Proceedings of the 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design

2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD) | 978-1-6654-0527-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/CSCWD54268.2022.9776045

PassImg: A Secure Password Generation and Management Scheme without Storing

Yuhua Yin, Julian Jang-Jaccard, Nilufar Baghaei
School of Natural and Computational Sciences
Massey University
Auckland, New Zealand
yuhua.yin.1@uni.massey.ac.nz, j.jang-jaccard@massey.ac.nz, n.baghaei@massey.ac.nz

***Abstract*—Text password is an important authentication method for computer supported cooperative systems and is usually required to be memorable, strong, and non-reusable. Some password managers can help users manage site-specific passwords locally or on the cloud. However, existing solutions still have a single point of failure risk if the password management scheme is breached. To address the risks identified in current password managers, we proposed a password generation and management scheme called PassImg that can generate consistent passwords through hashing a master password with a user-specified salt image. In this scheme, a weak master password is also allowed while not affecting the security of the scheme.We've also proposed a solution for offline parameters synchronization, which would decrease online attack vectors. The user-recognizable image as a parameter can be configured on different devices and synchronized through a QR code. PassImg can ensure password usability and security through an offline managing process without storing any data on the server.**

***Keywords*—Password Security, Password generator, Password manager, Hashing**

I. Introduction

In recent years, many computer supported cooperative works have been proposed, which facilitates the user's life and work. However, along with emerging computer services, password security and usability have become a serious challenge for the society [1]. As a widely used authentication method, text passwords can provide good usability and accuracy at a reasonable cost, and it is considered to be playing an important role for a long time [2]. Generally, a password with high entropy is regarded to be stronger which needs to meet a certain length and characters complexity requirements [3]. Previous data breaching incidents have warned that bad habits like reusing passwords on different websites may bring risks to accounts [4]. Although many password security policies are suggesting some best practices, inconvenience and memory burden still make some users choose insecure password strategies. There are some dilemmas in using text passwords, including (1) choosing a dictionary-word based password would be memorable but also guessable (2) making slight changes to a base phrase as a password could meet mandatory complexity policies but vulnerable to dictionary attack (3) reusing passwords for different accounts [5]. To mitigate these problems, many approaches and tools have been proposed. Password managers(PM) are commonly used to help users from memory burdens to manage passwords on different websites using a master password [6]. Although traditional password managers facilitate users to use site-specific strong passwords and improve memorability, brute-force or dictionary attacks targeting the master password could be a vulnerability as the single point of failure [7].

In this paper, we present a password generation scheme called PassImg that allows users to use a master password to regenerate consistent site-specific passwords. PassImg mainly uses a configured image, a master password, and some metadata including website name, username, version, and length as inputs to generate high-entropy passwords as outputs. Two hash algorithms Angron2 and PBKDF2 will be used to generate binary hash values, and base85 encoding will convert binary values into high-entropy strings with demanded character sets. Images usually have larger amounts of data than text metadata, so they can mitigate offline brute-force attacks. Also, as visual memory is better accepted by human beings, it helps users to recognize, memorize, save, and reload the configuration. Once an image has been configured at first use, users only need to remember the master password in daily use. Besides, weak master password are allowed for usabilities. PassImg supports using a QR code to synchronize configurations on different clients without a central server or network for transmission, which can largely reduce attack vectors. We have implemented a desktop application as a demonstration for the scheme. In the following sections, we will introduce and evaluate PassImg.

II. RELATED WORK

Password managers (PM) can manage complex and non-literal site-specific passwords only through a master password. There are two main approaches for PMs. In the wallet approach, the master password encrypts and protects the file where site-specific passwords are stored so randomly generated strong passwords can be used for different accounts [6]. There is also a deterministic strong password generation scheme called the hash approach used by some password managers [8]. It uses a hash function to generate site-specific passwords through the website URL and the master password. The hash approach has the advantages of offline availability, cross-devices, and no need for storage [9].

