Passwords for Everyone: Secure Mnemonic-based Accessible Authentication

Umut Topkara Mercan Topkara Mikhail J. Atallah
Department of Computer Science
Purdue University
utopkara,mkarahan,mja@cs.purdue.edu

Abstract

In many environments, a computer system is severely constrained to the extent that the practical input mechanisms are merely binary switches. Requiring the user to remember a long random bit string and to authenticate by entering each bit in the available binary input mechanism, is completely impractical. This paper deals with the question of authentication in such environments where the inputs are constrained to be yes/no responses to statements displayed on the user's screen. We present PassWit, a mnemonic-based system for such environments that combines good usability with high security, and has many additional features such as (to mention a few) resistance to phishing, keystroke-logging, and compatibility with currently deployed systems and password file formats (hence it can co-exist with existing login mechanisms).

1 Introduction

We present PassWit, an authentication system that can be used in environments where the input mechanisms are constrained to low bandwidth switches. For example, a disabled person may only be capable of yes/no responses to prompts from the screen (by different nods of the head, eye movements, or even by different thought patterns that are captured by a sensor). Alternatively, the user may not suffer from any impairment yet the environment precludes the use of a keyboard or keypad, as happens with tiny portable devices such as some of the smaller mp3 players, voice sensors at the doors of restricted-access areas, and hands-free situations such as construction work sites, operation of a motor vehicle, etc. Finally, a case can be made, in situations where shoulder-surfing is prevalent (such as in crowded cyber-cafes), for deliberately restricting the input to be a response that is hard to detect by a shoulder-surfer (e.g., left-click vs right-click).

PassWit enables users to achieve security of truly random passwords while authenticating themselves by just answering a series of "yes/no" questions. An important ingredient in our recipe is the use of a mnemonic that enables the user to produce a long enough (hence more secure) string of appropriate yes/no answers to displayed prompts (i.e., challenges). Each user is required to remember a mnemonic sentence of which they have several choices to pick from. Another important ingredient is the non-adaptive nature of these challenges – so they are inherently non-revealing to a shoulder-surfer or phisher. The mnemonic is a sentence or a set of words known only to the user and authenticating server (in the server they are stored in a cryptographically protected way rather than in the clear) – the users are never asked to enter their mnemonics to the system, they only use the mnemonic to answer the server's challenge questions. PassWit is safe against many attacks including shoulder surfing, phishing, and acoustic attack.

In PassWit, authentication can be achieved without requiring any special input device, or any computation at the client site. The authentication questions are designed in such a way that a short mnemonic sentence can encode a long password. There is no restriction on the size of the mnemonic sentence or the password, and the security (hence the length) of the passwords can be increased by requiring the user to remember more than one sentence.

Our design is compatible with pure text based interfaces as well as other media interfaces that can represent the text mnemonics. Our usage of text for mnemonics is not necessary but it is what we implemented for reasons of convenience and compatibility with existing login mechanisms; similar methods can be easily used in conjunction with speech, video, or pictures.

Moreover, using text for mnemonics (as opposed to pictures, video or audio) brings more flexibility to our system, since it is compatible with text consoles and LED screens. There is no language restriction for PassWit, it can be implemented for languages other than English.

2 Challenges and The Adversary Model

The primary challenge that we seek to overcome is to develop an authentication system which would be able to work in input-constrained environments. The ability to provide yes/no inputs makes it possible to transmit any random bit string but it does not help at all for remembering which bit string to transmit. A useful scheme needs to involve a mnemonic that makes it possible to securely remember a long random bit string, by remembering only a relatively short sentence.

A desirable authentication system should not require the user to carry an extra portable device (e.g. calculator) or even a paper and a pen to be widely applicable.

The password initialization and reset should be easy. In our case, initialization consists of asking the users on which topic they would like their mnemonic sentence to be, and later providing them alternative mnemonic sentences. Note that the users can use only yes/no answers to input their choice of mnemonic sentence.

We assume that all of the information used by the system during mnemonic creation is public and the adversary has equal (or more) computational power compared to our system.

The text passwords as well as the mnemonics in our system have to be secure against *dictionary attacks*. Mnemonics can not be a popular quote, or the lyrics of a well known song. We achieve this security by employing a mnemonic sentence generation method which was proposed as a part of a scheme for remembering conventional text passwords [16]. An adversary who has access to the passwords file (e.g., "/etc/passwd") does not gain any advantage when our system is used for authentication; our system does not weaken the security of existing authentication, it improves it by helping the users to have a truly random password.

