

MNEMONIC EVOLUTION OF PASSWORD DESIGN

Ihor Jakowec June 2017

While longer passwords are generally deemed more difficult for a hacker to crack; they are harder to remember. Comments have been posted, to not use passwords with repeating patterns; as these have been added to hacker search lists. This creates a new problem for those who need to remember longer passwords. Presented here is a technique for remembering long passwords by taking advantage of visual memory hints.

At around y2k, I bought a used IBM keyboard. Since that time. I have been privately using an evolving strategy to generate passwords.

Disclaimer: This should not be confused with the PsychoPass Method (2012), or published corrections to it (2012).

Below are some results of successive generations of password design with corresponding examples, showing how to generate them.

WHAT TO KNOW, BUT NOT TO USE

(first generation)

For those of you that are old enough, recall the rectangular grid that could have represented a character or glyph in the earliest dot matrix printers. Lets keep the resolution very low, say 3 x 3. This could be represented on a calculator keypad, or the numeric keypad on some keyboards, as:



On this array, a pattern could be traced out in the shape of a character, while pressing successive keys. Call this a dance move, and the character associated with it a dance hint character. A dance hint character need not be restricted to a keyboard character. You can use unicode characters and symbols, to remind you of a dance move. Or better yet, instead of characters from a language, use your own design.

Examples:

<p>M</p> <p>ASCII 77</p>	<p>𐌆 or 𐌇</p> <p>(this refers to the BOPOMOFO letter M)</p> <p>Unicode: U+3107</p> <p>UTF-8: E3 84 87</p>	<p>Sail Outline¹</p> <p>similar symbols in unicode:</p> <p>🚢 U+26F5 𐤀 U+10903</p> <p>(Phoenician letter delt)</p>
		

their corresponding key sequences ...

1 4 7 5 9 6 3

1 4 7 8 9 6 3

. 3 2 1 5 9 6

¹ For some, the digit 4 is reminiscent of a sailboat sail. Also, the first letter of the word sail, is S. So, the following matrix representation, could be associated with the dance hint character 4, or better yet, S, for sail. [\(more obscure\)](#)

The keys used for the dance move associated with the letter M; can also be used for H, or N, or W. Ambiguity in the dance hint character, can confuse or help hackers, (depending on the search technique used).

(The period is below the 3 on many keyboards with a numeric keyboard. Be mindful of this, if you change keyboards. Having a unique keyboard could be a slight security advantage, or a mobility problem).

A LITTLE MORE CREATIVITY IS BE NEEDED

(second generation, still not as secure as later generations)

Combine several dance moves to form a password. Call the characters associated with each dance move a dance hint.

Note: **DO NOT** use what I call a dance hint here in the hint part of your login manager program. (Since others may see it, and guess your dance moves, if they are too obvious.) Instead, consider something like the following less apparent approach:

To add an extra degree of hacker confounding. You can have some parts of the hint, map directly to a single keyboard character.

$P \mapsto v$ $A \mapsto O$

This may shorten the password length. However, more importantly, the password characters are not confined to a region surrounding a given key on the keyboard.

Another adjustment:

At a random location, insert an extra character(s) character, say Q.

Combining the above the hint characters with two hint characters from the previous example, generates this password:



The diagram shows each hint character and its corresponding part of the password. If you forget the password, fall back on the dance move hint: S P A M To remember dance moves. You can keep a home made booklet of flash cards showing a rectangular section of a keyboard, (similar to the posters dance schools use, to show where you put your feet).

Caution:

This technique worked for me until a similar idea was made public.

(refer below to PsychoPass)

However, further developments in hacking made even this password design **less secure**.

(refer to: <http://www.dailymail.co.uk/sciencetech/article-2331984/Think-strong-password-Hackers-crack-16-character-passwords-hour.html>)

REFER TO A KEYBOARD TEMPLATE

(third generation)

Even better still: Consider a keyboard layout that should differ from the standard U.S. keyboard, such as: Colemak, or Dvorak, or Workman. (But really, you should use a keyboard layout of randomly arranged keys.) Here's how: Begin with a standard keyboard layout. Then sequentially assign a unique numerical value to each password useable key. I refer to this as a ***sequentially tagged keyboard***:



These numerical tags are referred to as the ***co-ordinates*** for a password useable key. For the example, the ***co-ordinate*** for the key q is ⑪.