There are many PMs designed by the hash approach. PwdHash provides a browser-side password generation scheme, which uses the MD5 hash function to generate deterministic site-specific passwords utilizing the master password and domain names of different websites as secret and salt respectively [10]. It simplifies generating and restoring process of site-specific passwords and can avoid single points of failure due to centralized password storage. However, it has some shortcomings including transportability issues and vulnerability to offline brute-force attacks [6]. Passpet, as a browser plug-in password manager, adds pet icon and pet name as a local setup based on the hash approach, aiming to avoid offline attacks [11]. Since the setup parameters will be stored on the central server, there are parameter breaching worries. Biddle et al designed ObPwd to replace the text master password and use a digital Object's (file, image) hash to generate site-specific passwords [12]. Non-text objects largely avoid dictionary attacks and increase the difficulty of brute-force attacks. However, transmitting objects on different devices will increase inconvenience. Besides, using digital Objects to regenerate passwords are slower than text master password. After changing a password, users also need to memorize different objects. Marky et al proposed a scheme adding metadata such as account name, username, version to generate passwords, which can increase the entropy of the generation process [13]. BCrypt that is considered as a secure hash algorithm for passwords is used in their scheme. Then, PassMan simplified this process, only needing to set up metadata the first time it is used on a device [14]. Nevertheless, numerous meta fields may increase memory burdens. A central server has been introduced in the designs of MonoPass and AutoPass [15], which can be used to store and transmit metadata. AutoPass needs users' registrations to transmit metadata, while MonoPass uses real-time identify codes for synchronization [16]. Although using a server to transmit metadata can bring convenience to users across devices, the central server may have a single point of failure issue. In addition, users lack trust in online solutions [6]. PassImg uses Argon2, one of the most advanced password hashing algorithms, to generate a password. Different from previous works, PassImg emphasizes the recognizability of images as a file object and sets it as a unique security configuration. In addition, configuration synchronization between different devices does not involve the network.

III. Scheme Overview

In this section, we will explain the workflow of PassImg. PassImg has three main functional components: password generation module, password management module, and parameter synchronization module. The password generation module is mainly responsible for generating site-specific passwords based on user inputs. Each of the inputs has different purposes

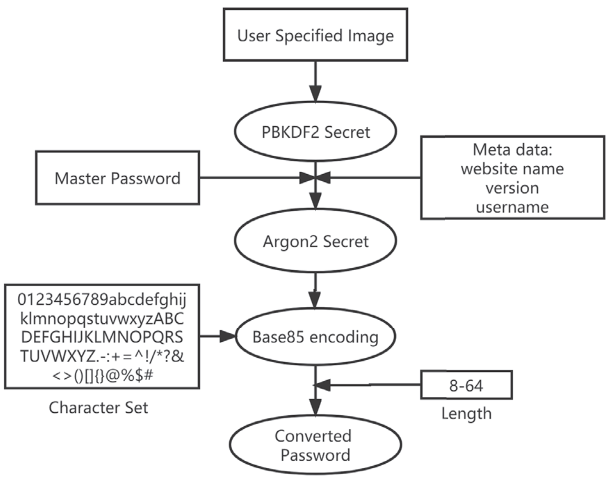

User Specified Image
PBKDF2 Secret
Master Password
Meta data: website name version username
Argon2 Secret
0123456789abcdefghij klmnopqstuvwxyzABC DEFGHIJKLMNOPQRS TUVWXYZ.-:+=^!/*?& <>()[]{}@%$#
Character Set
Base85 encoding
8-64
Length
Converted Password

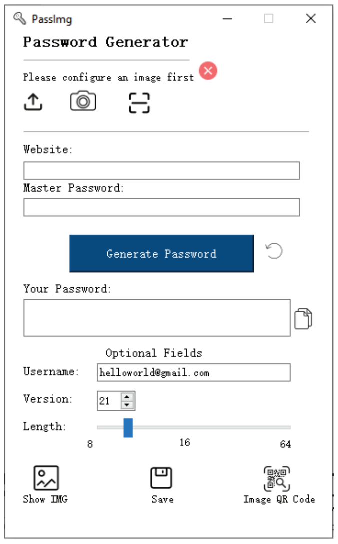
Fig. 1. Password generation flowchart. (Squares are user inputs and ovals are generation procedures)

for the password generation process. There are three types of inputs: image configuration, master password, and metadata. The password management module mainly manages the meta-data used by users daily, avoiding repeated input to ensure user experience. The parameter synchronization module provides the function of transmitting configuration information, which can ensure that all devices have the same configuration information and therefore can generate a consistent password. The explanation of these three parts will be shown in detail in the following three parts.

