Multi-item Passphrases: A Self-adaptive Approach Against Offline Guessing Attacks

Jaryn Shen[1], Kim-Kwang Raymond Choo[2], and Qingkai Zeng[1(✉)]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, China
jarynshen@gmail.com, zqk@nju.edu.cn

[2] Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249-0631, USA
raymond.choo@fulbrightmail.org

Abstract. While authentication has been widely studied, designing secure and efficient authentication schemes for various applications remains challenging. In this paper, we propose a self-adaptive authentication mechanism, Multi-item Passphrases, which is designed to mitigate offline password-guessing attacks. For example, "11th July 2018, Nanjing, China, San Antonio, Texas, research" is a multi-item passphrase. It dynamically monitors items and identifies frequently used items. Users will then be alerted when there is need to change their passphrases based on the observed trend (*e.g.*, when a term used in the passphrase consists of a popular item). We demonstrate the security and effectiveness of the proposed scheme in resisting offline guessing attacks, and in particular using simulations to show that schemes based on multi-item passphrases achieve higher security and better usability than those using passwords and diceware passphrases.

Keywords: Offline guessing attacks · Self-adaptive · Authentication Passphrases

1 Introduction

Access control schemes, such as those based on biometrics, graphical passwords, and hardware tokens [1,11], are fundamental in ensuring the security of systems and data. Due to the associated benefits of using (textual) passwords (*e.g.*, low cost, and ease of use and implementation), the latter is the most commonly used form of authentication and likely to remain popular in real-world applications.

There have been a number of incidents where hashed passwords were exfiltrated or leaked due to the servers being compromised (*e.g.*, by exploiting existing vulnerability(ies) in the servers) [4,8]. As most user passwords are human-memorable with low entropy [14,24], it would also be easy to brute-force

F. Breitinger and I. Baggili (Eds.): ICDF2C 2018, LNICST 259, pp. 204–221, 2019.
https://doi.org/10.1007/978-3-030-05487-8_11

such passwords offline, particularly using advanced probabilistic password guessing/cracking techniques [13,15,17]. In other words, organizations can no longer rely on only *salt* and hashing to protect user passwords.

In this paper, we propose a self-adaptive approach, Multi-item Passphrases. A multi-item passphrase is a sequence of multiple items. For example, "3rd August 2005, Jack, wife, holiday, Maldives" is a multi-item passphrase. Bonneau *et al.* [3] explained that passphrases are vulnerable to guessing attacks in practice, because users select popular words in natural language. Our approach is designed to mitigate this limitation because Multi-item Passphrases can self-adaptively remove popular words. In other words, our approach builds on existing password-based authentication and hence, will be significantly less expensive for e-commerce organizations to adopt this approach (rather than one that requires a complete overhaul of the entire system).

Our approach monitors items that are being used and flags frequently used words in a self-adaptive way. The items in a multi-item passphrase can be input in a different order than initially enrolled. The proposed Multi-item Passphrases approach includes our self-adaptive algorithm, which is designed to recognize and remove popular items dynamically. To demonstrate the potential of this approach, we build three text password-based authentication systems for evaluation. Findings from the quantitative experiment show that multi-item passphrases achieve higher security and better usability than passwords and diceware passphrases. The participants also rated multi-item passphrases higher. We also evaluate participants' attitude towards frequent passphrase changes and summarize their remarks.

In the next section, we will describe the relevant background materials.

2 Background

Offline attackers are able to guess user private passwords correctly from the files of hashed passwords, mainly due to advances in computing technologies and the low entropy of the passwords (*e.g.*, users selecting easy-to-remember passwords and the set that comprises these passwords is small).

For simplicity, we consider a setting with two users, Alice and Bob. All the possible passwords in the password space constitute the set P. In theory, Alice can choose any password in P. In fact, Alice only selects those that are easy to remember in P, say set A. In other words, passwords in $\bar{A}$ (*i.e.*, $P - A$) are hard to remember for Alice, and hence Alice does not choose passwords in $\bar{A}$. The same can be said for Bob, where Bob does not choose passwords in $\bar{B} = P - B$, where B is the set of easy-to-remember passwords for Bob.

In reality, there are passwords that are deemed to be easy to remember by both Alice and Bob. Thus, Alice's password set overlaps Bob's: $A \bigcap B \neq \emptyset$, which is illustrated in Fig. 1a. If there are three users: Alice, Bob and Charlie, then these three users' password sets contain certain common passwords (see Fig. 1b). If there are more users, they are most likely to share common passwords (see Fig. 1c). This is evident from several real-world incidents [15,24]. The implication

of this is that these commonly used passwords (the central dark circle in Fig. 1c) can then be used to facilitate password guessing, and this is also leveraged in password cracking tools such as John the Ripper.

Therefore, in this paper, we seek to remove the overlapping user password sets, such as the central dark circle in Fig. 1c.

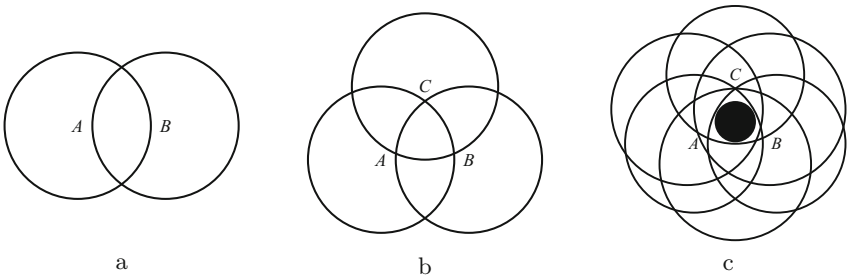

Fig. 1. (a) The area of circle A represents all potential passwords that Alice may choose, the area of circle B represents those that Bob may choose. Alice's password set overlaps Bob's: $A \bigcap B \neq \emptyset$. (b) Alice, Bob and Charlie's password sets contain common passwords: $A \bigcap B \bigcap C \neq \emptyset$. (c) Users' password sets contain common passwords. The central dark circle are the common passwords for all users.

