Towards Human Computable Passwords[*]

Jeremiah Blocki[1], Manuel Blum[2], Anupam Datta[3], and Santosh Vempala[4]

1 **Purdue University, West Lafayette, USA**
`jblocki@purdue.edu`

2 **Carnegie Mellon University, Pittsburgh, USA**
`mblum@cs.cmu.edu`

3 **Carnegie Mellon University, Pittsburgh, USA**
`danupam@cmu.edu`

4 **Georgia Tech, Altanta, USA**
`vempala@cc.gatech.edu`

Abstract

An interesting challenge for the cryptography community is to design authentication protocols that are so simple that a human can execute them without relying on a fully trusted computer. We propose several candidate authentication protocols for a setting in which the human user can only receive assistance from a semi-trusted computer – a computer that stores information and performs computations correctly but does not provide confidentiality. Our schemes use a semi-trusted computer to store and display public challenges $C_i \in [n]^k$. The human user memorizes a random secret mapping $\sigma : [n] \to \mathbb{Z}_d$ and authenticates by computing responses $f(\sigma(C_i))$ to a sequence of public challenges where $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ is a function that is easy for the human to evaluate. We prove that any statistical adversary needs to sample $m = \tilde{\Omega}\left(n^{s(f)}\right)$ challenge-response pairs to recover σ, for a security parameter $s(f)$ that depends on two key properties of f. Our lower bound generalizes recent results of Feldman et al. [26] who proved analogous results for the special case $d = 2$. To obtain our results, we apply the general hypercontractivity theorem [45] to lower bound the *statistical dimension* of the distribution over challenge-response pairs induced by f and σ. Our *statistical dimension* lower bounds apply to arbitrary functions $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ (not just to functions that are easy for a human to evaluate). As an application, we propose a family of human computable password functions f_{k_1,k_2} in which the user needs to perform $2k_1 + 2k_2 + 1$ primitive operations (e.g., adding two digits or remembering a secret value $\sigma(i)$), and we show that $s(f) = \min\{k_1 + 1, (k_2 + 1)/2\}$. For these schemes, we prove that forging passwords is equivalent to recovering the secret mapping. Thus, our human computable password schemes can maintain strong security guarantees even after an adversary has observed the user login to many different accounts.

1998 ACM Subject Classification D.4.6 Authentication

Keywords and phrases Passwords, Cognitive Authentication, Human Computation, Planted Constraint Satisfaction Problem, Statistical Dimension

Digital Object Identifier 10.4230/LIPIcs.ITCS.2017.10

[*] This work was partially supported by the NSF Science and Technology TRUST and the AFOSR MURI on Science of Cybersecurity. This work was completed in part while the first author was at Carnegie Mellon University where the first author partially supported by an NSF Graduate Fellowship.

1 Introduction

A typical computer user has many different online accounts which require some form of authentication. While passwords are still the dominant form of authentication, users struggle to remember their passwords. As a result users often adopt insecure password practices (e.g., reuse, weak passwords) [28, 21, 39, 18] or end up having to frequently reset their passwords. Recent large-scale password breaches highlight the importance of this problem [1, 21, 11, 2, 50, 3, 4, 5, 6, 7, 8, 9]. An important research goal is to develop usable and secure password management scheme – a systematic strategy to help users create and remember multiple passwords. Blocki et al. [13] and Blum and Vempala [17] recently proposed password management schemes that maintain some security guarantees after a small constant number of breaches (e.g., an adversary who sees three of the user's passwords still has some uncertainty about the user's remaining passwords).

In this work we focus on the goal of developing human computable password management schemes in which security guarantees are strongly maintained after *many* breaches (e.g., an adversary who sees one-hundred of the user's passwords still has high uncertainty about the user's remaining passwords). In a human computable password management scheme the user reconstructs each of his passwords by *computing* the response to a public challenge.

Our human computable password schemes admittedly require more human effort than the password management schemes of Blocki et al. [13] and Blum and Vempala [17], and, unlike Blocki et al. [13], our scheme requires users to do simple mental arithmetic (e.g., add two single-digit numbers) in their head. However, our proposed schemes are still human usable in the sense that a motivated, security-conscious user would be able to learn to use the scheme and memorize all associated secrets in a few hours. In particular, the human computation in our schemes only involves a few very simple operations (e.g., addition modulo 10) over secret values (digits) that the user has memorized. More specifically, in our candidate human computable password schemes the user learns to compute a simple function $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$,[1] and memorizes a secret mapping $\sigma : [n] \to \mathbb{Z}_d$. The user authenticates by responding to a sequence of single digit challenges, i.e., a challenge-response pair $(C, f(\sigma(C)))$ is a challenge $C \in X_k \subseteq [n]^k$ and the corresponding response.

One of our candidate human computable password schemes involves the function

$$f(x_0, x_1, x_2, x_3, x_4, x_5, \ldots, x_{13}) = x_{13} + x_{12} + x_{(x_{10}+x_{11} \mod 10)} \mod 10 .$$

If the user memorizes a secret mapping σ from n images to digits then each challenge $C = (I_0, \ldots, I_{13})$ would correspond to an ordered subset of 14 of these images and the response to the challenge is $f\big(\sigma(I_0), \ldots, \sigma(I_{13})\big)$. We observe that a human would only need to perform three addition operations modulo 10 to evaluate this function. The user would respond by (1) adding the secret digits associated with challenge images I_{10} and I_{11} to get a secret index $0 \le i \le 9$, (2) finding image I_i, (3) adding the secret digits associated with images I_i, I_{12} and I_{13} to produce the final response. To amplify security the user may respond to $\lambda \ge 1$ single-digit challenges $C_1, \ldots, C_\lambda$ to obtain a λ digit password $f(\sigma(C_1)), \ldots, f(\sigma(C_\lambda))$. We note that the challenge C does not need to be kept secret and thus the images can be arranged on the screen in helpful manner for the human user – see Figure 1 for an example and see Appendix A for more discussion of the user interface.

[1] In our security analysis we consider arbitrary bases d. However, our specific schemes use the base $d = 10$ that is most familiar to human users.

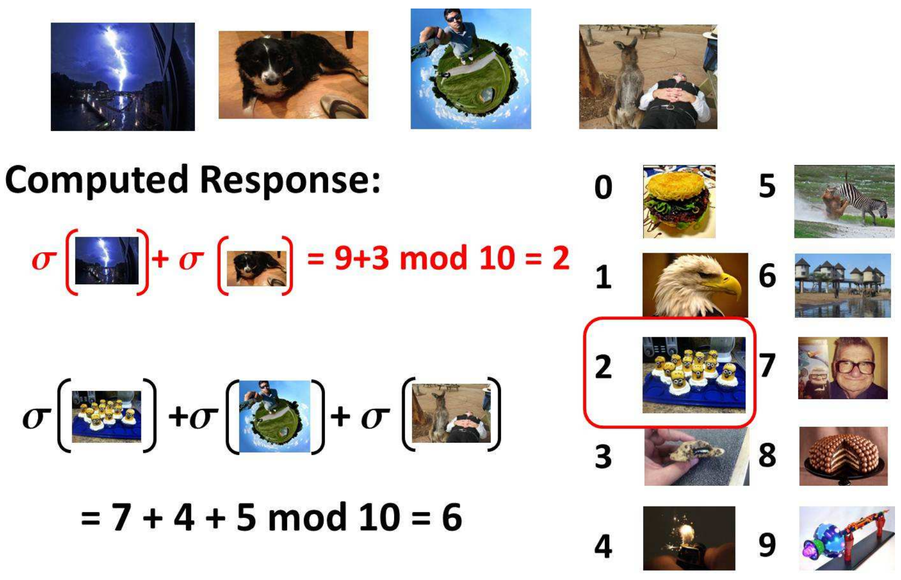


Figure 1 Computing the response $f(\sigma(C)) = 6$ to a single digit challenge C.

We present a natural conjecture which implies that a polynomial time attacker will need to see the responses to $\tilde{\Omega}\left(n^{s(f)}\right)$ random challenges before he can forge the user's passwords (accurately predict the responses to randomly selected challenges)[2]. Here, $s(f)$ is a security parameter that depends on two key properties of the function f (in our above example $s(f) = 3/2$). Furthermore, we provide strong evidence for our conjecture by ruling out a broad class of algorithmic techniques that the adversary might use to attack our scheme.

Following Blocki et al. [13] we consider a setting where a user has two types of memory: *persistent memory* (e.g., a sticky note or a text file on his computer) and *associative memory* (e.g., his own human memory). We assume that persistent memory is reliable and convenient but not private (i.e., an adversary can view all challenges stored in persistent memory, but he cannot tamper with them). In contrast, a user's associative memory is private but lossy – if the user does not rehearse a memory it may be forgotten. Thus, the user can store a password challenge $C \in X_k$ in persistent memory, but the mapping σ must be stored in associative memory (e.g., memorized and rehearsed). We allow the user to receive assistance from a semi-trusted computer. A semi-trusted computer will perform computations accurately (e.g., it can be trusted to show the user the correct challenge), but it will not ensure privacy of its inputs or outputs. This means that a human computable password management scheme should be based on a function f that the user can compute entirely in his head.

Contributions

We provide precise notions of security and usability for a human computable password management scheme (Section 2). We introduce the notion of UF-RCA security (Unforgeability under Random Challenge Attacks). Informally, a human computable password scheme is

[2] We stress that, unlike [13, 17], our security guarantees are not information theoretic. In fact, a computationally unbounded adversary would need to see at most $O(n)$ challenge-response pairs to break the human computable password management scheme.

UF-RCA secure if an adversary cannot forge passwords after seeing many example challenge-response pairs.

We present the design of a candidate family of human computable password management schemes f_{k_1,k_2}, and analyze the usability and security of these schemes (Section 3). Our usability analysis indicates that to compute $f_{k_1,k_2}\left(\sigma\left(C\right)\right)$ the user needs to execute $2k_1{+}2k_2{+}1$ simple operations (e.g., addition of single digits modulo 10). The main technical result of this section (Theorem 10) states that our scheme is UF-RCA secure given a plausible conjecture about the hardness of random planted constraint satisfiability problems (*RP-CSP*). Our conjecture is that any polynomial time adversary needs to see at least $m = n^{\min\{r(f)/2, g(f)+1-\epsilon\}}$ challenge-response pairs $(C, f\left(\sigma\left(C\right)\right))$ to recover the secret mapping σ. Here, $s(f) = \min\{r(f)/2, g(f)+1\}$ is a composite security parameter involving $g(f)$ (how many inputs to f need to be fixed to make f linear?) and $r(f)$ (what is the largest value of r such that the distribution over challenge-response pairs are $(r-1)$-wise independent?). We prove that $g\left(f_{k_1,k_2}\right) = k_1$ and $r\left(f_{k_1,k_2}\right) = (k_2+1)$.

Next we prove that any statistical adversary needs at least $\tilde{\Omega}\left(n^{r(f)/2}\right)$ challenge-response pairs $(C, f\left(\sigma\left(C\right)\right))$ to find a secret mapping σ' that is ϵ-correlated with σ (Section 4). This result may be interpreted as strong evidence in favor of the *RP-CSP* hardness assumption as most natural algorithmic techniques have statistical analogues (see discussion in Section 4). While Gaussian Elimination is a notable exception, our composite security parameter accounts for attacks based on Gaussian Elimination – an adversary needs to see $m = \tilde{\Omega}\left(n^{1+g(f)}\right)$ challenge-response pairs to recover σ using Gaussian Elimination. Moving beyond asymptotic analysis we also provide empirical evidence that our human computable password management scheme is hard to crack. In particular, we used a CSP solver to try to recover $\sigma \in \mathbb{Z}_{10}^n$ given m challenge-response pairs using the functions $f_{1,3}$ and $f_{2,2}$. Our CSP solver failed to find the secret mapping $\sigma \in \mathbb{Z}_{10}^{50}$ given $m = 1000$ random challenge-response pairs with both functions $f_{1,3}$ and $f_{2,2}$. Additionally, we constructed public challenges for cryptographers to break our human computable password management schemes under various parameters (e.g., $n = 100$, $m = 1000$).

Our lower bound for statistical adversaries is based on the *statistical dimension* of the distribution over challenge-response pairs induced by f and σ. We stress that our analysis of the statistical dimension applies to arbitrary functions $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$, not just to functions that are easy for humans to compute. Our analysis of the statistical dimension generalizes recent results of Feldman et al. [26] for binary predicates and may be of independent interest. While the analysis is similar at a high level, we stress that our proofs do require some new ideas. Because our function f is not a binary predicate we cannot use the Walsh basis functions to express the Fourier decomposition of f and analyze the statistical dimension of our distribution over challenge-response pairs as Feldman et al. [26] do. Instead, we use a generalized set of Fourier basis functions to take the Fourier decomposition of f, and we apply the general hypercontractivity theorem [45] to obtain our bounds on the statistical dimension.

We complete the proof of Theorem 10 in Section 5 by proving that forging passwords and approximately recovering the secret mapping are equivalent problems for a broad class of human computable password schemes, including our candidate family f_{k_1,k_2}. This result implies that any adversary who can predict the response $f(C)$ to a random challenge C with better accuracy than random guessing can be used as a blackbox to approximately recover the secret mapping.

2 Definitions

2.1 Notation

Given two strings $\alpha_1, \alpha_2 \in \mathbb{Z}_d^n$ we use $H(\alpha_1, \alpha_2) \doteq |\{i \in [n] \mid \alpha_1[i] \neq \alpha_2[i]\}|$ to denote the Hamming distance between them. We will also use $H(\alpha_1) \doteq H\left(\alpha_1, \vec{0}\right)$ to denote the Hamming weight of α_1. We use $\sigma : [n] \to \mathbb{Z}_d$ to denote a secret random mapping that the user will memorize. We will sometimes abuse notation and think of $\sigma \in \mathbb{Z}_d^n$ as a string which encodes the mapping, and we will use $\sigma \sim \mathbb{Z}_d^n$ to denote a random mapping chosen from $\mathbb{Z}_d^n$ uniformly at random. Given a distribution $\mathcal{D}$ we will use $x \sim \mathcal{D}$ to denote a random sample from this distribution. We also use $x \sim S$ to denote an element chosen uniformly at random from a finite set S.

▶ **Definition 1.** We say that two mappings $\sigma_1, \sigma_2 \in \mathbb{Z}_d^n$ are ϵ-correlated if $\frac{H(\sigma_1,\sigma_2)}{n} \leq \frac{d-1}{d} - \epsilon$, and we say that a mapping $\sigma \in \mathbb{Z}_d^n$ is δ-balanced if $\max_{i\in\{0,\ldots,d-1\}} \left| \frac{H(\sigma,\vec{i})}{n} - \frac{d-1}{d} \right| \leq \delta$.

Note that for a random mapping σ_2 we expect σ_1 and σ_2 to differ at $\mathbb{E}_{\sigma_2 \sim \mathbb{Z}_d^n}[H(\sigma_1, \sigma_2)] = n\left(\frac{d-1}{d}\right)$ locations, and for a random mapping σ and $i \sim \{0, \ldots, d-1\}$ we expect σ to differ from $\vec{i}$ at $\mathbb{E}_{i\sim\mathbb{Z}_d, \sigma\sim\mathbb{Z}_d^n}\left[H\left(\sigma, \vec{i}\right)\right] = n\left(\frac{d-1}{d}\right)$ locations. Thus, with probability $1 - o(1)$ a random mapping σ_2 will not be ϵ-correlated with σ_1, but a random mapping σ will be δ-balanced with probability $1 - o(1)$.

We let $X_k \subseteq [n]^k$ denote the space of ordered clauses of k variables without repetition. We use $C \sim X_k$ to denote a clause C chosen uniformly at random from X_k and we use $\sigma(C) \in \mathbb{Z}_d^k$ to denote the values of the corresponding variables in C. For example, if $d = 10$, $C = (3, 10, 59)$ and $\sigma(i) = (i + 1 \mod 10)$ then $\sigma(C) = (4, 1, 0)$.

We view each clause $C \in X_k$ as a *single-digit challenge*. The user responds to a challenge C by computing $f(\sigma(C))$, where $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ is a *human computable* function (see discussion below) and $\sigma : [n] \to \mathbb{Z}_d$ is the secret mapping that the user has memorized. For example, if $d = 10$, $C = (3, 10, 59)$, $\sigma(i) = (i + 1 \mod 10)$ and $f(x, y, z) = (x - y + z \mod 10)$ then $f(\sigma(C)) = (4 - 1 + 0 \mod 10) = 3$. A length-$\lambda$ password challenge $\vec{C} = \langle C_1, \ldots, C_\lambda \rangle \in (X_k)^\lambda$ is a sequence of t single digit challenges, and $f(\sigma(\vec{c})) = \langle f(\sigma(C_1)), \ldots, f(\sigma(C_\lambda)) \rangle \in \mathbb{Z}_d^\lambda$ denotes the corresponding response (e.g., a password).

Let's suppose that the user has m accounts $A_1, \ldots, A_m$. In a human computable password management scheme we will generate m length-λ password challenges $\vec{C}_1, \ldots, \vec{C}_m \in (X_k)^\lambda$. These challenges will be stored in persistent memory so they are always accessible to the user as well as the adversary. When our user needs to authenticate to account A_i he will be shown the length-t password challenge $\vec{C}_i = \langle C_1^i, \ldots, C_\lambda^i \rangle$. The user will respond by computing his password $p_i = \left\langle f\left(\sigma\left(C_1^i\right)\right), \ldots, f\left(\sigma\left(C_\lambda^i\right)\right) \right\rangle \in \mathbb{Z}_d^\lambda$.

2.2 Requirements for a Human Computable Function

In our setting we require that the composite function $f \circ \sigma : X_k \to \mathbb{Z}_d$ is human computable. Informally, we say that a function f is *human-computable* if a human user can evaluate f *quickly* in his head.

▶ **Requirement 2.** *A function f is $\hat{t}$-*human computable *for a human user H if H can reliably evaluate f in his head in $\hat{t}$ seconds.*

We argue that a function f will be *human-computable* whenever there is a fast streaming algorithm [10] to compute f using only very simple primitive operations. A streaming algorithm is an algorithm for processing a data stream in which the input (e.g., the challenge C) is presented as a sequence of items that can only be examined once. In our context the streaming algorithm must have a very low memory footprint because a typical person can only keep 7 ± 2 'chunks' of information in working memory [41] at any given time. Our streaming algorithm can only involve primitive operations that a person could execute quickly in his head (e.g., adding two digits modulo 10, recalling a value $\sigma(i)$ from memory).

▶ **Definition 3.** Let P be a set of primitive operations. We say that a function f is $(P, \tilde{t}, \hat{m})$-computable if there is a space $\hat{m}$ streaming algorithm $\mathcal{A}$ to compute f using only $\tilde{t}$ operations from P.