Use a random number generator enough times to reassign a new unique co-ordinate to each key, by sampling without replacement, in the range 1 to 47. (Only, re-label the 47 password useable keys.) To illustrate: These co-ordinates are arranged as a key-board ...



In the above diagram, go from left to right, and top to bottom: The first random value generated is ⑪. It's corresponding position on the **sequentially tagged keyboard** is ④⑦. (⑪ \mapsto ④⑦) This in effect, moves the face value of q (the ⑪ 'th co-ordinate), of the **sequentially tagged keyboard**, to the ④⑦ 'th co-ordinate. So, on the randomized keyboard, q re-places `. Repeat this for the remaining 46 password usable keys.

Note: The only key that remains in the same position is ⑩⑥ . By not excluding these identity mappings, $2^{47} - 1$ of them; the total number of possible arrangements grows to 47! (factorial). Still, ideally, your password itself should contain few, if any, identity mappings.

(Refer to the **sequentially tagged keyboard** and the **above** diagram, the result is in the **below** randomized keyboard.)

By restoring the face value to the shuffled keys, the randomized keyboard layout becomes:



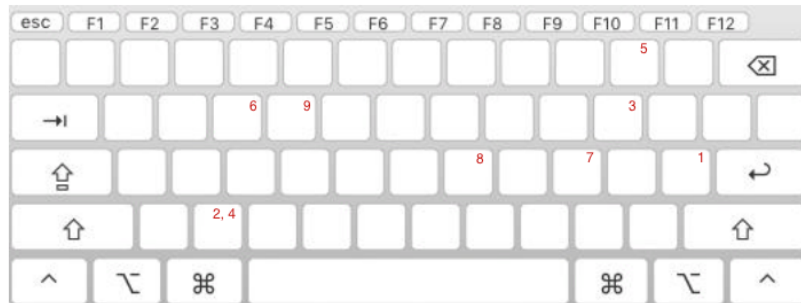
To generate a password, lets use a memorable phrase: POGOSTICK
Red digits above the character show what keys are used, and the order they are pressed.

(You type POGOSTICK while looking at the above layout. However, what you are actually typing is shown below.) The password becomes:

' x p x - e l j r



Note, the used keys below are disjoint and do not form a single rhomboid. The rhomboid key cluster is replaced with the randomized anagram of the sequentially tagged keyboard.



(Incidentally, POGOSTICK is not used in most dictionaries.)

You may say why not just write down a randomly generated password. You can if you don't have to worry where you leave it. Note that, dance moves and randomized keyboards do not give away your passwords as easily. (Also, refer to the below section on using dice.)

(refer below to: Microsoft security guru: Jot down your passwords)

Caution:

Be careful as to how you use your keyboard layout template at sites that are less secure. Some access manager programs echo the answer to security questions. **DO NOT** look at your randomly generated keyboard to type in answers. Answers like your fathers age follow a bell curve with not much variance. That is, the most common age is close to thirty with about at most a few years that spread above and below the peak. You could expose as many as ten characters (as in the answer : Twenty-Eight).

A more useable example:

In reality type a longer phrase, (as of this writing ≥ 16 characters), while looking at your randomized keyboard, like:

JABBERWOCKY POGOSTICK

Dictionary phrases have associated letter frequencies. This can be obfuscated by shuffling two weaker passwords together. A hacker could use frequency and pattern analysis to narrow the down a search. In POGOSTICK, the second and fourth letter repeat. The hacker could search for what was at one time 62615 nine letter words. A further observation: only the second and fourth letter repeat. This prunes the search to 668 words.

Simple

One shuffling of the words JABBERWOCKY and POGOSTICK could be:

P J O A B G B E R O S W O C T K Y I C K



1 2 3 0 3 2

I refer to the red digits above as insertion descriptors. Each digit describes how many consecutive letters of the first word are inserted after the next successive letter in the second word. It is easier to remember digits in groups of three, so you should be able to remember two groups of three digits, and two words. You should randomly generate the insertion descriptor, (by using dice or a computer). The random digits should range from zero to the remaining length of the word being inserted. The 20 character long password becomes:

' a x 6 p w w x [b 8 - x e j r l y j r

Note: When to use a shift or other meta keys can also be applied by using a random number generator.

