CYQUR: A SECURE SECRET DATA MANAGEMENT SYSTEM

The Binarii Labs Team
Dublin, Ireland info@binariilabs.com

ABSTRACT

Secret management refers to the tools and methods for storing and managing digital authentication credentials, including passwords, keys, API tokens or more generally any data blurb that can be considered a secret.

At its core, a secret data management system exists as a simple credential database that stores a user's secrets across different applications, products and services. In its most basic form, it serves the purpose of a password manager to store login and password credentials across different websites. As it handles sensitive user data, a secret data manager needs to ensure: a) confidentiality of the data being managed, b) its integrity, and c) availability of the credential database.

Most modern secret data managers however require users to trust third-party vendors to guarantee all the three security desiderata. However, several hacks in the past have shown that this trust on third-party vendors could well be misplaced as private data of millions of end users have been repeatedly exposed by hackers.

In this work, we introduce Cyqur—an information-theoretically secure secret data management system that relinquishes control over the credential database to the end user thereby giving them full control over managing their data and removing reliance on third-party vendors.

We show that this can be achieved without compromising confidentiality, integrity and availability of the data while improving the overall usability. To provide confidentiality, Cyqur leverages Shamir's secret sharing cryptographic primitive to split an encrypted credential database into several shares that can be stored across different independent entities. A minimum number of such shares is required to reconstruct the password. This also presents a way to socially recover the credential database via a distributed trust system. Furthermore, Cyqur leverages blockchain's P2P network to ensure data availability and integrity of the shares. A hash digest of the shares is committed to a blockchain's network that provides a trust-minimised data storage layer.

*K*eywords Security · Passwords · Privacy · Trust · Data Availability · Social Recovery · Secret Sharing · Secret data.

Contents

1. Introduction

Password-based authentication is still the prevalent method of authentication on the Internet. Given the importance of passwords as the first line of defence, users are encouraged to use unique passwords for each application they sign up for. Unfortunately, it is extremely hard if not outright impossible to remember more than a few unique and unrelated passwords and therefore users often end up reusing the same password or using related passwords across multiple applications. This inherent behaviour of password abuse can put users at severe risk of being compromised [26] [15] [5] [3]. Indeed, several password leaks in the past ([16] [8] [14]) show that users who use a single password across multiple applications are the most at risk as recovering a user's password for one application allows the adversary to compromise all the other applications where the same or related passwords were used.

Password managers (such as LastPass [11], 1Password [2], amongst others) aim to solve the aforementioned issues by assisting users in generating strong passwords, and securely storing and managing them for end users. At its core, a password manager is essentially a database to store a user's passwords and usernames on different applications. The password manager controls access to this database via a master username/password. A secure password manager, with a strong master password, ensures that a user can rely on distinct, high-entropy passwords for each application without the associated cognitive burden of memorising all them. Instead, the user only has to remember one strong master password. While password managers can be extremely helpful in addressing the password conundrum, their adoption has been paltry at best [17] [19] [4]. One of

the main reasons is that users have to trust a third-party vendor such as LastPass. Popular vendors of password managers typically store a salted hash of the master password for authentication purposes and a copy of the encrypted credential database (using the master username/password) on the cloud or on their servers. As sensitive information is stored on the vendor side, password managers themselves become a honeypot for adversaries. In fact, several password managers have been targets of hacks in the past [25] [12]. LastPass, one of the largest vendors in the market with over 25 million global users as reported in 2020 became the victim of a major hack in 2015 [23]. Hackers were able to intrude into the company's servers and compromised LastPass account email addresses, password reminders, server per user salts, and authentication hashes. OneLogin, another vendor, suffered a major security breach in 2017 [10] in which a large amount of customer data was compromised, including the ability to entirely decrypt the encrypted credential database. The attacker had obtained access to a set of AWS keys that made it possible to decrypt the credential database of customers.