In this paper we consider the following primitive operations P: **Add**, **Recall** and **TableLookup**. **Add** : $\mathbb{Z}_{10} \times \mathbb{Z}_{10} \to \mathbb{Z}_{10}$ takes two digits x_1 and x_2 and returns $x_1 + x_2$ mod 10. **Recall** : $[n] \to \mathbb{Z}_{10}$ takes an index i and returns the secret value $\sigma(i)$ that the user has memorized. **TableLookup** : $\mathbb{Z}_{10} \times [n]^{10} \to [n]$ takes a digit x_1 and finds the x_1'th value from a table of 10 indices. We take the view that no human computable function should require users to store intermediate values in long-term memory because the memorization process would necessarily slow down computation. Therefore, we restrict our attention to space $\hat{m}$ streaming algorithms and do not include any primitive operation like **MemorizeValue**.

▶ **Example.** The function $f \circ \sigma(i_1, \ldots, i_5) = \sigma(i_1) + \ldots + \sigma(i_5)$ requires 9 primitive operations (five **Recall** operations and four **Add** operations) and requires space $\hat{m} = 3$ (e.g., we need one slot to store the current total, one slot to store the next value from the data stream and one free slot to execute a primitive operation).

Similar primitive operations have been studied by cognitive physchologists (e.g., [51]). The time γ_H it takes a human user H to execute one primitive operation will typically improve with practice (e.g., [33]). We note that we allow this computation speed constant γ_H to vary from user to user in the same way that two computers might operate at slightly different speeds. We conjecture that, after training, a human user H with a moderate mathematical background will be able to evaluate a $(P, \tilde{t}, 3)$-computable function in $\hat{t} \leq \tilde{t}$ seconds – the first author of this paper found that (after some practice) he could evaluate $(P, 9, 3)$-computable functions in 7.5-seconds ($\gamma_H \leq 1$).

▶ **Conjecture 4.** *Let $P = \{\textbf{Add}, \textbf{Recall}, \textbf{TableLookup}\}$. For each human user H there is a small constant $\gamma_H > 0$ such that any $(P, \tilde{t}, 3)$-computable function f will be $\hat{t}$-human computable for H with $\hat{t} = \gamma_H \tilde{t}$.*

2.3 Password Unforgeability

In the password forgeability game the adversary attempts to guess the user's password for a randomly selected account after he has seen the user's passwords at m/λ other randomly selected accounts. We say that a scheme is UF-RCA (Unforgeability against Random Challenge Attacks) secure if any probabilistic polynomial time adversary fails to guess the user's password with high probability. In the password forgeability game we select the secret mapping $\sigma : [n] \to \mathbb{Z}_d$ uniformly at random along with challenges $C_1, \ldots, C_{m+\lambda} \sim X_k$. The adversary is given the function $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ and is shown the challenges $C_1, \ldots, C_{m+\lambda}$ as well as the values $f(\sigma(C_i))$ for $i \in \{1, \ldots, m\}$. The game ends when the adversary $\mathcal{A}$ outputs a guess $\langle q_1, \ldots, q_\lambda \rangle \in \mathbb{Z}_d^\lambda$ for the value of $\langle f(\sigma(C_{m+1})), \ldots, f(\sigma(C_{m+\lambda})) \rangle$. We say that the adversary wins if he correctly guesses the responses to all of the challenges $C_{m+1}, \ldots, C_{m+\lambda}$, and we use $\textbf{Wins}(\mathcal{A}, n, m, \lambda)$ to denote the event that the adversary wins the game (e.g.,

$\forall i \in \{1,\ldots,\lambda\}.q_i = f\left(\sigma\left(C_{m+i}\right)\right)$). We are interested in understanding how many example single digit challenge-response pairs the adversary needs to see before he can start breaking the user's passwords.

▶ **Definition 5** (Security). We say that a function $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ is $\mathbf{UF-RCA}\left(n,m,\lambda,\delta\right)-$ *secure* if for every probabilistic polynomial time (in n,m) adversary $\mathcal{A}$ we have

$$\Pr\left[\mathbf{Wins}\left(\mathcal{A},n,m,\lambda\right)\right] \le \delta \ ,$$

where the randomness is taken over the selection of the secret mapping $\sigma \sim \mathbb{Z}_d^n$, the challenges $C_1,\ldots,C_{m+\lambda}$ as well as the adversary's coins.

Discussion

Our security model is different from the security model of Blocki et al. [13] in which the adversary gets to adaptively select which accounts to compromise and which account to attack. While our security model may seem weaker at first glance because the adversary does not get to select which account to compromise/attack, we observe that the password management schemes of Blocki et al. [13] are only secure against one to three adaptive breaches. By contrast, our goal is to design human computable password schemes that satisfy $\mathbf{UF-RCA}$ security for large values of m (e.g. 1000), which means that it is reasonable to believe that the user has at most m/λ password protected accounts. If the user has at most m/λ accounts then union bounds imply that an adaptive adversary – who gets to compromise all but one account – will not be able to forge the password at any remaining account with probability greater than $m\delta/\lambda$ (typically, $m \ll \lambda/\delta$)[3].

2.4 Security Parameters of f

Given a function $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ we define the function $Q^f : \mathbb{Z}_d^{k+1} \to \{\pm 1\}$ s.t. $Q^f\left(x,i\right) = 1$ if $f(x) = i$; otherwise $Q^f\left(x,i\right) = -1$. We use Q_σ^f to define a distribution over $X_k \times \mathbb{Z}_d$ (challenge-response pairs) as follows: $\Pr_{Q_\sigma^f}\left[C,i\right] \doteq \frac{Q^f\left(\sigma(C),i\right)+1}{2|X_k|}$. Intuitively, Q_σ^f is the uniform distribution over challenge response pairs (C,j) s.t. $f\left(\sigma\left(C\right)\right) = j$. We also use $Q^{f,j} : \mathbb{Z}_d^k \to \{\pm 1\}$ $\left(Q^{f,j}\left(x\right) = Q^f\left(x,j\right)\right)$ to define a distribution over X_k. $\Pr_{Q_\sigma^{f,j}}[C] = \frac{Q^{f,j}\left(f(\sigma(C))\right)+1}{2|\{C' \in X_k : f(\sigma(C'))=j|} = \Pr_{Q_\sigma^f}\left[(C,i) \mid i = j\right]$. We write the Fourier decomposition of a function $Q : \mathbb{Z}_d^k \to \{\pm 1\}$ as follows

$$Q(x) = \sum_{\alpha \in \mathbb{Z}_d^k} \hat{Q}_\alpha \cdot \chi_\alpha\left(x\right) \ \text{, where the basis functions are} \quad \chi_\alpha\left(x\right) \doteq \exp\left(\frac{-2\pi\sqrt{-1}\left(x\cdot\alpha\right)}{d}\right) .$$

We say that a function Q has degree ℓ if $\ell = \max\left\{H\left(\alpha\right) \,\middle|\, \alpha \in \mathbb{Z}_d^k \wedge \hat{Q}_\alpha \neq 0\right\}$ – equivalently if $Q(x) = \sum_i Q_i(x)$ can be expressed as a sum of functions where each function $Q_i : \mathbb{Z}_d^k \to \mathbb{R}$ depends on at most ℓ variables.

▶ **Definition 6.** We use $r(Q) \doteq \min\left\{H\left(\alpha\right) \,\middle|\, \exists \alpha \in \mathbb{Z}_d^k . \hat{Q}_\alpha \neq 0 \wedge \alpha \neq \vec{0}\right\}$ to denote the distributional complexity of Q, and we use $r(f) = \min\left\{r\left(Q^{f,j}\right) \mid j \in \mathbb{Z}_d\right\}$ to denote the distributional complexity of f. We use $g(f) \doteq$

$$\min\left\{\ell \in \mathbb{N}\cup\{0\} \,\middle|\, \exists \alpha \in \mathbb{Z}_d^\ell, S \subseteq [k], \hat{d} \in \mathbb{Z}_d . \text{s.t } |S| = \ell \ \& \ f_{|S,\alpha} \text{ is a linear function mod } \hat{d}\right\} ,$$

[3] We assume in our analysis that the adversary does not get to pick the challenges C that the user will solve.

to denote the minimum number of variables that must be fixed to make f a linear function. Here, $f_{|S,\alpha} : \mathbb{Z}_d^{k-\ell} \to \mathbb{Z}_d$ denotes the function f after fixing the variables at the indices specified by S to α. Finally, we use $s(f) \doteq \min\{r(f)/2, g(f)+1\}$ as our composite security measure.

We conjecture that a polynomial time adversary will need to see $m = n^{s(f)}$ challenge-response pairs before he can approximately recover the secret mapping σ. We call this conjecture about the hardness of random planted constraint satisfiability problems RP-CSP (Conjecture 7). In support of RP-CSP we prove that any statistical algorithm needs to see at least $m = \tilde{\Omega}\left(n^{r(f)/2}\right)$ challenge response pairs to (approximately) recover the secret mapping σ and we observe that a polynomial time adversary would need to see $m = O\left(n^{g(f)+1}\right)$ challenge-response pairs to recover σ using Gaussian Elimination. In Section 5 we show that the human computable password scheme will be UF-RCA secure provided that RP-CSP holds and that f satisfies a few moderate properties (e.g., the output of f is evenly distributed).

▶ **Conjecture 7** (RP-CSP)**.** *For every probabilistic polynomial time adversary $\mathcal{A}$ and every $\epsilon, \epsilon' > 0$ there is an integer N s.t. for all $n > N$, $m \leq n^{\min\{r(f)/2, g(f)+1-\epsilon'\}}$ we have* $\Pr\left[\textbf{Success}\left(\mathcal{A}, n, m, \epsilon\right)\right] \leq \mu(n)$*, where* $\textbf{Success}\left(\mathcal{A}, n, m, \epsilon\right)$ *denotes the event that $\mathcal{A}$ finds a mapping σ' that is ϵ-correlated with σ given m randomly selected challenge response pairs $\left(C_1, f\left(\sigma\left(C_1\right)\right)\right), \ldots, \left(C_m, f\left(\sigma\left(C_m\right)\right)\right)$ and $\mu(n)$ is a negligible function. The probability is over the selection of the random mapping σ, the challenges $C_1, \ldots, C_m$ and the random coins of the adversary.*

3 Candidate Secure Human Computable Functions

In this section we present a family of candidate human computable functions. We consider the usability of these human computable password schemes in Section 3.1, and we analyze the security of our schemes in Section 3.2.

We first introduce our family of candidate human computable functions (for all of our candidate human computable functions $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ we fix $d = 10$ because most humans are used to performing arithmetic operations on digits). Given integers $k_1 > 0$ and $k_2 > 0$ we define the function $f_{k_1,k_2} : \mathbb{Z}_{10}^{10+k_1+k_2}$ as follows

$$f_{k_1,k_2}\left(x_0, \ldots, x_{9+k_1+k_2}\right) = x_j + \sum_{i=10+k_1}^{9+k_1+k_2} x_i \mod 10, \quad \text{where } j = \left(\sum_{i=10}^{9+k_1} x_i\right) \mod 10 .$$

Authentication Process

We briefly overview the authentication process – see Algorithms 2 and 3 in Appendix A for a more formal presentation of the authentication process. We assume that the mapping $\sigma : \{1, ..., n\} \to \mathbb{Z}_{10}$ is generated by the user's local computer in secret. The user may be shown mnemonic helpers (see discussion below) to help memorize σ, but these mnemonic helpers are discarded immediately afterward. After the user has memorized σ he can create a password pw_i for an account A_i as follows: the user's local computer generates λ random single-digit challenges $C_1^i, \ldots, C_\lambda^i \in X_\lambda$ and the user computes $pw_i = f\left(\sigma\left(C_1\right)\right), \ldots, f\left(\sigma\left(C_\lambda\right)\right)$. The authentication server for account A_i stores the cryptographic hash of pw_i, while the challenges $C_1^i, \ldots, C_\lambda^i \in X_\lambda$ are stored in public memory (e.g., on the user's local computer), which means that they can be viewed by the adversary as well as the legitimate user. To authenticate the user retrieves the public challenges $C_1^i, \ldots, C_\lambda^i$ for account A_i and computes pw_i. The server for A_i verifies that the cryptographic hash of pw_i matches its records. To protect users from offline attacks in the event of a server breach, the password pw_i should be stored using a slow cryptographic hash function $\mathbf{H}$ like BCRYPT [47].

3.1 Usability

In our discussion of usability we focus on the time it would take a human user to compute a password once he has memorized the secret mapping σ. Other important considerations include the challenge of memorizing and rehearsing the secret mapping σ to ensure that the user remembers the secret mapping σ over time.

3.1.1 Computation Time

Given a challenge $C = (c_0, \ldots, c_{9+k_1+k_2}) \in X_{10+k_1+k_2}$ we can compute $f_{k_1,k_2}(\sigma(C))$ we compute $j = \sum_{i=10}^{9+k_1} \sigma(c_i) \mod 10$ using $k_1 - 1$ **Add** operations and k_1 **Recall** operations. We then execute **TableLookup** $(j, c_0, \ldots, c_9)$ to obtain c_j. Now we need k_2 **Add** operations and $k_2 + 1$ **Recall** operations to compute the final response $\sigma(c_j) + \sigma(c_{10+k_1}) + \ldots + \sigma(c_{9+k_1+k_2})$.

▶ **Fact 8.** *Let* $P = \{\textbf{Add}, \textbf{Recall}, \textbf{TableLookup}\}$ *then* $f_{k_1,k_2} \circ \sigma$ *is* $(P, 2k_1 + 2k_2 + 1, 3)$-*computable.*

Fact 8 and Conjecture 4 would imply that $f_{1,3}$ and $f_{2,2}$ are $\hat{t}$-human computable with $\hat{t} = 9$ seconds for humans H with computation constant $\gamma_H \leq 1$. The functions $f_{1,3}$ and $f_{2,2}$ were both $\hat{t}$-human computable with $\hat{t} = 7.5$ seconds for the main author of this paper. While the value of γ_H might be larger for many human users who are less comfortable with mental arithmentic, we note we may have $\gamma_H \ll 1$ for many human users after training (e.g., see `https://youtu.be/_-2L6ZxFacg` for a particularly impressive demonstration of mental arithmetic by young children.).

3.1.2 Memorizing and Rehearsing σ

Memorizing the secret mapping might be the most difficult part of our schemes. In practice, we envision that the user memorizes a mapping from n objects (e.g., images) to digits. For example, if $n = 26$ and $d = 10$ then the user might memorize a random mapping from characters to digits. The first author of this paper was able to memorize a mapping from $n = 100$ images to digits in about 2 hours. We conjecture that the process could be further expedited using mnemonic helpers – see discussion in the appendix.

After the user memorizes σ he may need to rehearse parts of the mapping periodically to ensure that he does not forget it. One of the benefits of our human computable password schemes is that the user will get lots of practice rehearsing the secret mapping each time he computes a password. In fact users who authenticate frequently enough will not need to spend any extra time rehearsing the secret mapping as they will get sufficient natural practice to remember σ.

3.2 Security Analysis

Claim 9 demonstrates that $s(f_{k_1,k_2}) = \min\{(k_2+1)/2, k_1+1\}$. Intuitively, the security of our human computable password management scheme will increase with k_1 and k_2. However, the work that the user needs to do to respond to each single-digit challenge is proportional to $2k_1 + 2k_2 + 1$ (See Fact 8).

▶ **Claim 9.** *Let* $0 \leq k_1$ *and* $k_2 > 0$ *be given and let* $f = f_{k_1,k_2}$ *we have* $g(f) = \min\{k_1, 10\}$, $r(f) = k_2 + 1$ *and* $s(f) = \min\left\{\frac{k_2+1}{2}, k_1 + 1, 11\right\}$.

An intuitive way to see that $r\left(f_{k_1,k_2}\right) > k_2$ is to observe that we cannot bias the output of f_{k_1,k_2} by fixing k_2 variables. Fix the value of *any* k_2 variables and draw the values for the other $k_1 + 10$ variables uniformly at random from $\mathbb{Z}_{10}$. One of the $k_2 + 1$ variables in the sum $x_j + \sum_{i=10+k_1}^{9+k_1+k_2} x_i \mod 10$ will not be fixed. Thus, the probability that the final output of $f_{k_1,k_2}\left(x_0, \ldots, x_{9+k_1+k_2}\right)$ will be r is exactly $1/10$ for each digit $r \in \mathbb{Z}_{10}$. Similarly, an intuitive way to see that $r\left(f_{k_1,k_2}\right) \leq k_2 + 1$ is to observe that we can bias the value of $f_{k_1,k_2}\left(x_0, \ldots, x_{9+k_1+k_2}\right)$ by fixing the value of $k_2 + 1$ variables. In particular if we fix the variables $x_0, x_{10+k_1}, \ldots, x_{9+k_1+k_2}$ so that $0 = x_0 + \sum_{i=10+k_1}^{9+k_1+k_2} x_i \mod 10$ then the output of $f_{k_1,k_2}\left(x_0, \ldots, x_{9+k_1+k_2}\right)$ is more likely to be 0 than any other digit. The full proof of Claim 9 can be found in Appendix F.

Theorem 10 states that our human computable password management scheme is UF-RCA secure as long as RP-CSP (Conjecture 7) holds. In Section 4 we provide strong evidence in support of RP-CSP. In particular, no statistical algorithm can approximately recover the secret mapping given $m = \tilde{O}\left(n^{r(f)/2}\right)$ challenge-response pairs. To prove Theorem 10 we need to show that an adversary that breaks UF-RCA security for f_{k_1,k_2} can be used to approximately recover the secret mapping σ. We prove a more general result in Section 5.

▶ **Theorem 10.** *Let $\epsilon, \epsilon' > 0, \lambda \geq 1$ be given. Under the RP-CSP conjecture (Conjecture 7) the human computable password scheme defined by f_{k_1,k_2} is $\mathbf{UF} - \mathbf{RCA}\left(n, m, \lambda, \delta\right) -$ secure for any $m \leq n^{\min\{(k_2+1)/2, k_1+1-\epsilon'\}} - \lambda$ and $\delta > \left(\frac{1}{10} + \epsilon\right)^{\lambda}$.*

▶ Remark. In the Appendix we demonstrate that our security bounds are asymptotically tight. In particular, there is a statistical algorithm to break our human computable password schemes $\left(f_{k_1,k_2}\right)$ which requires $m = \tilde{O}\left(n^{(k_2+1)/2}\right)$ to 1-MSTAT to recover σ (See Theorem 27 in Section G). We also demonstrate that there is a attack based on Gaussian Elimination that uses $m = \tilde{O}\left(n^{k_1+1}\right)$ challenge-response pairs to recover σ.

