Distributed Single Password Protocols

by

Devriş İşler

A Dissertation Submitted to the

Graduate School of Sciences and Engineering

in Partial Fulfillment of the Requirements for

the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ
UNIVERSITY

July 12, 2018

Distributed Single Password Protocols

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Devriş İşler

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Assist. Prof. Alptekin Küpçü (Advisor)

Assist. Prof. Aykut Coşkun

Prof. Dr. Kemal Bıçakcı

Date:

To my beloved family

ABSTRACT

Passwords are the most widely used factor in various areas such as secret sharing, key establishment, and user authentication. Single password protocols are proposed (starting with [Belenkiy et al., 2011]) to overcome the challenges of traditional password protocols and provide provable security against offline dictionary, man-in-the-middle, phishing, and honeypot attacks. While they ensure provable security, they allow a user securely to use a single *low-entropy human memorable* password for all her accounts. They achieve this with the help of a cloud or mobile storage device. However, an attacker corrupting both the login server and storage can mount an offline dictionary attack on user's single password.

In this thesis, we introduce a framework for distributed single password protocols (DiSPP) that analyses existing protocols, improves upon them regarding novel constructions and distributed schemes, and allows exploiting alternative cryptographic primitives to obtain secure distributed single password protocols with various trade-offs. Previous single password solutions can be instantiated as part of our framework. We further introduce a secure DiSPP instantiation derived from our framework enforcing the adversary to corrupt several cloud and mobile storage devices in addition to the login server in order to perform a successful offline dictionary attack. We also provide a comparative analysis of different solutions derived from our framework. We define ideal and real world indistinguishability for DiSPP, and formally prove security of our proposed solution via ideal-real simulation. Finally, we implement two DiSPP instantiations (based on mobile and cloud) and assess their usability with their counterparts (traditional password and two-factor authentication). We conclude that DiSPP systems overall constitute a usable alternative to existing solutions that do not provide offline dictionary attack protection.

ÖZETÇE

Parolalar, çevrimiçi kullanıcı kimlik doğrulamada kullanılan en yaygın yöntemdir. Geleneksel parola sistemlerinde, bir kullanıcı bir giriş sunucusuna kimlik doğrulamasını düşük dağıntılı hatırlanması kolay bir parola ile yapar. Maalesef bu alandaki mevcut çözümler ne oltalama (phishing), aradaki adam (man-in-the-middle), balküpü (honeypot) ve çevrimdışı sözlük saldırısı gibi saldırılara karşı güvenli ne de portatiftirler. Dağıtılmış tek parolalı sistemler yukarıda bahsi geçen saldırılara karşı güvensiz olan geleneksel parola protokollerinin zayıflıklarını gidermek amacıyla önerilmiştir. Dağıtılmış tek parolalı sistemler bu saldırılara karşı güvenliği normal sistemlere ek bir depolama sağlayıcısı ekleyerek sağlar (örneğin; bir bulut depolama veya taşınabilir bir mobil cihaz). Bu sistemler teorik olarak güvenlik ispatı sunmalarının yanı sıra, kullanıcının bütün hesapları için düşük dağıntılı tek bir parolayı güvenli bir şekilde kullanmasına olanak sağlarlar. Bu tezde, güvenli dağıtılmış tek parolalı sistemler için genel bir yapı (çerçeve) sunulmuştur. Bu yapıdan ve bu yapıda sunulan özelliklere sahip farklı kriptografik taslaklardan faydalanarak güvenli dağıtılmış tek parolaları sistem örnekleri sunulmuş ve bu örneklerin performans değerlendirmeleri sayısal olarak ortaya konmuştur. Genel yapının teorik ispatı için ideal ve reel dünya güvenlik tanımlamarını yapılmış ve bu yapıdan oluşturulan bir sistem örneği tanıtılmıştır. Bu örneğin teorik ispatını da bu ideal-reel simulasyon ile kanıtlanmıştır. Son olarak, sunduğumuz örneğin ve daha önceden literatürde sunulmuş ancak kullanıcı deneyi yapılmamış olan bir dağıtılmış tek parolalı sisteminin kullanıcı deney çalışmaları gerçekleştirilmiştir. Bu sistemlerin kullanıcı deneylerini günümüzde yaygın olarak kullanılan ve bir çok saldırıya karşı güvensiz olan geleneksel parola ve iki-faktörlü kimlik doğrulama sistemleriyle karşılaştırılmış ve dağıtılmış tek parolalı sistemlerin bu alternatiflerine göre daha kullanılabilir olduğu gözlemlenmiştir.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Assist. Prof. Alptekin Küpçü for the continuous support of my MSc. study and research, for his patience, motivation, enthusiasm, and immense knowledge. It has been an honour to be his student. I am thankful to my committee members Asst. Prof. Dr. Aykut Coskun and Prof. Dr. Kemal Bıçakcı for their time and support during the validation of this thesis.

I am very grateful to Elif Dilek Salık and Merve Saimler for being my friends and all the valuable time we spent together. I like to thank to my lovely crypto group members and friends Sanaz Taheri and Yahya Hassanzadeh whom I enjoy to spend time together and discuss interesting research topics. I also like to thank to Yeşim Dündar, İpek Kızıl, Sepeedeh Shahbeigi, Adnan Hassan, Saliha Akbaş, Ahsan Ali, Beakal Gizachew Assefa, Seyhan Uçar, Buket Yüksel, and all the other friends and officemates for their support, understanding, and assistance, throughout the duration of my studies.

I would like to thank the pearls of my life, my family. It is not possible to express the value of the family. My mother, Hatice İşler and my father, Ercan İşler are basically the reasons why I am the person I am today. My profound gratitude to my parents and siblings for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. I can not express how much I am grateful for their faith in me. They believed in me and helped me to follow my dreams.

I also thank Arjen Kılıç and İlker Kadir Öztürk for their efforts on implementation of the user studies and acknowledge the support of TÜBİTAK (the Scientific and Technological Research Council of Turkey) under the project numbers 114E487 and

115E766, European Union COST Action IC1306, and the Royal Society of UK Newton Advanced Fellowship NA140464.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Passwords are used in many different contexts such as authentication, key exchange, and sensitive information storage. In password-based authentication, a user registers with a server using her low-entropy password. The server stores the user's information (e.g. $< username, hash(password) >$). Later on, the user authenticates herself to the server whenever she wants to get a service. In password authenticated key exchange, following registration, a user and a server wish to establish a session key for a secure and authenticated channel [Bellovin and Merritt, 1992, Boyen, 2009b, Boyko et al., 2000, Katz et al., 2001, MacKenzie et al., 2002]. In password protected sensitive information storage, a user stores her sensitive information (e.g. credentials) among a server or multiple servers using her password [Boyen, 2009a, Camenisch et al., 2015a, Camenisch et al., 2014, Jarecki et al., 2014, Jarecki et al., 2017]. Whenever she wants to reconstruct the sensitive information, she has to convince the server(s) that she is the legitimate user holding the correct password.

However, passwords are vulnerable to many prevalent online and offline attacks such as *phishing*, *man-in-the-middle*, *honeypot*, and *offline dictionary* attacks. The damage of a successful attack increases with password reuse, which is common in practice [Florencio and Herley, 2007a]. This brings many issues, where an attacker compromising user's password can use it to gain access to the services on behalf of the user. Such an attacker may be a malicious server (as in phishing, honeypot, or man-in-the-middle attacks), or hackers obtaining the server database and mounting offline dictionary attacks. Indeed, such attacks are very prevalent, and recent studies even propose improved offline dictionary attacks [Tatli, 2015].

Single password protocols (starting with [Acar et al., 2013] with their patent application dating 2010 [Belenkiy et al., 2011], and [Jarecki et al., 2016a]) are proposed to provide *provable security* against the aforementioned attacks that traditional password protocols are vulnerable to. These protocols enable users to use a single password for all their accounts securely, where users do not store any information locally. They achieve this with the help of a cloud or mobile storage device.

The general idea of a distributed single password protocol (DiSPP) is to create a high entropy secret *independent* of the user's password and registering a verification information based on this secret with the login server(s). Later, the user stores the secret among storage providers (e.g. personal devices, online storage providers) using her password. Whenever the user wants to authenticate herself (implicitly or explicitly) to the login servers, she reconstructs the secret and authenticates with the server accordingly.

In this thesis, we firstly present a framework for *distributed* single password protocols that can employ possibly *more than one* storage provider (any combination of cloud and mobile devices) and *more than one* login server. Our framework consists of four phases: registration and authentication (between a user and login servers) and secret storage and retrieval (between a user and storage providers). Secondly, we discuss possible cryptographic building blocks that can be employed in these phases so that the combination of these building blocks constitute a secure DiSPP solution. In our framework, we employ a total of n_{stor} storage providers and n_{ls} login servers with a threshold $1 \leq t_{stor} \leq n_{stor}$ for the storage providers and a threshold $1 \leq t_{ls} \leq n_{ls}$ for the login servers. This setting serves two purposes. Firstly, for an adversary to be able to successfully mount an offline dictionary attack, he must corrupt t_{ls}-many login servers in addition to t_{stor}-many storage providers. Secondly, to login, the user must access t_{stor} storage providers out of n_{stor}; thus availability can be balanced against security easily by setting these parameters. While the underlying techniques are different, in terms of security, all previous solutions correspond to setting $t_{ls} = n_{ls} = 1$ and $t_{stor} = n_{stor} = 1$.

Finally, we conduct user studies on our proposed DiSPP solution which is a cloud-based DiSPP instance against the traditional approach in a daily use scenario, and mobile-based DiSPP instance (based on [Acar et al., 2013]) against two-factor authentication in an online banking scenario. Quantitative and qualitative results support that both DiSPP solutions have usability and security advantages compared to their counterparts. Based on the feedback reported by the participants, we suggest that cloud-based DiSPP solutions should be deployed for daily use, where users wish to login to a site frequently, and mobile-based DiSPP solutions should be deployed for online banking type of settings, where more complicated solutions are expected (at least seemingly more complicated, regardless of the underlying cryptography). Observations also indicate that there is potentially a trade-off between usability and perceived security, which is worth exploring as future work.

1.1 Contributions

- (Chapter 4) For the first time, we present a framework for *distributed* single password protocols that can employ possibly *more than one* storage provider (any combination of cloud and mobile devices) and *more than one* login server. Our framework for distributed single password protocols (DiSPP) analyzes existing protocols, improves upon them regarding novel constructions and distributed schemes, and allows exploiting alternative cryptographic primitives to obtain secure distributed single password protocols with various trade-offs. We show how the existing solutions can be realized by our framework.

- (Chapter 4) Until now, game-based proof techniques are used to prove such systems. For the first time in this thesis, we formally define ideal and real world definitions for security of DiSPP framework.

- (Chapter 5) We present a DiSPP instantiation derived from our framework enhancing existing DiSPP solutions. Our proposed solution is secure against offline dictionary attack, phishing and man-in-the- middle attacks during authentication, after a secure registration, and honeypot attacks during registration and authentication. Our construction does not require any change at the login server side. While other solutions do not give an access control to the user (e.g. mobile device must participate for authentication), our DiSPP instantiation can work with a variety of storage providers and allow a user to define her own access control for authentication.

- (Chapter 5) We formally prove the security of our DiSPP instantiation via ideal-real simulation, showing impossibility of offline dictionary attacks. We also discuss how our DiSPP instantiation provides security against malicious login servers (e.g. phishing, man-in-the-middle).

- (Chapter 5) We present performance evaluation numerically, showing that our techniques are easily applicable with today's hardware. We discuss efficiency of different DiSPP solutions based on different cryptographic primitives.

- (Chapter 6) We implement our proposed DiSPP solution and an existing DiSPP instance [Acar et al., 2013] and conduct usability studies of these two DiSPP solutions against two commonly-employed authentication systems: traditional username-password authentication and two-factor authentication.

- (Chapter 6) We provide our findings (based on both numerical and anecdotal data) on user perspective against the idea of using one single password securely. We discuss in what type of settings these DiSPP solutions provide better usability. Overall, our findings show that DiSPPs provide better usability and security.

Chapter 2

RELATED WORK

2.1 *Related Work on DiSPP*

Traditional password-based authentication takes place between two parties (a *user* and a *login server*). However, [Boyen, 2009a] showed that any password-based authentication between a user and a login server is vulnerable to an *offline dictionary attack* by the login server (or hackers obtaining its database). [Tatli, 2015] improved offline dictionary attacks on password hashes to find some additional passwords assumed to be strong and complex.

[Ford and Kaliski, 2000] suggest a *password hardening* protocol where the user, holding a weak-password *pwd*, interacts with one or more servers by blinding the password to create a secret credential (to decrypt, or authenticate herself to a login server, etc.) from shares received by the hardening server(s) (which is like storage providers). The hardening server(s) cannot learn anything about the password and the secret unless all of them collude. During the authentication, for each login server, the user runs the password hardening protocol to retrieve the same secret as in the registration by communicating with hardening servers. The solution proposed do not have a formal proof and requires interaction with all of the servers to be able to reconstruct the secret. [MacKenzie et al., 2002] propose a threshold PAKE where the password is secure unless threshold-many servers collude. [MacKenzie et al., 2002] requires servers to know each other. [Juels and Rivest, 2013] proposes to create a password file storing false passwords called *honeywords* per user account. In case the adversary steals the password file, and mistakenly employs a honeyword, the system is alerted.

[Mannan and van Oorschot, 2007] propose to secure user's password from untrusted user computer (malicious browser) assuming the server holds the user password. [Camenisch et al., 2015b] distribute the password verification over multiple servers to secure the password against server compromise where the server keeps the hash of username and password $hash(username||password)$. PwdHash [Ross et al., 2005] produce a different password for each login server by simply computing the hash of the password and login server domain where the hash is a pseuodorandom function of the domain keyed with the user password $H(password, domain) = PRF_{password}(domain)$ and the server stores the hash value. The discussed solutions do not provide security against offline dictionary attack in case the server database is compromised. Increasing the number of parties by adding storage provider(s) is one way to help prevent offline dictionary attacks.

[Acar et al., 2013] (with their patent application dating 2010) present the first provably secure single password authentication protocols where the user employs a cloud or mobile storage provider to keep her secret to prevent offline dictionary attacks. The user's password is secure against offline dictionary attacks unless the storage provider and the server are colluding. Their mobile device based solution inputs the password to the device, and hence provides security against *malware* on the public terminal. They provide security against *phishing* indirectly because the user identifier used at the storage provider depends on the server name. Since the phishing site name is *different* from the actual login server name, the retrieval of the user secret fails. Our main differences are to enable a fully secure threshold construction for the first time, without requiring any changes at the login server, making our solution much easier to deploy.

Following [Acar et al., 2013], [Jarecki et al., 2016a] provided a device enhanced password authenticated key exchange protocol employing a mobile device storage. Similar to [Acar et al., 2013], their protocol is secure against offline dictionary attacks assuming the login server and the mobile device are not colluding. They provide a recovery procedure in case the device is lost. They leave threshold authentication as future work, which is what we achieve.

[Bicakci et al., 2011a] discuss briefly a single password solution employing a storage provider for unique blind signatures (similar to [Acar et al., 2013]), but they neither delve into the details of their solution nor present a security proof. Nevertheless, we are influenced by their work in terms of requiring no change at the login server, and achieve to distribute the storage provider for the first time.

2.2 Related Work on Usability Study

We discuss usability studies of various authentication systems below.

2.2.1 Traditional Password Authentication

In these schemes, the username and the output of a deterministic function (e.g., hash) of the password is stored at the server. For authentication, the user types her username and password, and the server compares this information against its database. The user has to remember the corresponding password for each server registered with. The traditional approach is vulnerable to offline dictionary attacks, whereas DiSPP systems ensure security even under server database compromise. The effect of these attacks increases dramatically if the user uses the same password for multiple servers, which is common in practice [Florencio and Herley, 2007b]. [Zviran and Haga, 1993] discusses the traditional password authentication usability. [Zviran and Haga, 1993] provides a quantitative point of reference for the difficulty of remembering random passwords, which is necessary to employ traditional solutions securely.

2.2.2 Two-Factor Authentication

These schemes generally employ any combination of two of what you know (e.g., password), what you have (e.g., token), and who you are (e.g., biometric). Two-factor authentication aims to strengthen the security of traditional password authentication by deploying secondary authentication token (e.g., SMS sent to mobile device). To pass the authentication, the user needs to provide a valid password and token. Despite the widespread use in banking, these systems still suffer from users' negative influence such as reusing the same password. [De Cristofaro et al., 2014] conducted a comparative study of the usability of two-factor authentication technologies, where they found that two-factor authentication is perceived as usable, regardless of motivation or use. [Gunson et al., 2011] showed that two-factor authentication provides more security but lower level of usability. [Sun et al., 2012] proposed a two-factor authentication solution, where they found their system is reliable and usable. [Shirvanian et al., 2014] analyzed different communication channels based two-factor authentication (e.g., QR code, bluetooth), where they concluded when there exists a browser extension and radio interface, their full bandwidth WiFi to WiFi system provides highest security and usability.

2.2.3 Password Managers

In this setting, the user holds a master password to generate server-specific passwords (e.g., $hash(password||domain)$). The generated passwords are usually resistant to dictionary attacks and have high entropies. iPMAN [Bicakci et al., 2011b] (where the master password is created based on objects), LastPass [Corporate, 2016], PwdHash[Ross et al., 2005], Password Multiplier [Halderman et al., 2005] are some examples of password manager type solutions where their usability studies are conducted as well. [Chiasson et al., 2006, Karole et al., 2010, Li et al., 2014] compare the usability of some existing password managers, where they found that users were not comfortable with leaving the control of their passwords to a manager and did not feel that password managers provided greater security. [Li et al., 2014] also suggests that it is still a challenge for password managers to be secure. Indeed, DiSPP solutions remain secure even when the password-protected storage at the cloud or mobile device is compromised.

SPHINX [Shirvanian et al., 2017] is a mobile-phone-based password-manager-type DiSPP solution that uses cryptographic tools to ensure password security against aforementioned attacks. It is efficient, relatively simple to use, and provides better security capabilities compared to many other password managers, such as security in the case of mobile device compromise. Similarly, [Acar et al., 2013] mobile-based DiSPP solution is also secure in such a case, but has a different design goal: SPHINX ensures that the password is input to the client computer and not the mobile device, whereas [Acar et al., 2013] intentionally use the mobile device for inputting the password, rather than the computer (considering a potentially malware-infected public terminal scenario). Since the usability of SPHINX is already examined in [Shirvanian et al., 2017], we studied [Acar et al., 2013] mobile-based DiSPP solution in this paper, which does not require client-side installation (useful for public terminal scenarios).

2.2.4 Other Techniques

Users create secure passwords based on objects (e.g., an image) using an object-based password authentication application (e.g., extension). [Bicakci et al., 2009a, Bicakci et al., 2009b, Mannan and van Oorschot, 2008] are some examples that provided usability studies on object-based passwords. [Mannan and van Oorschot, 2008] points that the user needs to keep the object (e.g., picture) with herself (e.g., via flash driver) to login to a site. In general, [Bicakci et al., 2009b, Mannan and van Oorschot, 2008] showed that creation of the password in object-based systems is easy to accomplish by users.

Password encoding strategies are proposed to make offline dictionary attacks ineffective [Chatterjee et al., 2015]. [Chatterjee et al., 2015] introduces the notion of outputting decoy passwords to an attacker corrupting the manager and tries to decrypt the passwords with a wrong master password. Since the attacker does not have any idea about the correct password, any trial to login with the decoy passwords can be prevented and alerted. However, an attack presented in [Golla et al., 2016] (which is based on distribution of the passwords differences) showed that such a scheme seems to be vulnerable.

Chapter 3

PRELIMINARIES

3.1 Negligible Function

Let $\lambda \in N$ be security parameter. A probabilistic polynomial time (PPT) algorithm A is a probabilistic algorithm taking 1^λ as an input and has running time bounded by a polynomial in λ. We say that a function $negl(\lambda)$ is negligible if for every positive polynomial $poly(\lambda)$ there exists a $\lambda' \in N$ such that $\forall\lambda > \lambda'$ $negl(\lambda)<1/poly(\lambda)$.

3.2 Hash Function

A hash function H is a deterministic function from an arbitrary size input to a fixed size output, denoted $H : \{0,1\}^* \rightarrow \{0,1\}^l$. The hash function is assumed to be *collision resistant* if it is hard to find two different inputs $x \neq y$ that hash to the same output $H(x) = H(y)$.

3.3 Oblivious Pseudorandom Function (OPRF)

A pseudorandom function (PRF) F is a deterministic function that takes two inputs: a secret key k and an input x to compute on, and outputs $F_k(x)$. A function chosen randomly from a PRF family (a PRF with random key k) is secure if it is distinguishable from a random function with the same domain and range with only negligible probability for all PPT distinguishes given oracle access. An Oblivious PRF (OPRF) [Freedman et al., 2005] is a protocol between two parties (*sender* and *receiver*) that securely computes $F_k(x)$ where k and x are the inputs of *sender* and *receiver*, respectively, such that the sender learns *nothing* from the interaction and the receiver learns *only* $F_k(x)$. Threshold Oblivious Pseudorandom Function (TOPRF), introduced by

[Jarecki et al., 2017], is a PRF between multiple senders and a receiver. A Unique Blind Signature [Boldyreva, 2003] (where the signer acts as the sender) can be used as an alternative to OPRF.

3.4 Symmetric Encryption Scheme

It consists of three PPT algorithms: $KeyGen(1^\lambda)$ generates a secret key sk, $Enc_{sk}(msg)$ encrypts the message using the secret key and outputs the ciphertext c, and the decryption algorithm $Dec_{sk}(c)$ uses the secret key sk to decrypt the ciphertext c, and outputs the original message msg. The encryption scheme we use needs to be semantically secure (encryptions of different messages are indistinguishable).

