PUFPass: A password management mechanism based on software/hardware codesign

Qingli Guo a,b, Jing Ye a,b,∗, Bing Li c, Yu Hu a,b, Xiaowei Li a,b,∗∗, Yazhu Lan a,
Guohe Zhang d

a State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, PR China
b University of Chinese Academy of Sciences, Beijing, 100049, PR China
c Duke University, Durham, NC, 27708, USA
d School of Microelectronics, Xi’an Jiaotong University, Xi’an, Shanxi, 710049, PR China

A R T I C L E   I N F O
Keywords:
Password
Password management mechanism
PUF
Security
Usability

A B S T R A C T

Secure passwords need high entropy, but are difficult for users to remember. Password managers minimize the memory burden by storing site passwords locally or generating secure site passwords from a master password through hashing or key stretching. Unfortunately, they are threatened by the single point of failure introduced by the master password which is vulnerable to various attacks such as offline attack and shoulder surfing attack. To handle these issues, this paper proposes the PUFPass, a secure password management mechanism based on software/hardware codesign. By introducing the hardware primitive, Physical Unclonable Function (PUF), into PUFPass, the random physical disorder is exploited to strengthen site passwords. An illustration of PUFPass in the Android operating system is given. PUFPass is evaluated from aspects of both security and preliminary usability. The security of the passwords is evaluated using a compound heuristic algorithm based PUF attack software and an open source password cracking software, respectively. Finally, PUFPass is compared with other password management mechanisms using the Usability-Deployability-Security (UDS) framework. The results show that PUFPass has great advantages in security while maintaining most benefits in usability.

1. Introduction

Password authentication is the primary way for ensuring security, but it has too many problems. Secure passwords should be unique and have high entropy. However, they are hard for users to remember. The memorability problem is further aggravated by the increasing number of passwords for users to manage. These problems lead users to insecure coping mechanisms such as choosing simple passwords that are easy for adversaries to guess, and reusing passwords across different sites. In addition, common use of password offers no resistance to phishing attack, shoulder surfing attack, spyware, and so on.

Password managers are designed to alleviate the memory burden of users. To guarantee security, the master password is leveraged to protect site passwords or generate site passwords of high entropy. There are mainly two kinds of password managers: the retrieval password manager and the generative password manager. The retrieval password manager simply stores site passwords locally in plaintext or protects site passwords under a master password. The site passwords are retrieved when required. The generative password manager can generate unique high-entropy passwords for different sites by hashing (e.g. PwdHash [1], Versipass [2]), or key stretching (e.g. Password Multiplier [3], Passpet [4], Client-CASH [5]). Password managers may also pro-

☆ This paper is supported in part by National Natural Science Foundation of China (NSFC) under grant No. (61532017, 61704174, 61432017, 61521092).
∗ Corresponding author. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, PR China.
∗∗ Corresponding author. State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, PR China.
E-mail addresses: guoqingli@ict.ac.cn (Q. Guo), yejing@ict.ac.cn (J. Ye), bing.li.ece@duke.edu (B. Li), huyu@ict.ac.cn (Y. Hu), lxw@ict.ac.cn (X. Li), yulan@ict.ac.cn (Y. Lan), zhangguohe@xjtu.edu.cn (G. Zhang).
https://doi.org/10.1016/j.vlsi.2018.10.003
Received 25 July 2018; Received in revised form 10 September 2018; Accepted 6 October 2018
Available online 26 October 2018
0167-9260/© 2018 Elsevier B.V. All rights reserved.
vide extra services such as converting other authentication types into passwords [6–8]. Although password managers offer many benefits, they still have more or less security issues. Retrieval password managers cannot directly avoid password reuse, and the storage of site passwords is a security threat. Generative password managers can generate unique passwords for different sites, but there exists a single point of failure introduced by the master passwords through which the adversary can obtain all the site passwords. What’s worse, the master password is confronted with offline attack, shoulder surfing attack, phishing attack, and so on. Offline attack is considered the most serious.

In recent years, Physically Unclonable Functions (PUFs) have emerged as promising hardware security primitives [9]. Their key idea is to leverage the random physical disorder that occurs during manufacturing integrated circuits (ICs). Manufacturing variations are uncontrollable and unpredictable at the current state-of-the-art of semiconductor fabrication, so they are also physically unclonable, even by the original manufacturer. Therefore, it is intrinsically secure. PUF shows wide application prospects in IP protection [10], device authentication orductor fabrication, so they are also physically unclonable, even by the original manufacturer. Thus, to ensure security, users are asked to generate passwords that are random and unique, but users have difficulty to recall them. Although users can be trained to remember a single secure password, the increasing number of passwords due to the increasing number of accounts no doubt imposes memory burden [19]. To lift memory barrier, users often adopt insecure mechanisms such as choosing low-entropy passwords and reusing them across different sites. Password reuse severely threatens password security. A study [20] shows that a typical user averages types eight passwords each day, and maintains 25 passwords, but actively uses only about seven distinct passwords [21]. As a result, hackers who crack the password of a low-security site may access a high-security site.

To encourage users to choose high-entropy passwords, various password rules (e.g. requirements of minimum length, symbols, numbers, and upper case letters) are adopted by the service provider. However, although these rules effectively increase security [22,23], passwords are still under threat [24]. On the other hand, there are no effective measures taken by service providers to avoid password reuse.

In addition to low-entropy and reuse of passwords, phishing attack and shoulder surfing attack continue threatening password security. Shoulder surfing attack is a direct observation technique to obtain information. Anyone who monitors the password input procedure by looking over the users’ shoulder may memorize the password. In phishing attack, users are lured to a website or an app that looks like the legitimate one to type in their passwords. Both attacks are tough for users to deal with.

2.2. Password managers

Password managers are designed to alleviate the memory burden of users while ensuring security. Users only need to remember one master password, and password managers handle the pairs of user name and site password. More specifically, password managers can transparently act as a middle-man between the users and the service provider. Password managers can be classified into two main categories: the retrieval password manager and the generative password manager.

