Improving Recall and Security of Passphrases Through Use of Mnemonics

Simon S. Woo and Jelena Mirkovic

University of Southern California, Information Sciences Institute

Abstract. Passphrases are regarded as more secure than passwords because they are longer than passwords. Yet, users use predictable word patterns and common phrases to make passphrases memorable, which in turn significantly lowers security. We explore a novel use of *mnemonics*, multi-letter passphrase abbreviations, to make passphrases more memorable and more secure. We use mnemonics during authentication as user hints to achieve cued-recall. We also explore use of mnemonics to guide passphrase creation – we generate a random mnemonic and require a user to produce a passphrase, which matches it. This guides the users away from common phrases and improves security. We evaluate these uses of mnemonics in several IRB-approved user studies with participants from Amazon Mechanical Turk. We find that mnemonics displayed as authentication hints increase recall of passphrases by 30–36% after three days, and by 51–74% after seven days. When used to guide passphrase creation, mnemonics reduce the use of common phrases from 52% to under 5%, while passphrase recall remains high. Users also rate usability of passphrases with mnemonics (for creation or for authentication) higher than usability of classical passphrases.

1 Introduction

Textual passwords are widely used for today's user authentication. Users are advised to choose long character sequences, and utilize characters from multiple classes (e.g., special characters, letters, numbers) to make passwords hard to guess. Because users have many online accounts today (25 in 2006 [22]), remembering passwords is challenging. Users reuse existing passwords, and include predictable word and character patterns [30].These practices decrease security of passwords against automated guessing to a much lower value, than expected solely based on length and composition.

One way to make passwords more secure is to make them longer. Longer passwords should be harder to guess by automated attacks, as the guessing space will be larger. A passphrase is one example of longer passwords, and is usually made by stringing up several words together. These words could be unrelated, e.g., "mother chicken apple", or form a sentence, e.g. "I love apple juice". Passphrases also tend to be more memorable than passwords, as they may contain expressions familiar to a user (e.g., verses of a favorite song) and follow grammatical rules [26, 27, 33, 36]. Therein lies the problem, though! Underlying grammatical structure of passphrases and use of common phrases (e.g., verses from songs) lowers their security well below the security expected by length alone [28, 30]. And if these patterns in passphrases are broken by forcing users to use system-generated passphrases, security increases but recall plummets [32]. Thus there appears to be a trade-off between passphrase recall and security – it is hard to improve one without jeopardizing the other.

In this paper we explore use of *mnemonics* – multi-letter abbreviations of passphrases, to improve both their recall and security. We form a mnemonic out of the first letters of each word in a passphrase. For example, a passphrase "I love apple juice" would be abbreviated to the mnemonic ILAJ. We first explore use of mnemonics as authentication hints to aid user recall. We further explore use of mnemonics during passphrase creation to constrain user choices to only those passphrases, which match the mnemonics. We expect that this approach will reduce the presence of grammatical constructs or common phrases in passphrases, and thus improve their security.

We evaluate mnemonics-aided passphrases in several user studies, which were approved by our IRB, and compare these passphrases against user-chosen and system-chosen passphrases. When displayed as user hints during authentication, mnemonics improve recall by 30–36% after three days, and by 51–74% after seven days. When users are asked to generate passphrases, which match a given mnemonic, use of common phrases reduces from 50% to under 5%. We can combine these two uses of mnemonics to arrive at passphrases, which have good recall and high security (low use of common phrases). While mnemonics as authentication hints lower security against statistical guessing attacks, we can recoup this loss, while retaining high recall, by allowing the system to generate one word of the passphrase, or by requiring longer passphrases. Users further find that mnemonics improve usability of passphrases.

2 Related Work

There is much related work on passwords and alternative authentication methods. We discuss here only those works that are closely related to our proposed use of mnemonics.

Rao et al. [30] discovered that long passwords have a distinct grammatical structure. They analyzed part-of-speech (POS) tag sequences from the Brown Corpus [23] and found that the grammatical structure decreases search space for passwords by more than 50%. Veras et al. [34] explored semantic patterns of passwords and showed how these patterns can be used to greatly improve attack success.

Bonneau and Shutova [15] studied short user-chosen passphrases (2+ words), and showed that they are vulnerable to dictionary attacks, and that they have simple noun structure. Our work focuses on longer passphrases (5+ words), which we find to have a more complex sentence structure.

Shay et al. [32] found that both system-generated passphrases and system-generated passwords are annoying to users and easy to forget. Our studies confirm this finding, but we focus on evaluation of mnemonic-aided passphrases.

Cued-recall systems (e.g., [17,18,20]) have been proposed for graphical passwords, as summarized by Biddle et al. [9]. In these systems a user is shown an image or a set of images as a cue, and must recall which points on the image she clicked, or which images she selected, in order to authenticate. Bicakci and van Oorschot further propose gridWords [8], textual, multi-word passwords that can be entered by selecting them from a dropbox or by locating them on a grid, which serves as a cue. Our use of mnemonics as hints is also an example of cued-recall, but the one applied to textual passphrase and using a textual cue.

Kuo et al. [28] researched the *mnemonic passwords*, which are derived as abbreviations of common phrases such as movie titles. Kuo et al. found 65% of mnemonic passwords via Google searches. We show that our *mnemonic passphrases* do not suffer from the same deficiency – fewer than 7.5% can be found using Bing searches.

User training has also been shown to improve password recall [10, 19], and it may use mnemonic techniques. Our mnemonic structure differs from these works (we use word abbreviations and not visual cues or narratives) and we use mnemonics to guide creation or as authentication hints, rather than for user training.

3 Mnemonics and Passphrases

In this Section we define passphrases and mnemonics, and describe how we use mnemonics to improve recall and security of passphrases.

3.1 Passphrases

A passphrase is a sequence of characters, usually much longer than a password, used for authentication. Passphrases can contain any character, but usually all characters are alphabetic. Passphrases can further contain capitalization, punctuation, numbers and special characters. In this work, we focus on letter-only passphrases and we *normalize* them, by removing capitalization and punctuation from user input. Thus, the only information used for authentication is the *alphabetical content* of the passphrase. This allows us to reason only about the content, which carries most of the meaning for the user, and how this content affects recall and security.

A passphrase with letter-only content will consist of *words*. Most of these words will come from the user's natural language, and may be found in a dictionary or may be proper names of people, objects and locations with significance to the user. The words in a passphrase could be separated by spaces or they could be input together by the user and segmented using a semantic classifier (e.g., [34]).