3.5 Message Authentication Code (MAC)

A MAC scheme is a symmetric scheme consisting of three PPT algorithms: $MACKey$ $Gen(1^\lambda)$ generates a key K, $MAC_K(msg)$ generates a MAC tag sig on the message msg using the MAC key K, and $MACVerify_K(sig, msg)$ outputs **accept** if the sig is valid for the given message msg using the MAC key K, and outputs **reject** otherwise. The MAC we employ needs to be secure against adaptive existential forgery attacks [Katz and Lindell, 2014], meaning that even though the adversary adaptively obtains many msg, sig pairs on his choice of messages, he cannot forge a valid sig on a new message.

3.6 Digital Signature

A digital signature scheme is an asymmetric scheme consisting of three PPT algorithms, where $SignKeyGen(1^\lambda)$ generates a secret signing key ssk and a public verification key svk, $Sign_{ssk}(msg)$ generates a signature σ on the message msg using secret signing key ssk, and $SignVerify_{svk}(\sigma, msg)$ outputs *accept* if the given signature σ is a valid signature on msg given the public verification key svk, and outputs *reject* otherwise. The digital signature scheme we employ needs to be secure against adaptive existential forgery attacks (no PPT adversary holding svk can come up with a valid signature on a new message that the oracle has not created a signature on).

3.7 Secret Sharing (SS)

A secret sharing (SS) scheme is a method such that a dealer holding a secret (user in our context) distributes the secret among participants in a way that only the authorized set of participants can reconstruct the secret. The authorized subset varies depending on the secret sharing scheme (e.g. any subset with threshold many participants). The security is that any unauthorized subset of participants cannot reveal any partial information about the secret. For simplicity, let the access structure Γ be a collection of sets of authorized participants who can reconstruct the original secret, and γ denote an authorized set such that $\gamma \in \Gamma$. $\{s_i^\gamma\}$ refers all elements (shares) in an authorized set γ. An SS protocol consists of two PPT algorithms: $\{{s_i}^\gamma\}_{\gamma \in \Gamma} \leftarrow SS(S, \Gamma)$ to create the shares of the secret S, and we use $S \leftarrow SSRecon(\{s_i^\gamma\}, \Gamma)$ to reconstruct the original secret.

3.8 Threshold Secret Sharing (TSS):

In a TSS protocol [Shamir, 1979, Blakley et al., 1979], any subset containing threshold-many participants is an authorized set (i.e. for each γ, we have $|\gamma| = t$). The security is that fewer than threshold-many (t) shares provide no information regarding the original secret.

3.9 Access Controlled Secret Sharing (ACSS):

An ACSS [Ito et al., 1989, Benaloh and Leichter, 1990, Bertilsson and Ingemarsson, 1992, Brickell, 1989, Farràs et al., 2012] is the general form of secret sharing, where Γ is defined explicitly, as long as it is monotonic (i.e. if some $\gamma \in \Gamma$ then for any χ such that $\gamma \subset \chi$ we have $\chi \in \Gamma$).

3.10 Non-Interactive Verifiable Secret Sharing (NIVSS):

NIVSS was introduced by [Pedersen, 1991], where the shareholder can verify whether or not the share received is consistent with other shares without learning the secret itself. The secret sharing and reconstruction algorithms employ some verification information regarding the shares: $\{s_i^\gamma, v_i^\gamma\}_{\gamma\in\Gamma} \leftarrow NIVSS(S,\Gamma)$ to create the shares s_i and corresponding proofs v_i and $S \leftarrow NIVSSRecon(\{s_i^\gamma, v_i^\gamma\},\Gamma)$ to reconstruct and verify the original secret. In addition to secret sharing and reconstruction algorithms, an NIVSS has a third PPT algorithm $0/1 \leftarrow NIVSSVerify(s_i, v_i)$ that is used to verify that the share s_i is a valid share using the proof v_i. Any party holding s_i, v_i can run $NIVSSVerify(s_i, v_i)$ offline.

3.11 Password Protected Secret Sharing (PPSS):

A PPSS [Bagherzandi et al., 2011, Jarecki et al., 2014, Abdalla et al., 2016], [Jarecki et al., 2017, Jarecki et al., , Camenisch et al., 2014] is an SS with the password being involved both in secret sharing and reconstruction steps: $\{s_i^\gamma\}_{\gamma\in\Gamma} \leftarrow PPSS(pwd, S, \Gamma)$ to create the shares of the secret S protected by the password pwd, and $S \leftarrow PPSSRecon(pwd, \{s_i^\gamma\}, \Gamma)$ to reconstruct the original secret. The security is that fewer than threshold-many shares provide no information regarding the original secret and only the legitimate user who knows the password can reconstruct the secret.

3.12 Server-Side Password Protocols

A *server-side password protocol* is a protocol between a user and n_{ls}*-many* login servers. Basically, a user holding a low entropy human-memorable password registers with the servers. During registration, the servers store a verification information based on user password and/or a key. Later on, the user and *threshold-many* (t_{ls}) servers jointly compute a password protocol where the user and each server receive outputs as $output_U$ and $output_i$, respectively. A server-side password protocol has two purposes in the literature (to the best of our knowledge):

1. **Key Establishment:** A user and (login) server(s) jointly compute a *password authenticated key exchange (PAKE) protocol* to establish a joint session key K. Later on, they use K for establishing secure and authenticated communication channel between the server(s) and the user. This is also called **implicit authentication**, since establishment of a joint key ensures that the user is the one who indeed previously registered with the server. The key K can be generated from symmetric (PAKE) or asymmetric (APAKE) information. [Boyen, 2009b, Bellovin and Merritt, 1992, Boyko et al., 2000, Jarecki et al., 2016a, Katz et al., 2001, MacKenzie et al., 2002, Jarecki et al., 2018] are some examples.

   Observe that standard (A)PAKE solutions depend on the complexity of the password for security, and are vulnerable to offline dictionary attacks, whereas in DiSPP, security will depend on the k-bit entropy secret generated independently of the password and offline dictionary attacks will not be possible.

2. **User Authentication:** A user and login server(s) compute a password protocol where server(s) authenticate the user. The authentication can be either symmetric (e.g. based on stored hashes) or asymmetric (e.g. based on signature verification). Message authentication code (MAC) [Turner, 2008], digital signatures [Goldwasser et al., 1988], and identification schemes [Schnorr, 1991, Feige et al., 1988, Fiat and Shamir, 1986] are some of the known examples of such **explicit authentication** schemes.

Chapter 4

DISTRIBUTED SINGLE PASSWORD PROTOCOL FRAMEWORK

4.1 *DiSPP Model*

In a DiSPP, there are three types of players. There are ***users*** who register with one or more ***login servers*** using (possibly) the same password, and later on compute a server-side password protocol with these login servers. For this purpose, the users securely store some secret information (that is needed for the password protocol with the login servers) at one or more ***storage providers***, which consist of personal user devices (e.g. mobile phone, tablet) and online storage providers (e.g. Dropbox, Google Drive). The main objective of a DiSPP solution is to protect the user's password against offline dictionary attacks by the storage providers, the login servers, and many other adversaries (including honeypots and phishing sites).

In a DiSPP, the user creates a *k-bit* entropy secret (e.g., k-bit random string rnd) *independent* of the password pwd (which only has l-bit entropy such that $l << k$). The password is assumed to resist only *online* dictionary attacks, but is not secure enough to resist offline dictionary attacks. Then, associated verification information based on the secret is computed (e.g., $H(rnd||ls)$ where ls is the domain name of the login server) and shared with the login servers depending on the server-side protocol, whereas the k-bit entropy secret is securely shared with the storage providers. This independent generation of the k-bit entropy secret for each registration enables a DiSPP to employ a single low-entropy password securely. But to authenticate or establish keys with the login server, the user needs to retrieve this secret from the storage providers (because it is of high entropy, it is not human-memorable). During the secure storage and retrieval of the secret at the storage providers, a DiSPP should ensure that only the legitimate user holding the correct password can retrieve these shares and reconstruct the secret. Since verifying whether or not the user is legitimate requires another authentication step (causing a chicken-egg problem), a DiSPP solves this paradox by enforcing the user (and hence any attacker) to employ an online protocol with the storage providers (e.g. OPRF).

A DiSPP consists of four main phases (see Figure 4.1): **Registration** where the user registers with login server(s) creating a high entropy secret and its associated verification information, **Secret Storage** where the user *securely* stores the secret using her password at the storage providers, **Secret Retrieval** where the user reconstructs the secret using her password employing (an authorized subset of) storage providers, and **Authentication** where the user implicitly or explicitly authenticates herself to the login server(s) using the secret reconstructed during the *secret retrieval* phase.[1]

The **registration** phase is for the user to register with the login server(s) with domain name ls. The user registers using a low-entropy password *pwd* (only secure against *online* attacks). Each login server obtains the user's verification information $vInfo_i$ such that the login server can authenticate the legitimate user (implicitly via a key establishment protocol or explicitly). The user obtains a k-bit entropy secret information S that is associated with the verification information to facilitate later protocols. More formally we have the following multi-party protocol:

Registration:

- **The user's input** is a password *pwd* and the server identifier *ls* (e.g. domain/url).

- **Each login server's input** is an identifier *ls*.

- **The user receives as output** a secret S (which will be used in secret storage) associated with the verification information.

- **Each login server receives as output** a verification information $vInfo_i$ based on the secret S, and stores this information in his database. The verification information $vInfo_i$ is used by the login servers to authenticate the user implicitly or explicitly during the authentication protocol.

The **secret storage** phase is for the user to store the $k-bit$ entropy secret S (generated during registration) among n_{stor}-many storage providers using a low entropy password. Each storage provider obtains a secret share $share_i$, while the user obtains nothing.

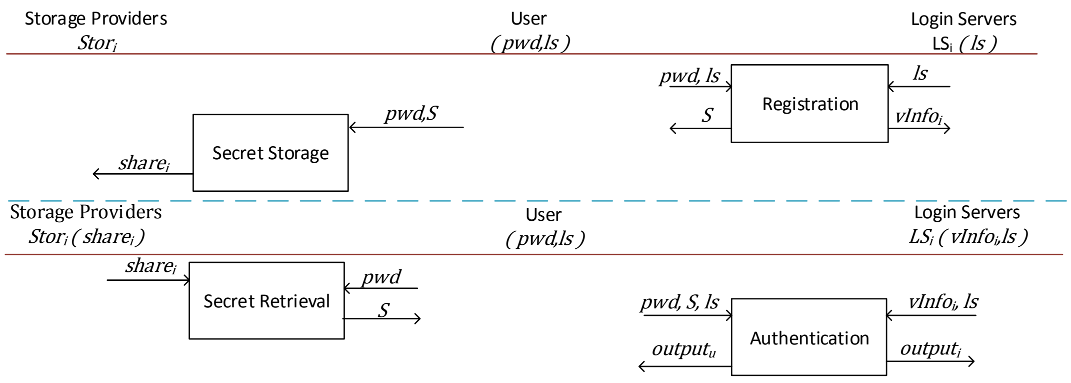
Secret Storage:

- **The user's inputs** are a password pwd, and the secret S.

- **Each storage provider receives as output** a share $share_i$ and stores the data received in its database.

The **secret retrieval** phase is for the user to retrieve and reconstruct the secret needed for authentication by interacting with an authorized subset of (e.g. *threshold* t_{stor}-many) storage providers.

Figure 4.1: DiSPP Overview



Secret Retrieval:

- **The user's input** is the password *pwd*.
- **Each storage provider's input** is the share $share_i$ that they hold for that user.
- **The user receives as output** the reconstructed secret S.

In Figure 4.2, we show an overview of the **registration** and **authentication** phases of a DiSPP in an combined version in Figure 4.1, considering a single user who registers with a login server and stores the secret at n storage providers.

Threshold in this context refers to the fact that the user must communicate with some subset (defined by the threshold) of storage providers to facilitate authentication with the login server. It furthermore refers to the security of the solution: An offline dictionary attack is possible only when the adversary controls the login server *and* at least threshold many storage providers.

The **registration** phase is for the user to register with the login server and store the secret among storage providers. The user registers with the login server whose domain is ls using a low-entropy password *pwd* (only secure against online attacks). The login server obtains the user's verification information $vInfo$ and identifier $userID$ such that the login server can authenticate the legitimate user whenever the user wants to login. The user further stores some secret information $share_i$ with the storage providers, in a distributed manner. Some identifier $storUID_i$ is associated with this secret to facilitate later retrieval. More formally we have the following multi-party protocol:

Registration:

1. **The user's inputs** are a user name *userID* for the login server whose domain is ls, and a password *pwd*.
2. **Each storage provider receives as output** an identifier $storUID_i$ and a share $share_i$ and stores the data received in the database. This share is what the user wants to store among the storage providers depending on the DiSPP protocol. The identifier is employed for later retrieval of the stored share.

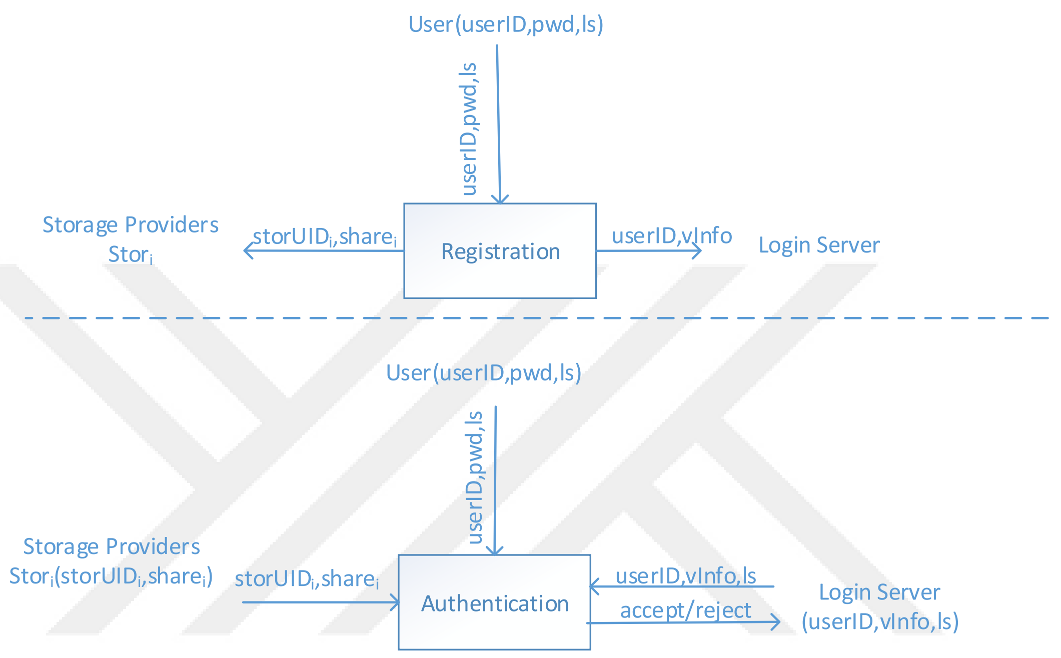


Figure 4.2: DiSPP Overview. The registration and authentication protocols are separated by the dashed line.

3. **The login server receives as output** an identifier *userID* and a server verification information based on user's password *vInfo* of the user, and stores them in his database. The verification information *vInfo* is used by the login server to verify the user during the authentication phase.

The **authentication** phase is for the user who remembers the user name *userID* and the password *pwd* to authenticate herself to the login server with domain *ls* by interacting with *threshold-many* (t) storage providers to retrieve and reconstruct the secret needed for authentication. Of course, in general it is possible that $t = n$ and hence all storage providers may need to be contacted.

Authentication:

1. **The user's inputs** are as before: the user name *userID*, the password *pwd*, and the domain *ls* of the login server to authenticate with.

2. **The login server's inputs** include the user identifier *userID*, as well as the verification information *vInfo* corresponding to the user, and its domain ls.
3. **Each storage provider's inputs** are the share $share_i$ that they hold for that user and the identifier $storUID_i$ of that user.
4. **The login server outputs** accept or reject. The domain name ls is employed to prevent phishing/man-in-the-middle attacks.

4.2 Security Definition

We define the *ideal world* and the *real world* for a DiSPP protocol (based on Figure 4.2), in the spirit of [Canetti, 2000].

Ideal World: The ideal world consists of a user $\mathcal{U}$, a login server $\mathcal{LS}$, *n-many* storage providers $\mathcal{SP} = (Stor_1, Stor_2, \ldots, Stor_n)$ (realize that $\mathcal{SP}$ denotes the set of storage providers), and the universal trusted party $\mathcal{TP}$ **(which is not a real entity, and only exists in the ideal world)**.

Registration :

1. $\mathcal{U}$ sends $< userID, pwd >$ to $\mathcal{TP}$.
2. $\mathcal{TP}$ computes the necessary steps to obtain the shares $share_i$ and identifiers $storUID_i$, and the verification information $vInfo$.
3. $\mathcal{TP}$ sends $< userID, vInfo >$ to $\mathcal{LS}$ and $< storUID_i, share_i >$ to each storage provider in $\mathcal{SP}$.

Authentication:

1. $\mathcal{U}$ sends $< userID, pwd >$ to $\mathcal{TP}$.
2. $\mathcal{TP}$ sends $userID$ to $\mathcal{LS}$ for login request.
3. $\mathcal{TP}$ sends $storUID_i$ to *at least threshold-many* storage providers in $\mathcal{SP}$ for retrieving the secret shares (wlog. assume all storage providers are employed).
4. $\mathcal{SP}$ send their shares $share = \{share_1, share_2, \ldots, share_t\}$.
5. $\mathcal{TP}$ calculates the verification information *vInfo* using the shares from the $\mathcal{SP}$ and the *pwd* from $\mathcal{U}$, and sends *vInfo* to $\mathcal{LS}$.

Real World: The real world consists of a user $\mathcal{U}$, a login server $\mathcal{LS}$, and storage providers $\mathcal{SP} = (Stor_1, Stor_2, \ldots, Stor_n)$. There is no universal trusted party $\mathcal{TP}$ for a real world protocol π for the threshold-single password authentication. The parties $\mathcal{U}$, $\mathcal{LS}$, and $\mathcal{SP}$ are involved in the real execution of the protocol π.

Definition 1 (Secure Distributed Single Password Protocol) *Let π be a probabilistic polynomial time (PPT) protocol for a threshold single password authentication. We say that π is secure if for every non-uniform PPT real world adversary $\mathcal{A}$ attacking π, there exists a non-uniform PPT ideal world simulator $\mathcal{S}$ such that for both registration and authentication phases, the real and ideal world interactions and outputs are computationally indistinguishable;*

$$\{IDEAL_{\mathcal{S}(aux)}(userID, pwd, ls, \lambda)\} \equiv_c \{REAL_{\pi,\mathcal{A}(aux)}(userID, pwd, ls, \lambda)\}$$

where $aux \in \{0,1\}^*$ denotes the auxiliary input, and λ is the security parameter.

Note that such an ideal world definition assumes secure and authenticated channels between parties. Furthermore, as there is only a single login server in the ideal world, it does not include phishing (this is why ls domain is not part of the ideal world). But it provides security against offline dictionary attacks. In Section 5.3 we discuss the security of our solution for attacks like phishing not covered by this ideal model definition.

4.3 Distributed Single Password Protocol Construction

We now present how to achieve a secure DiSPP in our framework by employing secure cryptographic building blocks. We firstly discuss possible cryptographic solutions for registration and authentication between *a user* and *one* login server (and later extend to n_{ls}-many login servers). Secondly, we present possible cryptographic solutions for the secret storage and retrieval phases that takes place between a *user* and *storage provider(s).*

4.3.1 Registration and Authentication Phases

Since registration and authentication phases are both server-side protocols, they need to be picked to work together. In a DiSPP, the essential step in the registration phase is to create a k-bit entropy secret *independent* of the user password, together with the associated verification information. Depending on the password protocol employed, the structures of the secret S and verification information $vInfo$ generated vary. Below we present and discuss various schemes that can be employed for registration and authentication purposes securely. We start by implicit authentication (key establishment) and then continue with explicit authentication methods.

Key Establishment (Implicit Authentication): A user and login server(s) run a setup phase where later on they can jointly compute a password authenticated key exchange protocol to establish a session key for a secure and authenticated channel. Standard PAKE protocols keep the password and APAKE protocols keep a deterministic function of the password (e.g. $H(pwd)$) at the server database, they are insecure against offline dictionary attacks. In a DiSPP, to provide provable security, key establishment works over the k-bit entropy secret S rather than the passwork. The **registration** and **authentication** of key exchange protocol is as follows;

Registration: The **user**:

- generates a k-bit entropy random string rnd as $rnd \leftarrow \{0,1\}^k$ or by running a key generation scheme of OPRF as $K \leftarrow OPRFKeyGen(1^\lambda)$.
- computes the verification information $vInfo$ as $H(rnd)$ or $F_K(pwd)$ using the random string or OPRF key respectively.
- sends $vInfo$ and assigns random strings rnd or K as the secret S.

The **login servers** receive $vInfo$ and store it.

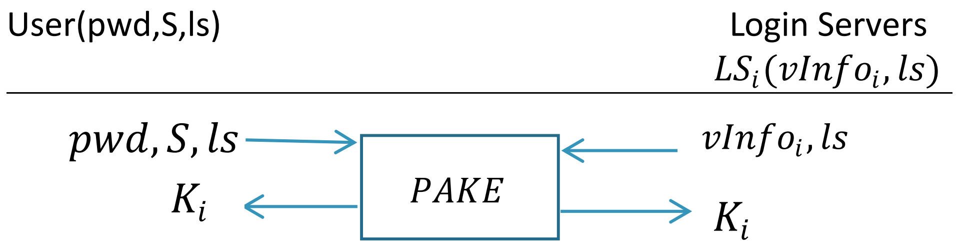
Authentication: The user and login server(s) jointly compute a password authenticated key exchange protocol to establish a session key K. Later on, they can use K for establishing a secure and authenticated communication channel, where the server(s) implicitly authenticate the user. A PAKE protocol is computed as follows;

- The **user** holding the secret S, password pwd and domain name of login server ls computes a PAKE protocol with the login server(s) [Boyen, 2009b, Bellovin and Merritt, 1992, Boyko et al., 2000, Jarecki et al., 2016a, Katz et al., 2001, MacKenzie et al., 2002, Jarecki et al., 2018], where each login server holds a verification information $vInfo_i$ and domain name ls (see Figure 4.3).

