Spelling-Error Tolerant, Order-Independent Pass-Phrases via the Damerau-Levenshtein String-Edit Distance Metric

Gregory V. Bard

Dept of Applied Mathematics and Scientific Computation, University of Maryland at College Park.
gregory.bard@ieee.org

Abstract

It is well understood that passwords must be very long and complex to have sufficient entropy for security purposes. Unfortunately, these passwords tend to be hard to memorize, and so alternatives are sought. Smart Cards, Biometrics, and Reverse Turing Tests (human-only solvable puzzles) are options, but another option is to use pass-phrases.

This paper explores methods for making pass-phrases suitable for use with password-based authentication and key-exchange (PAKE) protocols, and in particular, with schemes resilient to server-file compromise. In particular, the Ω-method of Gentry, MacKenzie and Ramzan, is combined with the Bellovin-Merritt protocol to provide mutual authentication (in the random oracle model (Canetti, Goldreich & Halevi 2004, Bellare, Boldyreva & Palacio 2004, Maurer, Renner & Holenstein 2004)). Furthermore, since common password-related problems are typographical errors, and the CAPSLOCK key, we show how a dictionary can be used with the Damerau-Levenshtein string-edit distance metric to construct a case-insensitive pass-phrase system that can tolerate zero, one, or two spelling-errors per word, with no loss in security. Furthermore, we show that the system can be made to accept pass-phrases that have been arbitrarily reordered, with a security cost that can be calculated.

While a pass-phrase space of 2^{128} is not achieved by this scheme, sizes in the range of 2^{52} to 2^{112} result from various selections of parameter sizes. An attacker who has acquired the server-file must exhaust over this space, while an attacker without the server-file cannot succeed with non-negligible probability.

Keywords: Passwords, Password-Based Authentication and Key Exchange (PAKE), Damerau-Levenshtein String-Edit Distance Metric, Usable Security.

1 Introduction

The purpose of this paper is to create a scheme whereby a short list of words, from a natural alphabet-based language (such as English), can serve as a measurably secure password for use in a password-based authentication or key-exchange scheme. In particular, the pass-phrase will consist of about 4 to 7 words chosen from a list of 2^{13} to 2^{16} words, and will be tolerant of zero, one, or two spelling errors per word, any arbitrary reordering of the words, and any choice of capitalizations. Pass-phrases must contain words from a large dictionary to be secure, and so may contain many unfamiliar or bizarre words, including ones that users may find hard to spell correctly. The error-corrections which bring about these tolerances are designed to make pass-phrases less irritating to the common user. Since these pass-phrases are entirely machine generated, a very high entropy is achievable, and security parameters can be chosen to make the scheme resistant to dictionary-based attacks.

In Section 1.1 we discuss the need for a large password space in the presence of adversaries who can achieve server-file compromise. In Section 2.1 we introduce the Damerau-Levenshtein distance metric, and provide further details in Appendix A. We describe the intended implementation in Section 2.3 and give the details of the protocol in Section 3. We describe how the spelling-error correction is done in Section 4, correction of reordering in Section 6.1 and how to build the dictionary in Section 5. We show how to select correct parameter sizes in Section 6 and describe a scheme for giving the user more choice in the pass-phrase selection in Section 6.2. We mention an option for "duress" words in Section 6.3, to protect users who are forced to log-in by physical adversaries. We conclude with an additional application (a "password bank") in Section B.

1.1 Motivation

Recently, the classical problem of password-based authentication has received a great deal of attention, both in the cryptographic community and the security world in general (Jablon 1996, Jablon 1997, Lucks 1997, Boyko, MacKenzie & Patel 2000, Bellare, Pointcheval & Rogaway 2000, MacKenzie, Patel & Swaminathan 2000, Katz, Ostrovsky & Yung 2001, Goldreich & Lindell 2001, Canetti, Halevi, Katz, Lindell & MacKenzie 2005). There are now several protocols available that deliver distinct services in various security models, including (Bellovin & Merritt 1992, Gong, Lomas, Needham & Saltzer 1993, Gong 1995, Ford & B. Kaliski 2000, Kaufmann & Perlman 2001, Kwon 2001, MacKenzie 2001, Jiang & Gong 2004, Zhang 2004, Nguyen & Vadhan 2004, Katz, MacKenzie, Taban & Gligor 2005, Gentry, MacKenzie & Ramzan 2000). Most schemes involve a server, to which the client must prove his/her identity, by providing a password. Many provide for mutual authentication as does this paper. However, all of these schemes share a common limiting factor.

The server must store some information for each user in order to engage in authentication, particularly if users are not to be permitted to log in as each other. This information, whether in one file or several, will be called the server-file. Sometimes the server-file is public (as in UNIX), and sometimes private (as in Bellovin-Merritt (Bellovin & Merritt 1992)). However, the capture or unauthorized release of this file can occur in practice, and so it is desirable to design systems resilient to server-file compromise.

