How Do We Create a Fantabulous Password?

Simon S. Woo
Computer Science and Engineering Department
Applied Data Science Department
Sungkyunkwan University, Suwon, Republic of Korea
swoo@skku.edu

ABSTRACT

Although pronounceability can improve password memorability, most existing password generation approaches have not properly integrated the pronounceability of passwords in their designs. In this work, we demonstrate several shortfalls of current pronounceable password generation approaches, and then propose, *ProSemPass*, a new method of generating passwords that are pronounceable and semantically meaningful. In our approach, users supply initial input words and our system improves the pronounceability and meaning of the user-provided words by automatically creating a *portmanteau*. To measure the strength of our approach, we use attacker models, where attackers have complete knowledge of our password generation algorithms. We measure strength in guess numbers and compare those with other existing password generation approaches. Using a large-scale IRB-approved user study with 1,563 Amazon MTurkers over 9 different conditions, our approach achieves a 30% higher recall than those from current pronounceable password approaches, and is stronger than the offline guessing attack limit.

CCS CONCEPTS

• **Security and privacy** → **Social aspects of security and privacy**; **Usability in security and privacy**; **Economics of security and privacy**; **Authentication**.

KEYWORDS

Password, Pronounceable Password, Semantic Password

ACM Reference Format:
Simon S. Woo. 2020. How Do We Create a Fantabulous Password?. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380222

1 INTRODUCTION

Considerable research [19, 34, 41, 43] has shown that users are not good at creating strong textual passwords, or creating passwords in predictable ways. Additionally, people heavily reuse passwords for multiple online accounts [13, 44]. On the other hand, while system-generated passwords are stronger, they are often not easy to remember, and they offer significantly lower usability because they are often based on random strings, thereby forcing users to write them down. Additionally, as shown by Shay et al. [36], the rate of recalling these passwords decreases since they are not memorable to the user. Passphrases are regarded as more secure than passwords because they are longer, yet often users use predictable word patterns and common phrases to make passphrases memorable. In turn, this significantly lowers security as shown by numerous researchers [6, 24, 34, 36, 48]. In spite of the aforementioned problems with passwords and frequent password breaches, including Yahoo's one-billion password breach [33], Bonneau et al. [5] show that we do not have better alternatives than the textual password.

To address these challenges, password managers are becoming more popular to prevent password reuse and to generate strong random passwords automatically for users' multiple different accounts. However, users have to create at least one strong and memorable master password for a password manager. Therefore, users cannot be completely free from not creating textual passwords. Our primary objective in this paper is to make such textual passwords more memorable and secure by generating both semantically meaningful and pronounceable passwords from initial user-supplied input words.

Prior research by White et al. [47] suggested that pronounceable authentication strings can offer a promising alternative to traditional passwords. Also, Shay et al. in [36] showed that pronounceable passwords generated from an automated password generator (APG) [31] performed significantly better than other approaches. Zviran and Haga [49] found that system-assigned passwords were easier to memorize if they were pronounceable. Hence, a pronounceable password is shown to be promising for increasing usability and memorability. However, there are critical design issues with current pronounceable password approaches. One of the major problems is that automated pronounceable password generators produce passwords that are weak and difficult to pronounce [9, 25]. In addition, none of the previous pronounceable password generation methods [9, 17, 31] are capable of accepting user inputs, while we strongly believe that pronounceable passwords can be much more memorable if users provide their own inputs. If a system generates a pronounceable password, it will still be difficult for users to remember, since there are no personal associations that users can make.

In order to overcome these and other limitations and weaknesses of past pronounceable password generation methods, we propose a new password generation system, called *ProSemPass*, that can automatically generate both pronounceable and meaningful passwords for users. We implement and evaluate a new pronounceable password generation approach, in which a user and a system coproduce a pronounceable, meaningful, and strong password. Unlike prior approaches that depend on structured rules or heuristics for generating pronounceable strings, we employ the statistical data-driven approach to enhance pronounceability and meaning of output passwords leveraging *portmanteaus*—new words that fuse both the sounds and meanings of their component words. We preserve both

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380222

the meanings (semantic element) and sounds (phonetic element) of arbitrary input words to generate a new word. Although this pioneering idea was first proposed by White et al. [47] for lexically blending two different English words, (e.g., fantastic + fabulous, to create a new word, (e.g., fantabulous), we are the first to provide a constructive method and practical implementation to generate portmanteau-based passwords and demonstrate the effectiveness through a large-scale user study.

In order to make a generated portmanteau, e.g., fantabulous, even stronger, we add uniform noise to an output by randomly assigning an uppercase letter or a digit/symbol at a non-predictable character location, such as producing "*fAntabulou1s*." In addition, we extend the length of the generated portmanteau to improve its strength. One might speculate that a portmanteau would be similar to a multi-word passphrase. However, ProSemPass is not a simple multi-word passphrase, since we produce a new word. Also, our current research focus is to improve the current pronounceable password systems, not passphrases. But to compare our approach with passphrases, we compare the recall and strength of our approach with two-word passphrases generated by a user or a system using `pwgen` [2].

We conducted a comprehensive IRB-approved user study with 1,563 Amazon MTurkers and validate and demonstrate the several advantages of our approach, improving the successful recall by 30% and 10% compared to that of current system-generated pronounceable generation methods and the 3class8 policy (3c8OP), where users create a password with a minimum of 8 characters, and choose at least 3 classes from lowercase-letters, uppercase-letters, symbols, or digits in the 3class8 policy. For strength, instead of using a leaked password dataset for training a statistical guessing algorithm, we generate new training sets to attack our approaches, where an attacker has complete knowledge and access to our portmanteau generation algorithms. Our approaches are stronger against guessing attacks than the compared systems and are able to withstand offline attacks, based on the statistical guessing limit of 10^{14} guesses [14]. The key contributions of this paper are summarized as follows:

- We propose a novel, pronounceable, and meaningful password generation system, *ProSemPass*, constructed from portmanteaus, which can take arbitrary user inputs.
- We conducted a large-scale user study to measure the recall, demonstrating that our passwords are highly memorable compared to user-chosen 3class8 passwords and all other system-generated pronounceable passwords.
- We measured the strength of the passwords using a Monte-Carlo method [10] with a password generation method-aware attacker, who has access to our algorithms, and showed that our approach can perform above the offline guessing attack limit.

2 RELATED WORK

A pronounceable password was proposed by Gasser in [18] and later standardized by NIST (FIPS-181) [31], as an automated password generator (APG) standard. The APG generates a password by forming pronounceable syllables and concatenating them to form a word. But, pronounceability rules are hard-coded in a table and rules are used to determine whether a given character is legal or illegal, based on its position within the syllable and adjacent units. Hence, most rules and checks are syllable-oriented and do not depend on anything outside the current syllable. According to Ganesan and Davies [17], the APG produces a non-uniform distribution which significantly reduces the password search space. Crawford and Aycock [9] proposed the syllable-based algorithm that improves Gasser's approach. Their approach produces a better quality of pronounceable words. However, memorability and security of their approach was not evaluated and validated.

Leonhard and Venkatakrishnan [25] proposed a pronounceable password generation method based on randomly selecting letters to form strings under the following two simple constraints: (1) passwords may not begin or end with two consonants, and (2) passwords cannot contain three consecutive consonants or vowels. Their approach is based on templates, and achieves more uniform distribution than that of Gasser [18]. However, their reported successful recall rate was less than 17%, which may not be memorable enough to be used as a password. In most Unix systems, the `pwgen` [2] program is widely deployed for generating pronounceable passwords. However, pwgen employs a simple rule to randomly select words from the Unix English dictionary file, which merely concatenates dictionary words, and adds a symbol or digit in-between words such as "censor4Errant" and "motive%fiance."