3.2.1 Exact Security Bounds

We used the Constraint Satisfaction Problem solver from the Microsoft Solver Foundations library to attack our human computable password scheme[4]. In each instance we generated a random mapping $\sigma : [n] \to \mathbb{Z}_{10}$ and m random challenge response pairs $\left(C, f\left(\sigma\left(C\right)\right)\right)$ using the functions $f_{2,2}$ and $f_{1,3}$. We gave the CSP solver 2.5 days to find σ on a computer with a 2.83 GHz Intel Core2 Quad CPU and 4 GB of RAM. The full results of our experiments are in Appendix C. Briefly, the solver failed to find the random mapping in the following instances with $f = f_{2,2}$ and $f = f_{1,3}$: (1) $n = 30$ and $m = 100$, (2) $n = 50$ and $m = 1,000$ and (3) $n = 100$ and $m = 10,000$.

▶ Remark. While the theoretical security parameter for $f_{1,3}$ $\left(s\left(f_{1,3}\right) = 2\right)$ is slightly better than the security parameter for $f_{2,2}$ $\left(s\left(f_{2,2}\right) = 1.5\right)$, we conjecture that $f_{2,2}$ may be more secure for small values of n (e.g., $n \leq 100$) because it is less vulnerable to attacks based on Gaussian Elimination. In particular, there is a polynomial time attack on $f_{1,3}$ based on Gaussian Elimination that requires at most n^2 examples to recover σ, while the same attack would require n^3 examples with $f_{2,2}$. Our CSP solver was not able to crack $\sigma \in \mathbb{Z}_{10}^{100}$ given $10,000 = 100^2$ challenge response pairs with $f_{2,2}$.

[4] Thanks to David Wagner for suggesting the use of SAT solvers.

Human Computable Password Challenge.

We are challenging the security and cryptography community to break our human computable password scheme for instances that our CSP solver failed to crack (see Appendix B for more details about the challenge). Briefly, for each challenge we selected a random secret mapping $\sigma \in \mathbb{Z}_{10}^n$, and published (1) m single digit challenge-response pairs $(C_1, f(\sigma(C_1)))$,. . ., $(C_m, f(\sigma(C_m)))$, where each clause C_i is chosen uniformly at random from X_k, and (2) 20 length–$\lambda = 10$ password challenges $\vec{C}_1, \ldots, \vec{C}_{20} \in (X_k)^{10}$. The goal of each challenge is to correctly guess one of the secret passwords $p_i = f\left(\sigma\left(\vec{C}_i\right)\right)$ for some $i \in [20]$. The challenges can be found at `http://www.cs.cmu.edu/~jblocki/HumanComputablePasswordsChallenge/challenge.htm`. There is a \$20 prize associated with each individual challenge (total: \$360). We remark that these challenges remain unsolved even after they were presented during the rump sessions at a cryptography conference and a security conference[12].

4 Statistical Adversaries and Lower Bounds

Our main technical result (Theorem 14) is a lower bound on the number of single digit challenge-response pairs that a statistical algorithm needs to see to (approximately) recover the secret mapping σ. Our results are quite general and may be of independent interest. Given *any* function $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ we prove that *any* statistical algorithm needs $\tilde{\Omega}\left(n^{r(f)/2}\right)$ examples before it can find a secret mapping $\sigma' \in \mathbb{Z}_d^n$ such that σ' is ϵ-correlated with σ. We first introduce statistical algorithms in Section 4.1 before stating our main lower bound for statistical algorithms in Section 4.2. We also provide a high level overview of our proof in Section 4.2.

4.1 Statistical Algorithms

A statistical algorithm is an algorithm that solves a distributional search problem $\mathcal{Z}$. In our case the distributional search problem $\mathcal{Z}_{\epsilon,f}$ is to find a mapping τ that is ϵ-correlated with the secret mapping σ given access to m samples from Q_σ^f – the distribution over challenge response pairs induced by σ and f. A statistical algorithm can access the input distribution Q_σ^f by querying the 1-MSTAT oracle or by querying the VSTAT oracle (Definition 11).

▶ **Definition 11.** [26] [1-MSTAT(L) oracle and VSTAT oracle] Let D be the input distribution over the domain X. Given any function $h : X \to \{0, 1, \ldots, L-1\}$, 1-MSTAT$(L)$ takes a random sample x from D and returns $h(x)$. For an integer parameter $T > 0$ and any query function $h : X \to \{0, 1\}$, VSTAT (T) returns a value $v \in [p - \tau, p + \tau]$ where $p = \mathbb{E}_{x\sim D}[h(x)]$ and $\tau = \max\left\{\frac{1}{T}, \sqrt{\frac{p(1-p)}{T}}\right\}$.

In our context the domain $X = X_k \times \mathbb{Z}_d$ is the set of all challenge response pairs and the distribution $D = Q_\sigma^f$ is the uniform distribution over challenge-response pairs induced by σ and f. Feldman et al. [26] used the notion of statistical dimension (Definition 12) to lower bound the number of oracle queries necessary to solve a distributional search problem (Theorem 13). Before we can present the definition of statistical dimension we need to introduce the *discrimination norm*. Intuitively, if the discrimination norm is small then a statistical algorithm will (whp) not be able to distinguish between honest samples $(C, f(\sigma(C))$ and samples from reference distribution T over $X_k \times \mathbb{Z}_d$ which is completely independent of σ [5]. We define our reference distribution as follows:

$$\Pr_T[(C,i)] = \frac{\Pr_{x\sim\mathbb{Z}_d^k}[f(x)=i]}{|X_k|} \ .$$

Now given a set $\mathcal{D}' \subseteq \mathbb{Z}_d^n$ of secret mappings the discrimination norm of $\mathcal{D}'$ is denoted by $\kappa_2(\mathcal{D}')$ and defined as follows:

$$\kappa_2(\mathcal{D}') \doteq \max_{h,\|h\|=1}\left\{\mathbb{E}_{\sigma\sim\mathcal{D}'}\left[|\Delta(h,\sigma)|\right]\right\} \ ,$$

where $h : X_k \times \mathbb{Z}_d \to \mathbb{R}$, $\|h\| \doteq \sqrt{\mathbb{E}_{(C,i)\sim X_k\times\mathbb{Z}_d}\left[h^2(C,i)\right]}$ and

$$\Delta(h,\sigma) \doteq \mathbb{E}_{C\sim X_k}\left[h\left(C, f\left(\sigma\left(C\right)\right)\right)\right] - \mathbb{E}_{(C,i)\sim T}\left[h\left(C,i\right)\right] \ .$$

► **Definition 12.** [26][6]. For $\kappa > 0$, $\eta > 0$, $\epsilon > 0$, let d' be the largest integer such that for any mapping $\sigma \in \mathbb{Z}_d^n$ the set $\mathcal{D}_\sigma = \mathbb{Z}_d^n \setminus \{\sigma' \in \mathbb{Z}_d^n \mid \sigma' \text{ is } \epsilon\text{-correlated with } \sigma\}$ has size at least $(1-\eta)\cdot|\mathbb{Z}_d^n|$ and for any subset $\mathcal{D}' \subseteq \mathcal{D}_\sigma$ where $|\mathcal{D}'| \geq |\mathcal{D}_\sigma|/d'$, we have $\kappa_2(\mathcal{D}') \leq \kappa$. The **statistical dimension** with discrimination norm κ and error parameter η is d' and denoted by $\mathrm{SDN}(\mathcal{Z}_{\epsilon,f},\kappa,\eta)$.

Feldman et al. [26] proved the following lower bound on the number of 1-MSTAT and VSTAT queries needed to solve a distributional search problem. Intuitively, Theorem 13 implies that many queries are needed to solve a distributional search problem with high statistical dimension. In Section 4.2 we argue that the statistical dimension our distributional search problem (finding σ' that is ϵ-correlated with the secret mapping σ given m samples from the distribution Q_σ^f) is high.

► **Theorem 13.** *[26, Theorems 10 and 12] For $\kappa > 0$ and $\eta \in (0,1)$ let $d' = \mathrm{SDN}(\mathcal{Z}_{\epsilon,f},\kappa,\eta)$ be the statistical dimension of the distributional search problem $\mathcal{Z}_{\epsilon,f}$. Any randomized statistical algorithm that, given access to a VSTAT$\left(\frac{1}{3\kappa^2}\right)$ oracle (resp. 1-MSTAT(L)) for the distribution Q_σ^f for a secret mapping σ chosen randomly and uniformly from $\mathbb{Z}_d^n$, succeeds in finding a mapping $\tau \in \mathbb{Z}_d^n$ that is ϵ-correlated with σ with probability $\Lambda > \eta$ over the choice of distribution and internal randomness requires at least $\frac{\Lambda-\eta}{1-\eta}d'$ (resp. $\Omega\left(\frac{1}{L}\min\left\{\frac{d'(\Lambda-\eta)}{1-\eta},\frac{(\Lambda-\eta)^2}{\kappa^2}\right\}\right)$) calls to the oracle.*

As Feldman et al. [26] observe, almost all known algorithmic techniques can be modeled within the statistical query framework. In particular, techniques like Expectation Maximization[25], local search, MCMC optimization[30], first and second order methods for convex optimization, PCA, ICA, k-means can be modeled as a statistical algorithm even with $L = 2$ – see [16] and [22] for proofs. One issue is that a statistical simulation might need polynomially more samples. However, for $L > 2$ we can think of our queries to 1-MSTAT(L) as evaluating L disjoint functions on a random sample. Indeed, Feldman et al. [26] demonstrate that there is a statistical algorithm for binary planted satisfiability problems using $\tilde{O}\left(n^{r(f)/2}\right)$ calls to 1-MSTAT $\left(n^{\lceil r(f)/2\rceil}\right)$.

[5] Observe that this implies that a statistical algorithm cannot find the secret σ. In particular, because the distribution T is independent of the secret mapping σ samples from T will not leak any information about σ.

[6] For the sake of simplicity we define the discrimination norm and the statistical dimension using our particular distributional search problem $\mathcal{Z}_{\epsilon,f}$. Our definition is equivalent to the definition in [26] once we fix the reference distribution T.

▶ Remark. We can also use the statistical dimension to lower bound the number of queries that an algorithm would need to make to other types of statistical oracles to solve a distributional search problem. For example, we could also consider an oracle MVSTAT(L, T) that takes a query $h : X \to \{0, \ldots, L-1\}$ and a set $\mathcal{S}$ of subsets of $\{0, \ldots, L-1\}$ and returns a vector $v \in \mathbb{R}^L$ s.t for every $Z \in \mathcal{S}$

$$\left|\sum_{i \in Z} v[i] - p_Z\right| \leq \max\left\{\frac{1}{T}, \sqrt{\frac{p_Z(1-p_Z)}{T}}\right\},$$

where $p_Z = \Pr_{x \sim D}[h(x) \in Z]$ and the cost of the query is $|\mathcal{S}|$. Feldman et al. [26, Theorem 7] proved lower bounds similar to Theorem 13 for the MVSTAT oracle. In this paper we focus on the 1-MSTAT and VSTAT oracles for simplicity of presentation.

4.2 Statistical Dimension Lower Bounds

We are now ready to state our main technical result[7].

▶ **Theorem 14.** *Let $\sigma \in \mathbb{Z}_d^n$ denote a secret mapping chosen uniformly at random, let Q_σ^f be the distribution over $X_k \times \mathbb{Z}_d$ induced by a function $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ with distributional complexity $r = r(f)$. Any randomized statistical algorithm that finds an assignment τ such that τ is $\left(\sqrt{\frac{-2\ln(\eta/2)}{n}}\right)$-correlated with σ with probability at least $\Lambda > \eta$ over the choice of σ and the internal randomness of the algorithm needs at least m calls to the 1-MSTAT(L) oracle (resp. VSTAT$\left(\frac{n^r}{2(\log n)^{2r}}\right)$ oracle) with $m \cdot L \geq c_1\left(\frac{n}{\log n}\right)^r$ (resp. $m \geq n^{c_1 \log n}$) for a constant $c_1 = \Omega_{k,1/(\Lambda-\eta)}(1)$ which depends only on the values k and $\Lambda - \eta$. In particular if we set $L = \left(\frac{n}{\log n}\right)^{r/2}$ then our algorithms needs at least $m \geq c_1\left(\frac{n}{\log n}\right)^{r/2}$ calls to 1-MSTAT(L).*

The proof of Theorem 14 follows from Theorems 16 and 13. Theorems 14 and 16 generalize results of Feldman et al. [26] which only apply for binary predicates $f : \{0,1\}^k \to \{0,1\}$. An interested reader can find our proofs in Appendix D. At a high level our proof proceeds as follows: Given any function $h : X_k \times \mathbb{Z}_d \to \mathbb{R}$ we show that $\Delta(\sigma, h)$ can be expressed in the following form: $\Delta(\sigma, h) = \sum_{\ell=r(f)}^{k} \frac{1}{|X_\ell|} b_\ell(\sigma)$, where $|X_\ell| = \Theta\left(n^\ell\right)$ and each function b_ℓ has degree ℓ (Lemma 21). We then use the general hypercontractivity theorem [45, Theorem 10.23] to obtain the following concentration bound.

▶ **Lemma 15.** *Let $b : \mathbb{Z}_d^n \to \mathbb{R}$ be any function with degree at most ℓ, and let $\mathcal{D}' \subseteq \mathbb{Z}_d^n$ be a set of assignments for which $d' = d^n / |\mathcal{D}'| \geq e^\ell$. Then $\mathbb{E}_{\sigma \sim \mathcal{D}'}[|b(\sigma)|] \leq 2(\ln d'/c_0)^{\ell/2}\|b\|_2$, where $c_0 = \ell\left(\frac{1}{2ed}\right)$ and $\|b\|_2 = \sqrt{\mathbb{E}_{x \sim \mathbb{Z}_d^n}\left[b(x)^2\right]}$.*

We then use Lemma 15 to bound $\mathbf{E}_{\sigma \sim \mathcal{D}'}[\Delta(\sigma, h)]$ for any set $\mathcal{D}' \subseteq \mathbb{Z}_d^n$ such that $|\mathcal{D}'| = \left|\mathbb{Z}_d^k\right| / d'$ (Lemma 25). This leads to the following bound on $\kappa_2(\mathcal{D}') = O_k\left((\ln d'/n)^{r(f)/2}\right)$.

[7] We remark that for our particular family of human computable functions f_{k_1,k_2} we could get a theorem similar to Theorem 14 by selecting $\sigma \sim \{0,5\}^n$ and appealing directly to results of Feldman et al. [26]. However, this theorem would be weaker than Theorem 14 as it would only imply that a statistical algorithm cannot find an assignment σ' that is $\frac{1}{2} - \frac{1}{10} + \epsilon$-correlated with σ for $\epsilon > 0$. In contrast, our theorem implies that we cannot find σ' that is ϵ-correlated.

▶ **Theorem 16.** *There exists a constant $c_Q > 0$ such that for any $\epsilon > 1/\sqrt{n}$ and $q \geq n$ we have*

$$\mathrm{SDN}\left(\mathcal{Z}_{\epsilon,f}, \frac{c_Q (\log q)^{r/2}}{n^{r/2}}, 2e^{-n\cdot\epsilon^2/2}\right) \geq q \ ,$$

where $r = r(f)$ is the distributional complexity of f.

Discussion

We view Theorem 14 as strong evidence for RP-CSP (Conjecture 7) because almost all known algorithmic techniques can be modeled within the statistical query framework[16, 22]. Thus, Theorem 14 rules out most known attacks that an adversary might mount. It also implies that many popular heuristic based SAT solvers (e.g., DPLL[24]) will not be able to recover σ in polynomial time. While Theorem 14 does not rule our attacks based on Gaussian Elimination we consider this class of attacks separately. We need $m = \tilde{O}\left(n^{g(f)+1}\right)$ examples to extract $O(n)$ linear constraints and solve for σ (see Appendix G.2). However, our composite security parameter $s(f) \geq g(f) + 1$ accounts for attacks based on Gaussian Elimination.

5 Security Analysis

In the last section we presented evidence in support of RP-CSP (Conjecture 7) by showing that any statistical adversary needs $m = \tilde{\Omega}\left(n^{r(f)/2}\right)$ examples to (approximately) recover σ. However, RP-CSP only says that it is hard to (approximately) recover the secret mapping σ, not that it is hard to forge passwords. As an example consider the following NP-hard problem from learning theory: find a 2-term DNF that is consistent with the labels in a given dataset. Just because 2-DNF is hard to learn in the *proper* learning model does not mean that it is NP-hard to learn a good classifier for 2-DNF. Indeed, if we allow our learning algorithm to output a linear classifier instead of a 2-term DNF then 2-DNF is easy to learn [37]. Could an adversary win our password security game without properly learning the secret mapping?

Theorem 18, our main result in this section, implies that the answer is no. Informally, Theorem 18 states that any adversary that breaks UF-RCA security of our human computable password scheme f_{k_1,k_2} can also (approximately) recover the secret mapping σ. This implies that our human computable password scheme is UF-RCA secure as long as RP-CSP holds. Of course, for some functions it is very easy to predict challenge-response pairs without learning σ. For example, if f is the constant function – or any function highly correlated with the constant function – then it is easy to predict the value of $f(\sigma(C))$. However, any function that is highly correlated with a constant function is a poor choice for a human computable passwords scheme. We argue that any adversary that can win the password game can be converted into an adversary that properly learns σ provided that the output of function f is evenly distributed (Definition 17).

▶ **Definition 17.** We say that the output of a function $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ is *evenly distributed* if there exists a function $g : \mathbf{Z}_d^{k-1} \to \mathbb{Z}_d$ such that $f(x_1, \ldots, x_k) = g(x_1, \ldots, x_{k-1}) + x_k \mod d$.

Clearly, our family f_{k_1,k_2} has evenly distributed output. To see this we simply set $g = f_{k_1,k_2-1}$. We are now ready to state our main result from this section.

▶ **Theorem 18.** *Suppose that f has evenly distributed output, but that f is not* $\mathbf{UF-RCA}\ (n,m,\lambda,\delta)$ *– secure for $\delta > \left(\frac{1}{d}+\epsilon\right)^{\lambda}$. Then there is a probabilistic polynomial time algorithm (in n, m, λ and $1/\epsilon$) that extracts a string $\sigma' \in \mathbb{Z}_d^n$ that is $\epsilon/8$-correlated with σ with probability at least $\frac{\epsilon^3}{(8d)^2}$ after seeing $m+\lambda$ example challenge response pairs.*

The proof of Theorem 18 is in Appendix E. We overview the proof here. The proof of Theorem 18 uses Theorem 19 as a subroutine. Theorem 19 shows that we can, with reasonable probability, find a mapping σ' that is correlated with σ given predictions of $f\left(\sigma\left(C\right)\right)$ for each clause as long as the probability that each prediction is accurate is slightly better than a random guess (e.g., $\frac{1}{d}+\delta$). The proof of Theorem 19 is in Appendix E.

