** respectively. The **bsk** is scoped to a single **Bind()** session and is generated using a PAKE that uses the **sk** from long-term storage as a password. The bound session key is not strictly necessary since the long-term storage of Authenticators and TCs will never be accessible to attackers under our trust model. However, these keys are included in our protocol to conform to the threat model proposed in [60], which allows adversaries to compromise access channels between Authenticators and Clients. The use of per-session keys gives TECTAP the same forward secrecy guarantees as sPACA.

One related distinction between the **Bind()** phases of sPACA and TECTAP is that authenticators in sPACA support multiple **Bind()** sessions (one for each bound client) while TECTAP authenticators only bind to a single Trusted Client at once. TECTAP makes this restriction because the secret message exchanged during the **Setup()** phase is only known by the TC that generates it and the Authenticator. Without some scheme to share these secrets between TCs, PAKE would fail

during the **Bind()** phase for all TCs without the correct **sk**. Therefore, if such functionality is desired, it is necessary to implement a scheme to notify a bound TC that its key has been deprecated. One such scheme would be a publicly requestable TC counter that is stored on the Authenticator which increments every time that a new TC is set up on that Authenticator. A TC can request this counter and observe if they will have to re-run **Setup()** to generate a new secret with the Authenticator. The sPACA analog of this process would be configuring a new user with a different user PIN, which is designed to be an infrequent occurrence. Functionally, this modification does not modify the capabilities of authenticators to bind to clients, it just means that **Setup()** calls are required every time an authenticator binds to a new TC. It is important to note that this protocol still supports multiple Clients, since any client can use the trusted ECALL interface exposed by the TC.

Another important distinction between TECTAP and sPACA is the additional need to consider Client/Trusted Client revocation. In the sPACA protocol, a Client is not capable of binding to an Authenticator without additional information, namely a PIN, that is provided by the User. Therefore, sPACA Clients are not capable of binding to an Authenticator by themselves. However, TECTAP Trusted Clients are capable of performing **Bind()** without external information. So, Users are no longer capable of preventing a Client from binding to an Authenticator by refusing to enter their PIN. In TECTAP, a User is still capable of revoking Trusted Client access by perfomring **Setup()** on a new Trusted Client. Since the Authenticator can only be bound to a singular Trusted Client, once the User performs **Setup()** for the new Trusted Client, the previously **Setup()** Trusted Client will no longer be able to perform a PAKE with the Authenticator and will therefore not be able to bind to the Authenticator. That is, the User is able to revoke the access of a Trusted Client by setting up any new Trusted Client.

Memory accesses to volatile and static storage are distinguished using the following notation. Solid left arrows ($\leftarrow$) denote storing data from volatile storage into a volatile storage location. Double left arrows ($\Leftarrow$) denote storing data from long-term storage into a volatile storage location. Dotted left arrows ($\dashleftarrow$) denote storing data from volatile storage into a long-term storage location. Further, we let **PINS** denote the set of all valid pin codes. For CTAP2, **PINS** is the set of all

4 - 63 byte UTF-8 strings [64]. Also, let a left arrow with a dollar sign on top ($\xleftarrow{\$}$) denote the assignment of a uniformly random element from a set.

TECTAP uses several cryptographic primitives throughout its operation, which are also described in [60]. $\mathsf{ECKG}_{G,\mathbb{G}}()$ is the key-generation function for the NIST P-256 elliptic curve Diffie-Hellman protocol. Here, G is a point on an elliptic curve that is a generator for $\mathbb{G}$, a cyclic group with prime cardinality $|G|$. H denotes the SHA-256 hash function and H' denotes the SHA-256 hash function truncated to the first $\lambda = 128$ bits. CBC_0 represents the AES-256-CBC encryption scheme with an IV of 0. Finally, HMAC' denotes the HMAC-SHA-256 message authentication code scheme with output truncated to $\lambda = 128$ bits.

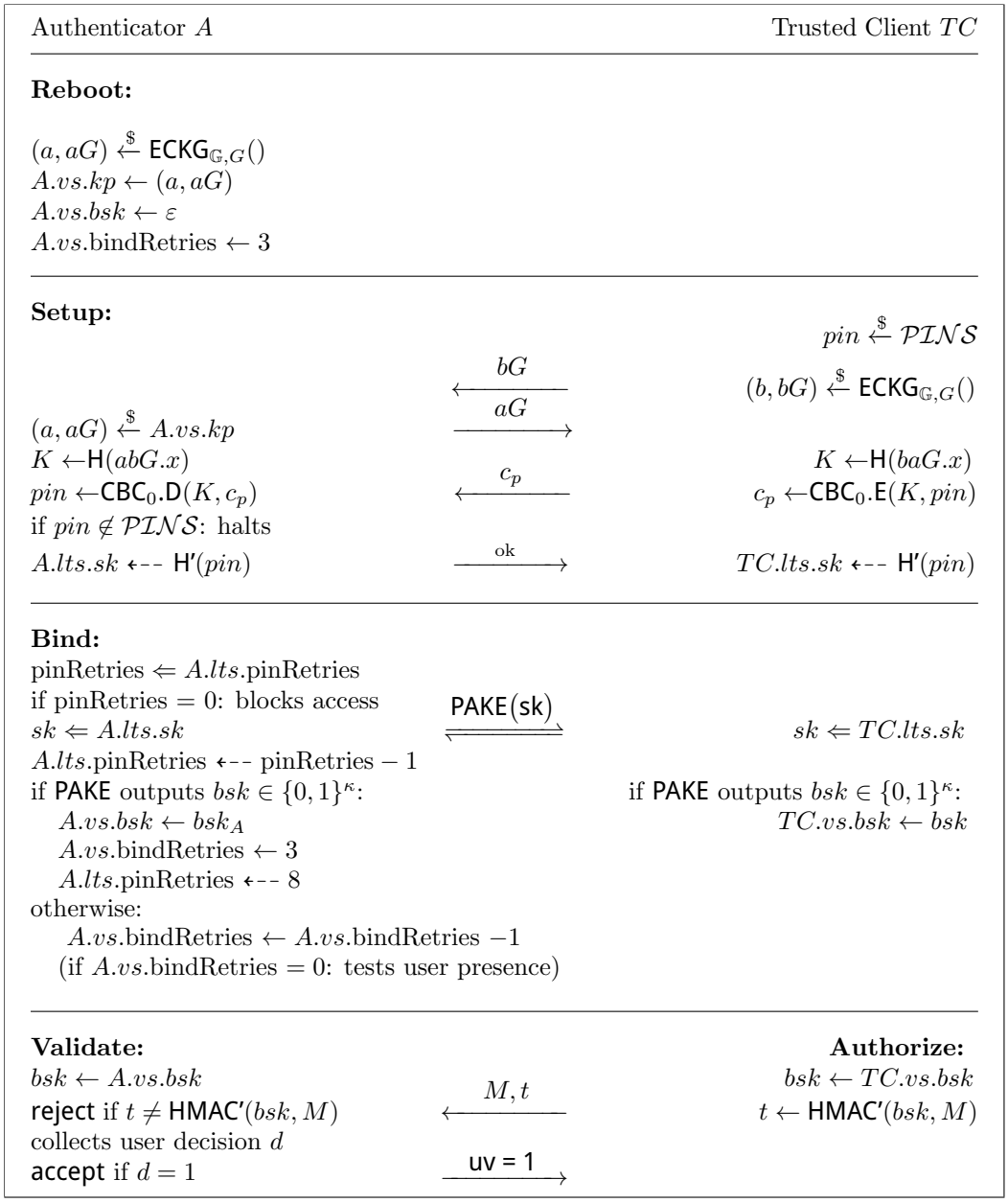

Figure 3.2: TECTAP Protocol Activity Diagram, adapted from Barbosa et al. [60]

Similarly, several modifications must be made to the WebAuthn protocol since no secure channel exists between the Authenticator and Client. Instead, a channel is established between the Authenticator and Trusted Client during **Bind**, and a Client can initiate trusted behavior that

sends communications over this secure channel using ECALLs. Specifically we define two ECALLs, `answer_rchallenge` and `answer_achallenge` which are used to send the parameters of a server challenge to the Authenticator for attestation. The ECALLs both return the resulting Authenticator signature to the Client. There are multiple ways to implement ECALL return functionality. One was is to expose a memory region that is write-only for the Trusted Client Enclave and read-only for the Client.

Apart from the use of ECALLs to communicate with the Authenticator, the rest of WebAuthnTE involves the same transmissions and calculations as standard WebAuthn. Therefore, WebAuthnTE relies on two main cryptographic primitives. A cryptographic hash function $\mathbb{H}$ and public-private signature scheme **Sig**. **Sig** consists of three functions, the first of which is **Sig.Kg**() which generates a public key **pk** and a private signature key sk. The next function is **Sig.Sign**(sk, msg) which signs a message using the private key sk. Finally, the last function is **Sig.Ver**$(pk, signature)$ which verifies that *signature* was signed with the private key associated with the public key pk.