In the event of a server-file compromise, the attacker is always able to faithfully simulate both parties of the client-server authentication protocol, and guess all the possible passwords (Narayanan & Shmatikov 2005). Therefore the security of the scheme is lower-bounded by the size of the password-space, in the case of server-file compromise. This bound is tight, as achieved by (Gentry, MacKenzie & Ramzan 2006, Bellovin & Merritt 1993, Jablon 1997, Wu 1998, Kwon 2001), who show that no attack faster than this exists. Of course, there exist other schemes which are completely insecure in the event of server-file compromise. Alternatively, a threshold-based approach is also possible (DiRaimondo & Gennaro 2003, Jablon 2001, MacKenzie, Shrimpton & Jakobsson 2006).

The April 2006 NIST recommendations (Burr, Dodson & Polk 2006) state that current user password choices "in the real world" have entropy between 18–30 bits, guaranteeing a successful brute-force attack by a PC in a short time (see also (Burr n.d.)). Likewise the survey (Patrick n.d.) states that 26% of passwords (47,642) out of 186,126 found stored in a particular cache could be guessed by a password-guessing tool. That survey also notes that the average password reset occurs once every four or five months, or that a corporation of 100,000 users should anticipate 20,000–25,000 password resets per month.

Typical remedies are to add complexity via more complex passwords, or switch to pass-phrases, biometrics, or Reverse Turing Tests (RTT's), such as human-only solvable puzzles (Canetti, Halevi & Steiner 2006, Pinkas & Sander 2002). While biometrics and RTT's have advantages, they are not always possible. Long and complex passwords involving symbols, numbers, and so forth, are rarely chosen by users (Adams & Sasse. 1999, Wu 1999), and it is hoped that the pass-phrases created by the scheme in this paper will be more palatable for the user.

Another important factor are alarm levels. It is considered a proper security practice to lock an account after 3–5 bad password guesses, and this is effective at preventing dictionary attacks in the absence of server file compromise. Nonetheless, as password systems get more complicated, making three typographical errors in three entries is quite a possibility, even in the absence of forgetfulness. For this reason, spelling correction is desirable.

It should be noted that error-correcting passwords were proposed by Dodis, Reyzin and Smith in 2004 (Dodis, Reyzin & Smith 2004), in the context of fuzzy-extractors. However, their scheme requires very large entropy domains (256+ bits in some cases), and human-readable English phrases would need to be quite long to provide for that (about 14 words in length).

2 Preliminaries

2.1 The Damerau-Levenshtein Distance Metric

The Damerau-Levenshtein distance metric arose from research by Damerau and Levenshtein on spelling-errors (Damerau 1964, Levenshtein 1966).

The Damerau-Levenshtein distance metric is a function, from finite strings drawn from an alphabet, to the integers. It is a distance metric in the sense that given strings s_1, s_2, s_3, the following conditions apply.

- Non-negativity: $d(s_1, s_2) \geq 0$.
- Non-degeneracy: $d(s_1, s_2) = 0$ if and only if $s_1 = s_2$.
- Symmetry: $d(s_1, s_2) = d(s_2, s_1)$.
- Triangle Inequality: $d(s_1, s_2) + d(s_2, s_3) \geq d(s_1, s_3)$.

The distance $d(s_1, s_2)$ is defined as follows. One "operation" can be inserting a character, deleting a character, substituting one character for another in one location, or transposing two adjacent characters. There may be many combinations of these four operations that can convert the string s_1 to s_2, but the length of the shortest sequence is the distance between the two strings. It should be noted that the four operations can be given weights other than one, but we do not consider that here. Damerau showed that 80% of typographical errors are distance 1 in this model (Damerau 1964).

A highly related-metric is the Levenshtein metric (Levenshtein 1966), which excludes the transposition, and is frequently used in spelling checkers. Note that a transposition can be thought of as an insertion followed by a deletion, and so transposition errors are still covered by the Levenshtein metric. The Levenshtein metric is much easier to compute, where $d(a, b)$ can be computed in time $O(|a||b|)$, with several techniques available (Wagner & Fisher 1974, Ukkonen 1985) for acceleration in the case of calculating many $d(a, b)$ for the same a and all b in a dictionary (as would be the case in this paper's scheme). However, since transposition errors are probably very likely in entering a password, we elect to use the Damerau-Levenshtein metric instead. The complexity of computing this metric is $O(|a||b|max(|a|, |b|))$ with the naive algorithm, and $O(|a||b|)$ with the algorithm of Lowrance and Wagner (Lowrance & Wagner 1975), given in Appendix A.

We prove the four metric properties in the Appendix A, and discuss the algorithms used to compute the distance and their complexities. Also in Appendix A.3 are some empirical tricks to accelerate calculation in the special case of this problem.

2.2 Notation

The pass-phrases will consist of n words from a dictionary D of size $|D| = 2^c$, where c is some positive integer approximately near to 14. Using the notation of the Ω-method (Gentry et al. 2006), the function $H'(x)$ is a hash function with the additional property of behaving as a random oracle (and therefore is also a one-way function). The function $H(x)$ must also behave like a random oracle, but must have output-length equal to cn, which is much shorter than typical hash function output lengths. Therefore, $H(x)$ is recommended to be a truncation of $H'(x)$ to the length of cn bits. The functions $E_k(m)$ and $E'_k(m)$ are given by

$$\begin{aligned} E_k(m) &= H(k) \oplus m \\ E'_k(m) &= H(k) \oplus m || H'(m) \end{aligned}$$

Furthermore, observe that $E_k(m)$ is essentially a one-time pad, since $H(k)$ is a random oracle, and $E'_k(m)$ is a slight modification to render the scheme non-malleable. Therefore, it is pivotal that a single k is used at most once, namely by using one call to $E_k(m)$, and never using k for any other purpose.