Retrieval password managers simply store site passwords in a “wallet”, i.e., a password storage medium, instead of human mind. Later, the site passwords can be retrieved and used to log into accounts. Retrieval password managers are components of most modern browsers, including Internet Explorer, Google Chrome, Safari, Firefox, Opera, and so on. Retrieval password managers also have two categories based on whether the site passwords are stored in plain text or encrypted under a master password. Unencrypted password wallets are obviously insecure. However, the master password of an encrypted password wallet is still vulnerable to offline attack. In offline attack, for a password $pwd_0$, which is hashed to $pwd_0$, if an adversary obtains $pwd_1$, he can repeatedly guess $pwd_0$. Every guessed password is hashed and compared with $pwd_1$, until the right one is found. Brute-force attack and dictionary attack are two typical offline attacks. In brute-force attack, the adversary tries every possible password until he succeeds. In dictionary attack, the adversary only tries likely possibilities, such as words in a dictionary, or combinations of words in multiple dictionaries.

Generative password managers allow users to generate multiple site passwords from one master password and facilitate better site password quality and a reduction in password reuse by leveraging hashing, key stretching or salting on the client-side. PwdHash [1], Password Multiplier [3], Client-CASH [5], Versipass [2], and Passpet [4] all belong to the generative password managers. In PwdHash, the user inputs the master password to a domain salted one-way function, which generates site passwords of high entropy. Password Multiplier leverages key stretching, i.e., iterated hash [25], instead of the hash function. Client-CASH exploits a client side cost asymmetric key stretching to reduce the delay of authentication with correct master password. Versipass uses cued graphical password to address the memorability issues. Passpet improves both convenience and security of website logins through a combination of the Password Multiplier and the Petname Tool. Unlike password managers mentioned above, commercial password managers like 1Password [26] and LastPass [27] rely on a trust server to derive site passwords from the master password.

This paper is organized as follows. Section 2 reviews passwords, password managers, and PUFs. Section 3 proposes the PUFPass. Section 4 and Section 5 evaluate the PUFPass based on security and usability respectively. Section 6 compares PUFPass with existing password management mechanisms using the UDS framework. Finally, Section 7 concludes the paper.

2. Previous work

2.1. Passwords

Password is often the primary entryway to access users’ confidential information on a website or in an app. However, too many problems relate to it. In order to ensure security, users are asked to generate passwords that are random and unique, but users have difficulty to recall them. Although users can be trained to remember a single secure password, the increasing number of passwords due to the increasing number of accounts no doubt imposes memory burden [19]. To lift memory burden, users often adopt insecure mechanisms such as choosing low-entropy passwords and reusing them across different sites. Password reuse severely threatens password security. A study [20] shows that a typical user averages types eight passwords each day, and maintains 25 passwords, but actively uses only about seven distinct passwords [21]. As a result, hackers who crack the password of a low-security site may access a high-security site.

To encourage users to choose high-entropy passwords, various password rules (e.g. requirements of minimum length, symbols, numbers, and upper case letters) are adopted by the service provider. However, although these rules effectively increase security [22,23], passwords are still under threat [24]. On the other hand, there are no effective measures taken by service providers to avoid password reuse.

In addition to low-entropy and reuse of passwords, phishing attack and shoulder surfing attack continue threatening password security. Shoulder surfing attack is a direct observation technique to obtain information. Anyone who monitors the password input procedure by looking over the users’ shoulder may memorize the password. In phishing attack, users are lured to a website or an app that looks like the legitimate one to type in their passwords. Both attacks are tough for users to deal with.

2.2. Password managers

Password managers are designed to alleviate the memory burden of users while ensuring security. Users only need to remember one master password, and password managers handle the pairs of user name and site password. More specifically, password managers can transparently act as a middle-man between the users and the service provider. Password managers can be classified into two main categories: the retrieval password manager and the generative password manager.

Retrieval password managers simply store site passwords in a “wallet”, i.e., a password storage medium, instead of human mind. Later, the site passwords can be retrieved and used to log into accounts. Retrieval password managers are components of most modern browsers, including Internet Explorer, Google Chrome, Safari, Firefox, Opera, and so on. Retrieval password managers also have two categories based on whether the site passwords are stored in plain text or encrypted under a master password. Unencrypted password wallets are obviously insecure. However, the master password of an encrypted password wallet is still vulnerable to offline attack. In offline attack, for a password $pwd_0$, which is hashed to $pwd_0$, if an adversary obtains $pwd_1$, he can repeatedly guess $pwd_0$. Every guessed password is hashed and compared with $pwd_1$, until the right one is found. Brute-force attack and dictionary attack are two typical offline attacks. In brute-force attack, the adversary tries every possible password until he succeeds. In dictionary attack, the adversary only tries likely possibilities, such as words in a dictionary, or combinations of words in multiple dictionaries.

Generative password managers allow users to generate multiple site passwords from one master password and facilitate better site password quality and a reduction in password reuse by leveraging hashing, key stretching or salting on the client-side. PwdHash [1], Password Multiplier [3], Client-CASH [5], Versipass [2], and Passpet [4] all belong to the generative password managers. In PwdHash, the user inputs the master password to a domain salted one-way function, which generates site passwords of high entropy. Password Multiplier leverages key stretching, i.e., iterated hash [25], instead of the hash function. Client-CASH exploits a client side cost asymmetric key stretching to reduce the delay of authentication with correct master password. Versipass uses cued graphical password to address the memorability issues. Passpet improves both convenience and security of website logins through a combination of the Password Multiplier and the Petname Tool. Unlike password managers mentioned above, commercial password managers like 1Password [26] and LastPass [27] rely on a trust server to derive site passwords from the master password.
However, PwdHash, Password Multiplier, Client-CASH, Versipass, and Passpet are all vulnerable to the single point of failure introduced by the master password, through which the adversary can access all the accounts of users. The master password is vulnerable to offline attack. This dangerous possibility was highlighted by the recent crack of LastPass [28]. Moreover, many password managers offer no protection against shoulder surfing attack.

2.3. Physical Uncloneable Function

Physical Uncloneable Function (PUF) is a hardware security primitive. PUF is a physical structure that can map input challenges to output responses, called Challenge-Response Pairs (CRPs). The number of bits in each challenge is called bitlength. The CRPs of a PUF are determined by the random physical disorder that occurs during manufacturing ICs. Even with the same design, different PUFs will have different CRPs, which are hard to be predicted before manufacturing, hard to be controlled during manufacturing, and hard to be cloned after manufacturing.

Uniformity and uniqueness are two intrinsic properties of PUF. Uniqueness refers to the capability of producing different responses for different PUFs with the same design under the same challenge. Uniformity refers to the capability of producing response bits “0”s and “1”s with similar proportion.