Aside:

The major event that caused passwords to be more easily hacked, had more to do with human psychology, than an increased power in computing. When online entertainment and gaming sites started to attract a membership of over 50k. The hash tables on their servers had something of value that only some hackers at first realized. A database of passwords most people use. Different people from different walks of life ended up using the same password. To realize how prevalent this is, ask a classroom of people to remember the name of a flower. It is highly likely that more than one person will have chosen the same flower. People even used the same methods to obfuscate their password. That is: appending digits to a dictionary word; or swapping some alternate letters with a digit; or using simple patterns (like a staircase or a rhomboid with the middle column skipped). Knowing this, some hackers even used Markov chains, (from probability theory), to assign probabilities as to whether to choose a neighbouring character, or alternate with a digit.

So, be cautious if you think you have a unique password someone else using a gaming site may have had the same idea.

(refer to: <http://www.dailymail.co.uk/sciencetech/article-2331984/Think-strong-password-Hackers-crack-16-character-passwords-hour.html>)

VULNERABILITY OR STRENGTH

The search size expressed as information entropy:

(refer below to Wikipedia on password strength)

dictionary words	417099
keyboard keys shuffled	47
ways to collate a 9 letter word with a 11 letter word	$\binom{20}{11}$

$$\log_2\left(\binom{20}{11} \cdot 417099^2 \cdot 47!\right) \cong 252$$

Looking at information entropy expressed in terms of the number of keys (symbols) used:

$$\log_2(96^{20}) \cong 131.7$$

An excerpt from [wikipedia.org](https://en.wikipedia.org/wiki/Password_strength)

from the page: en.wikipedia.org/wiki/Password_strength

“Thus a random password's [information entropy](#), H , is given by the formula

$$H = \log_2(N^L) = L \cdot \log_2(N)$$

Where N is the number of possible symbols and L is the number of symbols in the password. H is measured in [bits](#).^{[2][7]} In the last expression, \log can be to any [base](#).”

Aside: The first method requires patience (repeated throw of a dice), to obtain the latter key mappings.

(If you have a freshly new default install, generated at the same time as others, and you don't want to use: `/dev/urandom`)

I modified a manual approach to randomly generate 48 possible outcomes. This is the number of keys on the main keypad, (including the spacebar), that could be used in a password.

(refer to: https://en.wikipedia.org/wiki/Random_password_generator) (specifically, Mechanical Methods)

Begin by using three dice. I use the first to choose a column, from the table below. The second is used to choose a sub-table. (Sub-tables are indicated by different background colours.) While the third is used to choose the row within the selected sub-table.

Specifically for the second or third dice, an outcome of:

1 or 2 chooses the first sub-table, or first row

3 or 4 chooses the second sub-table, or second row

5 or 6 chooses the third sub-table, or third row

An example of mappings, (refer to table below):

















































1st 2nd 3rd
(1, 1 or 2, 1 or 2) \mapsto q
(2, 3, 1 or 3, or 4) \mapsto 0

1 st							2nd
3rd	1	2	3	4	5	6	
1 or 2	a	b	c	d	e	f	1
3 or 4	g	h	i	j	k	l	or
5 or 6	m	n	o	p	q	r	2
1 or 2	s	t	u	v	w	x	1
3 or 4	y	z	-	=	[]	or
5 or 6	\	;	'	,	.	/	2
even	1	2	3	4	5	6	3
odd	7	8	9	0	`	space	6

Aside: The second method requires deck of cards, and the following table.

To obtain an acceptable degree of randomness the deck must be cut and shuffled seven times.

(refer to: <http://www.ams.org/samplings/feature-column/fcarc-shuffle>)

	a		b		c		d		e		f
	g		h		i		j		k		l
	m		n		o		p		q		r
	s		t		u		v		w		x
	y		z		-		=		[]
	\		;		“		<		>		/
	1		2		3		4		5		6
	7		8		9		0		~		space

The above mapping is for the standard US keyboard. For other keyboards you may need to add



and the face cards.