Although in encrypted form, as password managers store all the private data in one place, the risk of losing the credentials and other data across all applications increases in scenarios where the attacker gets hold of the credential database. Moreover, if the password manager has the only copy of the credential database, a user can lose access to all of her accounts if she forgets the password or the server goes down. In essence, the availability of the data itself is also entrusted with a third-party vendor. In face of the issues with popular password managers, alternative designs have been proposed in the literature. SPHINX [22] for instance leverages oblivious pseudo random function that transforms a human-memorable password into a random password with the aid of a device without the need to store the passwords on the device and without the device learning anything about the password even when computing on it. Specifically, when using SPHINX, for each application with which the user has an account, the device stores a unique key k . This key is used to map the user-memorised password pwd (input by the user into the client) into a randomised password rwd = Fk(pwd|domain) using the (oblivious) PRF Fk based on a protocol between the device and the client. Other advanced password management solutions have been proposed that do not require storage of rwd [9] [20] [27]. For example, PwdHash [20] maps pwd to a rwd by hashing (pwd, domain) pair and registering it as a strong password with the server. PwdHash deterministically transforms a user's password into a more complex password but this transformation does not protect against dictionary attacks at a compromised server. Moreover, if a user uses the same memorable password pwd with PwdHash for different services, the compromise of a single server leads to the discovery of pwd via an offline dictionary attack and then to the (deterministic) calculation of all the user's passwords derived from pwd . Tapas proposed in [13] eliminates the need of a master password by introducing the idea of dual-possession authentication,
whereby the credential database of passwords is encrypted and stored on a mobile device, while the decryption key stays inside the browser on the paired computer. Keeping the decryption key in the browser can however be risky. Moreover if the paired computer is compromised, it is often trivial for the attacker to also get hold of the user's mobile device.

Our work is inspired by [7] [1] [6], which leverages Shamir's secret sharing technique [21] to cryptographically split an encrypted password, into parts, called shares, that are distributed among a group of entities. The complete password can be recovered even when only a sufficiently sized subset of shares can be supplied, while any smaller subset of shares does not leak any information about the original data (perfect secrecy). While most of the previous works address the security of the credential database, some recent work [18] [24] cater to the availability issues, i.e., what if the vendor storing the credential database becomes unavailable or how to ensure that the integrity of the data is maintained throughout its lifespan. In order to guarantee availability of data, solutions like b.lock [24] have been proposed where the encrypted credential database is stored on a

blockchain. As blockchains are P2P and decentralised networks, the data is perennially available for the end user and thereby removes the central point of failure typical of password managers that store the credential database on the vendor-side.

Contributions: In this work, we identify three key security desiderata for password managers, namely: a) confidentiality of the data being managed, b) its integrity, and c) availability of the credential database. Based on these desiderata, we build CYQUR — a simple yet secure secret data management system. For confidentiality, CYQUR relies on a password-based public symmetric key encryption scheme. Once the data is encrypted, the system splits the encrypted credential database (encrypted under the master password) into n shares (decided by the end user) out of which $t < n$ shares are required to reconstruct the encrypted password. These shares can then be stored with independent parties The system guarantees security as long as no more than $t-1$ entities storing the shares are compromised. The fragmentation idea is inspired by Shamir's secret sharing scheme [7] [1] [6]. In order to ensure the integrity and availability of the shares, CYQUR leverages blockchain's P2P network. A hash digest of the shares is committed to a blockchain's network that provides a trust-minimised data storage and indexing layer as proposed in [18] [24]). We also present a full end-to-end implementation of CYQUR and discuss its UX and future extensions.

2. Background

In this section, we present the essential background needed for the rest of the paper. In particular, we present the mechanics behind a secret data manager, Shamir's secret sharing cryptographic primitive and a primer on blockchains.

2.1 Secret Data Managers

Secret data managers (such as LastPass [11], 1Password [2], amongst others) are tools that allow users to create strong secret credentials, one for each application and also help to store and manage them for end users. Modern vendors even integrate second-factor authentication services such as Google Authentication and also allow sharing of the data with other collaborators, for example, sharing of a company-wide credential with different stakeholders.