- The **user** and the **login server(s)** receive a session key K_i, where the user receives a session key per login server.

The security at the login server side can be increased using a threshold-PAKE (TPAKE) solution [Jarecki et al., 2014, MacKenzie et al., 2002, Katz et al., 2005]. TPAKE settings involve n_{ls} login servers such that a threshold t_{ls} of them must participate in the user authentication. An attacker corrupting any fewer than t_{ls} servers *cannot* perform an offline dictionary attack on the user's password.

Figure 4.3: Password authenticated key exchange protocol for implicit authentication.



Explicit Authentication: We categorized explicit authentication protocols into two: interactive (e.g. challenge-response) and non-interactive (e.g. $H(S||ls)$). In explicit authentication, we assume a single login server (i.e. $t_{ls} = n_{ls} = 1$).[2] Firstly, we describe the registration and authentication phases of interactive authentication protocols and later we discuss the non-interactive authentication protocols.

For all interactive authentication protocols below, the login server sends a challenge *chal* to the user.Then, the **user** runs the authentication scheme (based on the registration phase) to generate a response *resp* based on the challenge *chal* using her secret S and the domain name *ls* of the login server.

MAC-based Registration: The **user**:

- generates a MAC key K by running key generation algorithm as $K \leftarrow MACKeyGen(1^\lambda)$ (see Figure 4.4(a)).,
- sends the MAC key K as the verification information key $vInfo$ to the login server,
- assigns the MAC key as the secret ($S = K$).

The **login server** receives and stores $vInfo = K$.

MAC-based Authentication: The user runs $resp \leftarrow MAC_K(chal||ls)$ and sends the response *resp* to the login server, who runs $MACVerify_K(resp, chal||ls)$ (see Figure 4.4(b)).

Digital Signature-based Registration: The **user**

- generates signing and verification keys by running digital signature key generation algorithm as $ssk, svk \leftarrow SignKeyGen(1^\lambda)$ (see Figure 4.4(c)),
- sends the signature verification key *svk* as the verification information key $vInfo$ to the login server,
- assigns the signing key as the secret ($S = ssk$).

The **login server** receives and stores $vInfo = svk$.

Digital Signature-based Authentication: The user runs $resp \leftarrow Sign_{ssk}(chal||ls)$ and sends the response $resp$ to the login server, who runs $SignVerify_{svk}(resp, chal||ls)$ (see Figure 4.4(d)).

OPRF-based Registration: The **user**

- generates a key K by running OPRF key generation algorithm as $K \leftarrow OPRF(1^{\lambda})$ (see Figure 4.4(e)),
- sends verification information as $vInfo = F_K(pwd)$,
- assigns the OPRF key as the secret ($S = K$).

The **login server** receives and stores $vInfo = F_K(pwd)$.

OPRF-based Authentication: The user computes $vInfo \leftarrow F_K(pwd)$ where K is the secret S (see Figure 4.4(f)) and then sends the response as $resp = H(chal||ls||vInfo)$ to the login server, who checks whether locally computed $H(chal||ls||vInfo)$ is equal to the response of the client or not.

Hash-based Registration: The **user**:

- generates a *k-bit* entropy random string *rnd* as $rnd \leftarrow \{0, 1\}^k$ (see Figure 4.4(g)),
- sends the verification information as $vInfo = H(rnd||ls)$ to the login server,
- assigns the generated random as the secret ($S = rnd$).

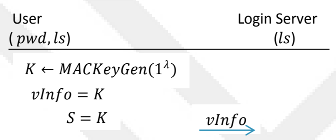
The **login server** receives and stores $vInfo = H(rnd||ls)$.

Hash-based Authentication: The user computes the response as $resp \leftarrow H(chal||ls||H(rnd||ls))$ (see Figure 4.4(h)) and sends $resp$ to the login server, who checks whether locally computed $H(chal||ls||vInfo)$ is equal to the response of the client or not.

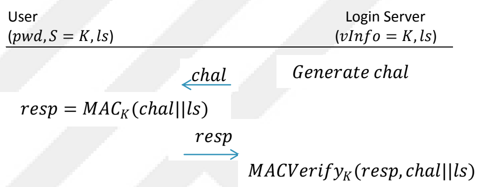
Non-interactive authentication: In non-interactive authentication, the user sends only one message to the login server as a login request. Such a round-optimal

Figure 4.4: Registration and authentication phases of interactive authentication schemes.

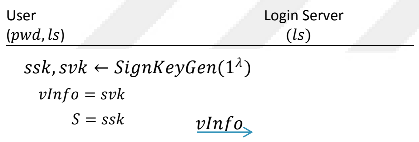
(a) MAC-based registration.



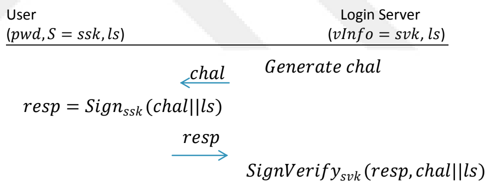
(b) MAC-based authentication.



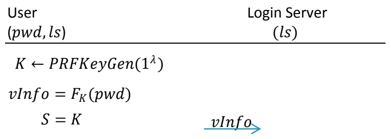
(c) Digital signature-based registration



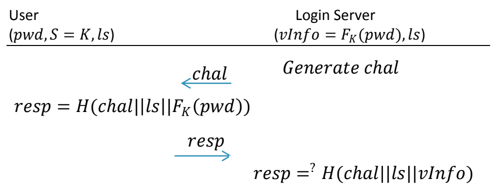
(d) Digital signature-based authentication.



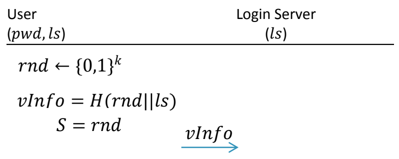
(e) OPRF-based registration



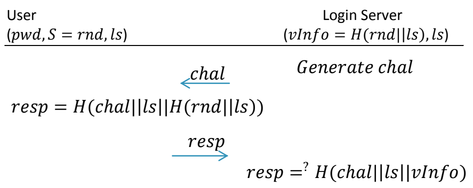
(f) OPRF-based authentication



(g) Hash-based registration



(h) Hash-based authentication


User
$(pwd, S = rnd, ls)$
Login Server
$(vInfo = H(rnd||ls), ls)$
$chal$
Generate $chal$
$resp = H(chal||ls||H(rnd||ls))$
$resp$
$resp =^? H(chal||ls||vInfo)$

solution can be based on challenge-response (non-interactive transformation of interactive authentication) or one-way functions such as a hash or a pseudo-random function. Non-interactive authentication **requires a secure and server-authenticated communication channel**, since it does not provide security against eavesdropping and replay attacks (except maybe some synchronous time-based measures). The reason is that an attacker can use the messages captured in earlier authentications to impersonate the user to the login server.

To convert the interactive protocols above to their non-interactive versions, the registration phases of the protocols above do not change. During authentication, the user picks a challenge *chal* randomly and sends $chal, resp$ to the server, who performs the same verification as above.

Alternatively, assuming $vInfo = H(rnd||ls)$ can be seen as a site-specific password, the hash-based authentication method can be simplified to the user computing $resp \leftarrow H(rnd||ls)$ and sending *resp* to the login server, who checks it against the locally stored $vInfo$ (or $H(vInfo)$ or $H(vInfo, salt)$) **without the need to modify the existing login servers**.

4.3.2 Secret Storage and Secret Retrieval Phases

We discuss possible building blocks for the secret storage and retrieval phases. The user stores the secret S (generated during the registration phase) among n_{stor} storage providers using her single password. Password protected secret sharing (PPSS) enables only the user who has the correct password to reconstruct the secret during secret retrieval. We firstly describe overview of password protected secret sharing employed in secret storage and retrieval phases. Later, we show how existing building blocks can be employed to have a secure PPSS.

PPSS-based Secret Storage: The user runs $\{share_i{}^{\gamma}\}_{\gamma\in\Gamma} \leftarrow PPSS(pwd, S, \Gamma)$ and sends $share_i$ to storage provider i, who stores it in its database.

PPSS-based Secret Retrieval: The user communicates with *threshold-many* storage providers to reconstruct the secret S via running the $S \leftarrow PPSSRecon(pwd, \{share_i{}^{\gamma}\}_{\gamma\in\Gamma}, \Gamma)$

protocol with t_{stor}-many storage providers where $share_i$ and pwd are the inputs of each authorized storage provider and the user, respectively.

A PPSS can be constructed using any secret sharing scheme together with an OPRF as follows: The user (dealer) runs the secret sharing method on a secret S to create its shares. To protect the shares, the user encrypts each share using OPRF evaluation result on the password. In details, the secret sharing and OPRF based solution (storage and retrieval) works as follows;

Secret Sharing and OPRF-based Secret Storage:

- **The user**
  1. generates an *OPRF* key $k_i \leftarrow OPRFKeyGen(1^{\lambda})$ per storage provider,
  2. runs a secret sharing scheme on the secret S to create the shares as $\{{s_i}^{\gamma}\}_{\gamma \in \Gamma} \leftarrow SS(S, \Gamma)$,
  3. encrypts each share using *oblivious pseudorandom function* of the password *pwd* using generated *OPRF* key of each storage provider obtaining $c_i \leftarrow Enc_{F_{k_i}(pwd)}(s_i)$,
  4. sends $share_i = (c_i, k_i)$ to each storage provider.
- **Each storage provider** receives a share $share_i$ and stores it in his database.

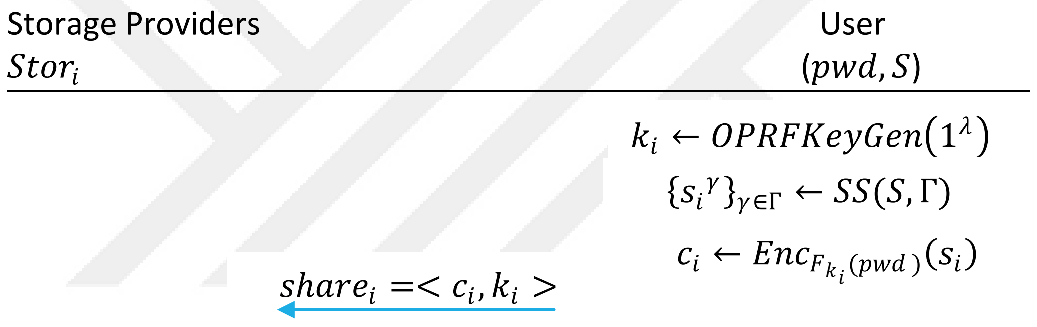
Secret Sharing and OPRF-based Secret Retrieval:

1. **The user and each storage provider jointly** execute the oblivious pseudo-random function (OPRF) protocol. Each storage provider acts as the sender and the user acts as the receiver in these protocol executions. The user obtains the OPRF value (with key k_i) of the password $F_{k_i}(pwd) \leftarrow OPRF(pwd, k_i)$ as the output.

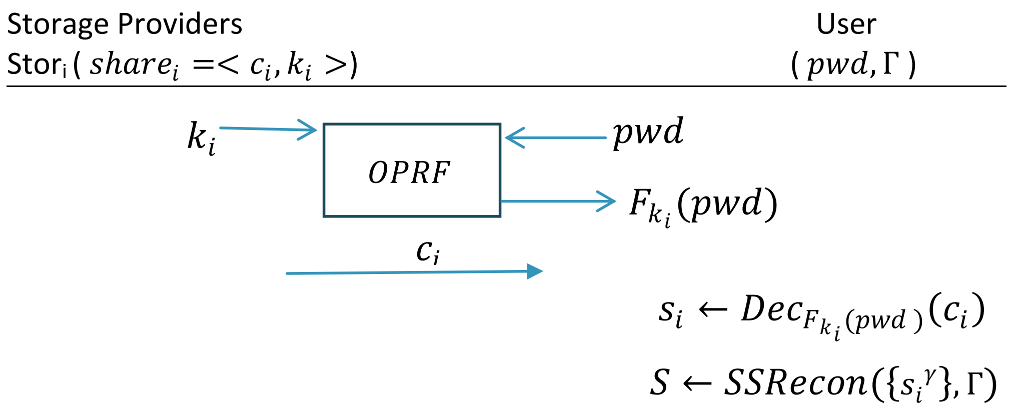
   Remark: The *OPRF* result is received *only* by the user and the storage providers are oblivious to the password. It is enough that this protocol is run among some authorized subset of storage providers and the user, rather than all storage providers.

Figure 4.5: PPSS secret storage and reconstruction based on SS-and-OPRF.

(a) SS-and-OPRF-based secret storage.


Storage Providers
$Stor_i$
User
(pwd, S)
$k_i \leftarrow OPRFKeyGen(1^\lambda)$
$\{s_i^\gamma\}_{\gamma\in\Gamma} \leftarrow SS(S,\Gamma)$
$c_i \leftarrow Enc_{F_{k_i}(pwd)}(s_i)$
$share_i = < c_i, k_i >$

(b) SS-and-OPRF-based secret reconstruction.


Storage Providers
Storᵢ($share_i = < c_i, k_i >$)
User
(pwd, Γ)
k_i
$OPRF$
pwd
$F_{k_i}(pwd)$
c_i
$s_i \leftarrow Dec_{F_{k_i}(pwd)}(c_i)$
$S \leftarrow SSRecon(\{s_i^\gamma\}, \Gamma)$

2. **Each storage provider** sends c_i to the user.

3. **The user** decrypts each ciphertext c_i using the corresponding OPRF output already received to obtain the secret shares $s_i \leftarrow Dec_{F_{k_i}(pwd)}(c_i)$ and computes threshold secret sharing reconstruction algorithm to reconstruct the secret as $S \leftarrow SSRecon(\{s_i^\gamma\}, \Gamma)$.

The secret sharing employed takes an important role for security of DiSPP. In the following, we investigate the secure secret sharing and OPRF instances. If a **threshold secret sharing** scheme is employed [Jarecki et al., 2016a, Jarecki et al., 2017, Acar et al., 2013], the user runs the secret retrieval phase with threshold-many (t_{stor}) storage providers. Alternatively, one can employ an **access control secret sharing** solution. Our framework allows usage of such a solution securely, and this brings more flexibility to the protocol as discussed below.

Consider, for example, a user employing n_{stor} storage providers in total, which consist of n_{mob} user personal devices (e.g. mobile phone, tablet) and n_{os} online storages (e.g. Dropbox). The user can define the authorized set of participants by choosing the thresholds as t_{mob} and t_{os} for specified personal devices and online storage providers, respectively. This means, to reconstruct the secret, (at least) t_{mob} personal devices and t_{os} online storage providers need to be involved in the secret retrieval phase.

For instance, consider a user who stores the secret by employing Dropbox (D) and GoogleDrive (G) as online storage providers and her mobile phone (M) and tablet (T) as personal devices. The scenario described above with mobile and online storage thresholds being one correspond to setting $\Gamma = \{\{D, M\}, \{D, T\}, \{G, M\}, \{G, T\}\}$ (and sets containing these subsets). Thus, the user retrieves the secret by accessing one of the online storage providers and one of her personal devices. Any attacker, who corrupts a set χ such that for any $\gamma \in \Gamma$ we have $\gamma \nsubseteq \chi$, cannot perform an offline dictionary attack to find user password. In particular, in the scenario above, consider an attacker who corrupts both online storage providers. As long as the attacker does not additionally have access to a personal device of the user, the protocol remains secure against offline dictionary attacks.

4.4 Inappropriate uses of primitives

4.4.1 Offline dictionary attack on NIVSS-and-OPRF-based solution

In DiSPP, the security property of secret sharing plays an important role. If NIVSS and OPRF based PPSS is employed in a distributed password protocol as secret storage, then an attacker can successfully perform an offline dictionary attack to find user password. The attack scenario is as follows;

A user runs secret storage protocol with n_{stor} storage providers described as in Figure 4.5(a), where NIVSS is employed as SS along with OPRF.

Consider an attacker $\mathcal{A}$, holding a dictionary $\mathcal{D}$ containing the user password, corrupts one of the storage providers called $Stor_c$, where the storage provider is holding a share as $< k_c, c_c >$. The attacker $\mathcal{A}$ runs the attack code as described in Algorithm 1 to find the user password. Since NIVSS is non-interactive, share verification algorithm can be computed by $\mathcal{A}$ without communicating with other stakeholders. If share verification is successful, then it would mean that the correct password was chosen while decrypting the c_c. Therefore, a DiSPP solution must *not* employ a NIVSS-and-OPRF-based PPSS solution.

4.4.2 How to encrypt secret shares

The particular attack above works because inside the ciphertext c_c, some extra information is stored that enables the adversary to verify the decryption password offline. To generalize the attack above, as long as the decryption of the ciphertext c_c may lead to some verifiable information, the solution would be insecure. For instance, in [Shamir, 1979] secret sharing, a share consists of a share number i (essential to reconstruct the secret) and the secret share s_i. If one encrypts this information as $c_i \leftarrow Enc_{F_{k_i}(pwd)}(s_i || i)$, the above attack still works simply by decrypting with the passwords in the dictionary and checking whether or not a proper integer i is ob-

```
Input: $\mathcal{D}, c_c, k_c$
Output: $password$
foreach $pwd' \in \mathcal{D}$ do
    $F_{k_c}(pwd') \leftarrow OPRF(pwd', k_c)$;
    $s_c' || v_c' \leftarrow Dec_{F_{k_c}(pwd')}(c_c)$;
    $b = NIVSSVerify(s_c', v_c')$;
    if $b == 1$ then
        return $pwd'$;
    end
end
```

Algorithm 1: Attack code on PPSS based on NIVSS-and-OPRF

tained.

Therefore, for SS-and-OPRF-based secret storage and retrieval to remain secure, it is important to only encrypt the random values, as first observed by [Acar et al., 2013]. Since, in [Shamir, 1979] secret sharing, the share number i is a public value that does not allow reconstruction of the secret by itself, it can be sent in clear, and only the random share value s_i is encrypted: $c_i \leftarrow Enc_{F_{k_i}(pwd)}(s_i)$ and each storage provider receives $share_i = (c_i, k_i, i)$ during secret storage, and sends c_i, i during secret retrieval.

4.4.3 Malware and Phishing Protection

We consider a strong phishing attack with man-in-the-middle between the user and the login server(s) during authentication (not registration). This means, the user registered with a legitimate server with ls (e.g. ls = paypal.com), but now is trying to authenticate with an attacker with ls' (e.g. ls'=paypa1.com).

During the authentication protocol between the user and login server(s), a DiSPP should employ ls information as an input to the cryptographic function evaluation (e.g. MAC). For instance, while computing the response $resp$ during authentication, the ls is concatenated with $chal$ (see Figure 4.4) and used as part of server-side verification. Therefore, if a phishing man-in-the-middle attacker with ls' exists, the user's response will be on $chal||ls'$, whereas the real server would verify using $chal||ls$, making the attack impossible. Without the use of ls during authentication, such attacks would have been possible.

(A)PAKE settings already ensure security against man-in-the-middle attacks intrinsically.

4.5 DiSPP instances

4.5.1 Existing DiSPP Instantiations

We discuss the DiSPP systems in the literature that provide provable security, employ a storage provider (e.g. mobile device or online storage provider) and ensure security with a *single password*. We investigate how these proposed systems fit in our DiSPP framework. [Acar et al., 2013] (with their patent application dating 2010 [Belenkiy et al., 2011]), [Bicakci et al., 2011a], and [Jarecki et al., 2016a]are the known examples of DiSPP. These DiSPP systems are secure against offline dictionary attacks, even by the server, hackers obtaining the server database, or up to threshold-many storage providers that are corrupted by the same adversary. We compare the aforementioned DiSPP systems in Table 4.1. All the existing DiSPP solutions ([Acar et al., 2013, Bicakci et al., 2011a, Jarecki et al., 2016a]) are built on an environment where there is one storage provider and one login server.

Acar et. al [Acar et al., 2013] proposed four DiSPPs in different settings with different concerns such as storage provider optimality, login server optimality, user anonymity, and mobile device as a storage provider. For our framework, user's anonymity is not a major concern since revealing the identity of the user does not break the security captured in our model, and we assume the user identifiers (e.g. usernames) already leak this information. Below, we only discuss their server optimal DiSPP instance which employs digital signatures for registration and authentication, and unique blind signatures instead of OPRF.

Registration:

Table 4.1: DiSPP solutions. TO: Threshold for online storage providers (t_{os}), TM: Threshold for personal devices (t_{mob}), TL: Threshold for login servers (t_{ls})

	TO	TM	TL
[Acar et al., 2013] Cloud Version	1	0	1
[Acar et al., 2013] Mobile Version	0	1	1
[Bicakci et al., 2011a]	1	0	1
[Jarecki et al., 2016a]	0	1	1
Our instantiation	t_{os}	t_{mob}	t_{ls}

- The user and login server run the digital signature registration as in Figure 4.4(c) where the secret S is the signing key ssk.

Secret Storage:

- The user and the storage provider run the secret storage protocol as in Figure 4.5(a). Since there is only one storage provider ($t_{stor} = n_{stor} = 1$), the user does not run a secret sharing on S and sends only the encryption of the S to the storage provider.

Secret Retrieval:

- The user and the storage provider run the secret reconstruction as in Figure 4.5(b) where the user receives the secret S as an output.

  Remark: As in the secret storage phase, the user does not run a secret sharing reconstruction. She computes one unique blind signature (as an OPRF) with the storage provider and performs one decryption locally to obtain the secret without further reconstruction.

Authentication:

- The user and login server run the digital signature authentication as shown in Figure 4.4(d).