White et al. [47] proposed the use of a portmanteau, a pronounceable authentication string, as an alternative to traditional passwords. White et al. [47] performed a comparative pronounceability study on APG and GPW [38], focusing on phonemes but not the semantics of input words. Although they hypothesized the potential benefits of pronounceable passwords with respect to memorability and security, they neither provided a real system to generate pronounceable passwords, nor conducted a user study to validate their hypotheses. On the other hand, the trigraph-based GPW [38] uses the relative frequencies in English of three-letter sequences representing distinct sounds and produces more pronounceable than APG according to preliminary evaluation in [47]. However, generated password only focuses on phoneme/sounds and does not preserve the meaning of input words. All of these previously proposed pronounceable passwords are generated by the systems. However, we believe system-generation significantly lowers users' recall and curtails the benefit of pronounceability. In this work, we propose a new hybrid password generation approach, which allow a user and a system to co-produce a pronounceable, usable, and secure password. Users provide their initial input words and our system improves pronounceability based on portmanteau construction.

Portmanteaus are new words that fuse not only the sounds but also meanings of their component words. They have been considered in advertising, social media, domain names, and newspapers. Some, like "frenemy" (friend + enemy), "brunch" (breakfast + lunch), and "smog" (smoke + fog), express such unique concepts, and they are highly pronounceable and memorable. While generating a portmanteau appears to be trivial to humans, it is actually a combination of complex and challenging natural language processing tasks [11, 32, 37]. First, a generation system has to identify components words that are both semantically and phonetically compatible. Second, a system has to blend those words into the final portmanteau. Because of the complexity of selecting the best component word and blending, previous approaches had several limitations. For example, the Nehovah system by Smith et al. [37] combines words only at

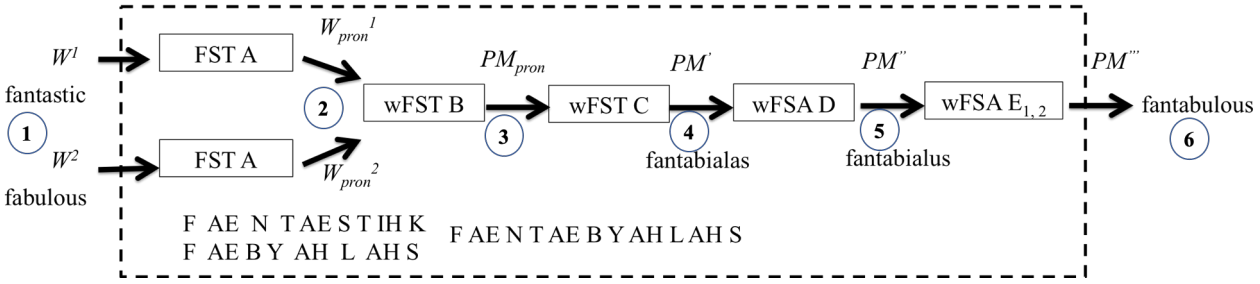


Figure 1: PMGen(2,1) Baseline model: two input words ("fantastic", "fabulous") and one output ("fantabulous") PM generation

exact grapheme matches. Hence, their system cannot generate more complex phonetic blends like "frenemy" or "brunch." The approach by Ozbal and Strappavara [32] blends words phonetically and allows inexact matches. However, it relies on encoded human knowledge, such as sets of similar phonemes and semantically related words. Both systems are rule-based, rather than data-driven. Furthermore, they do not train or evaluate their systems with real-world portmanteaus.

On the other hand, Deri and Knight's approach [11] takes a data-driven and statistical modeling approach to automatically generate a portmanteau, using training examples to learn weights in a cascade of finite state machines. Then, it blends two given words into a portmanteau. In this work, we leverage Deri and Knight's portmanteau generation implementation and extend the system to generate a pronounceable portmanteau password. To further improve strength of portmanteau-based passwords without sacrificing usability, we adopt the similar strategy by Forget et al. [15], where the authors explored the use of persuasive technology by placing randomly chosen characters at random positions into the users' passwords. Similarly, we add random characters at random locations after the generated portmanteau to increase security without lowering usability, as the benefit of this approach was demonstrated by Forget et al. [15].

3 OUR PRONOUNCEABLE PASSWORD DESIGN

In order to create a pronounceable and semantically meaningful password, a user first provides his/her own input words. Next, we generate portmanteaus from user provided input words. Then, we further refine the generated baseline portmanteaus to improve strength. This section provides the details of our password generation algorithms.

Portmanteau (PM) Generation: In our PM generation, finite-state machines (FSMs) are fundamental building blocks to produce pronounceable character sequences. We use the following two types of FSMs: 1) finite state acceptors (FSAs) [20] and 2) finite state transducers (FSTs) [21], where an FSA defines a formal language by defining a set of acceptable strings. And an FST defines relations between sets of strings. Hence, FSAs and FSTs create a character sequence that satisfies specific pronunciation and meaning constraints, and constructs strings that are both pronounceable and meaningful. Often, FSTs and FSAs are cascaded to generate more complex models and improve the natural language generation performance, as shown in Fig. 1. We first define the following terms to explain portmanteau (PM) generation process in Fig. 1:

- W^1 and W^2: 1st and 2nd input component word from a user
- W^1_{pron} and W^2_{pron}: pronunciation of W^1 and W^2
- PM_{pron}: pronunciation of an initial PM generated from n-best lists
- PM', and PM'': intermediate PMs
- PM''': a generated portmanteau
- PPW: final pronounceable password generated for a user

In Fig. 1, we describe the baseline generation model **PMGen(m, n)**, where m is the number of user-provided input words and n is the number of output portmanteaus. In this work, we use m=2 and n=1 and use two input words "fantastic" and "fabulous" as an example to illustrate the process. We use this baseline PMGen(2,1) as a building block to generate more complex and different variations of portmanteau-based passwords.

For convenience, we consider all user inputs and generated PM to be lowercase letters. In Fig. 1, a user first provides two input words W^1 ("fantastic") and W^2 ("fabulous"). Then, *FST A* generates a pronunciation of each word, W^1_{pron} ("F AE N T AE S T IH K") and W^2_{pron} ("F AE B Y AH L AH S") using the CMU Pronouncing Dictionary [45] as shown in Step ② in Fig. 1 and 2. Next, the weighted *FST B* (*wFST B*) translates W^1_{pron} and W^2_{pron} to a combined word, PM_{pron} ("F AE N T AE B Y AH L AH S") in Step ③. Then, *wFST C* spells out PM_{pron} as PM' using aligned graphemes and phonemes based on the CMU Pronunciation Dictionary [16] in Step ④.

After *wFST C* step, the n-best list is used to find PM_{pron} candidates in Step ④, where we used n=1,000. It is possible that this stage

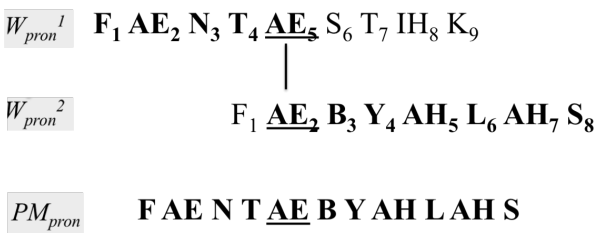
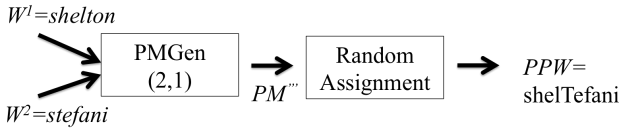


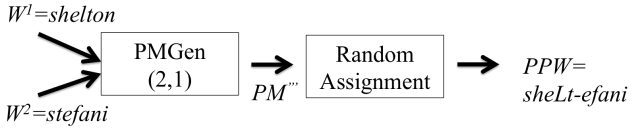
Figure 2: An examples of PM_{pron} generation after *wFST B* in Step ③, where subscripts indicate the step applied to each phoneme for W^1_{pron} and W^2_{pron} in Fig. 1

can be probabilistically configured to generate a non-deterministic PM_{pron} for the same input words to increase search space and security from an attack. Next, *wFSA D* is used to further improve the PM', where it is trained on a language model to enhance pronounceability. After *wFSA D* (Step ⑤), the final portmanteau ("fantabulous"), PM''', is generated from the PMGen(2,1) baseline model in Step ⑥.