2.2 Shamir's Secret Sharing

Secret Sharing [21] is a protocol to divide a secret data into parts called shares. These shares can be distributed among a group of independent entities such that the secret can only be reconstructed when a subset of participants combine their shares. Shamir's secret sharing protocol is a threshold secret sharing scheme, which means that the secret can be reconstructed from any set of at least k shares, where k is a given threshold. This implies redundancy directly: if at least k shares remain accessible, the secret can be reconstructed even if the other shares are lost or destroyed. Moreover, Shamir's scheme is perfectly secure in the sense that any subset of less than k shares reveal absolutely no information about the secret. The scheme is based on the fact that a $(k-1)$ degree polynomial $f(x)$ is uniquely defined by (at least) k distinct pairs $(x,f(x))$. The scheme consists of two sub-protocols, one to generate the shares from a given secret aka the sharing phase, and the second to reconstruct the secret from a subset of those shares. In the sharing phase, the person owning the secret chooses a polynomial $f(x) = a_0+a_1x+a_2x^2+...+a_{k-1}x^{k-1}$, where, $a_i \in Z_q = \{0,...,(q-1)\}$ and k is the minimum number of shares required to reconstruct the secret. The secret is

assumed to be Z_q and a_0 is set to take its value. The i−th share is simply set to $f(i)$, the value of the polynomial in i. In the reconstruction phase, any k (or more) shares can be used to reconstruct the polynomial via a simple polynomial interpolation.

2.3 Blockchains

Blockchains such as Ethereum and Bitcoin are a distributed ledger maintained by a permissionless network of participants. The ledger is designed to be append-only and represents a state machine and the nodes in the network decide on the state transitions. For example, in case of the Bitcoin network, participants of the network keep track of how many units of the Bitcoin currency (aka BTC) is held by a user and when a user sends a transaction to the network to transfer a certain number of BTC to a recipient, the network checks the validity of the transaction and whether the user has enough funds. Once the network agrees on the validity of the transfer, the state of the system gets updated accordingly. All this without relying on any centralised third-party. Ethereum builds an extension of the Bitcoin network by allowing users to run arbitrary programs (aka smart contracts) on these nodes and thereby agree on the final state of these programs. The decentralised nature of the blockchain networks is ideal to build applications that can benefit from a trustless, transparent and open development environment that allows applications to compose in a natural way.

Another usage of the blockchain technology has been around building solutions for decentralised storage of data. Distributed data storage platforms are better alternatives to centralised vendors as the data is not at the mercy of a single vendor but instead is hosted and maintained by a network of incentivized participants. Moreover, many of these data storage platforms also offer a way for users to efficiently verify whether a certain piece of data exists on the network.

3. Threat Model

We assume that the user's device is not compromised as no secret data management system will be able to operate securely on a device compromised by malware. We also assume that if a user distributes a secret among n parties, such that any k out of n secret shares can be used to reconstruct the secret, then no more than $k - 1$ parties can be byzantine and collude to recover the secret. As the system utilises blockchain for integrity and availability, we assume that the blockchain network is secure under the assumption it operates in, for example, a network like Bitcoin is secure as long as there is no single entity that controls more than 50protocols that use byzantine fault tolerant protocols as the underlying consensus protocol, we assume that no more than 1 3 of the nodes participating in the consensus protocol are byzantine at any given point in time. As an extension of this, we assume that all cryptographic primitives such as hash function, encryption scheme, etc., are secure under the respective security assumptions. The basic system design attempts to avoid any centralised server and so we assume that any software service running on a third-party vendor can be compromised either internally or by an external adversary. Even in the scenario where the service provided by a third party is assumed to be secure, we assume that there is no guarantee on the liveness of the service. In other words, it is possible that the service may go off-line or its access be restricted by the service provider.

