Novel Password Systems Where Enter Derivation of Password instead of Actual Password

6.858 Proposal

Michael Plasmeier <theplaz>
Jonathan Wang <jwang7>
Miguel Flores <mflores>

We propose exploring systems where the user enters a representation of their password, instead of their actual password.

The Problem

The problem with many password systems is that users must type their entire, full password each time they log on. This makes the password vulnerable to key logging and interception during transmission.

How we plan to address it

We seek to explore systems in which the user does not enter their direct password, but a derivation of the password **which changes on each log in**. The user proves that he or she knows the password without subsequently ever providing the password itself.

ING Password Keyboard

A simple example is ING Direct's PIN pad. Under ING's system, the user enters the letters corresponding to their PIN instead of the PIN itself. The mapping between numbers and letters is randomly generated on every log in. This method does not survive an attack where the attacker has access to the mapping, but it does prevent simple keylogging.

Figure 1 ING's Pin Pad. The user enters the letters corresponding to their PIN in the box.

Windows 8 Picture Passwords

Our proposal does not cover systems such as Windows 8's new "Picture Passwords" feature. With Picture Passwords, the user's password is a series of user selected vectors which a user draws on top of a user specified picture. However, the vector must be repeated exactly the same each time. We propose to concentrate on systems where the user enters a different derivation of their password each time.

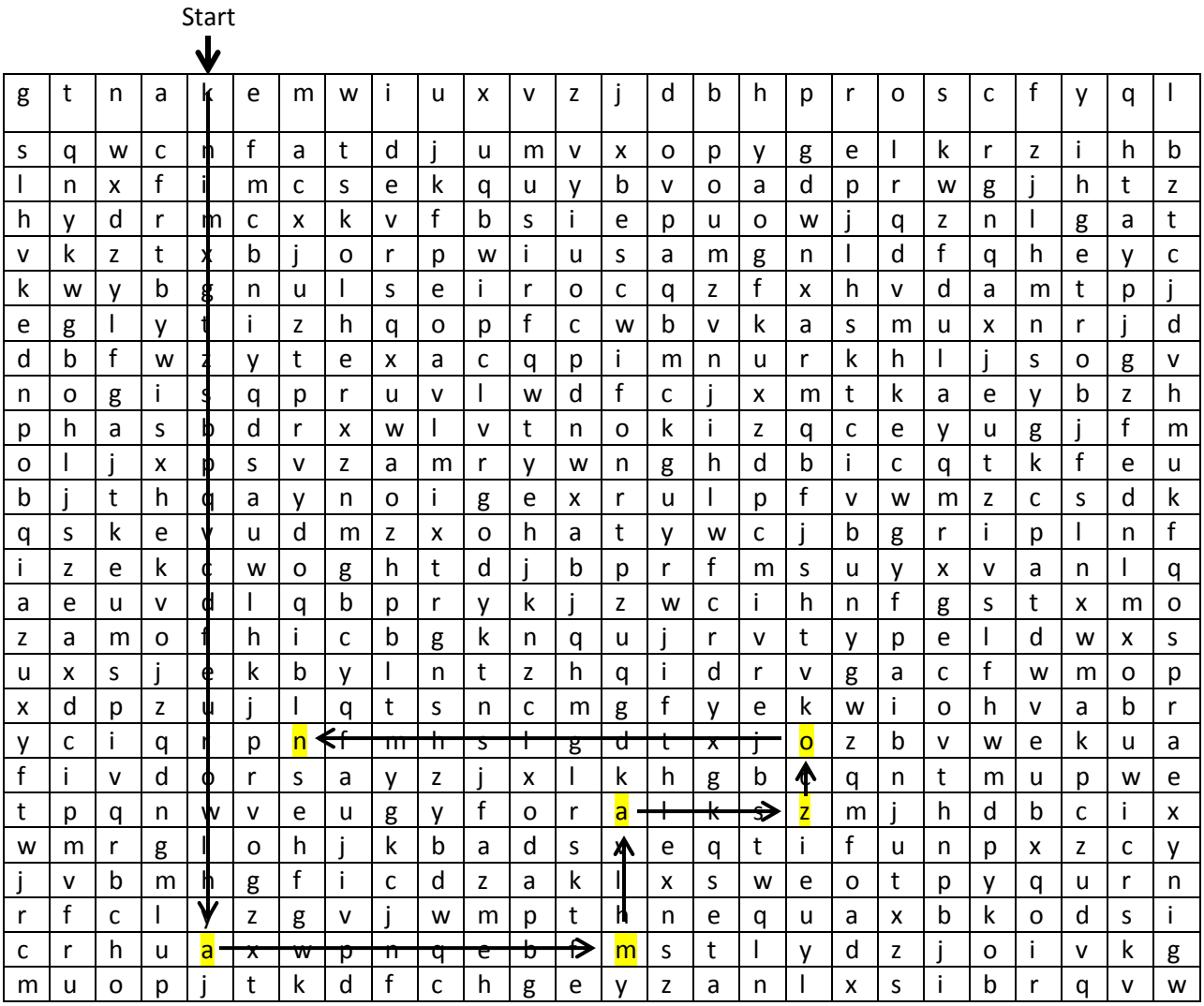
Trace the Grid

Under this system the user chooses a password from a lower case Latin alphabet. However, instead of entering the password each time the user logs in, the user is presented with a randomized grid of 26x26 letters with each letter only once per row and column (this is called a Latin Square. Sudoku is the most famous example of a Latin Square). The user then enters a derivation, called a *trace* of their password which is transmitted back to the server.