3.2 Mnemonics

Mnemonics improve recall of information, by associating it with other representation, such as abbreviation, a rhyme, or a pattern. In our work, we use *abbreviations of passphrases* as mnemonics, which we create out of the first letters of passphrase words.

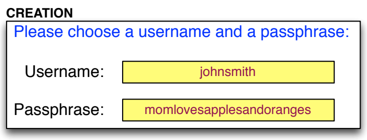
3.3 Using Mnemonics

One could use of mnemonics in two ways. First, they can be used as user hints, to improve recall (aka recall cues [9]) – we will call these ***hint-mnemonics***. A user chooses a passphrase, and the system creates a hint-mnemonic and stores it with the passphrase. For example, a user may choose "Mom loves apples and oranges" and the resulting hint-mnemonic becomes "MLAAO". At authentication, the system prompts the user for her passphrase, and displays the hint-mnemonic. Figure 1(c) illustrates regular authentication with no hints, and Figure 1(d) illustrates authentication with hint-mnemonics.

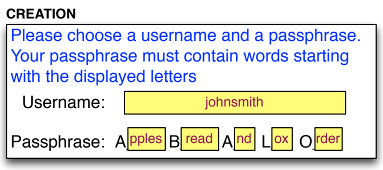
Use of mnemonics as hints will lower security. Passphrases, like passwords, are stored hashed and salted, but mnemonics must be stored in clear since they are displayed to users during authentication. Thus a statistical attacker (see Section 4) can tailor his guessing to words starting with letters of the mnemonic, which greatly reduces the search space. We evaluate this security cost in Section 6, and propose ways to recoup it.

The second possible use of mnemonics is during passphrase creation – we will call these ***guide-mnemonics***. Because users tend to use common word sequences, popular phrases, and grammatical rules in passphrases [15, 28, 30], many passphrases can be guessed by mining these common patterns from public sources. Mnemonics can be used during creation to improve randomness of word choices in passphrases, and to reduce the reuse of passphrases across different accounts.

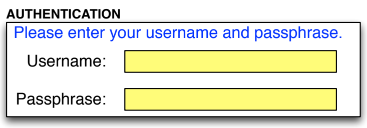
Guide-mnemonics are generated by choosing letters from the alphabet according to some algorithm. A user is then prompted to generate a passphrase matching this mnemonic. Each passphrase word must start with one mnemonic letter, in order. For example, a system may generate a guide-mnemonic "ABALO" and the user may input a matching passphrase like "**A**pples **b**read **a**nd **l**ox **o**rder". Figure 1(a) illustrates regular passphrase creation, and Figure 1(b) illustrates creation with guide-mnemonics. We further allow extraneous passphrase words, which are not part of the mnemonic, because they may aid recall. For example, a user may input "**A**pples **b**read **a**nd the **l**ox are **o**rdered," with "the" and "are" being extraneous words. If guide-mnemonic is also to be used as a hint-mnemonic, we adjust it before storing to reflect all the passphrase words (e.g., ABALO becomes ABATLAO).


CREATION
Please choose a username and a passphrase:
Username: johnsmith
Passphrase: momlovesapplesandoranges

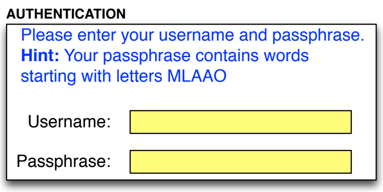
(a) Regular passphrase creation


CREATION
Please choose a username and a passphrase. Your passphrase must contain words starting with the displayed letters
Username: johnsmith
Passphrase: A pples B read A nd L ox O rder

(b) Creation with guide-mnemonics


AUTHENTICATION
Please enter your username and passphrase.
Username:
Passphrase:

(c) Regular passphrase authentication


AUTHENTICATION
Please enter your username and passphrase.
Hint: Your passphrase contains words starting with letters MLAAO
Username:
Passphrase:

(d) Authentication with hint-mnemonics

Fig. 1. Different passphrase creation and auth. methods using hint- and guide-mnemonics

4 Attacker Models and Strength

In this Section we discuss our attacker models, and passphrase strength metrics.

4.1 Attacker Models

We consider the following attacker models, also used in past research [12, 14].

A **brute-force attacker** tries all possible passphrases in random order. Without mnemonics, the strength against brute-force attacks is c^l, where c is the number of characters in the passphrase alphabet and l is the average passphrase length in characters. Since we consider normalized, letter-only passphrases, c would be 26. With hint-mnemonics, a brute-force attacker would try passphrases containing all possible words starting with the given letters in the mnemonic. A brute-force attacker model grossly overestimates passphrase and password strength, as shown in [15, 30]. We thus do not use it for calculation of passphrase strength, but we use it to determine length of guide-mnemonics (see Section 5.3).

A **statistical attacker** compiles lists of common sequences of words, and tries them in order of popularity. We further distinguish between: (1) a **language-model (LM) attacker**, which compiles probabilities for word sequences occurring together, and uses these to guide his guessing, and (2) a **phrase-dictionary attacker [28]**, which compiles lists of common phrases from online content, and tries them whole or in part.

4.2 Strength Against Attacks

Because brute-force attacks are suboptimal for the attacker, we do not evaluate strength against them. For statistical attacks on passwords, prior works use 'heuristic measure of password strength" [14], also known as the *guesswork* [11], which measures the expected number of guesses until authentication success. Pliam [29] suggests using α-work factor as a measure of password strength against an ideal attacker, which knows probability distribution of passwords in the specific corpus. This is unrealistic, and also underestimates strength of passwords in small corpuses, like the one that we have. We use the *guess number* measure of strength by Dell'Amico and Filippone [21], 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.

For a language-model attacker, we calculate the maximum probability of each passphrase, among all possible passphrases of the same word-length from 3.6B words corpus, using an n-gram model. We then convert this probability into *guess number* using Monte Carlo Sampling as proposed in [21].

For a phrase-dictionary attacker, we cannot calculate *guess number* directly, since it is hard to build a large-enough dictionary of common phrases. Instead, we measure the longest overlap between each passphrase and our collection of common phrases. The longer the overlap, the lower the strength against phrase-dictionary attackers, as there are fewer words that the attacker must guess using other methods.

LM Attack Strength. We regard a passphrase as a sequence of words in English language. We then build a language model (LM) [25] to estimate the probability of this sequence of words in English. A language model is a probability distribution over words and word sequences [25]. The n-gram model estimates the probability of every n-word sequence in the corpus, with the probability of single words being their frequency of occurrence in the corpus. The n-gram model captures popular grammatical constructs (e.g., "I love"), and word co-occurrence (e.g., "apple juice", "iced latte").