Begin by using a deck of cards shuffled at least seven times. Shuffling more does not significantly improve the randomness factor. Select one of the cards from the deck of 48 cards. The cooreponding keyboard value for this card (from the above table), is assigned to position 1 of the sequentially tagged keyboard (mentioned above). The next selected card is assigned to position 2 of the sequentially tagged keyboard. This is repeated for the remaining keys. You may want to re shuffle your deck of cards later if you leave them in a publically accessible place.

REFERENCES

from [betterbuys.com](https://www.betterbuys.com)

From their page:

[https://www.betterbuys.com/estimating-password-cracking-times/\(2015\)](https://www.betterbuys.com/estimating-password-cracking-times/(2015))

Amount of Time to Crack Passwords	
"abcdefg" 7 characters	🕒 .29 milliseconds
"abcdefgh" 8 characters	🕒 5 hours
"abcdefghi" 9 characters	📅 5 days
"abcdefghij" 10 characters	📅 4 months
"abcdefghijk" 11 characters	📅 1 decade
"abcdefghijkl" 12 characters	📅 2 centuries

“Over the years, passwords weaken dramatically as technologies evolve and hackers become increasingly proficient. For example, a password that would take over three years to crack in 2000 takes just over a year to crack by 2004. Five years later, in 2009, the cracking time drops to four months. By 2016, the same password could be decoded in just over two months. This demonstrates the importance of changing passwords frequently.”

An excerpt from msecure.com (2010)

Password Length: Time to Crack*

6 characters: 11 hours

7 characters: 6 weeks

8 characters: 5 months

9 characters: 10 years

*assumes each character can be any ASCII character.

How to Create Memorizable and Strong Passwords

Pietro Cipresso^{1,2}, MSc, PhD; Andrea Gaggioli^{1,2}, MPsy, PhD; Silvia Serino^{1,2}, MPsy; Sergio Cipresso³; Giuseppe Riva^{1,2}, MA, MPsy, PhD

¹Applied Technology for Neuro-Psychology Lab, IRCCS Istituto Auxologico Italiano, Milano, Italy

²Psychology Department, Catholic University of Milan, Milano, Italy

³Freelancer, Milan, Italy

Security Analysis and Improvements to the PsychoPass Method

Monitoring Editor: Gunther Eysenbach

Reviewed by Pietro Cipresso and Robin Kok

[Bostjan Brumen](#), BSc, PhD,#1 [Marjan Heričko](#), MSc, PhD,#1 [Ivan Rozman](#), MSc, PhD,#1 and [Marko Hölbl](#), BSc, PhD#1

Excerpts from cnet.com

Microsoft security guru: Jot down your passwords (May, 23, 2015)

Companies should not ban employees from writing down their passwords because such bans force people to use the same weak term on many systems, according to a Microsoft security guru.

"Since not all systems allow good passwords, I am going to pick a really crappy one, use it everywhere and never change it," Johansson said. "If I write them down and then protect the piece of paper--or whatever it is I wrote them down on--there is nothing wrong with that. That allows us to remember more passwords and better passwords."

Johansson said the security industry had been giving out the wrong advice about passwords for 20 years.

Also:

- Komanduri, S., Shay, R., Kelley, P.G., Mazurek, M.L., Bauer, L., Christin, N., ... Egelman, S. (2011). [Of passwords and people: Measuring the effect of password-composition policies](#). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 2595-2604. (Warning: link leads to a **PDF file**.)
- Shay, R., Komanduri, S., Durity, A.L., (Seyoung) Huh, P., Mazurek, M.L., Segreti, S.M., ... Cranor, L.F. (2014). [Can long passwords be secure and usable?](#). In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 2927-2936. (Warning: link leads to a **PDF file**.)
- <https://labs.mwrinfosecurity.com/blog/a-practical-guide-to-cracking-password-hashes> (Sep 25, 2015 - PACK is a set of tools developed by **Peter Kacherginsky** to perform analysis on sets of cracked passwords)

DISCLAIMER

Any words not defined, in this collection of documents; use the definition given to them, by the OED and/or American Webster's dictionary. It is not my intention to use double entendre: either from English or in any other language, or from pop subculture (past or present).

Any words redefined by me, or phrases defined by me, are simply technical in nature. They are not intended to refer to any:

group, institution, organization, person or persons.

- No alternate meaning to any word or phrase is implied or intended.
- My intent is not to: slight, insult, affront, or offend.

Ihor Jakowec
Monday 5 December 2016