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

6.858 Final Project

Michael Plasmeier <theplaz>

Jonathan Wang <jwang7>

Miguel Flores <mflores>

December 14, 2012

# Motivation

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, shoulder surfing, and interception during transmission.

We 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.**

### Answering Questions

One could answer questions about the password, instead of inputting the password itself. For example, say a user’s password is “tennis ball.” The system could prompt “what color is it?” The user would respond “green.” The next time the system could ask “what shape is it?” The user would respond “round.” This way the user only transmits their actual object during registration, but never during log in.

## Hashing a Response

In an ideal world, the user could prove to the server that it knew the secret by producing a cryptographic hash of the user’s secret combined with a server-selected nonce.

However, people are not particularly good at being able to calculate cryptographic hashes in their heads, so we need to seek an alternate system.

# Inspiration

## Original Off the Grid

We were inspired by the “Off the Grid” system from the Gibson Research Corporation.[[1]](#footnote-2) The “Off the Grid” proposal is designed to allow users to use a personal printed paper grid to encipher the domain name of the website they are currently on into a string of pseudo-random characters.

The Off the Grid system works entirely on the user’s side. Websites do not need to do anything to support Off the Grid.

To use Off the Grid, the user first generates a grid from a grid-providing website such as <https://www.grc.com/offthegrid.htm>. This website generates a grid using client-side scripting (i.e. JavaScript) to generate the grid on the user’s machine. The user then prints the grid onto a sheet of letter paper. At this point the Grid is offline and thus impossible to access by malware. As an alternative, there is at least one application for Android which produces and stores a grid; however, the grid is now accessible to malware on the Android phone which is able to defeat the inter-process sandboxing.

The grid that is generated is a Latin Square. A Latin Square is an *n* × *n* array filled with *n* different symbols, each occurring exactly once in each row and exactly once in each column.[[2]](#footnote-3) The most famous Latin Square is the popular puzzle game Sudoku. (Note however, that we do not divide up the grid into 9 smaller 3x3 mini-squares in which each symbol must be unique). For example, here is a 11x11 Latin Square with 11 alphabetic characters: Normally, a 26x26 Latin Square is used.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| g | e | a | m | o | n | z | k | i | r | c |
| k | a | n | c | m | z | o | r | g | i | e |
| n | k | c | z | a | m | r | g | o | e | i |
| z | o | i | a | n | g | e | c | r | k | m |
| m | r | z | n | g | a | k | i | e | c | o |
| a | g | e | i | z | r | n | o | c | m | k |
| r | c | g | k | e | i | m | n | z | o | a |
| e | n | k | g | i | o | c | z | m | a | r |
| i | m | o | e | r | c | g | a | k | n | k |
| c | i | r | o | k | e | a | m | n | z | g |
| o | z | m | r | c | k | i | e | a | g | n |

**Figure 2 An 11x11 Latin Square; normally 26x26, but reduced in size here to save space.**

Once the user has a grid, they use the grid to create or change the password for each website. The Off the Grid specification has a number of variants, but we will use the base variant described on the GRC website. The author recommends that each user adopt slight variations to the rules in order to increase security. To provide a consistent analysis, we assume the user ignores this suggestion.

In the Off the Grid specification, the user traces the name of the website twice to provide additional entropy. In the start of the first phase, the user always starts along the first row of the grid.

|  |  |  |  |
| --- | --- | --- | --- |
| b  Start | d | a | c |
| a | c | b | d |
| d | b | c | a |
| c | a | d | b |

**Figure 3**

The user then traces out the first 6 characters of the domain name. 6 characters was chosen by the author to provide a 12 character password, which the author chose to balance ease of use with entropy. Again, a user may choose their own scheme. The user alternates between looking horizontally and vertically.

Start

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| g | e | a | m | o | n | z | k | i | r | c |
| k | a | n | c | m | z | o | r | g | i | e |
| n | k | c | z | a | m | r | g | o | e | i |
| z | o | i | a | n | g | e | c | r | k | m |
| m | r | z | n | g | a | k | i | e | c | o |
| a | g | e | i | z | r | n | o | c | m | k |
| r | c | g | k | e | i | m | n | z | o | a |
| e | n | k | g | i | o | c | z | m | a | r |
| i | m | o | e | r | c | g | a | k | n | k |
| c | i | r | o | k | e | a | m | n | z | g |
| o | z | m | r | c | k | i | e | a | g | n |

**Figure 4**

In the second phase, the user starts at the character that they ended with at the end of Phase 1. The user then selects two more characters from the grid in the same direction of travel. The user then appends those two characters to their password.

|  |  |  |  |
| --- | --- | --- | --- |
| b | d | a | c |
| a | c | b | d |
| d | b | c | a |
| c | a | d | b |

**Figure 5** **The user arrives at c traveling to the right. The user appends the next two characters “bd” to their password, and then continues up/down from the last character they read “d”.**

The user wraps around if their characters go off the grid.

|  |  |  |  |
| --- | --- | --- | --- |
| b | d | a | c |
| a | c | b | d |
| d | b | c | a |
| c | a | d | b |

**Figure 6 The user arrives at b traveling to the right. The user appends the next two characters to their password, wrapping around if they go off the edge of the grid. Here those characters are “da”. The user then continues up/down from the last character “a”.**

For example here is Phase 2 of our Amazon example.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| g | e | a | m | o | n | z | k | i | r | c |
| k | a | n | c | m | z | o | r | g | i | e |
| n | k | C | z | a | m | r | g | o | e | i |
| z | o | I | a | n | g | e | c | r | k | m |
| m | r | Z | n | g | a | k | i | e | c | o |
| a | g | e | i | z | r | n | o | c | m | k |
| r | c | g | k | e | i | m | n | z | o | a |
| e | n | k | g | i | o | c | z | m | a | r |
| i | m | o | e | r | c | g | a | k | n | k |
| c | i | R | o | k | e | a | m | n | z | g |
| o | z | m | r | c | k | i | e | a | g | n |

**Figure 7** **Phase 2 of Off the Grid. The password is “gaznegmacmzg”**

Here are Phase 1 and Phase 2.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| g | e | a | m | o | n | z | k | i | r | c |
| k | a | n | c | m | z | o | r | g | i | e |
| n | k | C | z | a | m | r | g | o | e | i |
| z | o | I | a | n | g | e | c | r | k | m |
| m | r | Z | n | g | a | k | i | e | c | o |
| a | g | e | i | z | r | n | o | c | m | k |
| r | c | g | k | e | i | m | n | z | o | a |
| e | n | k | g | i | o | c | z | m | a | r |
| i | m | o | e | r | c | g | a | k | n | k |
| c | i | R | o | k | e | a | m | n | z | g |
| o | z | m | r | c | k | i | e | a | g | n |

**Figure 8** **Phase 1 and 2 of Off the Grid.**

To log in, the user retraces exactly the same steps as when creating a password. This means the password is exactly the same for each domain. This is an obvious requirement for a system designed to fit within the existing password infrastructure. However, we wanted to explore ideas in which the user does not enter the same password each time.