The server first randomly generates a 26x26 Latin square and a start row or column. These are transmitted to the user. The user then visually traces out his or her password on the grid, alternating between rows and columns. For example, the user would locate the first letter of their password on the start row or column. The user would then look for the next letter of his or her password in either the column (if the start was a row) or row (if the start was a column) that contained the user's first character. The user would then continue alternating for the length of their password.

The user enters the directions (up, down, left, right) that they follow as they trace out their password. The user should include the direction from the start marker. This combination of directions that the user inputs is referred to as a *trace*. The trace is then sent from the client to the server, and the server can easily verify that a trace corresponds to the correct password.

Example: entering the password Amazon with the 5th column as the start row/column. The grid as well as the start row/column are randomly generated be the server for each log in.



g	t	n	a	k	e	m	w	i	u	x	v	z	j	d	b	h	p	r	o	s	c	f	y	q	l
s	q	w	c	n	f	a	t	d	j	u	m	v	x	o	p	y	g	e	l	k	r	z	i	h	b
l	n	x	f	i	m	c	s	e	k	q	u	y	b	v	o	a	d	p	r	w	g	j	h	t	z
h	y	d	r	m	c	x	k	v	f	b	s	i	e	p	u	o	w	j	q	z	n	l	g	a	t
v	k	z	t	x	b	j	o	r	p	w	i	u	s	a	m	g	n	l	d	f	q	h	e	y	c
k	w	y	b	g	n	u	l	s	e	i	r	o	c	q	z	f	x	h	v	d	a	m	t	p	j
e	g	l	y	t	i	z	h	q	o	p	f	c	w	b	v	k	a	s	m	u	x	n	r	j	d
d	b	f	w	z	y	t	e	x	a	c	q	p	i	m	n	u	r	k	h	l	j	s	o	g	v
n	o	g	i	s	q	p	r	u	v	l	w	d	f	c	j	x	m	t	k	a	e	y	b	z	h
p	h	a	s	b	d	r	x	w	l	v	t	n	o	k	i	z	q	c	e	y	u	g	j	f	m
o	l	j	x	p	s	v	z	a	m	r	y	w	n	g	h	d	b	i	c	q	t	k	f	e	u
b	j	t	h	q	a	y	n	o	i	g	e	x	r	u	l	p	f	v	w	m	z	c	s	d	k
q	s	k	e	y	u	d	m	z	x	o	h	a	t	y	w	c	j	b	g	r	i	p	l	n	f
i	z	e	k	c	w	o	g	h	t	d	j	b	p	r	f	m	s	u	y	x	v	a	n	l	q
a	e	u	v	d	l	q	b	p	r	y	k	j	z	w	c	i	h	n	f	g	s	t	x	m	o
z	a	m	o	f	h	i	c	b	g	k	n	q	u	j	r	v	t	y	p	e	l	d	w	x	s
u	x	s	j	e	k	b	y	l	n	t	z	h	q	i	d	r	v	g	a	c	f	w	m	o	p
x	d	p	z	u	j	l	q	t	s	n	c	m	g	f	y	e	k	w	i	o	h	v	a	b	r
y	c	i	q	r	p	n	f	m	h	s	t	g	d	t	x	j	o	z	b	v	w	e	k	u	a
f	i	v	d	o	r	s	a	y	z	j	x	l	k	h	g	b	q	n	t	m	u	p	w	e	
t	p	q	n	w	v	e	u	g	y	f	o	r	a	l	k	s	z	m	j	h	d	b	c	i	x
w	m	r	g	l	o	h	j	k	b	a	d	s	x	e	q	t	i	f	u	n	p	x	z	c	y
j	v	b	m	h	g	f	i	c	d	z	a	k	l	x	s	w	e	o	t	p	y	q	u	r	n
r	f	c	l	y	z	g	v	j	w	m	p	t	h	n	e	q	u	a	x	b	k	o	d	s	i
c	r	h	u	a	x	w	p	n	q	e	b	f	m	s	t	l	y	d	z	j	o	i	v	k	g
m	u	o	p	j	t	k	d	f	c	h	g	e	y	z	a	n	l	x	s	i	b	r	q	v	w

The resulting trace would be: Down, Left, Up, Right, Up, Left.

Figure 2 A trace of the password "amazon"

An attacker with knowledge of the grid, including the start location and the *trace,* cannot easily determine your password.

Some limitations of the system include:

- an attacker can eventually figure out the password by observing enough grids and traces
- passwords are stored in plain text on the server
- a user moving the mouse pointer or finger to manually trace out the password on the grid could be observed by an "over the shoulder" adversary

However, this scheme allows for the trace to be observed.

What we plan to build

We plan to build implement of at least one of the systems we describe. We will build a library for Python Flask which would allow developers to easily adopt this authentication system. The library would insert the grid of letters into the login page that it serves, and it would process the user input to verify that it correctly traces out the password in the grid. The system will also impose restrictions on the passwords that a user can have in order to lower the chance of randomly guessing a correct trace.

Building the system would help us better understand the amount of effort needed to implement the system. We will also run usability testing to evaluate the system which we implement. This would help us comment on how easy it is for users to adopt the new system.