Learning Humanly Usable and Secure Password Schemas

author names withheld

Editor: Under Review for COLT 2016

Abstract

Keywords: List of keywords

1. Introduction

1.1. Human Usability of Password Schemas

1.2. Security Measures for Password Schemas

2. Digit Schema

2.1. Notation

Let $\mathcal{A}$ be the set of alphabet. We assume that $|\mathcal{A}| = N$. For the case of passowrd generation, $\mathcal{A} = \{A, B, ..., Z\}$ and $N = 26$. We denote the set of digits by $\mathcal{D}$, i.e., $\mathcal{D} = \{0, ..., 9\}$. Let's $\mathcal{C}$ denotes the set of possible challenges. We denote the i^{th} coordinate of a vector $\mathbf{u}$ by u_i. Given a positive integer k, we define $[k] = \{0, ..., k-1\}$

2.2. Description

Preprocessing step

- Memorize a a random map $f : \mathcal{A} \to \mathcal{D}$
- Memorize a random string $s = s_1...s_{d-1} \in \mathcal{D}^{d-1}$

Processing Step

Algorithm 1: Digit schema

Input: Challenge $c = c_1...c_l$
$g \overset{10}{\equiv} f(c_1) + ... + f(c_k)$
Output: Response sg

- [S AMIRA]: Should we write the Digit schema in the general format (reading k characters of challenge) here? i think it's better to keep the simple algorithm here and also do the analysis on this algorithm. In the later sections [maybe experiments], we can say that: As

the experiments show, even reading the first few letters of the challenge is enough to get high values of Q, so one can only compute the parity of say the first three letters.

2.3. Human Usability Model

In order to calculate the HUM, we first need to write the steps of Alg. 2.2 in more details

Algorithm 2: Digit schema

Input: Challenge $c = c_1...c_l$
Set $i = 1$, SUM$= 0$
While not EndOfChallenge :
    Compute $f(c_i)$ (Applying the map)
    SUM $\overset{10}{\equiv}$ SUM $+ f(c_i)$ (Add to the running sum)
    $i = i + 1$ (shift pointer)
Print fixed string s
Print SUM

HUM = $2\times$ (initialization) + $l\times$ (while loop condition) + $l\times$ (map) + $l\times$ (add) + $l\times$ (shift pointer) + (end while) + $2\times$ (print) = $4l + 5$

3. Security Analysis

3.1. Best Possible Q

- [SAMIRA]:TODO

3.2. Digit Schema achieves best possible Q for strong linearly independent challenges

We assume that challenges are coming uniformly at random from the set $\mathcal{C}$. Lemma 7 shows that with high probability the first N challenges will be strong linearly independent $(\mathrm{mod}\ 10)$. Therefore, the probability that the adversary guess the correct response for the N^{th} challenge is still $1/10$.

Definition 1 *Given prime numbers p and q, we say that set of challenges $\{c_1, ..., c_n\}$ is strong linearly independent $(\mathrm{mod}\ pq)$, if $\{c_1, ..., c_n\}$ is linearly independent $(\mathrm{mod}\ p)$ and $(\mathrm{mod}\ q)$. Note that a direct consequence of strong linear indepence is linear independence.*

Theorem 2 *Denote the output of OneDigit schema on a challenge c, by $p(c)$. We define $\mathcal{R} = \{p(c) \,|\, c \in \mathcal{C}\}$. For any challenge $c \in \mathcal{C}$ and any response $r \in \mathcal{R}$*

(a) $\Pr[p(c) = r] = \dfrac{1}{10^d}$

Furthermore, assume that we have made k observations $\big(c_1, p(c_1)\big), ..., \big(c_k, p(c_k)\big)$. Then, $\forall g_{k+1} \in \mathcal{D}$ and $\forall c_{k+1} \in \mathcal{C}$ s.t. $\{c_1, ..., c_k, c_{k+1}\}$ is strong linearly independent $(\mathrm{mod}\ 10)$

(b) $\Pr[p(c_{k+1}) = sg_{k+1} \,|\, (p(c_1) = sg_1), ..., (p(c_k) = sg_k)] = 1/10$

Part (a) is saying that without having any prior information, the probability of guessing the correct response to any single challenge is $1/10^d$. In other words, for any two responses r_1 and r_2

$$\Pr[p(c) = r_1] = \Pr[p(c) = r_2]$$

Now assume that the adversary has observed k (challenge, response) pairs and she is trying to guess the response to a new challenge c_{k+1}. After seeing the first (challenge, response) pair, she will know the value of s. So the only unknown part of $p(c_{k+1})$ is the single digit g_{k+1}. Part (b) is saying that for any new challenge c_{k+1} which forms a strong linearly independent set with k previously observed challenges, the adversary can not do better than guessing g_{k+1} completely randomly.