A. Password Generation

The password generation module generates a password by converting the inputs entered by the user. Figure 1 shows the flow chart of password generation. First of all, when using PassImg for the first time, the user will be asked to select an image as a parameter to ensure the uniqueness of the configuration. There are many ways to be provided at this step including uploading an image, taking a photo through the camera, and importing parameters through the previous configuration QR code. Configured images are suggested to be easy to recognize. For example, an image with special meaning to the user can help the user to remember. However, it is not recommended to use default images in the system or images downloaded from the Internet, because there are risks for brute force attacking. The hash algorithm (SHA256, SHA384, or SHA 512) in PBKDF2 [17] will be used to hash the image, and then the hashed value will be stored in the img_hash variable in the configuration file.

Subsequently, the user can enter the master password and some other metadata including website name, version, username, and length. Metadata is mainly used to distinguish different passwords to ensure that the same master password can generate different site-specific passwords. The name or URL of the web service can be entered in the website input. In



Website:

Master Password:

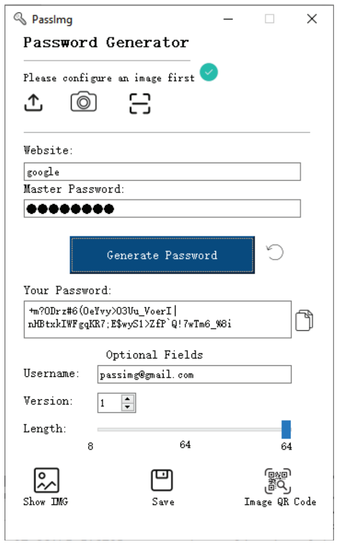
Your Password:

Username: helloworld@gmail.com

Version: 21

Length: 16

Fig. 2. Configuration and User Inputs



Website: google

Master Password: ●●●●●●●●

Your Password: +m?ODrz#6(OeYvy>O3Uu_VoerI nHBtxkIWFgqKR7;E$wyS1>ZfP`Q!7wTm6_%8i

Username: passimg@gmail.com

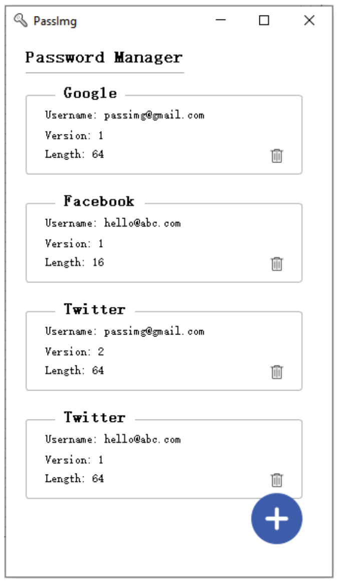
Version: 1

Length: 64

Fig. 3. Password Generation Results

the username input, the username or email address associated with the account can be entered. Version is mainly used to distinguish changed passwords of the same account of the same web service. Figure 2 and figure 3 show the password generator interface.

Argon2 is a memory-hard and CPU-hard hash algorithm for passwords that can mitigate brute-force attacks [18]. Master Password will be used as the secret for Argon2 while website name, image hash, version, and username will be used as salt. Argon2's hash_len parameter should be set properly to ensure the length of the generated value. The generated binary result will be converted by base85 encoding. Base85 encoding contains all 0-9, a-z, A-Z, and special characters, so the final output may be any combination of all the characters. According to the requirements for strong passwords in the NIST 800-63-3 guideline [19], special characters are no longer enforced



Google
Username: passimg@gmail.com
Version: 1
Length: 64

Facebook
Username: hello@abc.com
Version: 1
Length: 16

Twitter
Username: passimg@gmail.com
Version: 2
Length: 64

Twitter
Username: hello@abc.com
Version: 1
Length: 64

Fig. 4. Password Manager Interface

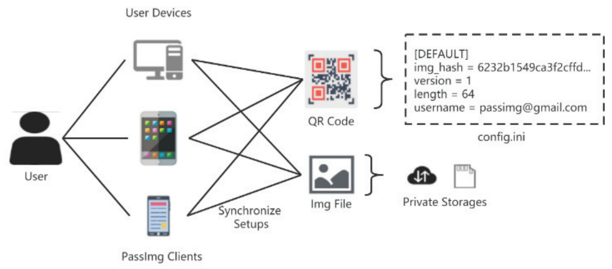
but the length is considered more important (maximum 64 characters). In this scheme, it is not mandatory to generate all types of characters, but an 8-64 length option is provided. The final password will be cut based on the length selected by the user.