2.3 Implementation

First, create a dictionary of selected words from the English[1] (or any other) language, augmented by personal names and place names. We discuss how to build such a dictionary in Section 5. Each user would receive, in person, a short list of $n \approx 5$ words from the dictionary, which they must memorize and then destroy in the presence of the system administrator, preferably after some trial log-ins. This list of words will be their pass-phrase.

The user is not permitted to choose his/her own pass-phrase, as it would be hard to make entropy guarantees in that case. However, we present a scheme whereby a user could choose from among a list of 4, 8, or 16 randomly chosen pass-phrases (See Section 6.2).

The log-in protocol will handle a potential pass-phrase entry as follows. First, spelling-error and reordering repairs will be applied (to be explained in Sections 4 and 6.1). If any word of the potential phrase is still found not to be in the dictionary, the attempt is rejected. Otherwise, the phrase is converted to a binary string either with a hash function, or the function g in the next section. Next, the Bellovin and Merritt protocol is applied, which provides for mutual authentication. Finally, this is amplified by the Ω-method from Gentry, MacKenzie, and Ramzan (Gentry et al. 2006), which provides for resilience in the event of server-file compromise.

Alternatively, this scheme can be used to protect a password-bank, where all the user's passwords are stored (much like modern web browsers do). This is explained further in Appendix B. Bellovin and Merritt also give an application of PAKE to secure public encrypted phones (Bellovin & Merritt 1992).

Note that since all calculations are done on words, and not on characters (except the metric which we consider as a black-box), the system can be case-insensitive and there are no capitalization considerations, and therefore we will not mention capitalization again.

3 Overview

Knowing a secret string of uniformly generated random bits, of length cn is equivalent to knowing n strings, in some specific order, of length c bits, provided all the strings are again uniformly randomly generated. Consider a dictionary D of words of the English (or any other) language, of size $|D| = 2^c$. One could number the words from 0 to $2^c - 1$, and create bijections f and g such that $f(i)$ is the ith word in the dictionary, and $g(w_i) = i$ when w_i is the ith word. Since bijections preserve entropy according to the data-processing inequality, knowing n such words, in some specific order, is equivalent to knowing a cn bit secret key.

This pass-phrase issuing protocol is given in Table 1. Steps 6, 7, and 8, are from the Ω-method (Gentry et al. 2006). The public-private key mentioned in Step 7 is for the signature scheme used in the pass-phrase verification algorithm, Steps 12 and 13, as given in Table 2.

The dictionary D is public, thus the attacker can always make guesses that use correct spelling, and use words only from the dictionary. Therefore, any spelling-error correction would only aid the attacker if it could transform one dictionary word into another. We will show this is impossible in our scheme. Errors

1	User enters username u.
2	Randomly generate $x \leftarrow \{0,1\}^n c$.
3	Divide x into $x_1 \| x_2 \| \cdots \| x_n$.
4	Let $w_i = f(x_i)$.
5	Display($w_1 \| w_2 \| \cdots \| w_n$).
6	Let $h = H(w_1 \| w_2 \| \cdots \| w_n)$.
7	Generate a public-private key pair (pk, sk).
8	Store $(u, h, pk, E_x(sk))$.

Table 1: Pass-Phrase Issuing Protocol

made by reordering of the words will be dealt with in Section 6.1.

4 Error Correction

As mentioned earlier, the dictionary will consist of 2^c words from the English (or any other) language, with the additional requirement that any two distinct words from the dictionary must be distance five or greater in the Damerau-Levenshtein distance metric. We will discuss how to construct such a dictionary in Section 5.

The user will input n words. For each of these words, either there exists a word in the dictionary which is distance two or less away, or there does not exist such a word. If no such word exists, then the log-in is rejected immediately. If such a word exists, we claim it is unique.

Suppose the user entered the word e and the distinct words w_1, w_2 were both distance two or fewer from e. Since $d(w_1, e) \leq 2$ and $d(w_2, e) \leq 2$, then we know

$$d(w_1, e) + d(e, w_2) \leq 4$$

but from the triangle inequality, we know that

$$d(w_1, e) + d(e, w_2) \geq d(w_1, w_2)$$

and therefore $d(w_1, w_2) \leq 4$. It is a rule of our dictionary that no two distinct words are closer than distance five. This is a contradiction. Therefore, no two dictionary words can both be distance two or closer to some user-entered string. Alternatively, this means if one dictionary word is distance two or less from an entered string, all the other words must be distance three or greater away. Finally, this means that so long as only zero, one, or two errors are made per word, the errors can be corrected to their unique closest dictionary member.

A last note is that since only corrections of distance zero, one, or two are made, it is impossible to change one dictionary word into another, as that would require distance five. This meets the requirements as specified in Section 3.

The protocol that this forms, when coupled with the Ω-method, is given in Table 2. Step 5 is optional (for correcting reorders), and is explained in Section 6.1. Steps 8 . . . 13 are from the Ω-method. In Step 8, note E and E' are distinct encryption schemes defined in (Gentry et al. 2006). The phrase "execute key agreement" can indicate either Bellovin-Merritt (Bellovin & Merritt 1992), PAK (MacKenzie 2002), or any other secure password-based symmetric mutual authentication. The Ω-protocol surrounding it adds the resilience to server compromise.