Similar to WebAuthn, WebAuthnTE requires three parameters to the protocol. The first two are ak_t and vk_T. ak_T is the private signing key of the Authenticator and vk_T is the public verification key of the Authenticator that is shared with the Server. The next parameter is id_S which is the identity of the server that is inputted into the client. Typically, this is a URL. Furthermore, the Server must also be aware of the id_S. That is, the server must knows the specific URL used to access its FIDO functionality.

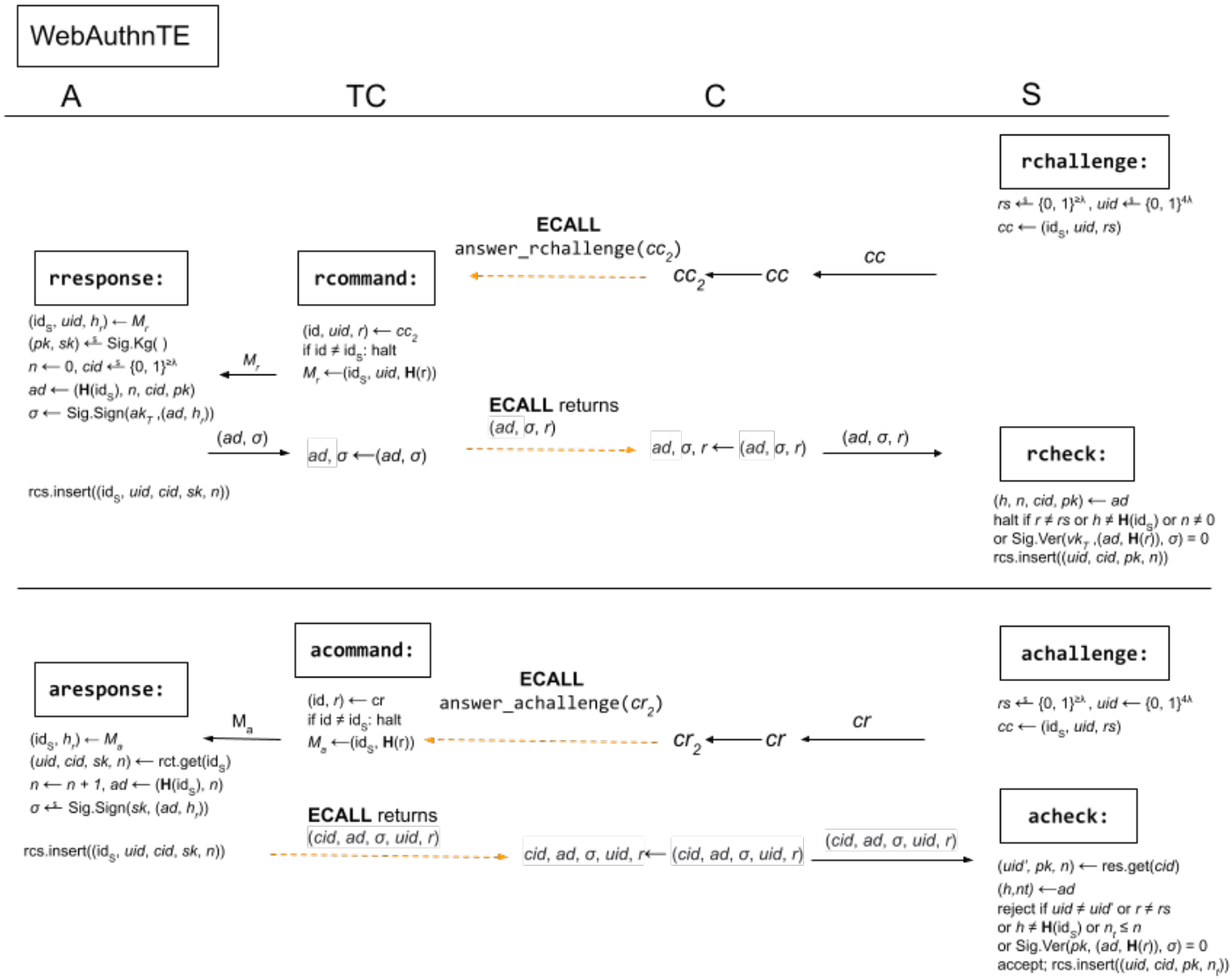

Figure 3.3: WebAuthnTE Protocol Activity Diagram, adapted from Barbosa et al. [60]

3.4.2 Example Implementation

To demonstrate the feasibility of TECTAP, an example system was constructed using a simple authenticator and an Intel(R) SGX enabled machine. The authenticator used is a Raspberry Pi Zero W "USB dongle," which is a microcontroller running the authenticator portion of TECTAP. The Client and Trusted client are executables running on a Debian 11.2 desktop with an Intel(R) Core(TM) i7-7700 CPU. The Trusted Client code was written using the SGX Linux SDK. Since WebAuthnTE is a more complex protocol that involves foreign server code and interactions with graphical clients like web browsers, its implementation is a future project.

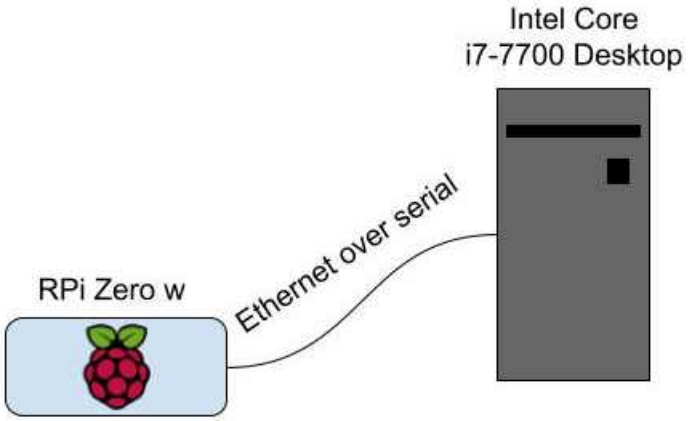

Figure 3.4: Diagram of hardware setup for the TECTAP example implementation

The authenticator portion of the implementation is a single executable written in C++. The executable can be run in one of two modes which are setup mode and bind mode, for the two phases of TECTAP. Both modes utilize the C socket framework to start a TCP server that waits for connections from clients. Upon receiving a connection, the authenticator will execute the phase of the TECTAP protocol corresponding to the mode in which it is running. The structure of the trusted client portion of the code is more complex since the SGX SDK requires an untrusted code segment to properly initialize and spawn a new instance of the trusted client enclave. Furthermore, certain portions of the enclave code such as accessing socket I/O resources requires the use of OCALLs since SGX enclaves can only run user mode code. Therefore, the trusted client consists

of three code bases, one trusted code base and two untrusted code bases, one which is used to initialize the trusted client enclave and another which contains the OCALLs used by the trusted client enclave. The trusted code base exposes two ECALLs to the untrusted enclave initialization code that interacts with the enclave. These two ECALLs are **setup()** and **bind()** for the two phases of the TECTAP protocol.

The implementation uses the OpenSSL library to provide all cryptographic primitive functionality. The ECDH key exchange that occurs in **Setup()** is handled by the OpenSSL Envelope(EVP) public key infrastructure. The public-private key pairs are generated using the NIST P-521 curve. The hash function used in **Setup()** is implemented by calling the SHA256 function. Finally, after a shared secret is established AES256 CBC mode is used to encrypt all traffic between the authenticator and trusted client.

3.5 Results

3.5.1 Trust Model

In order for TECTAP to achieve SUF, there are additional trust assumptions that must be made about TEEs and interactions with TEEs. First, the Trusted Code Base (TCB) that is loaded into a trusted enclave must be an accurate implementation of TECTAP. Next, it is assumed that the TEE will faithfully execute this code. Furthermore, assumptions are made based on the security properties of a TEE in Sabt et. al, namely, data separation, sanitization, control of information flow, and fault isolation [63]. Finally, it is assumed that the data sealed by a TEE enclave is only accessible by future instances of an enclave that resume code execution in the exact state in which the enclave initially exited.

3.5.2 Proof of Security

The following is a proof that the security of our protocol reduces to the security of sPACA which was shown to be SUF in Barbosa et. al. [60]

Theorem 1. *TECTAP is Strongly Unforgeable (SUF) if sPACA is SUF.*

Proof. First, note that the TECTAP and sPACA protocols are nearly identical, except that:

1. The client C in sPACA is replaced by a trusted client TC in TECTAP. The TC executes in a Trusted Execution Environment (TEE), which guarantees data separation, sanitization, control of information flow, fault isolation, and data sealing.

2. sPACA uses a secret pin code $\boldsymbol{pin_U} \in \mathcal{PINS}$, which is input by the user during each execution of **Setup**() and **Bind**(). TECTAP replaces $\boldsymbol{pin_U}$ with a uniformly random $\boldsymbol{pin} \in \mathcal{PINS}$ that is generated by the trusted client in **Setup**(). The hash of the random $\boldsymbol{pin}$ is retrieved from long-term storage by the authenticator and client during each execution of **Bind**().

3. TECTAP only allows a single bound client at once (denoted by a single bound session key bsk), whereas sPACA allows for multiple bound clients (denoted by the separate bind states $bs_{C,j}$).