A desired system should have resistance to attacks that involve capturing the user input for *replay attacks* (e.g. keystroke logging) or capturing the challenges (e.g. shoulder surfing). In our system the challenges and the user's responses are meaningless to an adversary, unless both are successfully captured at the same time.

Resistance to *phishing attacks* is a desired feature. Mnemonics are shared secrets between the user and the authenticating server and the users are never asked to enter their mnemonics to the system. The users can detect an adversary that does not know the shared secret. Otherwise an adversary does not gain any information about the password even if the user answers random phishing challenges.

Refer to [17] for a more detailed analysis of the challenges involved in building an authentication system for input constrained environments.

3 System Overview

We propose PassWit, an authentication system that is based on mnemonic passwords [16], whose details will be described in the following subsections, where $\mathcal{P}$ denotes the user's previously existing (and securely generated) password bit string (for now we assume $\mathcal{P}$ is 40 bits long, but we can accommodate any other length).

3.1 Password initialization step

1. The system generates a number of random sentences $s_1, \ldots, s_\lambda$ and displays them to the user. Each sentence has a length of μ words (not counting functional words such as "the", "a", "with"). In our implementation we used $\mu = 10$. For example, s_2 could come from tracing a random left-to-right path along the columns of Table 1, using some of the password bits to select one word from each column. In this case, 4 password bits are used per column and first column shows the bit string encoded by the words in the same row. For example, if $\mathcal{P} =$ 0101100101010011111101001000101010001101 then the resulting s_2 is *Angry union artists simply dismissed demand to forgive the laziness of the crazy mayor.* Each s_i is selected from a separate table like Table 1 which was derived from a different text source (e.g., sports news, stock markets, etc).

2. The user selects one of the above s_i's, suppose it consists of the successive words $m_1, m_2, \ldots, m_\mu$.

3. The column C_j from which word m_j was selected contains what we call the equivalence class (in that table) of the word m_j. We use r to denote the size of an equivalence class; in our example $r = 16$. The user does not need to memorize the equivalence class (only m_j needs to be remembered).

3.2 Authentication step

For $j = 1, \ldots, \mu$ in turn, the system asks the user , $\ell = \log_2 r$ questions ($\ell = 4$ in our example) about column C_j, as follows.

1. The system randomly permutes the entries of column C_j before creating the challenges at each session (which foils a replay attack). For the ith entry of C_j in the permuted order, let $b_{i,3}b_{i,2}b_{i,1}b_{i,0}$ be the binary representation of i. For instance last column of Table 1 might be permuted as {leader, senator, enemy, foe, king, queen, president, chairman, children, mayor, friend, ally, associate, assistant, manager, supporter}.

2. The system creates 4 sets Q_3, Q_2, Q_1, Q_0 such that the ith word of the permuted C_j is included in Q_k if and only if $b_{i,k} = 1$. For the permutation of C_{10} in the previous step, Q_0 would be {senator, foe, queen, chairman, mayor, ally, assistant, supporter}, and Q_1 would be {enemy, foe, president, chairman, friend, ally, manager, supporter}. Since the entry *leader* has index $i = 0$ in the permutation of this example session, it does not appear in any of the Q_k. See Figure 1 for the challenges of this session, which are displayed in random order (as opposed to alphabetical order) as CAPTCHAs for added security against sophisticated malicious software (the random re-ordering as well as the CAPTCHA representation are not needed if there is no threat of such an adversary).

3. For $k = 3, 2, 1, 0$ in turn, the system displays Q_k to the user who answers "Yes" if the mnemonic word m_j (corresponding to the current column C_j) is in Q_k, and answers "No" otherwise.

We note that (i) the user's answers uniquely identify to the server the mnemonic word in each column; (ii) the total number of questions is logarithmic in the size r of each column, so that password security can be increased by a factor of 2^μ by doubling the size of a column yet adding only 1 extra question per column (and, more importantly, without any increase in the size of the mnemonic, i.e., without further burdening the user's memory); (iii) a *shoulder-surfer* adversary sees the questions but not the user's yes/no answers (hence learns nothing); (iv) that a *keystroke-logger* sees the answers but cannot use them to authenticate itself or to obtain the passwords unless it can relate these answers to the challenges (which are preferably obfuscated as in the figure); (v) that a *phisher* adversary does not even know what questions to display, immediately alerting the user that something seriously phishy is going on (even if the phisher got a user to respond to very unfamiliar challenges, those responses are useless to such an attacker).