The need for Step 7 is explained in (Gentry et al. 2006), but essentially K'' is the key generated for future communications between the two parties, whereas K' is used to finish the protocol. In short, if one key were used for both, the one-time pad properties of E would be compromised. Also note that the server never stores sk, but rather $E'_x(sk)$. Instead, sk

[1]It should be noted that highly agglutinative languages such as Finnish, Estonian and Hungarian are problematic with these metrics, as every possible declination of every noun must be considered a distinct word, giving rise to tens of millions of words in the dictionary (Schulz & Mihov 2002).

```
1    User enters username $u$.
2    User enters pass-phrase $w'_1, \ldots, w'_n$.
3    Capitalize the $w'_i$'s.
4    For each $w'_i$ do
4a       If $\exists w \in D$ such that $d(w'_i, w) \leq 2$
4b       Then $w''_i = w$.
4c       Else abort.
5    (Optional) Alphabetize $w''_1, \ldots, w''_n$.
6    For $i = 1 \ldots n$, let $x'_i = g(w''_i)$.
7    Let $x' = x'_1 || x'_2 || \cdots || x'_n$.
8    Execute key agreement with
     $h = H(w_1 || w_2 || \cdots || w_n)$,
     obtain key $K$.
9    Let $K' = H(K)$ and $K'' = H(H(K))$.
10   Server challenges with $c = E_{K'}(E'_{x'}(sk))$.
11   Let $sk = D'_{x'}(D_{K'}(c))$.
12   Let $s = Sign_{sk}(\text{Transcript})$.
13   Server checks that $\text{Verify}_{pk}(\text{Transcript}, s)$.
```

Table 2: The Pass-Phrase Verification Protocol.

is destroyed after the pass-phrase issuing protocol is run, and pk is used to verify any signatures generated by sk. It is this feature that makes the Ω-method immune to server-file compromise.

5 Constructing the Dictionary

Consider a word corpus W, consisting of all words of a particular language, including every possible conjugation and declination. In our case, this list would be augmented by city names and personal names. Finding the largest subset of words D, such that each word in D is at least distance 5 apart from every other word in D, is difficult. In particular, if W forms the vertices of a graph, we can draw an edge between two words if and only if they are distance 4 or closer in the metric. The dictionary D would then be a maximum independent set, or set of vertices such that there is no edge between them. Unfortunately, this is an NP-Complete problem (Karp 1972), and it is known that an approximation algorithm (which guarantees to find an independent set at least $0 < c < 1$ times the size of the maximal one) can only exist if $P = NP$ (Garey & Johnson 1979). We do not claim the dictionary building problem is NP-Complete however, as a faster method may exist, compared to the graph-theoretic independent set problem.

Hoping to find a dictionary of size $|D| \approx 2^{13} = 8192$, we attempted the following very naive approach. Using the file `/usr/share/dict/linux.words` from Fedora Linux 5.0, which has 479,625 words, we read one word at a time, and added it to D if and only if it was distance five away from all the current members of D (checked by exhaustion) and free of non-alphabetic characters. This method produced a dictionary of size $34,538 > 2^{16}$, which is more than ample. However, some of these words are uncommon, like "abarticular," "draughtmanship", and "galaxiidae." The calculation took several hours on a desktop PC.

6 Selecting Security Parameters

While a dictionary of size 2^{16} was achieved, many of these words are unfamiliar. Intersecting this dictionary with a codex from a major newspaper or some other method may produce a smaller dictionary, with the requisite properties, of size $\geq 2^{13}$, but with fewer bizarre words. With dictionaries of size 2^{13} to 2^{16}, and between 4 and 9 words per pass-phrase, it is easy to calculate the effective size of the pass-phrase space

Re-ordering Not Used

Dictionary Size	2^{13}	2^{14}	2^{15}	2^{16}
4 Words	2^{52}	2^{56}	2^{60}	2^{65}
5 Words	2^{65}	2^{70}	2^{75}	2^{80}
6 Words	2^{78}	2^{84}	2^{90}	2^{96}
7 Words	2^{91}	2^{98}	2^{105}	2^{112}
8 Words	2^{104}	2^{112}	2^{120}	2^{128}
9 Words	2^{117}	2^{126}	2^{135}	2^{144}

Re-ordering Used

Dictionary Size	2^{13}	2^{14}	2^{15}	2^{16}
4 Words	$2^{47.4}$	$2^{51.1}$	$2^{55.1}$	$2^{59.1}$
5 Words	$2^{58.1}$	$2^{63.1}$	$2^{68.1}$	$2^{73.1}$
6 Words	$2^{68.5}$	$2^{74.5}$	$2^{80.5}$	$2^{86.5}$
7 Words	$2^{78.7}$	$2^{85.7}$	$2^{92.7}$	$2^{99.7}$
8 Words	$2^{88.7}$	$2^{96.7}$	$2^{104.7}$	$2^{112.7}$
9 Words	$2^{98.5}$	$2^{107.5}$	$2^{116.5}$	$2^{125.5}$

Table 3: Effective Pass-Phrase Spaces for various Parameter Values.

if reordering is not used. If it is used, one must adjust as explained below. The results are listed in Table 3. Note, that $n = 9$ or even $n = 8$ is not recommended as this may be too long, except perhaps for super-user passwords.