We built a unigram, bigram, and trigram language model from our corpus (higher-order models are prohibitively expensive to build and have a high number of out-of-vocabulary sequences). We then calculate the probability of each passphrase using these three models, and convert the largest measure to *guess number*. We assume that an LM-attacker can build similar models, and select her guesses in the decreasing order of probability.

When building a language model, it is critical to have a large amount of training data. Otherwise, there would be many out-of-vocabulary (OOV) words and sequences, and our model would significantly underestimate probability of passphrases, and over-estimate strength against statistical attacks. We build our language model using several widely-used, large sources: (1) UMBC WebBase corpus [24] – a collection of English paragraphs, obtained from February 2007 crawl, containing 100 million web pages from more than 50,000 websites, and 3.6 B words, (2) "One Billion Word Language Modeling Benchmark" [16] – a collection of English paragraphs, obtained from WMT 2011 News Crawl data, (3) the texts from Gutenberg Top 100 books [6] – a collection of famous books, (4) the Brown Corpus [23] - containing 500 samples of English-language text, totaling roughly one million words.

After obtaining a maximum probability of a passphrase from LM, we convert it into *guess number* by using Monte Carlo Sampling on a corpus of 100,000 randomly-generated passphrases, as described in [21].

LM Adjustments for Mnemonics. When mnemonics are used as hints during authentication, this changes our language models as some words and some sequences become invalid. We adjust our language models for each passphrase by re-normalizing the word distributions, so we do not overestimate the strength. Let M be the hint-mnemonic for one given passphrase P_M. We adjust our corpus for P_M by keeping only those unigram, bigram, and trigram sequences, which contain the words starting with letters in M, and which follow the order of letters in M. We then use these sequences to build our language models and calculate the probability of P_M.

Out-of-Vocabulary Words. Any language model must cope with out-of-vocabulary (OOV) n-grams. We apply the standard natural language processing approach to this problem, by adding *unknown* words and n-grams to corpus, and using the smoothing to estimate probability of OOV n-grams [25]. This essentially assigns the lowest, non-zero probability to OOV n-grams out of all n-grams in the corpus.

Phrase-Dictionary Strength. Prior research shows that users use common phrases, such as famous quotes, lyrics, poems and movie titles, for generating mnemonic passwords [28]. The same trend could occur for passphrases. In a phrase-dictionary attack, the attacker may use only a portion of a common phrase, or use it in its entirety.

We calculate the overlap between a passphrase P and a phrase dictionary as:

$$overlap(P) = argmax_i(\mathit{WO}(P, phrase_i)) \tag{1}$$

where $phrase_i$ is the i-th phrase in the dictionary, and *WO* is the function that returns the longest word-overlap between P and $phrase_i$. For example, if P="Today early bird catches a fly and worm" and $phrase_i$="Early bird catches a worm", then $\mathit{WO}(P, phrase_i) = 4$. The overlap is calculated ignoring capitalization and punctuation, i.e. the passphrase and the common phrases are both normalized. The overlap, however, must occur in sequence and without gaps. While we cannot convert this measure into *guess number*, the measure correlates inversely with passphrase strength against phrase-dictionary attackers. Attackers can leverage a phrase-dictionary to speed up guessing in the following manner. Assume an attacker knows or can guess the word-length of a passphrase, e.g., five words. First, the attacker would create a phrase-dictionary and try all five-word subsequences of each phrase. If these do not lead to success, she would try all four-word subsequences of each phrase and she would try to guess the word in the fifth or the first position using brute-force or language-model search. If none of these result in authentication success, the attacker would proceed to three-word subsequences, etc.

We built two dictionaries of common phrases. First, we collected famous quotes, lyrics, poems, popular movie titles, and quotes from [31], [1], [4], [3], [35], [15], [7] into our "Famous phrases" dictionary. This dictionary contains 280,550 phrases. Second, we used the Bing search engine to search for each passphrase in our study, select the top 10 matching pages and insert them into our "Popular pages" dictionary. We will show our results in the later section.

5 Passphrase Models

We now provide more details about different passphrase creation and authentication models, which we consider in our work (summarized in Table 1). Our baseline is the **UPass** model – where users select a passphrase freely and receive no hints at authentication. We then transform this model into **UPassHint** – where users freely select a passphrase, but are shown a hint-mnemonic at authentication. These two models help us evaluate how much hint-mnemonic aid recall. We next consider use of mnemonics both as guide-mnemonics and as hint-mnemonics. This model allows us to evaluate the effect of guide-mnemonics on passphrase strength; they do not affect recall. We further consider cases where a system generates one or all words in the passphrase, to increase the security of the passphrase. Our generic model **MNPass(s)** uses the parameter s to denote the number of words generated by the system (zero or one). Finally, we explore how passphrases entirely generated by the system affect security and recall, and how much hint-mnemonics can help the recall of such passphrases. These are our models **SysPass** and **SysPassHint**.

5.1 Baseline: User-Chosen Passphrases

Today, users freely select a passphrase, and the system only enforces a given length policy, or requires presence of specific character classes At authentication, users are

Model	Created		Example	Authentication hint
	User	Sys		
User-chosen passphrases				
UPass	all	0	My Cat Is Very Funny	no
UPassHint	all	0	**A**pple **B**anana **O**range **G**rape **P**ear	yes (**ABOGP**)
Mnemonics-guided passphrase choices				
MNPass(0)	all	0	**I**mportant **U**ganda **G**reg **A**rbitrary **B**ountiful	yes (**IUGAB**)
MNPass(0)-Long	all	0	**S**he **U**ses **L**emon **P**olish **H**igh **U**p **R**ight	yes (**SULPHUR**)
MNPass(1)	all -1	1	**E**uropean **U**nion **S**trange **P**ostings **O**nline	yes (**EUSPO**)
System-chosen passphrases				
SysPass	0	all	Omnipresent Texture Monaco Narcotic Disney	no
SysPassHint	0	all	**P**recocious **B**ase **G**raze **B**lazoned **S**pecialty	yes (**PBGBS**)

Table 1. Passphrase models considered in our work.

prompted for a passphrase, and are not shown any hints. This is our baseline **UPass** model, which we seek to improve with regard to recall and security.

5.2 Improve Recall: Authentication Hints

We allow users to freely select all the words in a passphrase. The system segments the input passphrase into words, and abbreviates each word to its first letter. The letters are then concatenated in order to form a hint-mnemonic. The hint-mnemonic is shown to the user at authentication time. We refer to this passphrase model as **UPassHint**.

