A User's Guide for Selecting and Maintaining Password Portfolios

Sreekanth Malladi, Robert Clark
Saint Leo University
33701 State Road 52
Saint Leo, FL 33574-6665
(352)-588-8190
Sreekanth.Malladi@saintleo.edu,
robert.clark04@email.saintleo.edu

ABSTRACT
Passwords have been used for authentication in various scenarios for decades and their use has always been on the increase. They are often preferred over other factors of authentication due to their simplicity, and ease of use. However, users currently are facing problems in selecting and maintaining a portfolio of passwords (on an average 35 usernames & passwords per user). Little guidance exists in literature that helps users in this, except general advises on selecting strong passwords and warnings on not losing them. In particular, no guidance exists on selecting passwords that are of appropriate strengths for different servers based on their purpose, type, and security. This is the problem we address in this paper: We give prudent practices for users in selecting the right passwords such that their effort is minimized in selecting and managing them, while at the same time maintaining their security.

Categories and Subject Descriptors
D.3.3 [**Cyber security**]: Authentication, Access Control.

General Terms
Password strength, Dictionary attacks, Guessing attacks.

Keywords
Passwords, Online/Offline attacks, Salt, Hashing, Encryption.

1. INTRODUCTION
Passwords are a simple yet effective mechanism for authentication that have been around a while. They are used in many situations, including networks, systems and apps. Though many alternative authentication mechanisms were proposed and discussed, the use of passwords has always been on the increase.

However, as easy as they are, passwords are also beset with difficulties both for administrators and users. Until recently, administrators had little guidance in choosing their password storage mechanisms and in setting their password selection policies for users. To remedy this, Florencio et al. have published an excellent guide that enlightens administrators on many previously unknown facts about passwords [1]. For instance, offline guessing attacks happen much more rarely than previously thought and in limited situations. Also, in order to resist an off-line attack, users need to choose passwords that resist 10^{14} guesses, as opposed to just 10^6 guesses that they need, to resist on-line attacks.

But administrator guidance is only one part of the equation. The other part is, guidance to the users. It is also important to guide users on how to select their passwords in order to keep their accounts secure while minimizing their efforts. Unfortunately, currently there is no such guide available, other than those that simply give strategies to choose "strong" passwords (e.g. [2]). However, the need of the hour is not in aiming to create strong passwords for every account, but in selecting them such that strong passwords are chosen only for those accounts where they are needed.

The average computer user struggles with the following problems on password selection:

a) Choosing them so that they are easy to remember, yet hard to be guessed, which were proven to be conflicting goals [3];

b) Maintaining a portfolio of passwords. The average user has to memorize 25 usernames-passwords, and more difficultly, the mapping for each username and its password;

c) Uncertain over the ways to store and manage the large portfolio of passwords;

d) Unsure about where to and where not to enter passwords (e.g. public machines are commonly infected with keystroke loggers);

e) Unsure about how to recover and reset lost, forgotten or stolen passwords.

All users hear are the same general advices: "*need to select a strong password, remember it and not write it down*" regardless of the server's type, importance, and security measures. These advices are often unnecessary. For instance, a user with an account on a web server that stores passwords in plain, needs to simply select a password that withstand 10^6 guesses. No real gain in security is achieved with a stronger password, since on-line attacks are thwarted by the server, and off-line attacks are a concern only if the server were to salt, hash or encrypt the passwords, not if it is storing them in plain.

To remedy this, we are working on a guide that helps users spend the appropriate time and effort in choosing and maintaining large number of passwords. We describe preliminary results and the major contributions of our work here, while saving some of the details and auxiliary sections that are in progress, for the eventual presentation of the paper.

The main contributions of our paper are derived from Florencio et al.'s paper, but from a user's point of view, not an administrator's. We give many takeaway points that will guide users in selecting adequately strong passwords.

Organization. In Section 2, we give a background on passwords, including their history, concepts and terminology. In Section 3, and summarize the main takeaway points from Florencio et al.'s work. In Section 4, we advocate prudent practices on choosing passwords, both for new and existing accounts. We conclude in Section 5 with a description of our work in progress.

2. Background

2.1 History of passwords

Passwords have been introduced initially as part of operating system security (in MULTICS which is the basis of Linux), after scientists have figured out that it is important to safeguard access to systems as well, not just their physical security. Since then, passwords are being ubiquitously used for authenticating users to networks, devices and applications. Being the "something you know" factor, they are often preferred over the "something you have" factor (e.g. smartcards which need to be carried) and the "something you are" factor (e.g. biometrics that need physical presence).