The famous paper (Miller 1956), showed that sets of 5, 7, or 9 objects are particularly easier for human beings to memorize than any other sizes, giving rise to the "rules of thumb" 7 ± 2. In particular, breaking a set of 7 into a set of 3 and a set of 4 is particularly effective, and for these reasons, phone numbers in the USA were accordingly designed, first with 5 and later with 7 digits.

Finally, note that a 4, 5, or 6, digit PIN could be used to augment the pass-phrase, adding 13.3, 16.6, or 19.9 bits of security. Some users might prefer a 4 word pass-phrase augmented by a 4 digit PIN over a 5 word pass-phrase.

6.1 Dealing with Word Re-orderings

Normally, typing the words of a pass-phrase out of order is considered an incorrect log-in. But, on the other hand, if one alphabetizes the words of the pass-phrase after error-correction is applied, but before hashing, then any reordering of the words by the user will have no effect. This unfortunately reduces the size of the pass-phrase space. Since there are $n!$ ways to order n words, the security lost is $\log_2(n!)$. However, the gain to the user is easier memorization, or lower probability of error. Perhaps high-security systems will prefer to omit alphabetization, while systems with more inexperienced users will prefer to use it.

6.2 Giving the User a Choice of Several

If the user is enabled to select their own pass-phrase, it becomes very difficult to guarantee the entropy of these selections. However, computer-generated passwords will almost certainly be harder to memorize. To mitigate this, the password generation protocol may, for example, generate $u = 4$ or $u = 8$ pass-phrases, and enable the user to pick one of these.

The security lost by this flexibility is difficult to calculate. Suppose that there is a subset S such that if a pass-phrase from S is displayed, that the user will always choose the pass-phrase from this "easy subset." If $|S|$ is small, then it is easy to search but phrases from it will not frequently occur, and if $|S|$ is large, it is likely to occur, but hard to search. If the attacker searches by exhausting S first, then the rest of D^n, one can calculate the expected number of guesses. If the phrase is in S, it is $|S|/2$, otherwise if not, then $|S| + (|D|^n - |S|)/2$. The probability of a pass-phrase from S appearing is $1 - (1 - \frac{|S|}{|D|^n})^u$, and so the expected number of guesses required to recover a pass-phrase in general is

$$\begin{aligned} E[guesses] &= \frac{|S|}{2}\left(1 - \left(1 - \frac{|S|}{|D|^n}\right)^u\right) + \\ &\left(|S| + \frac{|D|^n - |S|}{2}\right)\left(1 - \frac{|S|}{|D|^n}\right)^u \end{aligned}$$

To clarify, let $\phi = |S|/|D|^n$, or the fraction of the available pass-phrases which are in S. Substitution yields

$$E[guesses] = \frac{|D|^n}{2}\left(\phi + (1-\phi)^u\right)$$

Taking the first derivative with respect to ϕ yields,

$$\frac{dE[guesses]}{d\phi} = \frac{D^n}{2}\left(1 - u(1-\phi)^{u-1}\right)$$

For the value $u = 4$, $E[guesses]$ has a global minimum at $\phi = 0.3700$. This implies an expected number of guesses equal to $0.2638|D|^n$, or a loss of security of 0.922 bits (compared to the usual $0.5|D|^n$).

For the value $u = 8$, there is a global minimum at $\phi = 0.2570$. This implies an expected number of guesses equal to $0.1749|D|^n$, or a loss of security of 1.52 bits.

Even for the value $u = 16$, there is a global minimum at $\phi = 0.1688$, an expected number of guesses equal to $0.1104|D|^n$, or a loss of security of 2.18 bits.

Therefore under the very generous definition of an "easy-subset" of S, unless the user is given very many choices, these choices do not adversely affect security by a significant margin. This is not meant to be a realistic model of user behavior in the presence of a choice of u options, but rather, a pessimistic model to establish a security lower bound, which is not far from the security in the absence of choice.

6.3 Duress Words

For certain security systems, it may be desirable to offer a duress word, which can be used in the event that the user is forced to log-in under threat of force. Since the user may be under grave stress at that time, the scheme should be as simple as possible. We propose issuing each user an $n + 1$th word, which can be substituted for any of the n words and still result in a successful log-in, while raising a silent alarm of some sort. This requires $n + 1$ entries in the database per user, n of which are marked for alarm, and 1 of which is not.

If the use of a duress word causes the computer to, for example, pretend to crash, then the adversary gains nothing by finding one, and security is unaffected. If instead, the log-in proceeds as normal, but perhaps turns on some cameras, then $\log_2 n$ bits are lost, which would be between 2 and 3 bits for $4 \leq n \leq 8$, a negligible cost.

7 Conclusions

The concept of pass-phrases is not new, and the idea of spelling-error correction of passwords has been proposed before (Dodis et al. 2004). However, heretofore that correction was performed by fuzzy extractors, which are far less efficient than the scheme proposed here. The author also believes that this is the first use of the Damerau-Levenshtein distance method in authentication.