# Description of System

## CrossPassword

We wanted to design a system similar to the Off the Grid system, but where the password the user transmits over the network is different each time. With this system, the website presents the user with a grid and the user enters only a deviation of their password.

When the user creates an account, he provides his or her password to the webserver. The user may use characters from the lower case Latin alphabet [a...z]. The password may not have consecutive repeating characters, for example, “aardvark” has the repeating characters “aa” so it would not be allowed. The password is stored on the server such that the plain text can be accessed in order to verify the trace.

When the user logs in, the server randomly generates a 26x26 Latin square with the characters [a…z] called the *Grid*. The server also randomly selects a cell and a direction (either horizontal or vertical) as the *start location*. The server transmits this Grid to the user. The Grid and the start location are unique for each log in. The server stores the Grid and start location in temporary state and provides a pointer to this state called the *token* to the user. The user’s browser returns the token to the server on each log in attempt.

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 between vertical and horizontal for the length of their password. If the first letter is the start location, the server will select another random start location.

The user enters the directions (up, down, left, right) that they follow as they trace out their password. This is called the *trace* of the password. The trace and the token are sent back to the server.

The server verifies that the trace by replaying the trace and making sure the password letters match the provided trace.

The server will only accept 2 traces per token. If a user guesses incorrectly twice, the server will present the user with a new Grid and Start Location. The server will lock the account and the IP address after four incorrect tries until the user completes an email loop.

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

Start: Vertical

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | https://www.grc.com/image/transpixel.gifhttps://www.grc.com/image/transpixel.gifg | | 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 | v | 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 | l | 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 | c | 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 | v | 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, Down, Left, Down.

**Figure 9 A trace of the password “daisies”**

## Modified CrossPassword

We also explored a modified version of this system designed to increase usability. This system uses a 13x13 grid, instead of a 26x26 grid to make it easier for users to visually scan the grid.

In addition, we no longer generate a Latin Square. Instead, we first randomly distribute the user’s password in an empty grid. We first randomly select either a row or a column from our 26 choices. We then place the first letter somewhere in that row or column. For this example, say we select the 3rd column to start with. We then place the “a” somewhere in this first column. We then place the second letter “m” in the row in which we have placed the first letter. We continue this scheme until the password has been placed.

For example:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | a |  |  |  | m |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | n |  |  |  |  |  |  |  |  | o |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | a |  |  |  |  | z |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**Figure 10 Password Characters Randomly Filled In**

We then randomly fill in the remaining letters on the grid from the set of 26 lower case letters. We make sure each row and column only contains each letter only once by backtracking. For each spot we first start with the entire set [a…z]. We then remove all letters that are currently in the same row and column that we are in. We then randomly select a character from the remaining set.

This is not a Latin Square because we have a 13x13 Grid, but 26 possible characters.

With this system we must prevent an attacker from looking at a grid a certain amount of times. A user can only look at up to 4 random grids before their IP address is locked out. In addition, a particular user account can only have 4 grids shown before that account is locked out as well, in the event the attacker is using a distributed attack.

## GrIDsure

As a comparison, we will evaluate our proposal against GrIDsure as described in The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes.[[3]](#footnote-4) GrIDsure is also a cognitive authentication scheme in which the user attempts to prove to the server that it knows a secret without actually revealing the secret.

At registration, the user is presented with a 5x5 grid and selects a pattern of 4 cells. We call this the *password* for consistency with our other methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 |  | 3 |  |
|  |  |  |  |  |
|  |  | 2 |  |  |
|  |  |  |  |  |
|  |  |  |  | 4 |

**Figure 11 A length-4 pattern selected**

At login, the user is again presented with a 5x5 grid, this time with a random digit in each cell. The user then transcribes the digits from his or her length-4 pattern into the nearby box. For consistency, we call this the *trace*. The digits differ each time because they are chosen randomly each time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 3 | 4 | 8 | 9 |
| 2 | 2 | 7 | 4 | 4 |
| 3 | 6 | 9 | 6 | 3 |
| 8 | 4 | 6 | 7 | 0 |
| 9 | 8 | 1 | 0 | 7 |

**Figure 12 The password here would be “3987”**

# Analysis of Attacks

We evaluate the security of the three systems described above using a probability analysis. In eavesdropping on a user, an adversary may obtain the user’s board/grid, the trace, or both. Given each of these pieces of information we see how the user may be compromised. The attacker may be able to just log in through brute force chance; or the attacker may be able to backsolve for the original password.

## Original Off the Grid

The Off the Grid system offers users a unique password given the domain name. The user simply traces the domain name in the two phases described above.

### Intercepted Grid (Theft or Video Observation)

The first attack may occur if an adversary obtains a user’s grid. This is usually held on hand by the user, and may occur if their wallet is stolen.

If an attacker is able to retrieve the grid of a user, he has access to all of the user’s passwords. The attacker can follow the Off the Grid protocol for any possible domain name in order to obtain the password for that site. This is especially true if the user sticks to the standard Off the Grid protocol. For example, the algorithm specifies starting in the first row.

However, the protocol author recommends making personal tweaks to the algorithm, for example, choosing a custom, personal, secret start location. If someone adopts a personal start location, this leads to 104 possible passwords if the attacker gets control of the grid. 104 choices is still easily brute forcible. Other tweaks can add further additional possibilities.

We do not, however, officially consider these tweaks in evaluating the algorithm. Nevertheless, the secrecy of the grid is paramount in preserving the integrity of the Off the Grid; if the grid is stolen, an adversary has access to all passwords for all websites.

### Observing a Single Site (Internal Observation, SSL Middleman, or Phishing Site)

If an attacker instead obtains the user’s password for a single site, only interactions with that website are compromised. In the description described above, the password “gaznegmacmzg” is obtained by following the two phases of the Off the Grid system for Amazon.com. If the attacker obtains “gaznegmacmzg,” he can use it to log in the user’s account for Amazon.