2.2 Types and Ways of Password Use

We will describe the simplest and most general way in which passwords are used, wherein there is a set of users and a server that stores a set of passwords in a file containing records for users with attributes username and password entry. The password entry could be a plain-text password, or hashed password possibly after adding a salt. Lastly, the password file itself could have been encrypted with a symmetric key that is known only to a subset of users called admins.

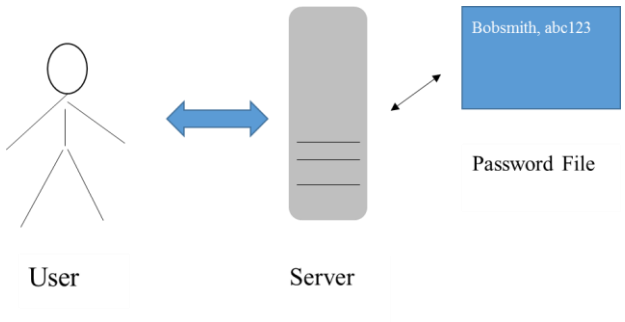

Fig 1. Password storage and checking

When a user sends the <*username*, *password*> combination to the server, it matches it with the corresponding entry in the password file, with or without hashing and salting the password first, depending on the nature of the password file entries. The server also has to first decrypt the password file if it was reversibly encrypted.

2.3 Concepts and Terminology

It is perhaps needless to mention that there have been numerous instances where passwords have been breached through different ways. We would like to describe those ways along with some basic definitions:

Attack points. There are three points at which passwords can be learnt (a) Client machine through malware (b) Server's public-facing side such as the login page for a web server and (c) Server's back-end where the password file is stolen using attacks such as SQL-injection or remote code exploitation.

On-line guessing attacks. Perhaps the simplest way to learn a password is by guessing online, as in entering guesses repeatedly into the public-facing login page of a web server. Though defenses exist such as rate-limiting and locking out after a few unsuccessful attempts, they can be abused as well, to lock out legitimate users. CAPTCHAs are also not always employed or successful.

Off-line guessing attacks. These require that the attacker somehow gains the password file present at the server's back-end. If the server stores passwords in plain, no effort is needed to learn the passwords, so they are not termed off-line guessing attacks. If the server stores only hashes, then guesses also need to be hashed and hashes have to be compared. If the server encrypts the password file with a secret key known only to the admins, then off-line guessing attacks will not work, assuming the key is not leaked.

Dictionary attacks. Both on-line and off-line guessing attacks can be automated by taking guesses from dictionaries suitable to users. In this case, they are termed on-line (off-line) dictionary attacks.

Rainbow tables. If the server stores hashes of passwords, guesses don't have to be hashed: they can be obtained from rainbow tables, which are "processed" dictionaries in that they have guesses hashed already.

Malware attacks. If a client's machine has malware such as keystroke loggers installed, then passwords can be learnt directly without the need for on-line or off-line guessing.

Note that in all the above and throughout the paper, we only consider technical attacks, but not passwords breached through social engineering techniques, including learning passwords by answering questions following "forgot your password?" link.

Password strength. The resistance against being guessed correctly is said to be the strength of a password. There are no theoretical boundaries, but based on practical experience, Florencio et al. have noted that currently, it is enough to select a password that resists 10^6 and 10^{14} guesses to make online and offline guessing attacks useless: it would take more time for the attacker to arrive at the correct guess than he can afford to put the learnt password to any use (4 months).

Salt. A cheap way to improve password strength is by simply adding a random number called "salt" to it before hashing it and storing it in the password file. An attacker who guesses passwords and hashes them to match with the password file also has to guess the salt first, which greatly increases the effort and time for an off-line attack and effectively defeats it.

Reversible encryption. An alternative to salting plus hashing is to encrypt the password file with a key known only to the admins. In this case, passwords are stored in plain, and the only way for an attacker to learn them is by first learning the decryption key. Offline guessing is not needed in that case since passwords are in plain.

Password meters. A relatively new technique to assist users in choosing good passwords is to have web servers embed JavaScript inside their login pages that checks if a password satisfies certain preset conditions to determine strength (e.g. length, uppercase letter, punctuation symbol, numeric etc.). Since the checking is only done as client-side script, these meters are known to inform "Pa$$w0rd" as a good password.

Telepathwords. Probably derived from a combination of "telepathy" and "passwords", "Telepathwords" have been invented by Komanduri et al [4]. Basically, Telepathwords is a technique where a program guesses the probable next letter to be typed by the user when choosing a password and displays it, forcing the user to select a different character.