Our proposed Multi-item Passphrases approach is based on the premise that if both Alice and Bob construct their passwords independently and in a way that both their potential password sets are different (*i.e.*, $A \bigcap B = \emptyset$), then the adversary will have no leverage in the password guessing. We will present the proposed approach in the next section.

3 Proposed Multi-item Passphrases Approach

Our approach seeks to identify and remove popular passwords from the password space.

3.1 Passphrase

There are a number of questions we need to consider. First, if a specific password is popular, should we remove the whole password or some characters in this password? If we do the former, how do we address other passwords that similar to this one and how do we define this kind of similarity? If the latter, which characters in this password should be removed? Additionally, there are only 95 printable characters in total, if we remove some characters, then the password space shrinks fast. Hence, we utilize passphrases instead of passwords.

A passphrase composes of items, and these are mainly English words. Common English words include nouns and noun phrases, such as "sun", "flower",

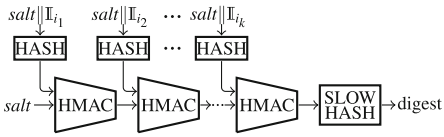

Fig. 2. The process from a user's passphrase to the final digital digest. The items in a passphrase are sorted in this process.

"Romeo and Juliet". As there are significantly more numerals and some numbers are private to users (*e.g.*, concatenation of user's and user next of kin's birthdays), attackers will take a longer time to correctly guess the numerals. Verbs and nouns can form sentences, and sentences may be clues to the attackers. Hence, we do not suggest users select a verb as an item. Adjectives and nouns always constitute frequently-used set structures, which decrease the attackers' guessing difficulty. Hence, we also do not suggest users select an adjective as an item. In order to enrich the item space, characters (including English letters, digits and symbols) and their arbitrary combinations are also considered as items in passphrases.

3.2 Modeling

The noun space is $\mathbb{N}_1$, the noun-phrase space is $\mathbb{N}_2$, the numeral space is $\mathbb{N}_3$, and the character space is $\mathbb{N}_4$. $\mathbb{N} = \mathbb{N}_1 \bigcup \mathbb{N}_2 \bigcup \mathbb{N}_3 \bigcup \mathbb{N}_4$. The total number of nouns, noun phrases, numerals, character combinations in $\mathbb{N}$ is $n = |\mathbb{N}|$. Each noun, noun-phrase, numeral, or character combination is an item, and each item belongs to $\mathbb{N}$. $\mathbb{N}$ is the item space. A passphrase $\mathcal{P}$ consists of k items: $\mathcal{P} = \{\mathbb{I}_{i_1}, \mathbb{I}_{i_2}, \mathbb{I}_{i_3}, ..., \mathbb{I}_{i_k}\}$. Each item is sorted by alphanumeric order in $\mathbb{N}$. For example, if $r < s$, $\mathbb{I}_r < \mathbb{I}_s$. Each passphrase $\mathcal{P}$ is just a set of items whose order does not matter. For example, if $\mathcal{P}_1 = \{\mathbb{I}_r, \mathbb{I}_s\}$ and $\mathcal{P}_2 = \{\mathbb{I}_s, \mathbb{I}_r\}$, then $\mathcal{P}_1 = \mathcal{P}_2$. Here, r, s, k, i_1, i_2, i_3, ..., i_k are all natural numbers.

When a user creates an account and specifies a passphrase $\mathcal{P} = \{\mathbb{I}_{i_1}, \mathbb{I}_{i_2}, \mathbb{I}_{i_3}, ..., \mathbb{I}_{i_k}\}$, the authentication system of the server computes the digital digest of this passphrase $\mathcal{P}$. The authentication system needs to sort the items in the passphrase $\mathcal{P}$ because the user can input these items in any order. Without loss of generality, we assume $i_1 < i_2 < i_3 < ... < i_k$. First, the authentication system generates a random *salt* value for this passphrase. Then, it computes the hash value for each item attached this *salt* with a hash function HASH: $H_1 = \text{HASH}(salt\|\mathbb{I}_{i_1})$, $H_2 = \text{HASH}(salt\|\mathbb{I}_{i_2})$, $H_3 = \text{HASH}(salt\|\mathbb{I}_{i_3}), ..., H_k = \text{HASH}(salt\|\mathbb{I}_{i_k})$. After that, the authentication system iteratively computes HMAC values with an HMAC function HMAC: $hmac_1 = \text{HMAC}(salt, H_1)$, $hmac_2 = \text{HMAC}(hmac_1, H_2)$, $hmac_3 = \text{HMAC}(hmac_2, H_3)$, ..., $hmac_k = \text{HMAC}(hmac_{k-1}, H_k)$. Finally, it computes the digest with a slow hash function SLOWHASH: $digest = \text{SLOWHASH}(hmac_k)$. These procedures are illustrated in Fig. 2.

Prior to computing the final digital digest with the slow hash function, we use *salt*, a hash function and an HMAC function to preprocess each item. *salt* is used to prevent offline attackers from building potential rainbow tables in advance. We use the hash function to avoid the need to keep plaintext items of passphrases in the memory. We use the HMAC function to turn each passphrase and its corresponding *salt* into the final digital digest. The SlowHash function is used to increase the time required for offline guessing attacks. In other words, the model is designed to increase the difficulty of offline attacks.

The authentication system stores the final digital digest with the account name and the *salt* value in a disk file, which is used for verifying the account information when a user logs on each time.