Jarecki et. al [Jarecki et al., 2016a] is an instance of DiSPP that employs the mobile device as storage provider. In [Jarecki et al., 2016a], PAKE is embedded as server-side password protocol.

Registration:

- The user and the login server compute the registration of a PAKE protocol. The login server stores an OPRF evaluation of password as $F_K(pwd)$ shown in Figure 4.4(e).

Secret Storage:

- The user and mobile device compute the secret storage protocol as described in Figure 4.5(a).

  Remark: In their setting, since there is only one storage provider, secret sharing is not employed, and the user sends the secret (which is the OPRF key K) as it is without an encryption ($share = K$).

Secret Retrieval: The user holding her password and the storage provider (mobile device) compute the secret retrieval solution as in Figure 4.5(b).

Remark: They only run the OPRF part, and no decryption or secret sharing is employed.

Authentication: The user holding the $F_K(pwd)$ retrieved from the secret retrieval runs PAKE with the login server as described in Figure 4.3.

Remark: Since they perform a PAKE protocol (an implicit authentication), their authentication protocol is not same as Figure 4.4(f) while registration is the same.

Bicakci et. al [Bicakci et al., 2011a] proposed a DiSPP (without a formal security proof), where they employed one login server and one storage provider. Their model is captured by our framework as follows;

Registration:

- The user generates a pair of keys (ssk, svk) for a unique blind signature based on (deterministic) RSA, where the high entropy secret is generated as $sig \leftarrow$

$Sign_{ssk}(H(pwd||username))$ and the verification information is computed as $vInfo = H(sig||ls)$.

Secret Storage:

- The user and the storage provider compute the secret storage protocol as in Figure 4.5(a).

  Remark: Since there is only one storage, secret sharing is not computed as well as no encryption on the secret ($share = ssk$).

Secret Retrieval:

- The user runs secret retrieval as in Figure 4.5(b) where unique blind signature is employed instead as OPRF (recall both OPRF and unique blind signature provides the same functionality and security). The output to the user is $sig \leftarrow Sign_{ssk}(H(pwd||username))$.

Authentication:

- The user runs (non-interactive) hash-based authentication protocol with the login server as described in Section 4.3.1.

4.5.2 Another DiSPP Instantiation

Our framework enables us to derive other DiSPP instances not in the literature.

Registration:

- uses the hash-based registration as described in Figure 4.4(g) that outputs a high entropy random-string rnd as the secret S and verification information as $vInfo = H(rnd||ls)$.

  Remark: To further strengthen login servers against offline dictionary attacks, rather than employing a single server, threshold password authenticated key exchange (TPAKE) can be employed so that an attacker is required to compromise *threshold-many* login servers in addition to *threshold-many* storage providers to mount a successful offline dictionary attack.

Secret Storage:

- employs ACSS-and-OPRF-based secret storage as described in Section 4.3.2.

Secret Retrieval:

- runs ACSS-and-OPRF-based secret retrieval protocol with an authorized set of storage providers to retrieve the secret $S = rnd$.

Authentication:

- runs hash-based authentication as shown in 4.4(h) using the reconstructed secret S during the secret retrieval.

  Remark: Alternatively, one may employ non-interactive version or TPAKE as discussed above.

4.5.3 Performance Evaluation

In this section, we first evaluate the performance of some building blocks, and then look at the performance of the DiSPP instance proposed above. We omit the evaluations of registration and authentication for the login servers, since schemes employed are run in constant time and with non-interactive hash-based registration and authentication, the login servers remain unmodified.

Performance measurements are processed on a standard laptop machine with Intel Core(TM) i7-5600U CPU 2.60 GHz, 8.00 GB RAM, 64-bit OS, and implemented in Java. For our implementation, we chose AES [Daemen and Rijmen, 2013] for encryption, implemented OPRF in [Jarecki et al., 2016a], TSS in [Shamir, 1979], ACSS in [Benaloh and Leichter, 1990], TOPRF in [Jarecki et al., 2017], and PPSS in [Camenisch et al., 2014] with various thresholds. Table 5.1 shows the local computations of the secret storage and retrieval phases at the user side. For the secret storage and registration, the storage providers and the login server *do not* compute anything; they only receive and store some values.

Table 4.2: Performance evaluation of DiSPP phases (in milliseconds).

User-Secret Storage			
	ACSS-OPRF [Benaloh and Leichter, 1990, Jarecki et al., 2016a]	TOPRF [Jarecki et al., 2017]	PPSS [Camenisch et al., 2014]
2-5 Threshold	9.30	3.6	66.35
3-6 Threshold	11.88	4.6	66.60
5-10 Threshold	17.8	7.8	68.40
User-Secret Retrieval			
	ACSS-OPRF [Benaloh and Leichter, 1990, Jarecki et al., 2016a]	TOPRF [Jarecki et al., 2017]	PPSS [Camenisch et al., 2014]
2-5 Threshold	5.22	4.0	952.18
3-6 Threshold	7.8	5.5	1273.12
5-10 Threshold	12.4	8.4	1603.21
Storage Provider-Secret Retrieval			
	ACSS-OPRF [Benaloh and Leichter, 1990, Jarecki et al., 2016a]	TOPRF [Jarecki et al., 2017]	PPSS [Camenisch et al., 2014]
2-5 Threshold	2.1	3.3	807.15
3-6 Threshold	3.15	3.9	901.34
5-10 Threshold	5.2	6.5	1523.02

Finally, we show the performance evaluation of our proposed instantiation for the user and storage providers in Table 4.3. From the communication round perspective, the user can communicate with the storage providers in parallel, which decreases the network round trip to 1.5 rounds for secret retrieval and authentication in total.

Table 4.3: Performance evaluation of our DiSPP instantiation (in milliseconds).

	User (Reg.)	User (Auth.)	Storage Provider (Retrieval)	Login Total
2-5 Threshold	9.90	5.96	2.1	8.06
3-6 Threshold	12.48	8.56	3.15	11.71
5-10 Threshold	18.40	13.13	5.2	18.33

Notes to Chapter 4

1 Protocols may include some username information, but we choose not to complicate our presentation with things not directly associated with security.

2 [Ford and Kaliski, 2000] proposed two-servers setting, where there is a main login server the user wants to authenticate herself with, and a helper login server such that the adversary needs to corrupt both.

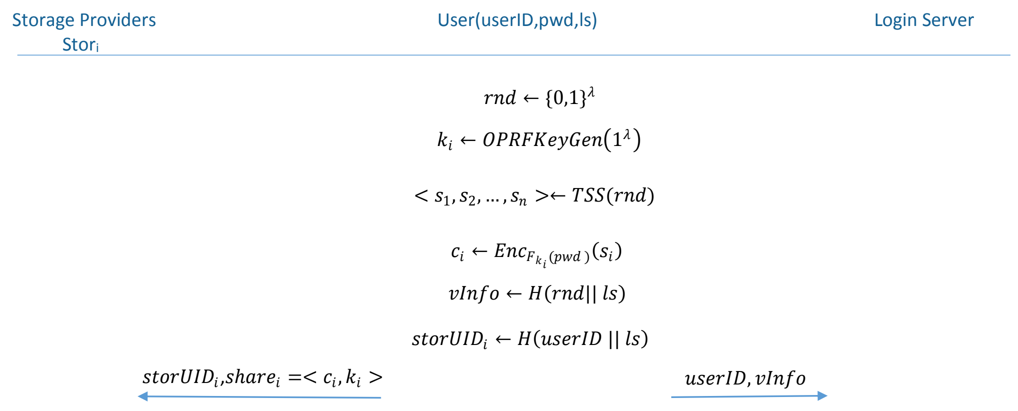
Chapter 5

DISPP INSTANTIATION

In this chapter, we present our proposed DiSPP instantiation by formally proving its security using our real and ideal world definitions and later on we discuss its performance evaluation.

5.1 *DiSPP Scheme*

Our DiSPP instantiation is represented visually in Figure 5.1 (*registration phase*) and Figure 5.2 (*authentication phase*). It is also described below.



Storage Providers Stor$_i$
User(userID,pwd,ls)
Login Server
$rnd \leftarrow \{0,1\}^\lambda$
$k_i \leftarrow OPRFKeyGen(1^\lambda)$
$< s_1, s_2, \ldots, s_n > \leftarrow TSS(rnd)$
$c_i \leftarrow Enc_{F_{k_i}(pwd)}(s_i)$
$vInfo \leftarrow H(rnd || ls)$
$storUID_i \leftarrow H(userID || ls)$
$storUID_i, share_i = < c_i, k_i >$
$userID, vInfo$

Figure 5.1: DiSPP Construction Registration Phase

Registration:

1. **The user**
   (a) generates a random rnd as $rnd \leftarrow \{0,1\}^\lambda$
   (b) generates one $OPRF$ key per storage provider as $k_i \leftarrow OPRFKeyGen(1^\lambda)$.

(c) runs threshold secret sharing construction scheme on rnd to create the secret share for each storage provider $< s_1, s_2, ..., s_n > \leftarrow TSS(rnd)$.

(d) encrypts each share using *oblivious pseudorandom function* of the password pwd using generated $OPRF$ key of each storage provider obtaining $c_i \leftarrow Enc_{F_{k_i}(pwd)}(s_i)$.

Remark: Since the secret shares are random bitstrings, offline dictionary attacks on these encryptions are impossible. Therefore, in our solution, even all the storage providers,without the help of the login server, they cannot break the security.

(e) computes verification information for the login server via a collusion-resistant hash function as $vInfo = H(rnd||ls)$

Remark: Salt is a randomstring with size of security parameter. For that reason, the login server, without colluding at least *t-many* storage provider, cannot perform a successful dictionary attack.

(f) computes the same identifier for all storage providers via a collusion-resistant hash function as $storUID_i \leftarrow H(userID||ls)$.

Remark: This identifier is only used to retrieve the correct values from the storage providers that serve multiple clients. Remember that ls is the domain name of the server the user is registering/connected to (e.g. ls=paypal.com).

(g) sends $< userID, vInfo = H(rnd||ls) >$ to the login server, and $< storUID_i$, $share_i = (c_i, k_i) >$ to each storage provider.

(h) can forget all the data she computed that are cumbersome for her to remember (e.g. K, $k_{i=1,..,n}$).

2. **The login server** receives $< userID, vInfo = H(rnd||ls) >$, and stores the pair in his database.
3. **Each storage provider** receives $< storUID_i, share_i = (c_i, k_i) >$ and stores in the database.

Authentication:

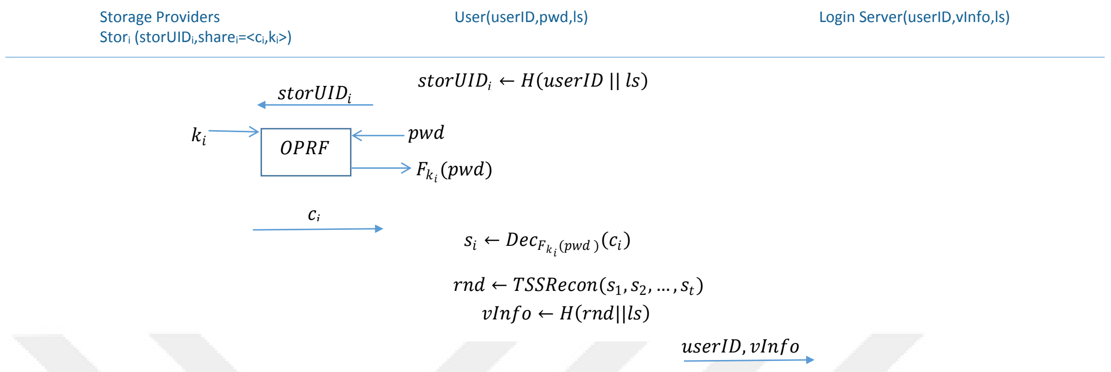

Storage Providers
Stor$_i$ (storUID$_i$,share$_i$=<c$_i$,k$_i$>)
User(userID,pwd,ls)
Login Server(userID,vInfo,ls)
$storUID_i \leftarrow H(userID \mid\mid ls)$
$storUID_i$
k_i
OPRF
pwd
$F_{k_i}(pwd)$
c_i
$s_i \leftarrow Dec_{F_{k_i}(pwd)}(c_i)$
$rnd \leftarrow TSSRecon(s_1, s_2, \ldots, s_t)$
$vInfo \leftarrow H(rnd||ls)$
$userID, vInfo$

Figure 5.2: DiSPP Construction Authentication Phase

1. **The user** who is trying to authenticate with the login server with domain ls computes the same storage identifier $storUID_i \leftarrow H(userID||ls)$ and sends it to at least t-$many$ storage providers, and sends $userID$ to login server.
2. **Each storage provider** finds the associated $< share_i = (k_i, c_i) >$ with $storUID_i$.
3. **The user and each storage provider jointly** execute the oblivious pseudo-random function (OPRF) protocol. Each storage provider acts the sender and the user acts as the receiver in these protocol executions. The user obtains the OPRF value (with key k_i) of the password $F_{k_i}(pwd) \leftarrow OPRF(pwd, k_i)$ as the output.

   Remark: The $OPRF$ result is received *only* by the user.
4. **Each storage provider** sends c_i to the user.
5. **The user** decrypts each ciphertext c_i using the corresponding OPRF output already received to obtain the secret shares $s_i \leftarrow Dec_{F_{k_i}(pwd)}(c_i)$ and computes threshold secret sharing reconstruction algorithm to reconstruct the secret $rnd \leftarrow TSSRecon(s_1, s_2, ..., s_t)$.

   Remark: Even when at least threshold-many storage providers collude and reconstruct the ciphertext rnd of the original secret rnd by trying different passwords in the dictionary, they still need to try the resulting rnds *online* against the login server, since each password in the dictionary would result in a valid *rnd* when decrypting shares S.

6. **The user** computes the verification information as $vInfo = H(rnd||ls)$ and sends $< userID, vInfo = H(rnd||ls) >$ to the login server.
7. **The login server** looks up the verification information *vInfo* associated with *userID*, and it accepts the response if and only if the *vInfo* sent by the user same as the *vInfo* in the database.[1]

   Remark: The domain name of the login server ls in the hash is to prevent a phishing/man-in-the-middle attacks. This attack prevention is discussed in Section 5.2 in details.

5.2 Security Proof

Theorem 1 *Our DiSPP protocol is secure according to Definition 1 against any non-uniform PPT adversary $\mathcal{A}$ corrupting* ***the login server $\mathcal{LS}$ and unauthorized set of many storage providers $\mathcal{SP}_c$****, assuming that the threshold secret sharing construction is secure, encryption scheme is semantically-secure, the oblivious pseudorandom function is secure, and the hash function is collision resistant.*

Proof 1 *The simulator $\mathcal{S}$ simulates honest parties in the real world (which are the user $\mathcal{U}$ and n-t+1 storage providers denoted by $\mathcal{SP}_h$ =$\{Stor_{i_h}\}$ where $i_h = t, ..., n$ wlog. since all storage providers in our solution are identical) and corrupted parties in the ideal world (which are the login server $\mathcal{LS}$ and t-1 storage providers denoted by $\mathcal{SP}_c$ =$\{Stor_{i_c}\}$ where $i_c = 1, ..., t-1$). $\mathcal{S}$ behaves as follows:*

Registration Phase:

1. *The user $\mathcal{U}$ in the ideal world sends <userID,pwd> to $\mathcal{TP}$*
2. *The universal trusted party $\mathcal{TP}$:*
   (a) *generates a random rnd as $rnd \leftarrow \{0,1\}^\lambda$*
   (b) *generates one OPRF key per storage provider as $k_i \leftarrow OPRFKeyGen(1^\lambda)$.*
   (c) *runs threshold secret sharing construction scheme on rnd to create the secret share for each storage provider $< s_1, s_2, ..., s_n > \leftarrow TSS(rnd)$.*

(d) *encrypts each share using oblivious pseudorandom function of the password pwd using generated OPRF key of each storage provider obtaining* $c_i \leftarrow Enc_{F_{k_i}(pwd)}(s_i)$.

(e) *computes verification information for the login server via a collusion-resistant hash function as* $vInfo = H(rnd||ls)$

(f) *computes the same identifier for all storage providers via a collusion-resistant hash function as* $storUID_i \leftarrow H(userID||ls)$.

(g) *sends* $< userID, vInfo = H(rnd||ls) >$ *to the login server (which is the simulator* $\mathcal{S}$*), and* $< storUID_i, share_i = (c_i, k_i) >$ *to each storage provider.*

3. $\mathcal{S}$ *receives* $< userID, vInfo = H(rnd||ls), \{storUID_i, share_i = (c_i, k_i)\}_{i=1,\ldots,t-1} >$ *from* $\mathcal{TP}$.

   Remark: *Since* $\mathcal{S}$ *simulates* $\mathcal{LS}$ *and* $\mathcal{SP}_c = \{Stor_{i_c}\}_{i_c=1,\ldots,t-1}$ *in the ideal world,* $\mathcal{S}$ *receives whatever they receive from* $\mathcal{TP}$. *Because of the symmetry of the actions of the storage providers in our construction, which ones are corrupted by the adversary does not change anything in the proof as long as the number of corrupted storage providers is below the threshold.*

4. $\mathcal{S}$ *sends* $< userID, vInfo = H(rnd||ls) >$ *to the adversarial* $\mathcal{LS}$ *in the real world.*

5. $\mathcal{S}$ *follows the protocol as a user choosing a random password* pwd' *from the dictionary and a secret share* s'_{i_c} *for each corrupted storage provider, and sends* $< storUID_{i_c}, share_{i_c} = (c'_{i_c}, k_{i_c}) >$ *where* $c'_{i_c} = Enc_{F_{k_i}(pwd')}(s'_{i_c})$ *to each adversarial storage provider* $\{Stor_{i_c}\}_{i_c=t,\ldots,n}$.

   Remark: *Adversarial storage providers receive encrypted shares of random values with the random password* pwd'. *There is no efficient way for adversarial storage providers to distinguish this from real behavior since one more storage provider needs to be corrupted to mount a successful offline dictionary attack. For our protocol, all* $storUID_i$ *values are the same.*

6. $\mathcal{S}$ *stores all the data in its database.*

Authentication Phase:

1. ***The user*** *who is trying to authenticate with the login server with domain* ls *in the ideal world computes the same storage identifier* $storUID_i \leftarrow H(userID||ls)$ *and sends it to at least t-many storage providers, and sends* $userID$ *to login server.*
2. $\mathcal{S}$ *receives* $< \{storUID_i\}_{i=1,...,t-1} >$ *from* $\mathcal{TP}$.

   Remark: *In general, since* $\mathcal{TP}$ *may pick any (threshold size) subset of storage providers to work with, and so not all adversarial storage providers may need to be contacted. We are assuming the most powerful adversary here, therefore suppose that all adversarial storage providers are contacted.*
3. $\mathcal{S}$ *sends* $storUID_{i_c}$ *to each storage provider* $Stor_{i_c}$ *where* $i_c = 1, ..., t-1$.