Other features of the TECTAP protocol, such as the communication traces between (trusted) clients and authenticators or the behavior of the oracle functions, are identical to those of sPACA.

Since sPACA is assumed to be SUF secure, any efficient adversary that forges authenticated messages (M, t) against TECTAP must exploit one of the 3 differences above to make TECTAP SUF insecure. Thus, we must show that each of the protocol differences results in, at most, a negligible advantage for a probabilistic polynomial-time adversary $\mathcal{A}$.

1. Since trusted clients have strictly stronger trust assumptions than sPACA clients, any adversary that relies on special properties of trusted clients to forge authenticated messages could use the same strategy against ordinary clients. Thus, running clients in TEEs alone does not impact SUF security.

2. In TECTAP, $\boldsymbol{pin_U}$ is replaced by a uniformly random $\boldsymbol{pin} \in \xleftarrow{\$} \mathcal{PINS}$. The hash of this random pin $\mathrm{H}'(\boldsymbol{pin})$ is saved to the long-term storage of the authenticator A and the trusted client TC as $A.lts.sk$ and $TC.lts.sk$, respectively. Clearly, if $\mathcal{A}$ can extract $sk \Leftarrow TC.lts.sk$ from the

TEE without calling the **Compromise** oracle against TC, then it can forge an authenticated message. To do so, $\mathcal{A}$ simply executes **PAKE**(sk) with an authenticator to obtain a bound session key bsk. Then, $\mathcal{A}$ can generate valid authenticated messages $(M, \text{HMAC}'(bsk, M))$. However, our trust model guarantees that any long-term or volatile storage inside the TC is confidential during execution (data separation) and inaccessible by other process after TC finishes execution (data sealing). Without gaining access to $TC.lts.sk$, the adversary has no more advantage than an efficient adversary against sPACA.

3. Allowing fewer bound clients at once can only serve to increase (or maintain) the security of TECTAP relative to sPACA. If $\mathcal{A}$ against TECTAP could forge authenticated messages against a single bound client TC, that same adversary could be used to forge authenticated messages against any bound sPACA client.

Since none of TECTAP's modifications to sPACA produce a non-negligible advantage for $\mathcal{A}$, TECTAP is SUF if SPACA is SUF.

□

3.6 Conclusion

The modification to the FIDO2 protocol proposed in this paper is a successful improvement to the security and usability of the protocol. Specifically, it resolves a MitM attack in which malicious software running on a corrupted OS can impersonate the actions of a valid user. Furthermore, previous attempts to resolve this vulnerability re-introduce password-like schemes which decrease the overall usability of FIDO. Therefore, our concept of using trusted computing to remove this added burden is a novel advancement of the FIDO2 protocol. Additionally, we developed a sample implementation of our proposed extension which demonstrates the feasibility of future TECTAP implementations.

Moving forward, there are several possible avenues that could lead to the widespread adoption of TECTAP. The first is the resolution of the security flaws associated with current TEE technology.

The trust model is largely based on the guarantees of the Intel SGX architecture, and our TECTAP implementation uses SGX enclaves to run Trusted Client code. However, numerous publications have demonstrated that SGX has a multitude of unaddressed software and hardware vulnerabilities [70]. Intel has demonstrated that they are actively working to resolve these vulnerabilities [70]; hopefully, this emerging technology will mature to the point where applications like TECTAP can securely take advantage of guarantees the technology provides. Another potential method of expanding the usage of TECTAP is the creation of a Trusted Client platform that exposes an abstract API for varying TEE implementations. This would allow developers to design Trusted Clients for devices such as smartphones and non-Intel computers with limited friction. As TEE technology improves and support of TEE computing is added to more consumer devices, TECTAP could potentially become a standard local protocol for future versions of FIDO.

4 FIDO Usability Experiment

4.1 Introduction

When it comes to authentication schemes, it is not enough to just develop a system that is secure. An authentication method will only have value if it is actually being used by clients, meaning it also needs to be accessible and easily navigable for its users. In an effort to measure the usability of the FIDO2 protocol, we designed an experiment to record user experiences with a FIDO-based web app that we developed, with the goal being to assess how "usable" FIDO is as a single-factor authentication method. Evaluation is primarily guided by the System Usability Scale (SUS), which is a widely accepted tool used to measure a system's usability [71]. Additionally, we collect timing information about credential registration and authentication time and ask participants some open-ended questions.

Due to time constraints, we were unable to actually carry out this experiment; however, we describe our background research, justifications, and methodology here anyway to serve as both a proof-of-concept and guideline for future study. Any gathered results could then be used to reasonably conclude if FIDO2 would be a usable single-factor authentication system and compare it to text-based passwords.

4.2 Background

The usability of the FIDO protocol has become a topic of recent research interest. Many early studies focused on the use of hardware tokens through the FIDO protocol [72, 73]. Using hardware tokens as authenticators for the treatment group is a sensible choice because tokens allow participants to sign in to their account from any device with an accessible USB port. Across many studies, the majority of subjects found hardware tokens to be more secure than passwords [72, 73]. However, results on usability are mixed. In the study conducted by Ciolino et al., subjects in a laboratory environment were asked to sign up for a web service using FIDO U2F authentication. Each participant was randomly assigned to one of four hardware authenticators: a YubiKey security key, a SecureClick security key, an ePASS security key, and SMS One Time Passcodes (OTPs). Although all but one of the authenticators achieved an average SUS score that is considered "acceptable," only SMS OTPs received "acceptable" SUS scores from all fifteen participants [73]. It is important to note that Ciolino et al. studied FIDO U2F, which is an earlier version of FIDO that relied on two-factor authentication. This meant that subjects in their experiment had to remember an account password on top of using their security key.

With the introduction of FIDO2, usability studies of passwordless authentication became more popular. For example, Lyastani et al. found that participants preferred YubiKey security keys in a passwordless setting over traditional text-based passwords [72]. In a laboratory setting with 94 participants split into YubiKey and password groups, participants in the YubiKey group reported an average SUS score of 81.74 (A grade) compared to 71.77 (C grade) in the password group [72, 74]. Users in the YubiKey group recognized not only the increased security but also the reduced cognitive effort with using a hardware device instead of a memorized password. Despite these usability improvements, many participants were hesitant to switch their accounts to FIDO2. Chiefly, participants were worried about the loss or theft of hardware tokens [72]. After all, losing a hardware token requires users to reset their credentials for every online account. With a forgotten password, the damage is limited to a single account. In the end, 35% of the YubiKey treatment group said that they would be willing to transition all of their online accounts to FIDO2 1FA [72].

Apart from hardware tokens, mobile phones are also a popular authenticator choice for FIDO usability studies. One of the earliest usability studies into FIDO2 was conducted by Oogami et al., who investigated the registration and authentication process using Android fingerprint-based authenticators on the web [75]. Oogami et al. found that Android's unintuitive interface for registering a WebAuthn credential caused 70% of participants to try scanning their fingerprint on their phone screen instead of their device's onboard fingerprint reader [75]. Despite this design flaw, 60% of subjects said that they would be willing to switch to using a fingerprint sensor for their account. Unfortunately, the small sample size (n = 10) and the lack of a control group limits the study's generalizability.

Another FIDO usability study that used smartphones is that conducted by Owens, where users tested a mobile app called Neo that allowed users to use their phones as roaming FIDO authenticators [76]. A roaming authenticator is one that connects to any device that a user wants to sign in on and conducts the CTAP2 protocol remotely. Due to limitations with the CTAP2 protocol, Neo is unable to directly interface with the CTAP2 and requires a custom browser extension to connect to the target device [76]. Neo's median setup time was 16 minutes and 40.1 seconds, which caused 55% of the participants in the Neo treatment group to drop out of the study before completion, as compared to only 9% in the password treatment [76]. Neo also performed worse than passwords in authentication success rates (87% vs. 97%) and average authentication time (20.9 seconds vs. 8.1 seconds) [76]. These results demonstrate FIDO's lack of built-in support for less traditional authenticators (e.g., ones that rely on HTTP requests). Currently, the only supported transport bindings for CTAP2 are Universal Serial Bus (USB), Near Field Communication (NFC), and Bluetooth Smart/Bluetooth Low Energy (BLE) [64].

Because FIDO has two separate phases (registration and authentication), two-part designs are common among FIDO usability studies [73, 76]. The first part of the study consists of a guided setup phase where subjects create accounts with a FIDO credential, and the second part involves signing into a FIDO web application multiple times over an extended period of study. Due to many subjects' unfamiliarity with FIDO, researchers will often provide an introduction to FIDO during the setup phase [72, 76]. However, some researchers give no introduction to FIDO and observe

how users register with outside influence [75]. The second phase of these usability studies captures how user attitudes towards an authentication method change as they become accustomed to FIDO. Although it might seem that opinions on usability would not change significantly over time, this is not always the case. For example, in the study conducted by Owens, the SUS score for the FIDO treatment group improved from 66.6 (C grade) in the entrance survey to 81.3 (A grade) in the exit survey. We also opted to use a two-phase approach in our study in order to capture how sentiments towards passwords and FIDO change as users become more familiar with our web app.