4. System Design of Cyqur

Cyqur is a simple yet powerful encryption and decryption application that facilitates the storage and transmission of plain text secret data. It aims to avoid the storage of the complete secret on a third party server (with the potential risk of the server not being available or the server being compromised) through encrypting and then fragmenting the encrypted secret across a number of storage sources and therefore providing the owner of the data or files the ability to store fragments and packets of groups of fragments on devices or cloud services they control.

Cyqur uses a main password to encrypt the secret data and this is the password that also decrypts the secret. It is possible, as a further layer of security, to use Cyqur to process this password as a secret in its own right. This password, like a private key, is extremely important and needs to be retained by the owner of the secret or transmitted securely to a trusted party for the purposes of succession management. Cyqur uses browser cryptography and aims to minimize exposure to third party code or services such that the key features work offline for added security.

The fragments of the encrypted secret can be of any determinable size or amount. Controllers can download partial or full fragments (contained within “packets”) along with where the fragments are located and in the order the fragments should be concatenated. The metadata may also contain information of the encryption algorithms used. This metadata and or fragments can then be forwarded to trusted parties such as lawyers, custodians and trustees.

Communication can be through existing digital mechanisms and protocols or written down on paper, imprinted in ceramics or recorded in other analogue innovative ways. In an event where users are no longer able to access their secrets, these trusted third parties (we call them Custodians of last resort or “COLR”) are able to collectively recompile the encrypted secrets. Unlike password managers that store the data on the device and often on a server too, Cyqur stores all the data in the browser for further use. Since data is secured in the browser, users can delete part or everything at the click of a button. This leaves no audit trail but there is also no way to recover. However, if recovery is important then users can download the fragmented encrypted data and store them in a password manager, email, cloud or elsewhere. As the data is encrypted, even if all the fragments are compromised, the attacker would still need the password to access the data. If a user had stored the partial fragments then a bad actor would need to have access to the location of all the fragments, along with the order of concatenation and the main password to reveal the secret.

Browser encryption protocols can also be chosen by the data controller and applied to each fragment. The use of
“spoof” fragments can also improve the security of the secret and minimize the breach by bad actors. However, the more complex methods require a full record and accessible process and procedures to enable recovery of the secret.

If the controller wishes to prove the secrets provenance, Cyqur will also write an index to a blockchain of choice. To do this, Cyqur will require credits to use this functionality. If Cyqur is written to a blockchain, then the application will need to be online. However, if the packets are created offline then the risk of re-concatenation is minimized.