5.3 Improve Security: Mnemonic-Guided Passphrase Creation

We first generate a guide-mnemonic, by choosing letters from an alphabet, following some algorithm. We then ask a user to choose passphrase words in order of the mnemonic letters, and ensure that each word starts with the given letter. The system then stores the mnemonic as hint-mnemonic in clear and associates it with the user's username, just like a password salt is stored today. At authentication time, the hint-mnemonic is displayed to aid recall. We call this passphrase model **MNPass**.

Mnemonics Generation. Our goal is to generate mnemonics, which result in passphrases exceeding some *target strength* against a brute-force attacker – *TS*.

The length and composition of letters in mnemonics will directly influence recall, diversity and strength of passphrases. The longer the mnemonic, the longer and more secure the passphrase. However, too long passphrases may lead to reduced recall. Further, each mnemonic letter guides a user to choose a word starting with this letter. Thus the guess space of a passphrase is the product of the guess spaces for each word. To correctly estimate a guess space for each word we must carefully choose a dictionary. Too small a dictionary will underestimate the guess space [32], and too large a dictionary will overestimate it.

We use Google 20K [2] data set, which includes the most common 20,000 English words in order of frequency, calculated over the Google's Trillion Word Corpus. We believe this data set sufficiently well represent the most popular English words in internet. We further pre-filter this dictionary to exclude words starting with q, x, y, and z, which have too few word choices. This leaves us with 22-letter alphabet for mnemonic generation.

We generate mnemonics by randomly selecting letters, one by one, from our 22-letter alphabet, and estimating the strength (guess number) of the resulting passphrase. For a mnemonic containing k letters the estimated strength *ES* is calculated as:

$$\mathit{ES} = \prod_{i=1}^{k} words(dict, mn(i)) \tag{2}$$

where k is the number of letters in the mnemonic, $mn(i)$ returns the letter at position i in the mnemonic, and $words(dict, mn(i))$ returns the number of words from dictionary $dict$, which start with the letter $mn(i)$. When *ES* exceeds the target strength *TS*, we stop and output the mnemonic.

System-Generated Words. Mnemonic-guided passphrase creation may still lead to low passphrase strength if users build their passphrase out of very popular words. We thus explore the password model where the system chooses one word in a passphrase, and the user chooses the rest. To accommodate this model, we parameterize MNPass by s, where s is the number of system-selected words. We explore **MNPass(0)** and **MNPass(1)**. System-selected words may have low recall, as they lack personal significance for a user. Our evaluation shows that this is not the case for MNPass(1), i.e., letting the system choose only one word does not greatly lower recall.

Longer Passphrases. We explore another approach to increase passphrase security, by requiring 20% longer passphrases in approach **MNPass(0)-Long**.

5.4 Improve Security: System-Chosen Passphrases

One can also improve passphrase security by letting the system to generate the entire passphrase. We explore this approach without authentication hints – **SysPass** model – and with hint-mnemonics – **SysPassHint** model.

6 Evaluation

We used Amazon Mechanical Turks to evaluate recall and strength of different passphrase models, described in the previous Section. All our user studies were reviewed and approved by our Institutional Review Board (IRB). We found that hint-mnemonics improve recall by 30-36% after three days, and by 51-74% after seven days (Section 6.2). We further found that guide-mnemonics reduce presence of common phrases in passphrases from 50% to under 5% (Section 6.3). Finally, users reported that mnemonics are easy to use and helpful (Section 6.4).

6.1 User Studies

Amazon Mechanical Turk participants were assigned at random to one passphrase model from the previous Section. We recruited participants with at least 1,000 completed Human Intelligence Tasks (HITs) and $>$95% HIT acceptance rate. We asked

Model	Participants	Avg. words / pass		Avg. chars / pass		Avg, chars / word	
		All	User	All	User	All	User
User-chosen passphrases							
UPass	44	5.3		24.5		4.6	
UPassHint	56	5.4		25.6		4.8	
Mnemonics-guided passphrase choices							
MNPass(0)	66	5.2		26.2		5.0	
MNPass(0)-Long	51	6.9		36		5.2	
MNPass(1)	62	5.0	4.0	30.6	23.3	6.1	5.8
System-chosen passphrases							
SysPass	58	5.1	0	38.3	0	7.4	0
SysPassHint	56	5.2	0	39.0	0	7.6	0
Total	393						

Table 2. Basic statistics on passphrases per passphrase model

each participant to create one passphrase in one sitting. The participant was then asked to return after three days and after one week to authenticate. We paid 35 cents for passphrase creation and 40 cents for each of the two authentication tasks.

Limitations and Ecological Validity. Our study had the following limitations, many of which are common for online password studies. First, it is possible but very unlikely that a participant may enroll into our study more than once. While the same Mechanical Turk user could not enter the study twice (as identified by her MTurkID), it is possible for someone to create multiple Mechanical Turk accounts. There is currently no way to identify such participants. Second, we cannot be sure that our participants did not write down or photograph their passphrases. We did not ask the participants if they have done this in post-survey, because we believed that those participants who cheated would also be likely to not admit it. We designed our study to disincetivize cheating. We promised to pay participants in full regardless of authentication success. Our study mechanisms further detected copy/paste actions and we have excluded any participant that used these (for whatever reason) from the study. 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 equally. Thus our data can still be used to study improvement of recall and security between password models. Third, while we asked Mechanical Turkers to pretend that they were creating passphrases for real servers, they may not have been very motivated or focused. This makes it likely that actual recall of real-world passphrases would be higher across all models. While it would have been preferable to conduct our studies in the lab, the cost would be too high (for us) to afford as large participation as we had through the use of Mechanical Turks.

Passphrase model parameterization. We wanted to make passphrase models comparable to each other in character-length. This allowed us to investigate factors like recall and security of passphrases, independent of length. The length of MNPass models is controlled by the target strength (*TS*) parameter. We use $TS = 95^8$, which is the theoretical, maximum strength against brute-force attacks for 3class8 passwords. These passwords are generated according to the frequently-used password policy: 8-character length and use of at least three out of the following four character classes: lowercase and uppercase letters, digits and special characters. While the exact mnemonic length depends on the randomly chosen letters of the mnemonic, most of our mnemonics were 5–6 characters in length for $TS = 95^8$, and thus led to 5–6 word long passphrases. To make the other models comparable, we instructed participants to enter passphrases, which contain at least five words. Also, for models where all words are chosen by the system, we mimicked the mnemonic-guided creation with $TS = 95^8$. We generated the mnemonic, and then drew all the words for the passphrase at random from our dictionary, to match the mnemonic. The MNPass(0)-Long model explores longer passphrases. We used target strength $TS = 95^{10}$ for this model, which mimics 3class10 passwords and which led to passphrases with 6–7 words.