Many of the recent FIDO usability studies use a between-subjects design with separate control and focus groups [72, 76]. Instead, our study employs a within-subjects design in which each participant uses both password and FIDO authentication. This approach allows us to directly compare the SUS scores and authentication times for both authentication methods, as Ciolino et al. do [73]. If we separated the control and focus groups, we could only compare usability data points in aggregate instead of on a per-subject basis. To prevent preferences based on treatment order [77], we chose to vary which authentication method each participant uses first. Half of the participants registered a biometric credential before a password, while the other half registered a password before a biometric credential.

4.3 Methodology

Surveys are an effective way to identify trends across responses across a population. The SUS is an example of a survey that has been shown to accurately measure the concept of usability [78]. One disadvantage of this approach compared with direct interviews is the potential for participants to interpret the questions asked on the SUS differently since an interviewer would not be present to clarify misunderstandings. However, we ultimately selected the SUS to gather results due to its wide acceptance as a reliable way to measure a system's usability [71]. In addition to the SUS scores, we will be collecting other statistics such as total login time and the number of failed login attempts using each authentication method.

If this study was conducted, we would predict that users would rate authentication with a

FIDO2-compatible fingerprint scanner as more usable than a standard text-based password. Additionally, we would expect the authentication time for users to be quicker when using FIDO2 with a fingerprint scanner in comparison to logging in with a text-based password. Finally, we expect that fewer failed login attempts when using FIDO2 compared to a traditional password.

Once all the data is collected, the analysis plan will involve comparing the FIDO2 results to the text-based password results. To begin, we will compare the SUS score from the first appointment to the SUS score after the second appointment for each of the participants for both authentication methods. The change in SUS scores between the two appointments will be calculated for each participant in the study, then the average change in SUS scores will be calculated for both FIDO2 with a fingerprint scanner and for text-based passwords. This analysis will capture the difference between the users' initial impressions of each scheme and their impressions after a week of daily use. The timing data will be analyzed in a similar fashion. The login times for the two methods will be tracked for all users across the duration of the study. Upon completion, any clear outliers of data, such as a user who displays clear inactivity, will be removed from the data. The login times for each method will be graphed and analyzed in an attempt to identify any patterns regarding the login times for each method as the users progressed throughout the study. The rate of unsuccessful login attempts will be analyzed in the same manner. However, any unsuccessful attempts to login will not likely be deemed as "outliers" because of the research team's inability to distinguish login attempts that do not submit data to the server (e.g., failed fingerprint scans).

4.3.1 Subject Selection

With each passing day, the number of people accessing the Internet seems to increase. Today, 93% of American adults use the Internet in some capacity [79]; globally, it is over half of the population [80]. In the years between 2010 and 2016, there was an average of 640,0000 first-time Internet users each day [80]. Although the subject matter of our experiment is universally applicable, we assumed the average American adult would be a suitable subject given that most American adults have access to the Internet in some capacity and that the United States is the country with the third-most Internet users [80]. Due to our location on a college campus, however,

recruiting would target enrolled students at the University of Maryland as a convenient sample population. Therefore, the overall design of our study operates under the assumption that it would be advertised and conducted at a university. While the subsequent sections will explain how our study was specifically planned, the general ideas can be replicated by another research team. Participation in the study was intended to be open to the entire student population, with the only constraint being that they needed access to a device with a fingerprint scanner. Additionally, it would be necessary for these students to have experience in effectively using said scanner. To recruit participants, we suggest primarily utilizing the Internet. Potential places we suggest advertising include Canvas, Reddit, and various honors college listservs.

We aimed to recruit 50 participants for reasons rooted in the relatively recent history of system usability studies. The purpose of such studies is to uncover usability issues that may not have been obvious in the design and development of a particular system. Jakob Nielsen and Thomas Landauer showed in 1993 that the number of usability problems found in a study with n users is given by:

$$N(1 - (1 - L)^n)$$

where N is the total number of existing usability problems in the design being evaluated and L is the proportion of those problems discovered by a single user [81]. If we are interested in the fraction of total problems discovered rather than the number discovered itself, the formula simplifies to:

$$N(1 - (1 - L)^n)/N = 1 - (1 - L)^n$$

This is where the often-claimed "five users is sufficient for a usability study" comes from. Nielsen and Landauer computed L to typically be 0.31, a value averaged across many projects. From the formula:

$$1 - (1 - 0.31)^5 = 0.84$$

That is, five study participants is enough to discover approximately 84% of usability issues; increasing n to 6 would yield about 89%, an improvement of only 5%. Nielsen argues that the costs

for such diminishing returns past the initial five users are not worth it, and thus n should stop there [82].

However, in 2020, there were approximately 4.54 billion active Internet users worldwide [83], and if the average person had 100 passwords during that same year [84], that is about 454 billion online accounts in need of authentication to access. In the hypothetical world where FIDO2 receives universal adoption, then a usability problem in the protocol that affects even 1% of the population would be experienced by a staggering 4.54 billion accounts. As such, maximizing the chance of discovering usability issues is a high-priority concern.

If we use Nielsen and Landauer's value of 0.31 for L, then our study having 50 participants would reveal 99.9999991% of usability problems. It is for this reason that, despite five participants being reasonable for our study, we sought to maximize participation within the limits of our budget at 50.

4.3.2 Procedures

In order to make an appointment to participate in the study, potential participants would be asked to confirm their eligibility and schedule a time through a Google Form. Eligible participants would select a 30-minute appointment block to register their credentials on our FIDO test app and complete an initial survey. They would choose from an initial appointment day from a three-day window, Monday-Wednesday. The participant would need to also schedule a return appointment for the same day during the following week; however, it would not need to be the same time window. We expect this process of scheduling the two appointments and to fill out the eligibility form to take less than 10 minutes to complete. The study would be conducted at AV Williams in the Systems Engineering and Integration Laboratory (SEIL).

On the day of the first appointment, the participant would need to arrive with their device of choice with a fingerprint scanner. When the participant arrives at their appointment, they will be greeted by a member and given a brief explanation of what they should expect. After this explanation, the participant will be handed a physical copy of the consent form to review and to sign before the registration process begins. Filling out the consent form includes a section for

listing their UID, which will be required for payment. Following signed consent, a team member will assist with the registration process. Each participant will be registering an account to the Team Pass app, at https://teampassexperiment.com/login. The registration process has the participant registering two sets of credentials: one set will be generated using the fingerprint scanner and the other will be a standard text-based password. Prior to registering, all participants will be randomly assigned into Group A or Group B. Group A will first register with passwords, then FIDO. Group B will register with FIDO, then passwords. Both sets of credentials will be tied to an anonymized email, which will be created by Team PASS. The text-based password will have to be at least eight characters long. During this process we will be tracking registration time for each method. Once a participant has registered, they will attempt to authenticate themselves using both sets of credentials. The authentication time or "login time" will be tracked by the app for each method. The first appointment is concluded with the participant filling out survey questions by hand based on their first impressions. The participant will then receive their first $10 payment through Terrapin Express, a reminder sheet, and a photocopy of their consent form. Before they leave, they will be asked to fill out the portion of the reminder sheet that asks them to write down the password they used. This is meant to help avoid situations where the participant forgets their password and is unable to login using it during the study. Since the appointments are for 30 minute blocks, the expected time commitment to complete the first appointment is 30 minutes. They will be verbally reminded to not to have their password "remembered" by the device they are using, and it is also included in the reminder sheet they will receive.

In the week leading up to the participants' second appointment, they will log onto the app daily using their biometrics and using the text password. The participant will use their biometric to login using FIDO, and they will use their text-password to login without FIDO. The order of which method is used does not matter, but the participant will need to logout in-between logging in using the biometric and the text-based password. Logging in using both methods should take the participants approximately 2 minutes to complete each day. Each day that a participant logs in using both methods, they will earn $1 of credit, which will be distributed at the second appointment. For example, if a participant conducts their first appointment on Monday, they can

potentially earn $1 each day from Tuesday till Sunday, for a total of $6 earned from successfully logging in. Participants will only be compensated for the days they login successfully using both methods, and if they arrive at their second appointment and complete the questionnaire.

Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8
In-Person Appointment ($10)	Remote Login ($1)	Remote Login ($1)	Remote Login ($1)	Remote Login ($1)	Remote Login ($1)	Remote Login ($1)	In-Person Appointment ($4)

Figure 4.1: Table showing tasks and compensation for each day of the experiment

On the day of the second appointment, the participant will once again need to arrive with their own device with a fingerprint scanner. On arrival, we will simulate the account recovery process by having the participant register a new set of credentials. This will include registering a new set of biometric credentials and a new text-password. Once again, the registration time for both methods will be tracked. This appointment will be concluded with the participant filling out the survey for a second time based on their experience with logging in to the app for a week as well as the account recovery process. The participant will then receive their second payment through terrapin express equal to $4 plus any credit accumulated by logging in during the week leading up to the final day. Since the second appointments are also for 30 minute blocks, the expected time commitment to complete the second appointment is 30 minutes. The survey questions are listed in the questionnaire in the appendix. Our questionnaire includes:

- 10 questions from the System Usability Scale about FIDO2 authentication
- 10 questions from the System Usability Scale about password authentication
- 4 short answer questions comparing FIDO2 and passwords

The System Usability Scale (SUS) is a tool for evaluating usability that asks respondents to rank 10 different statements on a 5-point Likert scale (1 = strongly disagree with this statement, 5 = strongly agree with this statement). We will interpret and normalize all responses to the SUS

questions according to the SUS scoring procedure. We will use the resulting scores to compare the perceived usability of FIDO authentication to password authentication.