4 Implementation Details

We assume that the environment enables the user to read (or hear) the challenges displayed on the screen and the user can input the yes/no answers through a switch. The system includes a large set of tables, S, which are already populated offline. These tables, such as Table 1, are used for generating mnemonic sentences and challenges. Each table has a unique ID. Every table corresponds to a source sentence from a corpus, and these source sentences are stored in the first row of the table. Table 1 was generated using an example source sentence "Leading U.S. couturiers are strongly resisting pressure to regulate the thinness of the popular models." Every column in this table shows a possible candidate word set for replacing the original word in the first row (functional words are excluded).

4.1 Mnemonic Creation

At mnemonic-creation time, the system first generates a random password, $\mathcal{P}$, for the user (e.g. a random string of 40 bits), or the user's existing password is used. Next step is generating the possible candidates for mnemonic sentences that will encode this password.

If the source sentence has 10 words as in our example sentence, and each of these words have 16 alternatives; a random password chooses one word out of each of these 16 alternative words, hence encodes 4 bits per word, 40 bits in total.

The system selects the words that encode $\mathcal{P}$. This process generates one candidate mnemonic sentence per such table. All of the candidate mnemonic sentences encode the same $\mathcal{P}$.

At the end, the user is provided with a set of candidate mnemonic sentences to pick from and the random password, $\mathcal{P}$, to use in a keyboard setting if needed.

Mnemonic creation concludes with the user's selection of one of the candidate mnemonic sentences for remembering as a mnemonic.

Once the user selects which mnemonic sentence to use, the ID of the corresponding table that generated it is recorded in the least significant $\log_2 |S|$ bits of the salted hash of the password file entry.

Since we have many source sentences (say 1024 of them), the user can choose from 1024 different mnemonic sentences generated for each source sentence. However there might be psychological attacks [8] to such a flexible system; hence we advise only a small random portion of these possible mnemonics be given as choice to the users.

4.2 Mnemonic Usage

The mnemonic sentence is not stored in the system, instead the source table ID is stored in the salted password hash. The authentication involves a conversion of the yes/no answers of the user into a password.

We achieve this by generating the challenges in such a way that every yes/no answer narrows the search space by one-bit, similar to the idea behind the "20-Question Game". In our scheme, instead of looking for one object, we are searching for a password that is composed of concatenation of several substrings, each of which is encoded by a different word of the mnemonic sentence. Each mnemonic word is a member of an equivalence

	leading	U.S.	couturiers	strongly	resist	pressure	regulate	thinness	popular	models
0000	peaceful	viking	tailor	alarmingly	welcome	attempt	modify	rent	passive	queen
0001	thoughtful	romanian	cartoonist	hardly	agree	haste	alter	wisdom	inept	leader
0010	rich	city	beekeeper	suddenly	reject	duress	cement	culture	able	senator
0011	uninterested	rural	realist	simply	embrace	pressure	manipulate	education	dull	supporter
0100	provoked	irish	firefighters	warily	resist	demand	secure	diligence	hot	king
0101	angry	suburban	artist	doubtfully	renounce	bid	fix	weakness	skilled	ally
0110	outraged	texan	architect	remarkably	submit	call	quantify	salary	adept	foe
0111	neutral	aussie	police	again	honor	ultimatum	measure	pension	dormant	manager
1000	furious	canadian	cubist	blindly	recognize	struggle	forgive	thinness	crazy	friend
1001	poor	union	farmer	suspiciously	allow	operation	change	obedience	gifted	president
1010	average	british	fantasist	delicately	accept	order	limit	laziness	bright	enemy
1011	determined	european	developer	fiercely	surrender	imperative	throttle	spirit	witless	children
1100	strong	downtown	farmer	repeatedly	tolerate	hurry	harness	tenuity	exhausted	associate
1101	calm	urban	goldsmith	reluctantly	permit	insistence	deregulate	slenderness	talented	mayor
1110	silent	italian	musician	discreetly	refuse	ban	restrict	citizenship	clumsy	chairman
1111	ordinary	french	drivers	slowly	dismiss	decree	fiddle	discipline	sharp	assistant

Table 1: The mnemonic generation table for the sentence "Leading U.S. couturiers are strongly resisting pressure to regulate the thinness of the popular models." The order of words within a column is randomly determined.

class, and we need to ask several questions that will deterministically find the exact mnemonic word within a class. We ask $\log_2 r$, (e.g., 4), questions to determine one mnemonic word, where r, (e.g., 16), is the number of words in an equivalence class.