### Rebuilding the Grid

With one password, the attacker obtains very little information regarding the grid, and the resulting passwords for other website domains. It is very difficult to use one, or even a collection of passwords, to rebuild the Grid. Given the stock Off the Grid implementation described, there are at least boards for an n-sized board, which leads to at least boards for . Even if you knew the start location (the first row), there are 26 possible slots for the first character. There are then 26 possibly for the next character, etc. In addition, because one wraps around, no information is leaked about boundaries. Even with many, many domain and password pairs it is infeasible to recreate the grid and/or generate new passwords for a given new domain name.

### Brute Forcing a Password

Under the standard Off the Grid protocol (where we always use the first 6 characters of the domain name), there are possible submissions to the website. This is hard to brute force.

## CrossPassword

Our CrossPassword implementation relies on the security of the Latin Square. Since each password must be made up of only lower case letters and no-repeating characters, the password can be searched using a Latin Square as specified above, resulting in at least possible grids.

### Intercepted Board

If the attacker gains access to the board the user sees along with the start location, the attacker gains very little information on the password of the user even after multiple board configurations are given. Because the first letter of the password will never be the start location, with many copies of the grid, the attacker could eventually see that there is one letter which is never the start location. This occurs 1/26 times, but an attacker needs ~50-70 boards to confidently say that the 1 letter will never show up.

With 78 boards they have a

= .000005 chance of having a location never show by accident, so it is likely that the space that remains is the start location. This only reveals the first letter.

### Brute Forcing Traces

An adversary can do a brute-force guess on the submission. We can use the fact that the direction changes each turn, thus Left and Right will always be followed by Up or Down along with the start location to realize that there are only 2 possible responses for each character of the password.

If the attacker doesn’t know the length of the password, then they must try multiple combinations

This sum does not converge. However, if the attacker assumes a maximum length of 10, then there are 2046 possibilities. This is not very large.

### Intercepting Just the Trace

If the attacker instead gains only access to the user’s key inputs (but not the grid) through key-logging software, thus obtaining their input, they can never retrieve the password. The password will be impossible to obtain from only getting “up, right, up, left, down” and so on. The only information gained is the length of the password. Even with multiple traces, the password will be impossible to obtain without the accompanying board. However given the length of the password, in a brute force attack the user knows exactly how long the trace to input; an attacker can now do a smarter brute force attack of the exact length of the password.

If the attacker knows there are 6 characters in the password then there are only possible combinations which we can then brute force. After finding out the length of a password through the trace, we know there are then possibilities given, , the length of the password.

### Lockout

Because this scheme is highly susceptible to brute force attacks with a probability of guessing correctly, we can reduce the effectiveness of brute force attacks by the frequency of the board change and lockout. If we lockout after every 4 attempts, then there is a chance of guessing a specific user’s password of

before being locked out given that all users choose passwords less than or equal to 10 characters. This means an attacker will likely figure out the password of about every 500th user. The attacker would need to use a range of IP addresses to avoid IP blocking.

### Intercepted One Board and Trace: Password Recovery?

Lastly, in the case when an attacker gains information to the board, the start-location, and the trace. Even when the trace is intercepted along with its board, it is extremely difficult to recover the password with one of these sets as can be seen in the example below where the password is “daisies.”

Start: Vertical

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | i | 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 | v | 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 | d | 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 | c | 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 | l | 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 | c | 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 | v | 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 |

Trace: Down, Left, Up, Right, Down, Left, Down

**Figure 14: A board, start location, and trace have been intercepted. It remains difficult to recover the password. Possibilities for the first letter are in black, 2nd letter in dark gray.**

There are many possible combinations of letters that will satisfy the trace. An attacker will have to obtain all combinations that satisfy the trace, then try to determine which is the password. For example in Figure 14, there are 19 options for the first letter (in black) then 19\*21 possible options for the second letter (in medium gray) because we know our start column is the 5th column. However, because this is a Latin Square, we know that within these 399 cells exist all 26 letters.

Start: Vertical

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | i | 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 | v | 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 | l | 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 | c | 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 | v | 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 |

Trace: Down, Left, Up, Right, Down, Left, Down

**Figure 14: A board, start location, and trace have been intercepted. It remains difficult to recover the password. Possibilities for the third character are in light gray.**

For the third character, we know we are going up again. We know that we are somewhere within the light grey. Note that we know that we can’t be on the bottom row for this character. This is 19\*25 possibilities; we know again that again all 26 letters exist because it is a Latin Square.

Start: Vertical

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | i | 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 | v | 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 | l | 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 | c | 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 | v | 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 |

Trace: Down, Left, Up, Right, Down, Left, Down

**Figure 14: A board, start location, and trace have been intercepted. It remains difficult to recover the password. Possibilities for the fourth character are in light gray.**

For the fourth character, we know that we can be anywhere except the last column (since we just went up) and the last row (since we are going right). There are 625 possible cells where we can be at now, again containing all 26 characters, by the properties of a Latin Square.

It continues like this for the rest of the password. There are

possible sequences of grids which we visited, leading to the same number of possible unique passwords (since it is a Latin Square). Thus for each board, location, and trace there are significantly too many possible passwords, that knowing the board and the trace does not reveal the underlying password.

#### Dictionary Word

If the user selects a dictionary word, it is much easier for the attacker to recover the user’s original password. The attacker can attempt to run every dictionary word through the observed trace offline to see if any words fit the entire trace. The attacker then has only a few possibilities to try online when guessing on a new grid. There are about 500,000 words in the dictionary.[[4]](#footnote-5) Since the attacker can try these offline, they can process through this quite fast.

### Multiple Boards and Traces

If an attacker however collects multiple sets of boards, locations, and traces, he can attempt to cut down the window of possible letter combinations by narrowing down the number of possibilities. Say for instance the attacker has two grids and traces. In each one, there are a number of possible letters which are to the known direction from the start location.

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

**Figure 14: Two different first columns where the first letter is c.**

The attacker will then look at the union of the two shaded sections. In this case they can see that the only characters in the union are c, d, and q. With yet another grid and trace, the attacker can narrow this.

With 1 trace, the attacker will know have 12.5 possible characters on average; with 2 traces, there are 6.25 possible characters on average. With 5 traces there is likely to be only 1 possible character.

|  |  |
| --- | --- |
| 1 trace/board | 12.5 possible characters |
| 2 traces/boards | 6.25 possible characters |
| 3 traces/boards | 3.125 possible characters |
| 4 traces/boards | 1.5625 possible characters |
| 5 traces/boards | .78125 possible characters |

**Table Expected number of letters that will be still be present at random after specified number of traces/boards**

The attacker can then step through each letter of the password looking at one letter at a time; using the grids and traces he already has. He can thus recover the password with approximately 5-6 complete grids and traces.