The short-answer responses will not be numerically scored. They will be used as qualitative metrics to determine authentication preferences in addition to identifying aspects of FIDO usability that can be improved.

On the first day of the study, participants will only be asked to complete the first 20 questions—that is, all of the SUS questions. On the last day, they will be asked to complete all 24 questions, meaning that they will answer the SUS questions a second time and respond to the short answer prompts.

4.3.3 Risks

There is minimal risk associated with participating in this study. To mitigate the risk of exposing user passwords, we use end-to-end encryption (HTTPS) for all communication between participants and our web server. We also use salting/hashing, which is a common security practice to prevent attackers from recovering passwords in the unlikely event of a data breach. As an extra precaution, we would clear our database of all entries at the end of the study period. Since FIDO2 only stores biometric credentials locally on user devices, there is minimal risk of exposing any sensitive biometric through our web app. Additionally, since we would be using unique anonymized emails for each participant, no identifiable email would be stored in the database, so in the event of a data breach, none of the participants' information is at risk. We would ask participants to not reuse any existing text-based passwords that they are currently using. This would be done to minimize the risks of potential breaches to users' other accounts in the unlikely event that participants' passwords are leaked.

4.3.4 Benefits

There are no direct explicit benefits in participating in this study. However, participants may gain an improved understanding of password security and better password managing practices, as

well as introduction to rising alternatives on the Web such as FIDO2. Additionally, the information gathered in this study can assist in the development of usability in the authentication field.

4.3.5 Confidentiality

All participants would fill out a physical consent form on the first day of the study period when they initially arrive. We would then make a photocopy of the completed consent form and provide each participant with a copy upon conclusion of the first appointment. By having the participants sign the consent form, we would be collecting the first and last name of all participants. The biometric data used to login with FIDO would not be collected by the researchers. Once a photocopy is made of each, the original would be kept in a locked drawer in the principal investigator's office until three years after the conclusion of the study. Then, the consent forms would be shredded and discarded. The university ID that the participant provides would not be stored anywhere else but the physical from itself. Therefore, the data regarding the student's UID would be destroyed at the same time the consent form is destroyed. Because we are generating anonymized emails for each participant, each participant's actual email would not be stored in the app's database. Data regarding registration times and survey responses from both appointments would be stored securely using the team's Google drive database. The Principal Investigator as well as all co-investigators would have access to the data stored on the Google drive database.

4.3.6 Web App Architecture

The experiment can be conducted through a custom web application that participants would access remotely using their personal devices such as computers, mobile phones, tablets, etc. The web app is split into frontend and backend services hosted on a remote server. The web app's architecture is described in the following outline:

- Backend:
  - **Flask:** a lightweight web framework for creating Web Server Gateway Interface (WSGI) apps [85]

  - **webauthn:** a Python library for Web Authentication relying parties created by Duo Labs [86]

- Frontend:
  - **Bootstrap:** a responsive frontend framework used to style our website [87]
  - **Jinja:** Flask's HTML templating language [88]
  - **Web Authentication API:** a JavaScript API for accessing FIDO2 commands through a web browser [59]

- Infrastructure:
  - **MariaDB:** an open-source relational database [89]
  - **Caddy:** an easily-configurable web server with automatic HTTPS [90]
  - **Gunicorn:** a production-quality HTTP server for the WSGI protocol [91]
  - **Docker Compose:** a containerization tool used to orchestrate our different services [92]

The app is open source and the code can be found on GitHub: `https://github.com/team-pass/FIDO-login`.

The frontend of the web app features a login page, a registration page, and a user profile page.

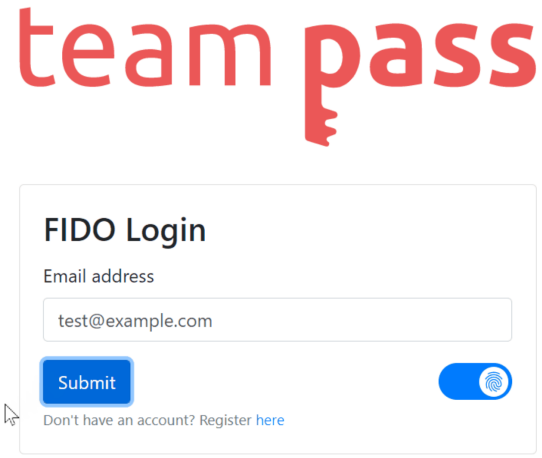

Figure 4.2: Screenshot of the web app's login page.

On the login page, users can sign in to the application by entering their email address and a password or FIDO2 credential. The user can click a toggle icon to switch between passwords and FIDO2 as their preferred authentication method.

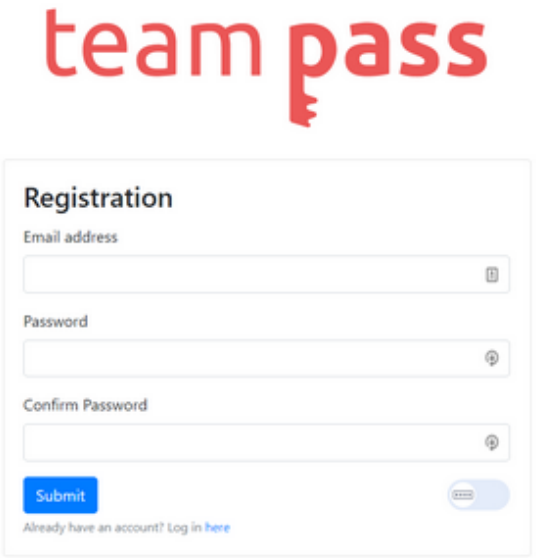

Figure 4.3: Screenshot of the web app's registration page.

Similarly, the registration page allows users to register a new account with a password or a FIDO2-compatible authenticator using the same toggle icon. If the user wanted to register a new account with a FIDO2-compatible authenticator, then they would enter the email associated with the account and use their device to authenticate themselves. Otherwise, the user must choose a password that is at least eight characters long (to meet NIST standards) and enter it twice to confirm their selection.

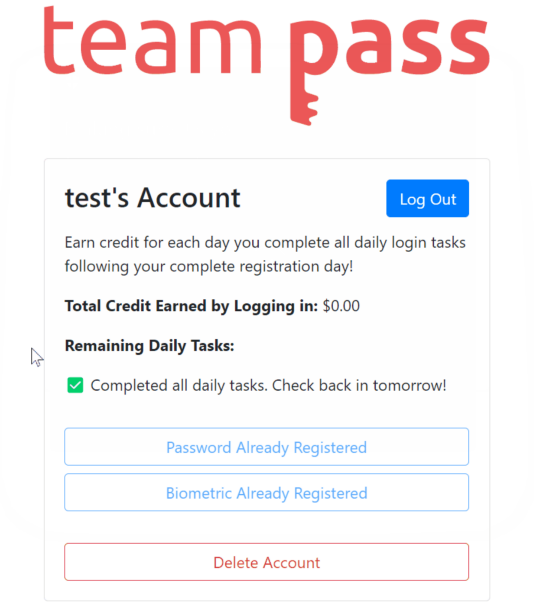

Figure 4.4: Screenshot of the web app's profile page.

After logging in, the user can see their profile page, which displays key information for participants such as the cumulative monetary credit the user has earned by completing their daily login tasks and a list of daily tasks to complete. From the profile page, the user can also add another authentication method to use with their account (either a password or FIDO2 biometric depending on which one they have not employed yet, or neither if both are already registered), log out, or delete their account entirely.

Since the usability study expects that participants will check into the web app each day, the list of daily tasks exists as a reminder to users to log out and re-authenticate themselves using the method they did not use to initially log in. Of course, this relies on the user having both a password and a biometric associated with their account. If they are missing one, the "Remaining Daily Tasks" will instead remind them to register the absent credential. Once a user has logged in using both authentication methods in a single day, the list will simply display "Completed all daily tasks!" and the total earned credit displayed will be increased by a fixed, predetermined amount.

On top of this basic functionality, the website collects various usage statistics for participants.

The site records a timestamp whenever a user loads the website, focuses on a text box, clicks the "authenticator toggle" button, or submits a form. We would use this timing information to analyze how long it takes users to authenticate and register using passwords and biometric scanners. In addition, all interaction data gathered is tracked by a session, which is a level of storage indirection that allows us to associate a set of page interactions with a user even if that user switches browsers or operates within a private browsing mode. Furthermore, we have metrics to attribute failed login attempts to a specific user and identify the login success rate for both authentication methods.