According to the number of CRPs, PUF can be classified into two main categories: weak PUF and strong PUF. Owing to their different characteristics, they have different application areas. The weak PUF has a few CRPs. To generate more response bits, weak PUF normally needs more hardware cost. The SRAM PUF [10, 29] is a typical weak PUF, which uses the logic value stored in each SRAM cell when powered up as the response. Besides SRAM, other storage cells such as Flash [30], DRAM [31], and Memristor [32] can also be used to construct weak PUFs. Different from weak PUFs, the strong PUF normally has a large number of CRPs which is exponential to the bitlength of PUF. Arbitrarily PUF is a typical strong PUF. It works by comparing the delays of two symmetrical paths [33]. The path segments are chosen by the challenge. The arbitrator PUF has many variants, such as XOR arbitrator PUF [34], lightweight arbitrator PUF [35], feed forward arbitrator PUF [33], and slender arbitrator PUF [36].

In practice, due to noises and other physical uncertainties such as supply voltage and temperature, PUF responses are hard to maintain completely stable. Fortunately, many literatures such as [37–39] have researched a lot on how to design stable PUFs. Error Correction Code (ECC) can also be leveraged to ensure the reliability of PUF.

3. PUFPass mechanism

In PUFPass, password management is handled across both software and hardware. The key idea is leveraging the uniqueness and the uniformity of PUF to protect the whole mechanism in hardware level. This section firstly shows the data flow of password in PUFPass, and then describes implementation details by an instantiation.

3.1. Data flow of password

The data flow of PUFPass is shown in Fig. 1. PUFPass is deployed through hardware and software. PUF is implemented in the hardware. The pretreatment module, the strengthening module, and posttreatment module are deployed in the operating system. Receiving user password from the app, PUFPass calls the PUF to strengthen it and generate strong site password. Then, the strengthened site password is sent to the server to perform authentication.

1. Input phase. The user inputs the user name and user password $pwd_u$. This operation is different from normal applications. However, $pwd_u$ will be strengthened by the PUF to generate the strong site password. Therefore, the user can choose an easy-to-remember password without worrying about offline attack and password reuse problem. For each user name, PUFPass will also generate a random label $lab$ which will be stored locally in the device. The app then sends the triple $(lab, pwd_u, dom)$ to the API in operating system. The $dom$ refers to the domain name of the app.

2. Pretreatment Phase. The API constructs a PUF challenge based on the received triple. If a PUF is stimulated by the same challenges, the responses will be same too. To ensure the uniqueness of site passwords, the challenges generated for each site should also be unique. Since the hardware design of PUF has fixed bitlength, the triple is pre-processed in three steps to construct a valid challenge. In the first step, $lab, pwd_u$ and $dom$ are concatenated into the raw string $Sr$, which contains $lf$ characters. In the second step, this string is formatted into one or more challenge strings $Sc$. The formatting process is as follows. Supposing the bitlength of the PUF is $n$ and one character is converted to $n_b$ binary bits, then each challenge string contains $lf = \frac{n}{n_b}$ characters, and the string $Sc$ is split into $n_c$ challenge strings, where $n_c = \left\lceil \frac{n}{lf} \right\rceil$. If the character length of the last challenge string is less than $lf$, it will be padded by duplicating itself. For example, if $Sr$ is “j9kuserpasswordfacebookcom”, and $lf = 16$, then $Sc$ will be split into two challenge strings: “j9kuserpassword” and “facebookcom”. Since the length of the second challenge string “facebookcom” is less than 16, it will be padded and become “facebookcomfacebookcom”. In the last step, according to the ASCII table, each character in the challenge strings is converted to bits which construct valid challenges. The reason of the padding mechanism by duplicating the challenge string is to ensure the diversity of challenges and make the most of triple. For example, considering two strings “abcd” and “efgh”, if we pad them using the same character “i”, then the challenge strings are “abcdiiiiiiii” and “efghihtiiiiii”, in which many characters are the same. However, if we pad them by duplicating themselves, the challenge strings are “abcdabcdecd BCE” and “efghgefggegh”, which are totally different. The PUFPass also contains a label in the triple in order to generate different challenges and site passwords for different accounts of the same app.

3. Strengthening Phase. The strengthening module requires access to the PUF and stimulates it with the challenges. To generate a unique password for each site, the strong PUF which has many CRPs is
3.2. Instantiation in CPU

In this subsection, an illustration of PUFPass in Xilinx Zynq-7000 CPU + FPGA SoC is shown in Fig. 2. PUFPass is implemented in FPGA, and Android is implemented in CPU. Please understand that PUFPass is not limited to this software/hardware platform.

3.2.1. PUF

The arbiter PUF implemented in FPGA is shown in Fig. 3, which has been proved usable in many previous works [40]. Certainly, in PUFPass, other PUFs rather than the implemented one can also be deployed. In this design, when a challenge is prepared, a transition is propagated through two paths to the flip-flop. If the challenge bit is 0, the transition is propagated through $t_{in} \rightarrow LUT_0 \rightarrow LUT_2 \rightarrow t_{out}$ in the corresponding ChallengeStage module; if the challenge bit is 1, the transition goes through $t_{in} \rightarrow LUT_1 \rightarrow LUT_2 \rightarrow t_{out}$. If the transition arrives at $D$ port of flip-flop earlier than at $Clock$ port, the response bit $r$ is 1, else $r$ is 0.

The AdjustStage modules are used to balance the delays of two paths. In ideal situation, the two paths should have the same nominal delays so that the process variations can determine the CRPs. The uniformity of response bits, i.e., the percentage of response bit 1s of different challenges, is expected to be 50%. However, due to the limitation of placement and routing in FPGA and the setup/hold time of flip-flop, the two paths may be imbalanced. If most response bits of different challenges are 1s, certain bits of $a_{0,1}$-$a_{0,2}$ are set to 1. By doing so, extra delays are added to the path arriving at the $D$ port, so the number of response bit 0s is increased. The values of $a_{0,1}$-$a_{0,2}$ and $a_{1,1}$-$a_{1,2}$ are fixed when using the arbiter PUF.