While pass-phrases can be very hard to guess, and easier to remember than complex passwords involving symbols and numerals, errors can be made in their entry. In particular, we have shown that the Damerau-Levenshtein distance metric can be used to create an error-correcting code over the space of words of the English language. We have shown that giving each user a five-word to seven-word subset of a dictionary of particular structure results in a measurably secure password-space, with concrete security guarantees in the event of server compromise, and higher guarantees in the absence of that event. Of course this is only possible because the user is forbidden from choosing their own password—a requirement which some would view as a disadvantage.

To remedy that disadvantage, we show that allowing the user to choose one pass-phrase out of a list of 16 choices results in a negligible security loss. We also show that the system can be made tolerant of the reordering of pass-phrases also with negligible security loss. Moreover, we also introduce methods for allowing a "duress" word.

Appendix B contains an outline of using this system as the control for a password-bank, that would allow distinct, very secure and complex passwords (too difficult for any human to remember) to be used on all systems. The human would not be asked to remember these passwords, which would be recovered from a pass-phrase locked database of passwords. This also provides for backwards compatibility with legacy systems in the obvious way.

It should be noted that we have not performed a usability study, and several parameters of this algorithm could be adjusted based on such a survey. It should be also noted that different sets of users might have different tolerances for word-length, pass-phrase length, and distinct special needs (reordering tolerance or duress words, for example).

8 Acknowledgments

The author is indebted to Nicolas Courtois, who encouraged him to submit this paper to this conference. The author also would like to thank Larry Washington, Juliana Belding, and Patrick Studdard for commenting on an earlier draft of this work, and James Birkett, for listening to an extended outline of the ideas herein while at Eurocrypt'06.

References

Adams, A. & Sasse., M. (1999), 'Users are not the enemy: Why users compromise computer security mechanisms and how to take remedial measures.', *Communications of the Association of Computing Machinery* . Vol. 42.

Bellare, M., Boldyreva, A. & Palacio, A. (2004), An uninstantiable random-oracle-model scheme for a hybrid-encryption problem, *in* 'Advances in Cryptology, Eurocrypt'.

Bellare, M., Pointcheval, D. & Rogaway, P. (2000), Authenticated key exchange secure against dictionary attacks, *in* 'Advances in Cryptology, Eurocrypt'.

Bellovin, M. & Merritt, M. (1992), Encrypted key exchange: Password-based protocols secure against dictionary attacks, *in* 'Proc. IEEE Symp. on Research in Security and Privacy'.

Bellovin, M. & Merritt, M. (1993), Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise, *in* 'Proc. 1st ACM Conf. on Computer and Communications Security'.

Boyko, V., MacKenzie, P. & Patel, S. (2000), Provably secure password authentication and key exchange using diffie-hellman, *in* 'Advances in Cryptology, Eurocrypt'.

Burr, W. (n.d.), Estimating password strength. Draft for Comment, Slide Presentation.

Burr, W., Dodson, D. & Polk, W. (2006), Special publication 800-63: Electronic authentication guide, Technical Report Version 1.0.2, NIST. Recommendations of the National Institute of Standards and Technology.

Canetti, R., Goldreich, O. & Halevi, S. (2004), 'The random oracle methodology, revisited', *Journal of the Association of Computing Machinery* **Vol 51, No. 4**.

Canetti, R., Halevi, S., Katz, J., Lindell, Y. & MacKenzie, P. (2005), Universally composable password-based key exchange, *in* 'Advances in Cryptology, Eurocrypt'.

Canetti, R., Halevi, S. & Steiner, M. (2006), Mitigating dictionary attacks on password-protected local storage, *in* 'Advances in Cryptology, Crypto'.

Damerau, F. (1964), 'A technique for computer detection and correction of spelling errors'.

DiRaimondo, R. & Gennaro, R. (2003), Provably secure threshold password authenticated key exchange, *in* 'Advances in Cryptology, Eurocrypt'.

Dodis, Y., Reyzin, L. & Smith, A. (2004), Fuzzy extractors: How to generate strong keys from biometrics and other noisy data, *in* 'Advances in Cryptology, Eurocrypt'.

Ford, W. & B. Kaliski, J. (2000), Server-assisted generation of a strong secret from a password., *in* 'Proc. 5th IEEE International Workshop on Enterprise Security'.

Garey, M. & Johnson, D. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Fransisco, p. Page 194.

Gentry, C., MacKenzie, P. & Ramzan, Z. (2000), Password authenticated key exchange using hidden smooth subgroups, *in* 'Proc. 12th ACM Conf. on Computer and Communications Security'.

Gentry, C., MacKenzie, P. & Ramzan, Z. (2006), A method for making password-based key exchange resilient, *in* 'Advances in Cryptology, Crypto'.

Goldreich, O. & Lindell, Y. (2001), Session-key generation using human passwords only, *in* 'Advances in Cryptology, Crypto'.

Gong, L. (1995), Optimal authentication protocols resistant to password guessing attacks, *in* 'Proc. 8th IEEE Computer Security Foundations Workshop'.

Gong, L., Lomas, T., Needham, R. & Saltzer, J. (1993), 'Protecting poorly chosen secrets from guessing attacks', *IEEE Journal on Selected Areas in Communications* .

Jablon, D. (1996), 'Strong password-only authenticated key exchange', *ACM Computer Communication Review* .