## Modified CrossPassword

While our CrossPassword implementation relies on the security of the Latin Square, the modified version is no longer a Latin Square having a reduced size, , but still using all 26 letters as possible letters.

### Intercepted Board

Unlike our previous implementation, the adversary gains information looking at even a single board and start location, but without a trace. After looking at a few boards, the adversary may be able to determine all the letters of the password. To reiterate, the attacker just needs to refresh the log in screen!

The adversary is able to get a copy of the board and start location by simply visiting the login page of the server and entering the target’s username. The fatal flaw is that the password letter is always present in the start row/column. The other letters are present with a 12/25 chance. To reiterate, we know that one letter will show up 100% of the time, with the remaining 12 letters having been randomly selected from the remaining 25 letters. This already cuts down the possibilities for the first letter of the underlying password to about ½ of all possible letters. By simply refreshing the page and getting a new board to see, the adversary can cut down the possible letters even more.

|  |  |
| --- | --- |
| 1 board | 12 letters |
| 2 boards | 6 |
| 3 boards | 3 |
| 4 boards | 1.5 |
| 5 boards | .75 |

**Table Expected number of letters that will be present at random.**

He continues this process until there is only 1 letter that is always present across all boards in the start row/column; this letter is the first letter of the password. This will take about 4-6 grids.

After the first letter is known, the adversary can repeat the process one letter at a time to calculate all letters in the password. The adversary can also reuse the grids they have already retrieved; there is no need to get new grids. The adversary knows to stop (i.e. the length of the password) when there is no longer a letter that is present 100% of the time in the next row/column.

### Intercepted Trace

Similar to our Original CrossPassword implementation, the adversary gains very information about the underlying grid when they observe just a trace, or when they observe multiple traces, without the accompanying board. They do however know the length of the password, which greatly reduces the space they must brute force over.

### Brute Forcing

The Modified CrossPassword is also vulnerable to the same brute force attack as the regular system.

### Intercepted Board and Trace

Lastly, there is the possibility that the attacker has all information: the board, the start-location, and the trace. This equates to the information gained from have the board and its accompanying starting location, and the information gained from having the trace. The adversary can do a similar process using both the direction from the trace, as well as eliminating letters which only show up once. This leads to a greatly reduced number of possible letters.

Using just information for the trace:

|  |  |
| --- | --- |
| 1 trace/board | 6 possible letters |
| 2 traces/boards | 3 letters |
| 3 traces/boards | 1.5 possible letters |
| 4 traces/boards | .75 possible letters |

**Table Expected number of letters that will be still be present at random after specified number of traces/boards**

So with two traces/boards what is the probability that a letter is present twice in given direction at random?

= 5%

This means that is a letter is present, it is likely that it since it is the password, not that this letter has appeared at random. Thus with just two grids/traces you are likely to have the password.

## GrIDsure

There are possible combinations of length-4 patterns. However, there are possible inputs to return.

### Intercepted Board

If an attacker is able to successfully intercept the board and no other information, he gains no information about the password. Each slot in the board is randomly filled with an integer. Even after intercepting multiple copies, the attacker gains no information for the user’s password.

### Brute Forcing

There are possibilities, which is ok by our standards.

### Intercepted Trace

Equally, if an attacker is able to intercept the 4-digit trace but not the board, again no information is gained about the user’s password. Each number can be equally represented by any of the 25 slots.

### Intercepted Board and Trace

We cannot exactly map one “trace” from the grid to the squares that the user selected because there are 25 grid locations but only 10 digits. Thus each digit will be in the grid an average of 2.5 times.

So if an attacker is able to intercept the board and the trace, after intercepting multiple copies, he would be able to determine the user’s password. We can begin by looking at the first digit, because we know each digit is located on the grid an average of 2.5 times, we can say that the first digit is located in 3 locations on the grid, one of which will correspond to the first slot of the password. The next time we intercept a new board and trace, we will again get a new digit that is located in approximately 3 locations on the grid. We know for a fact that one of these locations must overlap because it will be a part of the user’s password.

However, there is a small possibility that the other 2 locations may overlap with the original 2 other locations and thereby having more than one possibility for the underlying cell which is part of the user’s password. The probability of the second two location overlapping exactly with the first two locations is . This is a low probability of gaining conflicting information about the slot that corresponds to the each “character” of the password. We need to extend this to multiple characters in the password. After intercepting more copies, this probability decreases significantly. So we know that at each new board, we will likely gain information to determine the slot that corresponds to the first slot in the password. This same analysis can be done to determine the slot for the second through fourth slot of the password.

In the original paper, the authors conduct a Monte Carlo simulation on 1,000,000 attacks and find that they need an average of 2.66 grids/traces, with a maximum of 8 grids/traces to recover the original password.[[5]](#footnote-6)

|  |  |
| --- | --- |
| # of grids/traces captures | # of grids reverse engineered (of 1,000,000) |
| 1 | 129 |
| 2 | 422230 |
| 3 | 496400 |
| 4 | 72871 |
| 5 | 7527 |
| 6 | 770 |
| 7 | 69 |
| 8 | 4 |

**Table Number of captures needed to reverse engineer a password.**

# Other Factors

We evaluate each system according to the criteria set out in The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes.[[6]](#footnote-7)

## Improvements to Criteria

### Resilient-to-Physical Observations Category

We think that the **Resilient-to-Physical Observations** category should be split in two: **casual observation** and **video observation**. Casual observation is if an attacker is just able to watch the user enter their password once. This is feasible for short passwords and/or if the user types slow. An attacker can see which keys are hit on the keyboard. This is especially true if the user types slowly, has a short, and/or easily remember-able password.

However, the attacker seeing the user trace out the password on the grid once would have trouble remembering the entire grid, preventing the total loss of the password scheme. For that specific domain name, many attackers would have trouble remembering the sequence of 12 random characters, providing some additional security.

Video observation is defined as the attacker having the full ability to carefully watch and study users’ movements because the attacker is able to pause and replay the user’s log in actions.

### Resilient-to-Throttled-Guessing

To better demonstrate the differences between our protocols, we assign **Resilient-to-Throttled-Guessing** if there are more than 10 possible choices all of equal weight. This is much small than the original paper requires. The paper considers choices to be NOT **Resilient-to-Throttled-Guessing**.

### Inherently-Discoverable

Must the user seek out the new password system? Or does the server require that the user use it? Often new schemes that fit within the structure of existing passwords remain undiscoverable to the user. We want to highlight schemes where the website helps the user discover them. A scheme gets a YES here if the server is required to notify and teach the user of the new scheme.