3.3 Usage

When a user creates an account, the authentication system will prompt the user to select passphrases. The user may select some nouns, noun phrases, numerals or character combinations to form the passphrase. A user can, for example, choose the items for the passphrase based on something that only the user knows (*e.g.*, a particular event such as a meeting) and can remember easily. For instance, a user selects "11th July 2018, Nanjing, China, San Antonio, Texas, research" to form the passphrase associated with the research collaboration between teams from Nanjing, China and San Antonio, Texas, and the kickoff meeting was held on July 11th, 2018. In other words, passphrases can be personalized and private to the users. Other example multi-item passphrases include "smile:-), @_@, '_', *_*, (-:elims" and "M_08032005, M_Jack, M_wife, M_holiday, M_Maldives". After specifying the passphrase, the user can then log on the system using the passphrase. The input order of the items in a passphrase is irrelevant.

4 Self-adaptive Algorithm

4.1 Definitions and Assumptions

Definition 1. *δ denotes the least upper bound of the total number of times needed to correctly guess a secure cryptographic secret key. If the least upper bound of the total number of times to correctly guess a password is no less than δ, then the password is secure.*

A cryptographic key is randomly generated. If the b-bit key is the shortest secure key, then $\delta = 2^b$. A password is selected by a user. Given a password, if the number of guesses to break this password is at least 2^b, then this password is secure and cannot be brute forced by the attackers in a reasonable amount of time. We give the following assumption.

Assumption 1. $\delta = 2^{128}$.

Here, $\delta = 2^{128}$ means that a 128-bit cryptographic key is currently secure. With advances in hardware and software, we need to increase the value of δ accordingly. Multi-item Passphrases is robust, and it is not affected by the exact value of δ. $\delta = 2^{128}$ in Assumption 1 is for demonstration purpose only.

Definition 2. *ϵ denotes a maximal negligible probability of occurrence. If the probability of an event is not greater than ϵ, then this event is negligible and unlikely to happen. However, if the probability of an event is greater than ϵ, then this event is non-negligible and likely to happen.*

Assumption 2. $\epsilon = 2^{-80}$.

For example, consider an event: p users select the same item ($p \geq 2$). Assume that each item is selected with equal probability in the item space $\mathbb{N}$, we can obtain P, the probability that p users select the same item: $P = |\mathbb{N}|^{-p}$. If $P > \epsilon$, this event is likely to happen; otherwise (*i.e.*, $P \leq \epsilon$), this event is unlikely to happen.

4.2 Algorithm

In order to explain the self-adaptive algorithm, we start with the simplest case and work our way up to the general case. Consider the simplest case: there are only two users, Alice and Bob. Alice chooses a passphrase $\mathcal{P}_A$, which consists of $|\mathcal{P}_A|$ items, and Bob chooses $\mathcal{P}_B$ comprising $|\mathcal{P}_B|$ items. $\mathcal{P}_A$ and $\mathcal{P}_B$ have q identical item(s): $|\mathcal{P}_A \cap \mathcal{P}_B| = q$, $q \geq 1$. $P_{|\mathcal{P}_A \cap \mathcal{P}_B|=q}$ denotes the probability that q item(s) are identical in $\mathcal{P}_A$ and $\mathcal{P}_B$. Assume each item is selected with equal probability in the item space $\mathbb{N}$, and $n = |\mathbb{N}|$, we obtain the following equation:

$$P_{|\mathcal{P}_A \cap \mathcal{P}_B|=q} = \frac{\binom{n}{|\mathcal{P}_A|}\binom{|\mathcal{P}_A|}{q}\binom{n-|\mathcal{P}_A|}{|\mathcal{P}_B|-q}}{\binom{n}{|\mathcal{P}_A|}\binom{n}{|\mathcal{P}_B|}} \tag{1}$$

If $P_{|\mathcal{P}_A \cap \mathcal{P}_B|=q} \leq \epsilon$, then $\mathcal{P}_A \cap \mathcal{P}_B$ are popular item(s). Reasons are as follows:

It is unlikely that both $P_{|\mathcal{P}_A \cap \mathcal{P}_B|=q} \leq \epsilon$ and $|\mathcal{P}_A \cap \mathcal{P}_B| = q \geq 1$ are true according to Definition 2. Because it is the established fact that $|\mathcal{P}_A \cap \mathcal{P}_B| = q \geq 1$, which must be true, it is false that $P_{|\mathcal{P}_A \cap \mathcal{P}_B|=q} \leq \epsilon$. Therefore, $P_{|\mathcal{P}_A \cap \mathcal{P}_B|=q} > \epsilon$. This means that Alice and Bob select these q item(s) with a higher probability in $\mathbb{N}$. Hence, $\mathcal{P}_A \cap \mathcal{P}_B$ are popular item(s).

Thus, if the value of Eq. 1 is not greater than ϵ, then the self-adaptive algorithm should remove $\mathcal{P}_A \cap \mathcal{P}_B$ from $\mathbb{N}$ and Alice and Bob should change their passphrases.

Consider the general case: there are u users in the system and $n = |\mathbb{N}|$. $P_{|\bigcap_{i=1}^{p} \mathcal{P}_i|=q}$ denotes the chance of $|\bigcap_{i=1}^{p} \mathcal{P}_i| = q$, where $2 \leq p \leq u, 1 \leq q \leq \min_{i=1}^{p} |\mathcal{P}_i|$:

$$P_{|\bigcap_{i=1}^{p} \mathcal{P}_i|=q} = \frac{\binom{n}{q}\binom{n-q}{|\mathcal{P}_1|-q}\prod_{i=2}^{p}\binom{n-q-\sum_{j=1}^{i-1}(|\mathcal{P}_j|-q)}{|\mathcal{P}_i|-q}}{\prod_{i=1}^{p}\binom{n}{|\mathcal{P}_i|}} \tag{2}$$

Similarly to Eq. 1, when both $P_{|\bigcap_{i=1}^{p}\mathcal{P}_i|=q} \leq \epsilon$ and $|\bigcap_{i=1}^{p}\mathcal{P}_i| = q$, one of the two must be false. Since $|\bigcap_{i=1}^{p}\mathcal{P}_i| = q$ is the established fact, $P_{|\bigcap_{i=1}^{p}\mathcal{P}_i|=q} \leq \epsilon$ is false. Therefore, $P_{|\bigcap_{i=1}^{p}\mathcal{P}_i|=q} > \epsilon$ and $\bigcap_{i=1}^{p}\mathcal{P}_i$ are popular items. The self-adaptive algorithm removes $\bigcap_{i=1}^{p}\mathcal{P}_i$ from $\mathbb{N}$, and these p users should change their passphrases.