Passphrase Creation. We provided a short tutorial and examples for each passphrase model and then asked participants to create one passphrase. All users were asked not to write down or copy their answers, and to rely on memory only. We automatically checked if passphrase constraints (e.g., word length, mnemonic match) were met during passphrase creation, and on failure, we asked the user to recreate the passphrase.

Passphrase Authentication. Each user was asked to make two authentication visits, one after three days, and one after one week since passphrase creation. We allowed at most five trials to authenticate per passphrase and per visit. All users were asked not to paste their answers. We had automated detection of copy or paste attempts in our forms, and we rejected the users who were detected to perform either of these two actions. We further displayed a notice to participants, at both the creation and authentication screens, that they will get paid regardless of their authentication success. This ensured that participants had no monetary incentive to cheat. After the second authentication visit, we asked participants to complete a usability survey.

Participant and Passphrase Statistics. In total, there were 1,273 participants who created a passphrase. Out of 1,273 participants, 731 (57.47%) participants returned for the first authentication and 426 (58.3%) returned for the second authentication. Total of 393 participants completed both the first and the second authentication, and we only include their data in our analysis. Some basic statistics for the passphrases per passphrase model are shown in Table 2. We had 44–66 participants per password model. Except for MNPass(0)-Long, all other models generated passphrases of comparable word-length. User-chosen words tended to be shorter (4.21–5.8) than system-chosen words (7.4–7.6 characters), which directly maps into differences in character length between user-chosen passphrases, and those that had some words chosen by the system.

Similar to the prior research [15, 30] we evaluated the semantic structure of passphrases, by applying part-of-speech (POS) tagging and dependency parsing. We grouped the passphrases into three categories: (1) noun sequence (contain nouns only), (2) sentence (contain subject, verb and object) and (3) segment (all others). For space reasons, we only summarize these results. About half of user-chosen passphrases follow the sentence structure, and about a third are sequences of nouns. Conversely Bonneau and Shutova found in [15] that most short passphrases were segments. The difference between our findings and theirs is likely due to our use of longer passphrases (5+ words instead of 2+). Mnemonic-guided passphrase creation disturbs the user tendency towards the sentence structure, and leads to more balanced distribution of semantics structures (towards noun sequences and segments), which increases attacker's guessing effort. Longer (6-7 word) passphrases tend to favor sentence structure as much as user-chosen passphrases, but they have higher percentage of segments and lower percentage of noun sequences. System-chosen passphrases all have the segment structure.

6.2 Mnemonics Improve Recall

We considered recall successful if users matched the entire passphrase in its normalized form – with removed capitalization, punctuation and whitespaces. We denote this match criterion *exact match*. We also considered a *relaxed match* criterion, where we normalize nouns to their singular form, and verbs to their stem form using the Porter stemming algorithm [5], before both storing and authentication. We hypothesized, and our results prove, that this relaxed matching further improves recall, and does not greatly decrease strength against statistical attacks. Table 3 shows the authentication success per model, for the exact and the relaxed matching.

Passphrase Model	**Authentication success**			
	exact match		**relaxed match**	
	3 day	**7 day**	**3 day**	**7 day**
w/o hint				
UPass	52.3%	40.0%	63.6%	45.0%
SysPass	20.7%	12.5%	22.4%	14.3%
w hint				
UPassHint	71.4%	69.6%	76.8%	73.2%
SysPassHint	26.8%	18.9%	28.7%	19.6%
MNPass(0)	69.7 %	66.7%	80.3%	66.7%
MNPass(1)	69.3%	67.7%	75.8%	69.3%
MNPass(0)-Long	66.7%	62.8%	78.4%	72.5%

Table 3. User recall three days and seven days after passphrase creation

Hint-Mnemonics Improve Recall. Within the same passphrase model (UPass vs. UPassHint, SysPass vs. SysPassHint) there was a statistically significant increase (Welch Two Sample t-test, 95% confidence interval), in recall rates when hint-mnemonics were used. Recall rates of UPassHint were higher than those of UPass (t(88)=1.76, p=0.04), and recall rates of SysPassHint were higher than those of SysPass (t(85)=2.75, p=0.003). Hint-mnemonics improve recall by 30–36% after three days. The role of hint-mnemonics becomes more prominent as time goes on. After seven days, hint-mnemonics improve recall by 51–74%. Improvement is also greater for user-chosen passphrases, than for system-chosen ones. User-chosen passphrases have personal significance to the user, and hints help users recall this association and thus recall the passphrase. Over four days (day 3 vs day 7 recall), recall of user-chosen passphrases declines by 24%, but when hints are used, recall declines only by 3%. System-chosen passphrases lack personal significance. Over four days, their recall declines by 40%, without hints, and by 30% with hints. Overall, recall rate of system-chosen passphrases is around 2.5–3.2× lower than that of user-chosen passphrases.

Hints Aid Recall of Important Facts. Users could fail to recall passphrases because they forgot the words they chose during creation, or because they forgot details about these words, e.g., the exact form of the verb they used (e.g., "work" vs "working" vs "worked") or if they used plural or singular form of a noun (e.g., "egg" vs "eggs"). To investigate these aspects we compare recall for exact match versus relaxed match criteria. Relaxed matching improves recall by 3–21% over exact matching. The improvement is larger when no hints are used (8–21%) than with the hints (3–17%). Yet, these improvements are much lower than improvements from using hints (e.g., 12% versus 72% for UPass), signifying that hints aid recall of important facts.

MNPass Recall Comparable to UPass Recall. Mnemonic-guided passphrase creation may jeopardize personal significance of passphrases to the user, and thus impair recall. We investigate this possibility by comparing recall of our three MNPass models with UPassHint. MNPass(0) is structurally the most similar to UPassHint – both models contain around five user-chosen words per passphrase. Recall of MNPass(0) is a little lower than that of UPassHint (69.7% vs 71.4% after three days, 66.7% vs 69.6% after seven days). When one of the words is chosen by the system – MNPass(1) – the recall stays roughly the same, and comparable to that of UPassHint. Thus system-based generation of one passphrase word has minor impact on recall. But the generation of all words by the system (as in SysPass and SysPassHint) drastically lowers recall. When we increase the length of the MNPass from five to seven words, the recall declines further. It is 66.7% for MNPass(0)-Long vs 71.4% for UPassHint after three days, and 62.8% vs 69.6% after seven days).