### Resilient-to-SSL-Proxy-Man-in-the-Middle

Assume that there is someone who is listening in on the wire who can decrypt SSL, for example, a corporate SSL proxy. Does this person have enough information to log in? YES, if they can do so after observing 1 log in. QUASI, if they must observe several log ins in order to have this power. We assume that the initial registration process is outside this scheme.

### Allows -User-to-Choose-Any-Password

Does the system allow the user to choose any password (as defined by the usual set of characters allowed in a password)? Or does the system limit the user’s set of password to a certain length or set of characters? Or use a totally different memory scheme? Users might use the same password on multiple sites or have an external scheme to generate a password. One could argue that preventing a user from using the same password on each site is a good thing, but a password scheme should not do so by limiting the choice that a user has in selecting a password.

### Denial-of-Serviceable

An active attacker can cause a denial-of-service attack by submitting a sufficient quantity of incorrect passwords such that the system locks the user out. Lockouts can add additional security by preventing more than a handful of guesses by the attacker. However, they can considerably impede usability as they can require a user to either wait or to seek out help from a system administrator. If these are tied to a user account, an attacker can deliberately use up these guesses to mount a Denial-of-Service attack on the user. If this is possible with a few incorrect submissions from any IP address, we assign a YES here.

## Original Off the Grid

### Usability benefits

1. **Memorywise-Effortless YES** There are no secrets to be remembered in the base case. The description mentions a more advanced case, where the user could start at a different location, but we are assuming the base case where the user automatically selects the same location.
2. **Scalable-for-Users YES** The user only needs one grid for all of their sites.
3. **Nothing-to-Carry NO** User must carry 1 sheet of paper
4. **Physically-Effortless NO** The user must trace out their password on paper
5. **Easy-to-Learn NO** Using the same rubric as the paper does, the scheme is quite complicated
6. **Efficient-to-Use NO** The scheme requires a fair amount of effort for each authentication.
7. **Infrequent-Errors NO** Tracing out the password on the grid twice is easy to mess up.
8. **Easy-Recovery-from-Loss KINDA** If the user lost their Grid, they must have another copy of their Grid, or the key used to generate that Grid. A user can always reset their passwords on each site. The paper rates generic passwords as **Easy-Recovery-from-Loss YES**.
9. **Inherently-Discoverable NO** A user must learn about this scheme by visiting the GRC website.
10. **Allows -User-to-Choose-Any-Password NO** The password is based off of the domain name of the site.

### Deployability benefit

1. **Accessible NO** There could conceivably be a braille-based grid, but not at this moment. In addition, someone with poor motor control will find this scheme very difficult.
2. **Negligible-Cost-per-User YES** The user is required to print one sheet of paper which costs < 05 cents.
3. **Server-Compatibility YES** One of the primary benefits of this scheme is that it is compatible with existing servers which use passwords
4. **Browser-Compatibility YES** No special browser is needed
5. **Mature KINDA** The scheme has been published for some length of time; at least one Android app exists with support.
6. **Non-Proprietary YES** The scheme is published fully.

### Security benefits

1. **Resilient-to-Physical Observations-Casual KINDA** The attacker would have to remember 12 random characters in order to observe the user’s password for that site.

With just a casual observation there is no way the attacker can memorize the entire Grid.

1. **Resilient-to-Physical Observations-Video NO** If the attacker can take a picture of the Grid, for example, a video camera over the shoulder, then the attacker would have access to all of the users’ passwords assuming the user is using the standard Of the Grid scheme.
2. **Resilient-to-Targeted-Impersonation YES** Personal knowledge cannot help for the Off the Grid scheme. However, the normal password recovery mechanisms of the website remain, which are generally very vulnerable to Targeted Impersonation.
3. **Resilient-to-Throttled-Guessing YES** The user’s password is 12 random alphanumeric characters. This means there are possible passwords.
4. **Resilient-to-Unthrottled-Guessing YES** There are possible passwords.
5. **Resilient-to-Internal-Observation NO** Off the Grid reduces to a normal 12 character password unique for each domain. This password is the same for each log in.
6. **Resilient-to-SSL-Proxy-Man-in-the-Middle NO** The password is the same for each log in; it must be protected with some additional protection (such as SSL) in transit.
7. **Resilient-to-Leaks-from-Other-Verifiers YES** Ideally the server should be hashing the password. Regardless, each domain has a unique password so leaking one password does not give one feasible information about another domains’ password.
8. **Resilient-to-Phishing NO** If the attacker is able to spoof the domain name of the site, then the user will follow the same trace on the grid, providing the attacker their password.
9. **Resilient-to-Theft NO!** If the attacker gets your grid, it’s game over, assuming you are sticking to the base Off the Grid algorithm. The author suggests that you make small personal tweaks to the algorithm in order to add resilience to theft.
10. **No-Trusted-Third-Party YES** The third party provides the code to generate the grid. However, that code runs in JavaScript on your local computer, allowing you to verify that the code is actually generating a unique grid and is not sending a copy to the third party. One could also write ones’ own implementation of the Grid generation scheme to be sure.
11. **Requires-Explicit-Consent YES** The user must trace their password on the grid and then enter it onto the computer.
12. **Unlinkable YES** Since each user’s Grid is so different, there is no feasible way to link users using the same scheme.
13. **Denial-of-Service-able NO** This is the same as normal passwords. Under a normal password system, services generally do not add a lockout provision.

## CrossPassword

Goal: prevent from seeing over wire

Note all are for the actual log in experience. This analysis does not consider creating a password; the process of which is similar to traditional password schemes.

### Usability benefits

1. **Memorywise-Effortless NO** The user must remember a password to use CrossPassword. Ideally, that password should be different between sites. Since we only allow lowercase alphabetic characters without repeating letters, we may prevent users from using the same password on a site running CrossPassword than the user uses on all of their sites.
2. **Scalable-for-Users NO** Ideally the user has a different password for each site
3. **Nothing-to-Carry YES** There is nothing to carry
4. **Physically-Effortless NO** The user must trace out their password on-screen
5. **Easy-to-Learn NO** Using the same rubric as the paper, the scheme is quite complicated
6. **Efficient-to-Use NO** The scheme requires a fair amount of effort for each authentication.
7. **Infrequent-Errors NO** Tracing out the password on screen is easy to mess up
8. **Easy-Recovery-from-Loss YES** CrossPassword falls back on the same recovery mechanisms as traditional password sites, which is rated YES in the paper.
9. **Inherently-Discoverable YES** A user will discover the CrossPassword scheme when attempting to create an account on a server that uses CrossPassword
10. **Allows -User-to-Choose-Any-Password NO** The user can only choose a password using the letters [a…z] and the user cannot repeat the same characters twice, as in “aardvark.”