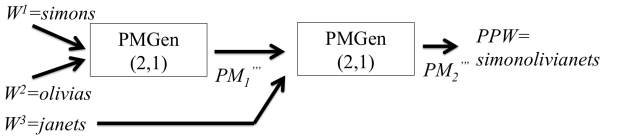
Improving Baseline PMGen(2,1) model to generate ProSemPass: Intuitively, a portmanteau would be difficult to attack using a dictionary or rainbow table, because it is a new word and does not exist in a common dictionary. However, if input words have a small number of characters, e.g., "*motor*" + "*hotel*", then the resulting portmanteau will be small as well, e.g., "*motel*". But, it is not a suitable password because passwords with a length less than 8 characters can be easily cracked by a brute-force attack [27]. Hence, input words have to be long enough to output a long *PM* which can withstand a brute-force attack. To address this issue, we constrain all user input words to a minimum of 6 characters in length and the output to be a minimum of 8 characters from the baseline PMGen(2,1) model in Fig. 1.



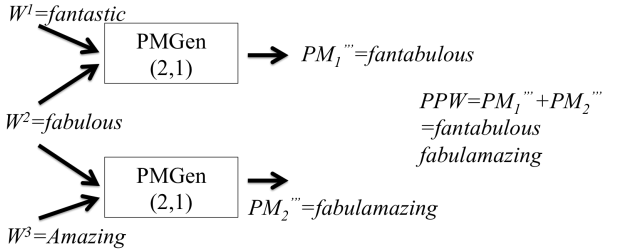
(a) **ProSemPass(2,1) model:** Two user-supplied input words and one output word with either randomly capitalizing one of output characters, or adding uppercase, digit, or symbol (2class8).



(b) **ProSemPass(2,1,S) model:** Two user-supplied words and one output word with two of the following: random uppercase, digit, or symbol (3class8).



(c) **ProSemPass(3,1) model:** Three user-supplied input words and one output word (1class8).



(d) **ProSemPass(3,2) model:** Three user-supplied input words and two output words (1class8).

Figure 3: Four different systems to generate pronounceable passwords.

In order to further improve strength, we can extend the length of an output password, or add additional randomness such as symbols, digits, or uppercase letters to the baseline output from the PMGen(2,1) model in Fig. 1. We propose four different approaches to produce different variations of portmanteau-based pronounceable and semantically meaning passwords, ProSemPasses, and evaluate their memorability and strength. In the first approach, in Fig. 3(a) we randomly capitalize one character from the baseline model, producing a 2class8 password. We denote this model as **ProSemPass(2,1)**. We implement the **ProSemPass(2,1)** system by extending the baseline model, as shown in Fig. 3(a). For example, using input words $word_1$ = *shelton* and $word_2$ = *stefani*, *shelTefani* is generated from this model, as also shown in Table 1.

In the second model, we further improve strength by randomly adding a digit or a symbol to the ProSemPass(2,1) model, as shown in Fig. 3(b) which produces a 3class8 password. We denote this model as **ProSemPass(2,1,S)**. Hence, we can generate a password such as *sheLt-efani* as shown in Table 1.

In the third model in Fig. 3(c), users supply an arbitrary three input words of their choice. Hence, this system enables an output password to be longer, producing the PM from three input words (denoted as **ProSemPass(3,1)**), as shown in Fig 3(c). The system takes the first two input words, and the baseline model generates PM_1. Then it generates PM_2 from combining PM_1 and third input word W^3. For example, input words *simons* + *olivias* + *janets* outputs a password, *simolivianets*, as shown in Fig 3(c). However, even after combining three words, it is still a 1class8 password.

The last model we consider to harden strength is **ProSemPass(3,2)**, as shown in Fig. 3(d). In this generation system, the first portmanteau PM_1 is generated from the first two input words, W^1 and W^2. The second PM_2 is generated from the second and third words, W^2 and W^3, respectively. Then, the final output password is two portmanteaus-generated PMs separated by a space. For example, given *fantastic*, *fabulous*, and *amazing* as a first, second, and third word, respectively. **ProSemPass(3,2)** produces the final password *fantabulous fabulamazing* as shown in Fig 3(d). Hence, **ProSemPass(3,2)** generates the longest output password, but it is still a 1class8 password.

The examples of all the pronounceable models we consider in this work are summarized in Table 1. In particular, our approaches are denoted as *hybrid* in column 6 in Table 1, as user and systems jointly produce passwords.

4 DESCRIPTION OF PASSWORD GENERATION METHODS AND CONDITIONS

In this section we describe nine different pronounceable password generation methods that we considered in this paper. First, current system-generated pronounceable password generation methods are compared with our approaches. It can be argued that it is unfair to directly compare our hybrid approach with the system-generated passwords. However, to the best of our knowledge, there has been no practical pronounceable password generation systems which rely on using user inputs. Additionally, the focus of our paper is to

Table 1: Pronounceable password models considered in our work.

Approach	Num. of Input Words	Num. of Classes	Examples of Inputs and Resulting Output Passwords	Words from Dict.	Generation Type
Our 2-in/1-out pron. sem. pass. (**ProSemPass(2,1)**)	2	2	shelton + stefani ⇒ shelTefani	No	Hybrid
Our 2-in/1-out secure pron. sem. pass. (**ProSemPass(2,1,S)**)	2	3	shelton + stefani ⇒ sheLt-efani	No	Hybrid
Our 3-in/1-out pron. sem. pass. (**ProSemPass(3,1)**)	3	1	simons + olivias + janets ⇒ simolivianets	No	Hybrid
Our 3-in/2-out pron. sem. pass. (**ProSemPass(3,2)**)	3	1	fantastic + fabulous + amazing ⇒ fantabulous fabulamazing	No	Hybrid
Linux gen. pron. pass (**S-ProPass**) [2]	2	3	censor4Errant	Yes	System
Gasser's length-8 pron. pass. (**GPW8**) [18]	0	1	netrigod	No	System
Gasser's length-12 pron. pass. (**GPW12**) [18]	0	1	isslartustle	No	System
Ordinary 3class8 password (**3c8OP**)	–	3	Pa$$word123!	–	User
User chosen 2-in/1-out pron. secure pass (**U-ProPass**)	2	2+	forever3love, roron@verona	Yes/No	User

objectively evaluate the existing pronounceable password generation systems and propose an improved approach.

To balance this issue, we compare our hybrid approach with two other user-created password approaches to examine the password recall and strength performance between hybrid and user-created passwords—thereby giving us the ability to analyze how well our hybrid approach performs against current widely used password generation methods, as well as system-generated approaches. Based on the system and user input made during the password creation process, we classify each password generation method into one of the following groups: 1) system-generated, 2) hybrid (user + system), and 3) user-generated.