   Remark: *While* $\mathcal{S}$ *could already contact the* $\mathcal{TP}$ *regarding the storage providers at this point (since it already possesses the necessary shares), this may be distinguishable by the adversary. It is possible that the adversarial storage providers will not provide correct values in the real world, and hence the real authentication may fail. The simulator must ensure in that case that the ideal authentication also fails. The following steps are hence necessary for indistinguishability.*
4. $\mathcal{S}$ *executes the OPRF protocol with each* $\{Stor_{i_c}\}_{i_c=1,...,t-1}$ *using the password* pwd'*, and receives* $p_{i_c} = F_{k_{i_c}}(pwd')$ *and also* c_{i_c} *from each real* $Stor_{i_c}$.
5. $\mathcal{S}$ *checks whether or not each* $\{Stor_{i_c}\}_{i_c=1,...,t-1}$ *used the correct corresponding* $share_{i_c} = (c_{i_c}, k_{i_c})$ *values.* $\mathcal{S}$ *already possesses the correct values obtained from* $\mathcal{TP}$ *during registration in the database. For each* p_{i_c}, c_{i_c} *received from* $Stor_{i_c}$, $\mathcal{S}$ *does the following: Using the corresponding* c'_{i_c}, k_i *stored in its database during registration, it computes* $p_i = F_{k_i}(pwd')$ *locally and checks whether or not* $p_{i_c} = p_i$ *and* $c_{i_c} = c'_{i_c}$*. There are two cases for each* $Stor_{i_c}$*:*

   (a) ***Case 1: Correct*** $share_{i_c} = (c_{i_c}, k_{i_c})$ ***employed by the adversary in the real protocol***. $\mathcal{S}$ *detects this by verifying that* $p_{i_c} = p_i$ *and* $c_{i_c} = c'_{i_c}$*. Therefore,* $\mathcal{S}$ *sends* (c_i, k_i) *in its database to* $\mathcal{TP}$ *where* c_i, k_i *was sent by* $\mathcal{TP}$ *during the registration.*

   (b) ***Case 2: Incorrect*** $share_{i_c} = (c_{i_c}, k_{i_c})$ ***employed by the adversary***

in the real protocol. *$\mathcal{S}$ detects this by verifying that $p_{i_c} \neq p_i$ or $c_{i_c} \neq c'_{i_c}$.*

  i. *If $p_{i_c} = p_i$ and $c_{i_c} \neq c'_{i_c}$, $\mathcal{S}$ sends (c_{i_c}, k_i) to $\mathcal{TP}$, where k_i was in its database.*

  ii. *If $p_{i_c} \neq p_i$, $\mathcal{S}$ generates a random OPRF key $k'_i \neq k_i$, and sends (c_i, k'_i) to $\mathcal{TP}$ where c_i sent by $\mathcal{TP}$ during registration.*

Remark: *Even though $\mathcal{S}$ does not have any knowledge about k_{i_c} used by $Stor_{i_c}$, he can easily understand if each $Stor_{i_c}$ used the correct input k_i by computing the OPRF locally using k_i in the database. Then, if incorrect k_{i_c} or c_{i_c} are employed in the real protocol, $\mathcal{S}$ also sends incorrect values to $\mathcal{TP}$, in which case both the real and ideal responses will fail.*

6. *$\mathcal{TP}$ calculates and sends the verification information vInfo and userID to $\mathcal{S}$ based on the $\{c_i, k_i\}_{i=1,\dots,t-1}$ received from $\mathcal{S}$, together with (at least) one (c_i, k_i) pair from one of the remaining n-t+1 honest storage providers to reach the threshold t.*

   Remark: *$\mathcal{TP}$ employs the ideal user provided password in the ideal world. Therefore, if the adversarial storage providers in the real world acted honestly meaning that the simulator provided correct c_i, k_i pairs, then the calculated verification information will be valid, since it is computed using the actual password. On the other hand, if the storage providers acted maliciously in the real world, $\mathcal{S}$ would have detected this in the previous step, and would have provided wrong pairs to $\mathcal{TP}$ in the ideal world, so in both worlds the response will be invalid.*

7. *$\mathcal{S}$ forwards $< userID, vInfo >$ to the adversarial $\mathcal{LS}$ in the real world.*

Claim 1 *The view of adversary $\mathcal{A}$, controlling the login server $\mathcal{LS}$ and unauthorized set of storage providers $\mathcal{SP}_c$, in his interaction with the simulator $\mathcal{S}$ is indistinguishable from the view of his interaction with a real honest party.*

Proof 2 *$\mathcal{S}$ acts differently while sending shares c'_i calculated based on randomly chosen pwd' instead of sending actual c_i (sent by $\mathcal{TP}$) calculated based on actual password* *pwd and executing the OPRF with the* $Stor_{i_c}$ *using the password* pwd' *chosen randomly because* $\mathcal{S}$ *does not have the correct password. If* $\mathcal{A}$ *can distinguish these behaviors, then we can construct another adversary* $\mathcal{A}'$ *which breaks either the OPRF construction or TSS construction. We skip this relatively straightforward reductions for the sake of space, but intuitively;*

1. ***Reduction 1:*** *The OPRF security ensures that the sender (the adversarial storage providers) cannot distinguish the receiver (the simulated user) input, whether it is the actual password pwd or another randomly chosen password* pwd'*. Such a reduction will be a hybrid proof, where if at least one adversarial storage provider distinguishes the simulator from the real user, that can be used to distinguish the OPRF receiver input.*
2. ***Reduction 2:*** *The TSS security ensures that less than threshold many providers cannot reconstruct the secret and also cannot check if the shares are indeed related to the same secret. Intuitively, if adversarial storage providers can distinguish the simulator, who employs random secret shares during the registration, from the real user, then that can be used to break the security of the underlying threshold secret sharing scheme.*

Moreover, even though $\mathcal{A}$ *knows the verification information* $vInfo = H(rnd||ls)$ *and* $\{c_i, k_i\}_{i=1,\ldots,t-1}$ *from the registration,* $\mathcal{A}$ *cannot perform an offline dictionary attack on the password because he needs one more* (c_i, k_i) *to reach the threshold* t *to reconstruct the secret rnd. This part can be information theoretically secured if an information theoretically secure threshold secret sharing scheme (e.g. [Shamir, 1979]), semantically secure encryption scheme and collision resistant hash function are employed.*

Theorem 2 *Our DiSPP instantiation is secure according to Definition 1 against any non-uniform PPT adversary* $\mathcal{A}$ *corrupting* ***an authorized set of storage providers*** $\mathcal{SP}_c$*, assuming that the threshold secret sharing construction is secure, encryption scheme is semantically-secure, the oblivious pseudorandom function is secure, and the hash function is collision resistant.*

Proof 3 *The simulator* $\mathcal{S}$ *simulates honest parties (which are the login server* $\mathcal{LS}$

and the user $\mathcal{U}$) in the real world and corrupted parties (which are n storage providers denoted by $\mathcal{SP}_c = \{Stor_{i_c}\}_{i_c=1,...,n}$) in the ideal world. $\mathcal{S}$ behaves as follows:

Registration Phase:

1. *$\mathcal{S}$ receives $< storUID_i, share_i = (c_i, k_i) >$, where $i = 1, ..., n$ from $\mathcal{TP}$. $\mathcal{S}$ follows the protocol as a user choosing a random password pwd' from the dictionary and a secret share s_{ic} for each corrupted storage provider, and sends $< storUID_{i_c}, share_{i_c} = (c'_{i_c}, k_{i_c}) >$ where $c'_{ic} = Enc_{F_{k_i}pwd'}(s'_{ic})$ to ℓ-many adversarial storage providers $\{Stor_{i_c}\}_{i_c=1,...,\ell}$ and for the rest, it sends $< storUID_{i_c}$, $share_{i_c} = (c_i, k_i) >$ in the real world*
2. *$\mathcal{S}$ stores the all the data in its database.*

Authentication Phase:

1. *$\mathcal{S}$ receives $\{storUID_i\}_{i=1,...,n}$ from $\mathcal{TP}$.*

   Remark: *As $\mathcal{TP}$ may choose a (threshold size) subset of storage providers, we assume the worst case to give the maximum power to the adversary controlling all storage providers and all their values are employed in the protocol.*
2. *$\mathcal{S}$ sends $storUID_{i_c}$ to each $Stor_{i_c}$ where $i_c = 1, ..., n$.*
3. *$\mathcal{S}$ executes OPRF protocol with each $\{Stor_{i_c}\}_{i_c=1,...,n}$ using the password pwd', and receives $p_{i_c} \leftarrow OPRF(pwd', k_{i_c})$ and c_{i_c} from each $Stor_{i_c}$ in real.*
4. *$\mathcal{S}$ checks whether or not each $\{Stor_{i_c}\}_{i_c=1,...,n}$ used the correct corresponding $share_{i_c} = (c_{i_c}, k_{i_c})$ values. $\mathcal{S}$ already holds the correct corresponding values during registration in the database. For each (p_{i_c}, c_{i_c}) received from $Stor_{i_c}$, $\mathcal{S}$ does the following: Using the corresponding c_i, k_i stored in its database during registration, it computes $p_i = F_{k_i}(pwd')$ locally and checks whether or not $p_{i_c} = p_i$, $c_{i_c} = c_i$ for corresponding $n - \ell$ $Stor_{i_c}$ and $c_{i_c} = c'_{i_c}$ for ℓ many $Stor_{i_c}$. There are two cases for each $Stor_{i_c}$:*

   (a) ***Case 1: Correct** $share_{i_c} = (c_{i_c}, k_{i_c})$ **employed by the adversary in the real protocol**. $\mathcal{S}$ detects this by verifying that $p_{i_c} = p_i$ and $c_{i_c} = c'_{i_c}$ for ℓ-many $Stor_{i_c}$ and $c_{i_c} = c_i$ for n-ℓ-many $Stor_{i_c}$. Therefore, $\mathcal{S}$ sends (c_i, k_i) in its database to $\mathcal{TP}$.*

(b) ***Case 2: Incorrect*** $share_{i_c} = (c_{i_c}, k_{i_c})$ ***employed by the adversary in the real protocol***. *$\mathcal{S}$ detects this by verifying that $p_{i_c} \neq p_i$ or $c_{i_c} \neq c'_{i_c}$ for ℓ-many $Stor_{i_c}$ and $c_{i_c} \neq c_i$ for n-ℓ-many $Stor_{i_c}$.*

i. *If $p_{i_c} = p_i$ and $c_{i_c} \neq c_i$ are sent by $\alpha -$ many $Stor_{i_c}$ and $p_{i_c} = p_i$ and $c_{i_c} \neq c'_{i_c}$ are sent by $\beta -$ many $Stor_{i_c}$, $\mathcal{S}$ sends $\alpha + \beta$ many (c_{i_c}, k_i) to $\mathcal{TP}$,where k_i was in its database, in case $\alpha + \beta \geq n - t + 1$. Otherwise, meaning that $\alpha + \beta < n - t + 1$, $\mathcal{S}$ sends $\alpha + \beta$ many (c_i, k_i) and $n - (\alpha + \beta)$ many (c_{i_c}, k_i) to $\mathcal{TP}$.*

ii. *If $p_{i_c} \neq p_i$, $\mathcal{S}$ generates a random OPRF key $k'_i \neq k_i$, and sends (c_{i_c}, k'_i) to $\mathcal{TP}$.*

Remark: *Even though $\mathcal{S}$ does not have any knowledge about k_{i_c} used by $Stor_{i_c}$, he can easily understand if each $Stor_{i_c}$ used correct input k_i by computing the OPRF locally using k_i in the database. Assume $\alpha + \beta$ adversarial storage providers employed incorrect values. Then, $\mathcal{S}$ also sends exactly $\alpha + \beta$ incorrect values to $\mathcal{TP}$. If $\alpha + \beta \geq n - t + 1$ meaning that less than threshold-many correct values were employed, the response will be invalid in the ideal world to the ideal login server, as well as the real response. On the other hand, if at least t values were correct in the real protocol ($\alpha + \beta < n - t + 1$), responses in ideal and real worlds will both be valid.*

5. *$\mathcal{S}$ will not receive anything from $\mathcal{TP}$, and hence halts.*

Claim 2 *The view of adversary $\mathcal{A}$, controlling all n-storage providers $\mathcal{SP}_c$, in his interaction with the simulator $\mathcal{S}$ is indistinguishable from the view of his interaction with a real honest party.*

Proof 4 *$\mathcal{S}$ acts differently while sending ℓ-many shares c'_i calculated based on randomly chosen pwd' instead of sending actual c_i (sent by $\mathcal{TP}$) calculated based on actual password pwd and executing the OPRF with the $Stor_{i_c}$ using the password pwd' chosen randomly because $\mathcal{S}$ does not have the correct password. If $\mathcal{A}$ can distinguish this* *behavior, then we can construct another adversary $\mathcal{A}'$ which breaks either the OPRF construction (as in Theorem 1) or password based encryption scheme.*

If adversarial storage providers can distinguish the simulator, who employs ℓ random secret shares and $n - \ell$ actual shares during the registration, from the real user, than it can distinguish actual secret share c_i based on pwd from chosen random share c'_{i_c} based on pwd', that can be used to break the security of the underlying encryption scheme. Moreover, $\mathcal{A}$ can compute $s_{i_c} \leftarrow Dec_{F_{k_{i_c}}(pwd^*)}(c_{i_c})$ for each pwd^* in the dictionary, then compute the threshold secret sharing reconstruction algorithm to reconstruct the $rnd^* \leftarrow TSSRecon(s_1, s_2, ..., s_n)$. For $\mathcal{A}$ to verify if rnd^* (and hence pwd^*) is correct, he needs to have actual verification information $vInfo = H(rnd||ls)$ to compare, which he does not have, since only the login server has that information.

5.3 Further Analysis

Phishing protection: We consider a strong phishing attack with man-in-the-middle between the user and the login server during authentication (not registration). This means, the user registered with a legitimate server with ls (e.g. ls = paypal.com), but now is trying to authenticate with an attacker with ls' (e.g. ls'=paypat.com). Therefore, during registration, the user computed $storUID_i \leftarrow H(userID||ls)$, but now for authentication, $storUID'_i \leftarrow H(userID||ls')$ values are computed instead. Thus, honest storage providers will not proceed with the OPRF protocol if a phishing domain ls' is used. Even when all storage providers are corrupted by the phishing attacker and the correct rnd is obtained, remember that the original registered $vInfo \leftarrow H(rnd||ls)$, whereas during attack, the user will send $vInfo' \leftarrow H(rnd||ls')$ to the attacker. This means the phishing/man-in-the-middle attacker cannot authenticate with the original login server on the user's behalf. Furthermore, because of the security of rnd, the adversary cannot obtain any information about the user password, unless threshold-many storage providers are also corrupted.

Handling different domains of the same login server: [Ross et al., 2005] suggest an approach that enables recognizing that *amazon.com.de* and *amazon.co.uk* accounts belong to the same login server and one registration is indeed enough. Using the same approach for setting ls values, we can also enable the user to authenticate with any one of the valid domains of the login server.

Remembering the storage providers: The human user is not required to remember the storage providers. There are several easy solutions. As addressed by Camenisch et al. [Camenisch et al., 2014], the user can remember only a few storage providers who can help direct to other storage providers. Alternatively, a browser extension or a mobile device may remember the list of storage providers employed. Finally, if all storage providers in the whole system are employed by all users, such a public list can be employed, and t of them may be contacted by the user for any given authentication attempt. Observe that publicly listing storage providers does not affect cryptographic security. Our ideal model allows the adversary to know all the storage providers. Therefore, their identities are not hidden when protecting against offline dictionary attacks.

5.4 Performance Evaluation of DiSPP Instantiation

In this section, we discuss performance evaluation for the user and storage providers. Since the login server acts the same as current servers, we did not discuss its efficiency. Performance measurement is processed on a standard laptop machine with Intel Core(TM) i7-5600U CPU 2.60 GHz, 8.00 GB RAM, and 64-bit OS. For our implementation, we choose HMAC [Turner, 2008], AES[Daemen and Rijmen, 2013], OPRF in [Jarecki et al., 2016a], and TSS [Shamir, 1979] with various thresholds. Table 5.1 shows the computational performance of the authentication and registration phases. For the registration, the storage providers *do not* compute anything, only receive and store some value. Finally, the user can communicate with the storage providers in parallel, which decreases the network round trip to 1.5 rounds per authentication, which should be added to the login total time in practice.

	User (Reg.)	User(Auth.)	Storage Provider	Login Total
2-5 Threshold	3.6	4.0	3.3	7.3
3-6 Threshold	4.6	5.5	3.9	9.4
5-10 Threshold	7.8	8.4	6.5	14.9

Table 5.1: Performance evaluation of DiSPP instantiation (in milliseconds).

Chapter 6

USABILITY STUDY OF DISPP INSTANTIATION

6.1 Usability Overview

In this chapter, we first describe our methodology, and then present our findings in the next section. In our study, we compared the cloud-based DiSPP against traditional password-based authentication, and the mobile-based DiSPP against two-factor authentication. We implemented our DiSPP instantiation based on cloud-based as a Chrome browser extension that simply asks for username and password. Thus, experience-wise, this is similar to the traditional password-based authentication. We designed three email-branded websites and asked the user study participants to register with and login to these three websites using the browser extension and separately using the traditional approach. We implemented the mobile-based DiSPP protocol of Acar et. al [Acar et al., 2013] as an Android application that employs a challenge-response mechanism using a mobile device, where a short random string is sent by the server during authentication via SMS. This should be familiar to those who used two-factor authentication for online banking, where a bank employs such a random code for authentication purposes and a mobile device is the second factor (in addition to the password). The participants were presented with three online banking type websites, and were asked to register with and login to these websites using the mobile-based DiSPP technique and separately using the two-factor authentication. For two-factor authentication implementation, we used Google authenticator[1] to provide the smart codes the server asks for. Therefore, we conducted these two separate studies:

1. **Study I-** *cloud-based DiSPP with browser extension* and *traditional password authentication:* We implemented our proposed protocols in Section 4 as a

Chrome browser extension.

2. **Study II-** *mobile-based DiSPP* and *two-factor authentication with Google authenticator:* We implemented an Android application to represent the mobile-based DiSPP protocol in [Acar et al., 2013], and SMS is used for the challenge.

The tasks in our studies were pre-determined as explained below, and these tasks were carefully constructed to preserve the reality as much as possible. For our user study, the users did not need any training to use the system as they will not in real life. Our user studies were reviewed and approved by the university ethics committee. We took precautions according to the European Union General Data Protection Regulation [GDP, 2016] and local data protection laws [TR1, 2017, TR2, 2016] to protect personally-identifiable information of the participants.

We measure usability of our proposed DiSPP instantiation described in Section 5 and Acar et al. [Acar et al., 2013] as well as their counterparts by considering different aspects: **effort expectation**, **anxiety**, **behavioral intention to use the system**, **attitude towards using technology**, **performance expectancy**, and **perceived security**.

6.2 Participants

There were 25 participants who attended Study I (browser extension vs. traditional), and 25 other participants attended Study II (mobile vs. two-factor). The participants were composed of graduate and undergraduate level students and faculty from various departments (both technical and non-technical), as well as staff. The demographic can be found in Table 6.1, and the technical information can be found Table 6.2 and Table 6.3. Despite the fact that deciding how many participants are needed for the user study remains vague, [Faulkner, 2003] justifies that twenty users *"can allow the practitioner to approach increasing levels of certainty that high percentages of existing usability problems have been found in the testing."*

6.3 Testing Environment

Our user studies were conducted in the Koç University's Media and Visual Arts Lab. There was an observer in the room who observed the user actions and received feedback from each participant. As a token of appreciation, we gifted each participant with a mug with the logo of our research group on it.

We provided the participants with a ready setup: pre-installed desktop computer[2] and Android mobile phone[3]. We did not enforce the participants to install the browser extension and mobile applications (both Mobile-based DiSPP application and Google Authenticator) from scratch since mobile-based DiSPP mobile application setup is the same as a regular mobile application installation, and cloud-based DiSPP Chrome extension installation is the same as any other browser extension installation. For mobile-based DiSPP, we used our own SIM card and configured our servers to send SMS messages to our number using NEXMO online service; hence, we did not need to collect participants' phone numbers.

We also created our own websites just for the purposes of the study. Three websites created for Study I were framed as *email sites*, and three websites for Study II were framed as *online banking sites*. These choices were intentional: traditional password authentication is commonly used for email type of daily purposes, whereas two-factor authentication is widely employed for online banking. No website had any data; we just created registration and login pages, and displayed success or failure messages. The only information these websites collected were usernames and (hashed) passwords (which were deleted after data evaluation was completed), and success/failure logs, for the purposes of this study. Each participant was allocated a 30 minute time slot.

6.3.1 Measures

Before conducting the study, participants were first asked to complete a demographics and technical background questionnaire, whose data is kept anonymous, where they were given a general idea about single password authentication. In addition to sex, age interval, and education level, the users were also asked about their experience with browser extensions and password managers, and if they have prior knowledge of password security. We then assigned the participants to two different studies, considering an even distribution across groups, i.e. age, sex, educational level. To collect the data for observation, we had two different methods:

Post-questionnaire: Measures from post-questionnaire were 4-point Likert-scale (strongly disagree, disagree, agree, strongly agree)[4]. Participants answered 23 questions per phase (e.g., 23 questions for traditional password-based authentication and 23 questions for cloud-based DiSPP browser extension in Study I). We followed the standard questions in [Venkatesh et al., 2003] because it is a commonly used standardized questionnaire measuring system usability, and added single-password specific questions ourselves to measure the perceived security. The questions in the post-questionnaire formed six sets that considered different aspects of the systems: **effort expectation**, **anxiety**, **behavioral intention to use the system**, **attitude towards using technology**, **performance expectancy**, and **perceived security** (see Table 6.4 and Table 6.5 for questions and groups). Since we requested participants to evaluate the systems using 4-point Likert-scale (strongly disagree, disagree, agree, strongly agree), we first converted the responses to their numerical values from 1 to 4. For each aspect, we then calculated means, standard deviations, and t-test values based on the numerical values of users' responses. Dependent t-test (paired t-test), which is common in usability studies on password authentication systems [Chiasson et al., 2006, Karole et al., 2010, McCarney et al., 2012], is applied to compare the systems in each study, since each participant tested two systems per study (either cloud-based DiSPP and traditional passwords, or mobile-based DiSPP and two-factor authentication).

Comments to the observer: At the end of the study, the observer had a discussion with the participants about each system they tested, where the users freely commented about their feelings and concerns.

6.3.2 Testing Procedure

Before the participants started the study, written signed consent of the participants were taken. We did not collect personally-identifiable information unnecessarily, and used the names only for the consent forms, which are not linkable to the anonymous post-questionnaires and comments.

Tasks of Study I: Each participant registered with three different websites (*e.g.,* Mail A) separately using the traditional approach and the cloud-based DiSPP Chrome extension [5]. The order of which password authentication system a participant started with was random, where either they began with the conventional approach and then DiSPP Chrome Extension, or vice versa. After registration, they logged in to the three websites in random order (as we pre-determined). If a participant failed to login to a website three times, we accepted it as a login failure and asked the user continue to login to the next website. This represented a realistic scenario where if a user enters an incorrect password three times, the user is asked to go through a CAPTCHA process or the user's account is blocked temporarily.

The users were explicitly told to behave as in their regular life as much as they could. For that reason, some participants wrote down each password they created during the test of traditional password authentication on a piece of paper (that they took away after the test), as they noted this is how they remember their passwords in their regular life. More specifically, the tasks of Study I are described as follows:

Traditional Password Authentication Registration: The user

1. selects a strong password with at least eight characters containing at least one of each category: lower case and upper case letters, numerical character, and special character,

   Remark: The users are asked to choose a different password for each website since ideally the users are expected not to use a password for more than one website for security. The username may be chosen the same or differently for each website.

2. types her username and the password,

3. confirms the password (see Figure 3(d)),
4. presses the signup button,
5. is informed whether the registration is successful or not (e.g., password confirmation does not match).

Traditional Password Authentication Login: The user

1. types username and password (see Figure 3(e)),
2. presses the login button,
3. is informed whether the login attempt is successful or not.

DiSPP Chrome Extension Registration: The user

1. selects a strong password with at least eight characters containing at least one of each category: lower case and upper case letters, numerical character, and special character,

   Remark: The participant is told to use the same password during all three account registrations.
2. opens the extension by clicking its button next to the address bar,
3. types her username and the password into the extension (see Figure 1),
4. presses the registration button,
5. is informed whether the registration is successful or not.

DiSPP Chrome Extension Login: The user

1. opens the DiSPP extension,
2. types her username and password using the extension (see Figure 1),
3. presses the login button,
4. is informed whether the login attempt is successful or not.