The arbiter PUF alone is proved vulnerable to modeling attacks [41] if adversaries can directly access its CRPs. In our prior work, we proposed a voter based PUF [42]. It employs multiple arbiter PUFs. Each arbiter PUF is equipped with an on-line reliability checker. A weighted voter determines the final response bit based on the response bit and its reliability level of each arbiter PUF. The voter based PUF does not only have high modeling attack resistance, but also achieves high reliability. The details can be found in Ref. [42]. In practice, the PUF is not required to be implemented in FPGA, which means more reliable PUF designs can be adopted, such as [38]. Furthermore, error correcting methods can be leveraged to solve the reliability issue. Since the reliability enhancement is not the focus of this paper, it is not described in detail.

3.2.2. Deployment of PUFPass in android platform

Android is a software stack for mobile devices, which includes an operating system, middleware and key applications. The platform can be divided into five layers, from bottom to top: Linux kernel, hardware abstraction layer (HAL), Android runtime (ART) and native C/C++ libraries, application framework, and applications.

Linux kernel. The kernel is the foundation of the Android platform. Linux kernel allows Android to provide core system services such as security, memory and driver model. To access PUF, a hardware driver is required in the Linux kernel.

Hardware abstraction layer (HAL). The HAL provides standard interfaces that expose hardware functionalities to the application framework. The HAL implementations are packaged into modules and loaded.
by the Android system when a framework API makes a call to access device hardware. In HAL, PUF interface module is required to access the driver for PUF.

Android runtime and Native C/C++ libraries. Every application owns process and an instance of the Android Runtime (ART). The ART allows a device to run multiple virtual machines efficiently. Android also includes a set of native libraries written in C/C++. Many core system components and services, such as ART and HAL are built using these libraries. The native libraries can be called by developers through the application framework APIs. Each call for hardware module in the HAL is executed by the Java Native Interface (JNI) in ART. Therefore, the specific JNI for PUF interface module is required.

Application framework. The application framework provides developers with the entire feature set of the Android OS through APIs written in Java language. Developers can create Android applications using these APIs, which includes view system, activity manager, notification manager, and so on. Class EditText provides the function getText() to capture the user password typed into the input field.

Applications. Android Application layer contains applications that come with the operation system and applications installed by users. All applications are written using Java programming language. It is the layer that mobile applications will fit into. Application developers can define their own functions to capture and further operate the passwords typed by users. For most applications, the typed passwords are encrypted using MD5 or RSA before sending to the authentication server.

Whether using PUFPass to strengthen the user password can be mandatory or determined by the user. According to [43], users are uncomfortable with “relinquishing control” of their passwords, so it is suggested to allow users to determine whether using PUFPass. The user interface is illustrated in Fig. 4. To use PUFPass, users only need to check the PUFPass button in the application interface.

In summary, to deploy PUFPass in Android OS, the developer needs to write the hardware driver in the Linux kernel for the PUF, and then write the PUF interface model in the HAL to access the PUF driver. In order to call the hardware interface model which is written in C/C++, a specific JNI need to be written because the code in the upper layer of Android is written in Java. Finally, the developers can develop applications using the APIs in the Application framework just as developing other applications.

From Fig. 2, it can be seen that the whole system ranges from hardware to software. The API is located in Android. PUF is implemented in FPGA. Android and FPGA interact to perform the task of password management.

The working flow is as follows.

1. After the user inputs user name and $pwdu$ in the app, the triple of (label, $pwd_u$, dom) is sent to the API for PUFPass in application framework.

2. The pretreatment module in application framework receives the triple and generates challenges from them.

3. The challenges are received by the strengthening module and sent to the PUF interface module in HAL through JNI of Android runtime. The PUF interface then calls PUF in FPGA through AXI interface. After stimulated by challenges from the CPU, the PUF generates response bits and sends them back to the CPU.

4. The response bits are sent to the posttreatment module in the application framework to be processed into the form required by the authentication server.

5. The strengthened site password $pwd_s$ is sent to the application for authentication.

4. Security evaluation

The security is evaluated by analyzing whether the adversary can get access to a target account. All the analysis assumes that the adversary knows the user name and the working mechanism of PUFPass. There are mainly four vulnerable points in PUFPass: the user password, the device, the site password, and the server.

All the experiments are performed in Ubuntu 14.04 operating system on a desktop with an Intel i7 processor employing four 3.6GHz cores, a GPU of NVIDIA GTX 1080, and 32GB memory. We have four Zynq-7000 SoCs. Instead of finding some users to show us the user passwords for experiments, we randomly choose user passwords from a dictionary, and use the PUF to generate the site passwords from these user passwords.

4.1. User password

First of all, the scenario of single point of failure is analyzed. The vulnerable point, user password, is considered in this subsection. The user password may be obtained by the adversary through shoulder surfing attack or phishing attacks. However, without right device, the adversary knows neither the label nor the CRPs, so he cannot generate right challenges and right site password. This cannot be achieved by previous password management mechanisms, where the user password can be used in different devices.

4.2. Device

This subsection analyzes the single vulnerable point: device. Like previous password management mechanisms, if the adversary gets the device, he may perform online attack to guess the user password. In this scenario, the adversary will be limited by the site’s velocity control mechanism which imposes restrictions on the number of login attempts per unit time and the account will even be disabled if too many unsuccessful logins are tried. What’s more, the online attack faces a high risk of being detected.

4.3. Site password

If the adversary obtains the site password, like previous password management mechanisms, this site is not secure any more. Fortunately, since the site passwords are generated by the PUF, different site passwords are used for different sites. Thus, the adversary cannot infer site passwords of other sites from the obtained site password. In other words, the password reuse problem can be handled.

4.4. Server

The authentication server often stores the hash values of site passwords. If the adversary cracks the server and gains the hash values, he
can conduct offline attack on the site password with sufficiently long attack sessions. This paper uses Hashcat [44] to perform the offline attack on site passwords. Hashcat is a popular, freely available password-cracking software that supports advanced cracking techniques and performs probabilistically-ordered (intelligent per-position) brute-force attacks. In our experiments, hashcat runs in its "straight attack" mode and tries out all the 65117 randomly generated attack rules in the default generated rule ruleset on publicly available password dictionaries. Such rule based attack uses a programming language designed for password candidate generation. During the rule based attack, dictionary words are modified, cut, or extended to improve the flexibility and accuracy. Detailed explanations of the possible rules can be found in Ref. [42]. The password dictionary is downloaded from CrackStation [45]. This dictionary is sorted in alphabetical order, and the duplicates are removed, resulting in 1.2 billion unique words from leaked password database, Wikipedia database, books, and internet.