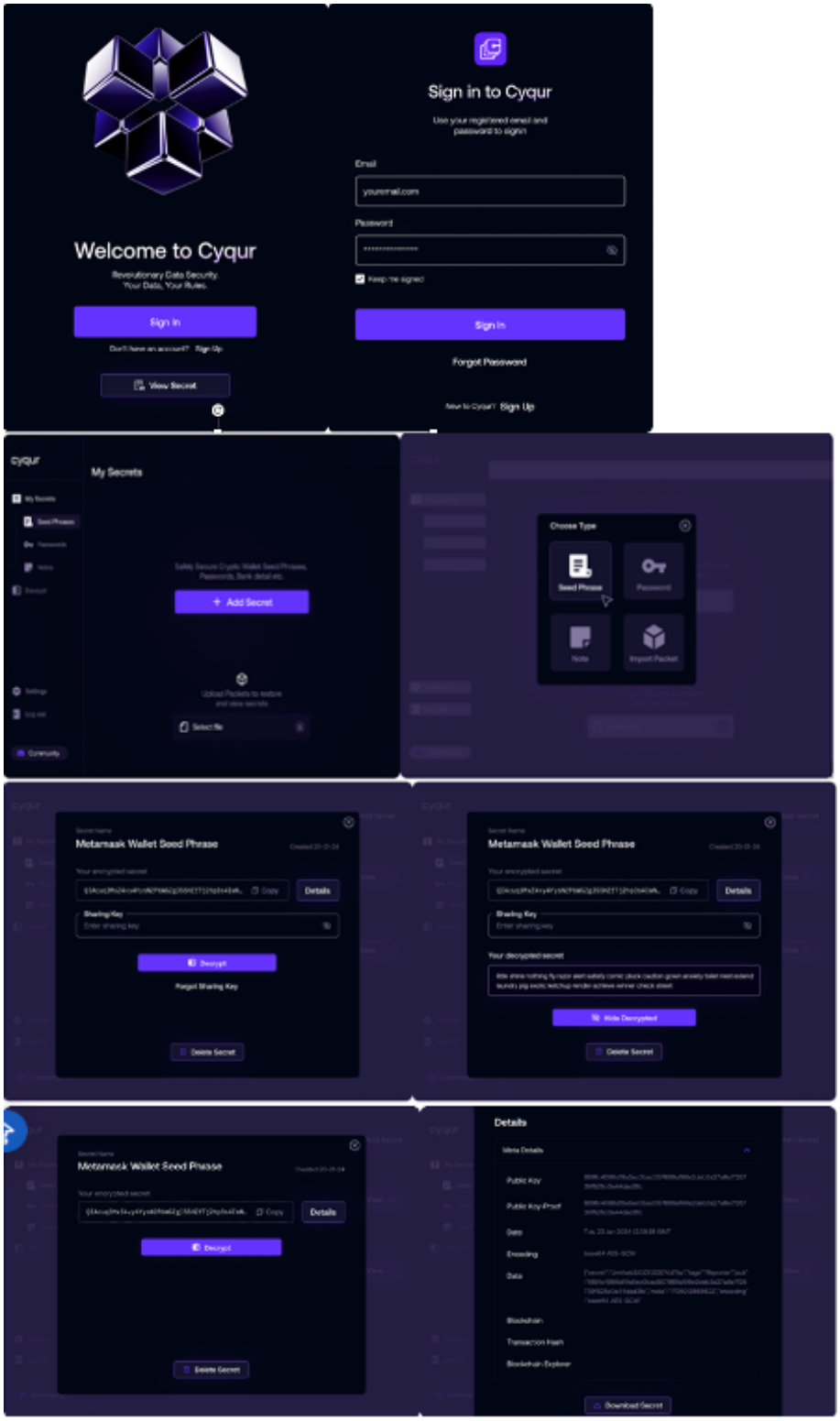
COLRs, who may be passive custodians, will be encouraged to store a Cyqur viewer with the fragments and metadata they are holding. This viewer will be a self contained javascript, CSS and HTML application that can run through a recognised web browser and therefore allow the COLR to recover the secrets through the uploading of packets of fragments.

5. Implementation

Cyqur is implemented as an open-sourced Chrome extension1 in JavaScript (JS) and is governed by the rules of what an extension can and cannot do. There are no dependencies on third party libraries such as jQuery or calls to CDNs. However, if there is integration with cloud providers or 2FA authentication token providers, we aim to disclose how this interaction works. We also aim to create threat models to help users understand their own risk appetites. The code is minified and can be unminified easily. There is no obscurification. The encrypted secret saved to the vault is in JSON format. The secret is encrypted as a (name, value) pair. The JSON is also encrypted. Certain data is stored in the browser. This can be in hidden fields on the webpage; session storage; local storage; local IndexedDB; Cookies (although Cyqur does not use Cookies). No data is sent to a server or requested from a server.

The only exception is where a token has been given to the user for the purposes of being able to backup fragments to other services along with tokens issued for usage of the app. Third party cloud service providers may also be used to store and for retrieval of fragmented secrets. The vault uses the IndexedDB storage. Local Storage is used for faster interaction and as a cache. Cyqur comes with a “Wipe Everything" option that deletes all the browser data. However,

Figure 1: Cyqur web application



the indexedDB may be open and may not delete properly so it is advised the app and the browser are closed down and then reopened.

6. Usage Modes

6 Usage Modes Cyqur is a powerful tool for end users. We describe below three usage modes for Cyqur depending on the actual application scenario.