4.1 System-Generated Pronounceable Passwords

Linux generated Pronounceable Password (S-ProPass): The pwgen [2] program is widely deployed in various Linux systems. A user can configure the strength of a password by supplying the number of bits in entropy and a system randomly selecting words from a dictionary file available in Linux OS (i.e., `/usr/share/dict/`). Then, pwgen randomly inserts either a digit or special character in-between the arbitrarily chosen words. Examples of generated pronounceable passwords are as follows: *"censor4Errant"* and *"motive+fiance"*. Therefore, this approach can be considered as a two-random word passphrase with an extra character between the words. In this work, we use the pwgen implementation found in Ubuntu 14.04.5 LTS, which is based on 99K American English words located in the /usr/share/dict/american-english dictionary file. We denote this pwgen model as the Linux system generated pronounceable password (**S-ProPass**) approach.

GPW: The GPW [38] is the program to generate pronounceable passwords developed by Morrie Gasser [18], which uses relative frequencies of three-letter sequences in English representing distinct sounds and producing pronounceable strings. GPW is often referred to as a trigraph-based approach. FIPS Standard 181, "Automatic Password Generator" (APG) [31], is a similar digraph-based generator derived from Gasser's. However, according to the preliminary evaluation by White et al. in [47], GPW produces more pronounceable strings than APG. Hence, we do not consider APG and focus only on GPW for our comparison. Specifically, we use the GPW JavaScript program developed by Van Vleck [38] that can generate a specific number of characters in a password. For our purpose, we fix the password length to be either 8 or 12 characters, since 8-characters is the minimum length of most of the password policies and 12-characters is the typical length that more stringent password policy requires, as described in other research [23, 27]. We define a length of 8-chars GPW, and 12-chars GPW as **GPW8** and **GPW12**, respectively. For example, *"enibleydrisk"* and *"leinklypoins"* are generated from GPW12, where three-letter sequences produce distinct sounds. Although GPW produces strings that are more pronounceable and less random, there is no semantic meaning associated with a generated password.

All of the existing pronounceable passwords cannot take user inputs and are all system-generated. We believe the system-generated approach is one of the most significant bottlenecks, and it should be improved. In our work, we compare and examine our hybrid approach with the aforementioned system-generated pronounceable passwords. In addition to system-generated approaches, we introduce two other user-created password conditions to compare how current user-created passwords can perform better or worse than our proposed hybrid generation approach with respect to strength and recall performance.

4.2 User-Generated Passwords

3class8 ordinary passwords (3c8OP): We consider the popular 3class8 policy. In 3c8OP policy, users create a password with a minimum of 8 characters, and choose at least 3 classes from lowercase-letters, uppercase-letters, symbols, or digits.

User-chosen two words and a symbol/digit (U-ProPass): We ask users to create a password based on their two input words and a symbol or digit. This produces the same kind of passwords as S-ProPass or GPWs, but they are entirely created by users. This condition directly allows gauging how user-supplied inputs can improve recall compared to the system-generated approaches. Also, we can compare how the performance of our hybrid (user+system) approach is comparable to passwords entirely generated by users.

4.3 Human and System-Generated Pronounceable Passwords

We consider four different variants of portmanteau-based passwords, which were depicted in Fig. 3. In **ProSemPass(2,1)**, a user provides two arbitrary input words, and the system generates a 2class8 (min. 8 chars with 2 different character classes) portmanteau-based password by adding a random character at random locations. In **ProSemPass(2,1,S)**, with the two input words from a user, it produces a

3class8 password by adding two of a random uppercase, digit, or symbol. In the **ProSemPass(3,1)** and **ProSemPass(3,2)** approaches, a user provides three input words. However, the system produces one 1class8 portmanteau for the ProSemPass(3,1) approach, while two 1class8 portmanteaus are generated in the ProSemPass(3,2) approach. We summarize the above approaches and provide an example produced from each approach in Table 1.

5 EVALUATION SETTINGS

We consider the *password generation method-aware attacker* who is an adversary with complete knowledge of how each target password is created under the specific password creation algorithm. This is a much stronger attacker model than a generic attacker, who would blindly crack passwords using a leaked passwords dataset, which has been extensively used in other research [4, 8, 12, 22, 23, 26, 27, 40, 42, 46]. We assume that a password generation method-aware attacker has a complete access to all the password generation algorithms in Table 1. Once an attacker has knowledge about the specific target password model and type, he can easily exploit popular lists to generate input candidate words that are more frequently used, as shown by much other research [4, 7, 34]. Then, he can utilize those popular wordlists to generate passwords using ProSemPasses, GPWs, S-ProPass, 3c8OP, and U-ProPass approaches. For example, to attack ProSemPass(2,1), the password generation method-aware attacker only uses passwords created from the ProSemPass(2,1) algorithm to train a guessing algorithm to optimize the attack. Therefore, this attacker model can significantly improve its guessing efficiency, instead of using popular leaked passwords.

5.1 Measuring Password Strength

For estimating the strength of a password, we use a *guess number* to emulate a number of attempts that an attacker would need in order to guess a password [4, 22, 42]. This allows evaluating the effectiveness of password-guessing attacks much more quickly than using existing cracking techniques. One of the most effective approaches is a probabilistic method, based on emulating an attack trained by a given dataset, [4, 22, 42], that produces a guess number. For generating a guess number, we use the Monte-Carlo sampling implementation by Dell'Amico and Filippone [10], which estimates the number of guesses until success using the sampling method over a probabilistic password model. They have shown the accuracy of such estimated strength against state-of-the-art attacks. We tried an $n-gram$ model, where n is 2, 3, and 4, where the n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n–1)th order Markov model.

Also, we employed the probabilistic context-free grammar (PCFG), and back-off model to estimate a guess number for passwords generated from each model. In particular, these two methods are shown to be effective for cracking certain types of passwords, which have underlying grammar structures [10, 46].

5.2 Attack Corpus and Training Statistical Guessing Algorithm

We chose the Google20K wordlist [1], which includes the most frequently used 20,000 English words ranked in order of frequency. Google20K includes the most popular ranked words used in the Internet and is much more efficient than larger corpus such as WordNet [29], where WordNet cannot easily capture the popularity of the words. Therefore, we aim to use Google20K as a popular word list to generate candidate passwords to train the Monte-Carlo guessing algorithm.

Using Google20K, we create two- to three-word combinations according to their frequency and input them to each ProSemPass algorithm. We then generate 100K training passwords per each password model, where the 100K training passwords are shown to reliably estimate strength [10] without minimizing accuracy and performance. Then, we use the produced 100K candidate passwords per each password model to separately train the Monte-Carlo statistical guessing algorithms. Therefore, the *password generation method-aware attacker* uses 100K ProSemPass(3,1) training passwords to crack the target ProSemPass(3,1) passwords, while not using passwords generated from any other approaches to perform a targeted attack. For attacking 3class8 passwords, we used the 21 million leaked password dataset from RockYou, LinkedIn, and eHarmony, where an attacker can easily have access to those leaked passwords that have been extensively used in other research [4, 8, 12, 22, 23, 26, 27, 40, 42, 46]. Similarly, in attacking U-ProPass, we use the same Google20K to create the 100K candidate passwords for training the statistical guessing algorithm.

6 USER STUDY DESIGN

All of the user studies were reviewed and approved by our Institutional Review Board (IRB). In our study, we assigned MTurk participants at random to only one of the 9 password conditions we described in Section 4, where those 9 conditions are 4 hybrid ProSemPass approaches with differnt inputs/outputs, 3 system-generated approaches (S-ProPass, GPW8, and GPW12), and 2 user-created approaches (3c8OP and U-ProPass).

We recruited participants with at least 1,000 completed human intelligence tasks (HITs) and $>$ 95% HIT acceptance rate. We asked each participant to create one password in one sitting. We invited participants to return and to authenticate after two days from creation. Since a two-day recall time interval has been predominantly used for most prior password recall research [22, 36, 39], we chose to use the two-day recall to be comparable to other previous research results. At the end of the authentication, we provided a short survey to assess user evaluation of each approach. We paid $1.00 for pronounceable password creation and $2.00 and $2.50 for the first and second authentication task, respectively.