4.4 Conclusion

Having been unable to conduct our usability experiment, we cannot draw upon any data to discuss results. We have offered in its stead complete plans for the study that we hope can be carried out by an interested party in the future. In particular, we reviewed several similar studies, sought to justify a target participant count of 50, detailed the procedures for what the experimenters and participants would be doing on each day of the week-long plan, and developed an open source web application expressly for this study. The app offers both password and FIDO2 registration/login capabilities, internally tracks the time that users spend interacting with certain page elements, reminds users of daily login tasks with a simple list on their profile page, and records day-to-day completion of tasks to inform the user of their cumulative earned compensation credit. We believe the combination of our descriptions and publicly available code on GitHub to be broadly sufficient to reproduce the requisites of the experiment.

Naturally, the next step is to actually carry out the study that we designed. A future experiment group can do so by following a logistical plan similar to our own. Begin by offering an initial appointment date for participants to meet in person, receive a briefing, register password and FIDO credentials on accounts created on our test application, and take a survey of initial perceptions. After some length of time where study participants log in independently each day using both authentication methods, invite them back for a concluding appointment in which they are debriefed and surveyed again. The results from the internal timing data collected by our web application and

the participant responses to the SUS can then be derived quantitatively.

We would also like to acknowledge some of the limitations of this usability experiment. For one, study participants only interact with our custom web app, not any other FIDO2-enabled system, which limits the extent to which we can generalize user perceptions. For another, although our app functions with any type of FIDO2-compatible authenticator, we restrict participant eligibility to those who own and are comfortable using a device with a fingerprint scanner. While this ensures that we do not compare password usability with another knowledge-based authentication scheme such as a PIN, it causes the experiment to lose the characteristic freedom of authentication method choice that FIDO2 offers. Furthermore, had we conducted the study ourselves, many of the participants would have been young university students for convenience, which would have limited the generalizability of our results. The Internet, and therefore online authentication, is used by people of all ages and education levels, so an ideal experiment would include participants that are representative of that diverse population.

5 Equity Impact Report

5.1 FIDO Makes Secure Authentication Possible for All

With the rise in popularity of mobile devices and the services offered through them, the "digital divide"—the gap between those who have access to online services and those who do not— has not been shrinking [93]. In order to combat inequity among underserved populations, Sultan recommends creating a training which covers "key cybersecurity terms and concepts ... cyber-hygiene and best practices... [and] downloading, installing, and use of anti-virus and malware software" [93]. There are commercially available products such as password managers and antivirus software that help address the second and third items of Sultan's proposed training to promote "cyber-hygiene" and safety in downloading things from the Internet, respectively. However, by virtue of having an associated cost, low income users are unlikely to purchase these services as they are not necessary for Internet access. This puts such users at greater risk than those who are more willing and able to spend money on self-protective measures. Wider acceptance of FIDO on the web provides a way to universally improve the security of systems and users at no necessary cost to the users, mitigating this inequality between income groups.

FIDO promises security as a default by allowing users to define a digital identity by the most secure means available to them. In traditional single-factor password schemes, individuals associate their unique identity with a username and password combination. As previously discussed, passwords suffer from usability issues and promote unsafe authentication practices, intentionally or

not. Solutions such as password managers can be used to move the root of trust from the memory of a user to an arbitrary factor such as a biometric. However, such software can be costly and is subject to the drawbacks previously outlined in our literature review. Once again, FIDO offers a no-cost solution that allows individuals to select the most secure and usable authentication factor available to them.

It is also important to consider that individuals interact with technology uniquely. Some differences in user behavior are the result of personal preferences, while others are dictated by the physical capabilities of a user. For example, some individuals lack the dexterity to manipulate a fingerprint sensor. Contemporary smartphones are hosts to entire sensor suites, with biometric sensors being included in nearly 80% of all smartphones sold annually [94,95] and fingerprint scanners being among the most common, but it is still important in such a case to give individuals who wish to take advantage of biometric authentication the option to utilize a different biometric supported by their device such as face or voice recognition. Fortunately, facial recognition technology is increasingly common as well, as it exists as Face ID on Apple products, and comes standard on nearly all iPhones since 2018 and as Facial Recognition on Android products, with simpler versions of contemporary facial recognition appearing as early as 2011 [96, 97]. As these systems become more robust, popular, and usable, it will become even easier to authenticate one's identity into their device and thereby easier to use and integrate FIDO into existing systems. In particular, authentication platforms like FIDO allow remote servers to support all such authenticators without implementing specific functionality for each biometric. Consequently, FIDO allows users to authenticate using the factor that is best suited to their preferences and ability and has the potential to impact the majority of online users.

5.2 Privacy as a Product

Privacy and other non-security threats to individuals are growing as society becomes more dependent on digital technologies. Many sites, including the majority of popular social media networks, collect user data which is often sold in advertising products. These sites collect this data

by linking activities to a unique user identity. This identity is established when a user logs in to a website using their username and password. The site verifies that the individual requesting to make actions is the entity associated with the provided username by verifying that the password provided is correct. In this scheme, the remote service is responsible for both the validation of the user gesture and the authentication of that user to a service. FIDO separates the process of validation and authentication. In the FIDO protocol, validation is a local process that occurs on the trusted device owned by the user. The user permits this trusted device to perform authentication only after the successful validation of a user gesture. This trust is achieved using the public-private key pair that is stored on the authenticator and used to answer challenges in the WebAuthn protocol. Local validation is significant since, in traditional authentication schemes, the digital identity of a user is completely controlled by a remote website. In this system, a user's identity is established and then controlled by each service. FIDO, through local verification, removes the ownership and control of a user's identity from the services and transfers it fully to the user. In a reality where service providers often share the data they collect with other corporations and sell recorded user actions as a product, FIDO offers a significant shift in agency since many sites that would otherwise have control over one's digital identity that are not using that identity in the best interest of the user can no longer freely use it as they wish.

6 Appendices

6.1 FIDO Usability Study Questionnaire

FIDO Authentication

Please answer the following questions on a 1-5 scale (1 = strongly disagree, 5 = strongly agree):

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

Password Authentication

Please answer the following questions on a 1-5 scale (1 = strongly disagree, 5 = strongly agree):

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

Short Answer Questions

The following questions will only be administered during the second appointment with participants

Please answer the following questions in a few sentences:

1. Would you prefer to use text-based passwords or FIDO authentication in the future? Why?
2. Would you be willing to trust a website whose only authentication method is with FIDO? Why?
3. What did you like about using FIDO authentication?
4. What did you not like about using FIDO authentication?

6.2 Strengths & Weaknesses of Top Authentication Alternatives

Scheme	Fingerprint Recognition	Voice Recognition	Graphical Passwords
Accuracy	+ Widely used to justify biometric quality indices [35]	- Requires a wide variety of training data (whispering vs. yelling, background noise vs. silence, one speaker vs. multiple speakers, close vs. far) [49]	- Recall-based implementations suffer from initially low false-negative rates but improve with user acclimation [98] - Sensitive to changes in acceptability tolerance [98] - Accuracy is reduced by including stroke order as a factor [98]
Attack Vulnerabilities	- Physical fingerprint spoofing attacks [38]	- Human mimicry [99] - Hidden voice/obfuscated commands [46] - Susceptible to being overheard in phrase-specific implementations	- Shoulder surfing [100] - Smudge attacks [101] - Hot-spot exploitation (predicting likely user passwords using a given background image) [102]
Usability Problems	- Individuals with low dexterity could have trouble using a fingerprint scanner - A cut/wound to the finger can lock you out - Concerns over theft of biometric data	- Non-negligible false-negative rates [46] - Speaker recognition requires a large amount of training to ensure accuracy [46]	- Immaturity of graphical passwords means that there is a lack of research in public trust of the scheme - More accessible to certain demographics than others [10, 11]
Cost	- Requires a fingerprint sensor and fingerprint recognition software	- Requires a microphone and voice recognition software	- Requires more memory and data storage than passwords [103]
Advantages Over Passwords	+ Mature + Appropriate since most devices are operated using one's fingers + Cheap + Fingerprints are suitably unique	+ Voice is unique to each user, but different users can have the same password [49]	+ More convenient than traditional passwords [11] + Easier to learn [11] + More efficient and easier for human memory [103]

7 Bibliography

[1] Individuals using the internet (% of population) - united states | data. [Online]. Available: https://data.worldbank.org/indicator/IT.NET.USER.ZS?end=2017&locations=US&name_desc=false&start=1990

[2] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *2012 IEEE Symposium on Security and Privacy*, pp. 553–567. [Online]. Available: https://ieeexplore.ieee.org/document/6234436

[3] R. McMillan, "The world's first computer password? it was useless too." [Online]. Available: https://www.wired.com/2012/01/computer-password/

[4] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: empirical results," vol. 2, no. 5, pp. 25–31. [Online]. Available: https://ieeexplore.ieee.org/document/1341406

[5] L. Tam, M. Glassman, and M. Vandenwauver, "The psychology of password management: a tradeoff between security and convenience," vol. 29, no. 3, pp. 233–244. [Online]. Available: https://doi.org/10.1080/01449290903121386

[6] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the 16th international conference on World Wide Web - WWW '07*. ACM Press, p. 657. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1242572.1242661

[7] D. V. Bailey, M. Dürmuth, and C. Paar, "Statistics on password re-use and adaptive strength for financial accounts," in *Security and Cryptography for Networks*, M. Abdalla and R. De Prisco, Eds. Springer International Publishing, vol. 8642, pp. 218–235. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10879-7_13

[8] R. Zhao and C. Yue, "Toward a secure and usable cloud-based password manager for web browsers," vol. 46, pp. 32–47. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814001059

[9] D. Chadwick, "Federated identity management," in *IEEE Computer - COMPUTER*, vol. 38, pp. 96–120. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-03829-7_3

[10] D. Schwab, L. ALharbi, O. Nichols, and L. Yang, "Picture PassDoodle: Usability study," in *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*, pp. 293–298.