Use Case I: Cyqur in its simplest form serves as a basic encryption and decryption application. Users can enter a piece of data and a password and the tool returns the encrypted data. This encrypted data can be copied and passed to a recipient via any insecure channel. If the recipient has the password, then he/she can decrypt the secret using Cyqur.

Use Case II: As Cyqur has the ability to encrypt and decrypt arbitrary-sized data, an extension of the above use case is in storing and managing a secret that has value if lost or misappropriated (e.g., seed words to a crypto wallet). To use Cyqur as a secret data management tool, users with highly sensitive data should use a browser with no other extensions installed, turn off the Internet, open Cyqur and enter a memorable password. Once the data is encrypted on the device, users could store and then save the data in a local folder (not the ones that are auto stored on iCloud, Dropbox or other helpful cloud service), wipe everything (settings), close and reopen the browser and turn the internet on and email or message the encrypted data.

Use Case III: This mode triggers all the features of Cyqur. In this mode, a given piece of data owned by the user gets encrypted using a password supplied by the user and the encrypted secret is then fragmented into 10 parts and distributed across different trusted parties for example cloud providers. A packet that records the location of these fragments is also generated and then distributed to interested parties. This means all the Cyqur data can be deleted as the fragments will be held in the cloud.

7. Discussion and Future Work

The current implementation and product offering of Cyqur allows users to store and manage different types of data including but not limited to passwords, seed phrases for blockchain accounts, API keys among others. As a future work,
we plan to extend the functionalities of Cyqur with the following features:

Hardware support: Off-line hardware devices such as Ledger and Yubikey are often used to generate tokens and keys as the secrets stored therein never leave these devices. Therefore, it will be useful to be able to connect the Cyqur browser extension with these hardware devices. Many of these hardware devices come with SDKs that allow community developers to build clients that can communicate with the device firmware.

Cloud backup: This optional feature would routinely back up data to different cloud locations to ensure data recoverability. However, this adds dependency on the cloud service provider and comes with certain risks.

Collaboration-friendliness: An extension of the current implementation would be to allow teams within a company to manage a shared secret. This would allow for example sharing a secret with other fellow team members, revoking access, etc. Third-party integration such as Google Authentication for 2FA would also be a useful feature addition. The system can also be extended to serve as a document storage platform such as Dropbox.

Support for other cryptographic primitives: To make the system flexible, we would like to add support for other cryptographic primitives, in particular those that provide post-quantum security. Moreover, in the current implementation, the number of shares that get created is fixed which may

not be ideal for many users. A future work would consist in letting the users decide on the number of shares.

8. Conclusion

In this work, we introduced Cyqur — a simple yet powerful secret data management system. Cyqur facilitates the storage and transmission of secrets and is inspired by Shamir's secret sharing protocol. Cyqur has several usage modes. In its most basic form, it serves as a simple password-based encryption and decryption tool and therefore can act as a basic fully-local password manager. Unlike traditional password managers that store the data on the device and often on a server too, Cyqur by default stores all data locally. As a result, it eliminates attack vectors that target password manager vendors. As the data is stored locally, users can also delete part or everything at the click of a button. This leaves no audit trail but there is no way to recover. Its second usage mode gives the possibility to recover the data by allowing them to download the encrypted data and store them in a password manager, email, cloud or elsewhere. In the most advanced mode, it allows users to encrypt a piece of data, and then fragment the encrypted data into 10 parts and distribute those across different trusted parties. The fragments are then indexed and the indexing is recorded on the blockchain. If the user has stored the shares with a bad actor, then the latter would need to have access to the location of all the other remaining shares along with the password to reveal the secret. Cyqur has been implemented as a browser extension and is open sourced. It uses browser cryptography and aims to minimise exposure to third-party code or services and all the key features work offline for added security.