Password Creation. All users were asked not to write down or copy their answers, and to only rely on memory. In our hybrid approaches, users were asked to enter their choice of two to three input component words. We automatically checked input of 6 characters for each input word during ProSemPass creation. If less than 6 characters, users were asked to enter different words. Then, our system generated a pronounceable password and presented it to a user. Right after the creation of a password, we asked users to practice their passwords twice in the same sitting. If practice authentication failed, we asked the user to recreate his/her password.

For other conditions, each respective system generated a password for a user when the user entered the study. In GPW8, we fixed the number of characters to be 8 to meet the minimum password length

requirement. Also, we generated 12 characters for GPW12 to test the longer password policy similar to other research [23, 27]. In the S-ProPass approach, we randomly generated a password containing 2 words and 1 symbol, in order to be comparable to ProSemPass approaches. For 3c8OP, we asked users to create their own ordinary 3class8 password. In the U-ProPass condition, we asked users to create a password of at least 8 characters using their choice of two words and a digit or symbol.

Authentication. Each user was asked to authenticate two days after creation. We allowed at most five trials to authenticate per password. All users were asked not to paste their answers. We automated the detection of copy and paste attempts in our forms and rejected users who performed either of these two actions. Further, we displayed a notice to participants at both the creation and authentication screens, assuring them that they would receive payment regardless of their authentication success. This ensured that participants had no monetary incentive to cheat. After the authentication visit, we asked participants to complete a short survey to assess their sentiments and preferences for each approach.

Statistical Tests. To analyze the statistical significance across different approaches, we performed an omnibus test, χ^2 (Pearson's Chi-squared test with Yates' continuity correction), on categorical data with $p = 0.05$. For quantitative data, we used a Kruskal-Wallis (KW) test, which does not assume normality for omnibus tests. If a KW test showed statistical significance, we used a Mann-Whitney U (MWU) test for pairwise comparisons. If χ^2 showed significance, we performed the pairwise Fisher's Exact Test (FET), which yields more accurate confidence for relatively smaller sample size. For multiple-comparison correction, we used a Holm-Bonferroni (HC) method.

7 RESULTS

In this section we present the results of our user study. We first report the demographic information, number of participants, statistics on the user-created passwords, and the recall performance for each authentication task. We then present the strength in guess numbers for each approach. We found that our approaches (ProSemPasses) are on average 30% more memorable than current system-generated pronounceable passwords (GPWs and S-ProPass), and have a 10% higher recall rate than that of user-created 3class8 passwords. Further, our results are 2% lower than the recall rate from the user-created U-ProPass approach. In addition, we found that both ProSemPass(2,1,S) and ProSemPass(3,2) are more resilient against an offline attack of 10^{14} guesses, while most of the existing pronounceable passwords, as well as the two-word approach from S-ProPass, are very easily guessable from the offline guessing attack. We provide the detailed results below.

7.1 Demographics and Participants Info

A total of 1,563 MTurkers participated in our user study. Among these participants, we detected copy/paste actions from 164 participants (10.49%). After excluding those, 1,399 participants successfully created passwords. Two days after creation, we sent an email asking them to return to perform authentication. Out of 1,399 participants, 932 participants returned—yielding an average return rate of 66.62%. The overall return rate for the authentication task for ProSemPass approaches was 67.65%, and was 64.37% for the other pronounceable approaches. The return rate of the user-created approach was 67.93%.

Table 2: Number of participants who created and authenticated after two days.

Model	Created	Auth after two days
ProSemPass(2,1)	151	103 (68.21%)
ProSemPass(2,1,S)	154	107 (69.48%)
ProSemPass(3,1)	180	119 (66.11%)
ProSemPass(3,2)	158	106 (68.35%)
All ProSemPasses	643	435 (67.65%)
S-ProPass	144	100 (69.44%)
GPW8	160	106 (66.25%)
GPW12	162	94 (58.02%)
All GPWs&S-ProPass	466	300 (64.37%)
3c8OP	145	99 (68.28%)
U-ProPass	145	98 (67.59%)
All 3c8OP&U-ProPass	290	197 (67.93%)
Total	1,399	932 (66.62%)

Among 1,399 participants, 51% and 49% reported their gender as male and female, respectively. Also, 82% reported that their native language was English. The age range between the 25-34 age group was the highest with 54%; the next-highest group was the 35-44 age range, with 23%. We found no statistically significant differences between each approach in the return rate, the reported gender, native language, and age distributions.

7.2 Password Statistics

The average character length of ProSemPass(2,1), ProSemPass(2,1,S), ProSemPass(3,1), and ProSemPass(3,2) are 10.14, 10.91, 10.82, and 19.87 characters, respectively. The U-ProPass approach produced passwords with 10.99 characters, comparable to other ProSemPass approaches. When asked to create a 3class8 password, participants created a similar length of passwords with 10.51 characters on average. Since ProSemPass(3,2) produces two output words, the average length of ProSemPass(3,2) is almost twice as much as those from other ProSemPass approaches. Participants created the second-longest (11.53 characters) password under U-ProPass, which is based on user-provided two-input words.

Although the average length is comparable (around 10 characters) among ProSemPass(3,1), ProSemPass(2,1), ProSemPass(2,1,S), S-ProPass, and 3c8OP, we performed the KW test to examine how underlying password length distribution is similar or different. The KW test rejects the null hypothesis (χ^2_8=841.74, $p = 2.07\times10^{-7}$) and confirms that underlying length distribution is significantly different. Next, we performed the pairwise tests with Holm-Bonferonni-corrected Mann-Whitney U (HC-MWU) on all possible pairs of conditions. In the next section, we will show that, though a password length is comparable, password strength can be significantly different.

7.3 Recall

Recall is successful if users match every character in the password. Table 3 shows the overall successful recall performance after two days. The recall rate for GPW8, GPW12, and S-ProPass are 43.40%, 41.94%, and 44.00%, respectively. The recall rate of 3c8OP and U-ProPass are 64.65%, and 75.51%. All of our ProSemPass approaches achieve higher recall performance over the system-generated pronounceable GPWs and S-ProPass approaches. ProSemPass(2,1), ProSemPass(2,1,S), and ProSemPass(3,1) achieve high recall rates with 74.76%, 74.77%, and 73.95%, respectively. The recall rate of ProSemPass(3,2) is slightly lower than 70% with 69.81%.

Table 3: Successful recall rate after two days, where approaches with the recall rate higher than 70% are highlighted.

Approach	Num. of Succ. Participants	Succ. Recall (%)
ProSemPass(2,1)	77	**74.76%**
ProSemPass(2,1,S)	80	**74.77%**
ProSemPass(3,1)	88	**73.95%**
ProSemPass(3,2)	74	69.81%
S-ProPass	44	44.00%
GPW8	46	43.40%
GPW12	39	41.94%
3c8OP	64	64.65%
U-ProPass	74	**75.51%**

The recall rates of system-generated approaches are very low (41%–44%). On the other hand, the recall rate of the widely used 3class8 was only 64.65%, even though users provide entire inputs. Hence, ProSemPass(2,1), ProSemPass(2,1,S), and ProSemPass(3,1) were recalled 10% more than 3class8, and were 30% higher than the system-generated approaches. The U-ProPass approach in which users provided their own two words had the highest recall rate among all, with 75.51%, which is within 1% higher than ProSemPass(2,1,S).

In order to better understand the statistical difference in recall, we first performed an omnibus test χ^2 and found significance (χ^2_8=75.44, $p = 4.028\times10^{-13}$). We mainly present the subsets of HC-FET pairwise tests involving FSTs with statistical significance. All ProSemPass approaches show strong statistical significance over GPWs and S-ProPass. ProSemPass(2,1) shows statistical significance from pairwise tests over S-ProPass, GPW8, and GPW12, respectively. Similarly, all ProSemPass(2,1,S), ProSemPass(3,2), and ProSemPass(3,2) show statistical significance with S-ProPass, GPW8, and GPW12. Hence, ProSemPass performed better than current system-generated pronounceable passwords, and this is statistically important.