In the experiment, the site passwords of various lengths (8, 10, 12, 14, 16) are evaluated. The number of site passwords in each length is 100000, totally 500000. The site password cracking results are presented in Table 1. The type of site passwords is “mix”, which means every site password contains at least two categories of characters among letters, numbers, and symbols. None of the site passwords are cracked with high reliability and use ECC to ensure the reliability, so attackers would not obtain unreliable CRPs for attack. Moreover, as previously mentioned, “lab” in the triple (lab, pwpda, dom) generates the challenge, so we can use the “lab” to ensure that a reliable site password is generated. Here, the “lab” is like a kind of “helper data”. In the experiments, the “lab” is known to attackers, and we will proved the security later. Reliability based attack [47] leverages reliable and unreliable CRPs to regain the model of PUF. In terms of PUFPass, reliability based attack still does not work. Firstly, even in the reliability based attack, attackers still need to access the PUF so to know whether a CRP is reliable or not, though attackers do not need to know the exact response of a challenge. Since in PUFPass, attackers cannot directly access the PUF, they still cannot obtain unreliable and reliable CRPs for attack. Secondly, to generate a reliable site password, PUFPass should use the PUF with high reliability and use ECC to ensure the reliability, so attackers would not obtain unreliable CRPs for attack. Moreover, as previously mentioned, “lab” in the triple (lab, pwpda, dom) can be utilized to ensure the reliability of the PUF.

4.5. Device and site password

This subsection considers the scenario that two vulnerable points are surrendered: the device and the site passwords. Since the adversary obtains the device, in this attack scenario, it is assumed that the adversary knows the circuit of PUF, the domain names, and the labels. If the adversary can crack the user password from obtained site passwords through offline attack, then he can use the user password in the device to access all the other sites.

The security of PUF may be threatened by traditional attack, helper data based attack, and reliability based attack. Traditional attack method [41] towards PUF is the machine learning based modeling attack, which is a kind of offline attack. The CRPs are modeled as \( R = F(C, P) \). The \( R \) represents a response. The \( C \) represents a challenge. The \( P \) represents the electronic parameters determined by process variations. The \( F() \) describes the circuit structure of the PUF. The adversary has a collection of some CRPs as the training set, and does not know the values of \( P \). Using machine learning techniques, \( P \) is calculated by the adversary to predict all the other CRPs. Fortunately, as mentioned before, we have proposed a voter based PUF [42] which can well resist such attacks. In Ref. [46], the adversary manipulates public helper data to attack the PUF. However, in Ref. [42], it has been proved that the reliability of a CRP can be checked online. That means, a challenge is input to the arbiter PUF (or the mutants of arbiter PUFs), using the design in Ref. [42], the PUF can know whether it can generate a reliable response or not. In PUFPass, the triple (lab, pwpda, dom) generates the challenge, so we can use the “lab” to ensure that a reliable site password is generated. Here, the “lab” is like a kind of “helper data”. In the experiments, the “lab” is known to attackers, and we will proved the security later. Reliability based attack [47] leverages reliable and unreliable CRPs to regain the model of PUF. In terms of PUFPass, reliability based attack still does not work. Firstly, even in the reliability based attack, attackers still need to access the PUF so to know whether a CRP is reliable or not, though attackers do not need to know the exact response of a challenge. Since in PUFPass, attackers cannot directly access the PUF, they still cannot obtain unreliable and reliable CRPs for attack. Secondly, to generate a reliable site password, PUFPass should use the PUF with high reliability and use ECC to ensure the reliability, so attackers would not obtain unreliable CRPs for attack. Moreover, as previously mentioned, “lab” in the triple (lab, pwpda, dom) can be utilized to ensure the reliability of the PUF.

Attacking PUFPass is different. In PUFPass, the challenge is determined by the label, the domain, and the user password. Hence the model \( R = F(C, P) \) becomes \( R = G(lab, dom, pwpda, P) \). Here, \( G() \) describes both the circuit structure of the PUF and how the challenge is generated. In such attack, both pwpda and \( P \) are unknown to the adversary. In the experiment, the Compound Heuristic Algorithm [48,49] is adopted to guess the user password.

The experimental results are shown in Table 2. The site passwords obtained by the adversary range from 1000 to 10000, which is generated from one user password using one PUF. Both site password and user password contain 6 characters. The type of passwords is “num”, which means the characters are all numbers. The match rate refers to the percentage of equal characters between the real user password and the guessed user password. From the experimental results, we can find that, even if the adversary obtains 10000 site passwords, the average match rate of user password is only 23.3%. In practice, a user averagely owns 25 accounts [20]. Therefore, the PUFPass can resist the machine learning based offline attack on the user password.

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline attack on site passwords.</td>
</tr>
<tr>
<td>Length</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline attack on user passwords.</td>
</tr>
<tr>
<td>Site Password Obtained by the Adversary</td>
</tr>
<tr>
<td>Length</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>3000</td>
</tr>
<tr>
<td>5000</td>
</tr>
<tr>
<td>10000</td>
</tr>
</tbody>
</table>
5. Usability analysis

This section analyzes the usability of PUFPass from those aspects mentioned in Section 2: ease of use, memory burden, transportability, and password modification.

5.1. Ease of use

PUFPass is embedded in the application and the operating system. As mentioned in Section 3.2.2, whether to use PUFPass can be mandatory or determined by the user. If PUFPass is mandatory, it is embedded into the application during development and the use of the application is different from that of traditional applications. If the decision is made by users, the users only need to perform one more operation—selecting or not selecting the PUFPass check box, as shown in Fig. 4. Other operations, including inputting user name and user password are the same as traditional ones. Therefore, users need negligible efforts on learning how to use PUFPass.

Throughput is also an important metric for evaluating an APP. The implemented PUFPass can work on 10.5 GHz, which means, in 1 s, $10.5 \times 10^9$ response bits can be generated. In PUFPass, the PUFPass is used to strengthen the site passwords. If the site password needs to be sent to the server through network for authentication, the communication time is the main bottleneck in the whole password authentication procedure. On the other hand, normally, it is not likely for users to repeatedly input passwords in high frequency.

5.2. Memory burden

As shown in Section 4, in PUFPass, the user password can resist shoulder surfing attack, phishing attack, and machine learning based offline attack. Even if a user only memorizes and uses one user password, the PUFPass can still help the user to generate site passwords which can handle the password reuse problem and resist the offline attack. Therefore, PUFPass effectively reduces the memory burden of users without sacrificing security.