Jablon, D. (1997), Extended password key exchange protocols immune to dictionary attack, *in* 'Proc. WETICE'97 Workshop on Enterprise Security'.

Jablon, D. (2001), Password authentication using multiple servers, *in* 'Proc. RSA Conference, Cryptographer's Track'.

Jiang, S. & Gong, G. (2004), Password based key exchange with mutual authentication, *in* 'Proc. Workshop in Selected Areas of Cryptography SAC'04'.

Karp, R. (1972), Reducibility among combinatorial problems, *in* 'Proc. Symposium on the Complexity of Computer Computations'.

Katz, J., MacKenzie, P., Taban, G. & Gligor, V. (2005), Two-server password-only authenticated key exchange, *in* 'Proc. Applied Cryptography and Network Security ACNS'05'.

Katz, J., Ostrovsky, R. & Yung, M. (2001), Practical password-authenticated key exchange provably secure under standard assumptions, *in* 'Advances in Cryptology, Eurocrypt'.

Kaufmann, C. & Perlman, R. (2001), Pdm: A new strong password-based protocol, *in* 'Proc. 10th USENIX Security Symposium'.

Kwon, T. (2001), Authentication and key agreement via memorable passwords, *in* 'Proc. Internet Society Network and Distributed System Security Symposium NDSS'01'.

Levenshtein, V. (1966), 'Binary codes capable of correcting deletions, insertions, and reversals', *Sov. Phys. Dokl.* .

Lowrance, R. & Wagner, R. (1975), 'An extension of the string-to-string correction problem', *Journal of the Association of Computing Machinery* .

Lucks, S. (1997), Open key exchange: How to defeat dictionary attacks without encrypting public keys, *in* 'Proc. of the Workshop on Security Protocols'.

MacKenzie, P. (2001), More efficient password-authenticated key exchange, *in* 'Proc. RSA Conference, Cryptographer's Track '01'.

MacKenzie, P. (2002), The pak suite: Protocols for password-authenticated key exchange, Technical report, DIMACS. Technical Report 2002-46.

MacKenzie, P., Patel, S. & Swaminathan, R. (2000), Password authenticated key exchange based on rsa, *in* 'Advances in Cryptology, Asiacrypt'.

MacKenzie, P., Shrimpton, T. & Jakobsson, M. (2006), 'Threshold password-authenticated key exchange', *Journal of Cryptology* **Vol 19, No. 1**.

Maurer, U., Renner, R. & Holenstein, C. (2004), Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology, *in* 'Proc. Theory of Cryptography Conference '04'.

Miller, G. (1956), 'The magical number seven, plus or minus two', *The Psychological Review* **Vol. 63**.

Narayanan, A. & Shmatikov, V. (2005), Fast dictionary attacks on passwords using time-space trade-off, *in* 'Proc. ACM Conf. on Computer and Communications Security CCS'05'.

Nguyen, M. & Vadhan, S. (2004), Simpler session-key generation from short random passwords, *in* 'Proc. Theory of Cryptography Conference '04'.

Patrick, A. (n.d.), Human factors of security systems: A brief review. http://citeseer.ist.psu.edu/patrick02human.html.

Pinkas, B. & Sander, T. (2002), Securing passwords against dictionary attacks, *in* 'Proc. 14th ACM Conf. on Computer and Communications Security'.

Schulz, K. & Mihov, S. (2002), 'Fast string correction with levenshtein-automata', *International Journal of Document Analysis and Recognition* **Vol 5, No. 1**.

Ukkonen, E. (1985), 'Algorithms for approximate string matching', *Information and Control* **Vol 64.**

Wagner, R. & Fisher, M. (1974), 'The string-to-string correction problem', *Journal of the Association for Computing Machinery* .

Wu, T. (1998), The secure remote password protocol, *in* 'Proc. Internet Society Network and Distributed System Security Symposium (NDSS)'.

Wu, T. (1999), A real-world analysis of kerberos password security, *in* 'Proc. Internet Society Network and Distributed System Security Symposium (NDSS)'.

Zhang, M. (2004), New approaches to password authenticated key exchange based on rsa, *in* 'Advances in Cryptology, Asiacryt'.

A More About the Damerau-Levenshtein Distance Metric

A.1 Proof of Metric Properties

Damerau (Damerau 1964) identified the four string-edit operations, but did not construct a metric. Levenshtein (Levenshtein 1966) constructed and proved a metric for the three operations excluding transposition. While it is well-known that one still has a metric when the fourth operation is used, neither paper discusses this and the proof is very short, so we include it here.

First, observe that for any sequence of string-edit operations, the number of operations is a non-negative integer. Given a set of sequences of operations that each transform the word w_1 into w_2, one can assign an integer to each, equal to the number of operations in it. Since this is a set of non-negative integers, it always has a unique least member in the set. Therefore the metric is well-defined. The non-negativity property also follows from this.

It is also clear that zero operations will transform w_1 to w_2 if and only if w_1 and w_2 are identical strings, and so non-degeneracy is true. Each of these operations can be reversed with one operation, and so its clear that $d(w_1, w_2) = d(w_2, w_1)$, and $d(,)$ is thus symmetric. The triangle inequality also follows, for the following reason.