Lemma 3 and lemma 4 provide the main tools in proving Theorem 2.

Lemma 3 *Given a prime number p, matrix $C_{k\times N}$ with k linearly independent rows $\pmod p$, and column vector g, the number of solutions to $Cx \overset{p}{\equiv} g$ is p^{N-k}.*

Proof By assumption, rows of matrix C are linearly independent $\pmod p$, thus there must be k columns $\{C^{j_1}, ..., C^{j_k}\}$ that are linearly independent $\pmod p$. Let's denote the j^{th} coordinate of vector x by x_j. We claim that for any set $\mathcal{X}_{N-k} = \{x_j \in \{0, ..., p-1\} : j \notin \{j_1, ..., j_k\}\}$, there will be a unique set $\mathcal{X}_k = \{x_j \in \{0, ..., p-1\} : j \in \{j_1, ..., j_k\}\}$ such that $x = \mathcal{X}_k \cup \mathcal{X}_{N-k}$ is a solution for system $Cx \overset{p}{\equiv} g$.

Given an arbitrary set $\mathcal{X}_{N-k}$, let's substitute values of x_j for $j \notin \{j_1, ..., j_k\}$ in x and simplify the equation $Cx \overset{p}{\equiv} g$:

$$[C^{j_1}, ..., C^{j_k}] \begin{bmatrix} x_{j_1} \\ \vdots \\ x_{j_k} \end{bmatrix} \overset{p}{\equiv} \begin{bmatrix} g'_1 \\ \vdots \\ g'_k \end{bmatrix}$$

Where $g'_l = g_l - \sum_{i \notin \{j_1,...,j_k\}} C_{li} x_i$ for all $l \in \{1, ..., k\}$.

Since matrix $[C^{j_1}, ..., C^{j_k}]$ is full rank $\pmod p$, the above linear equation has unique solution $x \pmod p$. So far we have proved that for every set $\mathcal{X}_{N-k} = \{x_j \in \{0, ..., p-1\} : j \notin \{j_1, ..., j_k\}\}$, there is a unique set $\mathcal{X}_k = \{x_j \in \{0, ..., p-1\} : j \in \{j_1, ..., j_k\}\}$ such that $x = \mathcal{X}_k \cup \mathcal{X}_{N-k}$ is a solution for system $Cx \overset{p}{\equiv} g$. This concludes that the number of solutions to $Cx \overset{p}{\equiv} g$ is equal to the number of possible sets $\mathcal{X}_{N-k}$ which is equal to p^{N-k}.

■

Lemma 4 *Given prime numbers p and q, matrix $C_{k\times N}$ with k strong linearly independent rows $\pmod{pq}$, and column vector g, the number of solutions to $Cx \overset{pq}{\equiv} g$ is $(pq)^{N-k}$.*

Proof

We want to count the number of solutions to $Cx \overset{pq}{\equiv} g$. Let's denote S_p to be the set of solutions to $Cx \overset{p}{\equiv} g$ and similarly S_q to be the set of solutions to $Cx \overset{q}{\equiv} g$. Using lemma 3, $|S_p| = p^{N-k}$ and

$|S_q| = q^{N-k}$. We claim that for every $x \in S_p$ and $y \in S_q$ there is a unique vector $z \pmod{pq}$ such that z is a solution for $Cz \overset{pq}{\equiv} g$. Furthermore, for every such vector $z \pmod{pq}$, there is a unique pair $x \in S_p$ and $y \in S_q$.

Given $x \in S_p$, and $y \in S_q$, we prove that there exists a k-dimensional vector z such that $z \equiv x \pmod{p}$ and $z \equiv y \pmod{q}$. Consider the following k systems of simultaneous congruences:

$$\begin{array}{ccc} z_1 \overset{p}{\equiv} x_1 & & z_k \overset{p}{\equiv} x_k \\ z_1 \overset{q}{\equiv} y_1 & \cdots & z_k \overset{q}{\equiv} y_k \end{array}$$

Using Chinese Remainder Theorem, there exist $z_1, ..., z_k$ satisfying the above congruences. Therefore

$$\begin{array}{ll} z \overset{p}{\equiv} x & \Rightarrow Cz \overset{p}{\equiv} Cx \overset{p}{\equiv} g \\ & \qquad\qquad \Rightarrow Cz \overset{pq}{\equiv} g \\ z \overset{q}{\equiv} y & \Rightarrow Cz \overset{q}{\equiv} Cx \overset{q}{\equiv} g \end{array}$$