▶ **Theorem 19.** *Let f be a function with evenly distributed output (Definition 17), let $\sigma \sim \mathbb{Z}_d^n$ denote the secret mapping, let $\epsilon > 0$ be any constant and suppose that for every $C \in X_k$ we are given labels $\ell_C \in \mathbb{Z}_d$ s.t. $\Pr_{C\sim X_k}\left[f\left(\sigma\left(C\right)\right) = \ell_C\right] \geq \frac{1}{d}+\epsilon$. There is a polynomial time algorithm (in n, m,$1/\epsilon$) that finds a mapping $\sigma' \in \mathbb{Z}_d^n$ such that σ' is $\epsilon/2$-correlated with σ with probability at least $\frac{\epsilon}{2d^2}$*

The remaining challenge in the proof of Theorem 18 is to show that there is an efficient algorithm to extract predictions of $f\left(\sigma\left(C\right)\right)$ given blackbox access to an adversary $\mathcal{A}$ that breaks UF-RCA security. However, just because the adversary $\mathcal{A}$ gives the correct response to an entire password challenge $C_1,\ldots,C_\lambda$ with probability greater than $\left(\frac{1}{d}+\epsilon\right)^{\lambda}$ it does not mean that the response to each individual challenge C is correct with probability $\frac{1}{d}+\epsilon$. To obtain our predictions for individual clauses C we draw λ extra example challenge response pairs $\left(C_1', f\left(\sigma\left(C_1'\right)\right)\right),\ldots,\left(C_\lambda', f\left(\sigma\left(C_\lambda'\right)\right)\right)$, which we use to check the adversary. To obtain the label for a clause C we select a random index $i \in [\lambda]$ and give $\mathcal{A}$ the password challenge $C_1',\ldots,C_\lambda'$, replacing C_i' with C. If for some $j < i$ the label for clause C_j' is not correct (e.g., $\neq f\left(\sigma\left(C_j'\right)\right)$) then we discard the label and try again. Claim 26 in Appendix E shows that this process will give us predictions for individual clauses that are accurate with probability at least $\frac{1}{d}+\epsilon$.

6 Related Work

The literature on passwords has grown rapidly over the past decade (e.g., see [40, 46, 18, 19, 38, 15].) Perhaps most related to our paper is the work of Blocki et al. [13, 14] and Blum and Vempala [17] on developing usable and secure password management schemes. While the password management schemes proposed in these works are easier to use (e.g., involve less memorization and/or computation) than our human computable password scheme, these schemes only remain secure up to their information theoretic limit – after a very small (e.g., 1–6) number of breaches security guarantees start to break down. By contrast, our schemes remain secure after a large (e.g., 100) number or breaches.

In contrast to our work, password management software (e.g., PwdHash [49] or KeePass [48]) relies strong trust assumptions about the user's computational devices. The recent breach at LastPass[8] highlights the potential danger of such strong assumptions.

Hopper and Blum [34] designed a Human Identification Protocol based on noisy parity, a learning problem that is believed to be hard[9]. We emphasize a few fundamental differences

[8] See `https://blog.lastpass.com/2015/06/lastpass-security-notice.html/` (Retrieved 9/1/2015).
[9] Subsequent work [35, 31, 20, 36] has explored the use of the Hopper-Blum protocol for authentication on pervasive devices like smartcards.

between our work and the work of Hopper and Blum. First, a single digit challenge in their protocol consists of an n-digit vector $x \in \mathbb{Z}_{10}^n$ and the user responds with the $\mod 10$ sum of the digits at $\ell \leq n$ secret locations (occasionally the user is supposed to respond with a random digit instead of the correct response so that the adversary cannot simply use Gaussian Elimination to find the secret locations). By contrast, a single digit challenge in our protocol consists of an ordered clause of length $k \ll n$. Second, their protocols allow for an $O\left(n^{\ell/2}\right)$-time attack called Meet-In-The-Middle [34] after the adversary has seen $\tilde{O}(\log\binom{n}{\ell})$ challenge-response pairs. Thus, it is critically important to select ℓ sufficiently large (e.g., $\ell = \Omega(\log(n))$) in the Hopper-Blum protocol to defend against this Meet-In-The-Middle attack. By contrast, we focus on computation of *very simple* functions over a *constant* number of variables so that a human can compute the response to each challenge quickly. In particular, we provide strong evidence that our scheme is secure against any polynomial time attacker even if the adversary has seen up to $O\left(n^{c \cdot k}\right)$ challenge-response pairs for some constant $c \geq 1$. Finally, computations in our protocols are deterministic. This is significant because humans are not good at consciously generating random numbers [53, 27, 42] (e.g., noisy parity could be easy to learn when humans are providing source of noise)[10].

Naor and Pinkas[43] proposed using visual cryptography[44] to address a related problem: how can a human verify that a message he received from a trusted server has not been tampered with by an adversary? Their protocol requires the human to carry a visual transparency (a shared secret between the human and the trusted server in the visual cryptography scheme), which he will use to verify that messages from the trusted server have not been altered.

A related goal in cryptography, constructing pseudorandom generators in NC^0, was proposed by Goldreich [32] and by Cryan and Miltersen [23]. In Goldreich's construction we fix $C_1, \ldots, C_m \in [n]^k$ once and for all, and a binary predicate $P : \{0,1\}^k \rightarrow \{0,1\}$. The pseudorandom generator is a function $G : \{0,1\}^n \rightarrow \{0,1\}^m$, whose i'th bit $G(x)[i]$ is given by P applied to the bits of x specified by C_i. O'Donnel and Witmer gave evidence that the "Tri-Sum-And" predicate ($TSA\left(x_1, \ldots, x_5\right) = x_1 + x_2 + x_3 + x_4 x_5 \mod 2$) provides near-optimal stretch. In particular, they showed that for $m = n^{1.5-\epsilon}$ Goldreich's construction with the TSA predicate is secure against subexponential-time attacks using SDP hierarchies. Our candidate human-computable password schemes use functions $f : \mathbb{Z}_{10}^k \rightarrow \mathbb{Z}_{10}$ instead of binary predicates. While our candidate functions are contained in NC^0, we note that an arbitrary function in NC^0 is not necessarily human computable.

On a technical level our statistical dimension lower bounds extend work of Feldman et al. [26], who considered the problem of finding a planted solution in a random *binary* planted constraint satisfiability problem. We extend their analysis to handle non-binary planted constraint satisfiability problems, and argue that our candidate human computable password schemes are secure. We will discuss this work in more detail later in the paper.

7 Discussion

7.1 Improving Response Time

The easiest way to improve response time is to decrease λ –the number of single-digit challenges that the user needs to solve. However, if the user wants to ensure that each of his passwords are strong enough to resist offline dictionary attacks then he would need

[10] Hopper and Blum also proposed a deterministic variant of their protocol called sum of k-mins, but this variant is *much* less secure. See additional discussion in the appendix.

to select a larger value of λ (e.g., $\lambda \geq 10$). Fortunately, there is a natural way to circumvent this problem. The user could save time by memorizing a mapping $w : \mathbb{Z}_{10} \to \{x \mid x \text{ is one of 10,000 most common english words}\}$ and responding to each challenge C with $w(f(\sigma(C)))$ – the word corresponding to the digit $f(\sigma(C))$. Now the user can create passwords strong enough to resist offline dictionary attacks by responding to just 3–5 challenges. Even if the adversary learns the words in the user's set he won't be able to mount online attacks. Predicting $w(f(\sigma(C)))$ is at least as hard as predicting $f(\sigma(C))$ even if the adversary knows the exact mapping w[11].

7.2 One-Time Challenges

Malware

Consider the following scenario: the adversary infects the user's computer with a keylogger which is never detected over the user's lifetime. We claim that it is possible to protect the user in this extreme scenario using our scheme by generating multiple (e.g., 10^6) one-time passwords for each of the user's accounts. When we initially generate the secret mapping $\sigma \sim \mathbb{Z}_d^n$ we could also generate cryptographic hashes for millions of one-time passwords $\mathbf{H}\left(\vec{C}, f_{k_1,k_2}\left(\sigma\left(\vec{C}\right)\right)\right)$. While usability concerns make this approach infeasible in a traditional password scheme (it would be far too difficult for the user to memorize a million one-time passwords for each of his accounts), it may be feasible to do this using a human computable password scheme. In our human computable password scheme we could select k_1 and k_2 large enough that $s(f_{k_1,k_2}) = \min\{k_1+1, (k_2+1)/2\} \geq 6$. Assuming that the user authenticates fewer than 10^6 times over his lifetime a polynomial time adversary would never obtain enough challenge-response examples to learn σ. The drawback is that f_{k_1,k_2} will take longer for a user to execute in his head.

Secure Cryptography in a Panoptic World

Standard cryptographic algorithms could be easily broken in a panoptic world where the user only has access to a semi-trusted computer (e.g., if a user asks a semi-trusted computer to sign a message m using a secret key sk stored on the hard drive then the computer will respond with the correct value $Sign(sk, m)$, but the adversary will learn the values m and sk). Our human computable password schemes could also be used to secure some cryptographic operations (e.g., signatures) in a panoptic world by leveraging recent breakthroughs in program obfuscation [29]. The basic idea is to obfuscate a "password locked" circuit $P_{\sigma,sk,r}$ that can sign messages under a secret key sk – we need a trusted setup phase for this step. The circuit $P_{\sigma,sk,r}$ will only sign a message m if the user provides the correct response to a unique (pseudorandomly generated) challenge for m.

7.3 Open Questions

Eliminating the Semi-Trusted Computer

Our current scheme relies on a semi-trusted computer to generate and store random public challenges. An adversary with full control over the user's computer might be able to extract the user's secret if he is able to see the user's responses to $O(n)$ adaptively selected password challenges. Can we eliminate the need for a semi-trusted computer?

[11] In fact, it is quite likely that it is much harder for the adversary to predict $w(f(\sigma(C)))$ because the adversary will not see which word corresponds with each digit.

Exact Security Bounds

While we provided asymptotic security proofs for our human computable password schemes, it is still important to understand how much effort an adversary would need to expend to crack the secret mapping for specific values of n and m. Our attacks with a SAT solver (see Appendix C) indicate that the value $n = 26$ is too small to provide UF-RCA security even with small values of m (e.g., $m = 50$). As n increases the problem rapidly gets harder for our SAT solver (e.g., with $n = 50$ and $m = 1000$ the solver failed to find σ). We also present a public challenge with specific values of n and m to encourage cryptography and security researchers to find other techniques to attack our scheme.

References

1 Cert incident note in-98.03: Password cracking activity. http://www.cert.org/incident_notes/IN-98.03.html, July 1998. Retrieved 8/16/2011.

2 Nato site hacked. http://www.theregister.co.uk/2011/06/24/nato_hack_attack/, June 2011. Retrieved 8/16/2011.

3 Zappos customer accounts breached. http://www.usatoday.com/tech/news/story/2012-01-16/mark-smith-zappos-breach-tips/52593484/1, January 2012. Retrieved 5/22/2012.

4 Oh man, what a day! an update on our security breach. http://blogs.atlassian.com/news/2010/04/oh_man_what_a_day_an_update_on_our_security_breach.html, April 2010. Retrieved 8/18/2011.

5 Apple security blunder exposes lion login passwords in clear text. http://www.zdnet.com/blog/security/apple-security-blunder-exposes-lion-login-passwords-in-clear-text/11963, May 2012. Retrieved 5/22/2012.

6 Update on playstation network/qriocity services. http://blog.us.playstation.com/2011/04/22/update-on-playstation-network-qriocity-services/, April 2011. Retrieved 5/22/2012.

7 An update on linkedin member passwords compromised. http://blog.linkedin.com/2012/06/06/linkedin-member-passwords-compromised/, June 2012. Retrieved 9/27/2012.

8 Data breach at ieee.org: 100k plaintext passwords. http://ieeelog.com/, September 2012. Retrieved 9/27/2012.

9 Important customer security announcement. http://blogs.adobe.com/conversations/2013/10/important-customer-security-announcement.html, October 2013. Retrieved 2/10/2014.

10 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.

11 Sam. Biddle. Anonymous leaks 90,000 military email accounts in latest antisec attack. http://gizmodo.com/5820049/anonymous-leaks-90000-military-email-accounts-in-latest-antisec-attack, July 2011. Retrieved 8/16/2011.

12 Jeremiah Blocki, Manuel Blum, and Anupam Datta. Human-computable passwords. ASIACRYPT Rump Session, 2013. URL: http://asiacrypt.2013.rump.cr.yp.to/b0279d7741ad5bab24cf5c55fd292d5c.pdf.

13 Jeremiah Blocki, Manuel Blum, and Anupam Datta. Naturally rehearsing passwords. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*,

volume 8270 of *Lecture Notes in Computer Science*, pages 361–380. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-42045-0_19.

14 Jeremiah Blocki, Saranga Komanduri, Lorrie Faith Cranor, and Anupam Datta. Spaced repetition and mnemonics enable recall of multiple strong passwords. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*, 2015. URL: http://www.internetsociety.org/doc/spaced-repetition-and-mnemonics-enable-recall-multiple-strong-passwords.

15 Jeremiah Blocki, Saranga Komanduri, Ariel Procaccia, and Or Sheffet. Optimizing password composition policies. In *Proceedings of the 14th ACM Conference on Electronic Commerce.* ACM, 2013.

16 Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138. ACM, 2005.

17 Manuel Blum and Santosh Vempala. Publishable humanly usable secure password creation schemas. *Proc. of HCOMP*, 2015.

18 J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 538–552. IEEE, 2012.

19 S. Boztas. Entropies, guessing, and cryptography. *Department of Mathematics, Royal Melbourne Institute of Technology, Tech. Rep*, 6, 1999.

20 Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. Hbˆ+ˆ+: a lightweight authentication protocol secure against some attacks. In *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006. SecPerU 2006. Second International Workshop on*, pages 28–33. IEEE, 2006.

21 I.A.D. Center. Consumer password worst practices. *Imperva (White Paper)*, 2010.

22 Cheng Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

23 Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in nc0. In *Mathematical Foundations of Computer Science 2001*, pages 272–284. Springer, 2001.

24 Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, July 1960. doi:10.1145/321033.321034.

25 Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

26 Vitaly Feldman, Will Perkins, and Santosh Vempala. On the complexity of random satisfiability problems with planted solutions. In *Proceedings of the 47th annual ACM symposium on Symposium on Theory of Computing.* ACM, 2015.

27 M. Figurska, M. Stanczyk, and K. Kulesza. Humans cannot consciously generate random numbers sequences: Polemic study. *Medical hypotheses*, 70(1):182–185, 2008.

28 D. Florencio and C. Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

29 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE, 2013.

30 Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

31 Henri Gilbert, Matthew Robshaw, and Herve Sibert. Active attack against hb+: a provably secure lightweight authentication protocol. *Electronics Letters*, 41(21):1169–1170, 2005.

32 Oded Goldreich. Candidate one-way functions based on expander graphs. In *Studies in Complexity and Cryptography*. Citeseer, 2000.

33 Graham J Hitch. The role of short-term working memory in mental arithmetic. *Cognitive Psychology*, 10(3):302–323, 1978.

34 N. Hopper and M. Blum. Secure human identification protocols. *Advances in cryptology-ASIACRYPT 2001*, pages 52–66, 2001.

35 Ari Juels and Stephen A Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology–CRYPTO 2005*, pages 293–308. Springer, 2005.

36 Jonathan Katz and Ji Sun Shin. Parallel and concurrent security of the hb and hb+ protocols. In *Advances in Cryptology-EUROCRYPT 2006*, pages 73–87. Springer, 2006.

37 Michael Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. The MIT Press, 1994.

38 S. Komanduri, R. Shay, P.G. Kelley, M.L. Mazurek, L. Bauer, N. Christin, L.F. Cranor, and S. Egelman. Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2595–2604. ACM, 2011.

39 H. Kruger, T. Steyn, B. Medlin, and L. Drevin. An empirical assessment of factors impeding effective password management. *Journal of Information Privacy and Security*, 4(4):45–59, 2008.

40 J.L. Massey. Guessing and entropy. In *Information Theory, 1994. Proceedings., 1994 IEEE International Symposium on*, page 204. IEEE, 1994.

41 G.A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

42 Dave. Munger. Is 17 the "most random" number? `http://scienceblogs.com/cognitivedaily/2007/02/is_17_the_most_random_number.php`, 2007. Retrieved 8/16/2011.

43 Moni Naor and Benny Pinkas. Visual authentication and identification. In *Advances in Cryptology-CRYPTO'97*, pages 322–336. Springer, 1997.

44 Moni Naor and Adi Shamir. Visual cryptography. In *Advances in Cryptology-EUROCRYPT'94*, pages 1–12. Springer, 1995.

45 Ryan O'Donnell. Analysis of boolean functions. *Textbook in Progress. Available online at http://analysisofbooleanfunctions.org/*, 2014.

46 J. Pliam. On the incomparability of entropy and marginal guesswork in brute-force attacks. *Progress in Cryptology-INDOCRYPT 2000*, pages 113–123, 2000.

47 N. Provos and D. Mazieres. Bcrypt algorithm.

48 D Reichl. Keepass password safe, 2013. *Retrieved July*, 10, 2013.

49 Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C Mitchell. Stronger password authentication using browser extensions. In *Usenix security*, pages 17–32. Baltimore, MD, USA, 2005.

50 Abe. Singer. No plaintext passwords. *;login: THE MAGAZINE OF USENIX & SAGE*, 26(7), November 2001. Retrieved 8/16/2011.

51 Saul Sternberg. Memory-scanning: Mental processes revealed by reaction-time experiments. *Cognitive psychology: Key readings*, 48, 2004.

52 Hedderik van Rijn, Leendert van Maanen, and Marnix van Woudenberg. Passing the test: Improving learning gains by balancing spacing and testing effects. In *Proceedings of the 9th International Conference of Cognitive Modeling*, 2009.

53 W.A. Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. *Psychological Bulletin*, 77(1):65, 1972.

54 PA Wozniak and Edward J Gorzelanczyk. Optimization of repetition spacing in the practice of learning. *Acta neurobiologiae experimentalis*, 54:59–59, 1994.

Image I			...	
σ(I)	9	3	...	6

Figure 2 A Random Mapping from Images to Digits.

A Authentication Process

In this section of the appendix we illustrate our human computable password schemes graphically. In our examples, we use the function $f = f_{2,2}$. To compute the response $f(\sigma(C))$ to a challenge $C = \{x_0, \ldots, x_{13}\}$ the user computes $f(\sigma(C)) = \sigma\left(x_{\sigma(x_{10})+\sigma(x_{11}) \mod 10}\right) + \sigma(x_{12}) + \sigma(x_{13}) \mod 10$

Memorizing a Random Mapping

To begin using our human computable password schemes the user begins by memorizing a secret random mapping $\sigma : [n] \to \{0, \ldots, 9\}$ from n objects (e.g., letters, pictures) to digits. See Figure 2 for an example.