B. Password Management

The role of the password management module is to help users manage the metadata of their daily passwords, but it does not store the user's master password or generated passwords. Website, username, and version can be used to distinguish different password entries. The first step is to create a relevant password and click the save button (see figure 3) to save relevant metadata to the list in the password manager interface. In daily use, if the metadata of relevant passwords has been saved in the password manager list, the user can click on the item and then enter the master password to regenerate the password (see figure 4). Also, items in the password manager list can be deleted by clicking the delete button.

C. Parameter Synchronizing

To facilitate users to regenerate the same password based on the same metadata and master password on different parameters and other default settings in the form of a QR code, and the other is to reconfigure the configuration image on different devices from backup. The synchronizing workflow is presented in figure 5. To use the QR code to synchronize data, camera permission needs to be obtained on the device to scan the previous configuration QR code. The configuration QR code can be obtained from the Image QR Code button on the password generator interface (see figure 6). Due to the limitation of the data capacity of the QR code, the synchronization process does not transmit the image itself but only the hash value of the image, so the configured image cannot be displayed on the subsequent



```
[DEFAULT]
img_hash = 6232b1549ca3f2cffd...
version = 1
length = 64
username = passimg@gmail.com
```

Fig. 5. Parameter Synchronization Workflow

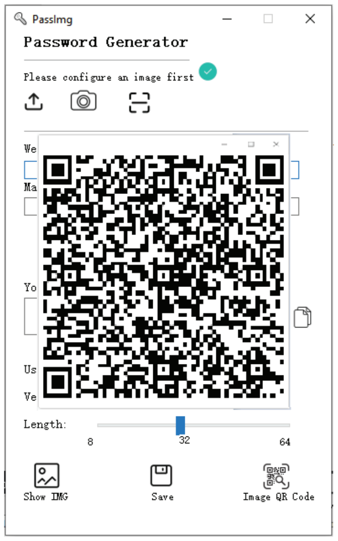


Fig. 6. Retrieve Image QR Code

client. Usually, we recommend users back up and remember the initial configuration images for future use. In the password generator interface(see figure 7), a show IMG button is provided to help review and download the current configuration image.

IV. Security Analysis

In this section, we will analyze the security of PassImg in 4 attack scenarios: online attacks against the password manager, offline attacks against the password manager, keylogging against passwords, and data breaching. In addition, the premise of these analyses is that the adversary has not taken over the physical devices.

A. Online Attack

Online attacks refer to attacks on web services remotely through the network. In the previous password manager solutions, many have introduced a central server to store and transmit master passwords or metadata [11], [13]–[16], which ensures the consistency of password generation on different devices. However, it also brings the problem of increasing the online attack vectors. If an adversary attacks the central

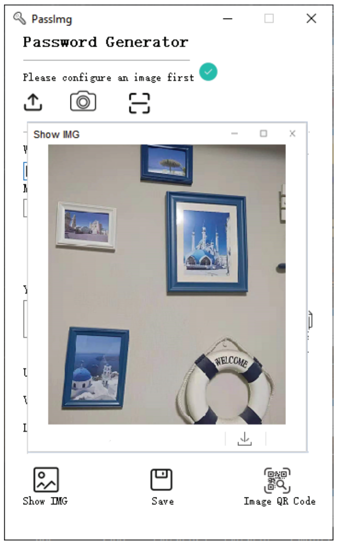


Fig. 7. Show Configured Image

server, it is possible to obtain metadata or master passwords stored in the server. Therefore, it is important to balance the parameter synchronization availability and online attacks vectors. As PassImg introduced synchronization parameters through QR Code, it does not require a central server to store or synchronize, which can avoid online attacks.