### Deployability benefit

1. **Accessible NO** A screen reader would be tedious to use. In addition, someone with poor motor control will find this scheme very difficult.
2. **Negligible-Cost-per-User YES** There is no cost.
3. **Server-Compatibility NO** The server must be provisioned with a new authentication library.
4. **Browser-Compatibility YES** No special browser is needed
5. **Mature NO** We are proposing it here
6. **Non-Proprietary YES** The scheme is published fully.

### Security benefits

1. **Resilient-to-Physical Observations-Casual POSSIBLY** If the attacker could see the screen and the keyboard they could not uncover the user’s password, unless the user traces the password with their finger.
2. **Resilient-to-Physical Observations-Video POSSIBLY** Even with being able to study the user as they enter their password, the attacker would not be able to recover a user’s password, unless the user traces the password with their finger. This is one of the major design goals of this system.
3. **Resilient-to-Targeted-Impersonation YES** Personal knowledge cannot help for the Off the Grid scheme. However, the normal password recovery mechanisms of the website remain, which are generally very vulnerable to Targeted Impersonation.
4. **Resilient-to-Throttled-Guessing YES** An attackercan only submit two tracers per grid/start location. After two tries, the server will issue a new grid. The user then gets two more tries at a trace submission before the account is locked until an email loop is performed.
5. **Resilient-to-Unthrottled-Guessing NO** Due to the very small number of possible responses (for example, =64 for a 6 character password, there are very few bits of entropy so the system falls fast.
6. **Resilient-to-Internal-Observation QUASI** This is the major design goal of this system. An attacker needs 5-6 observations of the *grid*, *Start Location*, and *trace* in order to crack the password. This is sharply reduced if the user picks a dictionary word, however.
7. **Resilient-to-SSL-Proxy-Man-in-the-Middle QUASI** This is the same as Internal-Observations. If a listener on the wire who was able to remove the SSL encryption, then they would need 5-6 observation in order to recover the password.
8. **Resilient-to-Leaks-from-Other-Verifiers NO** The password is stored in plain text on the server in order for the server to verify the password. This is not good practice.
9. **Resilient-to-Phishing YES** An attacker with just one trace could not submit that trace to another server, because the grid is randomized each time. At attacker could mount a man-in-the-middle attack and proxy the grid, but the rubric in the paper does not penalize for this.
10. **Resilient-to-Theft YES** There is nothing to steal
11. **No-Trusted-Third-Party YES** There are no 3rd parties involved
12. **Requires-Explicit-Consent YES** The user must trace their password on the computer and enter the trace.
13. **Unlinkable YES** Like passwords, this scheme is unlinkable.
14. **Denial-of-Service-able YES** An attacker can lock out an account by trying an incorrect password 4 times.

## Modified CrossPassword

The modified CrossPassword is more **Efficient-to-Use** and has less errors (**Infrequent-Errors**), however at the cost of a slightly decreased **Resilient-to-Physical Observations-Casual** and **Resilient-to-Physical Observations-Video** if a user traces the grid because of the smaller grid. The degree is reduced, but the broad scores remain the same.

However, **Resilient-to-Internal Observation** and **Resilient-to-SSL-Proxy-Man-in-the-Middle** take big hits as an attacker can discover a user’s password (or at least have a very high chance of finding it) using 5-10 copies of the random grid and start location. They don’t need any copies of the trace, though having at least 1 would let them need less copies of the grid. This makes the scheme vastly weaker. **Resilient-to-Throttled-Guessing**, **Resilient-to-Unthrottled-Guessing** switch to no because the attacker has that high chance of recovering the password and might only need to make 1-2 guesses. In fact, we don’t even have a category for how bad this is: **Crackable-From-Reloading-Log-In-Page**?

Because we now need to protect from the attacker seeing the grid multiple times, modified CrossPassword is even more **Denial-of-Service-able**.

## GrIDsure

GrIDsure is evaluated in The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. Here we evaluate the new metrics we have introduced and make additional comments about some metrics.

It is important to note that the authors rated it as not **Resilient-to-Throttled-Guessing** or **Resilient-to-Unthrottled-Guessing** because the space of possible is so small (. In this paper, we rated our other schemes as **Resilient-to-Throttled-Guessing** if the attacker has less than 10 possible choices, so we would rate this as **Resilient-to-Throttled-Guessing** if it has a rate limiter/lockout.

Because the server must tell the user about the scheme, GrIDsure is **Inherently-Discoverable**. However, it requires users to remember a sequence of 4 unmarked boxes in a 5x5 grid. Thus it is clearly not **Allows -User-to-Choose-Any-Password**.

When the user does not place their finger to the screen, it is **Resilient-to-Physical Observations-Casual.**

Because of the small number of possibilities it is not **Resilient-to-Internal-Observation**, **Resilient-to-SSL-Proxy-Man-in-the-Middle**, or **Resilient-to-Physical Observations-Video**. With two observations, it is pretty much game over, as the attacker is able to discover the original sequence of boxes.

## Comparison Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Of the Grid | CrossPassword | Modified CrossPassword | GrIDsure |
| **Memorywise-effortless** | Yes | No | No | No |
| **Scalable-for-users** | Yes | No | No | No |
| **Nothing-to-carry** | No | Yes | Yes | Yes |
| **Physically-effortless** | No | No | No | No |
| **Easy-to-Learn** | No | No | No | Yes |
| **Efficient-to-Use** | No | No | No (More) | Quasi |
| **Infrequent-Errors** | No | No | No (More) | Quasi |
| **Easy-Recovery-from-Loss** | Kinda | Yes | Yes | Yes |
| **Inherently-Discoverable** | No | Yes | Yes | Yes |
| **Allows -User-to-Choose-Any-Password** | No | No | No | No |
| **Accessible** | No | No | No | No |
| **Negligible-Cost-per-User** | Yes | Yes | Yes | Yes |
| **Server-Compatibility** | Yes | No | No | No |
| **Browser-Compatibility** | Yes | Yes | Yes | Yes |
| **Mature** | Kinda | No | No | No |
| **Non-Proprietary** | Yes | Yes | Yes | No |
| **Resilient-to-Physical Observations-Casual** | Kinda | Possibly | Possibly (Less) | Yes |
| **Resilient-to-Physical Observations-Video** | No | Possibly | Possibly (Less) | No |
| **Resilient-to-Targeted-Impersonation** | Yes | Yes | Yes | No |
| **Resilient-to-Throttled-Guessing** | Yes | Yes | No! | Yes |
| **Resilient-to-Unthrottled-Guessing** | Yes | No! | No (Less) | No |
| **Resilient-to-Internal-Observation** | No | ~5-6 | 4-6 just log in screen! | ~2 |
| **Resilient-to-SSL-Proxy-Man-in-the-Middle** | No | Yes | No! | No |
| **Crackable-From-Reloading-Log-In-Page** | No | No | Yes | No |
| **Resilient-to-Leaks-from-Other-Verifiers** | Yes | No! | No! | No |
| **Resilient-to-Phishing** | No | Yes | Yes | No |
| **Resilient-to-Theft** | No! | Yes | Yes | Yes |
| **No-Trusted-Third-Party** | Yes | Yes | Yes | Yes |
| **Requires-Explicit-Consent** | Yes | Yes | Yes | Yes |
| **Unlinkable** | Yes | Yes | Yes | Yes |
| **Denial-of-Service-able** | No | Yes | Yes (More) | No |