Tasks of Study II: Each participant was required to register to three different websites (*e.g.,* Bank A) using the two-factor authentication approach and the DiSPP mobile application. The order of which password authentication system a participant started with was random, where either they began with two-factor authentication and then continued with mobile-based DiSPP, or vice versa. After each registration, they logged in to the websites in random order. If a participant failed to login to a website three times, we accepted it as a login failure and asked user continue to login to the next website. More specifically, the tasks are described as follows:

Two-Factor Password Authentication Registration: The user

1. selects a strong password with at least eight characters containing at least one of each category: lower case and upper case letters, numerical character, and special character,

   Remark: The users are asked to choose a different password for each website since ideally the users are expected not to use a password for more than one site. The username can be chosen the same or different for each website.
2. types her username and password (see Figure 3(a)),
3. presses the signup button,
4. opens Google Authenticator application on the phone (see Figure 3(b)),
5. scans the QR code shown on the website (see Figure Figure 3(c)),
6. types the application-generated six-digit numerical code to the site and clicks the send button,
7. is informed whether the registration is successful or not.

Two-Factor Password Authentication Login: The user

1. types her username and password on the server site,
2. opens the Google authenticator application on the phone,
3. types the application-generated six-digit numerical code to the site,
4. is informed whether the login attempt is successful or not.

DiSPP Mobile Registration: The user

1. selects a strong password with at least eight characters containing at least one of each category: lower case and upper case letters, numerical character, and special character,

   Remark: The participant is told to use the same password during all three account registrations.
2. types her username (see Figure 2(a)),
3. presses the signup button,

4. opens mobile-based DiSPP application on the phone as it is told on the site,
5. clicks the register button on mobile-based DiSPP application (see Figure 2(b))
6. scans the QR code shown on the website (see Figure 2(c)),
7. types her password on the mobile application (see Figure 2(d)),
8. clicks the register button on the mobile application,
9. is informed whether the registration is successful or not.

DiSPP Mobile Login: The user

1. types the username on the website (see Figure 2(e)),
2. is shown on the website that an SMS code is sent to the mobile phone and should open DiSPP mobile application,
3. opens the mobile application and clicks the login button,
4. types the single password on the mobile application (see Figure 2(f)),
5. types the 8-digit alphanumeric code displayed by the mobile application to the website (see Figure 2(h)),

   Remark: The application automatically retrieves the SMS code and generates the code for the user; the user did not need to type SMS into the application (see Figure 2(g)).
6. is informed whether the login attempt is successful or not.

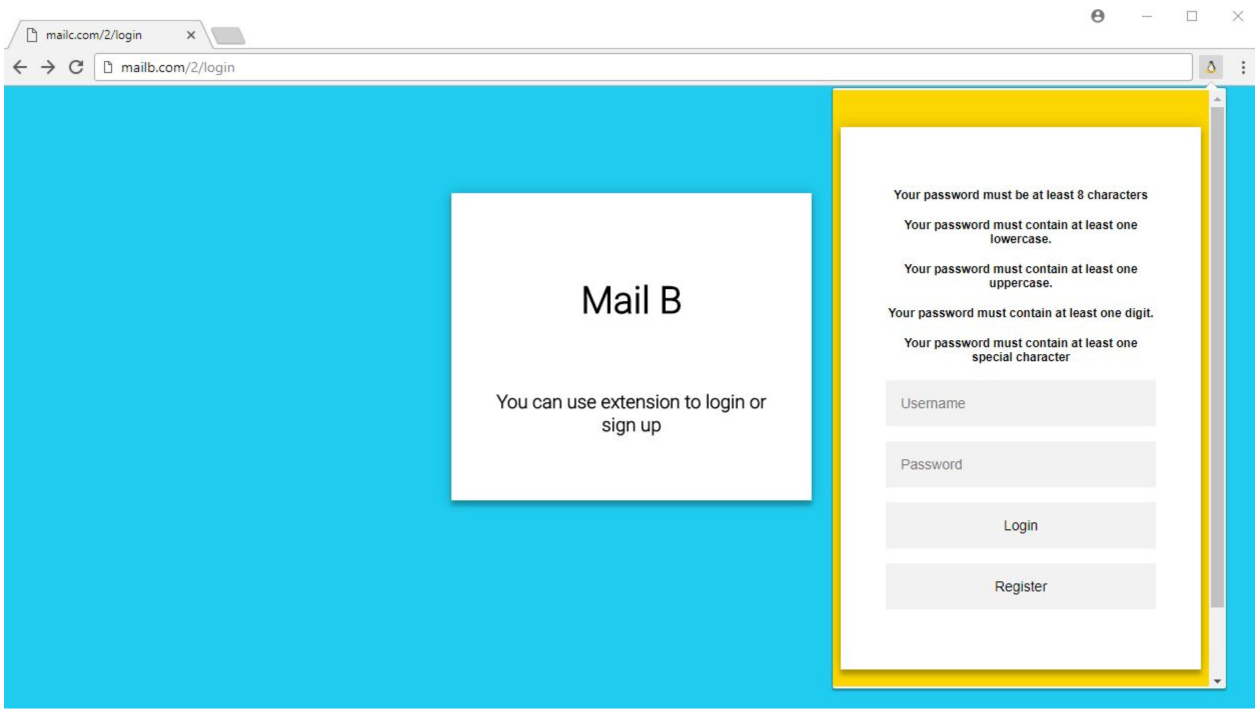
6.4 Results

Below, we provide a comparative analysis for each study based on 1) the statistical significance and quantitative response data such as mean values (see Table 6.10 and Table 6.11), 2) the range of responses (see Table 6.8 and Table 6.9)[6], 4) number of login attempts until success or failure (Tables 6.6 and 6.7), and 5) observations from users' comments.

6.4.1 The Usability of Cloud-based DiSPP

Considering the range of responses [7], the majority of participants (more than 50% per question) agreed (or strongly agreed) that cloud based DiSPP is easy to use, useful,

Figure 6.1: Cloud-based DiSPP registration and login screenshots.



Mail B

You can use extension to login or sign up

Your password must be at least 8 characters

Your password must contain at least one lowercase.

Your password must contain at least one uppercase.

Your password must contain at least one digit.

Your password must contain at least one special character

trustworthy, and not intimidating to use[8], as well as they have a positive attitude towards and intention to using this system (see Table 6.8). Indeed, this holds for every question except *"If I use the system, I will increase my chances of getting a raise"*, which received low agreement for all the systems we tested, as the participants did not link password security to their salaries.

As for the usability of cloud-based DiSPP compared to traditional password authentication, we found significant differences in terms of three dimensions: **effort expectancy**, **attitude towards using technology**, and **performance expectancy**. There was no significant difference between cloud-based DiSPP and traditional password authentication regarding **anxiety** ($t(24) = 2.03$ and $p = 0.053$), **behavioral intention to use the system** ($t(24) = 0.84$ and $p = 0.40$), and **perceived security** ($t(24) = 10.64$ and $p = 0.52$).

Effort Expectancy: DiSPP extension was easier to use (less effort) compared to traditional password authentication ($t(24) = 2.09$ and $p = 0.04$). Anecdotal observation supports this statistic, since the participants only needed to remember a single password, rather than several different passwords. A participant said that she feels under pressure while creating a password requiring to follow certain rules in her daily life. Consequently, she commented that she was using the same password for all her accounts (which is insecure in the traditional approach). She stated that this was because she would need to remember the passwords during login and it is hard for her to remember all these complicated passwords (see Section 6.4.3 for further observations).

Attitude towards using technology: Participants had a significantly more positive attitude towards cloud-based DiSPP compared to traditional password authentication ($t(24) = 3.82$ and $p = 0.0008$). 87% of the participants wanted to use the cloud-based DiSPP system because of its functionality. A participant asked when we planned to launch the system publicly. The same participant stated that he generally reseted his password while logging in to a site because he always forgot or exceeded the number of attempts to enter the correct password in his daily life.

Performance Expectancy: Cloud-based DiSPP performed significantly better than traditional password authentication ($t(24) = 3.27$ and $p = 0.003$). The majority of the participants commented that cloud-based DiSPP system was very useful and they could use the system in their real life. These participants commented that they liked the idea of holding only one single password since recalling passwords took some time and it got worse if they tried to login to a site that they did not login for a while.

In the comments to the observer, the users stated that cloud-based DiSPP was too "simple" for online banking. They expected a second authentication factor and a more complex system for online banking. Interestingly, 63% of the comments stated that if it is hard for a user to login, it should be hard for attackers as well. This observation is also important to understand users' point of view against cryptographic systems.

80% of the participants stated that they did not know if the extension was really performing as it was supposed to do (e.g., running the cryptographic protocols, not storing passwords). They commented that they trust this system more than traditional password authentication; however, they felt nervous because of the idea that the extension might have stored their passwords.

Another interesting point was that 52% of participants wanted to use the cloud-based DiSPP for their “unimportant” accounts, where they were okay if the password was compromised. These participants also stated that they had a hierarchy based on the sites they were creating account. They grouped the sites in categories and they created the password based on the category. For instance, they employ separate passwords for separate categories such as e-mail accounts (though the same password is employed for all email accounts), gaming accounts (same password for all gaming accounts, but different from the email account passwords), banking accounts (different from email and gaming passwords), etc. This way, they commented, if the password used for gaming accounts was compromised, it would be fine since that password is not used for “important” sites.

Success/Failure Rates: We measured that 88% of the participants successfully remembered their passwords at the first attempt using cloud-based DiSPP. 4% of the participants remembered their passwords at their second attempt and 3% of the participants remembered their password at their third trial. 5% of the participants did not remember their passwords within 3 trials and were counted as failed attempts. Overall, 95% of the participants accomplished to login while 5% of the participants failed to login. The average number of attempts by a user is 1.09 (see Table 6.6).

In comparison, for traditional authentication, we measured that 68% of the participants successfully remembered their passwords at their first attempt. 14% of the participants remembered their passwords at their second attempt and 16% of the participants remembered their password at their third trial. 2% of the participants did not remember their passwords within 3 trials and were counted as unsuccessful login attempt. Overall, 98% of the participants accomplished to login while 2% of the participants failed to login. The average number of attempts by a user is 1.44 (see Table 6.6).

Thus, both systems had similar overall failure rates (though surprisingly cloud-based DiSPP failure rate was slightly higher). However, the participants struggled to remember their passwords, which can be observed from the average number of attempts: 1.44 for traditional password authentication compared to 1.09 for cloud-based DiSPP. Hence, we deduce that cloud-based DiSPP fulfilled what it promises about easier recall of passwords.

6.4.2 The Usability of Mobile-based DiSPP

Considering the range of responses, the majority of participants (more than 50% per question) agreed (or strongly agreed) that mobile-based DiSPP is easy to use, useful, trustworthy, and not intimidating to use, as well as they have a positive attitude towards and intention to using this system (see Table 6.8). This holds for all 20 questions out of 23 asked. Except the salary raise note as in Study I, the only other two questions that the majority did not agree were "I plan to use the system in the next 6 months" and "Using the system increases my productivity", for both of which both the mobile-based DiSPP and two-factor authentication responses are almost identical.

As for the usability of mobile-based DiSPP compared to two-factor authentication, we found significant differences in terms of three dimensions: **anxiety**, **perceived security**, and **attitude towards using technology**. There was no significant difference between mobile-based DiSPP and two-factor authentication regarding **effort expectancy** ($t(24) = 1.10$ and $p = 0.28$), **behavioral intention to use the system** ($t(24) = 0.00$ and $p = 1.00$), and **performance expectancy** ($t(24) = 1.04$ and $p = 0.30$).

Anxiety: Mobile-based DiSPP was less threatening than two-factor authentication ($t(24) = 2.77$ and $p = 0.01$). 72% of the participants stated that they were not worried while using mobile-based DiSPP because they typed the password on their mobile phone (conceived as a personal device) rather than the website. 96% of the participants were not scared to lose a lot of information by hitting the wrong key in mobile-based DiSPP. A participant explained that there was nothing to worry, since he did not give any important information to websites.

Perceived Security: More than 80% of the participants felt secure while using mobile-based DiSPP based on the range of responses. The users trusted mobile-based DiSPP more than they trust two-factor authentication ($t(24) = 3.25$ and $p = 0.003$), including all sub-statements. 80% of the participants commented that typing the password on mobile device (conceived as a personal item) made them feel more secure, whereas they needed to type their passwords on websites in standard two-factor authentication.[9] One of the participants commented that seeing all works (computations) carried out on the mobile device made her feel more secure, and she felt to have the control of her password security, since she could see the steps (e.g., SMS challenge, smart code generated). Another participant pointed that he was aware of the danger if he used the same password for multiple websites (as 56% of participants agreed that they would feel insecure to use the same password for multiple websites in traditional password authentication).

Furthermore, 90% of the participants stated that mobile-based DiSPP provided a better security for online banking, and users felt secure in the online banking scenario because it was "complex" enough. Interestingly, the users essentially agreed that a "simple" single password solution without the mobile device (i.e., cloud-based DiSPP) does not feel secure enough for banking scenarios, but it is efficient for daily use, and a "complex" solution using the mobile device (i.e., mobile-based DiSPP) feels secure for banking since the password is typed on the phone, whereas it is inefficient for daily use (e.g., for e-mail and other frequently accessed sites).

Attitude towards using technology: Mobile-based DiSPP performed statistically significantly better compared to two-factor authentication ($t(24) = 2.71$ and $p = 0.01$), including all sub-statements. Similar to cloud-based DiSPP, the users are required to remember only one single password and used it all the time, while they need to remember each one of the passwords in the two-factor approach. One of the participants stated that she found two things she wanted at the same time, which are usability (easing her job by remembering one password) and more security (via employing a personal device and challenge).

Even though mobile-based DiSPP and two-factor authentication did not have a significant difference regarding **effort expectation**, more than 80% of the participants agreed that mobile-based DiSPP was easy to use. The users reported a high satisfaction with mobile-based DiSPP, even though the steps followed in mobile-based DiSPP were a little bit more complex (such as typing 8-character alphanumerical code to the site, while they type 6-digit numerical code in the two-factor authentication). Most of the users found that the mobile-based DiSPP is easy to learn, and they were fine with the steps they need to follow, since it was for online banking. Mobile-based DiSPP system was found unproductive for email type daily purposes due to its complexity, while it was considered more secure by the participants.

Success/Failure Rates We measured that 100% of the participants successfully remembered their passwords without any trials using mobile-based DiSPP. Therefore, the average number of password attempts by a user is 1 (see Table 6.7). However, we measured a 20% overall login failure rate, due to the participants' inability to type the correct authentication code within 3 attempts. This indicates that simpler smart codes should be employed in future systems and studies.

For two-factor authentication, we measured that 82% of the participants successfully remembered their passwords at the first attempt, out of which 91% could enter the authentication code (generated by Google Authenticator) at their first attempt and 9% at their second attempt. 5% of the participants remembered their passwords at their second attempt, out of which 80% could enter the authentication code at their first attempt and 20% at their second attempt. 4% of the participants remembered their passwords at their third attempt, out of which 67% could enter the authentication code at their first attempt and 33% at their second attempt. 9% of the participants did not remember their passwords within the first three attempts which resulted in a login failure. Overall, 91% of the participants accomplished to login while 9% of the participants failed to login. The average number of password attempts by a user is 1.17 (see Table 6.7).

We conclude that for both two factor authentication and mobile DiSPP, the participants had high login success rates. Using mobile-based DiSPP, the participants did not have problems with the password, but they had issues with the smart codes. Using two-factor authentication, the users did not have problems with the authentication codes, but they had issues remembering the password. We deduce that simpler smart codes should be employed in such systems, as they may make things as bad as remembering passwords.

6.4.3 Common Distributed Single Password Protocol Observations

The participants mentioned valuable statements and discussed their habits while creating, securing, and recalling the passwords. [Stobert and Biddle, 2018] observes how users manage, create, and secure their passwords and points out some challenges users face such as password creation (with the intent of reuse) and recall in traditional password authentication schemes. We observed how an DiSPP method (whether cloud-based or mobile-based) overcomes some of the challenges users face.

90% of the study participants were aware of password security. 85% of the comments (out of 45) stated that the participant always struggled while coming up with a password satisfying the requirements (e.g., at least one lowercase and one uppercase letter, and a number). The participants usually came up with a password after a number of trials. Once they created it, remembering the password was another struggle they bear. Thus, they created their own way to recall the passwords. More than 50% of participants noted that they wrote down their passwords to remember. One of the users commented that he stored password reminders (as hints helping him to recall the passwords) in a file where he emphasized that anyone who had the file could not learn the passwords. When we questioned why he needed this storage, he responded that it is hard for him to remember the password for some sites he rarely used and he came up with this solution. However, even this solution did not stop him from re-using the same password for multiple sites.

While there is a functionality to reset a password in traditional approaches, a participant found it is cumbersome, since password reset procedure requires steps such as logging in to a backup e-mail, which requires remembering another password, or entering all necessary information (such as security questions) to reset. Another participant shared his experience when he lost the paper where he noted a password for a site and wanted to reset the password. Unfortunately, he needed to follow a long official password reset procedure because of system requirements (e.g., personal application was required and he waited for a week). He stated that everything would be easier if he could use a secure DiSPP system that minimizes password remembering problems. Similar comments support that DiSPP systems are easing the burden on users by requiring them to remember only one password (in addition to the cryptographic benefits they provide such as provable security against offline dictionary attacks). In the light of these comments, we recommend that the DiSPP systems should investigate how a secure single password reset can be efficiently carried out.

Another frustration shared by 52% of the users was that they would use the DiSPP system and trust it if it is commonly used and advertised by a "trusted" authority (rather than university researchers) such as Facebook, Google, etc. One of the participants said that *"I feel secure while I am using Whatsapp, since Whatsapp is employed for secure messaging. They use something like encryption."* The participant was not aware of the cryptographic scheme employed in Whatsapp and had no idea what it was, but stated that it "feels" secure since Whatsapp was widely advertised and employed. While this idea might require further research, users may feel more secure when a new system is collectively used.

Our user studies concluded that DiSPP systems provide usability benefits. The main reasoning is that it is not convenient to expect users to create different passwords for each website and remember them. While this approach would be secure, it is not usable. On the other hand, DiSPP systems enable single password re-use securely. Also, considering the discussions on security and usability, there might be an inverse relationship between perceived security and ease of use, since cloud-based DiSPP was found better for daily use, whereas mobile-based DiSPP was found more secure for online banking. This interpretation is worth exploring for future research.

6.5 Post-Questionnaire Results

Table 6.1: Responses of the participants regarding demographic information.

	Study I	**Study II**
Sex		
Male	12	11
Female	13	14
Age Interval		
18-25	6	6
25-35	13	15
35-45	2	1
45-55	3	1
55+	1	2
Education Level		
Post-Graduate	10	10
Masters	7	7
Bachelor	5	6
High School	2	2
Primary school or under	1	0
How often do you use your mobile device?		
So often (Daily)	23	24
Few times in a day	1	1
Weekly	1	0

Table 6.2: Responses of the participants regarding technical information (Part 1).

	Study I	**Study II**
How often do you use mobile banking?		
Daily	5	4
Weekly	11	11
Monthly	5	5
Rarely	0	0
Never	4	5
How often do you use online banking?		
Daily	4	4
Weekly	5	9
Monthly	10	7
Rarely	4	3
Never	2	2
How often do you change your password?		
Weekly	1	1
Monthly	2	4
Every 3 months	2	4
Every 6 months	5	2
Once a year	1	0
If I have to	14	14

Table 6.3: Responses of the participants regarding technical information (Part 2).

	Study I	**Study II**
Do you have prior knowledge of password security?		
I heard from news, social media etc.	18	16
I had a course	3	6
Not me but someone I know had experience	4	3
Have you ever used a browser extension?		
Yes	16	16
No	5	4
Never Heard	4	5
Have you ever used a password manager?		
Yes	4	4
No	16	17
Never Heard	5	4

Table 6.4: Post-questionnaire form questions asked to the participants. The form employed a 4-point scale, where 1=Strongly Disagree, 2=Disagree, 3=Agree, and 4=Strongly Agree. The group names and questions' abbreviated numbering does not exist in the actual forms the participants filled; only the questions were shown.

Effort Expectation (EE)
(EE1) My interaction with the system would be clear and understandable
(EE2) It would be easy for me to become skillful at using the system
(EE3) I would find the system easy to use
(EE4) Learning to operate the system is easy for me
Anxiety (A)
(A1) I feel apprehensive (worried) about using the system
(A2) It scares me to think that I could lose a lot of information using the system by hitting the wrong key
(A3) I hesitate to use the system for fear of making mistakes I cannot correct
(A4) The system is somewhat intimidating to me
Behavioral intention to use the system (BIU)
(BIU1) I intend to use the system in the next 6 months
(BIU2) I predict I would use the system in the next 6 months
(BIU3) I plan to use the system in the next 6 months

Table 6.5: Post-questionnaire form questions asked to the participants. The form employed a 4-point scale, where 1=Strongly Disagree, 2=Disagree, 3=Agree, and 4=Strongly Agree. The group names and questions' abbreviated numbering does not exist in the actual forms the participants filled; only the questions were shown.

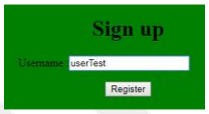
Attitude towards using technology (ATUT)
(ATUT1) Using the system is a good idea
(ATUT2) The system makes work more interesting
(ATUT3) Working With the system is fun
(ATUT4) I like working with the system
Performance Expectancy (PE)
(PE1) I would find the system useful in my job
(PE2) Using the system enables me to accomplish tasks more quickly
(PE3) Using the system increases my productivity
(PE4) If I use the system, I will increase my chances of getting a raise
Perceived Security (PS)
(PS1) I trust my password with this system
(PS2) I feel secure using this system for daily use
(PS3) I feel secure using this system for online banking
(PS4) I feel secure reusing the same password for multiple sites employing this system

Table 6.6: Cloud-based DiSPP (DiSPP Cloud) and traditional password authentication (Traditional) percentage distribution of password attempts (trials) by the participants to login. μ: mean, σ: standard deviation.