When a new passphrase $\mathcal{P}^*$ is submitted to the system, we need not compute Eq. 2 with all the passphrases $\bigcup_{i=1}^{u}\{\mathcal{P}_i\}$ because those passphrases $\bigcup_{i=1}^{u}\{\mathcal{P}_i\} - \{\mathcal{P}^*\}$ have been handled before $\mathcal{P}^*$ joins in. Hence, the self-adaptive algorithm for the general case (see Algorithm 1) is efficient.

Algorithm 1. Self-Adaptive Algorithm for the general case

Input: $\mathcal{P}^*$, $\bigcup_{i=1}^{u}\{\mathcal{P}_i\}$ //Input the passphrases, $\mathcal{P}^* \in \bigcup_{i=1}^{u}\{\mathcal{P}_i\}$.

Output: $\hat{\mathcal{P}}$, $\overline{\mathbb{N}}$ //Output the popular items, $\hat{\mathcal{P}} \subseteq \overline{\mathbb{N}}$.

1: $\mathbb{S} \leftarrow \bigcup_{i=1}^{u}\{\mathcal{P}_i\} - \{\mathcal{P}^*\}$ //$\mathbb{S}$ contains what are handled, $\mathcal{P}^*$ is a fresh one.
2: Compare $\mathcal{P}^*$ to every element in $\mathbb{S}$ //To see whether there are identical item(s).
3: Find all distinct groups that p passphrases from $\mathbb{S}$ which have the same q item(s) as $\mathcal{P}^*$ //That is, $|(\bigcap_{i=1}^{p}\mathcal{P}_i)\bigcap\mathcal{P}^*| = q \geq 1$.
4: **if** There is no passphrase from $\mathbb{S}$ that has the same item as $\mathcal{P}^*$ **then**
5: Exit the algorithm //Find no identical item(s), so exit.
6: **end if**
7: **for** each group that $p \geq 1$ **do** //If no such group, do Line 4 to Line 6.
8: $\hat{\mathcal{P}} \leftarrow (\bigcap_{i=1}^{p}\mathcal{P}_i)\bigcap\mathcal{P}^*$ //For convenience, let $\hat{\mathcal{P}}$ denote $(\bigcap_{i=1}^{p}\mathcal{P}_i)\bigcap\mathcal{P}^*$.
9: **if** $P_{|\hat{\mathcal{P}}|=q} \leq \epsilon$ **and** $|\hat{\mathcal{P}}| = q$ **then** //$P_{|\hat{\mathcal{P}}|=q} \leq \epsilon$ is false; $|\hat{\mathcal{P}}| = q$ is the established fact.
10: $\overline{\mathbb{N}} \leftarrow \overline{\mathbb{N}} + \hat{\mathcal{P}}$ //$\hat{\mathcal{P}}$ are q popular item(s), which are added to $\overline{\mathbb{N}}$.
11: Set these $p+1$ users' *PassphraseStatus* fields to "1"
12: **end if**
13: **end for**

Once a user registers or changes a passphrase, the self-adaptive algorithm executes. If the self-adaptive algorithm asks some users to change their passphrases, then these requests will be recorded in the disk file that stores the account information. The main structure of this disk file is as below:

$$|UserName|salt|DigitalDigest|PassphraseStatus|$$

The *PassphraseStatus* field stores the status of the corresponding user's passphrase. The default value of this field is "0". If the self-adaptive algorithm

asks a user to change the passphrase, then this user's *PassphraseStatus* field is set to "1". When a user logs on with success, if the value of this user's *PassphraseStatus* field is "1", this user is asked to change the passphrase at once. If this user changes the passphrase successfully, then the value of this field is reset to "0"; otherwise, the system rejects this user's further access until this user changes the passphrase successfully.

It is noteworthy that the self-adaptive algorithm does not directly compare the plaintext items of different passphrases other than the hash values of the items. All hash values of the items for every users' passphrase are kept in memory. The plaintext passphrases are discarded. For the purpose of resisting the rainbow table, we hash every item with the attached *salt* of the corresponding passphrase. This is performed to prevent a malicious insider from dumping the contents of memory and learning all the passphrases. The item comparison of different passphrases incurs little cost.

We do not employ the slow hash function to protect the items in memory just because slow hashing is time-consuming and Eq. 2 (see Line 9 in Algorithm 1) needs to compare each hashed item with attached *salt*. As each user's *salt* is generally different, it will need a lot of time if slow hashing is employed in the items in memory especially when there are millions of users in the system.

The digital digests of every passphrase are stored in the disk file. When rebooting the server, all the hash values of the items of every passphrase in memory will disappear. However, it does not matter since the system can recognize users because the disk file records the digests of user passphrases. The removal of the hash values in the memory results in $\mathbb{S} = \emptyset$ in Algorithm 1. If the rebooting is intentional, then all the hashed items in memory can be dumped before rebooting and restored after rebooting. If the rebooting is accidental, as users log on successively, $\mathbb{S}$ is filled up again. The accidental rebooting does not affect the registered user passphrases, but newly registering user passphrases may be under the influence of the accidental rebooting because $\mathbb{S}$ will not become full swiftly after the accidental rebooting. The solution to the accidental rebooting is simple: rerun Algorithm 1 for these newly registering user passphrases as long as $\mathbb{S}$ is changed by those registered user passphrases. The rebooting of the server does not destroy the self-adaptive algorithm (*i.e.*, Multi-item Passphrases is reliable).