Password managers. Almost all browsers now including Mozilla, and Chrome come with a password manager that stores passwords, which can help the user in reducing the number of passwords that need to be memorized. Some add-on software also offer the same facility and can be installed into browsers. Jana et al. present an analysis of these and find that password managers can sometimes be tricked by attackers into revealing a stored password or allowing an authenticated session of a legitimate user [5].

3. Administrator guidance: Florencio et al.

Let us now look at the main take-away points given by Florencio et al. (reworded slightly for simplification).

The first concept by Florencio et al. deals with categorization of user accounts based on their importance and the consequences if the corresponding passwords are breached. In particular, they mention that choosing weak passwords for inconsequential servers (e.g. conference registration servers like easychair) should not be something that should spark outrage.

T1. Categorizing accounts based on the consequence of password leak is important as it ensures distribution of password selection effort appropriately and reduces possibility of cross-category password re-use.

Using only entropy to establish password strength is not adequate or sensible, since they cannot detect passwords that would appear in dictionaries. E.g. `P@Ssw0rd!` might have good entropy, but will be easily detected by guessing attack tools like JohnTheRipper [6]:

T2. Crude entropy-based estimates are unsuitable for measuring password resistance to guessing attacks; their use should be discouraged.

As mentioned before, guessing attacks are not the only way to learn passwords. They can also be learnt at the client machine where they are entered or on the wire when they are transferred:

T3. The success of threats such as client-side malware, phishing and sniffing unencrypted wireless links are entirely unaffected by password choice.

We will actually take it further and state that password leaks using any social engineering technique and any unencrypted communication (wired or wireless) is unaffected by password strength.

T4. Password guessing attacks are either online or offline. Resistance to them requires resisting 10^6 and 10^14 guesses respectively.

This huge gap between offline and online guessing leads to the next takeaway:

T5. No major gain in security is achieved by increasing password strength after resisting online guessing, unless it also resists offline guessing as well.

But if offline guessing is a concern because a server is uncertain over the security of its password file, it could salt passwords before hashing and storing:

T6. Rainbow table attacks, which are essentially lookup tables with precomputed hashes can be defeated by salting or leakage of password hashes.

But it turns out that offline guessing attacks are a cause of concern only if the above is done:

T7. Offline guessing attacks are a concern only in the limited situations when password files are salted and hashed, the files are leaked and the leaks go undetected.

But if a server does not hash, it implies the following:

T8. If a server does not salt and hash but stores passwords in plain or reversibly encrypted, it does not gain anything by enforcing requirements for choosing strong passwords.

Indeed, because if an attacker gains the password file (and the key if it was reversibly encrypted), he gets direct access to the passwords. He does not need to guess them at all!

T9. Online attacks cannot be avoided entirely, but offline attacks can be, by ensuring password file does not leak or mitigated by having mechanisms that detect leaks and have disaster-recovery plan to force system-wide resets.

Obviously, T9 is not always guaranteed; otherwise, entire password literature would simply focus on preventing online attacks only.

4. User guidance

As one can notice, as simple and sensible Florencio et al.'s takeaway points are, not all of them give guidance to the end-user in choosing passwords. Hence, we will now use the concepts explained in Florencio et al.'s work in deriving prudent practices for users to follow in order to minimize their effort in choosing and remembering passwords, and at the same time ensuring their security against them being learnt by attackers.

Our first prudent practice P1 is related to T1 by Florencio et al., which is categorization of accounts. To tackle the problem of having to maintain a large number of accounts, we suggest users to start by categorizing them according to their consequence. Though not backed up by a strong theoretical foundation, the following categorization advocated by Florencio et al. seems adequate for most individuals at this point:

(1) Don't care accounts. These are basically those where a password breach has no impact at all (e.g. one-time accounts for obtaining VISAs, pay conference registration fee etc.).

(2) Low-consequence accounts. These are accounts where a breach has some impact, though not life-changing for most. Examples include social networking accounts (excluding high-profile or celebrity users).

(3) Medium-consequence accounts. User accounts such as online banking, or online shopping where credit card details are stored etc. where a password breach at the minimum causes extra work such as seeking credit card replacement or PIN change.

(4) High-consequence accounts. The most important accounts with highest consequence for most users. For instance, password to learning management system for a professor, or the password for a system admin.

(5) Ultra-sensitive accounts. These are servers and things where extremely few individuals would have access to (e.g. nuclear missile codes, multi-million dollar irreversible bank transfers).

We will focus only on the three categories in between, but not the "don't care" accounts or ultra-sensitive accounts since they are either of no consequence or rarely concern the regular public.