6.3 Mnemonics Improve Security Against Phrase-Dictionary Attacks

We next investigate the impact of guide-mnemonics on passphrase strength. Table 4 shows the median strength (guess number) against LM attacker per passphrase model, in columns 2–5. We use the exact and the relaxed matching for attacker guesses. We show the strength against the full LM attacker (when no hint-mnemonics are shown to users) in columns 2–3, and the strength against the adjusted LM attacker (when hint-mnemonics are shown to users) in columns 4–5. Columns 6–13 show the strength against phrase-dictionary attacker as the percentage of passphrases that have 0, 1, 2 and 3+ words overlapping with some phrase in the dictionary. The longer the overlap, the lower the strength against phrase-dictionary attacker. We show the overlap for our "Famous phrases" dictionary (columns 6–9) and for our "Popular pages" dictionary (columns 10–13).

Mnemonics Impact on Strength Against LM Attacks. When mnemonics are used only during passphrase creation, but not during authentication (columns 2–3 in Table 4), the strength against LM attacker does not change significantly. Welch Two Sample t-test shows no difference in means, (t(99)=0.07, p=0.95). System-chosen passphrases show statistically significant increase in strength (t(74)=-3.07, p=0.002) from $8 \cdot 10^{15}$ to $2.6 \cdot 10^{21}$. However, when mnemonics are used both during creation and authentication, the passphrase strength against LM attacker drops five orders of magnitude, from $4 \cdot 10^{17}$ to $1.1 \cdot 10^{12}$. Thus hint-mnemonics have a large impact on strength against LM attacker.

We can recoup much of this strength loss, if we allow the system to suggest one word in a passphrase (MNPass(1)) or if we ask for longer passphrases (MNPass(0)-

Passphr. Mod.	**LM attack. (guess num)**				**Phrase-dict attack. (%)**							
	w/o hint		**w hint**		**Famous phr.**				**Popular pgs**			
	exact	**relax.**	**exact**	**relax.**	**0**	**1**	**2**	**3+**	**0**	**1**	**2**	**3+**
User-chosen passphrases												
UPass UPassHint	$8 \cdot 10^{15}$	$4.2 \cdot 10^{15}$	$1.2 \cdot 10^{10}$	$8.8 \cdot 10^{9}$	0	8	40	52	0	20	29	51
Mnemonics-guided passphrases												
MNPass(0)	$4.0 \cdot 10^{17}$	$1.3 \cdot 10^{17}$	$1.1 \cdot 10^{12}$	$4.0 \cdot 10^{11}$	0	34	61	5	0	29	63	8
MNPass(1)	$2.6 \cdot 10^{19}$	$1.0 \cdot 10^{19}$	$1.3 \cdot 10^{16}$	$5.4 \cdot 10^{15}$	8	50	39	3	8	44	48	0
MNPass(0)-Long	$2.6 \cdot 10^{21}$	$7.1 \cdot 10^{20}$	$1.5 \cdot 10^{15}$	$1.3 \cdot 10^{15}$	6	17	59	18	6	16	60	18
System-chosen passphrases												
SysPass SysPassHint	$2.6 \cdot 10^{21}$	$7.9 \cdot 10^{20}$	$3.9 \cdot 10^{17}$	$3.9 \cdot 10^{17}$	0	100	0	0	0	97	3	0

Table 4. Passphrase strength against LM attacker and against phrase-dictionary attacker per passphrase model. For LM attacker, we show the average of the estimated strength guess number. For the phrase-dictionary attacker, we show the percentage of passhphrases, which have a given number of words overlapping with a common phrase.

	Usability								Deployability						Security										
Scheme	**Memorywise-Effortless**	**Scalable-for-Users**	**Nothing-to-Carry**	**Physically-Effortless**	**Easy-to-Learn**	**Efficient-to-Use**	**Infrequent-Errors**	**Easy-Recovery-from-Loss**	**Accessible**	**Negligible-Cost-per-User**	**Server-Compatible**	**Browser-Compatible**	**Mature**	**Non-Proprietary**	**Res.-to-Physical-Observation**	**Res.-to-Targeted-Impersonation**	**Res.-to-Throttled-Guessing**	**Res.-to-Unthrottled-Guessing**	**Res.-to-Internal-Observation**	**Res.-to-Leaks-from Other Verifiers**	**Res.-to-Phishing**	**Res.-to-Theft**	**No-Trusted-Third-Party**	**Requiring-Explicit-Consent**	**Unlinkable**
Web passwords			●		●	●	○	●	●	●	●	●	●	●		○						●	●	●	●
Graph. pass. (PCCP)			●		●	○ 🟥	○	●	🟥	●	🟥	●	🟥	●		● 🟩	○ 🟩			● 🟩	● 🟩	●	●	●	●
MNPass	○ 🟩	○ 🟩	●		●	○ 🟥	● 🟩	●	●	●	🟥	●	🟥	●			● 🟩	● 🟩		○ 🟩		●	●	●	●

● – offers the benefit, ○ – almost offers the benefit, *no circle* – does not offer the benefit

🟩 – better than passwords, 🟥 – worse than passwords, *no circle* – no change

Table 5. Comparing MNPass to passwords using the UDS framework [13]. MNPass outperform passwords in usability and security (6 categories) and do worse in deployability (2 categories). MNPass outperform PCCP in 6 categories, and do worse in 3 categories.

Long). Both of these approaches return the average passphrase strength against LM attacker to guess numbers above 10^{15}.

Mnemonics Impact on Strength Against Phrase-Dictionary Attacks. We report the percentage of passphrases, which have a given-length word overlap with common phrases in our dictionaries, in Table columns 6–13. We discuss only overlap with famous phrases (columns 6–9), the other overlap shows a similar trend. Having a 1-word overlap is common and expected, as passphrases use popular English words. 52% of user-chosen passphrases have 3+ word overlap with some popular phrase, and 40% have 2-word overlap. In addition, 5% of **UPass** and 13% of **UPassHint** passphrases are fully matched with popular quotes or search results in our phrase-dictionaries. Examples are “where there is will there is way”, “Seven Days Of The Week”, “roses are red violets are blue”, “eines schickt sich nicht fur alle”, etc. Therefore, phrase-dictionary attacks can be highly effective.

When mnemonics are used during passphrase creation, 3+ overlap reduces to only 5% for MNPass(0) and 2-word overlap increases to 61%. When the system is allowed to choose one word in a passphrase, the 3+ word overlap reduces to only 3% for MNPass(1) and 2-word overlap reduces to 39%. Finally, when users are asked to generate longer passphrases, 18% of them have 3+ word overlap with popular phrases, and 59% have 2-word overlap. We conclude that MNPass(0) and MNPass(1) models significantly increase strength of passphrases against phrase-dictionary attacks by reducing the incidence of 3+ word overlaps.