Scaling up is also not a concern, although Algorithm 1 might be time-consuming. For example, according to the *PassphraseStatus* field of the disk file, users can change their passphrases in a timely fashion. Without the slow hash operation, the system can protect user passphrases ahead of offline attackers. Moreover, we can obtain the value of Eq. 2 by a look-up table, so as to avoid unnecessary repetitive computation.

4.3 Item Space Construction

There are two methods to construct the item space: the direct method and the indirect method. The direct method is to set a large table representing $\mathbb{N}$, all the potential items are contained in this table. When need to remove some popular

items selected by users, just remove them from this large table. The indirect method is to set a table representing $\overline{\mathbb{N}}$. The items outside this table are all feasible items that users can select. Once need to remove some popular items chosen by users, just add them into this table.

We adopt the latter indirect method. Users are not allowed to select items in the table $\overline{\mathbb{N}}$. All of the elements in $\overline{\mathbb{N}}$ are generated dynamically by the self-adaptive algorithm (see the 10th line in Algorithm 1). As a supplement, we can also add popular words to $\overline{\mathbb{N}}$ in advance so as to stop users choosing these popular items.

The most main reason we do not employ the former but the latter is that a direct item space table $\mathbb{N}$ exposes users' password sets and is equivalent to a word list of a password cracking tool, which helps attackers. Instead, the table $\overline{\mathbb{N}}$ can include word lists of existing password cracking tools in advance, which resists password guessing attacks.

4.4 Size of Item Space

Because we take numerals and character combinations into account, the actual size of the item space is infinite. For maximum convenience of users, we manage to make our approach easy to use on the condition that only English words belong to the item space.

Given a k-item passphrase $\mathcal{P}$, according to Definition 1, for security, the number of guesses to certainly crack $\mathcal{P}$ should be at least δ. δ is made up of two parts: $\delta = \delta_1 \times \delta_2$. Part 1, δ_1, derives from the traversal of $\binom{n}{k}$ distinct passphrases. Part 2, δ_2, is the number of cycles for slow hash, which stems from the slow hash function in the model of Multi-item Passphrases.

A frequently-used slow hash function is *bcrypt* [18]. We take *bcrypt* as an example of the slow hash function in this paper. In *bcrypt*, $\delta_2 = 2^{cost}$, *cost* is an adjustable parameter. According to Assumption 1, $2^{128} = \binom{n}{k} \times 2^{cost}$. We assign 88 to *cost* and evaluate n with different k. The detailed results are listed in Table 1.

Table 1. Size of item space

k	n ($\delta_1 = 2^{128}, \delta_2 = 1$)	n ($\delta_1 = 2^{40}, \delta_2 = 2^{88}$)
2	$>2.6 \times 10^{19}$	1,482,911
3	$>1.2 \times 10^{13}$	18,756
4	9,506,325,306	2,268
5	132,496,421	669
6	7,910,346	307
7	1,080,111	181

When the slow hash function is out of use, $\delta_2 = 1$, $\delta_1 = 2^{128}$, and n exceeds one million even though $k = 7$. Because there are not so many nouns in English, the slow hash function is necessary to our approach in consideration of usability.

When $cost = 88$, $\delta_2 = 2^{88}$ and $\delta_1 = 2^{40}$. Assume the hash time of the slow hash function is 1 ms when $cost = 88$. The average time that attackers crack a passphrase equals 1 ms times 2^{39}, which means a large time span for more than 17 years. It is a time span long enough that we believe the passphrase is secure. In order to ensure this level of security of a passphrase, if $k = 7$, the item space needs only 181 items; if $k = 3$, the item space needs no more than 20,000 items. At the same time, there are 218,632 words inside the Second Edition of the *Oxford English Dictionary*, over half of them (more than 100,000 entries) are nouns, and noun phrases are much more [6,16]. Hence, provided the slow hash function comes into use, the item space is big enough to actualize our approach Multi-item Passphrases.

4.5 Relationship Between δ and $\delta_2 = 2^{cost}$

When hardware computing speed increases, δ and $\delta_2 = 2^{cost}$ should be augmented correspondingly. Thus, δ_1 need not be changed because $\delta = \delta_1 \times 2^{cost}$, as explained using the following example:

Assume the hash time of the slow hash function is 1 ms when $cost = 20$, and $\delta_1 = 2^{40}$, then $\delta = 2^{60}$ means a large time span for more than 34 years. If hardware computation speeds up 1000 times, the iterative times of the slow hash function $\delta_2 = 2^{20}$ is reduced from 1 ms to 1 microsecond. Then, we adjust *cost* from 20 to 30, remain $\delta_1 = 2^{40}$, and $\delta = 2^{70}$ still means a large time span for more than 34 years.

Hence, we do not need to expand the passphrase space as hardware computing speed improves. Multi-item Passphrases is a stable approach.

We assume $\delta = 2^{128}$ in Assumption 1 and assign 88 to *cost* in Table 1. They are just examples for demonstration purposes. In fact, because $\delta = \delta_1 \times \delta_2 = \delta_1 \times 2^{cost}$, we can always obtain an appropriate value of δ through adjusting the value of *cost* as long as δ_1 is large enough such as $\delta_1 = 2^{40}$.

4.6 Parameter Values

n can be assigned $\epsilon^{-0.5}$ in the self-adaptive algorithm although the actual item space is boundless. In Eq. 2, when n is assigned no less than $\epsilon^{-0.5}$, different users are restricted from selecting the same item; when n is assigned less than $\epsilon^{-0.5}$, the smaller n is, more of the same items can be selected by different users. We may decrease the value of n in order to reduce users' frequency of changing their passphrases due to popular items.

In Assumption 2, we set $\epsilon = 2^{-80}$, which is a widely used value in the cryptographic literature.

k is a small integer between 3 and 20 generally, which is used to strike a balance between security and the usability of passphrases. In general, given an

item space, the larger k is, the more secure passphrases are. The smaller k is, the more usable passphrases are.

