MonoPass: A Password Manager without Master Password Authentication

Hyeonhak Jeong
uosmorrie24@uos.ac.kr
University of Seoul
Seoul, Republic of Korea

Hyunggu Jung*
hjung@uos.ac.kr
University of Seoul
Seoul, Republic of Korea

ABSTRACT

Passwords are the most common user authentication methods. Password policies regulate passwords to a certain degree of complexity, which also makes it difficult for users to create and remember passwords. Password managers improve both security and usability by allowing users to memorize only one master password. However, authenticating to the password manager with the master password has the risk of exposing all passwords when the security of the password manager is breached. We present a password manager, MonoPass, that leverages a master password to regenerate consistent passwords across a variety of devices and passes password metadata through a central server. MonoPass enables users to synchronize passwords without storing user data on the server and without using authentication with the master password.

CCS CONCEPTS

• **Security and privacy → Usability in security and privacy**; *Authentication.*

KEYWORDS

Password management, Password generator, Password manager, Hashing

ACM Reference Format:
Hyeonhak Jeong and Hyunggu Jung. 2021. MonoPass: A Password Manager without Master Password Authentication. In *26th International Conference on Intelligent User Interfaces (IUI '21 Companion), April 14–17, 2021, College Station, TX, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3397482.3450720

1 BACKGROUND

Passwords are known as common user authentication methods. To protect passwords from adversaries, many websites (e.g., Google and Facebook) demand users to follow password policies that enforce users to create passwords with complexity to protect passwords (e.g., passwords should be longer than 9 characters, contain at least one special character, at least one lowercase character, and at least one single-digit number). However, Komanduri et al. reported that the password policy of the websites made it difficult for users to create and memorize passwords [13]. In order to use multiple websites and follow the password policy, some users use the same password for multiple websites, or use a small change of existing password [7]. These coping strategies could weaken the security of password authentication [11]. For example, when the password of one website is exposed, the password of another website using the same password would be also exposed. Password managers may provide a solution for users to easily create and remember passwords. For example, the password managers allow users to remember only one master password. They generate or encrypt multiple passwords of high complexity using the master password. Nevertheless, if the password manager stores all the passwords, an attack against the password manager risks exposing all passwords. For instance, if the master password authentication of the password manager is breached through offline brute force attacks, all passwords stored in the password vault will be exposed. Therefore, users would need a system that manages their passwords securely. In the following sections, we present MonoPass, a password manager that creates passwords and synchronizes passwords across multiple user devices without storing any passwords.

2 RELATED WORK

Prior work proposed a variety of password generation algorithms and management systems [1–5, 8–10, 15–21]. One way to create passwords while protecting them from attacks on storage was to store only partial data of the password and hash this data with the master password to generate final password [3, 9, 10, 16, 18]. For example, Marky et al. [16] suggested a password generation scheme that generates multiple passwords by hashing the master password. Furthermore, previous studies showed that the functionality of the password manager creates potential targets of the security attack. For example, PALPAS [10] offered a password synchronization function but used authentication with the master password which is exploitable by adversaries. To our knowledge, no password management systems offered password regeneration, password update and password synchronization without password storage, data storage on a central server and authentication with a master password. In a similar way as used by prior work [3, 9, 10, 16, 18], MonoPass generates passwords by hashing them with a master password and salt, because it stores less information about the password. However, MonoPass does not require the authentication with the master password, even for the password synchronization.

3 SYSTEM OVERVIEW

MonoPass consists of three components: a password generator, a password manager and a central server. The password generator performs three activities: (1) receiving a master password from

*Corresponding author.

IUI '21 Companion, April 14–17, 2021, College Station, TX, USA

ACM ISBN 978-1-4503-8018-8/21/04.
https://doi.org/10.1145/3397482.3450720

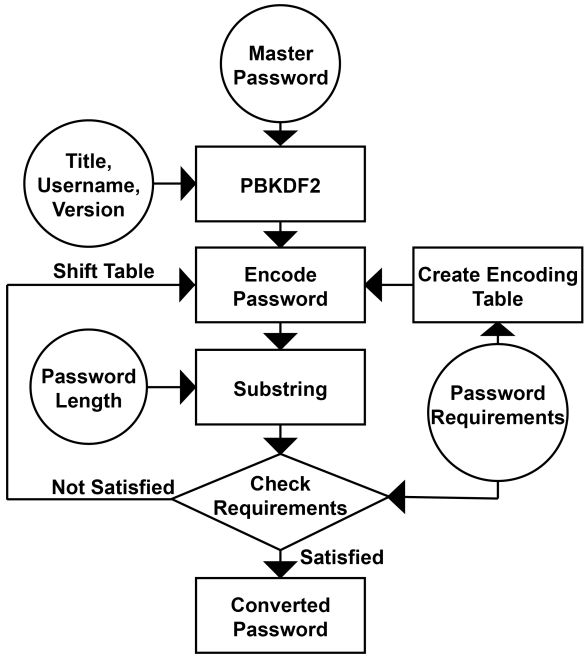

Figure 1: A password generation flowchart consists of user inputs as circle symbols, generation procedures as square symbols, and one password verification step as a diamond symbol.

the user, (2) collecting metadata to be used as a hash salt, and (3) hashing them to generate a final password. The detailed password generation procedure is described in the Section 3.1. The password manager controls the creation, modification, and deletion of password metadata, such as the username and password length. These functions are described in the Section 3.2. The central server controls the synchronization of data between the two password managers. The synchronization method is described in the Section 3.3.