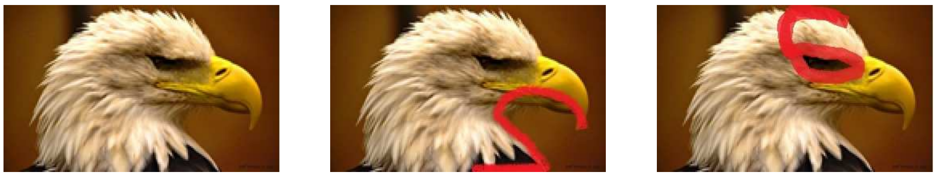
The computer can provide the user with mnemonics to help memorize the secret mapping σ – see Figures 3b and 3c. For example, if we wanted to help the user remember that $\sigma(eagle) = 2$ we would show the user Figure 3b. We observe that a $10 \times n$ table of mnemonic images would be sufficient to help the user memorize *any* random mapping σ. We stress that the computer will only save the original image (e.g., Figure 3a). The mnemonic image (e.g., Figure 3b or 3c) would be discarded after the user memorizes $\sigma(eagle)$.

Single-Digit Challenges

In our scheme the user computes each of his passwords by responding to a sequence of single-digit challenges. For $f = f_{2,2}$ a single-digit challenge is a tuple $C \in [n]^{14}$ of fourteen objects. See Figure 6 for an example. To compute the response $f(\sigma(C))$ to a challenge $C = \{x_0, \ldots, x_{13}\}$ the user computes $f(\sigma(C)) = \sigma\left(x_{\sigma(x_{10})+\sigma(x_{11}) \mod 10}\right) + \sigma(x_{12}) + \sigma(x_{13}) \mod 10$. Observe that this computation involves just three addition operations modulo ten. See Figure 1 for an example. In this example the response to the challenge $C = \{x_0 = \text{burger}, x_1 = \text{eagle}, \ldots, x_{10} = \text{lightning}, x_{11} = \text{dog}, x_{12} = \text{man standing on world}, x_{13} = \text{kangaroo}\}$ is

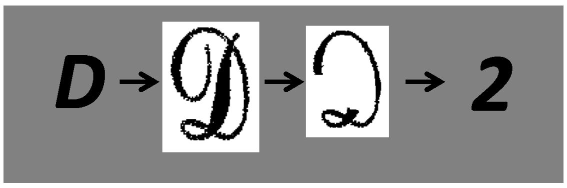
$$\begin{aligned}
f(\sigma(C)) &= \sigma\left(x_{\sigma(x_{10})+\sigma(x_{11}) \mod 10}\right) + \sigma(x_{12}) + \sigma(x_{13}) \mod 10 \\
&= \sigma\left(x_{\sigma(\text{lightning})+\sigma(\text{dog}) \mod 10}\right) \\
&\quad +\sigma(\text{man standing on world}) + \sigma(\text{kangaroo}) \mod 10 \\
&= \sigma\left(x_{9+3 \mod 10}\right) + \sigma(\text{man standing on world}) + \sigma(\text{kangaroo}) \mod 10 \\
&= \sigma(\text{minions}) + \sigma(\text{man standing on world}) + \sigma(\text{kangaroo}) \mod 10 \\
&= 7 + 4 + 5 \mod 10 = 6\ .
\end{aligned}$$

We stress that this computation is done entirely in the user's head. It takes the main author 7.5 seconds on average to compute each response.

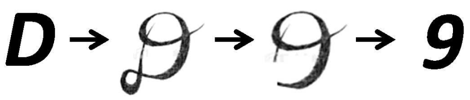


(a) Original photo (an eagle). **(b)** Mnemonic to help the user remember $\sigma(eagle) = 2$. **(c)** Mnemonic to help the user remember $\sigma(eagle) = 6$.

Figure 3 Mnemonics to help memorize the secret mapping σ.

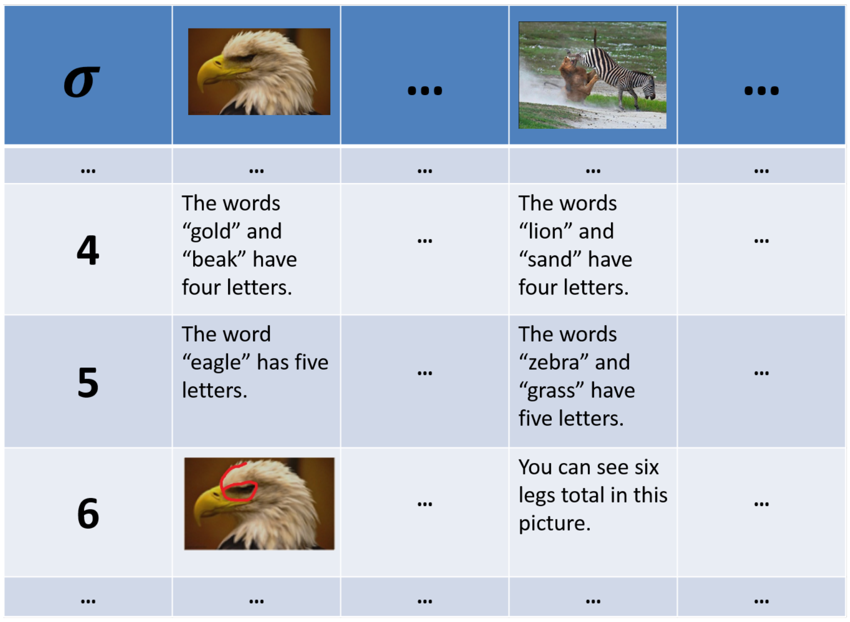


(a) $M_{D,2}$.



(b) $M_{D,9}$.

Figure 4 Mnemonics to help the user memorize the secret mapping σ.



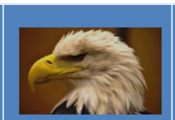
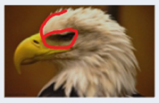
σ		...		...
...	...	...	...	...
4	The words "gold" and "beak" have four letters.	...	The words "lion" and "sand" have four letters.	...
5	The word "eagle" has five letters.	...	The words "zebra" and "grass" have five letters.	...
6		...	You can see six legs total in this picture.	...
...	...	...	...	...

Figure 5 Table of Mnemonic Helpers to Help Learn Any Secret Mapping

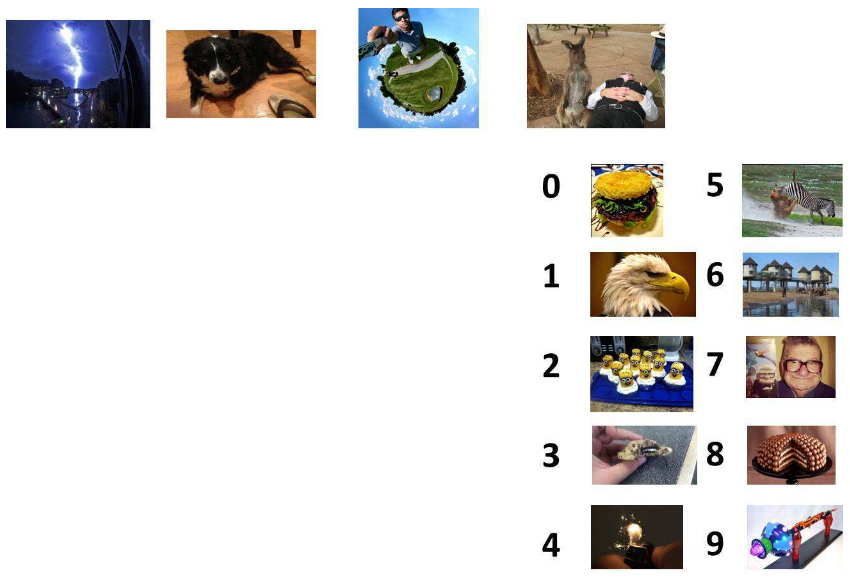


Figure 6 A single-digit challenge.

Creating an Account

To help the user create an account the computer would first pick a sequence of single-digit challenges $C_1, \ldots, C_\lambda$, where the security parameter is typically $\lambda = 10$, and would display the first challenge C_1 to the user – see Figure 7 for an example. To compute the first digit of his password the user would compute $f(\sigma(C_1))$. After the user types in the first digit $f(\sigma(C_1))$ of his password the computer will display the second challenge C_2 to the user – see Figure 8. After the user creates his account the computer will store the challenges $C_1, \ldots, C_{10}$ in public memory. The password $pw = f(\sigma(C_1)) \ldots f(\sigma(C_\lambda))$ will not be stored on the user's local computer (the authentication server may store the cryptographic hash of pw).

Authentication

Authenticating is very similar to creating an account. To help the user recompute his password for an account the computer first looks up the challenges $C_1, \ldots, C_\lambda$ which were stored in public memory, and the user authenticates by computing his password $pw = f(\sigma(C_1)) \ldots f(\sigma(C_\lambda))$. We stress that the single-digit challenges the user sees during authentication will be the same single-digit challenges that the user saw when he created the account. The authentication server verifies that the cryptographic hash of pw matches its record.

Helping the user remember his secret mapping

The computer keeps track of when the user rehearses each value of his secret mapping (e.g., $(i, \sigma(i))$ for each $i \in [n]$), and reminds the user to rehearse any part of his secret mapping that he hasn't used in a long time. One advantage of our human computable password scheme (compared with the Shared Cues scheme of Blocki et al.[13]) is that most users will use each part of their secret mapping often enough that they will not need to be reminded to rehearse – see discussion in Section A.2. The disadvantage is that we require the user to spend extra effort computing his passwords each time he authenticates.

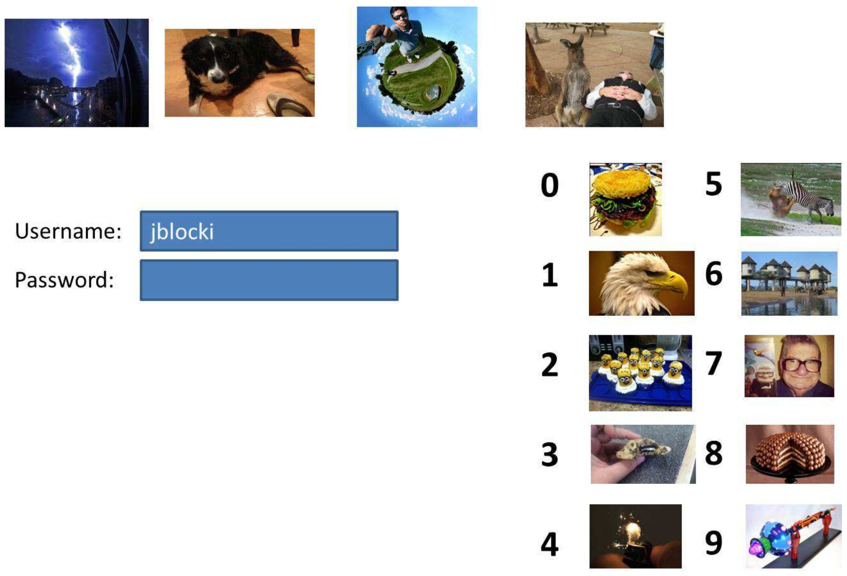


Figure 7 Login Screen

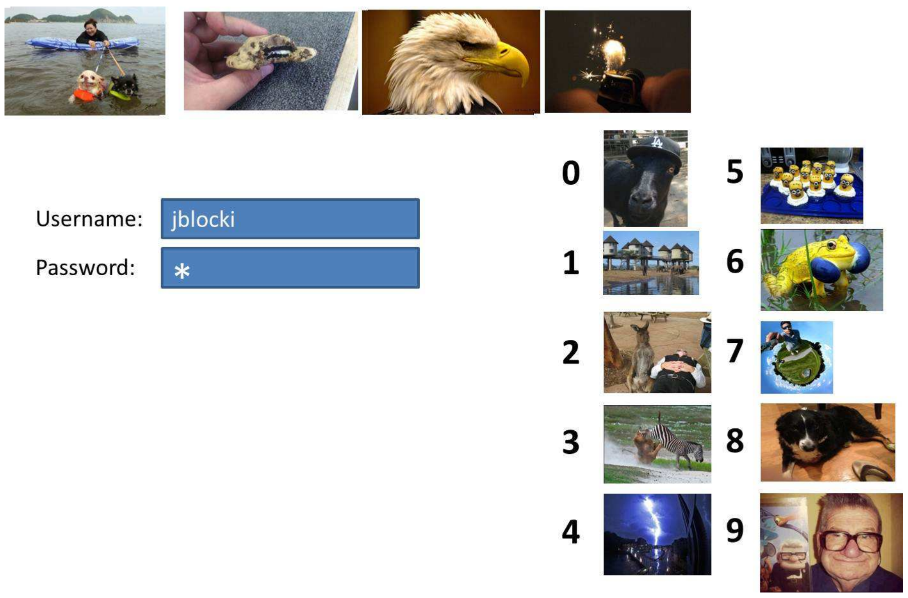


Figure 8 Login Screen after the user responds to the first single-digit challenge

Algorithm 1: MemorizeMapping

```
input : $I_1, ..., I_n$, $d$, $b$ and $M_{i,j}$ for $i \in [n], j \in \{0, \ldots, d-1\}$.
  Given $j \in \{0, \ldots, d-1\}$ and $i \in \{1, \ldots, n\}$ $M_{i,j}$ is a mnemonic to help the user
  associate image $I_i$ with the number $j$. $d$ is whatever base the user is familiar with
  (typically $d = 10$), and the value $b$ contains random bits which are used to select the
  secret mapping $\sigma$. ;
Generate and Memorize Secret Mapping;
for $i \leftarrow 1$ to $n$ do
    // Using random bits b
    $\sigma(i) \sim \{0, \ldots, d-1\}$ ;
    $M_i \leftarrow M_{i,\sigma(i)}$ ;
    (User) Using $M_i$ memorizes the association $(I_i, \sigma(i))$ for $i \in [n]$. ;
end
```

A.1 Formal View

We now present a formal overview of the authentication process. Algorithm 1 outlines the initialization process in which the user memorizes a secret random mapping σ generated by the user's computer, and Algorithm 2 outlines the account creation process. In Algorithm 2 the user generates the password for an account i by computing the response to a sequence of random challenges C generated by the user's computer. The sequence of challenges are stored in public memory. We assume that all steps in algorithms 1, 2 and 3 are executed on the user's local computer unless otherwise indicated. We also assume that the initialization phase (Algorithms 1 and 2) is carried out in secret (e.g., we assume that the secret mapping is chosen in secret), but we do not assume that the challenges are kept secret. We use **(User)** to denote a step completed by the human user and we use **(Server)** to denote a step completed by a third-party server. Algorithm 3 illustrates the authentication process.

A.2 Memorizing and Rehearsing σ

After the user memorizes σ he may need to rehearse parts of the mapping periodically to ensure that he does not forget it. How much effort does this require? Blocki et al.[13] introduced a usability model to estimate how much extra effort that a user would need to spend rehearsing the mapping σ. We used this model to obtain the predictions in Table 1.

Imagine that we had a program keep track of how often the user rehearsed each association $(i, \sigma(i))$ and predict how much longer the user will safely remember the association $(i, \sigma(i))$ without rehearsing again – updating this prediction after each rehearsal. The user rehearses the association $(i, \sigma(i))$ *naturally* whenever he needs to recall the value of $\sigma(i)$ while computing the password for any of his password protected accounts. If the user is in danger of forgetting the value $\sigma(i)$ then the program sends the user a reminder to rehearse (Blocki et al.[13] call this an *extra rehearsal*). Table 1 shows the value of $\mathbb{E}[ER_{365}]$, the expected number of extra rehearsals that the user will be required to do to remember the secret mapping σ during the first year. This value will depend on how often the user rehearses σ naturally. We consider four types of users: Active, Typical, Occasional and Infrequent. An Active user visits his accounts more frequently than a Infrequent user. See Appendix H for specific details about how Table 1 was computed.

Algorithm 2: CreateChallenge

```
input: $n$, $i$, $\lambda$, $d$, $b$ and $I_1, ..., I_n$.
 Generate Password Challenge for account $A_i$. $t$ is a security parameter which
 specifies how many digits the password will contain.;
for $j = 1 \rightarrow \lambda$ do
    $C_j^i \sim X_k$ ;
    // Using random bits b
end
$\vec{C}_i \leftarrow \left\langle C_1^i, \ldots, C_\lambda^i \right\rangle$ ;
Store record $\left(i, \vec{C}_i\right)$ ;
for $j = 1 \rightarrow \lambda$ do
    Load images from $C_j^i$. ;
    Display $C_j^i$ for the user. ;
    (User) Computes $q_j \leftarrow f\left(\sigma\left(C_i^j\right)\right)$ ;
end
Send $\langle q_1, \ldots, q_\lambda \rangle = f\left(\sigma\left(\vec{C}_i\right)\right)$ to server $i$;
// H is a strong cryptographic hash function
(Server $i$) Stores $h_i = \mathbf{H}\left(\vec{C}_i, \langle q_1, \ldots, q_\lambda \rangle\right)$ ;
```

Algorithm 3: Authenticate

```
input: Security parameter $\lambda$. Account $i \in [m]$. Challenges $\vec{C}_1, \ldots, \vec{C}_m$.
$\left\langle C_1^i, \ldots, C_\lambda^i \right\rangle \leftarrow \vec{C}_i$ ;
// Display Single Digit Challenges
for $j = 1 \rightarrow \lambda$ do
    (Semi-Trusted Computer) Load images from $C_j^i$.;
    (Semi-Trusted Computer) Displays $C_j^i$ to the user. ;
    (User) Computes $q_j \leftarrow f\left(\sigma\left(C_i^j\right)\right)$. ;
end
(Semi-Trusted Computer) Sends $\langle q_1, \ldots, q_\lambda \rangle$ to the server for account $i$. ;
(Server) Verifies that $\mathbf{H}\left(\vec{C}_i, \langle q_1, \ldots, q_\lambda \rangle\right) = h_i$ ;
```

Table 1 $\mathbb{E}\left[ER_{365}\right]$: Extra Rehearsals over the first year to remember σ in our scheme with $f_{2,2}$ or $f_{1,3}$. Compared with Shared Cues schemes SC-0,SC-1 and SC-2[13].

	Our Scheme ($\sigma \in \mathbb{Z}_{10}^n$)			Shared Cues		
User	$n = 100$	$n = 50$	$n = 30$	SC-0	SC-1	SC-2
Very Active	0.396	0.001	≈ 0	≈ 0	3.93	7.54
Typical	2.14	0.039	≈ 0	≈ 0	10.89	19.89
Occasional	2.50	0.053	≈ 0	≈ 0	22.07	34.23
Infrequent	70.7	22.3	6.1	≈ 2.44	119.77	173.92

Table 2 Single-Digit Challenge Layout in Scheme 1.