$\delta_1 = \sum_k \binom{n}{k}$.

5 Experiments

We performed a series of experiments to evaluate the multi-item passphrases, using an online website we built. There were four experiments on this website. These experiments contained three text password-based authentication systems: passwords, multi-item passphrases, and diceware passphrases [19]. For the experiments, we recruited a total of 372 undergraduate students, aged between 18 and 23, from the school of management. There were 189 female and 183 male participants. Our experiments were approved by Nanjing University's IRB.

We randomly grouped these 372 participants into four groups by their student numbers, in order to obtain four independent samples at random: Group 1: 86 persons for Experiment 1; Group 2: 81 persons for Experiment 2; Group 3: 86 persons for Experiment 3; and Group 4: 119 persons for Experiment 4.

The experiments were divided into two phases: day one and day four (*i.e.*, three days after they had started the experiment). In the second phase (*i.e.*, day four), only 293 participants (149 females and 144 males) returned to complete the experiments.

- Group 1: 68 persons
- Group 2: 74 persons
- Group 3: 73 persons
- Group 4: 78 persons

The experiment setup is now explained below.

Experiment 1: Passwords. We asked the recruited participants to visit our website on a certain day. They were asked to register and log on to their accounts using their user credentials. Their account names were arbitrary, and the passwords they specified must include at least three of the following four character types: lowercase English letters, uppercase English letters, numbers, and symbols, and the passwords were at least 16 characters long.

Three days later, we asked them to access our web site again and log on to their accounts. On this occasion, however, they were presented with their account names and they only needed to input their passwords. Participants were allowed to try as many times as possible, if they had entered an incorrect password. Then, we asked the participants to comment on the usability of this authentication system.

Experiment 2: Multi-item Passphrases. In this experiment, we assigned $\epsilon^{-0.5}$ to n in Eq. 2. Participants were asked to register and log on to their accounts using their chosen multi-item passphrases on our web site on a certain day. The passphrases must include five or more than five unpopular items and the order of these items was irrelevant. If an item they specified was popular, then they were

asked to change their passphrases. Similar to Experiment 1, the participants returned three days later to enter their passphrases. However, if an item in their passphrases had become popular during these three days, then they would be asked to change their passphrases when they logged on.

We asked for their comments on the usability of this authentication system, and particularly the following questions:

Q(1) How do you feel about the need to change your passphrase (repeatedly) when you select a popular item during account registration?
Q(2) How do you feel about the need to change your passphrase because others selected the same item(s) as you?

To these two questions above, we provided the following four options for the participants:

(A) It is good, because now I know which item is popular and can avoid using such item(s) from now on.
(B) I can accept it in order to ensure that my account is secure.
(C) It annoys me a little as I have to think of a different passphrase.
(D) It is absolutely unacceptable!

Experiment 3: Diceware Passphrases. Participants were told to register and log on accounts using diceware passphrases, which include five ordered diceware words selected at random from the diceware item list [19]. Three days later, they were asked to log on to their accounts on our website and comment on the usability of this authentication system.

Experiment 4: Rating. Participants were asked to finish all of the experiments in Experiments 1, 2 and 3. After that, based on usability and security, they were asked to rate these three password-based authentication systems where 1 is the lowest (worst) and 3 is the highest (best). They had to also provide a reason for the ratings.

5.1 Findings I: Multi-item Passphrases vs. Passwords

Quantitative Results. The findings are as follows: after creating their accounts, 28 of the 68 participants remembered their passwords and logged on to their accounts successfully three days later in Experiment 1; 40 of the 74 participants remembered their multi-item passphrases and logged on to their accounts successfully three days later in Experiment 2.

Strength. In Experiment 1, we asked participants to specify a 16-character-long password although most passwords were only 8 characters long, and the password space of a 16-character-long password is 95^{16}.

In Experiment 2, without even considering the numerical and character combinations, the total number of nouns and noun phrases is significantly larger than 669 based on what we obtained from the *Oxford English Dictionary* [6,16]. Thus, according to Table 1, $n \gg 669$ and $k \geq 5$, so $\delta_1 \gg 2^{40}$ and $\delta \gg 2^{128}$ when

$\delta_2 = 2^{88}$. Because $\delta \gg 2^{128} \gg 95^{16}$, considering only English words, the multi-item passphrases' space is significantly larger than that of passwords. Hence, taking the numerical and character combinations into account, the strength of multi-item passphrases in Experiment 2 is very much greater than that of passwords in Experiment 1.

Recall. Let p_1 and p_2 be the true proportion of users who can recall successfully their multi-item passphrases and passwords, respectively.

$H_0 : p_1 = p_2$.
$H_1 : p_1 > p_2$.
$\alpha = 0.1$.

The null hypothesis of H_0 is that multi-item passphrases and passwords are both equally hard to remember, and the alternative hypothesis H_1 indicates that multi-item passphrases are easier to remember than passwords. For the hypothesis testing, we use *"Two Samples: Tests on Two Proportions"* [23]:

$$z = \frac{\hat{p_1} - \hat{p_2}}{\sqrt{(\frac{1}{n_1} + \frac{1}{n_2})(\hat{p}(1-\hat{p}))}} \tag{3}$$

In Eq. 3, the sizes of the two independent samples are n_1 and n_2, respectively: $n_1 = 74, n_2 = 68$. The point estimates of p_1 and p_2 are $\hat{p_1}$ and $\hat{p_2}$ for the two samples respectively, and they are computed as $\hat{p_1} = \frac{40}{74}$ and $\hat{p_2} = \frac{28}{68}$. The pooled estimate of the proportion is $\hat{p} = \frac{40+28}{74+68}$. Thus, $z = 1.53$. Therefore, $P = P(Z > 1.53) = 0.0630$. Hence, we reject H_0 ($\alpha = 0.1$ and $p = .0630$) and accept H_1. In other words, multi-item passphrases are easier to remember than passwords.