3.1 Password generator

The password generator creates passwords by converting the master password entered by the user. Fig. 1 depicts the password generation process. The password generator receives the following metadata from the user: name of the password (e.g., the name of the website), username, version of the password (e.g., the password generation date), and password policy (e.g., length of the password, required special characters). The password generation procedure consists of hashing using the key derivation function, PBKDF2 [12] and an encoding process to follow the password policy. In the first hash, the password generator converts the master password using PBKDF2 and HMAC-SHA256 or HMAC-SHA512 [6, 14] for the hash functions. In this process, the metatdata (name of the password, user name and the versions of the password) is concatenated and used for the salt of PBKDF2. The password generator encodes the hash output to follow the password policy. The generator uses the password policy information (such as password length, special character types, uppercase alphabet, lowercase alphabet and numeric requirements) from the metadata to encode the hash. The generator creates an encoding table with the password policy requirements of letter case, number, and special character type. For example, if the password should contain at least one lowercase alphabetic character, at least one digit, and at least one special character "_ (underbar)", the encoding table will be filled with 26 lowercase alphabetic characters (a to z), 10 single-digit numbers (0 to 9), and one special character. Since the size of the encoding table is 256, after filling in the first 37 characters, repeating from the lowercase alphabet to fill the table. The generator converts the hash output using the encoding table and shortens the password by the password length required by the password policy. Since the hashing and encoding process do not guarantee that the password follows the password policy, the generated password must be verified if it meets the password requirements. If the encoded password does not follow the policy, the generator pushes the encoding table 26 characters and re-encodes the hash except a few verified characters. Otherwise, the generator displays the final password to the user.

3.2 Password manager

The password manager manages the metadata used for password generation. The password manager has three main tasks: creation, modification, and deletion of password metadata.

3.2.1 Creation of password metadata. Users are allowed to create new metadata by pushing the "Create new password" button in the password manager's options menu (see Fig 2). The password manager provides the user with a metadata entry form. Assuming that only one master password is used, one metadata corresponds to one password. Therefore, a user who uses multiple passwords would create metadata corresponding to each password.

3.2.2 Modification of password metadata. Users are allowed to update the final password by modifying the metadata of the password instead of the master password. When the user selects one of the metadata shown on the screen and pushes "Modify this password" button (see Fig 2), the password manager presents the user with a form in which the modified metadata should be entered as when creating metadata. If a simple password update is required, the user only needs to update the password version without changing the rest in the metadata. If the password policy is changed so it is necessary to change the password length or character types, the user should update the corresponding contents in the metadata. When the user completes the input, the password manager overwrites the modified metadata over the previous metadata.

3.2.3 Deletion of password metadata. Users are allowed to delete metadata from the password manager. If the user select one of the metadata and push the "Delete this password" button (see Fig 2), the meta data will be deleted.

3.3 Central server

In order to use the same password on other devices that do not have the same metadata, the function of transmitting the metadata to another device is essential. MonoPass synchronizes the metadata through the central server. First, the user enters an arbitrary identification code in the device to send the password metadata. If the code is the same as one of the codes existing on the server, the

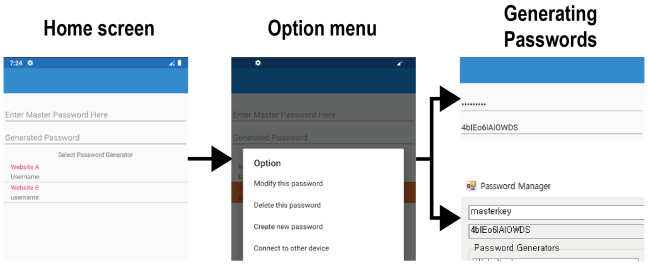

Figure 2: Implemented screens. Left and center are MonoPass Android app implementation screens. The right side shows a comparison of the password generation results on two different devices (i.e., Android smartphone and Windows PC).

server asks the user to enter a new code. If the code is a new one, then, the user enters the same code from the other device to receive the metadata. The central server identifies the two devices to which the metadata is to be delivered by the identification code. The user should confirm the connection on both devices to prevent incorrect connection. Metadata would be passed from one device to the other via the central server. Users are allowed to obtain consistent final passwords by using the same metadata and the master password (see Fig 2). To avoid the risk of password exposure from the security attacks against the central server, the server should keep the data only in the main memory temporarily and remove the metadata when all the metadata is transmitted.