Suppose $d(a, b) + d(b, c) < d(a, c)$, with $d(a, b) = d_1$, $d(b, c) = d_2$ and $d(a, c) = d_3$. Substitution yields $d_1 + d_2 < d_3$. Suppose further that the optimal sequence of operations s_1 transforms a into b, while s_2 transforms a into c, also optimally. (This means $|s_1| = d_1$ and $|s_2| = d_2$). Observe that the sequence of operations of s_1 followed by s_2 is a sequence of operations which transforms a into c, and has length $d_1 + d_2$. Since the length of the shortest series of operations to transform a into c is d_3, it must be the case that $d_1 + d_2 \geq d_3$. But this is a contradiction. Therefore no such a, b, c can exist, and the triangle inequality holds.

A.2 Algorithms for Computation

The code in Figure 1 calculates the metric, and is in the C language. It is based on an implementation from (Lowrance & Wagner 1975). The constant ALPHA is the size of the alphabet. The characters themselves are represented by the integers $1, 2, \ldots, 26$ for the English language. The integer arrays `s1`, `s2` are the strings being compared and are of maximum length `MAXLEN`. The data structure "thematrix" is a `MAXLEN` × `MAXLEN` matrix. Finally, the function `minimumFour` returns the least of its four integer inputs. The four COST constants are the costs of each Damerau-Levenshtein metric operation. The authors used 1 for each cost, as is convention.

A.3 Decoding Time

One observation is that if the first, third, and fifth letters of two words are each unequal (which only costs three comparisons to check), then no two Damerau-Levenshtein operations will change one word into the other. Therefore, the distance need not be calculated during decoding of an entered user pass-phrase word, as distances of 3 or larger "are effectively infinite."

To take advantage of this, one can create $3 \times |A|$ buckets, where A is the alphabet of the language. For each letter of the alphabet, there is a bucket with all words that contain that letter as its first letter, and also one for the third and fifth. To check if a word has a dictionary entry that is distance two or less away requires only checking 3 buckets, rather than the entire dictionary. Naturally, one can also stop when one finds the correct word (as it is unique), and unless the first letter was damaged by error, this will be rather fast. Finally, note that because

$$d(a, b) \geq | \quad |a| - |b| \quad |$$

it follows that if $||a| - |b|| > 2$ then no two Damerau-Levenshtein operations will equate a and b. Checking the length of two strings is very quick, and this guarantees that $d(a, b)$ need only be calculated for a few words when doing pass-phrase decoding. Using these methods, decoding was essentially instantaneous on an author's laptop.

B A Password Bank

Several commercial products consist of a database of user-passwords, either stored remotely or locally (e.g. Firefox), to reduce the mental burden of remembering passwords. These are often protected by one master password, or a pass-phrase. With this in mind, note that the current suite of password-based protocols, including the PAK-Z (MacKenzie 2001) and Ω-method

```
int d(int s1[], int s2[], int length1,
                           int length2) {
  int i,j,d, i1, j1, DB;
  int DA[ALPHA+3];

  for (i=0; i<=length1; i++) {
    thematrix[i+1][1]=i*COST_DEL;
    thematrix[i+1][0]=INF;
  }

  for (j=1; j<=length2; j++) {
    thematrix[1][j+1]=j*COST_INS;
    thematrix[0][j+1]=INF;
  }

  for (d=1; d<=ALPHA; d++) {
    DA[d]=0;
  }
 for (i=1; i<=length1; i++) {
    DB=0;
    for (j=1; j<=length2; j++) {
      i1=DA[s2[j-1]];
      j1=DB;

      d = (s1[i-1]==s2[j-1]) ? 0 : COST_CHG;

      if (s1[i-1]==s2[j-1]) DB=j;

      thematrix[i+1][j+1] = minimumFour(
               thematrix[i][j]+d,
               thematrix[i+1][j]+COST_INS,
               thematrix[i][j+1]+COST_DEL,

               thematrix[i1][j1]+
               (i-i1-1)*COST_DEL+
               COST_SUB +
               (j-j1-1)*COST_INS);
    }

    DA[s1[i-1]]=i;
  }

  return thematrix[length1+1][length2+1];
}
```

Figure 1: C Code to calculate the Damerau-Levenshtein Distance Metric

(Gentry et al. 2006), prevent any feasible attack if the server file is not compromised. In the event it is compromised, the fastest attack is to exhaust the dictionary, which would take around 2^{80} to 2^{112} guesses if $c = 16$ and $n = 5$ or $n = 7$, and reordering is not used. Since this is very difficult, it is reasonable to implement the following password-consolidation policy.

Users are often told not to use the same password on multiple systems. This prevents the compromise of one password on one system from causing problems elsewhere. Unfortunately, users find this irritating. But if all the user's passwords were stored on in a password bank, which could only be accessed via an SSL/TLS link, and if this link required an authentication via the scheme in this paper, then the user is free to have a distinct password on every system, without memorizing any of them (except the pass-phrase). Furthermore, this would allow the user to have very secure (uniformly generated printable ASCII) passwords on systems that do not permit pass-phrases. Only the bank master-password for each user would have to be modified to use this scheme.

It should be noted that such a password bank should require a re-entry of the pass-phrase for each query, so that the SSL/TLS link cannot be used by an adversary if the user walks away from the terminal while still logged in to the password bank.