For easy remembrance and avoidance of password sharing among inter-category accounts, we suggest the use of "seeds" that are unique for a category (Note that we also advocate against sharing usernames across categories as far as possible):

P1. It is prudent to choose a seed value that has enough entropy to withstand guessing attacks (online and/or offline as per the consequence of the accounts in the category and the password storage mechanism of the server). The seed should then be supplemented with server-specific information to derive unique passwords within a category.

As an example, one could choose `P@sswd` as the seed and derive passwords such as `myP@sswd4gmail`, `myP@sswd4fb` etc. for accounts in a low-consequence category.

The "password storage mechanism" in the above principle refers to the user knowledge on how the server stores its passwords, which is not always known to users. However, for some accounts it can

be found. For instance, if the server sends back passwords in email corresponding to "forgot my password" requests, it certainly stores passwords in plain-text. Also, if an account is employment-related, perhaps information on the level of security could be gathered on its storage. At any rate, once the mechanism of password storage is known, one can use the other principles below to transform the seed into a strong password.

P2. If a server emails back a password in plain or does not ask users for a strong password, most likely the server stores passwords in plain or encrypts the password file. Hence, it is prudent to not aim for a strong password. On the other hand, if a server asks to reset the password when a user forgot it, it most likely hashes and stores passwords. In those cases, depending on the account and the server, it is worth going for a password that resists online or offline guessing attacks.

For instance, a server that uses CAPTCHAS detects online guessing, it is not necessary to choose a password that resists online guessing. If the server seemingly does not detect online guessing attempts, depending on the consequence of the account, it is prudent to choose a password that would resist at least 10^6 guesses. Whether server is equipped against online guessing or not, if it is a high consequence account where the server seemingly hashes the passwords, a password (or even a "pass phrase" if needed and appropriate) that resists 10^{14} guesses is recommended to thwart off-line guessing attacks.

Even after choosing a seed for the category and creating a password, just basic entropy measures are not sufficient to check its strength, as explained in our next prudent practice:

P3. Just choosing a password that satisfies crude entropy based policies is <u>not</u> sufficient. It is important to create the password so that it cannot be created easily by finding a word in a suitable dictionary and changing some of the characters to symbols or numbers.

How much help are password meters? Florencio et al. make an interesting observation that we use to frame our next prudent practice in using password meters checking strength:

P4. Password meters are to be used as a rough guideline to measure password strength so as to resist guessing. Since they do not check against a dictionary but only static rules coded inside client-side script, a "bill of strength" from a meter does not guarantee strength against guessing.

Next we focus on password changes following expiration policies or self-suspicion of guessing attacks.

P5. Minor changes to passwords do not prevent guessing attacks (e.g. adding a numeric character at the end). Expiration policies are set to restrict the time for off-line guessing for attackers. But if password is learnt by guessing, the next password will be easily learnt.

For instance, changing one character to upper-case merely doubles the effort for the attacker in the case of guessing attacks [7].

We now move on to passwords entered on public machines or public wireless networks.

P6. If it is an account that is frequently accessed on public machines or public networks, changing the password frequently is the only option. If they are not learnt by guessing attacks, even minor changes suffice and strength doesn't matter.

Finally, we give a prudent practice for choosing usernames, once again following some observations pointed out by Florencio et al:

P7. Unless required by the server, refrain from using your email address as username, as it could lead you to use the same password. Also, if attacker gets the password file, he also gets your email address and can compromise your other accounts.

We have designed a flow-chart diagram based on these concepts, that can assist regular users in selecting their next password. The diagram is included as Fig 2 in Appendix due to its size.

5. Conclusion

In this paper, we have given a detailed history and concepts regarding passwords. We have analyzed Florencio et al.'s work in detail, summarized the important points and used them to derive prudent practices to guide users in choosing their passwords in a way that optimizes their effort while maximizing their security.

At present, we are working on expanding this, with precise guidelines to generate passwords using seeds for desired strength (e.g. to resist 10^6 guesses). We are also researching on prudent ways in which tools and techniques that can be utilized for strengthening passwords (e.g. using Telepathwords) or easing their management (e.g. using password managers). We are confident of finishing research on these aspects to present a fuller user guide on smart and secure password selection process.

6. References

[1] D. Florencio, C. Herley and P. C. van Oorschot, "An Administrator's Guide to Internet Password Research," in *USENIX Symposium*, 2014.

[2] B. Schneier, "Schneier on Security," March 2014. [Online]. Available: https://www.schneier.com/blog/archives/2014/03/choosing_secure_1.html.

[3] A. Narayanan and V. Shmatikov, "Fast Dictionary Attacks on Passwords Using Time-space Tradeoff," in *ACM CCS*, New York, NY, 2005.