Furthermore, for all $i \in [k]$, $z_i \pmod{pq}$ is unique. Therefore $z = [z_1, ..., z_k]^T$ will be the unique solution to $Cx \overset{pq}{\equiv} g$. So far we have shown that for every pair $(x, y) \in S_p \times S_q$ there is one and only one vector $z \pmod{pq}$ satisfying $Cz \overset{pq}{\equiv} g$. Therefore, the number of solutions to $Cz \overset{pq}{\equiv}$ is equal to $|S_p||S_q| = (pq)^{N-k}$

■

Lemma 5 *(Chinese Remainder Theorem) Suppose $n_1, ..., n_k$ are positive integers that are pairwise coprime. Then, for any given sequence of integers $a_1, ..., a_k$, there exists an integer x solving the following system of simultaneous congruences.*

$$\begin{cases} x \equiv a_1 \pmod{n_1} \\ \quad \vdots \\ x \equiv a_k \pmod{n_k} \end{cases}$$

Furthermore, any two solutions of this system are congruent modulo the product $N = n_1...n_k$. Hence, there is a unique (non-negative) solution less than N.

Now we have all the requirements to prove Theorem 2.

Proof (Theorem 2)

(a) For any $c \in \mathcal{C}, r \in \mathcal{R}$. Let $r = r_1...r_d$

$$\begin{array}{rcl} \Pr[p(c) = r] & = & \Pr[p(c)_1...p(c)_{d-1} = r_1...r_{d-1}] \Pr[p(c)_d = r_d] \\ & = & \Pr[s = r_1...r_{d-1}] \Pr[p(c)_d = r_d] \end{array}$$

Since each digit of string s is chosen independently at random, the above formula is equal to

$$\Pr[s_1 = r_1]... \Pr[s_{d-1} = r_{d-1}] \Pr[r(c)_d = r_d]$$

The first $d-1$ probabilities appearing above are each equal to $1/10$. Thus we only need to compute $\Pr[r(c)_d = r_d] = \Pr[f(c_1) + ... + f(c_l) \equiv r_d \pmod{10}]$. One way to compute this probability is to count the number of maps f that satisfy

$$f(c_1) + ... + f(c_l) \equiv r_d \pmod{10} \tag{1}$$

and divide it by the total number of maps $f : \mathcal{A} \to \mathcal{D}$. What is the number of maps f that satisfy Eq. 2 ? One can choose $f(c_1), ..., f(c_{l-1})$ arbitrarily, then $f(c_l)$ will be chosen uniquely by $f(c_l) \equiv r_d - \sum_{i=1}^{l-1} f(c_i) \pmod{10}$. So the total number of choices of f will be 10^{N-l} for the letters that are not present in c, 10^{l-1} for the first $l-1$ letters in c and 1 for the last letter in c. So the total number of choices is $10^{N-l}10^{l-1} = 10^{N-1}$. Note that the total number of maps $f : \mathcal{A} \to \mathcal{D}$ is 10^N. This leads to

$$\Pr[r(c)_d = r_d] = \Pr[f(c_1) + ... + f(c_l) = r_d] = \frac{10^{N-1}}{10^N} = \frac{1}{10}$$

Consequently, accounting for the fixed string s

$$\Pr[r(c) = r] = \frac{1}{10^{d-1}}\frac{1}{10} = \frac{1}{10^d}$$

(b) Now assume that the adversary have observed k (challenge, response) pairs $\big(c_1, p(c_1) = sg_1\big), ..., \big(c_k, p(c_k) = sg_k\big)$, and we want to compute

$$\Pr[\big(p(c_{k+1}) = sg_{k+1}\big) \mid \big(p(c_1) = sg_1\big), ..., \big(p(c_k) = sg_k\big)]$$

This is equal to

$$\Pr[\big(p(c_{k+1})_d = g_{k+1}\big) \mid \big(p(c_1)_d = g_1\big), ..., \big(p(c_k)_d = g_k\big)]$$

which is equal to

$$\frac{\Pr[\big(p(c_{k+1})_d = g_{k+1}\big), \big(p(c_1)_d = g_1\big), ..., \big(p(c_k)_d = g_k\big)]}{\Pr[\big(p(c_1)_d = g_1\big), ..., \big(p(c_k)_d = g_k\big)} \tag{2}$$

We start by computing the value of denominator. The nominator value can be achieved similarly. In order to compute $\Pr[\big(p(c_1)_d = g_1\big), ..., \big(p(c_k)_d = g_k\big)]$, we should count the number of mappings f that satisfy

$$\begin{cases} f(c_{11}) + ... + f(c_{1l}) \equiv g_1 \pmod{10} \\ \vdots \\ f(c_{k1}) + ... + f(c_{kl}) \equiv g_k \pmod{10} \end{cases} \tag{3}$$

Lemma 4 shows that the number of solutions to above k linear equations is 10^{N-k}. Therefore, the value of the ratio (2) is equal to

$$\frac{10^{n-k+1}}{10^{n-k}} = \frac{1}{10}$$

■

Lemma 6

$Q = N$ for lin. ind. challenges.