System-Aided Better Than Longer. We can improve strength of passphrases by making guide-mnemonics longer or by allowing the system to choose one word of the passphrase. Comparing the strength of the resulting passphrases, we find that both approaches have the same effect. Both increase the passphrase guess number for the LM attacker from $1.1 \cdot 10^{12}$ to $1.3 \cdot 10^{16}$ or $1.5 \cdot 10^{15}$, i.e. by 1,000–10,000$\times$. However, system-aided word selection has much lower negative impact on recall (Section 6.2) and leads to passphrases, which do not significantly overlap the common phrases (3+ word overlap is under 5%).

Mnemonic-Guided Comparable to System-Chosen. Another way to improve strength of passphrases is to let the system generate all the passphrase words, but that leads to low recall (Section 6.2). Comparing the strength of SysPass, with the strength of MNPass(1) and MNPass(0)-Long, we find that passphrases fully generated by system are only around 100$\times$ stronger than mnemonic-guided passphrases.

Relaxed Matching Is Acceptable. Relaxed matching lowers security, because more of the attacker’s guesses lead to successful authentication. We measure this effect by applying the exact and the relaxed matching to our language model for strength calculation. Relaxed matching lowers the strength by at most 10 $\times$, and thus has an acceptable security cost, while greatly improving recall.

6.4 Users Like Mnemonics

After each participant completed their 7-day authentication, they were asked to rate their agreement with the following statements, on a Likert-scale, from 1 (strongly disagree) to 10 (strongly agree): (1) Mnemonic hints were helpful for authentication, and (2) Authentication approach was easy to use. For space reasons, we summarize these results. Participants strongly agreed that hints were helpful for recall of MNPass (7.76) and UPass (6.86), but they were much less helpful for SysPass (4.46). Regarding ease of use, participants rated mnemonic-guided passphrases (6.98–8.25) as highly as user-chosen passphrases (6.57–7.86), and they rated system-chosen passphrases much lower (2.51–3.15).

6.5 Mnemonic Passphrases Outperform Passwords and PCCP

In [14], Bonneau et al. present the usability-deployability-security (UDS) framework. They define 25 properties of Web authentication schemes and use them to rate 35 password-replacement schemes. In Table 5 we reproduce the rating of passwords, the rating of PCCP, which is the closest authentication approach to use, and we add our rating for the MNPass model.

MNPass outperforms passwords and PCCP in six categories. Regarding usability, MNPass are *Quasi-Memorywise-Effortless*, as users benefit from hint-mnemonics but still forget some words. We also award *Quasi-Scalable-for-Users* based on design, since users are cued by hint-mnemonics, which should lower cognitive burden. Future work should validate or refute this claim through human user tests. MNPass are further *Infrequent-Errors*, thanks to our use of relaxed matching and normalization. Regarding security, MNPass are *Resilient-to-Throttled-Guessing* and *-Resilient-to-Unthrottled-Guessing*, because our evaluation shows that their strength approaches that of fully random 3class8 passwords. Because different servers generate guide-mnemonics independently, MNPass are also *Quasi-Resilient-to-Leaks-from Other Verifiers*. We do not award full resilience, because a user may still choose the same word for the same mnemonic letter. MNPass do worse than passwords in one usability and two deployability categories, and worse than PCCP in three categories. They are less *Efficient-to-Use*, as they take longer to create and authenticate. They are not *Server-Compatible* nor *Mature*, because they are a research technology and are not widely deployed. They are not *Resilient-to-Targeted-Impersonation* as a friend may be able to guess a passphrase when shown a hint-mnemonic. Finally, they are not *Resilient-to-Phishing* as an attacker who displays the same hint-mnemonic can harvest a user's passphrase.

7 Conclusion

It is difficult to create a passphrase mechanism, which has both a high recall and a high strength against statistical attacks. We explored use of mnemonics to this effect. We found that use of mnemonics as authentication hints significantly improves recall, because it helps users remember which words they chose during passphrase creation. Mnemonics can further be used to guide passphrase creation, which reduces use of common phrases and improves strength against statistical attacks. While the use of mnemonics at authentication lowers security, we can recoup this loss by allowing the system to choose one word in a passphrase. This passphrase model keeps both the security and recall of passphrases high – the security matches that of system-chosen passphrases, while the recall matches that of user-chosen passphrases. Mnemonics at passphrase creation would further prove a valuable aid to reduce passphrase reuse. We thus believe mnemonics are a promising technique to improve user authentication.

8 Acknowledgements

We thank Matteo Dell'Amico (Symantec Research Labs) for help in evaluating passphrase strength, and we thank anonymous reviews for their helpful feedback.

References

1. Famous Quotes at BrainyQuote. `http://www.brainyquote.com/`.
2. Google-20K-English Words. `https://github.com/first20hours/`.
3. Love Poems And Quotes. `http://www.lovepoemsandquotes.com/`.
4. Short Poems. `https://www.shortpoems.org/`.
5. The Porter Stemming Algorithm. `http://tartarus.org/martin/PorterStemmer/`.
6. Top 100 - Project Gutenberg. `https://www.gutenberg.org/browse/scores/top`.
7. Top 100 Favorite Movie Quotes. `http://www.imdb.com/`.
8. BICAKCI, K., AND VAN OORSCHOT, P. C. A Multi-word Password Proposal (gridWord) and Exploring Questions about Science in Security Research and Usable Security Evaluation. In *Proceedings of the 2011 New security paradigms workshop* (2011), ACM, pp. 25–36.
9. BIDDLE, R., CHIASSON, S., AND VAN OORSCHOT, P. C. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys (CSUR) 44*, 4 (2012), 19.
10. BLOCKI, J., KOMANDURI, S., CRANOR, L., AND DATTA, A. Spaced Repetition and Mnemonics Enable Recall of Multiple Strong Passwords. *arXiv preprint arXiv:1410.1490* (2014).
11. BONNEAU, J. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy* (May 2012).
12. BONNEAU, J., BURSZTEIN, E., CARON, I., JACKSON, R., AND WILLIAMSON, M. Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google. In *25th International World Wide Web Conference (WWW)* (May 2015).
13. BONNEAU, J., HERLEY, C., VAN OORSCHOT, P. C., AND STAJANO, F. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *2012 IEEE Symposium on Security and Privacy* (May 2012).
14. BONNEAU, J., HERLEY, C., VAN OORSCHOT, P. C., AND STAJANO, F. Passwords and the Evolution of Imperfect Authentication. *Communications of the ACM* (July 2015).
15. BONNEAU, J., AND SHUTOVA, E. Linguistic Properties of Multi-word Passphrases. In *Financial Cryptography and Data Security*. Springer, 2012, pp. 1–12.
16. CHELBA, C., MIKOLOV, T., SCHUSTER, M., GE, Q., BRANTS, T., AND KOEHN, P. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *CoRR abs/1312.3005* (2013).
17. CHIASSON, S., FORGET, A., BIDDLE, R., AND VAN OORSCHOT, P. C. Influencing Users Towards Better Passwords: Persuasive Cued Click-points. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 1* (2008), British Computer Society, pp. 121–130.
18. CHIASSON, S., VAN OORSCHOT, P. C., AND BIDDLE, R. Graphical Password Authentication Using Cued Click Points. In *European Symposium on Research in Computer Security* (2007), Springer, pp. 359–374.
19. DAS, S., HONG, J., AND SCHECHTER, S. Testing Computer-Aided Mnemonics and Feedback for Fast Memorization of High-Value Secrets. *Proceedings of the 2016 Usable Security Workshop*.
20. DAVIS, D., MONROSE, F., AND REITER, M. K. On User Choice in Graphical Password Schemes. In *USENIX Security Symposium* (2004), vol. 13, pp. 11–11.
21. DELL'AMICO, M., AND FILIPPONE, M. Monte Carlo Strength Evaluation: Fast and Reliable Password Checking. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), ACM, pp. 158–169.
22. FLORENCIO, D., AND HERLEY, C. A Large-scale study of Web Password Habits. In *Proceedings of the 16th WWW conference* (2007), ACM, pp. 657–666.