A	B	C	D
0	E	5	J
1	F	6	K
2	G	7	L
3	H	8	M
4	I	9	N

Table 3 Human Computable Password Challenges
n – Secret Length
m – # Challenge-Response Pairs

	Scheme 1 ($f_{2,2}$)		Scheme 2 ($f_{1,3}$)	
n	m	Winner	m	Winner
100 digits	1000	N/A	500	N/A
	500	N/A	300	N/A
	300	N/A	200	N/A
50 digits	500	N/A	300	N/A
	300	N/A	150	N/A
	150	N/A	100	N/A
30 digits	300	CSP Solver	150	N/A
	100	N/A	100	N/A
	50	N/A	50	N/A

Discussion

One of the advantages of our human computable passwords schemes is that memorization is essentially a one time cost for our Very Active, Typical and Occasional users. Once the user has memorized the mapping $\sigma : \{1, ..., n\} \to \mathbb{Z}_d$ he will get sufficient natural rehearsal to maintain this memory. In fact, our schemes require the user to expend *less* extra effort rehearsing his secret mapping than the Shared Cues password management scheme of Blocki et al. [13] (with the exception of SC-0 – the least secure Shared Cues scheme). Intuitively, this is because human computable password schemes require to recall $\sigma(i)$ for multiple different values of i to respond to each single-digit challenge C. To compute $f_{2,2}\left(\sigma\left(\{0, \ldots, 13\}\right)\right)$ the user would need to recall the values of $\sigma(10), \sigma(11), \sigma(12), \sigma(13)$ and $\sigma(j)$, where $j = \sigma(10) + \sigma(11) \mod 10$. If the user has 10 digit passwords then he will naturally rehearse the value of $\sigma(i)$ for up to fifty different values of i each time he computes one of his passwords. While the user needs to spend extra time computing his password each time he authenticates in our human computable password scheme, this extra computation time gives the user more opportunities to rehearse his secret mapping.

B Human Computable Passwords Challenge

While we provided asymptotic security bounds for our human computable password schemes in our context it is particularly important to understand the constant factors. In our context, it may be reasonable to assume that $n \leq 100$ (e.g., the user may be unwilling to memorize longer mappings). In this case it would be feasible for the adversary to execute an attack that takes time proportional to $10^{\sqrt{n}} \leq 10^{10}$. We conjecture that in practice scheme 2 ($f_{1,3}$) is slightly weaker than scheme 1 ($f_{2,2}$) when $n \leq 100$ despite the fact that $s\left(f_{2,2}\right) < s\left(f_{1,3}\right)$

because of the attack described in Remark G.2. This attack requires $\tilde{O}\left(n^{1+g(f)/2}\right)$ examples, and the running time $O\left(10^{\sqrt{n}}n^3\right)$ may be feasible for $n \leq 100$. To better understand the exact security bounds we created several public challenges for researchers to break our human computable password schemes under different parameters (see Table 3). At this time these challenges remain unsolved even after they were presented during the rump sessions at a cryptography conference and a security conference[12]. The challenges can be found at `http://www.cs.cmu.edu/~jblocki/HumanComputablePasswordsChallenge/challenge.htm`. For each challenge we selected a random secret mapping $\sigma \in \mathbb{Z}_{10}^n$, and published (1) m single digit challenge-response pairs $(C_1, f(\sigma(C_1))), \ldots, (C_m, f(\sigma(C_m)))$, where each clause C_i is chosen uniformly at random from X_k, and (2) 20 length $-$ 10 password challenges $\vec{C}_1, \ldots, \vec{C}_{20} \in (X_k)^{10}$. The goal is to guess one of the secret passwords $p_i = f\left(\sigma\left(\vec{C}_i\right)\right)$ for some $i \in [20]$.

C CSP Solver Attacks

Theorems 14 and 18 provide asymptotic security bounds (e.g., an adversary needs to see $m = \tilde{\Omega}\left(n^{s(f)}\right)$ challenge-response pairs to forge passwords). However, in our context n is somewhat small (e.g., $n \leq 100$). Thus, it is also important to address the following question: how many challenge-response pairs does the adversary need to see before it becomes feasible for the adversary to recover the secret on a modern computer? To better understand the exact security bounds of our human computable password schemes we used a Constraint Satisfaction Problem (CSP) solver to attack our scheme. We also created several public challenges to break our candidate human computable password schemes (see Table 3).

CSP Solver

Our computations were performed on a computer with a 2.83 GHz Intel Core2 Quad CPU and 4 GB of RAM. In each instance we generated a random mapping $\sigma : [n] \to \mathbb{Z}_{10}$ and m random challenge response pairs $(C, f(\sigma(C)))$ using the functions $f_{2,2}$ and $f_{1,3}$. We used the Constraint Satisfaction Problem solver from the Microsoft Solver Foundations library to try to solve for σ[12]. The results of this attack are shown in Tables 4 and 5. Due to limited computational resources we terminated each instance if the solver failed to find the secret mapping within 2.5 days, and if our solver failed to find σ in 2.5 days on and instance (n, m) we did not run the solver on strictly harder instances (e.g., (n', m') with $n' \geq n$ and $m' \leq m$). We remark that our empirical results are consistent with the hypothesis that the time/space resources consumed by the CSP solver increase exponentially in n (e.g., when we decrease n from 30 to 26 with $m = 100$ examples the CSP solver can solve for $\sigma \in \mathbb{Z}_{10}^{26}$ in 40 minutes, while the solver failed to find $\sigma \in \mathbb{Z}_{10}^{30}$ in 2.5 days and we observe similar threshold behavior in other columns of the table.).

D Statistical Dimension

At a high level our statistical dimension lower bounds closely mirror the lower bounds from [26] for binary predicates. For example, Lemmas 21, 22, 15, 23 and 25 are similar to Lemmas 2, 4, 5, 6 and 7 from [26] respectively.

[12] http://blogs.msdn.com/b/solverfoundation/ (Retrieved 9/15/2014).

Table 4 CSP Solver Attack on $f_{2,2}$
Key: UNSOLVED – Solver failed to find solution in 2.5 days; HARD – Instance is harder than an unsolved instance;

	$m = 50$	$m = 100$	$m = 300$	$m = 500$	$m = 1000$	$m = 10000$
$n = 26$	23.5 hr	40 min	4.5 hr	29 min	10 min	2 min
$n = 30$	HARD	UNSOLVED	2.33 hr	35.5 min	10 min	20 s
$n = 50$	HARD	HARD	HARD	HARD	UNSOLVED	7 hr
$n = 100$	HARD	HARD	HARD	HARD	HARD	UNSOLVED

Table 5 CSP Solver Attack on $f_{1,3}$
Key: UNSOLVED – Solver failed to find solution in 2.5 days; HARD – Instance is harder than an unsolved instance;

	$m = 50$	$m = 100$	$m = 300$	$m = 500$	$m = 1000$	$m = 10000$
$n = 26$	8.7 hr	53 min	1.33 hr	13.5 min	6.3min	2 min
$n = 30$	HARD	UNSOLVED	1 hr	41 min	2 min	15 s
$n = 50$	HARD	HARD	HARD	HARD	UNSOLVED	6.5 hr
$n = 100$	HARD	HARD	HARD	HARD	HARD	UNSOLVED

While the high level proof strategy is very similar, we stress that our lower bounds do requires new ideas because we are working with planted solutions $\sigma \in \mathbb{Z}_d^n$ instead of $\sigma \in \mathbb{Z}_d^n$. We use the basis functions χ_α where for $\alpha \in \mathbb{Z}_d^n$ is

$$\chi_\alpha(x) = \exp\left(\frac{-2\pi\sqrt{-1}\,(x\cdot\alpha)}{d}\right) .$$

Note that if $d > 2$ then the Fourier coefficients $\hat{b}_\alpha$ of a function $b : \mathbb{Z}_d^k \to \mathbb{R}$ might include complex numbers. While we need to take care to deal with the possibility that a Fourier coefficients might be complex, we are still able to apply powerful tools from Fourier analysis. For example, Parseval's identity

$$\sum_{\alpha\in\mathbb{Z}_d^k} \left|\hat{b}_\alpha\right|^2 = \mathbb{E}_{x\sim\mathbb{Z}_d^k}\left[b(x)^2\right] ,$$

still applies and there are versions of the hypercontractivity theorem [45, Chapter 10] that still apply even when Fourier coefficients are complex.

Another difference is that the reference distribution is defined over clauses and outputs $X_k \times \mathbb{Z}_d$ (instead of just clauses X_k) because we are working with a function $f : \mathbb{Z}_d^n \to \mathbb{Z}_d$ with non-binary outputs. Some care is needed in finding the right reference distribution. Unlike [26] we cannot just use the uniform distribution over $X_k \times \mathbb{Z}_d$ as a reference distribution – instead the reference distribution inherently depends on the function f (Of course it is must still be independent of σ).

The following definition will be useful in our proofs.

▶ **Definition 20.** Given a clause $C \in X_k$ and $S \subseteq [k]$ of size ℓ, we let $C_{|S} \in X_\ell$ denote the clause of variables of C at the positions with indices in S (e.g., if $C = (1,\ldots,k)$ and $S = \{1,5,k-2\}$ then $C_{|S} = (1,5,k-2) \in X_3$). Given a function $h : X_k \times \mathbb{Z}_d \to \mathbb{R}$, a clause $C_\ell \in X_\ell$ and $i \in \mathbb{Z}_d$ and a set $S \subseteq [k]$ of size $|S| = \ell$ we define

$$h_S^i(C_\ell) = \frac{|X_\ell|}{|X_k|} \sum_{C\in X_k, C_{|S}=C_\ell} h(C,i) .$$

We first show that $\Delta(\sigma, h)$ can be expressed in terms of the Fourier coefficients of $\hat{Q}$ as well as the functions h_ℓ. In particular, given a function $h : X_k \times \mathbb{Z}_d \to \mathbb{R}$ we define the degree ℓ function $b_\ell : \mathbb{Z}_d^n \to \mathbb{C}$ as follows

$$b_\ell(\sigma) \doteq \frac{1}{2} \sum_{\alpha \in \mathbb{Z}_d^\ell : H(\alpha) = \ell} \binom{k}{\ell} \sum_{i=0}^{d-1} \hat{Q}_\alpha^{f,i} \sum_{C \in X_\ell} \chi_\alpha(\sigma(C))\, h_\ell^i(C) \ .$$

Lemma 21 states that

$$\Delta(\sigma, h) = \sum_{\ell=1}^{k} \frac{1}{|X_\ell|} b_\ell(\sigma) \ .$$

This observation will be important later because we can use hypercontractivity to bound the expected value of $|b_\ell(\sigma)|$. Notice that if Q has distributional complexity r and $\ell \le r$ then $b_\ell(\sigma) = 0$ because $\hat{Q}_\alpha^{f,i} = 0$ for all $i \in \mathbb{Z}_d$ and $\alpha \in \mathbb{Z}_d^k$ s.t. $1 \le H(\alpha) \le r$. This means that first r terms of the sum in Lemma 21 will be zero.

▶ **Lemma 21.** *For every $\sigma \in \mathbb{Z}_d^k$, $j \in \mathbb{Z}_d$ and $h : X_k \to \mathbb{R}$ we have $\Delta(\sigma, h) = \sum_{\ell=1}^{k} \frac{1}{|X_\ell|} b_\ell(\sigma)$.*

Proof of Lemma 21. Before calculating we first observe that for any $j \in \mathbb{Z}_d$ we have

$$\hat{Q}_{\vec{0}}^{f,j} = \mathbb{E}_{x \sim \mathbb{Z}_d^k}\left[Q^{f,j}(x)\,\chi_\alpha(x)\right] = \mathbb{E}_{x \sim \mathbb{Z}_d^k}\left[Q^{f,j}(x)\right] = \sum_{x \in \mathbb{Z}_d^k} \frac{Q^{f,j}(x)}{d^k} = \frac{2-d}{d} \ .$$

Given $\alpha \in \mathbb{Z}_d^k$ we also define $S_\alpha \subset [k]$ to be the set of indices i s.t. $\alpha_i \neq 0$ – $|S_\alpha| = H(\alpha)$.

Now we note that

$$\begin{aligned}
\Delta(\sigma,h) &= \mathbb{E}_{C\sim X_k}\left[h\left(C, f\left(\sigma\left(C\right)\right)\right)\right] - \mathbb{E}_{(C,i)\sim T}\left[h\left(C,i\right)\right] \\
&= \sum_{C\in X_k}\left(\frac{h\big(C,f(\sigma(C))\big)}{|X_k|} - \sum_{i=0}^{d-1}\Pr_T[(C,i)]h(C,i)\right) \\
&= \sum_{C\in X_k}\left(\frac{h\big(C,f(\sigma(C))\big)}{|X_k|} - \sum_{i=0}^{d-1}\frac{\Pr_{x\sim\mathbb{Z}_d^k}[f(x)=i]}{|X_k|}h(C,i)\right) \\
&= \sum_{C\in X_k}\sum_{i=0}^{d-1}\left(\frac{h(C,i)\left(\frac{Q_\sigma^{f,i}(C)+1}{2} - \Pr_{x\sim\mathbb{Z}_d^k}[f(x)=i]\right)}{|X_k|}\right) \\
&= \sum_{C\in X_k}\sum_{i=0}^{d-1}\left(\frac{h(C,i)}{|X_k|}\left(\frac{1}{2} - \Pr_{x\sim\mathbb{Z}_d^k}[f(x)=i] + \sum_{\alpha\in\mathbb{Z}_d^k}\frac{\hat{Q}_\alpha^{f,i}\chi_\alpha\left(\sigma\left(C\right)\right)}{2}\right)\right) \\
&= \sum_{C\in X_k}\sum_{i=0}^{d-1}\left(\frac{h(C,i)}{|X_k|}\left(\frac{1}{d} - \Pr_{x\sim\mathbb{Z}_d^k}[f(x)=i] + \sum_{\alpha\in\mathbb{Z}_d^k:\alpha\neq\vec{0}}\frac{\hat{Q}_\alpha^{f,i}\chi_\alpha\left(\sigma\left(C\right)\right)}{2}\right)\right) \\
&= \sum_{C\in X_k}\sum_{i=0}^{d-1}\left(\frac{h(C,i)}{|X_k|}\left(\sum_{\alpha\in\mathbb{Z}_d^k:\alpha\neq\vec{0}}\frac{\hat{Q}_\alpha^{f,i}\chi_\alpha\left(\sigma\left(C\right)\right)}{2}\right)\right) \\
&= \sum_{C\in X_k}\sum_{i=0}^{d-1}\left(\frac{h(C,i)}{|X_k|}\left(\sum_{\ell=1}^{k}\sum_{\alpha\in\mathbb{Z}_d^k:H(\alpha)=\ell}\frac{\hat{Q}_\alpha^{f,i}\chi_\alpha\left(\sigma\left(C\right)\right)}{2}\right)\right) \\
&= \frac{1}{2\left|X_k\right|}\sum_{\ell=1}^{k}\sum_{C\in X_k}\sum_{i=0}^{d-1}\left(h(C,i)\left(\sum_{\alpha\in\mathbb{Z}_d^k:H(\alpha)=\ell}\hat{Q}_\alpha^{f,i}\chi_\alpha\left(\sigma\left(C\right)\right)\right)\right) \\
&= \frac{1}{2\left|X_k\right|}\sum_{\ell=1}^{k}\sum_{\alpha\in\mathbb{Z}_d^k:H(\alpha)=\ell}\sum_{C\in X_k}\sum_{i=0}^{d-1}\left(h(C,i)\left(\hat{Q}_\alpha^{f,i}\chi_\alpha\left(\sigma\left(C\right)\right)\right)\right) \\
&= \frac{1}{2\left|X_k\right|}\sum_{\ell=1}^{k}\sum_{\alpha\in\mathbb{Z}_d^k:H(\alpha)=\ell}\sum_{i=0}^{d-1}\hat{Q}_\alpha^{f,i}\sum_{C_\ell\in X_\ell}\sum_{C\in X_k,C_{|S_\alpha}=C_\ell}\chi_\alpha\left(\sigma\left(C\right)\right)h^i(C) \\
&= \frac{1}{2\left|X_k\right|}\sum_{\ell=1}^{k}\sum_{\alpha\in\mathbb{Z}_d^\ell:H(\alpha)=\ell}\binom{k}{\ell}\sum_{i=0}^{d-1}\hat{Q}_\alpha^{f,i}\sum_{C_\ell\in X_\ell}\sum_{C\in X_k,C_{|S_\alpha}=C_\ell}\chi_\alpha\left(\sigma\left(C_\ell\right)\right)h(C,i) \\
&= \sum_{\ell=1}^{k}\frac{1}{2\left|X_\ell\right|}\sum_{\alpha\in\mathbb{Z}_d^\ell:H(\alpha)=\ell}\binom{k}{\ell}\sum_{i=0}^{d-1}\hat{Q}_\alpha^{f,i}\sum_{C_\ell\in X_\ell}\chi_\alpha\left(\sigma\left(C_\ell\right)\right)h^i_{S_\alpha}(C) \\
&= \sum_{\ell=1}^{k}\frac{1}{|X_\ell|}b_\ell(\sigma)\ .
\end{aligned}$$

◀

The following lemma is similar to Lemma 4 from [26]. Lemma 22 is based on the general hypercontractivity theorem [45, Chapter 10] and applies to more general (non-boolean) functions.

▶ **Lemma 22.** *[45, Theorem 10.23] If $b : \mathbb{Z}_d^n \to \mathbb{R}$ has degree at most ℓ then for any $t \geq \left(\sqrt{2ed}\right)^{\ell}$,*

$$\Pr_{x\sim\mathbb{Z}_d^n}\left[|b(x)| \geq t\|b\|_2\right] \leq \frac{1}{d^{\ell}}\exp\left(-\frac{\ell}{2ed}t^{2/\ell}\right) \ ,$$

where $\|b\|_2 = \sqrt{\mathbb{E}_{x\sim\mathbb{Z}_d^n}\left[b\left(x\right)^2\right]}$

Lemma 15 and its proof are almost identical to Lemma 5 in [26]. We simply replace their concentration bounds with the concentration bounds in Lemma 22. We include the proof for completeness.