5.3. Transportability

The transportability indicates the capability of accessing the site passwords from different devices, such as from desktops or from mobile phones. However, with the wide use of mobile phones, the demand of transportability is declining, and the login mechanism developed by the service provider is becoming more and more convenient.

Firstly, high-security applications such as online payment and online banking can be installed on the mobile phone. Moreover, some applications, such as online banking provided by ICBC, from China, require users to use different passwords for different types of devices, i.e. different passwords are input in the desktop and the mobile phone for one user. Thanks to the convenience brought by the mobile phone, people are more and more inclined to use the mobile phone. As a result, the sales volume of desktops decreases dramatically these years.

Secondly, a machine readable label, the Quick Response (QR) code, has already been adopted to achieve transportability. For example, WeChat is a popular app in China with one billion active users. If a user wants to log in the WeChat on a PC, the service provider of wechat generates a QR code on the PC, which contains a Universal Unique Identifier (UUID). Then the user uses the phone, where he already logs in the wechat, to scan the QR code. After that, on the mobile phone, a token, which is generated from the user name and password, is bound with the UUID, and they are sent to the service provider. The service provider authenticates the received data. If it passes the authentication, the service provider sends the token to the wechat on the PC. In this way, once users log in the wechat on mobile phones by inputting user names and passwords, they can log in the wechat on the PC through the mobile phones without inputting user names and passwords any-

5.4. Password modification

Users may modify their passwords in the following scenarios.

1. The user loses his device or the device is old, which does not contain PUFPass, so the user uses a new device with PUFPass.

In this case, since the site password generated by the PUFPass is different from what the user inputs, the user has to update the password. This process is also called password migration. All the-generation password managers need password migration because the generated site passwords using different managers are different. PUFPass is no exception. Password migration requires the user to input the user name, the old user password, which is the old site password, and the new user password. The new user password and the old user password are allowed to be equal, so that the user does not need to remember a new user password. Finally, the old site password and the new site password generated by PUFPass are sent to the server to perform password update.

The number of accounts that need strong protection is small. With the help of Wechat, a popular social application of China, We do an online survey on 100 contacts from all walks of life to get the number of accounts that need the protection of strong passwords. The survey is designed as A. The average number is only 5.6. From this result we can see that it is unnecessary to protect all the passwords through PUFPass, so it is suggested that app developers let users determine whether to use PUFPass or not. Hence, password migration will not expend too much effort. In addition, even not using PUFPass, users are still suggested to modify their passwords regularly for ensuring the security.

2. The user forgets the password or he wants to replace his old device (containing PUFPass) with a new one (also containing PUFPass).

In this case, the user cannot update the password because he does not own the correct password for the new device. Hence, the user needs to reset his password using existing mechanisms, such as resetting password through answering preset questions or via SMS message. For example, the service provider sends a SMS message containing a one-time code to the phone number, which was bound by the user when registering, and the user sends the received code to the service provider to authenticate himself for resetting the password. The existing mechanisms for resetting password are not required to be modified. Similar to password update, this will not expend too much effort either.

3. The site password is leaked due to the service provider being cracked.

In this case, the user has to reset the site password to ensure the security of the account. PUFPass provides two means to change the site password. In the first method, the user changes the current user password $pwd_u$ to the new user password $pwd_u'$. PUFPass reads the current label lab for the account from local storage and generates the new

5.3. Transportability

The transportability indicates the capability of accessing the site passwords from different devices, such as from desktops or from mobile phones. However, with the wide use of mobile phones, the demand of transportability is declining, and the login mechanism developed by the service provider is becoming more and more convenient.

Firstly, high-security applications such as online payment and online banking can be installed on the mobile phone. Moreover, some applications, such as online banking provided by ICBC from China, require users to use different passwords for different types of devices, i.e. different passwords are input in the desktop and the mobile phone for one user. Thanks to the convenience brought by the mobile phone, people are more and more inclined to use the mobile phone. As a result, the sales volume of desktops decreases dramatically these years.

Secondly, a machine readable label, the Quick Response (QR) code, has already been adopted to achieve transportability. For example, WeChat is a popular app in China with one billion active users. If a user wants to log in the WeChat on a PC, the service provider of wechat generates a QR code on the PC, which contains a Universal Unique Identifier (UUID). Then the user uses the phone, where he already logs in the wechat, to scan the QR code. After that, on the mobile phone, a token, which is generated from the user name and password, is bound with the UUID, and they are sent to the service provider. The service provider authenticates the received data. If it passes the authentication, the service provider sends the token to the wechat on the PC. In this way, once users log in the wechat on mobile phones by inputting user names and passwords, they can log in the wechat on the PC through the mobile phones without inputting user names and passwords any-

5.4. Password modification

Users may modify their passwords in the following scenarios.

1. The user loses his device or the device is old, which does not contain PUFPass, so the user uses a new device with PUFPass.

In this case, since the site password generated by the PUFPass is different from what the user inputs, the user has to update the password. This process is also called password migration. All the generation password managers need password migration because the generated site passwords using different managers are different. PUFPass is no exception. Password migration requires the user to input the user name, the old user password, which is the old site password, and the new user password. The new user password and the old user password are allowed to be equal, so that the user does not need to remember a new user password. Finally, the old site password and the new site password generated by PUFPass are sent to the server to perform password update.

The number of accounts that need strong protection is small. With the help of Wechat, a popular social application of China, We do an on-line survey on 100 contacts from all walks of life to get the number of accounts that need the protection of strong passwords. The survey is designed as A. The average number is only 5.6. From this result we can see that it is unnecessary to protect all the passwords through PUFPass, so it is suggested that app developers let users determine whether to use PUFPass or not. Hence, password migration will not expend too much effort. In addition, even not using PUFPass, users are still suggested to modify their passwords regularly for ensuring the security.

2. The user forgets the password or he wants to replace his old device (containing PUFPass) with a new one (also containing PUFPass).

In this case, the user cannot update the password because he does not own the correct password for the new device. Hence, the user needs to reset his password using existing mechanisms, such as resetting password through answering preset questions or via SMS message. For example, the service provider sends a SMS message containing a one-time code to the phone number, which was bound by the user when registering, and the user sends the received code to the service provider to authenticate himself for resetting the password. The existing mechanisms for resetting password are not required to be modified. Similar to password update, this will not expend too much effort either.