**Table : A Comparison of Off the Grid, CrossPassword, Modified CrossPassword, and GrIDsure.**

# Usability

We will now explore what the field of usability tells us about our password schemes. [[7]](#footnote-8) The three core tenants of usability are: learnability, efficiency, and safety.

At the core, the simpler a system is, the more it will be used. Security is often a tradeoff between usability and security. A successful scheme should add security, without impacting usability too much.

## Learnability

### Discoverability

In order for a system to start being used, it must be discoverable.

CrossPassword is more discoverable than Off the Grid because the website you are creating an account with can let you know that the website uses CrossPassword. It is inherently discoverable. Off the Grid requires that you hear about the system in some way. Websites can still advise you of the presence of Off the Grid, but the Off the Grid system, as currently designed and designated, is not inherently discoverable.

### Training

It’s important that a user know how to use a particular system.

CrossPassword can be taught to users when they pick their password for the site. For example, sites could show users a video of how to use CrossPassword. Sites could also provide an interactive training tool using CrossPassword that uses JavaScript and HTML 5 to show the user how to trace their actual password. (Using the actual password would reveal the user’s password to a shoulder surfing attacker, but this may be appropriate for a secure room. The password would be stored in the DOM during registration, but this happens with a normal registration system as well)

### Mental Model

When users interact with a system, they form a mental model of how that system operates “behind the scenes.”

We believe that once CrossPassword is explained to a user, it is easy for that user to form a mental model of the system. The server asks you to solve a puzzle and you solve it. In addition, the rationale behind the system is also clear; it is clear that this prevents you from sending your password over the wire for subsequent log ins. Users should be able to understand how the system works. Each log in is consistent with the rules of the system and ones’ mental model of the system.

## Efficiency

Each log in should not take a long time. This is because user’s time is valuable. In addition, users will be more likely to keep using the system if it is fast.

Whereas Off the Grid requires users to trace the grid twice, CrossPassword only requires a user to trace the grid once.

Off the Grid also requires one to enter two characters for each letter in the domain name during its Phase 2. CrossPassword is more natural to use than Off the Grid because one can trace the system on the screen as one enters the keyboard traces. We feel that expert users of CrossPassword could use the arrow keys without taking their eyes off the screen. This could make password entry quite fast.

However, both CrossPassword and Off the Grid are slower than traditional password schemes, or even password managers, such as LastPass. Users may not want to adopt a system that is slower than what they already have.

### Chunking

Research has shown that people can remember 7 ±2 pieces of information at once.[[8]](#footnote-9) A piece of information could be one letter. When letters are combined into an English word, that word is now one piece of information. To reiterate: a collection of 7 random letters are 7 pieces of information. However, a word comprised of 7 letters is only 1 piece of information.

We can use this to evaluate whither a causal visual observer (shoulder surfer) could observe a password off the screen. With Off the Grid, it would be difficult for an attacker to remember 12 characters using just their short term memory. This is why we rated it as KINDA for Resilient-to-Physical Observations-Casual.

## Fitts’s Law

Fitts’s Law is an estimate of the time it takes someone to point to an object or steer among objects.[[9]](#footnote-10) The rule as formulated by Scott MacKenzie is as follows:[[10]](#footnote-11)

where:

* *T* is the average time taken to complete the movement
* *a* represents reaction time to start moving
* *b* stands for the speed of movement
* *D* is the distance from the starting point to the center of the target.
* *W* is the width of the target measured along the axis of motion. *W* can also be thought of as the allowed error tolerance in the final position, since the final point of the motion must fall within ±*W*⁄2 of the target's center.

We can use a more specific form to study steering tasks, the time to move your hand through a tunnel of length D and width S:

The index of difficulty is now linear.

We can use this to measure the amount of time it takes someone to trace through the grid, if they trace the grid with their finger or mouse. Ideally the user should not do that to maintain **Resilient-to-Physical Observations-Casual** and **Resilient-to-Physical Observations-Video**.

Assume a = 0 and b=200ms/bit for a mouse, using the upper limit of the empirical study.[[11]](#footnote-12) Assume the user must travel 26 cm to reach a 1 cm square block. If a user had to steer within a row, this leads to an approximate time to trace of

=5.2 seconds for traveling a row or column.

This is the worst case possibility: the user (worst case empirical user) is using a mouse to travel the entire length of a row/column and they cannot leave the row/column with their mouse at all. This is

## Improving Usability

We can do things to improve usability. For example, we can shade every other row or column, alternating between row and column on every user input.



**Figure 18 Every other column is shaded**

This gives us two benefits. First, the user can now easily see if they should move horizontally or vertically next. Second, it is easier for the user to keep their eye in the same column/row as they scan the grid vertically/horizontally for their next letter. This should decrease mistakes as well as decrease the time it takes people to solve the grid.

## Auto-Solver

It is possible to build a browser-based auto-solver for CrossPassword grids. This software would know the user’s password and use that to automatically solve grid challenges. This would break **Resilient-to-Internal Observation** because the user’s system would now need the password stored. However the system would still meet **Resilient-to-SSL-Proxy-Man-in-the-Middle** . It would do a great deal for usability, flipping **Physically-Effortless**, **Easy-to-Learn**, **Efficient-to-Use**, and **Infrequent-Errors** all to yes. In addition **Accessibility** would greatly improve.

# Code

We built a prototype of CrossPassword to demonstrate the feasibility of implementing the system. Our code consists of a server-side component and a client-side component. The server runs a Flask Python server and keeps track of the user accounts in a PostgreSQL database. The client code is a Javascript file and HTML login page that is delivered by the server.