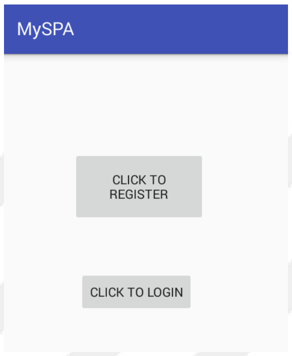
	Login Trial					
	μ	σ	1 Trial (%)	2 Trials (%)	3 Trials (%)	Fail (%)
DiSPP Cloud	1.09	0.38	88	4	3	5
Traditional	1.44	0.73	68	16	14	2

Figure 6.2: Mobile-based DiSPP registration and login screenshots.

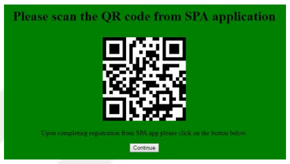
(a) Server site registration page



(b) Mobile application main page

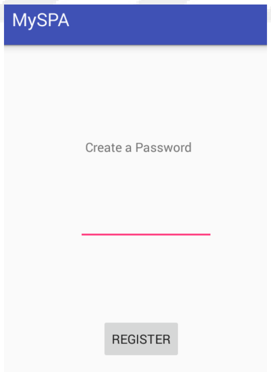


(c) Registration QR code

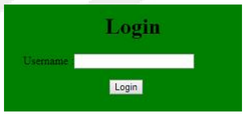

Please scan the QR code from SPA application

Upon completing registration from SPA app please click on the button below.

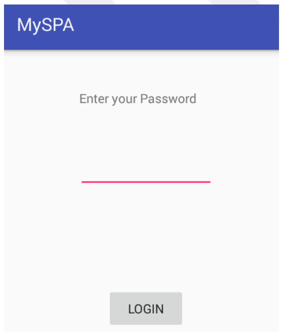
(d) Password creation



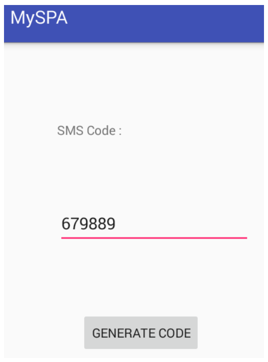
(e) Login page



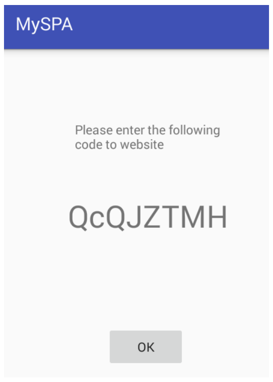
(f) Password entrance



(g) SMS code



(h) Generated smart code



Please enter the following code to website

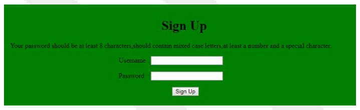
QcQJZTMH

Table 6.7: Mobile-based DiSPP (DiSPP Mobile) and two-factor authentication (Two Factor) percentage distribution of password attempts (trials) by the participants to login. μ: mean, σ: standard deviation.

	Login Trial					
	μ	σ	1 Trial (%)	2 Trials (%)	3 Trials (%)	Fail (%)
DiSPP Mobile	1.00	0	100	0	0	0
Two Factor	1.17	0.49	82	5	4	9

Figure 6.3: Two-factor and traditional authentications registration and login screenshots.

(a) 2FA Registration Password creation



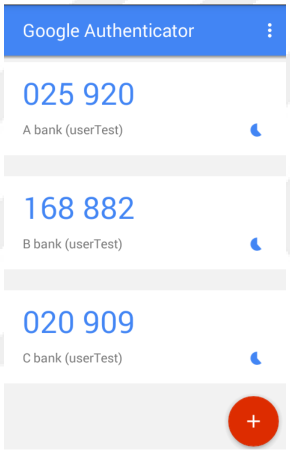
Sign Up

Your password should be at least 8 characters,should contain mixed case letters,at least a number and a special character.

Username

Password

(b) Google authenticator



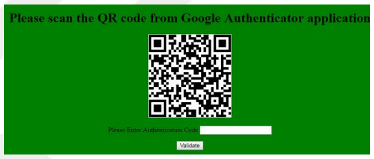
Google Authenticator

025 920
A bank (userTest)

168 882
B bank (userTest)

020 909
C bank (userTest)

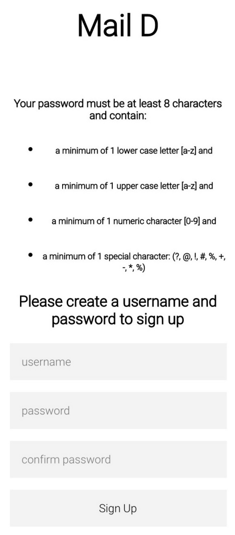
(c) Google authenticator registration via QR code



Please scan the QR code from Google Authenticator application

Please Enter Authentication Code

(d) Traditional password authentication and registration page.



Mail D

Your password must be at least 8 characters and contain:

- a minimum of 1 lower case letter [a-z] and
- a minimum of 1 upper case letter [a-z] and
- a minimum of 1 numeric character [0-9] and
- a minimum of 1 special character: (?, @, !, #, %, +, -, *, %)

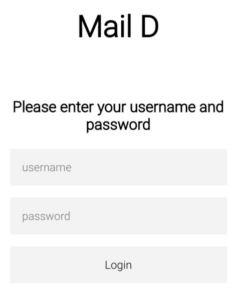
Please create a username and password to sign up

username

password

confirm password

(e) Traditional password authentication login page.



Mail D

Please enter your username and password

username

password

Table 6.8: Post-Questionnaire percentage distribution for cloud-based DiSPP and traditional password authentication

Cloud-based DiSPP																							
	EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
Strongly Disagree	4	8	4	4	40	20	36	32	4	8	8	0	4	0	0	0	4	16	20	4	8	12	12
Disagree	8	8	8	0	48	60	44	56	20	28	32	20	40	28	20	16	24	24	56	24	28	32	24
Agree	32	28	28	44	8	12	20	12	68	60	56	52	56	68	68	52	48	36	24	56	48	36	44
Strongly Agree	56	56	60	52	4	8	0	0	8	4	4	28	0	4	12	32	24	24	0	16	16	20	20
Traditional Password Authentication																							
	EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
Strongly Disagree	0	0	12	4	24	16	16	12	8	16	16	12	36	40	28	16	36	44	36	12	12	24	16
Disagree	16	16	40	36	48	48	40	56	36	36	32	44	44	44	36	48	44	44	52	20	24	20	40
Agree	52	60	20	28	28	20	40	24	48	40	40	36	12	8	28	24	12	4	8	48	52	40	32
Strongly Agree	32	24	28	32	0	16	4	8	8	8	12	8	8	8	8	12	8	8	4	20	12	16	12

Table 6.9: Post-Questionnaire percentage distribution for mobile-based DiSPP and two-factor authentication

	EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
Mobile- based DiSPP																							
Strongly Disagree	4	4	0	4	20	36	24	24	0	0	4	0	4	0	0	8	16	12	12	0	0	0	4
Disagree	16	8	12	12	64	60	64	56	40	48	52	20	12	28	24	28	32	40	72	12	12	24	12
Agree	52	56	56	48	12	4	12	16	48	36	32	48	52	44	44	28	32	36	16	64	64	40	52
Strongly Agree	28	32	32	36	4	0	0	4	12	16	12	32	32	28	32	36	20	12	0	24	24	36	32
Two Factor Authentication																							
	EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
Strongly Disagree	0	0	0	0	4	16	12	8	4	4	4	4	24	20	12	16	32	16	20	16	8	8	28
Disagree	4	8	16	0	72	56	68	60	32	36	52	36	20	16	28	28	32	36	68	40	32	40	28
Agree	60	68	60	52	20	16	16	28	52	48	32	52	48	56	44	40	28	36	8	32	48	36	32
Strongly Agree	36	24	24	48	4	12	4	4	12	12	12	8	8	8	16	16	8	12	4	12	12	16	12

Table 6.10: Post-Test Questionnaire and results for user studies on cloud-based DiSPP (DiSPP Cloud), traditional password authentication (Traditional), mobile-based DiSPP (DiSPP Mobile) and two-factor authentication (Two-Factor). Scores are out of the 4-point Likert scale employed. μ: mean, σ: standard deviation, t: t-statistic, and p: significance. Degrees of freedom are 24.

	DiSPP Cloud		**Traditional**		**t-test**		**DiSPP Mobile**		**Two-Factor**		**t-test**	
	μ	σ	μ	σ	t	p	μ	σ	μ	σ	t	p
EE	3.40	0.70	2.94	0.71	2.09	**0.04**	3.14	0.55	3.26	0.41	1.10	0.28
EE1	3.40	0.82	3.16	0.69	1.10	0.28	3.04	0.79	3.32	0.56	1.66	0.10
EE2	3.32	0.95	3.08	0.64	0.94	0.35	3.16	0.75	3.16	0.55	0.00	1.00
EE3	3.44	0.82	2.64	1.04	2.61	**0.01**	3.20	0.65	3.08	0.64	0.76	0.44
EE4	3.44	0.71	2.88	0.93	2.22	**0.03**	3.16	0.80	3.48	0.51	2.13	**0.04**
A	1.87	0.51	2.25	0.65	2.03	0.05	1.89	0.43	2.22	0.54	2.77	**0.01**
A1	1.76	0.78	2.04	073	1.13	0.27	2.00	0.71	2.24	0.60	1.29	0.20
A2	2.08	0.81	2.36	0.95	1.02	0.31	1.68	0.56	2.24	0.88	3.21	**0.003**
A3	1.84	0.75	2.32	0.82	2.21	**0.03**	1.88	0.60	2.12	0.67	1.23	0.22
A4	1.80	0.65	2.28	0.79	2.61	**0.01**	2.00	0.76	2.28	0.68	1.57	0.12
BIU	2.65	0.60	2.48	0.77	0.84	0.40	2.64	0.64	2.64	0.70	0.00	1.00
BIU1	2.80	0.65	2.56	0.77	1.00	0.32	2.72	9.68	2.72	0.74	0.00	1.00
BIU2	2.60	0.71	2.40	0.87	0.92	0.36	2.68	0.75	2.68	0.75	0.00	1.00
BIU3	2.56	0.71	2.48	0.92	0.73	0.34	2.52	0.77	2.52	0.77	0.00	1.00

Table 6.11: Post-Test Questionnaire and results for user studies on cloud-based DiSPP (DiSPP Cloud), traditional password authentication (Traditional), mobile-based DiSPP (DiSPP Mobile) and two-factor authentication (Two-Factor). Scores are out of the 4-point Likert scale employed. μ: mean, σ: standard deviation, t: t-statistic, and p: significance. Degrees of freedom are 24.

	DiSPP Cloud		**Traditional**		**t-test**		**DiSPP Mobile**		**Two-Factor**		**t-test**	
	μ	σ	μ	σ	t	p	μ	σ	μ	σ	t	p
ATUT	2.82	0.39	2.08	0.76	3.82	**0.0008**	3.08	0.66	2.55	0.78	2.71	**0.01**
ATUT1	3.08	0.70	2.40	0.82	2.88	**0.01**	3.12	0.73	2.64	0.70	2.61	**0.01**
ATUT2	2.52	0.59	1.92	0.91	2.68	**0.01**	3.12	0.78	2.40	0.96	2.97	**0.006**
ATUT3	2.76	0.52	1.84	0.90	3.99	**0.001**	3.00	0.76	2.52	0.92	2.00	0.05
ATUT4	2.92	0.57	2.16	0.94	3.26	**0.003**	3.08	0.76	2.64	0.91	1.74	0.09
PE	2.70	0.55	1.95	0.72	3.27	**0.003**	2.50	0.72	2.27	0.76	1.04	0.30
PE1	3.16	0.69	2.32	0.90	3.12	**0.004**	2.92	1.00	2.56	0.96	1.36	0.18
PE2	2.92	0.81	1.92	0.91	3.33	**0.002**	2.56	1.00	2.12	0.97	1.38	0.17
PE3	2.68	1.03	1.76	0.88	2.91	**0.007**	2.48	0.87	2.44	0.92	0.16	0.87
PE4	2.04	0.68	1.80	0.76	1.36	0.18	2.04	0.54	1.96	0.68	0.41	0.67
PS	2.73	0.72	2.57	0.81	0.64	0.52	3.12	0.64	2.48	0.81	3.25	**0.003**
PS1	2.84	0.94	2.76	0.93	0.30	0.76	3.12	0.60	2.40	0.91	3.39	**0.002**
PS2	2.72	0.95	2.64	0.86	0.31	0.75	3.12	0.60	2.64	0.81	2.38	**0.02**
PS3	2.64	0.84	2.48	1.05	0.51	0.60	3.12	0.78	2.60	0.87	2.31	**0.02**
PS4	2.72	0.75	2.40	0.91	1.01	0.31	3.12	0.78	2.28	1.02	3.67	**0.001**

Notes to Chapter 6

1 Or a hashed version of $vInfo$ can be stored in the database, as usual.

1 Google Authenticator Android app. https://goo.gl/Q4LU7k

2 A desktop computer running 64-bit Windows 8 on Intel Core i7-3770 3.4 GHz CPU and 16 GB RAM.

3 A Samsung Galaxy J1 with Android version 4.4.4.

4 The reason we employed a forced choice scale by eliminating neutral answer is to receive exact response whether a user agrees or disagrees and how much they agree. This type of likert scale is considered by [Allen and Seaman, 2007, Behnke, 2011] to account "exact" responses only.

5 The reason we created our own websites is that DiSPP systems require different authentication mechanisms compared to today's such as message authentication code (MAC) and they cannot be employed without a change at today's websites.

6 Anonymous individual responses can be found in the Appendix for completeness.

7 All the participants' responses for both systems are presented in Appendix B

8 For anxiety questions, "disagree" or "strongly disagree" responses are better, which are what we counted.

9 [Melicher et al., 2016] measures the usability and security of creating and entering textual passwords on mobile devices.

Chapter 7

CONCLUSION

We introduce a framework for distributed single password protocols where a user securely uses her one single password for all her accounts. DiSPP ensures security against offline dictionary attacks as long as the adversary does not corrupt the login server and the storage provider together, offline dictionary attacks will be prevented. We provide novel techniques to ensure that multiple storage providers can be employed. We provided an ideal and real world security definition and presented an ideal-real simulation proof. We further ensure phishing, man-in-the-middle, and honeypot attacks are also thwarted. Our proposed solution employs efficient symmetric key primitives and can easily work with today's hardware, even on mobile devices.

Finally, we conduct user studies on our proposed DiSPP solution which is a cloud-based DiSPP instance against the traditional approach in a daily use scenario, and mobile-based DiSPP instance (based on [Acar et al., 2013]) against two-factor authentication in an online banking scenario. Quantitative and qualitative results support that both DiSPP solutions have usability and security advantages compared to their counterparts. Based on the feedback reported by the participants, we suggest that cloud-based DiSPP solutions should be deployed for daily use, where users wish to login to a site frequently, and mobile-based DiSPP solutions should be deployed for online banking type of settings, where more complicated solutions are expected (at least seemingly more complicated, regardless of the underlying cryptography). Observations also indicate that there is potentially a trade-off between usability and perceived security, which is worth exploring as future work.

BIBLIOGRAPHY

[GDP, 2016] (2016). European Union General Data Protection Regulation 2016/679 (GDPR).

[TR2, 2016] (2016). Turkish Personal Data Protection Law no. 6698.

[TR1, 2017] (2017). Turkish Personal Data Protection Law no. 30224.

[Abdalla et al., 2016] Abdalla, M., Cornejo, M., Nitulescu, A., and Pointcheval, D. (2016). Robust password-protected secret sharing. In *European Symposium on Research in Computer Security*. Springer.

[Acar et al., 2013] Acar, T., Belenkiy, M., and Küpçü, A. (2013). Single password authentication. *Computer Networks*.

[Allen and Seaman, 2007] Allen, I. E. and Seaman, C. A. (2007). Likert scales and data analyses. *Quality progress*.

[Aloul et al., 2009] Aloul, F., Zahidi, S., and El-Hajj, W. (2009). Two factor authentication using mobile phones. In *IEEE/ACS AICCSA*.

[Anderson, 1993] Anderson, R. (1993). Why cryptosystems fail. In *ACM CCS*.

[Bagherzandi et al., 2011] Bagherzandi, A., Jarecki, S., Saxena, N., and Lu, Y. (2011). Password-protected secret sharing. In *Proceedings of the 18th ACM conference on Computer and Communications Security*. ACM.

[Behnke, 2011] Behnke, Andrew O., K. C. (2011). Creating programs to help latino youth thrive at school: The influence of latino parent involvement programs. *Journal of Extension*.

[Belenkiy et al., 2011] Belenkiy, M., Acar, T., Morales, H., and Küpçü, A. (2011). Securing passwords against dictionary attacks. US Patent 9,015,489.

[Bellovin and Merritt, 1992] Bellovin, S. M. and Merritt, M. (1992). Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*. IEEE.

[Benaloh and Leichter, 1990] Benaloh, J. and Leichter, J. (1990). Generalized secret sharing and monotone functions. In *Proceedings on Advances in cryptology*. Springer.

[Bertilsson and Ingemarsson, 1992] Bertilsson, M. and Ingemarsson, I. (1992). A construction of practical secret sharing schemes using linear block codes. In *International Workshop on the Theory and Application of Cryptographic Techniques*. Springer.

[Bicakci et al., 2009a] Bicakci, K., Atalay, N. B., Yuceel, M., Gurbaslar, H., and Erdeniz, B. (2009a). Towards usable solutions to graphical password hotspot problem. In *IEEE COMPSAC*.

[Bicakci et al., 2011a] Bicakci, K., Atalay, N. B., Yuceel, M., and van Oorschot, P. C. (2011a). Exploration and field study of a browser-based password manager using icon-based passwords. In *Workshop on Real-Life Cryptographic Protocols and Standardization*.

[Bicakci et al., 2011b] Bicakci, K., Atalay, N. B., Yuceel, M., and van Oorschot, P. C. (2011b). Exploration and field study of a browser-based password manager using icon-based passwords. In *RLCPS*.

[Bicakci et al., 2009b] Bicakci, K., Yuceel, M., Erdeniz, B., Gurbaslar, H., and Atalay, N. (2009b). Graphical passwords as browser extension: Implementation and usability study. *Trust Management III.*

[Blakley et al., 1979] Blakley, G. R. et al. (1979). Safeguarding cryptographic keys. In *Proceedings of the national computer conference.*

[Boldyreva, 2003] Boldyreva, A. (2003). Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *International Workshop on Public Key Cryptography.* Springer.

[Boyen, 2009a] Boyen, X. (2009a). Hidden credential retrieval from a reusable password. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security.* ACM.

[Boyen, 2009b] Boyen, X. (2009b). HPAKE: Password authentication secure against cross-site user impersonation. In *Cryptology and Network Security.* Springer.

[Boyko et al., 2000] Boyko, V., MacKenzie, P., and Patel, S. (2000). Provably secure password-authenticated key exchange using diffie-hellman. In *EUROCRYPT 2000.* Springer.

[Brickell, 1989] Brickell, E. F. (1989). Some ideal secret sharing schemes. In *Workshop on the Theory and Application of of Cryptographic Techniques.* Springer.

[Camenisch et al., 2015a] Camenisch, J., Enderlein, R. R., and Neven, G. (2015a). Two-server password-authenticated secret sharing uc-secure against transient corruptions. PKC 2015.

[Camenisch et al., 2014] Camenisch, J., Lehmann, A., Lysyanskaya, A., and Neven, G. (2014). Memento: How to reconstruct your secrets from a single password in a hostile environment. In *CRYPTO 2014.* Springer.

[Camenisch et al., 2015b] Camenisch, J., Lehmann, A., and Neven, G. (2015b). Optimal distributed password verification. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[Camenisch et al., 2015c] Camenisch, J., Lehmann, A., Neven, G., and Samelin, K. (2015c). Virtual smart cards: How to sign with a password and a server. Cryptology ePrint Archive, Report 2015/1101. `http://eprint.iacr.org/`.

[Canetti, 2000] Canetti, R. (2000). Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY*.

[Chatterjee et al., 2015] Chatterjee, R., Bonneau, J., Juels, A., and Ristenpart, T. (2015). Cracking-resistant password vaults using natural language encoders. In *IEEE SP*.

[Chiasson et al., 2006] Chiasson, S., van Oorschot, P. C., and Biddle, R. (2006). A usability study and critique of two password managers. In *USENIX Security Symposium*.

[Corporate, 2016] Corporate, L. (2016). Lastpass-the last password you have to remember.

[Daemen and Rijmen, 2013] Daemen, J. and Rijmen, V. (2013). *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media.

[Das et al., 2014] Das, A., Bonneau, J., Caesar, M., Borisov, N., and Wang, X. (2014). The tangled web of password reuse. In *NDSS*.

[De Cristofaro et al., 2014] De Cristofaro, E., Du, H., Freudiger, J., and Norcie, G. (2014). A comparative usability study of two-factor authentication. In *NDSS USEC*.

[Diffie and Hellman, 1976] Diffie, W. and Hellman, M. E. (1976). New directions in cryptography. *Information Theory, IEEE Transactions on.*

[Farràs et al., 2012] Farràs, O., Martí-Farré, J., and Padró, C. (2012). Ideal multipartite secret sharing schemes. *Journal of cryptology.*

[Faulkner, 2003] Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers.*

[Feige et al., 1988] Feige, U., Fiat, A., and Shamir, A. (1988). Zero-knowledge proofs of identity. *Journal of cryptology.*

[Fiat and Shamir, 1986] Fiat, A. and Shamir, A. (1986). How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86.*

[Florencio and Herley, 2007a] Florencio, D. and Herley, C. (2007a). A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web.* ACM.