The key idea behind generating each challenge is very similar to the idea behind *non-adaptive blood testing* technique [9]. The area of combinatorial group testing concerns itself with performing group tests on subsets of a given set to identify defective elements in that set: A test for a subset tells whether that subset contains a defective element. If the set size is r and the number of defective elements is no more than d, then the goal is to pinpoint all the defective elements by making as few group tests as possible. The original problem was adaptive in the sense that test $i+1$ could be designed after the outcome of test i was known, thereby enabling a simple binary search for the defective element in the special case of $d = 1$. The non-adaptive version of the problem is when all the tests are done in a single round, with all the subsets to be tested determined in advance.

The analogy with our problem is as follows: For each mnemonic word m_j, the r "blood samples" are the r words in m_j's equivalence class. The mnemonic word is like the contaminated blood sample. The server presents the user with a subset of words from m_j's equivalence class, C_j, (possibly containing m_j) and the user is supposed to respond yes or no based on whether m_j is in that subset (i.e., whether that subset is "contaminated"). The server tests subsets in a manner that enables it to uniquely identify m_j, and then the server does a table lookup (local to the server) to derive the password bit string associated with m_j.

To prevent the adversary from learning anything by using the questions, it is imperative for the server to use a *non-adaptive* technique whereby *all* the questions have been pre-determined well in advance, as in non-adaptive combinatorial group testing. The questions are therefore independent of which item in the set is the "contaminated" one, and hence they reveal nothing to the adversary who sees them. Using adaptive group testing techniques (like binary search) for determining the questions would be lethal from the security point of view.

Our scheme will use $d = 1$, for which an $\ell = \lceil \log_2 r \rceil$ test non-adaptive solution is well known and in fact quite straightforward. We briefly sketch it, for the sake of making this paper self-contained. In what follows C_j is the equivalence class of word m_j, where $|C_j| = r$. (Recall the authentication step described in Section 3)

1. Let the words in C_j be listed in an order (which will be randomly changed at every authentication session) as $w_1, \ldots, w_\mu$.
2. For each word w_i, let the ℓ bit binary representation of i be denoted as the bit string $b_{i,\ell-1}, \ldots, b_{i,0}$.
3. For $k = 0, \ldots, \ell - 1$ in turn, the server's question Q_k is constructed as follows: Every w_j whose $b_{i,k} = 1$ is included in Q_k.

The server asks ℓ questions, and each question is constructed without any dependency on which element of C_j is the "contaminated" one, m_j. The server can easily determine m_j: It is the only word of C_j such that all of the Q_k's that contained it were answered with a "yes" by the user. Note that, this scheme is not restricted to inputting passwords using mnemonics, and it can be used to input plain text passwords when the challenges contain single ASCII symbols.

When the equivalence classes have a size of 16 as in our example, each challenge will have 8 words and the user will be asked 4 questions. An example question for finding the mnemonic word in the last column of Table 1 would be as follows:"Does your mnemonic sentence contain one of the following words?: { senator, foe, queen, chairman, mayor, ally, assistant, supporter }". The user answers 40 such questions in total (4 for each one of the 10 mnemonic words) with a "yes" or a "no" signal using the switch (equivalently with 1 for "yes" or 0 for "no").

After the answers are collected, the system extracts each password substring encoded by the mnemonic words and concatenates them in the order corresponding to the order of words in the source sentence to form the password $\mathcal{P}$. The hash of $\mathcal{P}$ with the salt is compared to the hash value kept in the password file (where the hash value is stored as in the regular UNIX password file). Note that the same password file can still be used with the ASCII passwords.