When the user requests to login as a specific username, the client-side code sends a request to the server for a board for that username. The server, upon receiving this request, generates a random 26x26 Latin square consisting of the lowercase letters of the alphabet and the starting coordinates and direction. This information is stored for the user in the database. The server then sends the board and starting square information back to the client. The client-side Javascript code inserts the board into the login page and marks the starting square and the starting direction (indicated by highlighting every other row/column and by text).

The user then traces out the password starting from the start square by using the arrow keys. The direction alternates between horizontal and vertical. For example, if the starting direction is horizontal, then the user will press either the left or right arrow key depending on where the first letter of the password is in that row. Then the board will change and indicate that the next direction is vertical. The user will then press either the up or down arrow key depending on where the second letter of the password is in that column. The user repeats this until the last letter of the password is traced, and then the user clicks the login button to submit the trace. As the user traced out the password, the client-side code converted the arrow key presses to the letters: 'u', 'd', 'l', 'r' and appended them to the end of the trace string. The final trace string might be something like 'dlurdl'. This trace string is sent send to the server through a form POST request (the username is also sent in a hidden field).

To verify the trace, the server looks up the username in the request, and gets the board and the starting information. The server then looks at the trace string and tests if the trace can be used to find the password in the board. If so, then the server logs the user in and directs the user to the home page that is only accessible to signed-in users.

New users can be created through the registration page by providing a username and password (and confirmation password by entering the password again). The password is checked to make sure that it can be place into a board. In order to be place in a board and satisfy the Latin square constraints, the password must consist only of lowercase alphabet letters and must not have consecutive repeating letters. There is also a minimum password length requirement.

To see a working version of the implementation, go to [crosspassword.herokuapp.com](http://crosspassword.herokuapp.com)

### Limitations of Current Code

We have not implemented any sort of lockout system in our sample implementation. This means that brute force attacks will be very easy to execute.

# Conclusion

CrossPassword is not recommended as a password system. Modified CrossPassword turned out to be even weaker than we first imagined.

We tried to build a zero-knowledge interactive proof. A zero-knowledge interactive proof is one in which the prover needs to show that they know the solution to the verifier. The prover in this case is the user, and the verifier is the server. The verifier asks questions to the prover, who responds with an answer. If the prover does in fact know the answer, he or she will always answer the verifier’s question correctly. If however, the prover does not actually know the answer, the prover may still get the question correct. However, over many questions the prover is likely to guess incorrectly at some point. Thus after enough guesses it is very unlikely that the prover is faking it. However, we only pose one question on each log in, which is not sufficient for a zero-knowledge interactive proof. Even with a super-aggressive lock out, CrossPassword still has false negative rates are well above the standards for cryptographic algorithms.

We controlled for the wrong thing. The password had a lot of information. However, the trace which we return to the server has very little information. For example, say we take a password and XOR the characters together to get 1 bit which is either yes or no. We transmit very little meaningful information of the password, but that very fact makes it easy for the attacker to guess!

The **Shannon entropy** of CrossPassword is where is the number of characters in the password. For example, a 6 character password has 6 bits of entropy. This makes it easy to brute force. In comparison, a single letter a-z has 26 possibilities or = 4.7 bits *per letter*. Thus our 6 character password is almost the equivalent of a password of a single letter! If we allow upper and lower case, digits, and 10 special characters, we have = 6.2 bits of entropy, which is more than we currently have!

GrIDsure also has a reduction of entropy from choices to choices. However, represents = 13.3 bits of entropy, which is a good deal more than CrossPassword. Remember each additional bit doubles the number of possible passwords, and thus doubles the brute-force password search time.

However, GrIDsure is even unable to fulfill its design goal if an attacker has even two complete observations of grids and the corresponding PIN code. Figuring out the actual password is easier with GrIDsure, but it is harder to brute force. This may be a better tradeoff.

The Modified Scheme ends up being even worse because we reveal information about the password to the user. This ends up being disastrous because an attacker only needs to receive 4-6 copies of the grid from the server, and no data from the user actually entering their password!

On top of all this, our scheme is slower to enter than a traditional password, especially when used with a password manager.

This shows the inherent complexity in producing password schemes. There are many different objectives to try to achieve at once. Trading off some objectives produces different outcomes in security. Objectives cannot be traded off one-for-one, since the factors are not evenly weighted. There are many different possible attacks on a password scheme. It is difficult to keep all of the possible attacks in mind as one designs a particular scheme. Although this scheme was weak, it was interesting to evaluate exactly why it was weak and to think of possible attacks against the scheme.

1. <https://www.grc.com/offthegrid.htm> and associated pages. Is still marked as “Work in Progress;” Retrieved 12/2/2012 [↑](#footnote-ref-2)
2. http://en.wikipedia.org/wiki/Latin\_square [↑](#footnote-ref-3)
3. From R. Jhawar, P. Inglesant, N. Courtois, and M. A. Sasse, “Make mine a quadruple: Strengthening the security of graphical one-time pin authentication,” in Proc. NSS 2011, pp. 81–88 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=06059963 [↑](#footnote-ref-4)
4. https://dazzlepod.com/uniqpass/ [↑](#footnote-ref-5)
5. From R. Jhawar, P. Inglesant, N. Courtois, and M. A. Sasse, “Make mine a quadruple: Strengthening the security of graphical one-time pin authentication,” in Proc. NSS 2011, pp. 81–88 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=06059963 [↑](#footnote-ref-6)
6. Bonneau, Joseph, Cormac Herley, Paul C. van Oorschot, Frank Stajanoy. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. University of Cambridge; Microsoft Research; Carleton University; University of Cambridge. Proc. IEEE Symp. on Security and Privacy 2012 (“Oakland 2012”). http://css.csail.mit.edu/6.858/2012/readings/passwords.pdf [↑](#footnote-ref-7)
7. Material from MIT’s 6.813 User Interface classes by Prof. Rob Miller Spring 2012. [↑](#footnote-ref-8)
8. De Groot, A. D., *Thought and choice in chess*, 1965 [↑](#footnote-ref-9)
9. Paul M. Fitts (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, volume 47, number 6, June 1954, pp. 381–391. [↑](#footnote-ref-10)
10. I. Scott MacKenzie and William A. S. Buxton (1992). Extending Fitts' law to two-dimensional tasks. Proceedings of ACM CHI 1992 Conference on Human Factors in Computing Systems, pp. 219–226. http://doi.acm.org/10.1145/142750.142794 [↑](#footnote-ref-11)
11. Soukoreff, R. William, and I. Scott MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts’ law research in HCI. York University. Department of Computer Science and Engineering. November 4, 2004. http://www.yorku.ca/mack/ijhcs2004.pdf [↑](#footnote-ref-12)