3. The site password is leaked due to the service provider being cracked.

In this case, the user has to reset the site password to ensure the security of the account. PUFPass provides two means to change the site password. In the first method, the user changes the current user password $pwd_u$ to the new user password $pwd_u'$. PUFPass reads the current label lab for the account from local storage and generates the new
site password $pwd'_s$ from the new triple $(lab, pwd'_s, dom)$. In the second method, instead of asking the user to change and remember a new user password, the app automatically generates a new label $lab'$ for the account. The new triple $(lab', pwd'_s, dom)$ generates a new site password $pwd'_s$ by the PUFPass. Therefore, in this case, the steps of resetting site password in PUFPass are no more complex than traditional ones without PUFPass.

6. Comparison with other password management mechanisms

This section compares PUFPass with other password management mechanisms previously mentioned in Section 2. “Legacy passwords” is considered as the baseline mechanism. For the browser manager, it is assumed that the autofilling feature is turned on, and the password “wallet” is protected under the master password. The comparison is assumed that the autofilling feature is turned on, and the password considered as the baseline mechanism. For the browser manager, it is mechanisms previously mentioned in Section 2. “Legacy passwords” is

1 Memorywise-Effortless & Scalable-for-Users. Except the mechanism of legacy passwords, all the other mechanisms are almost Memorywise-Effortless and Scalable-for-Users because users only need to remember one user password or one master password no matter how many site passwords there are. Comparatively, the mechanism of legacy passwords requires users to remember the password for each site.

2 Nothing-to-Carry. This benefit means that users do not need to carry an additional object to use the scheme. The mechanism of legacy passwords offers this benefit because all the passwords are stored in the memory of users. For browser managers, the site passwords may be stored in the local device or in the cloud. This paper only considers the former situation. For the site password stored in the cloud, it is insecure for users to use the cloud sync service in public computers, so it is better for the user to carry his own device. For other schemes, the user can get access to all his site passwords as long as he remembers his master password. However, since the master password is bound to all the site passwords, it is insecure to use the master password in public computers. Hence, it is better for the user to carry his own device as well. The site password of PUFPass is dependent on the device. Fortunately, mobile phones have been widely used nowadays, which are easy to carry. Meanwhile, using one device to authenticate other devices such as through QR code has been applied in many popular apps, as explained in Section 5.

3 Physically-Effortless. A mechanism which is Physically-Effortless does not need the user to take physical effort (cognitive). Even illiterate people can use it. All the mechanisms listed in the table need the user to type the password, so they are not Physically-Effortless.

4 Easy-to-Learn. If a mechanism is Easy-to-learn, even a user who does not know the mechanism can learn and recall how to use it without too much trouble. The mechanism of legacy passwords is the easiest. It only requires the user to input the password on the app. The browser manager needs the user to input the password and choose to remember it so that later he can use the remembered password without typing it again. PwdHash requires the user to type a prefix “#” before the master password. PUFPass may provide the option for users to determine whether using PUFPass or not. For other mechanisms, they often require the registration of a new account for the password manager so need more complex operations.

5 Efficient-to-Use. If a mechanism is Efficient-to-Use, the time the user must spend for authentication should be acceptable. The mechanism of legacy passwords only needs the user to directly type the site password. Browser managers, LastPass, and 1Password retrieve the site passwords from the local storage or cloud. PwdHash and Versipass use hashing to generate the site passwords. Password Multiplier and Passpet use key-stretching. The Client-CASH uses the asymmetric key-stretching. The PUFPass uses the hardware–PUF to generate the site passwords.

6 Infrequent-Errors. When a legitimate and honest user wants to log in the account, he should usually succeed. This is achieved by all the mechanisms.

7 Easy-Recovery-from-Loss. If a scheme is Easy-Recovery-from-Loss, a user can still conveniently authenticate in case the device is lost or the password is forgotten. In the scheme of legacy passwords, the site passwords are stored in the user’s memory. If one site password is forgotten, it can be easily reset. Hence it is Easy-Recovery-from-Loss. For other mechanisms, if the user/master password is forgotten, all the site passwords need to be reset, unless cloud or other services are provided to recover the user/master password. Hence they are not Easy-Recovery-from-Loss.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Comparison between PUFPass and other Mechanisms.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usability</strong></td>
<td><strong>Deployability</strong></td>
</tr>
<tr>
<td>Legacy Passwords</td>
<td>![rating]</td>
</tr>
<tr>
<td>Browser Managers</td>
<td>![rating]</td>
</tr>
<tr>
<td>PwdHash</td>
<td>![rating]</td>
</tr>
<tr>
<td>Password Multiplier</td>
<td>![rating]</td>
</tr>
<tr>
<td>Client-CASH</td>
<td>![rating]</td>
</tr>
<tr>
<td>Versipass</td>
<td>![rating]</td>
</tr>
<tr>
<td>Passpet</td>
<td>![rating]</td>
</tr>
<tr>
<td>LastPass</td>
<td>![rating]</td>
</tr>
<tr>
<td>1Password</td>
<td>![rating]</td>
</tr>
<tr>
<td>PUFPass</td>
<td>![rating]</td>
</tr>
</tbody>
</table>

* = offers the benefit; ** = almost offers the benefit; no circle = does not offer the benefit.

Q. Guo et al. Integration, the VLSI Journal 64 (2019) 173–183
8 **Accessible.** As long as users are physically able to input the passwords, the users should not be prevented due to disabilities or other physical (not cognitive) conditions. All the mechanisms are accessible.

9 **Negligible-Cost-per-User.** Due to the simplicity, the mechanism of legacy passwords is Negligible-Cost-per-User. For mechanisms other than PUFPass, users may need to install a plug-in or an app for password management. The cost is still negligible. For PUFPass, it needs the support of hardware. The hardware cost of the implemented PUF is 3235 LUTs and 65 FFs, which occupy 6% of FPGA LUT resources and 0.06% of FPGA FF resources, respectively. The implemented PUF can work on 10.5 GHz. The dynamic power is less than 1 mW. It should be noticed that, the PUF is only used when site passwords are required. Since it is not likely for users to frequently input passwords, the power is acceptable.

10 **Server-Compatible.** All the mechanisms produce text passwords and do not need the authentication server to make any changes.

11 **Browser-Compatible.** Browser managers are built in browsers. Other mechanisms all require specific browser extensions.

12 **Mature.** The mechanism of legacy passwords and browser managers have been implemented and widely deployed. The users of the commercial password managers, LastPass and 1Password, are still minority. Others are still in research phase.