▶ **Lemma 15** (restated). *Let $b : \mathbb{Z}_d^n \to \mathbb{R}$ be any function with degree at most ℓ, and let $\mathcal{D}' \subseteq \mathbb{Z}_d^n$ be a set of assignments for which $d' = d^n/\left|\mathcal{D}'\right| \geq e^{\ell}$. Then $\mathbb{E}_{\sigma\sim\mathcal{D}'}\left[\left|b\left(\sigma\right)\right|\right] \leq 2(\ln d'/c_0)^{\ell/2}\|b\|_2$, where $c_0 = \ell\left(\frac{1}{2ed}\right)$ and $\|b\|_2 = \sqrt{\mathbb{E}_{x\sim\mathbb{Z}_d^n}\left[b\left(x\right)^2\right]}$.*

Proof of Lemma 15. The set $\mathcal{D}'$ contains $1/d'$ fraction of points in $\mathbb{Z}_d^n$. Therefore,

$$\Pr_{x\sim\mathcal{D}'}\left[|b(x)| \geq t\|b\|_2\right] \leq \frac{d'}{d^{\ell}}\exp\left(-\frac{\ell}{2ed}t^{2/\ell}\right) \ ,$$

for any $t \geq \left(\sqrt{2ed}\right)^{\ell}$. For any random variable Y and value $a \in \mathbb{R}$,

$$\mathbb{E}[Y] \leq a + \int_a^{\infty} \Pr[Y \geq t]\, dt \ .$$

We set $Y = \left|b\left(\sigma\right)\right|/\|b\|_2$ and $a = \left(\frac{\ln d'}{c_0}\right)^{\ell/2}$. Assuming that $a > \left(\sqrt{2ed}\right)^{\ell}$ we get

$$\begin{aligned}\frac{\mathbb{E}_{\sigma\sim\mathcal{D}'}[|b(\sigma)|]}{\|b\|_2} &\leq \left(\frac{\ln d'}{c_0}\right)^{\ell/2} + \int_{(\ln d'/c_0)^{\ell/2}}^{\infty} \frac{d'}{d^{\ell}} \cdot e^{-c_0 t^{2/\ell}}\, dt \\ &= (\ln d'/c_0)^{\ell/2} + \frac{\ell \cdot d'}{2d^{\ell}\cdot c_0^{\ell/2}} \cdot \int_{\ln d'}^{\infty} e^{-z} z^{\ell/2-1} dz \ .\end{aligned}$$

Let $u(i) \doteq \int_{\ln d'}^{\infty} e^{-z}z^{\ell/2-i}dz$. Applying integration by parts we have

$$\begin{aligned} u(i) \doteq \int_{\ln d'}^{\infty} e^{-z}z^{\ell/2-i}dz &= \left(e^{-z}z^{\ell/2-i+1}\right)\Big|_{\ln d'}^{\infty} + \int_{\ln d'}^{\infty} e^{-z}z^{\ell/2-i+1}dz - \left(\frac{\ell}{2}-i\right)\int_{\ln d'}^{\infty} e^{-z}z^{\ell/2-i}dz \\ &= \left(e^{-z}z^{\ell/2-i+1}\right)\Big|_{\ln d'}^{\infty} + u(i-1) - \left(\frac{\ell}{2}-i\right)u(i) \ .\end{aligned}$$

Thus,

$$u(i-1) = \left(\frac{\ell}{2}-i+1\right)u(i) - \left(e^{-z}z^{\ell/2-i+1}\right)\Big|_{\ln d'}^{\infty} = \left(\frac{\ell}{2}-i+1\right)u(i) + \frac{(\ln d')^{\frac{\ell}{2}-i+1}}{d'} \ .$$

Unrolling the recurrence for $T \geq 1$ we get

$$u(1) = \frac{(\ln d')^{\frac{\ell}{2}}}{d'} + \sum_{i=1}^{T-1}\frac{(\ln d')^{\frac{\ell}{2}-i}}{d'}\prod_{j=1}^{i}\left(\frac{\ell}{2}-j+1\right) + u\left(T+1\right)\prod_{j=1}^{T}\left(\frac{\ell}{2}-j+1\right) \ .$$

We also note that for $T = \lceil \frac{\ell}{2} + 1 \rceil$ we have $u\left(T+1\right) \prod_{j=1}^{T} \left(\frac{\ell}{2} - j + 1\right) \leq 0$. This follows because $u(T+1) \geq 0$ for any integer $T \geq 0$ and $\prod_{j=1}^{T} \left(\frac{\ell}{2} - j + 1\right) \leq 0$. It follows that

$$u(1) \leq \frac{(\ln d')^{\frac{\ell}{2}}}{d'} + \sum_{i=1}^{\lceil \frac{\ell}{2}+1 \rceil} \frac{(\ln d')^{\frac{\ell}{2}-i}}{d'} \prod_{j=1}^{i} \left(\frac{\ell}{2} - j + 1\right) \leq \frac{(\ln d')^{\frac{\ell}{2}}}{d'} \left(1 + \sum_{i=1}^{\lceil \frac{\ell}{2}+1 \rceil} \ell^{-i} \left(\frac{\ell}{2}\right)^{i}\right) \leq \frac{2\,(\ln d')^{\frac{\ell}{2}}}{d'}$$

where we used the condition $d' \geq e^{\ell}$ to obtain the second to last inequality. Now we have

$$\begin{aligned}\frac{\mathbb{E}_{\sigma\sim\mathcal{D}'}[|b(\sigma)|]}{\|b\|_2} &\leq (\ln d'/c_0)^{\ell/2} + \frac{\ell \cdot d'}{2d^{\ell} \cdot c_0^{\ell/2}} \cdot u(1) \\ &\leq (\ln d'/c_0)^{\ell/2} + \frac{\ell \cdot d'}{2d^{\ell} \cdot c_0^{\ell/2}} \cdot \left(\frac{2\,(\ln d')^{\frac{\ell}{2}}}{d'}\right) \\ &\leq 2(\ln d'/c_0)^{\ell/2} .\end{aligned}$$

◀

▶ **Lemma 23.** *Let $\mathcal{D}' \subseteq \{0, \ldots, d-1\}^n$ be a set of assignments for which $d' = d^n / |\mathcal{D}'|$. Then*

$$\mathbb{E}_{\sigma\sim\mathcal{D}'}\left[\left|\frac{1}{|X_\ell|} b_\ell\left(\sigma\right)\right|\right] \leq 2\left(\binom{k}{\ell}\sqrt{\ell! d}\right)\left(\ln d'/c_0\right)^{\ell/2} \max_{\alpha\in\mathbb{Z}_d^\ell} \sum_{i=0}^{d-1} \frac{\|h_{S_\alpha}\left(\sigma\right)\|_2}{\sqrt{|X_\ell|}} .$$

Proof. For simplicity of notation we set $b = b_\ell$. Our first goal will be to find the Fourier coefficients of

$$b(\sigma) = \frac{1}{2} \sum_{\alpha\in\mathbb{Z}_d^\ell : H(\alpha)=\ell} \binom{k}{\ell} \sum_{i=0}^{d-1} \hat{Q}_\alpha^{f,i} \sum_{C\in X_\ell} \chi_\alpha\left(\sigma\left(C\right)\right) h_{S_\alpha}^i(C) .$$

Given $\alpha = (\alpha_1, \ldots, \alpha_\ell) \in \mathbb{Z}_d^\ell$ with $H(\alpha) = \ell$ and a clause $C = (c_1, ..., c_\ell) \in X_\ell$ we define the projection $\alpha^C \in \mathbb{Z}_d^\ell$ of α onto C to be the unique vector s.t. $\alpha_{c_i}^C = \alpha_i$ for each $i \leq \ell$ and $\alpha_j = 0$ for each $j \notin \{c_1, \ldots, c_\ell\}$ – note that $H(\alpha^C) = \ell$.

Given $\alpha, \alpha' \in \mathbb{Z}_d^k$ with $H(\alpha) = H(\alpha') = \ell$ and $C, C \in X_\ell$ we say that the pairs (α, C_ℓ) and (α', C_ℓ') are equivalent if and only if their projections are equal $\alpha^C = \alpha'^{C'}$[13]. We can partition the set $\{\alpha \in \mathbb{Z}_d^k : H\left(\alpha\right) = \ell\} \times X_\ell$ into equivalence classes $E_1, \ldots, E_t$. If the pairs (α, C_ℓ) and (α', C_ℓ') are equivalent then we observe that the clauses C and C' must contain the same variables though perhaps in a different order. Furthermore, given an equivalence class E_j such that $(\alpha, C) \in E_j$ we have $(\alpha', C) \notin E_j$ for any $\alpha' \neq \alpha \in \mathbb{Z}_d^k$. Thus each equivalence class has size $\ell!$ because there are $\ell!$ ways to reorder the ℓ variables in a clause C. We can rewrite $b(\sigma)$ as

$$b(\sigma) = \frac{\binom{k}{\ell}}{2} \sum_{j=1}^{t} \sum_{(\alpha,C)\in E_j} \sum_{i=0}^{d-1} \hat{Q}_\alpha^{f,i} h_{S_\alpha}^i(C) \chi_\alpha(\sigma(C))$$

Let $(\alpha, C) \in E_j$ then the Fourier coefficient of α^C is

$$\hat{b}_j = \hat{b}_{\alpha^C} = \frac{\binom{k}{\ell}}{2} \sum_{(\alpha',C')\in E_j} \sum_{i=0}^{d-1} \hat{Q}_\alpha^{f,i} h_\ell^i(C') .$$

[13] For example, if $C = (1, 2, 5)$, $C = (1, 5, 2)$ and $\alpha = (4, 5, 6)$ and $\alpha' = (4, 6, 5)$ then the pairs (α, C_ℓ) and $\left(\alpha', C_\ell'\right)$ are equivalent.

We also note that $t = \frac{|X_\ell|(d-1)^k}{\ell!}$.

Now we can apply Parseval's identity along with the Cauchy-Schwarz inequality to obtain

$$\begin{aligned}
\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[b(\sigma)\overline{b(\sigma)}\right] &= \mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[|b(\sigma)|^2\right] \\
&= \mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[\sum_{j=1}^{t}\left|\hat{b}_j\right|^2\right] \\
&= \frac{\binom{k}{\ell}^2}{4}\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[\sum_{j=1}^{t}\left|\sum_{(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\hat{Q}_\alpha^{f,i}h_{S_\alpha}^i(C)\right|^2\right] \\
&\le \frac{\binom{k}{\ell}^2}{4}\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[\sum_{j=1}^{t}\left(\sum_{(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|\hat{Q}_\alpha^{f,i}\right|^2\right)\left(\sum_{(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|h_{S_\alpha}^i(C)\right|^2\right)\right] \\
&\le \frac{\binom{k}{\ell}^2}{4}\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n} \\
&\quad\left[\sum_{j=1}^{t}\left(\ell!\max_{j\le t,(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|\hat{Q}_\alpha^{f,i}\right|^2\right)\left(\sum_{(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|h_{S_\alpha}^i(C)\right|^2\right)\right] \\
&\le \frac{\binom{k}{\ell}^2}{4}\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n} \\
&\quad\left[\left(\ell!\max_{j\le t,(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|\hat{Q}_\alpha^{f,i}\right|^2\right)\left(\sum_{j=1}^{t}\sum_{(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|h_{S_\alpha}^i(C)\right|^2\right)\right] \\
&\le \frac{\binom{k}{\ell}^2}{4}\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n} \\
&\quad\left[\left(\ell!\max_{j\le t,(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|\hat{Q}_\alpha^{f,i}\right|^2\right)\left(\max_{\alpha\in\mathbb{Z}_d^\ell}\sum_{C\in X_\ell}\sum_{i=0}^{d-1}\left|h_{S_\alpha}^i(C)\right|^2\right)\right] \\
&\le \frac{\binom{k}{\ell}^2}{4}\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n} \\
&\quad\left[\left(\ell!\max_{j\le t,(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|\hat{Q}_\alpha^{f,i}\right|^2\right)\left(|X_\ell|\max_{\alpha\in\mathbb{Z}_d^\ell}\sum_{i=0}^{d-1}\mathbb{E}_{C\sim X_\ell}\left[h_{S_\alpha}^i(C)^2\right]\right)\right] \\
&\le \frac{\binom{k}{\ell}^2 d\ell!\,|X_\ell|}{4}\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[\left(\max_{j\le t,(C,\alpha)\in E_j}\sum_{i=0}^{d-1}\left|\hat{Q}_\alpha^{f,i}\right|^2\right)\right]\max_{\alpha\in\mathbb{Z}_d^\ell}\sum_{i=0}^{d-1}\|h_{S_\alpha}^i\|_2^2 \\
&\le \frac{\binom{k}{\ell}^2 d\ell!\,|X_\ell|}{4}\max_{\alpha\in\mathbb{Z}_d^\ell}\sum_{i=0}^{d-1}\|h_{S_\alpha}^i\|_2^2 \ .
\end{aligned}$$

Before we can apply Lemma 15 we must address a technicality. The range of $b = b_\ell$ might include complex numbers, but Lemma 15 only applies to functions b with range $\mathbb{R}$. For $c, d \in \mathbb{R}$ we adopt the notation $Im\left(c + d\sqrt{-1}\right) = d$ and $Re\left(c + d\sqrt{-1}\right) = c$. We observe that

$$\begin{aligned}
\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[b(\sigma)\overline{b(\sigma)}\right] &= \mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[Re(b(\sigma))^2 + Im(b(\sigma))^2\right] \\
&= \|Re(b)\|_2^2 + \|Im(b)\|_2^2 \ .
\end{aligned}$$

We first observe that $Re(b)$ and $Im(b)$ are both degree ℓ functions because b is a degree ℓ function.

Now we can apply Lemma 15 to get

$$\begin{aligned}
\mathbf{E}_{\sigma\sim\mathcal{D}'}\left[\left|Re\left(b\left(\sigma\right)\right)\right|\right] &\le \frac{2\left(\ln d'/c_0\right)^{\ell/2}}{d^\ell}\|Re(b)\|_2 \\
&\le \frac{2\left(\ln d'/c_0\right)^{\ell/2}}{d^\ell}\sqrt{\mathbb{E}_{\sigma\sim\mathbb{Z}_d^n}\left[b\left(\sigma\right)\overline{b\left(\sigma\right)}\right]} \\
&\le \left(\binom{k}{\ell}\sqrt{\ell!d}\right)\left(\ln d'/c_0\right)^{\ell/2}\sqrt{|X_\ell|}\max_{\alpha\in\mathbb{Z}_d^\ell}\sum_{i=0}^{d}\|h_{S_\alpha}\left(\sigma\right)\|_2 .
\end{aligned}$$

A symmetric argument can be used to bound $\mathbf{E}_{\sigma\sim\mathcal{D}'}\left[Im\left(b\left(\sigma\right)\right)\right]$. Now because

$$\left|b\left(\sigma\right)\right| \le \left|Re\left(b\left(\sigma\right)\right)\right| + \left|Im\left(b\left(\sigma\right)\right)\right| \ ,$$

it follows that

$$\begin{aligned}
\mathbf{E}_{\sigma\sim\mathcal{D}'}\left[\left|\frac{1}{|X_\ell|}b\left(\sigma\right)\right|\right] &\le 2\left(\binom{k}{\ell}\sqrt{\ell!d}\right)\left(\frac{1}{|X_\ell|}\right)\left(\left(\ln d'/c_0\right)^{\ell/2}\right)\max_{\alpha\in\mathbb{Z}_d^\ell}\sum_{i=0}^{d-1}\|h_{S_\alpha}\left(\sigma\right)\|_2\sqrt{|X_\ell|} \\
&\le 2\left(\binom{k}{\ell}\sqrt{\ell!d}\right)\left(\ln d'/c_0\right)^{\ell/2}\max_{\alpha\in\mathbb{Z}_d^\ell}\sum_{i=0}^{d-1}\frac{\|h_{S_\alpha}\left(\sigma\right)\|_2}{\sqrt{|X_\ell|}} .
\end{aligned}$$

◀

We will use Fact 24 to prove Lemma 25. The proof of Fact 24 is found in [26, Lemma 7]. We include it here for completeness.

▶ **Fact 24.** *[26] If $h : X_k\times\mathbb{Z}_d\to\mathbb{R}$ satisfies $\|h\|_2^2 = 1$ then for any $i\in\mathbb{Z}_d$, $0\le\ell\le k$ and $S\subseteq[k]$ of size $|S| = \ell$ we have $\sum_{i=0}^{d-1}\|h_S^i\|_2^2\le d$.*

Proof. First notice that for any C_ℓ, $S\subseteq[k]$ s.t $|S| = \ell$

$$\left|\left\{C\in X_k \,\middle|\, C_{|S} = C_\ell\right\}\right| = \frac{|X_k|}{|X_\ell|} .$$

By applying the definition of h_ℓ along with the Cauchy-Schwartz inequality

$$\begin{aligned}
\sum_{i=0}^{d-1}\|h_S^i\|_2^2 &= \sum_{i=0}^{d-1}\mathbb{E}_{C_\ell\sim X_\ell}\left[h_S^i\left(C_\ell\right)^2\right] \\
&= \left(\frac{|X_\ell|}{|X_k|}\right)^2\sum_{i=0}^{d-1}\mathbb{E}_{C_\ell\sim X_\ell}\left[\left(\sum_{C\in X_k, C_{|S}=C_\ell}h\left(C,i\right)\right)^2\right] \\
&\le \left(\frac{|X_\ell|}{|X_k|}\right)^2\sum_{i=0}^{d-1}\mathbb{E}_{C_\ell\sim X_\ell}\left[\frac{|X_k|}{|X_\ell|}\left(\sum_{C\in X_k, C_{|S}=C_\ell}h\left(C,i\right)^2\right)\right] \\
&\le \left(\frac{|X_\ell|}{|X_k|}\right)d\mathbb{E}_{C_\ell\sim X_\ell, i\sim\mathbb{Z}_d}\left[\left(\sum_{C\in X_k, C_{|S}=C_\ell}h\left(C,i\right)^2\right)\right] \\
&= d\mathbb{E}_{C\sim U_k}\left[h\left(C\right)^2\right] = d\|h\|_2^2 = d .
\end{aligned}$$

◀

▶ **Lemma 25.** *Let $r = r(f)$, let $\mathcal{D}'\subseteq\{0,\ldots,d-1\}^n$ be a set of secret mappings and let $d' = d^n/|\mathcal{D}'|$. Then $\kappa_2(\mathcal{D}') = O_k\left(\left(\ln d'/n\right)^{r/2}\right)$*

Proof. Let $h : X_k \to \mathbb{R}$ be any function such that $\mathbb{E}_{U_k}\left[h^2\right] = 1$. Using Lemma 21 and the definition of r,

$$\begin{aligned} |\Delta(\sigma,h)| &= \left|\sum_{\ell=r}^{k} \frac{1}{|X_\ell|} b_\ell(\sigma)\right| \\ &\leq \sum_{\ell=r}^{k} \left|\frac{1}{|X_\ell|} b_\ell(\sigma)\right| . \end{aligned}$$

We apply Lemma 23 and Fact 24 to get

$$\begin{aligned} \mathbb{E}_{\sigma\sim\mathcal{D}'}\left[|\Delta(\sigma,h)|\right] &\leq 2\sum_{\ell=r}^{k}\binom{k}{\ell}\sqrt{\ell!}d\left(\ln d'/c_0\right)^{\ell/2}\max_{\alpha\in\mathbb{Z}_d}\sum_{i=0}^{d-1}\frac{\|h^i_{S_\alpha}(\sigma)\|_2}{\sqrt{|X_\ell|}} \\ &\leq \sum_{\ell=r}^{k}\left(\binom{k}{\ell}d\sqrt{\ell!}\right)\left(\frac{2\left(\ln d'/c_0\right)^{\ell/2}}{\sqrt{|X_\ell|}}\right) \\ &\leq O_k\left(\frac{(\ln d')^{\ell/2}}{n^{r/2}}\right) . \end{aligned}$$

◀

▶ **Theorem 16** (restated). *There exists a constant $c_Q > 0$ such that for any $\epsilon > 1/\sqrt{n}$ and $q \geq n$ we have*

$$\mathrm{SDN}\left(\mathcal{Z}_{\epsilon,f}, \frac{c_Q(\log q)^{r/2}}{n^{r/2}}, 2e^{-n\cdot\epsilon^2/2}\right) \geq q ,$$

where $r = r(f)$ is the distributional complexity of f.