23. Francis, W. N., and Kucera, H. Brown corpus manual. *Brown University* (1979).
24. Han, L., Kashyap, A., Finin, T., Mayfield, J., and Weese, J. UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics* (2013), vol. 1, pp. 44–52.
25. Jurafsky, D. *Speech & language processing*. Pearson Education India, 2000.
26. Keith, M., Shao, B., and Steinbart, P. A Behavioral Analysis of Passphrase Design and Effectiveness. *Journal of the Association for Information Systems 10*, 2 (2009), 2.
27. Keith, M., Shao, B., and Steinbart, P. J. The Usability of Passphrases for Authentication: An Empirical Field Study. *International journal of human-computer studies 65*, 1 (2007), 17–28.
28. Kuo, C., Romanosky, S., and Cranor, L. F. Human Selection of Mnemonic Phrase-based Passwords. In *Proceedings of the 2006 Symposium on Usable Privacy and Security*, pp. 67–78.
29. Pliam, J. O. On the Incomparability of Entropy and Marginal Guesswork in Brute-force Attacks. In *Progress in Cryptology, INDOCRYPT 2000*. Springer, pp. 67–79.
30. Rao, A., Jha, B., and Kini, G. Effect of Grammar on Security of Long Passwords. In *Proceedings of the third ACM conference on Data and application security and privacy* (2013), pp. 317–324.
31. Robins, G. Good Quotations by Famous People. `http://www.cs.virginia.edu/~robins/quotes.html`.
32. Shay, R., Kelley, P. G., Komanduri, S., Mazurek, M. L., Ur, B., Vidas, T., Bauer, L., Christin, N., and Cranor, L. F. Correct Horse Battery Staple: Exploring the Usability of System-assigned Passphrases. In *Proceedings of the 2012 Symposium on Usable Privacy and Security*, p. 7.
33. Spector, Y., and Ginzberg, J. Pass-sentence – a New Approach to Computer Code. *Computers & Security 13*, 2 (1994), 145–160.
34. Veras, R., Collins, C., and Thorpe, J. On semantic patterns of passwords and their security impact. In *NDSS* (2014).
35. Weir, M. Quotes to use in pass-phrase cracking. `https://sites.google.com/site/reusablesec/Home/custom-wordlists`.
36. Zviran, M., and Haga, W. J. A Comparison of Password Techniques for Multilevel Authentication Mechanisms. *The Computer Journal 36*, 3 (1993), 227–237.

A Examples of Generated Passphrases for Each Passphrase Model

In this Appendix, we provided 10 random samples of passphrases that the participants generated for different passphrase models.

UPass/UPassHint

- My Cat Is Very Funny
- My Dog Was Named Max
- Cat Came Drink Bowl Milk
- Bob And Ted And Carol
- The World Is Not Enough
- Brown Orange Double Deer Horse
- Citi Is The Ideal Bank
- Salvador Dali Max Ernst Joan Mitchell

- I Saw My Dad Died
- Please remember all your stuff

MNPass(0):

- Monday Tuesday One Day Xero
- Good Boys Love Orange Juice
- Public Works Corporation Hold Judicial Deployment
- Safe Cute Underwater Drinking Dorylaus
- Lazy Oscar Dog Goes Inside
- Big Jack Had King And Elephant
- More Power Vote Jim By Italy
- Often People Underestimate Fantastic News
- Good Gibbons Break In Nairobi
- Could Tigger Get Loved Once

MNPass(1):

- Red Ananders Mom Ananders Dog
- Alpine Surfand Turfweeds Rarely Ordered
- Can Agaya Really Pull Out
- Cats Are Running Torured Out
- My Intelligence Shows The Infearior
- Put Anything Towards Honey Onaben
- Under New Bridge Up Richyard
- Dog On Ground Funky Iselmux
- Street Unsures Pets Eat Rats
- Maigne Igloo Looks Toward Ocean

MNPass(0)-Long:

- Come Over Mother And Tell Us Lies
- She Uses Lemon Polish High Up Right
- Flying Lemurs Are Not Notably Expert Landers
- Soon Andrew Niece Drive A Reno
- Can A Kangaroo Eat What Al Likes
- Maybe Endor Noise Should Undergo Radical Action
- All Nine Ten Bring It Right Down
- Rice And Things In Our Nearest
- Free Of Long And Creepy Inaccurate Noises
- Parrot Elephant Rooster In Care Attention Room

SysPass/SysPassHint:

- Economist Compressed Lunacy Decreased Marine
- Happens Mouth Wyoming Java Uncritical
- Bandy Sophisticated Urgency Berkeley Capita
- Tripoli Crumbled Forming Expence Rapturous
- Encountered Jewelry Albany Rewritten Economize Upturned
- Mahdi Saskatchewan Fell Frosted Youthfulness
- Qualification Howitzer Conjuring Experimental Relentlessly Advisor
- Nearby Betty Competency Competitors Quince
- Jobs Quire Trident Yours Operations
- Expo Generic Overdue Houseboat Voiceless