[4] S. Komanduri, R. Shay, L. Cranor and C. Herley, "Telepathwords: preventing weak passwords by reading users' minds," in *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[5] D. Silver, S. Jana, E. Chen, C. Jackson and D. Boneh, "Password Managers: Attacks and Defenses," in *23rd USENIX Security Symposium (USENIX Security '14)*, San Diego, 2014.

[6] OpenWall, *John the Ripper password cracker*.

[7] M. Weir, S. Aggarwal, M. Collins and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *ACM CCS*, 2010.

Appendix

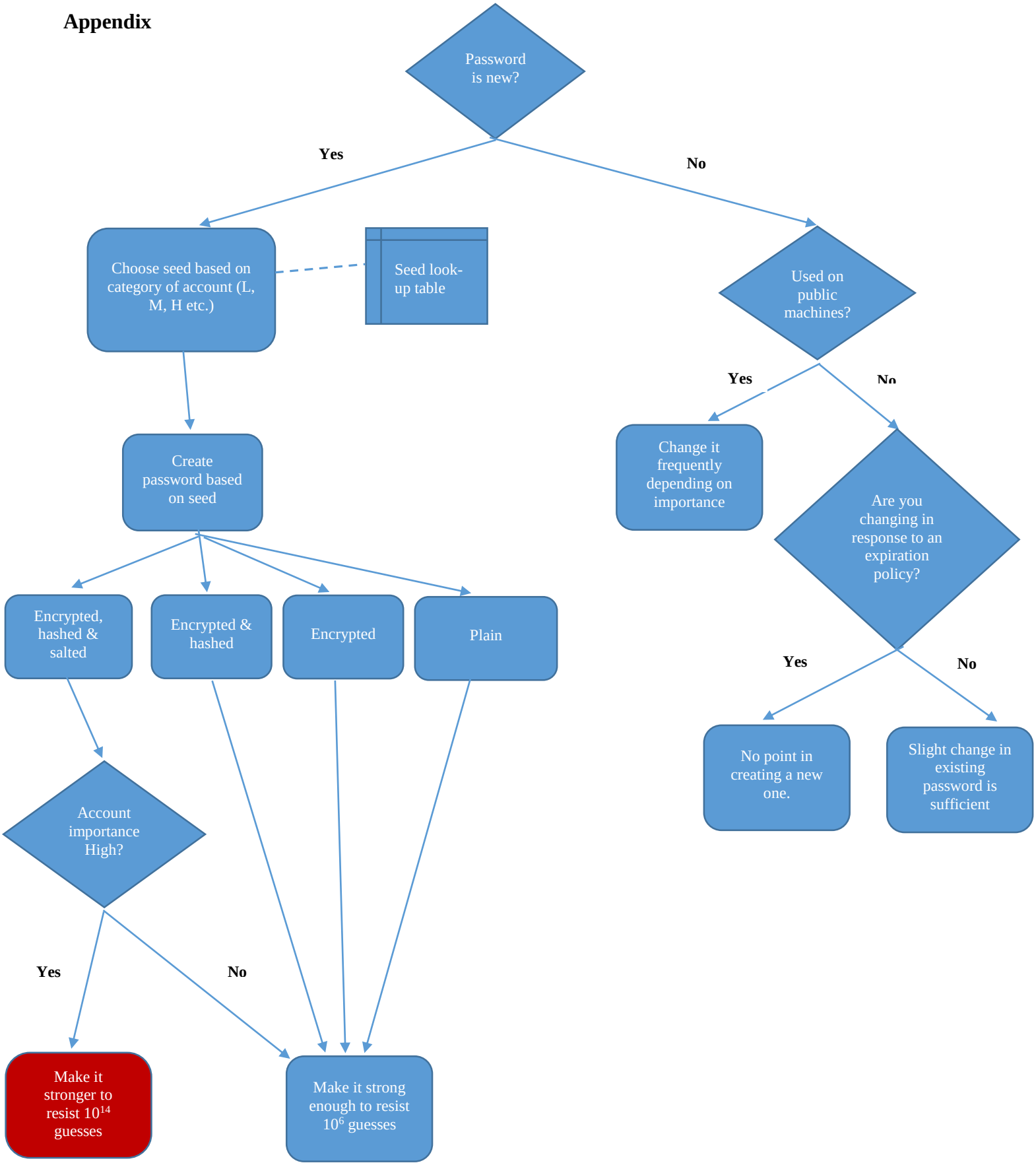

Fig 2. Flow-chart to select a password for new or existing accounts