- *SAMIRA]: TODO*

Lemma 7 *Given prime numbers p and q and matrix $C_{k\times N}$, such that each row C_i is chosen uniformly at random from $[pq]^N$. With probability higher than ... rows of C are strong linearly independent* $(\text{mod } pq)$.

Proof What are the number of $k \times N$ matrices which has k linearly independent rows $(\text{mod } p)$? There are $p^N - 1$ choices for the first row, as we can chose anything except the all zeros vector. We want the second row to be linearly independent $(\text{mod } p)$ of the first row. Therefore, we can chose any $C_2 \notin \{0, C_1, 2C_1, ..., (p-1)C_1\}$. This eliminates p choices and therefore the number of choices for the second row will $p^N - p$. Similarly, the k^{th} row should not belong to any linear combination of already chosen $k-1$ rows. Hence the number of choices for the k^{th} row will be $p^N - p^{k-1}$. Therefore, the number of matrices of size $k \times N$ and rank k $(\text{mod } p)$ is $\Pi_{i=0}^{k-1}(p^N - p^i)$. Similarly, the number of matrices of size $k \times N$ and rank k $(\text{mod } q)$ is $\Pi_{i=0}^{k-1}(q^N - q^i)$.

Given a matrix $X_{k\times N}$ with rank k $(\text{mod } p)$ and matrix $Y_{k\times N}$ with rank k $(\text{mod } q)$, we define matrix $Z_{k\times N}$ as following.

$$Z_i \equiv qX_i + pY_i \pmod{pq}$$

It is easy to see that rows of Z are linearly independent both $(\text{mod } p)$ and $(\text{mod } q)$ and hence strong linearly independent $(\text{mod } pq)$. Also note that the above transfomation $(X, Y) \rightarrow Z$ is one-to-one i.e., for every two pairs of matrices (X, Y) and (X', Y'),

$$\text{if } \begin{cases} X_i \neq X'_i \pmod{p} \\ \text{or} \\ Y_i \neq Y'_i \pmod{q} \end{cases} \Rightarrow qX_i + pY_i \neq qX'_i + pY'_i \pmod{pq}$$

Therefore, the number of matrices of size $k \times N$ with k strong linearly independent rows $(\text{mod } pq)$ is at least $\Pi_{i=0}^{k-1}(p^N - p^i)\Pi_{i=0}^{k-1}(q^N - q^i)$ Dividing it by the total number of $k \times N$ matrices $(\text{mod } pq)$ we get that the probability that a random matrix of size $k \times N$ has k strong linearly independent rows $(\text{mod } pq)$ is bigger than

$$\frac{\Pi_{i=0}^{k-1}(p^N - p^i)\Pi_{i=0}^{k-1}(q^N - q^i)}{(pq)^{kN}}$$

- [TASK] Find a concrete lower bound for the above probability

■

3.3. Experimental evaluation of Q for websites, English words and random strings.

Theorem. 2 is saying that as long as a new challenge forms a strong linearly independent set with already observed challenges, the adversary can not predict the response to this new challenge. In order to compute the value of Q for OneDigit schema we should answer the following question:

- ◇ Given a set of challenges $\mathcal{C}$, at each round, a new challenge $c \in \mathcal{C}$ is chosen uniformly at random. Let $\mathcal{C}_i$ be the set of challenges chosen till round i. What is the maximum i such that $\mathcal{C}_i$ is a strong linearly independent set?

To answer the above question, we ran the following experiment. We chose the challenge set $\mathcal{C}$ to be the set of all the valid website names. At each iteration, our program choses a challenge c uniformly at random from $\mathcal{C}$ and checks if c along with challenges chosen so far, forms a strong linearly independent set. If yes, it saves the number of iterations as the Q value. Otherwise, it will continue. We repeated this procedure for 1000 times and took the average of all the saved Q values. The following table shows the result:

Number of trials	Q
1000	~ 18

4. Conclusion and Questions

References