On the other hand, we did not find statistical significance between U-ProPass and all ProSemPasses. Therefore, this shows that our hybrid ProSemPass approaches, where humans and systems work together, do not statistically perform worse than the user-created approach.

7.4 Password Strength

We present the strength of each approach using the *password generation method-aware* attacker model. We report the results in Tables 4.

Table 4: Median guess number measured using $n-gram$, and back-off model with the password generation method-aware attacker using Google20K corpus.

Approach	2-*gram*	3-*gram*	4-*gram*	Back-off
ProSemPass(2,1)	2.43×10^{13}	9.07×10^{14}	1.71×10^{44}	5.37×10^{13}
ProSemPass(2,1,S)	$\mathbf{1.73\times10^{14}}$	$\mathbf{1.62\times10^{19}}$	$\mathbf{2.05\times10^{41}}$	$\mathbf{8.44\times10^{15}}$
ProSemPass(3,2)	1.17×10^{23}	1.58×10^{23}	8.88×10^{87}	1.44×10^{25}
ProSemPass(3,1)	2.22×10^{11}	1.83×10^{12}	9.14×10^{76}	2.18×10^{12}
S-ProPass	1.51×10^{13}	3.27×10^{11}	2.03×10^{11}	1.08×10^{9}
GPW8	6.61×10^{8}	9.19×10^{7}	9.96×10^{7}	4.02×10^{8}
GPW12	7.04×10^{12}	5.50×10^{11}	6.41×10^{11}	5.20×10^{12}
3c8OP	8.00×10^{14}	9.27×10^{13}	1.54×10^{14}	1.43×10^{14}
U-ProPass	1.43×10^{15}	2.43×10^{14}	2.36×10^{13}	3.56×10^{13}

And Fig. 4 provides the detailed distributions of the strength of password models with the back-off model, where the X-axis is the log10 of the number of guesses, and the Y-axis is the percentage of passwords being guessed. We used Google20K for training guessing algorithms for all ProSemPasses and U-ProPass. For GPWs and S-ProPass, we generated 100K passwords from each password generation algorithm for training guessing algorithms. We only report the guess numbers up to 10^{40} and a figure for the back-off model.

As demonstrated, ProSemPass(3,2) achieves the highest strength in guess numbers across all approaches. The second-highest approach is ProSemPass(2,1,S). The strength of both approaches is greater than that of the offline cracking limit of 10^{14} [14]. Although users provided 3 input words for the ProSemPass(3,1) approach, ProSemPass(3,1) is the weakest among all ProSemPass approaches, due to the simple 1class8 structure. Hence, overall ProSemPass(2,1) is stronger than ProSemPass(3,1) because of the more complex 2class8 password composition structures.

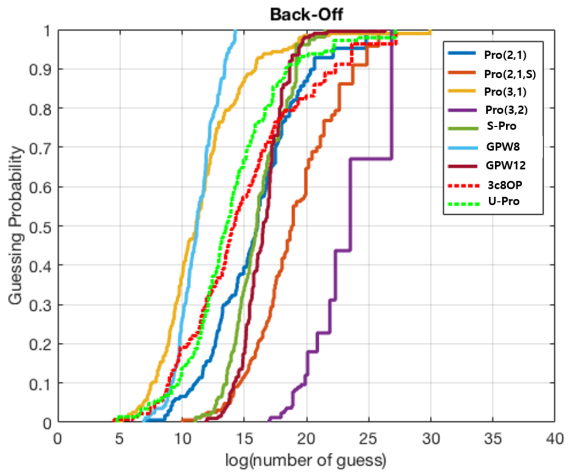


Figure 4: Guess numbers and guessing probability measured using back-off model with the password generation method-aware attacker using Google20K.

On the other hand, the median strength of 3c8OP is around 9.27×10^{13} for the 3-gram model. Also, another user-created condition, U-ProPass, is 2.36×10^{13} using the 4-gram model. Hence,

both user-created approaches are below the offline guessing attack strength. The strength of GPW8 is the weakest and the second weakest approach was S-ProPass. Although GPW12 is the strongest among existing pronounceable password approaches, it is still lower than the strength of ProSemPass(2,1). Also, we observe that the 2-gram based attack was the most effective against our approaches, since our passwords are generated from dictionary input words. Hence, one might suspect that rule-based substitutions from dictionary words can be effective. In fact, the S-ProPass approach is the most vulnerable to predictable rules and word substitutions, since it uses words directly from the dictionary with a simple concatenation rule.

7.5 Measuring similarity between user-provided input words

Can an attacker exploit correlations of user provided input words and use this correlation knowledge to attack? That is, do users tend to select two input words from the similar category or related words such as *Brad* and *Angelina*, and *orange* and *apple*? Or, would users choose seemingly unrelated words such as *motor* and *hotel*? We hypothesize that users might choose highly correlated words, which differ from random words chosen by S-ProPass. It would then be easier for attackers to crack. Hence, we aim to measure semantic correlations/distances between users' input word choices in the ProSemPass approaches.

For measuring this semantic distance or similarity (correlation) between words, we use pre-trained **word2vec** [28] which provides state-of-the-art word embeddings, i.e., distributed representations of words and phrases. The word2vec identifies not only similar words that tend to be close to each other, but also words that can have multiple degrees of similarity. The pre-trained word2vec model includes vectors for a vocabulary of 3 million words and phrases that were trained on roughly 100 billion words from a Google News dataset. The semantic distance is measured between -1 and 1, where -1 is negatively related and 1 is positively related. When similarity between two words is close to 0, then two words are not related. We measure the average and standard deviation (STD) of absolute value of similarity value returned from word2vec. We take the absolute values of similarity outputs in order not to cancel out the positive and negative similarity.

The measured average semantic distance is shown in Table 5. For ProSemPass(3,2) and ProSemPass(3,1), the first value is the average semantic distance between the first and the second input word from a user. The second value is the average between the second and the third input word, and the last value is the average between the third and the first input words. The standard deviations (STDs) are calculated in the same way.

In Table 5, S-ProPass has the lowest semantic correlations between input words as they are randomly selected, yielding the average correlation of 0.0787. The U-ProPass has the highest, with 0.167 on average. The averages of all ProSemPass approaches are less than 0.15. Where, for example, semantic distance between "France" and "Paris" yields 0.63, we observe that the average correlation is quite low in our approach. But it is greater than the correlation of the random words. In addition, the empirical cumulative distribution function (ECDF) of the semantic distance of each approach

Table 5: Semantic distance among user's input words using word2vec.

Approach	Avg. Semantic Dist.	STD
ProSemPass(2,1)	0.146	0.189
ProSemPass(2,1,S)	0.145	0.175
ProSemPass(3,2)	(0.162, 0.147, 0.135)	(0.183, 0.162, 0.178)
ProSemPass(3,1)	(0.131, 0.143, 0.124)	(0.175, 0.179, 0.163)
S-ProPass	0.079	0.068
U-ProPass	0.167	0.190

is shown in Fig. 5. As shown, 80% of user-provided inputs have a correlation measure of less than 0.4, which is promising, and more than half are less than 0.2, as shown in Fig. 5. In addition, we performed the statistical analysis to evaluate whether the differences are significant or not. We find that there are no statistical differences for ProSemPass(2,1) vs. S-ProPass (U=8667, HC-corrected p-value=0.282) Therefore, we conclude that attackers would not benefit from guessing words in a similar category of words to reduce the search space.