Our current password size of 40 bits falls short of the 52 bits commonly used in deployed systems, but we are confident that we will be able to exceed the 52 bits in the continuation and further refinement of this work. In the meantime, even in its present form our current implementation is suitable for use as a front-end to a 52-bit password system: We would use our system for entering 40 of the 52 bits, and the missing 12 bits would be handled as private salt in a similar fashion to what was described in [12] – by the front-end essentially trying all 2^{12} possibilities for the remaining 12 bits. The password file would stay the same as before our system was deployed, as would the password: We act only as a front end, when normal keyboard entry is either impossible or risky.

5 Related Work

Previous studies state the requirements for increasing the accessibility of electronic resources for the disabled users [1, 6, 5, 13, 2, 4] and suggest possible techniques to increase the bandwidth of input from these users [10, 7]. There is also a body of work for providing access to web through smaller devices which have limited input capabilities [3, 18]. These two research areas have a considerable overlap in the design requirements such as assumption of low input bandwidth, emphasis on usability, and the need for platform independence. Trewin discusses the overlap between the accessibility requirements for desktop browsers for Web, and the requirements for a usable Mobile Web in [18].

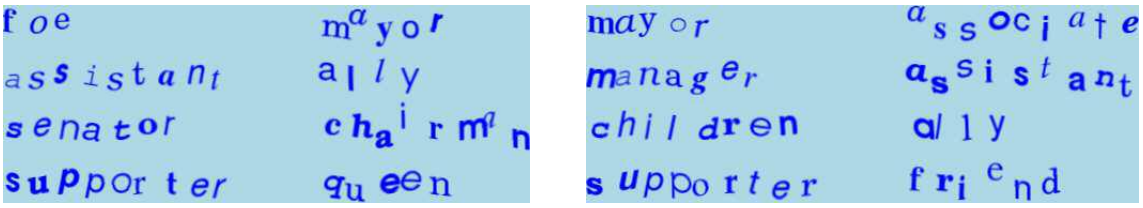
Mankoff et al. discuss the needs of motor disabled users for accessing the web in [13]. They study scenarios where the users control a switch through simple muscle movements such as raising an eye brow [10], or Brain Computer Interface (BCI) [19].

Pass-thoughts system [15] is based on recognition of unique brain signals send by the users. Thorpe et al. list the following set of requirements for an authentication system: i) changeability; ii) shoulder-surfing resistance; iii) theft protection (e.g. through acoustic attacks, or brute force attacks); iv) protection from user noncompliance; and, v) usability. The authors also present an authentication scheme that is solely based on training a user to think about the same idea (e.g. a place, a thing, or a melody), and recording the repeatable parts of the brain signal features extracted from this "pass-thought". Thorpe et al. reported that the BCI technology, that was available at the time of their study (September 2005), enabled the users to input approximately 25 bits per minute, which was not sufficient to provide enough bandwidth for the implementation of their sophisticated authentication scheme.

In 2004, Yan et al. conducted a controlled experiment to compare the effects of giving three alternative forms of advice about password selection [20]. The results of this study gave very valuable hints for designing a usable and secure authentication system: i) users have difficulty remembering random passwords (students in the group that was asked to write down their random passwords continued to carry the written copy for 4.8 weeks on the average.), ii) passwords derived from mnemonic phrases are indeed harder for an adversary to guess than naively selected passwords, iii) it is equally easy to remember strong passwords derived from mnemonic phrases and naively selected weak passwords.

Jeyaraman and Topkara proposed a system to increase the usability of text password authentication by automatically generating mnemonic sentences which help the users in remembering truly random passwords [11]. A more recently introduced mnemonic scheme for text password authentication by Topkara et al. [16] allows the users to maintain a multiplicity of truly random passwords, which are independently selected, by remembering only one mnemonic sentence. An adversary who breaks one of the passwords encoded in the mnemonic sentence does not gain information about the other passwords.

Note that in both schemes, [11, 16], the mnemonics are used as an aid to remember text passwords, whereas the current paper enables the use of the mnemonic sentence to serve as the password itself. In the current paper our main challenge is to construct an authentication mechanism that can work in restricted environments. We present a suggested mode of use for other mnemonic password schemes that use other media mnemonics including graphics, and audio besides text. The scheme in this paper provides resistance to phishing, to keystroke-



Q_0 ... Q_3

Figure 1: One of the possible set of challenges Q_0 to Q_3 that could be created for the last column of Table 1. The challenge words are presented in random order and in the form of CAPTCHAs for added security against sophisticated key-loggers. In the absence of such concerns the challenges can be displayed as text images in alphabetical order.

logging, to shoulder surfing as well as to dictionary attacks.