[Florencio and Herley, 2007b] Florencio, D. and Herley, C. (2007b). A large-scale study of web password habits. In *ACM WWW.*

[Florêncio et al., 2007] Florêncio, D., Herley, C., and Coskun, B. (2007). Do strong web passwords accomplish anything? *HotSec.*

[Ford and Kaliski, 2000] Ford, W. and Kaliski, B. S. (2000). Server-assisted generation of a strong secret from a password. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000.(WET ICE 2000).* IEEE.

[Forget et al., 2008] Forget, A., Chiasson, S., van Oorschot, P. C., and Biddle, R. (2008). Improving text passwords through persuasion. In *ACM SOUPS.*

[Freedman et al., 2005] Freedman, M. J., Ishai, Y., Pinkas, B., and Reingold, O. (2005). Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography Conference*. Springer.

[Goldwasser et al., 1989] Goldwasser, S., Micali, S., and Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on computing*.

[Goldwasser et al., 1988] Goldwasser, S., Micali, S., and Rivest, R. L. (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*.

[Golla et al., 2016] Golla, M., Beuscher, B., and Dürmuth, M. (2016). On the security of cracking-resistant password vaults. ACM SIGSAC.

[Gunson et al., 2011] Gunson, N., Marshall, D., Morton, H., and Jack, M. (2011). User perceptions of security and usability of single-factor and two-factor authentication in automated telephone banking. *Computers & Security*.

[Halderman et al., 2005] Halderman, J. A., Waters, B., and Felten, E. W. (2005). A convenient method for securely managing passwords. In *ACM WWW*.

[Halevi and Krawczyk, 1999] Halevi, S. and Krawczyk, H. (1999). Public-key cryptography and password protocols. *ACM Transactions on Information and System Security (TISSEC)*.

[İşler and Küpçü, 2017] İşler, D. and Küpçü, A. (2017). Threshold single password authentication. In *ESORICS DPM'17*. Springer.

[İşler and Coskun, 2018] İşler, Devriş, K. A. and Coskun, A. (2018). Usability study on single password authentication. In *arXiv.org.XXXX*.

[Ito et al., 1989] Ito, M., Saito, A., and Nishizeki, T. (1989). Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science).*

[Jarecki et al., 2014] Jarecki, S., Kiayias, A., and Krawczyk, H. (2014). Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model. ASIACRYPT'14.

[Jarecki et al.,] Jarecki, S., Kiayias, A., Krawczyk, H., and Xu, J. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online).

[Jarecki et al., 2017] Jarecki, S., Kiayias, A., Krawczyk, H., and Xu, J. (2017). Toppss: Cost-minimal password-protected secret sharing based on threshold oprf. In *International Conference on Applied Cryptography and Network Security*. Springer.

[Jarecki et al., 2016a] Jarecki, S., Krawczyk, H., Shirvanian, M., and Saxena, N. (2016a). Device-enhanced password protocols with optimal online-offline protection. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM.

[Jarecki et al., 2016b] Jarecki, S., Krawczyk, H., Shirvanian, M., and Saxena, N. (2016b). Device-enhanced password protocols with optimal online-offline protection. In *ACM ASIACCS*.

[Jarecki et al., 2018] Jarecki, S., Krawczyk, H., and Xu, J. (2018). Opaque: An asymmetric pake protocol secure against pre-computation attacks. In *EUROCRYPT 2018*.

[Juels and Rivest, 2013] Juels, A. and Rivest, R. L. (2013). Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM.

[Karole et al., 2010] Karole, A., Saxena, N., and Christin, N. (2010). A comparative usability evaluation of traditional password managers. In *ICISC*.

[Katz and Lindell, 2014] Katz, J. and Lindell, Y. (2014). *Introduction to modern cryptography*. CRC press.

[Katz et al., 2005] Katz, J., MacKenzie, P., Taban, G., and Gligor, V. (2005). Two-server password-only authenticated key exchange. In *International Conference on Applied Cryptography and Network Security*.

[Katz et al., 2001] Katz, J., Ostrovsky, R., and Yung, M. (2001). Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT 2001*. Springer.

[Kumaraguru et al., 2010] Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L. F., and Hong, J. (2010). Teaching johnny not to fall for phish. *ACM Transactions on Internet Technology (TOIT)*.

[Li et al., 2014] Li, Z., He, W., Akhawe, D., and Song, D. (2014). The emperor's new password manager: Security analysis of web-based password managers. In *USENIX Security Symposium*.

[MacKenzie et al., 2002] MacKenzie, P., Shrimpton, T., and Jakobsson, M. (2002). Threshold password-authenticated key exchange. In *CRYPTO 2002*. Springer.

[Mannan and van Oorschot, 2007] Mannan, M. and van Oorschot, P. C. (2007). Using a personal device to strengthen password authentication from an untrusted computer. In *Financial Cryptography and Data Security*. Springer.

[Mannan and van Oorschot, 2008] Mannan, M. and van Oorschot, P. C. (2008). Digital objects as passwords. In *HotSec*.

[McCarney et al., 2012] McCarney, D., Barrera, D., Clark, J., Chiasson, S., and van Oorschot, P. C. (2012). Tapas: design, implementation, and usability evaluation of a password manager. In *ACSAC*. ACM.

[Melicher et al., 2016] Melicher, W., Kurilova, D., Segreti, S. M., Kalvani, P., Shay, R., Ur, B., Bauer, L., Christin, N., Cranor, L. F., and Mazurek, M. L. (2016). Usability and security of text passwords on mobile devices. In *CHI Conference on Human Factors in Computing Systems*. ACM.

[M'raihi et al., 2005] M'raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and Ranen, O. (2005). Hotp: An hmac-based one-time password algorithm. Technical report.

[M'Raihi et al., 2011] M'Raihi, D., Machani, S., Pei, M., and Rydell, J. (2011). Totp: Time-based one-time password algorithm. Technical report.

[Pedersen, 1991] Pedersen, T. P. (1991). Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*. Springer.

[Ross et al., 2005] Ross, B., Jackson, C., Miyake, N., Boneh, D., and Mitchell, J. C. (2005). Stronger password authentication using browser extensions. In *Usenix security*. Baltimore, MD, USA.

[Sasse and Flechais, 2005] Sasse, M. A. and Flechais, I. (2005). Usable security: Why do we need it? how do we get it? O'Reilly.

[Schnorr, 1991] Schnorr, C.-P. (1991). Efficient signature generation by smart cards. *Journal of cryptology*.

[Shamir, 1979] Shamir, A. (1979). How to share a secret. *Communications of the ACM*.

[Shirvanian et al., 2014] Shirvanian, M., Jarecki, S., Saxena, N., and Nathan, N. (2014). Two-factor authentication resilient to server compromise using mix-bandwidth devices. In *NDSS*.

[Shirvanian et al., 2017] Shirvanian, M., Jareckiy, S., Krawczykz, H., and Saxena, N. (2017). Sphinx: A password store that perfectly hides passwords from itself. In *IEEE ICDCS*.

[Smith, 2002] Smith, R. E. (2002). The strong password dilemma. *Computer Security Journal*.

[Srinivas et al., 2015] Srinivas, S., Balfanz, D., Tiffany, E., and Czeskis, A. (2015). Universal 2nd factor (u2f) overview. *FIDO Alliance Proposed Standard*.

[Stobert and Biddle, 2014] Stobert, E. and Biddle, R. (2014). The password life cycle: user behaviour in managing passwords. In *ACM SOUPS*.

[Stobert and Biddle, 2018] Stobert, E. and Biddle, R. (2018). The password life cycle. *ACM TOPS*.

[Sun et al., 2012] Sun, H.-M., Chen, Y.-H., and Lin, Y.-H. (2012). opass: A user authentication protocol resistant to password stealing and password reuse attacks. *IEEE TIFS*.

[Tatli, 2015] Tatli, E. I. (2015). Cracking more password hashes with patterns. *Information Forensics and Security, IEEE Transactions on*.

[Turner, 2008] Turner, J. M. (2008). The keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication*.

[Venkatesh et al., 2003] Venkatesh, V., Morris, M. G., Davis, G. B., and Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly*.

[Yang and Shieh, 1999] Yang, W.-H. and Shieh, S.-P. (1999). Password authentication schemes with smart cards. *Computers & Security*.

[Zhao and Mannan, 2013] Zhao, L. and Mannan, M. (2013). Explicit authentication response considered harmful. In *Proceedings of the 2013 workshop on New security paradigms workshop*. ACM.

[Zviran and Haga, 1993] Zviran, M. and Haga, W. J. (1993). A comparison of password techniques for multilevel authentication mechanisms. *The Computer Journal*.

Appendix A

MOBILE-BASED DISTRIBUTED SINGLE PASSWORD PROTOCOL

Figure A.1: Mobile-based DiSPP registration phase ([Acar et al., 2013]).

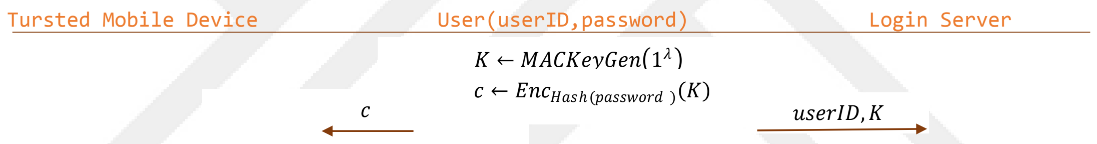
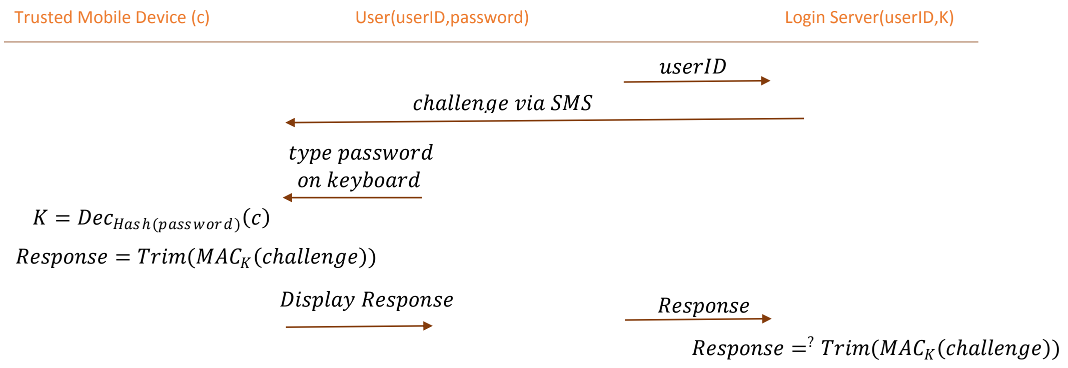

Tursted Mobile Device
User(userID,password)
Login Server
$K \leftarrow MACKeyGen(1^{\lambda})$
$c \leftarrow Enc_{Hash(password)}(K)$
c
$userID, K$

Figure A.2: Mobile-based DiSPP authentication phase ([Acar et al., 2013]).


Trusted Mobile Device (c)
User(userID,password)
Login Server(userID,K)
$userID$
$challenge\ via\ SMS$
$type\ password$
$on\ keyboard$
$K = Dec_{Hash(password)}(c)$
$Response = Trim(MAC_K(challenge))$
$Display\ Response$
$Response$
$Response \overset{?}{=} Trim(MAC_K(challenge))$

Appendix B

PARTICIPANTS RESPONSES PER QUESTION

Table B.1: Participants scores per question for two-factor authentication. Each row represents responses of a participant

EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
3	4	3	4	2	2	2	2	3	2	2	3	3	3	3	3	3	3	2	1	1	2	1
3	3	2	3	2	2	2	2	2	2	2	2	3	3	3	2	2	2	2	2	3	3	2
3	3	3	3	3	2	2	2	2	3	2	2	1	1	1	1	1	1	1	2	2	2	1
3	3	2	4	2	4	2	2	3	3	3	2	1	1	1	2	1	1	1	1	2	2	1
3	3	3	4	2	3	3	2	2	2	2	3	3	3	3	3	3	3	4	2	2	2	3
3	3	3	3	2	2	2	2	3	3	3	3	2	3	3	3	2	2	2	2	3	2	2
3	3	3	3	3	3	2	3	3	3	3	3	2	2	2	3	1	2	2	3	3	3	1
4	4	4	4	2	2	2	3	4	4	4	3	3	3	3	3	2	3	2	3	3	3	3
3	3	3	3	4	4	2	3	2	2	2	3	3	3	3	2	2	2	2	2	2	1	1
4	2	4	4	2	2	2	2	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3
3	3	3	3	3	2	2	3	4	3	4	2	1	3	4	2	1	3	2	3	3	3	2
4	4	4	4	2	1	2	2	2	2	2	2	3	3	2	3	3	2	1	1	1	1	1
3	4	4	4	2	3	3	2	4	4	4	4	4	4	4	4	4	4	3	4	4	4	4
4	4	3	4	2	1	1	3	3	4	3	4	4	4	4	4	1	4	2	3	3	4	3
3	3	3	3	2	2	2	2	3	3	3	3	3	3	3	4	4	4	3	4	4	4	4
4	3	4	4	2	4	4	4	3	3	2	2	3	3	4	4	3	3	2	4	4	4	4
4	3	3	3	2	2	2	2	3	3	3	3	2	2	2	3	2	2	2	3	3	3	3
3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	2	3	2	2	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	2	2	2	2	3	2	2	3	3	3	3	3	3	3	2	3	3	2	3
4	2	2	3	3	3	3	3	2	3	2	2	1	1	2	1	1	3	2	2	3	3	3
4	3	3	4	2	2	2	2	3	3	2	3	3	3	3	2	2	2	2	2	2	2	2
4	4	4	4	2	2	2	2	3	2	2	3	3	3	3	3	3	3	2	1	2	2	3
3	3	3	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	2
2	3	3	4	3	1	1	1	3	3	3	3	1	1	2	1	1	1	1	2	3	3	1

Table B.2: Participants scores per question for mobile-based DiSPP. Each row represents responses of a participant.

EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
4	3	3	4	1	1	2	2	3	2	2	3	3	3	3	3	3	3	2	4	4	4	4
3	4	3	3	2	2	2	2	3	2	2	3	3	3	3	3	3	3	3	4	4	4	4
4	4	4	4	2	2	2	2	2	3	2	3	4	4	4	4	3	3	2	4	4	4	4
3	3	3	3	2	2	3	2	2	2	2	3	3	3	3	2	2	2	2	3	3	3	3
4	4	3	3	2	1	1	2	2	2	2	3	1	2	2	2	2	1	1	3	3	3	3
3	3	3	3	1	2	2	1	3	3	3	4	3	2	2	1	1	1	1	3	3	2	3
3	3	3	3	2	2	2	2	2	2	2	4	4	4	4	4	4	3	2	3	3	3	3
4	3	4	4	2	2	2	2	3	3	4	4	3	3	3	4	4	3	3	3	3	3	3
3	3	2	2	3	3	3	3	2	2	1	2	3	3	3	2	2	2	2	3	3	3	2
2	4	4	4	1	1	1	4	3	3	2	3	4	4	4	2	2	2	2	3	3	3	3
3	3	3	2	2	2	2	3	3	3	2	2	3	2	2	3	2	2	2	2	2	2	2
4	1	4	4	3	1	1	1	4	4	4	4	4	4	4	4	4	2	2	3	3	3	3
2	3	2	3	1	1	1	1	2	2	2	4	4	4	4	4	2	1	2	4	4	4	4
4	4	4	4	3	2	2	2	3	3	3	3	3	3	3	3	3	3	2	3	3	2	3
4	3	3	4	2	1	1	3	3	4	3	4	4	4	4	4	1	4	2	3	3	4	3
3	4	4	3	2	2	2	2	4	4	4	4	4	4	4	4	4	4	3	4	4	4	4
3	3	3	3	2	2	2	2	3	3	3	3	3	3	3	4	4	4	3	4	4	4	4
3	3	2	2	2	2	2	3	2	2	2	2	3	2	2	2	1	2	2	3	3	3	4
1	2	3	1	2	2	2	2	3	3	3	3	2	2	2	3	3	2	2	3	3	3	3
2	3	3	3	2	2	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	2	2	2	2	3	2	2	3	3	3	3	3	3	3	2	3	3	2	3
3	4	4	4	1	1	1	1	2	4	3	4	4	3	3	2	2	3	2	3	3	4	4
3	3	3	3	2	1	2	1	4	3	3	3	2	2	3	1	1	2	1	3	3	4	3
3	4	4	4	4	1	2	1	2	2	2	3	3	3	4	4	3	3	2	3	3	3	3
2	2	3	3	2	2	2	2	3	2	3	2	3	3	3	3	3	2	2	2	2	2	1

Table B.3: Participants scores per question for traditional password authentication. Each row represents responses of a participant.

EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
4	4	4	4	2	4	2	2	3	3	2	4	4	4	4	4	4	4	2	4	4	4	3
2	3	1	1	2	2	3	2	1	1	1	1	1	1	1	1	1	1	1	3	2	2	1
3	3	2	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1
3	2	2	2	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	2	1	2	2	4	4	4	2	1	1	2	1	1	2	2	1	1	1	1	1	1	2
3	3	3	2	2	2	2	2	3	3	3	3	2	2	3	3	3	3	2	3	3	3	3
4	4	4	4	1	1	1	1	4	4	4	4	1	1	1	3	1	1	1	3	3	3	2
2	3	3	3	1	1	1	1	3	3	3	3	3	4	4	4	4	4	4	4	3	4	2
4	4	4	4	2	2	2	2	3	3	3	3	2	2	3	2	2	2	2	3	3	3	2
3	2	1	2	1	1	1	4	3	3	3	2	1	1	1	1	1	1	1	4	4	3	3
4	4	4	4	2	2	2	3	3	3	3	3	2	3	3	3	2	1	1	4	3	3	3
3	3	2	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	2	2	2	2
3	3	3	3	1	2	1	2	3	3	3	3	2	2	3	3	2	2	1	1	1	1	1
3	3	4	4	3	4	3	2	3	2	1	3	3	2	3	3	3	2	2	3	3	3	3
3	3	3	2	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2
4	3	2	3	3	4	3	3	2	2	3	2	2	2	2	2	1	1	2	3	4	4	3
4	3	2	4	3	2	3	2	3	3	3	3	2	2	2	2	2	2	2	3	3	3	3
4	4	4	4	1	2	2	1	4	4	4	3	1	1	1	2	1	1	1	2	2	1	2
3	3	2	2	2	2	2	3	2	2	2	2	3	2	2	2	1	2	2	3	3	3	4
2	3	2	3	2	2	2	3	2	2	1	2	2	2	2	2	2	2	2	3	3	1	1
4	3	4	4	2	2	3	2	3	1	4	2	4	1	3	4	3	2	1	4	3	4	4
2	2	2	2	1	1	2	2	1	1	2	1	1	1	2	2	2	1	2	1	1	1	2
3	3	3	3	2	2	2	2	3	3	3	3	2	3	3	3	2	2	2	3	3	3	2
3	3	2	2	3	3	3	2	2	2	2	2	1	1	1	1	1	1	3	3	3	3	3
3	4	2	2	2	3	3	2	2	2	2	1	1	1	1	2	2	1	3	3	3	1	4

Table B.4: Participants scores per question for cloud-based DiSPP. Each row represents responses of a participant.

EE1	EE2	EE3	EE4	A1	A2	A3	A4	BIU1	BIU2	BIU3	ATUT1	ATUT2	ATUT3	ATUT4	PE1	PE2	PE3	PE4	PS1	PS2	PS3	PS4
4	4	4	4	1	1	1	1	2	2	2	3	2	2	2	2	1	1	1	2	2	2	2
4	4	4	4	1	2	2	1	4	4	4	4	2	2	2	4	4	4	1	4	4	4	4
3	3	3	3	2	2	2	2	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3
3	4	4	3	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	4	4	3	2	4	3	2	3	2	2	4	3	3	3	4	4	3	2	3	3	1	2
2	3	2	3	2	2	2	2	3	3	3	3	2	3	2	3	2	2	2	3	3	3	3
4	3	3	3	2	2	2	2	3	3	3	3	3	3	3	4	3	3	3	3	3	3	3
3	4	3	3	3	4	2	2	2	2	2	2	3	3	3	2	2	2	2	3	3	3	3
1	1	1	1	4	2	3	2	2	2	2	2	3	3	3	2	2	2	3	2	1	1	2
4	4	4	4	2	2	1	3	3	3	3	3	2	2	3	4	4	4	1	2	2	1	1
3	4	4	4	1	3	3	1	3	3	2	4	2	2	4	4	4	4	2	3	2	2	3
4	4	4	4	2	2	1	1	3	3	3	4	2	3	3	4	3	2	1	4	4	4	4
3	3	2	3	2	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	2	3	3	1	2	1	2	2	2	2	3	3	3	3	3	4	4	1	2	2	2	2
3	4	4	4	2	1	3	2	3	3	2	4	2	3	3	4	4	4	2	4	4	4	3
4	4	4	4	1	2	2	1	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3
4	4	4	4	1	1	1	1	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4
4	4	4	4	1	2	2	1	3	3	3	4	3	3	3	4	3	3	2	3	3	2	4
4	4	4	4	2	2	2	2	3	3	3	3	2	2	3	3	3	3	2	2	3	2	2
4	3	3	3	2	2	2	2	3	3	3	3	2	2	2	3	2	2	2	3	3	3	3
4	4	4	4	3	3	2	2	3	3	3	3	3	3	3	3	3	3	2	3	3	2	3
4	3	4	4	2	2	3	2	4	1	1	4	3	4	4	3	3	4	2	3	1	4	4
2	3	3	3	2	2	1	2	1	1	1	2	1	3	3	3	2	1	2	1	2	2	3
4	1	4	4	1	1	1	3	3	3	3	3	3	3	3	3	3	1	3	3	3	3	1
4	2	3	3	1	2	1	3	3	2	3	2	3	3	4	3	3	1	3	3	2	3	1