B. Offline Attack

Offline attacks against password managers usually refer to dictionary attacks or brute force attacks against the possible master password and metadata if the adversary knows the generation method. Since there is no guarantee that the master password selected by the user is strong, in offline attacks, dictionary attacks and brute-force attacks may generate the accurate password in many attempts. Some previous password generation schemes only use text-based parameters to generate passwords, such as a master password and metadata, which is vulnerable to brute force or dictionary attacks [11], [13]–[16]. PassImg uses the Image as one of the salts and uses Argon2 as the memory-hard and cpu-hard hash algorithm to generate passwords. First of all, as file objects like images have larger amount of data than text, an image is difficult to be brute-forced attacked. Even if the adversary tries brute force attacks against binary values or dictionary attacks against commonly used images, argon2 will exhaust the computing power to make this attempt almost impossible.

C. Keylogging Against Passwords

Keylogging attacks will secretly record the user's keyboard input to steal passwords. There are two scenarios for this attack. One scenario is to steal the user's actual password entered into a web service, and the other is to steal the user's master password used in the password manager. In the first case, PassImg can resist keylogging because users do not need to enter the actual passwords, but only need to copy and paste the generated passwords. In the second case, even if

TABLE I
COMPARISON WITH OTHER PASSWORD MANAGERS USING HASH APPROACH

Password Manager	Storage		Attack Vector				Memorability
	Local	Cloud	Online Attack	Offline Attack	Keylogging	Data Breaching	
PwdHash [10]	Y	N	N	Y	Y	Y	easy
Passpet [11]	Y	Y	Y	N	N	N	hard
ObPwd [12]	Y	N	N	N	N	N	hard
MonoPass [15]	Y	N	N	N	Y	N	easy
AutoPass [16]	Y	Y	Y	N	Y	N	easy
Our Approach(PassImg)	Y	N	N	N	N	N	easy

the adversary obtains the master password, he cannot generate accurate passwords without the unique image configuration. It can be seen that many previous password generation scheme still requires users to use a strong master password to ensure security, but there is still a dilemma that the stronger the master password, the harder it is to remember [10], [13]–[16]. In PassImg, it is not mandatory to use a strong master password but allow to use a simple or memorable master password because the unique image configuration can guarantee the uniqueness of the generated password.

D. Data Breaching

In recent years, some major attacks have caused data breaching of some important password databases, which has brought challenges to password security and password management [4]. As not all web services use a secure hashing strategy for the user's password, using a strong site-specific password can reduce the risk of password cracking. PassImg follows the latest NIST guide for authentication and allows users to generate 8-64 length passwords consisting of any possible characters [19]. In addition, since the hash approach is used to generate passwords in PassImg, the passwords generated for different websites will be completely different. Even if the password of a certain website is cracked, the passwords of other websites are still secure.

V. Discussion

Comparing our scheme with other password managers using the hash approach, PassImg performs well in both security and usability (see table I). In terms of usability, although PassImg only supports local storage, due to its synchronization mechanism, users can well synchronize related configuration information across different devices. Without a central server for synchronization, PassImg is resistant to various attack types and can reduce potential attack vectors. In addition, the configuration of PassImg mainly depends on user specified Images instead of text secret. Since visual memory is usually more durable than literal memory, PassImg has good memorability and usability.

VI. Conclusion and Future Work

In this paper, we proposed PassImg as a secure password generation and management scheme without storage that facilitates users in managing passwords, and made a security analysis in 4 scenarios. PassImg helps users generate site-specific strong passwords by hashing the image, master password, and metadata configured by the user. In the security analysis, we found that PassImg can mitigate online attacks, offline attacks, keylogging attacks, and data breaching. The main contributions of this research are introducing the image configuration as a parameter that can mitigate brute-force attacks, and proposing a synchronization method that does not involve a central server. The image configuration ensures the uniqueness of the password generated by the client. Even if the adversary uses the same metadata and master password, correct passwords will not be regenerated without correct image configuration. An image as a parameter is easy to recognize and remember but difficult to be brute-force attacked. We provide three flexible configuration methods in the implementation, including uploading local images, taking a photo by camera, and obtaining from previous configurations. For configuration synchronization, the QR code method avoids a central server and the network, thus reducing the attack vectors. As the QR code can only synchronize the hash value of the image and cannot restore the image itself, we also encourage users to back up the configured images in time, whether through local storage, portable storage, or cloud storage. It can be said that the backups are completely controlled by the user because these storages are not centralized and not labeled. However, PassImg also has some limitations. First of all, saved metadata in the password manager list cannot be synchronized across devices and need to be manually set in each client. Secondly, the image selection procedure needs to be formalized, and there is a risk of brute force cracking in using network pictures or system default pictures. In addition, users may neglect the importance of image backup by using QR code synchronization. In future work, we plan to implement the scheme for multiple platform solutions including browser extensions and mobile applications. Besides, we will conduct a user study to validate the usability of PassImg and discover potential problems from users' perspectives.