5.2 Findings II: Multi-item Passphrases vs. Diceware Passphrases

Quantitative Results. The findings are as follows: after creating their accounts, 19 of the 73 participants remembered their diceware passphrases and logged on to their accounts successfully three days later in Experiment 3; 40 of the 74 participants remembered their multi-item passphrases and logged on to their accounts successfully three days later in Experiment 2.

Strength. In Experiment 2, there are a large number of non-popular items (*e.g.*, numeric and arbitrary combinations of characters). Since the diceware list only contains 7776 items, the diceware passphrase space is $P_5^{7776} = 5!\binom{7776}{5} = 120\binom{7776}{5}$ in Experiment 3. Hence, the strength of multi-item passphrases is much greater than that of diceware passphrases.

Recall. The findings here echoed those of Findings I, in the sense that multi-item passphrases are much easier to remember than diceware passphrases ($p < .0001$).

5.3 Findings III: Usability Under Frequent Passphrase Changes

The breakdown to the questions is as follows: Question (1): 41 participants chose (A), 68 participants chose (B), 42 participants chose (C), and one chose (D); and Question (2): 19 participants chose (A), 67 participants chose (B), 46 participants chose (C), and 20 participants chose (D). The findings are also reported in Fig. 3.

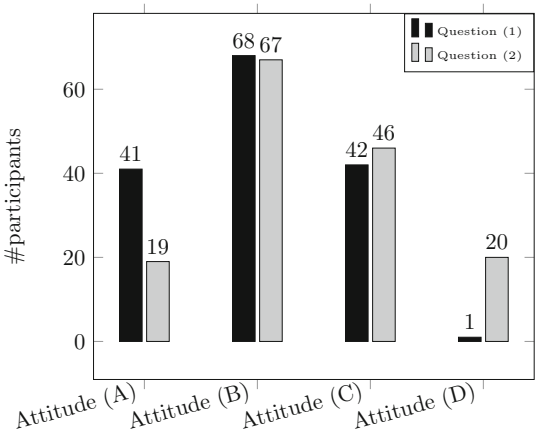

Fig. 3. Attitudes to frequent passphrase changes.

From the findings, we observe that the majority of the participants were positive towards Question (1), since they might be used to frequently changing their passwords due to stringent password composition policies (although they might not enjoy doing so, and this is beyond the scope of this study).

Participants were generally more negative towards Question (2) because they were new to the fact that they had to change the passphrase for their existing account just because it comprised a popular item. This is a mindset that will require some adjusting. Users could, for example, be educated on why they need to avoid popular passwords/passphrases (*e.g.*, the overlapping parts in Fig. 1c).

5.4 Findings IV: Rate Three Authentication Systems

In Experiment 4, 78 participants scored the three systems and the findings are shown in Table 2.

From Table 2, we observe that multi-item passphrases system has the highest score of 2.58, followed by passwords (score 1.87) and diceware passphrases (score 1.55). The average score of multi-item passphrases is much higher than those of the other two (Kruskal-Wallis test, $p < .0001$). The highest rating score suggests that the participants strongly believe that Multi-item Passphrases is a better password-based authentication system than passwords and diceware passphrases.

Table 2. Ratings for the three authentication systems

Systems	#Score			Average score
	#1	#2	#3	
Multi-item passphrases	4	25	49	2.58
Passwords	26	36	16	1.87
Diceware passphrases	48	17	13	1.55

5.5 Findings V: Qualitative Results from Participants' Remarks

Participants in Groups 1, 2, and 3 commented on the respective authentication systems, and participants in Group 4 explained their ratings. Some interesting findings are as follows:

(1) Although the space for passwords is much greater than the space for diceware passphrases (*i.e.*, $95^{16} \gg P_5^{7776}$), many participants believed that "*diceware passphrases were more secure than passwords*". They did not trust the security of passwords, which is perhaps because several participants "*had the experience of having the passwords stolen*".
(2) Many participants commented that "*diceware passphrases were secure*", but at the same time, they also said they "*would never consider making use of it because it was too difficult to remember*".
(3) Participants' security awareness was strong, which is not surprising consider the age group of these participants (*i.e.*, digital natives) and their educational background. Some participants even commented that they hoped to see the next release of our research.

We now present a snapshot of the participants' remarks on the three password-based authentication systems:

Passwords. "*A 16-character password was too long.*" Although it was the most commonly used, "*it was somewhat difficult to recall*" as it must contain three of the following four character types: lowercase English letters, uppercase English letters, numbers, and symbols. Hence, "*A shorter or simpler password was better*" for using but was easy to crack.

Multi-item Passphrases. "*It was the best of the three.*" "*It was a good tradeoff between user experience and security.*" "*The order of items in a passphrase was indifferent, which was humanized.*" "*The system could find popular items, which was good.*" However, "*five items were a bit too many*".

Diceware Passphrases. It was picked randomly, so "*it was very hard to remember*" although "*it was secure*" because of its randomness. "*It was a painful and difficult experience*" for users.

6 Related Work

Passwords have been used for decades, and one known limitation is password-guessing attacks such as offline guessing attacks [2,5,7,15]. Passphrases may be a slightly better option than passwords because passphrases are generally harder to guess, with minimal implementation changes or disruption to user experience [22].

A closely related work is Diceware [19], which comprises a Diceware list. In order to pick passphrases, Diceware selects items at random from the Diceware list. Although Diceware is simple and straightforward, participants rated our approach to be better than Diceware in terms of usability and security (see Findings II).

Another similar related work is Telepathwords [10]. Telepathwords predicts users' passwords with several fixed patterns to prevent users from creating weak passwords. These patterns include common character sequences, keyboard movements, repeated strings, and interleaved strings. This particular approach relies on the pattern of a password, while Multi-item Passphrases focuses on concrete elements (popular items) in passphrases. It is not sufficient to make use of finite fixed patterns to avoid weak passwords. For example, Telepathwords can detect "abcdefg", but cannot detect the weak password "gfedcba" because Telepathwords does not consider reverse character sequences. The constructive patterns of passphrases are much more complicated than those of passwords. Hence, Multi-item Passphrases does not employ fixed patterns to avoid weak passphrases. Instead, our approach prevents the use of popular items in user passphrases.