References

[1] Ka-Ping Yee and Kragen Sitaker. Passpet: convenient password management and phishing protection. In Lorrie Faith Cranor, editor, *Proceedings of the 2nd Symposium on Usable Privacy and Security, SOUPS 2006, Pittsburgh, Pennsylvania, USA, July 12-14, 2006*, volume 149 of *ACM International Conference Proceeding Series*, pages 32–43. ACM, 2006.

[2] Jeff Jianxin Yan, Alan F. Blackwell, Ross J. Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Secur. Priv.*, 2(5):25–31, 2004.

[3] Lance Whitney. Lastpass ceo reveals details on security breach, 2011.

[4] Vu Gaba Vineb. b.lock: the blockchain-powered password manager, 2018.

[5] Joe Siegrist. Lastpass hacked – identified early & resolved, 2015.

[6] Nina Polshakova. Secure password storage on the bitcoin blockchain, 2019.

[7] Josh Ong. Anonymous hackers claim to leak 28,000 paypal passwords on global protest day, 2012.

[8] Elinor Mills. Linkedin confirms account passwords hacked, 2012.

[9] Brian Krebs. Onelogin: Breach exposed ability to decrypt data, 2017.

[10] LastPass. Lastpass, 2022.

[11] Password Managers. Which password managers have been hacked?, 2022.

[12] Sharon Gaudin. Hackers compromised nearly 5m gmail passwords, 2014.

[13] 1Password. 1password, 2022.

[14] Maliheh Shirvanian, Stanislaw Jarecki, Hugo Krawczyk, and Nitesh Saxena. SPHINX: A password store that perfectly hides passwords from itself. In Kisung Lee and Ling Liu, editors, *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*, pages 1094–1104. IEEE Computer Society, 2017.

[15] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[16] Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C. Mitchell. Stronger password authentication using browser extensions. In Patrick D. McDaniel, editor, *Proceedings of the 14th USENIX Security Symposium, Baltimore, MD, USA, July 31 - August 5, 2005*. USENIX Association, 2005.

[17] Hirak Ray, Flynn Wolf, Ravi Kuber, and Adam J. Aviv. Why older adults (don't) use password managers. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 73–90. USENIX Association, 2021.

[18] Shikun Zhang, Sarah Pearman, Lujo Bauer, and Nicolas Christin. Why people (don't) use password managers effectively. In Heather Richter Lipford, editor, *Fifteenth Symposium on Usable Privacy and Security, SOUPS 2019, Santa Clara, CA, USA, August 11-13, 2019*. USENIX Association, 2019.

[19] Robert H. Morris Sr. and Ken Thompson. Password security - A case history. *Commun. ACM*, 22(11):594–597, 1979.

[20] Daniel McCarney, David Barrera, Jeremy Clark, Sonia Chiasson, and Paul C. van Oorschot. Tapas: design, implementation, and usability evaluation of a password manager. In Robert H'obbes' Zakon, editor, *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*, pages 89–98. ACM, 2012.

[21] J. Alex Halderman, Brent Waters, and Edward W. Felten. A convenient method for securely managing passwords. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 471–479. ACM, 2005.

[22] Masayuki Fukumitsu, Shingo Hasegawa, Jun-ya Iwazaki, Masao Sakai, and Daiki Takahashi. A proposal of a password manager satisfying security and usability by using the secret sharing and a personal server. In Leonard Barolli, Makoto Takizawa, Tomoya Enokido, Antonio J. Jara, and Yann Bocchi, editors, *30th IEEE International Conference on Advanced Information Networking and Applications, AINA 2016, Crans-Montana, Switzerland, 23-25 March, 2016*, pages 661–668. IEEE Computer Society, 2016.

[23] Merete Elle, Stig Mjølsnes, and Ruxandra Olimid. Distributed personal password repository using secret sharing. 09 2018.

[24] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 538–552. IEEE Computer Society, 2012.

[25] Ramakrishna Ayyagari, Jaejoo Lim, and Olger Hoxha. Why do not we use password managers? a study on the intention to use password managers. *Contemporary Management Research*, 15(4):227–245, Dec. 2019.

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