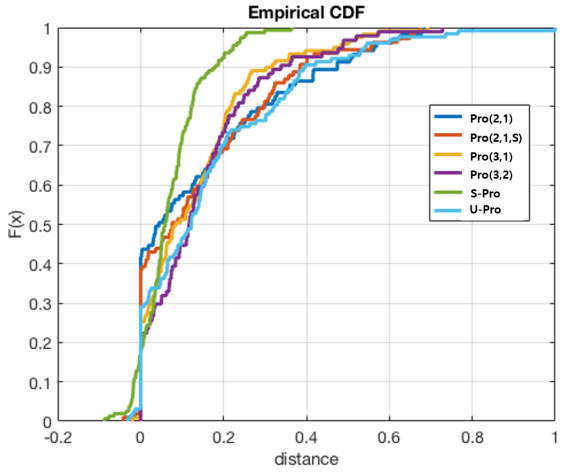


Figure 5: Empirical cumulative distribution function (ECDF) of semantic distance.

7.6 User Sentiments

At the end of the authentication, 901 participants completed the survey. There was no statistical significance among participants who completed the survey at the end for particular approaches. They were asked to rate their agreement with the following statement, "I like this password generation method" on a Likert-scale, from 1 (strongly disagree) to 10 (strongly agree) with 5 (neutral).

We find that ProSemPass(2,1) has the highest average user preference score with average around 7.55. The second preferred approach is ProSemPass(3,1) with 7.35. The third highest is U-ProPass, yielding the average score of 7.01. However, ProSemPass(3,2) is the least favored but slightly better than 3c8OP. One participant commented that "It is very hard to remember the three words." GPW12 has the lowest rating with 4.39. User's preference on 3c8OP (6.41)

was higher than S-ProPass (5.67) and GPW8 (5.35), but it is lower than all of ProSemPass approaches. Overall, most of our approaches was rated higher by users than current approaches including 3c8OP (6.41).

8 DISCUSSIONS, LIMITATIONS, AND FUTURE WORK

Recall: Our user study results confirmed that meaningful and pronounceable passwords achieve a higher recall through portmanteaus. We believe the higher recall in our approaches is attributed to both more pronounceable, and meaningful strings generated for users. On the other hand, users had difficulty in remembering passwords generated from GPW8 and GPW12, since they are not easy to pronounce and have no meanings in them. One participant remarked on GPW8, "It was difficult to pronounce so I am not sure if I remembered it correctly." Also, even though words are from a dictionary, typical users might have difficulty in associating personal associations to randomly generated S-ProPass passwords. One participant mentioned that "It's hard to memorize words that have no meaning; but that makes it a strong password." This is consistent with the finding of Shay et al. [36]. Also, the lower recall performance of 3c8OP demonstrates that a user is not good at creating a new memorable 3c8OP, when asked to create one for a new account. In addition, we were surprised that the recall rate of ProSemPass(2,1,S) is comparable to ProSemPass(2,1). While adding a random extra symbol can increase strength by 10^2 times in guess numbers, users did not have much difficulty in remembering this extra information. Therefore, it is much better to generate passwords with ProSemPass(2,1,S), which provides the comparable recall to ProSemPass(2,1) but generates much stronger passwords.

Strength: With the password generation method-aware attacker, ProSemPass(2,1,S) and ProSemPass(3,2) are still stronger under this attacker model with passwords generated with the inputs from a large text corpus. S-ProPass is the 2nd weakest model, since it is based on a simple rule to concatenate a sequence of dictionary words and characters in a predictable way. This implies that ProSemPass approaches provide a clear benefit over a two-word passphrase in strength. Overall, users liked ProSemPasses (except ProSemPass(3,2)) and S-ProPass, showing the importance for users to provide memorable inputs. Surprisingly, users did not like creating passwords under the 3class8 policy. We believe that enforcing specific character classes may not be the most effective for humans.

Ecological Validity: It would be ideal to conduct the lab study with a large number of participants. However, it is not easy to recruit more than 1,500 participants and interview them because of limited resources. Several research including marketing, medical, and security domains [3, 30, 35] has investigated the validity of Mturk online survey methods compared to the traditional approaches. They generally agree that online Mturk study design is an acceptable method and can match with traditional survey method results within 10% accuracy [3]. We designed our study to disincentivize cheating. We promised to pay participants in full regardless of authentication success. We also reminded the participants multiple times to rely on their memory only. If any cheating occurred it was likely to affect all the results uniformly. Thus our data can still be used to study the improvement of recall and security between password models. We did not evaluate long-term effects in how users create and recall passwords, which can be captured through a long-term field study. In addition, our study's ecological validity is also limited in other ways. Password creation was the primary task in our user study. Because of the Hawthorne effect, the actual recall rate may vary. In the more typical scenario of password creation being users' secondary task, it is possible that users might have a lower motivation to create strong and memorable passwords.

ProSemPass Algorithms: We detected that 1.2% and 0.51% of words produced from ProSemPass(2,1) and ProSemPass(3,1) approaches matched with dictionary words. Even though they are low percentages, the future portmanteau-based password system should cross-check the outputs with dictionary words. Also, like many other password generation methods, the output depends on users' strong inputs. If users supply weak strings such as "aaaaaa" and "bbbbbb," our resulting portmanteau output is "aaaabbbb." Although we did not find any users who provided such simple inputs, it can be trivially prevented by rule-based filtering. Also, we believe further guidance can help mitigate generating simple portmanteau outputs. We plan to help improve users' common mistakes in the future research. Although we did not explore the *n*-best lists in the system, it is possible to configure our system to produce a different pronounceable password for the same input words. This will also be interesting future work to further improve ProSemPass systems. Our current work does not focus on directly improving passphrases. However, our approach can be adopted within passphrase generation, where users can insert non-dictionary words from ProSemPasses within their passphrases. Then, passphrases can be more secure, while semantically still memorable.

9 CONCLUSIONS

Our proposed system keeps both the security and recall of passwords high—the security of most ProSemPass variants is greater than offline guessing attack limits, and the recall is better than that of 3class8 passwords and that of the current pronounceable passwords. In closing, our system can be a valuable aid to help users create longer, more memorable, secure, and pronounceable passwords. For example, our approach can be used for generating a large number of master or root passwords for a password manager system within corporate settings, where system administrators must manage large server farms. Also, when a user needs to manage many IoT devices with textual passwords, our system can be helpful for creating and managing multiple passwords for different IoT devices.

10 ACKNOWLEDGEMENTS

This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (No. 2019M3F2A1072217), the Basic Science Research Program through the NRF of Korea funded by the Ministry of Science, ICT (No. M2017R1C1B5076474). In addition, this work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421, AI Graduate School Support Program).

REFERENCES

[1] 2015. Google-20K-English Words. https://github.com/first20hours/google-10000-english/. Accessed: 2015-10-14.

[2] B Allbery. 1988. pwgen: random but pronounceable password generator. *USENET posting in comp. sources. misc* (1988).

[3] Frank R Bentley, Nediyana Daskalova, and Brooke White. 2017. Comparing the reliability of Amazon Mechanical Turk and Survey Monkey to traditional market research surveys. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1092–1099.

[4] Joseph Bonneau. 2012. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 538–552.

[5] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2012. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *2012 IEEE Symposium on Security and Privacy*. http://www.jbonneau.com/doc/BHOS12-IEEESP-quest_to_replace_passwords.pdf

[6] Joseph Bonneau and Ekaterina Shutova. 2012. Linguistic Properties of Multi-word Passphrases. In *Financial Cryptography and Data Security*. Springer, 1–12.