[14] presents a method that cleverly uses CAPTCHAs for assuring that an automated adversary will need more than a pre-determined amount of time to break a password through repeated login attempts.

6 Conclusion

We presented a password authentication system that is suitable for use in input-constrained environments, and that has many security and password-mnemonic advantages over existing keyboard-based schemes. Because of its compatibility with existing systems (to which it can act as a front-end), it can be used in an intermittent fashion alongside these existing systems: A user may prefer to use the normal keyboard entry most of the time (e.g., at home and in the office) but occasionally switch to using our system in certain situations, such as when the user fears the presence of shoulder-surfers or surveillance cameras, or has a temporary wrist injury that prevents the use of a keyboard, etc.

7 Acknowledgements

Portions of this work were supported by Grants IIS-0325345 and CNS-0627488 from the National Science Foundation, and by sponsors of the Center for Education and Research in Information Assurance and Security. The authors would like to thank the four anonymous reviewers for their helpful feedback and suggestions.

References

[1] Information technology accessibility and workforce-division, section 508: The road to accessibility, 1998.

[2] Australian banker's association inc., guiding principles for accessible authentication, Accessed on 4 December 2006.

[3] W3c mobile web initiative, Accessed on January 10, 2006.

[4] W3c web accessibilty initiative, Accessed on January 10, 2006.

[5] BBC-NEWS. Most websites failing disabled, Published on 2006/12/05.

[6] BROWN, C. Assistive technology computers and persons with disabilities. *ACM Communications* (1992).

[7] COPESTAKE, A. Applying natural language processing techniques to speech prostheses. In *Working Notes of the 1996 AAAI Fall Symposium on Developing Assistive Technology for People with Disabilities* (1996).

[8] DAVIS, D., MONROSE, F., AND REITER, M. K. On user choice in graphical password schemes. In *13th USENIX Security Symposium* (2004).

[9] DORFMAN, R. The detection of defective members of large populations. *The Annals of Mathematical Statistics* (1943).

[10] GRAUMAN, K., BETKE, M., LOMBARDI, J., GIPS, J., AND BRADSKI, G. Communication via eye blinks and eyebrow raises: video-based human-computer interfaces. *Universal Access in the Information Society* (2003).

[11] JEYARAMAN, S., AND TOPKARA, U. Have the cake and eat it too – infusing usability into text-password based authentication systems. In *21st Annual Computer Security Applications Conference* (2005).

[12] MANBER, U. A simple scheme to make passwords based on one-way functions much harder to crack. *Computers and Security* (1996).

[13] MANKOFF, J., DEY, A., BATRA, U., AND MOORE, M. Web accessibility for low bandwidth input. In *5th International ACM Conference on Assistive Technologies* (2002).

[14] PINKAS, B., AND SANDER, T. Securing passwords against dictionary attacks. In *ACM Computer and Security Conference* (2002).

[15] THORPE, J., VAN OORSCHOT, P. C., AND SOMAYAJI, A. Passthoughts: Authenticating with our minds. In *Workshop on New Security Paradigms* (2005).

[16] TOPKARA, U., ATALLAH, M. J., AND TOPKARA, M. Passwords decay, words endure: Secure and re-usable multiple password mnemonics. In *ACM Symposium on Applied Computing* (2007).

[17] TOPKARA, U., TOPKARA, M., AND ATALLAH, M. J. Passwords for everyone: Secure mnemonic-based accessible authentication. Tech. Rep. CSD TR #07-008, Purdue University, 2007.

[18] TREWIN, S. Physical usability and the mobile web. In *International Cross-disciplinary Workshop on Web Accessibility* (2006).

[19] VAUGHAN, T., HEETDERKS, W., TREJO, L., RYMER, W., WEINRICH, M., MOORE, M., KUBLER, A., DOBKIN, B., BIRBAUMER, N., DONCHIN, E., WOLPAW, E., AND WOLPAW, J. Brain-computer interface technology: a review of the second international meeting. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (2003).

[20] YAN, J., BLACKWELL, A., ANDERSON, R., AND GRANT, A. Password memorability and security: Empirical results. *IEEE Security and Privacy* (2004).