In the Bloom filter-based approach proposed by Schechter *et al.* [20], a key limitation is a high false positive (*i.e.*, unpopular passwords being wrongly flagged as popular). Moreover, their approach cannot resist offline attacks due to the inherent weakness of the Bloom filter.

Segreti *et al.* [21] undertook a user study, focusing on the approaches of Schechter *et al.* [20] and PathWell [12]. PathWell is similar to Telepathwords. Key differences between the study of Segreti *et al.* and ours are: the former focuses on passwords and does not include any qualitative study.

When constructing the item space, we construct a complement of the item space: $\overline{\mathbb{N}}$. The concept "a complement of the item space" is similar to the password blacklist studied by Habib *et al.* [9]. A key difference between them is that the complement of the item space is automatically generated and dynamically added in our approach.

7 Conclusion

It may not be realistic to expect that password-based systems will fade away in the near future, and hence we need to design sufficiently robust password-based authentication systems to mitigate existing known limitations (*e.g.*, users selecting weak and easy-to-guess passwords). Thus, this motivated the design of our proposed Multi-item Passphrases approach. We demonstrated the security of the proposed approach, as well as evaluated its performance.

Acknowledgement. We thank the anonymous reviewers for their constructive feedback. This work has been partly supported by National NSF of China under Grant No. 61772266, 61572248, 61431008.

References

1. Biddle, R., Chiasson, S., Van Oorschot, P.C.: Graphical passwords: learning from the first twelve years. ACM Comput. Surv. (CSUR) **44**(4), 19 (2012)
2. Bonneau, J., Herley, C., van Oorschot, P.C., Stajano, F.: Passwords and the evolution of imperfect authentication. Commun. ACM **58**(7), 78–87 (2015)
3. Bonneau, J., Shutova, E.: Linguistic properties of multi-word passphrases. In: Blyth, J., Dietrich, S., Camp, L.J. (eds.) FC 2012. LNCS, vol. 7398, pp. 1–12. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34638-5_1
4. Burnett, M.: Today I am releasing ten million passwords, February 2015. https://xato.net/passwords/tenmillion-passwords/
5. Chatterjee, R., Athayle, A., Akhawe, D., Juels, A., Ristenpart, T.: pASSWORD tYPOS and how to correct them securely. In: IEEE Symposium on Security and Privacy, pp. 799–818 (2016)
6. Oxford Living Dictionaries: How many words are there in the English language? (2018). https://en.oxforddictionaries.com/explore/how-many-words-are-there-in-the-english-language/
7. Wang, D., Cheng, H., Wang, P., Yan, J., Huang, X.: A security analysis of honeywords. In: Proceedings of the 25th Annual Network and Distributed System Security Symposium (2018)
8. D'Orazio, C., Choo, K.K.R., Yang, L.T.: Data exfiltration from Internet of Things devices: iOS devices as case studies. IEEE Internet Things J. **4**(2), 524–535 (2017)
9. Habib, H., et al.: Password creation in the presence of blacklists. In: Proceedings of USEC (2017)
10. Komanduri, S., Shay, R., Cranor, L.F., Herley, C., Schechter, S.E.: Telepathwords: preventing weak passwords by reading users' minds. In: USENIX Security Symposium, pp. 591–606 (2014)
11. Krol, K., Philippou, E., De Cristofaro, E., Sasse, M.A.: "they brought in the horrible key ring thing!" analysing the usability of two-factor authentication in UK online banking. In: Symposium on NDSS Workshop on Usable Security (2015)
12. Leininger, H.: Libpathwell 0.6.1 released, 2015 (2015). https://blog.korelogic.com/blog/2015/07/31/libpathwell-0_6_1
13. Li, Z., Han, W., Xu, W.: A large-scale empirical analysis of Chinese web passwords. In: Proceedings of 23rd USENIX Security Symposium, USENIX Security, August 2014
14. Mazurek, M.L., et al.: Measuring password guessability for an entire university. In: Proceedings of the 20th ACM SIGSAC Conference on Computer and Communications Security, pp. 173–186. ACM (2013)
15. Morris, R., Thompson, K.: Password security: a case history. Commun. ACM **22**(11), 594–597 (1979)
16. OED: Dictionary milestones: a chronology of events relevant to the history of the OED (2017). http://public.oed.com/history-of-the-oed/dictionary-milestones/
17. Paul: New 25 GPU monster devours passwords in seconds, December 2012. https://securityledger.com/2012/12/new-25-gpu-monster-devours-passwords-in-seconds/
18. Provos, N., Mazieres, D.: A future-adaptable password scheme. In: USENIX Annual Technical Conference, FREENIX Track, pp. 81–91 (1999)

19. Reinhold, A.G.: The diceware passphrase home page, October 2017. http://world.std.com/~reinhold/diceware.html
20. Schechter, S., Herley, C., Mitzenmacher, M.: Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In: USENIX Conference on Hot Topics in Security, pp. 1–8 (2010)
21. Segreti, S.M., et al.: Diversify to survive: making passwords stronger with adaptive policies. In: Symposium on Usable Privacy and Security (SOUPS) (2017)
22. Tazawa, H., Katoh, T., Bista, B.B., Takata, T.: A user authenticaion scheme using multiple passphrases and its arrangement. In: International Symposium on Information Theory and Its Applications (ISITA), pp. 554–559. IEEE (2010)
23. Walpole, R.E.: One- and two-sample tests of hypotheses. In: Probability and Statistics for Engineers and Scientists, 7 edn. Pearson (2001). Chapter 10
24. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords. IEEE Trans. Inf. Forensics Secur. **12**(11), 2776–2791 (2017)