[11] S. Wiedenbeck, J. Waters, J.-C. Birget, A. Brodskiy, and N. Memon, "PassPoints: Design and longitudinal evaluation of a graphical password system," vol. 63, no. 1, pp. 102–127. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1071581905000625

[12] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: A survey," vol. 2, no. 1, p. 1. [Online]. Available: http://www.mdpi.com/2410-387X/2/1/1

[13] S. Rane, Y. Wang, S. Draper, and P. Ishwar. Secure biometrics: Concepts, authentication architectures, and challenges - IEEE journals & magazine. [Online]. Available: https://ieeexplore.ieee.org/document/6582729

[14] FIDO alliance - open authentication standards more secure than passwords. [Online]. Available: https://fidoalliance.org/

[15] P. McDaniel, "Authentication." Pennsylvania State University. [Online]. Available: https://pdfs.semanticscholar.org/dd3f/ba9b89a1f912a49463f2c9c28d9d726331a3.pdf

[16] J.-J. Kim and S.-P. Hong, "A method of risk assessment for multi-factor authentication," vol. 7, no. 1, pp. 187–198. [Online]. Available: http://www.koreascience.or.kr/article/JAKO201113753748218.page

[17] Polybius. Histories, book 6, daily orders and watchwords. [Online]. Available: http://www.perseus.tufts.edu/hopper/text?doc=Perseus%3Atext%3A1999.01.0234%3Abook%3D6%3Achapter%3D34

[18] B. Ives, K. R. Walsh, and H. Schneider, "The domino effect of password reuse," vol. 47, no. 4, pp. 75–78. [Online]. Available: http://portal.acm.org/citation.cfm?doid=975817.975820

[19] W. Ma, D. Kleeman, D. Tran, and J. Campbell, "A conceptual framework for assessing password quality," vol. 7, no. 1, pp. 179–185. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.3266&rep=rep1&type=pdf

[20] P. A. Grassi, J. L. Fenton, E. M. Newton, R. A. Perlner, A. R. Regenscheid, W. E. Burr, J. P. Richer, N. B. Lefkovitz, J. M. Danker, Y.-Y. Choong, K. K. Greene, and M. F. Theofanos, "Digital identity guidelines: authentication and lifecycle management," pp. NIST SP 800–63b. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf

[21] N. Woods and M. Siponen, "Too many passwords? how understanding our memory can increase password memorability," vol. 111, pp. 36–48. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581917301581

[22] M. Abadi, M. A. Lomas, and R. Needham, "Strengthening passwords." [Online]. Available: https://pdfs.semanticscholar.org/b855/6c4b4276bbf5451cc0875731b0b0574f0162.pdf

[23] M. M. Kassim and A. Sujitha, "ProcurePass: A user authentication protocol to resist password stealing and password reuse attack," in *2013 International Symposium on Computational and Business Intelligence*, pp. 31–34.

[24] A. Shamir, "How to share a secret," vol. 22, no. 11, pp. 612–613. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[25] Product users. [Online]. Available: https://www.shibboleth.net/community/ourusers/

[26] B. Leiba, "OAuth web authorization protocol," vol. 16, no. 1, pp. 74–77.

[27] S.-T. Sun and K. Beznosov, "The devil is in the (implementation) details: An empirical analysis of OAuth SSO systems," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. ACM, pp. 378–390, event-place: Raleigh, North Carolina, USA. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382238

[28] E. Maler and D. Reed, "The venn of identity: Options and issues in federated identity management," vol. 6, no. 2, pp. 16–23. [Online]. Available: http://ieeexplore.ieee.org/document/4489845/

[29] J. Colnago, S. Devlin, M. Oates, C. Swoopes, L. Bauer, L. Cranor, and N. Christin, ""it's not actually that horrible": Exploring adoption of two-factor authentication at a university," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, pp. 1–11. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3173574.3174030

[30] A. P. Sabzevar and A. Stavrou, "Universal multi-factor authentication using graphical passwords," in *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, pp. 625–632.

[31] N. Ratha and R. Bolle, *Automatic Fingerprint Recognition Systems*. Springer Science & Business Media, google-Books-ID: KdiQC0OtuCIC.

[32] W. F. Leung, S. H. Leung, W. H. Lau, and A. Luk, "Fingerprint recognition using neural network," in *Neural Networks for Signal Processing Proceedings of the 1991 IEEE Workshop*, pp. 226–235.

[33] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. Springer Science & Business Media, google-Books-ID: 1Wpx25D8qOwC.

[34] E. Tabassi, "Fingerprint image quality," p. 72.

[35] Y. Chen, S. C. Dass, and A. K. Jain, "Fingerprint quality indices for predicting authentication performance," in *Audio- and Video-Based Biometric Person Authentication*, T. Kanade, A. Jain, and N. K. Ratha, Eds. Springer Berlin Heidelberg, vol. 3546, pp. 160–170. [Online]. Available: http://link.springer.com/10.1007/11527923_17

[36] T. N. Tan and H. Lee, "High-secure fingerprint authentication system using ring-LWE cryptography," vol. 7, pp. 23 379–23 387.

[37] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The impact of quantum computing on present cryptography," vol. 9, no. 3. [Online]. Available: http://arxiv.org/abs/1804.00200

[38] T. v. d. Putte and J. Keuning, "Biometrical Fingerprint Recognition: Don't get your Fingers Burned," in *Smart Card Research and Advanced Applications: IFIP TC8 / WG8.8 Fourth Working Conference on Smart Card Research and Advanced Applications September 20–22, 2000, Bristol, United Kingdom*, ser. IFIP — The International Federation for Information Processing, J. Domingo-Ferrer, D. Chan, and A. Watson, Eds. Boston, MA: Springer US, 2000, pp. 289–303. [Online]. Available: https://doi.org/10.1007/978-0-387-35528-3_17

[39] B. Nath, F. Reynolds, and R. Want, "RFID technology and applications," vol. 5, no. 1, pp. 22–24.

[40] K. Michael and M. G. Michael, "Microchip implants for humans as unique identifiers: a case study on VeriChip," p. 6.

[41] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong authentication for RFID systems using the AES algorithm," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Springer Berlin Heidelberg, vol. 3156, pp. 357–370. [Online]. Available: http://link.springer.com/10.1007/978-3-540-28632-5_26

[42] C. Perakslis and R. Wolk, "Social acceptance of RFID as a biometric security method," vol. 25, no. 3, pp. 34–42.

[43] E. Spinella, "Information security reading room biometric scanning technologies : Finger , facial and retinal scanning."

[44] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," vol. 16, no. 4, pp. 351–359. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X9900059X

[45] R. Joyce and G. Gupta, "Identity authentication based on keystroke latencies," vol. 33, no. 2. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.586.5114&rep=rep1&type=pdf

[46] P. Mishra, T. Vaidya, Y. Zhang, and M. Sherr, "Hidden voice commands," p. 18.

[47] J.-F. Bonastre, F. Bimbot, L.-J. Boe, J. P. Campbell, D. A. Reynolds, and I. Magrin-Chagnolleau, "Person authentication by voice: A need for caution," p. 4.

[48] L. Muda, M. Begam, and I. Elamvazuthi, "Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques." [Online]. Available: http://arxiv.org/abs/1003.4083

[49] A. Boles and P. Rad, "Voice biometrics: Deep learning-based voiceprint authentication system," in *2017 12th System of Systems Engineering Conference (SoSE)*, pp. 1–6.

[50] J. Ahmad, M. Fiaz, S.-i. Kwon, M. Sodanil, B. Vo, and S. W. Baik, "Gender identification using MFCC for telephone applications – a comparative study," vol. 3, no. 5. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1601/1601.01577.pdf

[51] E. R. Isaac, S. Elias, S. Rajagopalan, and K. Easwarakumar, "Trait of gait: A survey on gait biometrics." [Online]. Available: https://arxiv.org/pdf/1903.10744.pdf

[52] D. Sunehra, "Fingerprint based biometric ATM authentication system," vol. 3, no. 11, pp. 22–28. [Online]. Available: http://www.ijeijournal.com/papers/Vol.3-Iss.11/D030112228.pdf

[53] Y. Wang, C. Hahn, and K. Sutrave, "Mobile payment security, threats, and challenges," in *2016 Second International Conference on Mobile and Secure Services (MobiSecServ)*, pp. 1–5.