[7] Joseph Bonneau and Ekaterina Shutova. 2012. Linguistic properties of multi-word passphrases. In *USEC '12: Workshop on Usable Security*. http://www.jbonneau.com/doc/BS12-USEC-passphrase_linguistics.pdf

[8] Rahul Chatterjee, Joseph Bonneau, Ari Juels, and Thomas Ristenpart. 2015. Cracking-Resistant Password Vaults using Natural Language Encoders. In *2015 IEEE Symposium on Security and Privacy*. http://www.jbonneau.com/doc/CBJR15-IEEESP-cracking_resistant_password_vaults.pdf

[9] Heather Crawford and John Aycock. 2008. Kwyjibo: automatic domain name generation. *Software: Practice and Experience* 38, 14 (2008), 1561–1567.

[10] Matteo Dell'Amico and Maurizio Filippone. 2015. Monte Carlo Strength Evaluation: Fast and Reliable Password Checking. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 158–169.

[11] Aliya Deri and Kevin Knight. 2015. How to make a Frenemy: Multitape FSTs for portmanteau generation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 206–210.

[12] Markus Dürmuth, Fabian Angelstorf, Claude Castelluccia, Daniele Perito, and Chaabane Abdelberi. 2015. OMEN: Faster Password Guessing Using an Ordered Markov Enumerator.. In *ESSoS*. Springer, 119–132.

[13] Dinei Florencio and Cormac Herley. 2007. A large-scale study of web password habits. In *Proceedings of the WWW*.

[14] Dinei Florêncio, Cormac Herley, and Paul C Van Oorschot. 2016. Pushing on string: The'don't care'region of password strength. *Commun. ACM* 59, 11 (2016), 66–74.

[15] Alain Forget, Sonia Chiasson, Paul C van Oorschot, and Robert Biddle. 2008. Improving text passwords through persuasion. In *Proceedings of the 4th symposium on Usable privacy and security*. ACM, 1–12.

[16] Lucian Galescu and James F Allen. 2001. Bi-directional conversion between graphemes and phonemes using a joint n-gram model. In *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*.

[17] Ravi Ganesan, Chris Davies, and Bell Atlantic. 1994. A new attack on random pronounceable password generators. In *17th NIST-NCSC National Computer Security Conference*. Citeseer, 184–197.

[18] Morrie Gasser. 1975. *A random word generator for pronounceable passwords*. Technical Report. DTIC Document.

[19] Dan Goodin. 2016. Why passwords have never been weaker, and crackers have never been stronger. http://arstechnica.com/security/2012/08/passwords-under-assault/.

[20] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2006. Automata theory, languages, and computation. *International Edition* 24 (2006).

[21] Daniel Jurafsky and H James. 2000. Speech and language processing an introduction to natural language processing, computational linguistics, and speech. (2000).

[22] Patrick Gage Kelley, Saranga Komanduri, Michelle L Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. 2012. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 523–537.

[23] Saranga Komanduri, Richard Shay, Lorrie Faith Cranor, Cormac Herley, and Stuart E Schechter. 2014. Telepathwords: Preventing Weak Passwords by Reading Users' Minds.. In *USENIX Security Symposium*. 591–606.

[24] Cynthia Kuo, Sasha Romanosky, and Lorrie Faith Cranor. [n.d.]. Human Selection of Mnemonic Phrase-based Passwords. In *Proceedings of the 2006 Symposium on Usable Privacy and Security*. 67–78.

[25] Michael D Leonhard and VN Venkatakrishnan. 2007. A comparative study of three random password generators. *IEEE EIT* (2007), 227–232.

[26] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. 2014. A study of probabilistic password models. In *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 689–704.

[27] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Fast, lean, and accurate: Modeling password guessability using neural networks. In *Proceedings of the 25th USENIX Security Symposium*.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[29] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[30] Karoline Mortensen and Taylor L Hughes. 2018. Comparing Amazon's Mechanical Turk platform to conventional data collection methods in the health and medical research literature. *Journal of General Internal Medicine* 33, 4 (2018), 533–538.

[31] FIPS 181 NIST PUB. 1993. AUTOMATED PASSWORD GENERATOR (APG). (1993).

[32] Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 703–711.

[33] Suzanne Philion. 2016. An Important Message About Yahoo User Security. https://investor.yahoo.net/releasedetail.cfm?ReleaseID=990570.

[34] Ashwini Rao, Birendra Jha, and Gananand Kini. 2013. Effect of Grammar on Security of Long Passwords. In *Proceedings of the third ACM conference on Data and application security and privacy*. 317–324.

[35] Elissa M Redmiles, Sean Kross, and Michelle L Mazurek. 2019. How well do my results generalize? comparing security and privacy survey results from mturk, web, and telephone samples. In *2019 2019 IEEE Symposium on Security and Privacy (SP), Vol. 00. IEEE*. 227–244.

[36] Richard Shay, Patrick Gage Kelley, Saranga Komanduri, Michelle L Mazurek, Blase Ur, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2012. Correct horse battery staple: Exploring the usability of system-assigned passphrases. In *Proceedings of the eighth symposium on usable privacy and security*. ACM, 7.

[37] Michael R Smith, Ryan S Hintze, and Dan Ventura. 2014. Nehovah: A neologism creator nomen ipsum. In *Proceedings of the International Conference on Computational Creativity*. 173–181.

[38] The GitHub Blog Tom Van Vleck. 2016. Random Password Generator. http://www.multicians.org/thvv/gpw.html.

[39] Blase Ur, Felicia Alfieri, Maung Aung, Lujo Bauer, Nicolas Christin, Jessica Colnago, Lorrie Cranor, Harold Dixon, Pardis Emami Naeini, Hana Habib, Noah Johnson, and William Melicher. 2017. Design and evaluation of a data-driven password meter. In *CHI'17: 35th Annual ACM Conference on Human Factors in Computing Systems*.

[40] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, et al. 2012. How does your password measure up? The effect of strength meters on password creation.. In *USENIX Security Symposium*. 65–80.

[41] Blase Ur, Fumiko Noma, Jonathan Bees, Sean M. Segreti, Richard Shay, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2015. "I added '!' at the end to make it secure": Observing password creation in the lab. In *SOUPS '15: Proceedings of the 11th Symposium on Usable Privacy and Security*. USENIX. http://www.ece.cmu.edu/~lbauer/papers/2015/soups2015-password-creation.pdf

[42] Blase Ur, Sean M Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L Mazurek, William Melicher, and Richard Shay. 2015. Measuring Real-World Accuracies and Biases in Modeling Password Guessability.. In *USENIX Security Symposium*. 463–481.

[43] Rafael Veras, Christopher Collins, and Julie Thorpe. 2014. On the semantic patterns of passwords and their security impact. In *Network and Distributed System Security Symposium (NDSS'14)*.

[44] Rick Wash, Emilee Rader, Ruthie Berman, and Zac Wellmer. 2016. Understanding Password Choices: How Frequently Entered Passwords are Re-used Across Websites. In *Symposium on Usable Privacy and Security (SOUPS)*.

[45] Robert L Weide. 1998. The CMU pronouncing dictionary. *URL: http://www.speech. cs. cmu. edu/cgibin/cmudict* (1998).

[46] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stern. 2010. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 162–175.

[47] Andrew M White, Katherine Shaw, Fabian Monrose, and Elliott Moreton. 2014. Isn't that Fantabulous: Security, Linguistic and Usability Challenges of Pronounceable Tokens. In *Proceedings of the 2014 workshop on New Security Paradigms Workshop*. ACM, 25–38.

[48] Simon S Woo and Jelena Mirkovic. 2016. Improving Recall and Security of Passphrases Through Use of Mnemonics. In *Proceedings of the 10th International Conference on Passwords (Passwords)*.

[49] Moshe Zviran and William J. Haga. 1993. A Comparison of Password Techniques for Multilevel Authentication Mechanisms. *Comput. J.* 36, 3 (1993), 227–237.