13 **Non-Proprietary.** People do not have to pay royalties to use these mechanisms.

14 **Resistant-to-Physical-Observation.** If a scheme is Resistant-to-Physical-Observation, an adversary cannot impersonate a user after the adversary observes the authentication procedure of the user. The shoulder surfing attack mentioned before is an observation attack. In the mechanism of legacy passwords, site passwords can be recorded by human memory or video when they are typed, so it is not Resistant-to-Physical-Observation. Similarly, for other schemes, the user/master passwords can also be recorded by an attacker. However, for the browser managers which store the site passwords in local device and the PUFPass which is dependent on the hardware of PUF, without the user device, the attacker can still not obtain the site passwords. Hence they are Resistant-to-Physical-Observation. For other mechanisms, the site passwords can be generated from the observed master password.

15 **Resilient-to-Targeted-Impersonation.** If a mechanism is Resistant-to-Targeted-Impersonation, it should not be possible for an acquaintance (or investigator) to impersonate a specific user by exploiting knowledge of personal details. The legacy passwords, the site passwords in browser managers, and the user/master passwords of other mechanisms are determined by users themselves, so these passwords may contain personal information, such as birthday. The difference between the PUFPass and other mechanisms is that, even if the adversary successfully guesses the user password through the personal information, without the right device, he can still not access the user account.

16 **Resistant-to-Throttled-Guessing.** Due to the users’ deep rooted bad habit of password selection, the site passwords in the mechanisms of legacy passwords and browser managers, and the master passwords in the mechanisms of PwdHash, Password Multiplier, Client-CASH, Versipass, Passpet, LastPass, and 1Password may be easily guessed by the adversaries. Only for PUFPass, the adversary has to own the correct device for guessing the user password, so it is resistant-to-throttled-guessing.

17 **Resistant-to-Unthrottled-Guessing.** For a scheme which is Resistant-to-Unthrottled-Guessing, even if an attacker can repeatedly guess the password, the attacker can still not get access to an account. In the mechanism of Legacy passwords and browser managers, the site passwords are selected by users. Due to users’ deep rooted bad habit of password selection, they are not considered Resistant-to-Unthrottled-Guessing in Ref. [18]. The site passwords of other schemes are harder to be directly guessed because they are generated from the user/master password through hashing, key stretching, or manufacturing variation (PUFPass). However, in terms of the user/master passwords, not all the schemes are Resistant-to-Unthrottled-Guessing. The master password of PwdHash, Password multiplier, and Passpet may be guessed by offline attack [18]. Using a cost asymmetric key-stretching function, Client-CASH reduces the adversary’s success rate on master password by 21% [5]. LastPass and 1Password are both commercial password managers, the details of which are not public. However, the recent breach of LastPass reminds us of the dangerous possibility that their master password could be cracked offline [28].

18 **Resistant-to-Internal-Observation.** It means an adversary cannot impersonate a user by eavesdropping on the clear-text inside the device or eavesdropping on the clear-text communication between prover and verifier, such as by key logging malware. No matter the site passwords are generated or retrieved, the site passwords will be temporarily stored inside the device. If the adversary can observe them, then all the schemes are not resistant-to-internal-observation.

19 **Resistant-to-Leaks-from-Other-Verifiers.** It means that the leak of site passwords from some sites does not endanger the user accounts at other sites. The mechanism of Legacy passwords and browser managers are not Resilient-to-Leaks-from-Other-Verifiers due to site password reuse. For other schemes, the generated site passwords are different from each other and do not contain user information, so they offer this benefit.

20 **Resistant-to-Phishing.** The mechanism of legacy passwords offers no protection against phishing. For browser managers, the site password is bound with the site. Therefore, phishing attack will not happen if the user relies on site password autofilling. Other schemes are also resistant to phishing attack on the site password because the user does not know the site password at all. However, for the schemes except PUFPass, the site passwords may be recovered by phishing the master password. Due to the existence of the single point of failure introduced by the master password, these schemes are still dangerous. In PUFPass, the generation of site passwords not only needs the user password, but also the right PUF. Therefore, adversary who phishes the user password cannot access the user account in the adversary device.

21 **Resistant-to-Theft.** If a mechanism is Resistant-to-Theft, an adversary who gains possession of the user device cannot use it to access the user account. For the mechanism of legacy passwords, the passwords are stored in human memory. Hence although the device can be theft, the password cannot be theft. For other mechanisms, the user/master password is required to access the account.

22 **No-Trusted-Third-Party.** Except the mechanism of legacy passwords, all the other mechanisms need a third party to provide service such as generating or storing passwords for users.

23 **Requiring-Explicit-Consent.** All the mechanisms are Requiring-Explicit-Consent because the user must type the required password.

24 **Unlinkable.** All the mechanisms are unlinkable, because colluding verifiers cannot determine whether the same user is authenticating to them.

25 **Avoid the single point of failure.** The mechanism of legacy passwords does not have the single point of failure because for different sites, different site passwords are required. The single point of failure does not exist in PUFPass too because both right user password and right PUF are required to generate correct site passwords. Versipass also gets rid of the single point of failure due to the usage of different category passwords. As for the browser managers, which store and encrypt site passwords by the master password, and other generative password managers, including PwdHash, Password Multiplier, Client-CASH, Passpet, LastPass and 1Password, they are all confronted with the threat introduced by the master password. Once the master password is leaked, the adversary can get access to all the user accounts protected by it.
In summary, while maintaining most of the advantages in usability and deployability, PUFPass contributes a lot in improving security in the resistance to physical observation, targeted impersonation, throttled guessing, unthrottled guessing, and the single point of failure.

7. Conclusion

This paper proposes PUFPass, a secure password management mechanism based on software/hardware codesign. PUFPass is resistant to offline attack on the user password and the site password, and avoids anism based to software/hardware codesign. PUFPass is resistant to environmental condition monitoring in homes, IEEE Sensor. J. 13 (10) (2013) 3846–3853.


M. Gao, K. Lai, G. Qu, Hicic: hardware-assisted control-flow integrity checking, Internet Things J.

Appendix A. Survey on the number of account that need the protection of strong passwords

Question: In your mobile phone applications, if your passwords are stolen, how many applications may cause serious security issues, such as threatening your assets or leaking your important privacy information?

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16 or more than 16

Appendix B. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.vlsi.2018.10.003.

References