[54] Y.-P. Yu, S. Wong, and M. B. Hoffberg, "Web-based biometric authentication system and method," patentus 5 930 804A. [Online]. Available: https://patents.google.com/patent/US5930804A/en

[55] About face ID advanced technology. [Online]. Available: https://support.apple.com/en-us/HT208108

[56] I. Buciu and A. Gacsadi, "Biometrics systems and technologies: A survey," vol. 11, no. 3, pp. 315–330. [Online]. Available: https://core.ac.uk/download/pdf/236052448.pdf

[57] FIDO2: Moving the world beyond passwords using WebAuthn & CTAP. [Online]. Available: https://fidoalliance.org/fido2/

[58] O. Pereira, A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo, J. Garcia-Alfaro, F. Rochet, and C. Wiedling, "Formal analysis of the FIDO 1.x protocol," vol. 10723, pp. 68–82. [Online]. Available: http://link.springer.com/10.1007/978-3-319-75650-9_5

[59] J. Hodges, J. Jones, M. B. Jones, A. Kumar, E. Lundberg, D. Balfanz, V. Bharadwaj, A. Birgisson, A. Czeskis, H. L. V. Gong, A. Liao, and R. Lindemann. Web authentication: An API for accessing public key credentials level 2. [Online]. Available: https://www.w3.org/TR/webauthn/

[60] M. Barbosa, A. Boldyreva, S. Chen, and B. Warinschi, "Provable security analysis of FIDO2." [Online]. Available: http://eprint.iacr.org/2020/756

[61] H. Feng, H. Li, X. Pan, and Z. Zhao, "A formal analysis of the FIDO UAF protocol," in *NDSS*.

[62] S. Eskandarian, T. K. Sethi, V. Subbiah, M. Backes, G. Pellegrino, D. Boneh, J. Cogan, S. Birnbaum, P. C. W. Brandon, D. Franke, F. Fraser, G. Garcia, E. Gong, and H. T. Nguyen, "Fidelius: Protecting user secrets from compromised browsers," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, pp. 264–280. [Online]. Available: https://ieeexplore.ieee.org/document/8835331/

[63] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 57–64.

[64] C. Brand, A. Czeskis, J. Ehrensvärd, M. B. Jones, A. Kumar, R. Lindemann, A. Powers, J. Verrept, M. Antoine, A. Birgisson, V. Bharadwaj, and M. J. Ploch. Client to authenticator protocol (CTAP). [Online]. Available: https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html

[65] Y. Zhang, X. Wang, Z. Zhao, and H. Li, "Secure display for FIDO transaction confirmation," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '18. Association for Computing Machinery, pp. 155–157. [Online]. Available: http://doi.org/10.1145/3176258.3176946

[66] I. Guirat and H. Halpin. Formal verification of the w3c web authentication protocol. [Online]. Available: https://dl-acm-org.proxy-um.researchport.umd.edu/doi/epdf/10.1145/3190619.3190640

[67] S. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks," in *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Comput. Soc. Press, pp. 72–84. [Online]. Available: http://ieeexplore.ieee.org/document/213269/

[68] T. Wu, "The secure remote password protocol," in *In Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, pp. 97–111.

[69] B. Haase and B. Labrique, "AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT," pp. 1–48. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7384

[70] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security vulnerabilities of SGX and countermeasures: A survey," vol. 54, no. 6, pp. 1–36. [Online]. Available: https://dl.acm.org/doi/10.1145/3456631

[71] A. S. f. P. Affairs. System usability scale (SUS). Publisher: Department of Health and Human Services. [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[72] S. G. Lyastani, M. Schilling, M. Neumayr, M. Backes, and S. Bugiel, "Is FIDO2 the kingslayer of user authentication? a comparative usability study of FIDO2 passwordless authentication," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 268–285, ISSN: 2375-1207.

[73] S. Ciolino, S. Parkin, and P. Dunphy, "Of two minds about two-factor: Understanding everyday FIDO u2f usability through device comparison and experience sampling," pp. 339–356. [Online]. Available: https://www.usenix.org/conference/soups2019/presentation/ciolino

[74] J. Brooke, "SUS: a retrospective," vol. 8, pp. 29–40.

[75] W. Oogami, H. Gomi, S. Yamaguchi, S. Yamanaka, and T. Higurashi. Observation study on usability challenges for fingerprint authentication using WebAuthn-enabled android smartphones. [Online]. Available: https://www.semanticscholar.org/paper/Observation-Study-on-Usability-Challenges-for-Using-Oogami-Gomi/28715fe0e8ceac49675ba5237b4ca5b0624c2219

[76] K. Owens, O. Anise, A. Krauss, and B. Ur, "User perceptions of the usability and security of smartphones as {FIDO2} roaming authenticators," pp. 57–76. [Online]. Available: https://www.usenix.org/conference/soups2021/presentation/owens

[77] G. Charness, U. Gneezy, and M. A. Kuhn, "Experimental methods: Between-subject and within-subject design," vol. 81, no. 1, pp. 1–8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167268111002289

[78] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation In Industry*. CRC Press, num Pages: 6.

[79] Demographics of internet and home broadband usage in the united states. [Online]. Available: https://www.pewresearch.org/internet/fact-sheet/internet-broadband/

[80] M. Roser, H. Ritchie, and E. Ortiz-Ospina, "Internet." [Online]. Available: https://ourworldindata.org/internet

[81] J. Nielsen and T. Landauer. A mathematical model of the finding of usability problems. [Online]. Available: https://dl.acm.org/doi/10.1145/169059.169166

[82] J. Nielsen. Why you only need to test with 5 users. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[83] S. Kemp. Digital 2020: Global digital overview. [Online]. Available: https://datareportal.com/reports/digital-2020-global-digital-overview

[84] S. Williams. Average person has 100 passwords - study. [Online]. Available: https://securitybrief.co.nz/story/average-person-has-100-passwords-study

[85] Flask. [Online]. Available: https://palletsprojects.com/p/flask/

[86] "py_webauthn," original-date: 2017-11-10T18:02:28Z. [Online]. Available: https://github.com/duo-labs/py_webauthn

[87] M. Otto, J. Thornton, and B. Contributors. Bootstrap. [Online]. Available: https://getbootstrap.com/

[88] Jinja — jinja documentation (3.0.x). [Online]. Available: https://jinja.palletsprojects.com/en/3.0.x/

[89] MariaDB foundation. [Online]. Available: https://mariadb.org/

[90] Caddy 2 - the ultimate server with automatic HTTPS. [Online]. Available: https://caddyserver.com/

[91] Gunicorn - python WSGI HTTP server for UNIX. [Online]. Available: https://gunicorn.org/

[92] Empowering app development for developers | docker. [Online]. Available: https://www.docker.com/

[93] Sultan, Ahmad. Report: Improving cybersecurity awareness in underserved populations - CLTC UC berkeley center for long-term cybersecurity. [Online]. Available: https://cltc.berkeley.edu/2019/04/15/improving-cybersecurity-awareness-in-underserved-populations/

[94] S. Majumder and M. J. Deen, "Smartphone sensors for health monitoring and diagnosis," vol. 19, no. 9, p. 2164. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6539461/

[95] Fingerprints. Fingerprints - 2019 report. [Online]. Available: https://www.fingerprints.com/uploads/nasdaq-v2/reports/2020/04/fingerprints-report-202004290700-fingerprints-annual-report-2019.pdf

[96] iPhone and iPad models that support face ID. [Online]. Available: https://support.apple.com/en-us/HT209183

[97] Facial recognition on smartphones: Is it secure and should you use it? [Online]. Available: https://www.androidauthority.com/face-unlock-smartphones-3043993/

[98] M. Jali, S. Furnell, and P. Dowland, "Quantifying the Effect of Graphical Password Guidelines for Better Security," in *Future Challenges in Security and Privacy for Academia and Industry*, ser. IFIP Advances in Information and Communication Technology, J. Camenisch, S. Fischer-Hübner, Y. Murayama, A. Portmann, and C. Rieder, Eds. Berlin, Heidelberg: Springer, 2011, pp. 80–91.

[99] Y. W. Lau, M. Wagner, and D. Tran, "Vulnerability of speaker verification to voice mimicking," in *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, Oct. 2004, pp. 145–148.

[100] A. H. Lashkari and S. Farmand, "Shoulder surfing attack in graphical password authentication," vol. 6, no. 2, p. 10.

[101] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," p. 10.

[102] J. Thorpe and P. C. van Oorschot, "Human-seeded attacks and exploiting hot-spots in graphical passwords," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, ser. SS'07. USA: USENIX Association, Aug. 2007, pp. 1–16. [Online]. Available: https://dl.acm.org/doi/10.5555/1362903.1362911

[103] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin, "THE DESIGN AND ANALYSIS OF GRAPHICAL PASSWORDS," p. 15.