4 CONCLUSION AND FUTURE WORK

We present MonoPass for generating, managing, and synchronizing passwords. MonoPass provides users with password generation through the existing password generation scheme that hashes the master password with a salt. The key contributions of this study are twofold: implementing a password management system to show the device independence in two different environments, and showing a method of synchronizing passwords while maintaining the security advantages of the password generation algorithm. MonoPass provides security benefits against adversary threats. In addition to that exposing one password may not leak other passwords or the master password, the master password exists in the device memory only during the password creation and input process. Also, even if adversaries steal all of MonoPass' data, they still would need to perform brute force attacks against real services to get the master password. However, there still remain limitations in our study. First of all, password synchronization is only possible through a central server. It would not be possible to transmit the metadata to another device without an Internet connection. Second, we did not evaluate the usability and feasibility of MonoPass with users in real-life scenarios. It may be inconvenient for users to enter metadata, to enter a master password, and/or to copy and paste passwords from MonoPass. Last, while we did not conduct a security analysis of MonoPass, we may need to examine whether a password generated by MonoPass can protect its master password in a specific threat scenario. Future work still remain to integrate additional password synchronization methods, such as using a USB connection, into MonoPass to increase device independence. Also, we plan on evaluating the usability and feasibility of MonoPass with potential target users. MonoPass has potential to manage passwords for users who use passwords across multiple devices and are willing to secure the master password without storing passwords in any form. We hope that MonoPass enables users to interact with an intelligent user interface for authentication to access information technology and provides inspiration for researchers investigating systems that store, transmit, and use sensitive information.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2020R1G1A1009133).

REFERENCES

[1] 1Password. 2021. *1Password.* Retrieved January 6, 2021 from https://1password.com
[2] S Agholor, AS Sodiya, AT Akinwale, and OJ Adeniran. 2016. A Secured Mobile-Based Password Manager. In *2016 Sixth International Conference on Digital Information Processing and Communications (ICDIPC)*. IEEE, 103–108.
[3] Maarten Billemont. 2011. *Master Password.* Retrieved January 4, 2021 from https://masterpassword.app/
[4] Hristo Bojinov, Elie Bursztein, Xavier Boyen, and Dan Boneh. 2010. Kamouflage: Loss-resistant password management. In *European symposium on research in computer security*. Springer, 286–302.
[5] Bart Busschots. [n.d.]. *XKPasswd.* Retrieved January 4, 2021 from https://xkpasswd.net/s/
[6] Quynh H Dang. 2015. *Secure hash standard.* Technical Report.
[7] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. 2014. The tangled web of password reuse.. In *NDSS*, Vol. 14. 23–26.
[8] Masayuki Fukumitsu, Shingo Hasegawa, Jun-ya Iwazaki, Masao Sakai, and Daiki Takahashi. 2016. A proposal of a password manager satisfying security and usability by using the secret sharing and a personal server. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 661–668.
[9] J Alex Halderman, Brent Waters, and Edward W Felten. 2005. A convenient method for securely managing passwords. In *Proceedings of the 14th international conference on World Wide Web*. 471–479.
[10] Moritz Horsch, Andreas Hülsing, and Johannes Buchmann. 2015. PALPAS–PAssword Less PAssword Synchronization. In *2015 10th International Conference on Availability, Reliability and Security*. IEEE, 30–39.
[11] Blake Ives, Kenneth R Walsh, and Helmut Schneider. 2004. The domino effect of password reuse. *Commun. ACM* 47, 4 (2004), 75–78.
[12] Burt Kaliski. 2000. *PKCS# 5: Password-based cryptography specification version 2.0.* Technical Report. RFC 2898, september.
[13] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. 2011. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the sigchi conference on human factors in computing systems*. 2595–2604.
[14] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. 1997. HMAC: Keyed-hashing for message authentication.
[15] LogMeIn. 2021. *LastPass.* Retrieved January 6, 2021 from https://www.lastpass.com
[16] Karola Marky, Peter Mayer, Nina Gerber, and Verena Zimmermann. 2018. Assistance in Daily Password Generation Tasks. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. 786–793.
[17] Stephen Ostermiller. 2013. *Random Password Generator.* Retrieved January 4, 2021 from https://passwordcreator.org/
[18] Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C Mitchell. 2005. Stronger Password Authentication Using Browser Extensions.. In *USENIX Security Symposium*. Baltimore, MD, USA, 17–32.
[19] Kevin Albert Schmittle. 2018. *Regenerative Password Manager.* Ph.D. Dissertation. New Mexico Institute of Mining and Technology.
[20] Elizabeth Stobert and Robert Biddle. 2014. A password manager that doesn't remember passwords. In *Proceedings of the 2014 New Security Paradigms Workshop*. 39–52.
[21] Rui Zhao and Chuan Yue. 2014. Toward a secure and usable cloud-based password manager for web browsers. *Computers & Security* 46 (2014), 32–47.