References

[1] B. Naqvi and A. Seffah, "Interdependencies, conflicts and trade-offs between security and usability: why and how should we engineer them?" in *International Conference on Human-Computer Interaction*. Springer, 2019, pp. 314–324.

[2] K. Siddique, Z. Akhtar, and Y. Kim, "Biometrics vs passwords: a modern version of the tortoise and the hare," *Computer Fraud & Security*, vol. 2017, no. 1, pp. 13–17, 2017.

[3] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, “Of passwords and people: measuring the effect of password-composition policies,” in *Proceedings of the sigchi conference on human factors in computing systems*, 2011, pp. 2595–2604.

[4] S. Bhagavatula, L. Bauer, and A. Kapadia, “(how) do people change their passwords after a breach?” *arXiv preprint arXiv:2010.09853*, 2020.

[5] H. Murray and D. Malone, “Evaluating password advice,” in *2017 28th Irish Signals and Systems Conference (ISSC)*. IEEE, 2017, pp. 1–6.

[6] S. Chaudhary, T. Schafeitel-Tähtinen, M. Helenius, and E. Berki, “Usability, security and trust in password managers: A quest for user-centric properties and features,” *Computer Science Review*, vol. 33, pp. 69–90, 2019.

[7] S. Pearman, S. A. Zhang, L. Bauer, N. Christin, and L. F. Cranor, “Why people (don’t) use password managers effectively,” in *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, 2019, pp. 319–338.

[8] K. Bicakci, N. B. Atalay, M. Yuceel, and P. C. Van Oorschot, “Exploration and field study of a password manager using icon-based passwords,” in *International Conference on Financial Cryptography and Data Security*. Springer, 2011, pp. 104–118.

[9] R. H. Rooparaghunath, T. Harikrishnan, and D. Gupta, “Trenchcoat: Human-computable hashing algorithms for password generation,” in *International Conference on Cryptology and Network Security*. Springer, 2020, pp. 167–187.

[10] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, “Stronger password authentication using browser extensions.” in *USENIX Security Symposium*, vol. 17. Baltimore, MD, USA, 2005, p. 32.

[11] K.-P. Yee and K. Sitaker, “Passpet: convenient password management and phishing protection,” in *Proceedings of the second symposium on Usable privacy and security*, 2006, pp. 32–43.

[12] R. Biddle, M. Mannan, P. C. Van Oorschot, and T. Whalen, “User study, analysis, and usable security of passwords based on digital objects,” *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 970–979, 2011.

[13] K. Marky, P. Mayer, N. Gerber, and V. Zimmermann, “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*, 2018, pp. 786–793.

[14] J. B. Billa, A. Nawar, M. M. H. Shakil, and A. K. Das, “Passman: A new approach of password generation and management without storing,” in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*. IEEE, 2019, pp. 1–5.

[15] H. Jeong and H. Jung, “Monopass: A password manager without master password authentication,” in *26th International Conference on Intelligent User Interfaces-Companion*, 2021, pp. 52–54.

[16] F. Al Maqbali and C. J. Mitchell, “Autopass: An automatic password generator,” in *2017 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2017, pp. 1–6.

[17] A. Visconti, S. Bossi, H. Ragab, and A. Calò, “On the weaknesses of pbkdf2,” in *International Conference on Cryptology and Network Security*. Springer, 2015, pp. 119–126.

[18] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: new generation of memory-hard functions for password hashing and other applications,” in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 292–302.

[19] P. A. Grassi, J. L. Fenton, and M. E. Garcia, “Digital identity guidelines [including updates as of 12-01-2017],” *NIST*, 2017.