Proof of Theorem 16. First note that, by Chernoff bounds, for any solution $\tau \in \mathbb{Z}_d^n$ the fraction of assignments $\sigma \in \mathbb{Z}_d^n$ such that τ and σ are ϵ-correlated (e.g., $H(\sigma,\tau) \leq \frac{n(d-1)}{d} - \epsilon\cdot n$) is at most $e^{-2n\cdot\epsilon^2}$. In other words $|\mathcal{D}_\sigma| \geq \left(1 - e^{-2n\cdot\epsilon^2}\right)|\mathbb{Z}_d^n|$, where $\mathcal{D}_\sigma = \mathbb{Z}_d^n \setminus \left\{\sigma' \,\middle|\, H(\sigma,\sigma') \leq \frac{n(d-1)}{d} - \epsilon\cdot n\right\}$ Let $\mathcal{D}' \subseteq \mathcal{D}_\sigma$ be a set of distributions of size $|\mathcal{D}_\sigma|/q$. Then for $d' = d^n/|\mathcal{D}'| = q\cdot d^n/|\mathcal{D}_\sigma|$, by Lemma 25 we get

$$\begin{aligned} \kappa_2(\mathcal{D}') &= O_k\left(\frac{(\ln d')^{r/2}}{n^{r/2}}\right) && (1) \\ &= O_k\left(\frac{(\ln q)^{r/2}}{n^{r/2}}\right) , && (2) \end{aligned}$$

where the last line follows by Sterling's Approximation

$$q = d'|\mathcal{D}_\sigma|/d^n = d'|\mathcal{D}_\sigma|/d^n \approx d'c'\sqrt{\frac{d}{n}}$$

for a constant c'. The claim now follows from the definition of SDN. ◀

The proof of Theorem 14 follows from Theorem 16 and the following result of Feldman et al. [26].

▶ **Theorem 13** (restated). *[26, Theorems 10 and 12] For $\kappa > 0$ and $\eta \in (0,1)$ let $d' = \mathrm{SDN}(\mathcal{Z}_{\epsilon,f}, \kappa, \eta)$ be the statistical dimension of the distributional search problem $\mathcal{Z}_{\epsilon,f}$. Any randomized statistical algorithm that, given access to a VSTAT$\left(\frac{1}{3\kappa^2}\right)$ oracle (resp. 1-MSTAT(L)) for the distribution Q^f_σ for a secret mapping σ chosen randomly and uniformly from $\mathbb{Z}^n_d$, succeeds in finding a mapping $\tau \in \mathbb{Z}^n_d$ that is ϵ-correlated with σ with probability $\Lambda > \eta$ over the choice of distribution and internal randomness requires at least $\frac{\Lambda-\eta}{1-\eta}d'$ (resp. $\Omega\left(\frac{1}{L}\min\left\{\frac{d'(\Lambda-\eta)}{1-\eta}, \frac{(\Lambda-\eta)^2}{\kappa^2}\right\}\right)$) calls to the oracle.*

▶ **Theorem 14** (restated). *Let $\sigma \in \mathbb{Z}^n_d$ denote a secret mapping chosen uniformly at random, let Q^f_σ be the distribution over $X_k \times \mathbb{Z}_d$ induced by a function $f : \mathbb{Z}^k_d \to \mathbb{Z}_d$ with distributional complexity $r = r(f)$. Any randomized statistical algorithm that finds an assignment τ such that τ is $\left(\sqrt{\frac{-2\ln(\eta/2)}{n}}\right)$-correlated with σ with probability at least $\Lambda > \eta$ over the choice of σ and the internal randomness of the algorithm needs at least m calls to the 1-MSTAT(L) oracle (resp. VSTAT$\left(\frac{n^r}{2(\log n)^{2r}}\right)$ oracle) with $m \cdot L \geq c_1\left(\frac{n}{\log n}\right)^r$ (resp. $m \geq n^{c_1 \log n}$) for a constant $c_1 = \Omega_{k,1/(\Lambda-\eta)}(1)$ which depends only on the values k and $\Lambda - \eta$. In particular if we set $L = \left(\frac{n}{\log n}\right)^{r/2}$ then our algorithms needs at least $m \geq c_1\left(\frac{n}{\log n}\right)^{r/2}$ calls to 1-MSTAT(L).*

E Security Proofs

▶ **Theorem 19** (restated). *Let f be a function with evenly distributed output (Definition 17), let $\sigma \sim \mathbb{Z}^n_d$ denote the secret mapping, let $\epsilon > 0$ be any constant and suppose that for every $C \in X_k$ we are given labels $\ell_C \in \mathbb{Z}_d$ s.t. $\Pr_{C\sim X_k}[f(\sigma(C)) = \ell_C] \geq \frac{1}{d} + \epsilon$. There is a polynomial time algorithm (in n, m,$1/\epsilon$) that finds a mapping $\sigma' \in \mathbb{Z}^n_d$ such that σ' is $\epsilon/2$-correlated with σ with probability at least $\frac{\epsilon}{2d^2}$*

Proof of Theorem 19. Let $f : \mathbb{Z}^k_d \to \mathbb{Z}_d$ be a function with evenly distributed output. We select fix $C^{-1} \sim X_{k-1}$ and $i \sim [n] \setminus C^{-1}$. Given $j \in [n] \setminus C^{-1}$ we let $C_j = (C^{-1}, j) \in X_k$ denote the corresponding clause. Now we generate the mapping σ' by selecting $\sigma'(i)$ at random, and setting $\sigma'(j) = \sigma'(i) + \ell_{C_j} - \ell_{C_i} \mod d$ for $j \in [n] \setminus C_i$. For $j \in C^{-1}$ we select $\sigma'(j)$ at random. We let $\mathbf{GOOD}\left(C^{-1}, i, \sigma'\right)$ denote the event that

$$\Pr_{j\sim[n]\setminus C^{-1}}\left[\ell_{C_j} = f\left(\sigma\left(C_j\right)\right)\right] \geq \frac{1}{d} + \epsilon/2 \ ,$$

$\sigma'(i) = \sigma(i)$ and $\ell_{C_i} = f\left(\sigma\left(C_i\right)\right)$. Assume that the event $\mathbf{GOOD}\left(C^{-1}, i\right)$ occurs, in this case for each j s.t. $\ell_{C_j} = f\left(\sigma\left(C_j\right)\right)$ we have

$$\begin{aligned}
\sigma'(j) - \sigma(j) &\equiv \left(\sigma'(i) + \ell_{C_j} - \ell_{C_i}\right) - \sigma(j) \mod d\\
&\equiv \left(\ell_{C_j} - \ell_{C_i}\right) + \sigma(i) - \sigma(j) \mod d\\
&\equiv g\left(\sigma\left(C^{-1}\right)\right) + \sigma\left(j\right) - \left(g\left(\sigma\left(C^{-1}\right)\right) + \sigma\left(i\right)\right) + \sigma(i) - \sigma(j) \mod d\\
&\equiv 0 \mod d \ .
\end{aligned}$$

Therefore, we have

$$\frac{n - H\left(\sigma, \sigma'\right)}{n} \geq \frac{1}{d} + \epsilon/2 - \frac{k-1}{n} \ .$$

We note that by Markov's inequality the probability of success is at least

$$\Pr_{C^{-1}\sim X_{k-1}}\left[\mathbf{GOOD}\left(C^{-1}\right)\right] \geq \frac{\epsilon}{2d^2} \ . \qquad ◀$$

Before proving Theorem 18 we introduce some notation and prove an important claim. We use $\mathcal{A}_{C_1,\ldots,C_m} : (X_k)^\lambda \to \mathbb{Z}_d^\lambda$ to denote an adversary who sees the challenges $C_1,\ldots,C_m \in X_k$ and the corresponding responses $f(\sigma(C_1)),\ldots,f(\sigma(C_m))$. $\mathcal{A}_{C_1,\ldots,C_m}(C'_1,\ldots,C'_\lambda) \in \mathbb{Z}_d^\lambda$ denotes the adversaries prediction of $f(\sigma(C'_1)),\ldots,f(\sigma(C'_\lambda))$. Given a function $b : (X_k)^\lambda \to \mathbb{Z}_d^\lambda$, challenges $C'_1,\ldots,C'_\lambda \in X_k$ and responses $f(\sigma(C'_1)),\ldots,f(\sigma(C'_\lambda))$ we use $\mathcal{P}_{b,i,C'_1,\ldots,C'_m} : X_k \times [t] \to \mathbb{Z}_d \cup \{\perp\}$ to predict the value of a clause $C \in X_k$

$$\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i) = \begin{cases} b\left(\hat{C}_1,\ldots,\hat{C}_\lambda\right)[i], & \text{if } f\left(\sigma\left(\hat{C}_j\right)\right) = b\left(\hat{C}_1,\ldots,\hat{C}_\lambda\right)[j] \;\; \forall j < i \\ \perp, & \text{otherwise} \end{cases}$$

where $\hat{C}_i = C$ and $\hat{C}_j = C'_j$ for $j \neq i$. We allow our predictor $\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i)$ to output $\perp$ when it is unsure. Informally, Claim 26 says that for $b = \mathcal{A}_{C_1,\ldots,C_m}$ our predictor $\mathcal{P}_{b,i,C'_1,\ldots,C'_m}$ is reasonably accurate whenever it is not unsure. Briefly, Claim 26 follows because for $b = \mathcal{A}_{C_1,\ldots,C_m}$ we have

$$\Pr[\mathbf{Wins}(\mathcal{A},n,m,\lambda)] = \prod_{i=1}^{d} \Pr_{\substack{C\sim X_k \\ C_1,\ldots,C_m\sim X_k \\ C'_1,\ldots,C'_\lambda\sim X_k}} \left[\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i) = f(\sigma(C)) \,\middle|\, \mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i) \neq \perp\right] .$$

► **Claim 26.** *Let $\mathcal{A}$ be an adversary s.t $\Pr[\mathbf{Wins}(\mathcal{A},n,m,\lambda)] > \left(\frac{1}{d}+\epsilon\right)^\lambda$ and let $b = \mathcal{A}_{C_1,\ldots,C_m}$ then*

$$\Pr_{\substack{i\sim[\lambda],C\sim X_k \\ C_1,\ldots,C_m\sim X_k \\ C'_1,\ldots,C'_\lambda\sim X_k}} \left[\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i) = f(\sigma(C)) \,\middle|\, \mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i) \neq \perp\right] \geq \left(\frac{1}{d}+\epsilon\right) .$$

Proof of Claim 26. We draw examples $(C_1, f(\sigma(C_1))),\ldots,(C_m, f(\sigma(C_m)))$ to construct $b = \mathcal{A}_{C_1,\ldots,C_m}$. Given a random length-$\lambda$ password challenge $(C'_1,\ldots,C'_\lambda) \in (X_k)^t$ we let

$$p_j = \Pr_{C,C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda\sim X_k} \left[\mathcal{P}_{b,j,C'_1,\ldots,C'_t}(C) = f(\sigma(C)) \,\middle|\, \mathcal{P}_{b,j,C'_1,\ldots,C'_\lambda}(C) \neq \perp\right]$$

denote the probability that the adversary correctly guesses the response to the j'th challenge conditioned on the event that the adversary correctly guesses all of the earlier challenges. Observe that

$$\Pr_{C,C_1,\ldots,C_m,C'_1,\ldots,C'_{\lambda-1}\sim X_k, i\sim[t]} \left[\mathcal{P}_{b,i,C'_1,\ldots,C'_\lambda}(C,i) = f(\sigma(C))\right] = \sum_{i=1}^{\lambda} p_i/\lambda ,$$

so it suffices to show that $\sum_{i=1}^{\lambda} p_i/\lambda \geq \frac{1}{d}+\epsilon$. We obtain the following constraint

$$\begin{aligned}
\prod_{i=1}^{\lambda} p_i &= \prod_{i=1}^{\lambda} \Pr_{C,C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda\sim X_k} \left[\mathcal{P}_{b,j,C'_1,\ldots,C'_\lambda}(C) = f(\sigma(C)) \,\middle|\, \mathcal{P}_{b,j,C'_1,\ldots,C'_\lambda}(C) \neq \perp\right] \\
&= \prod_{i=1}^{\lambda} \Pr_{C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda\sim X_k} \\
&\quad \left[\mathcal{A}_{C_1,\ldots,C_m}\left(C'_1,\ldots,C'_\lambda\right)[i] = f\left(\sigma\left(C'_i\right)\right) \,\middle|\, \forall j < i.\ \mathcal{A}_{C_1,\ldots,C_m}\left(C'_1,\ldots,C'_\lambda\right)[j] = f\left(\sigma\left(C'_j\right)\right)\right] \\
&= \Pr_{C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda\sim X_k} \left[\mathcal{A}_{C_1,\ldots,C_m}\left(C'_1,\ldots,C'_\lambda\right) = \left(f\left(\sigma\left(C'_1\right)\right),\ldots,f\left(\sigma\left(C'_\lambda\right)\right)\right)\right] \\
&\geq \left(\frac{1}{d}+\epsilon\right)^\lambda .
\end{aligned}$$

If we minimize $\sum_{i=1}^{t} p_i/\lambda$ subject to the constraint $\prod_{i=1}^{\lambda} p_i \geq \left(\frac{1}{d}+\epsilon\right)^{\lambda}$ then we obtain the desired upper bound $\sum_{i=1}^{\lambda} p_i/\lambda \geq \frac{1}{d}+\epsilon$. ◀

▶ **Theorem 18** (restated)**.** *Suppose that f has evenly distributed output, but that f is not* $\mathbf{UF-RCA}\,(n,m,\lambda,\delta)$ *– secure for $\delta > \left(\frac{1}{d}+\epsilon\right)^{\lambda}$. Then there is a probabilistic polynomial time algorithm (in n, m, λ and $1/\epsilon$) that extracts a string $\sigma' \in \mathbb{Z}_d^n$ that is $\epsilon/8$-correlated with σ with probability at least $\frac{\epsilon^3}{(8d)^2}$ after seeing $m+\lambda$ example challenge response pairs.*

Proof of Theorem 18. Given random clauses $C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda \sim X_k$ we let $\mathbf{Good}\,(C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda)$ denote the event that

$$\Pr_{i\sim[t],C\sim X_k}\left[\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i)=f\left(\sigma\left(C\right)\right)\,\middle|\,\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i)\neq\perp\right]\geq\left(\frac{1}{d}+\frac{\epsilon}{2}\right).$$

By Markov's Inequality and Claim 26 we have $\Pr\left[\mathbf{Good}\,(C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda)\right]\geq\frac{\epsilon}{2}$. Here, $b=\mathcal{A}_{C_1,\ldots,C_m}$ and

$$\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,i)=\begin{cases} b\left(\hat{C}_1,\ldots,\hat{C}_\lambda\right)[i], & \text{if } f\left(\sigma\left(\hat{C}_j\right)\right)=b\left(\hat{C}_1,\ldots,\hat{C}_\lambda\right)[j]\ \ \forall j<i\\ \perp, & \text{otherwise}\end{cases}$$

Assuming that the event $\mathbf{Good}\,(C_1,\ldots,C_m,C'_1,\ldots,C'_\lambda)$ occurs we obtain labels for each clause $C\in X_k$ by selecting a random permutation $\pi:[\lambda]\to[\lambda]$, setting $i=1$ and setting $\ell_C=\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,\pi(i))$ – if $\ell_C\neq\perp$ then we increment i and repeat. Note that we will always find a label $\ell_C\neq\perp$ within t attempts because $\mathcal{P}_{b,C'_1,\ldots,C'_\lambda}(C,1)\neq\perp$. Let **GoodLabels** denote the event that

$$\Pr_{C\sim X_k}\left[G_C\right]\geq\frac{1}{d}+\frac{\epsilon}{4},$$

where G_C is the indicator random variable for the event $\ell_C=f\left(\sigma\left(C\right)\right)$. We have

$$\mathbb{E}\left[\frac{1}{|X_k|}\sum_{C\in X_k}G_C\right]\geq\frac{1}{d}+\frac{\epsilon}{2},$$

so we can invoke Markov's inequality again to argue that $\Pr\left[\mathbf{GoodLabels}\,|\,\mathbf{Good}\right]\geq\frac{\epsilon}{4}$. If the event **GoodLabels** occurs then we can invoke Theorem 19 to obtain σ' that is $\epsilon/8$-correlated with σ with probability at least $\frac{\epsilon}{8d^2}$. Our overall probability of success is

$$\frac{\epsilon}{8d^2}\times\frac{\epsilon}{4}\times\frac{\epsilon}{2}=\frac{\epsilon^3}{(8d)^2}.$$ ◀

F Security Parameters of f_{k_1,k_2}

▶ **Claim 9** (restated)**.** *Let $0\leq k_1$ and $k_2>0$ be given and let $f=f_{k_1,k_2}$ we have $g(f)=\min\{k_1,10\}$, $r(f)=k_2+1$ and $s(f)=\min\left\{\frac{k_2+1}{2},k_1+1,11\right\}$.*

Proof of Claim 9. Let $f(x)=f_{k_1,k_2}(x)=x_{\left(\sum_{i=10}^{9+k_1}x_i \mod 10\right)}+\sum_{i=10+k_1}^{9+k_1+k_2}x_i \mod 10$. We first observe that if we fix the values of $x_{10},\ldots,x_{9+k_1}\in\mathbb{Z}_{10}$ and let $i'=\sum_{i=10}^{9+k_1}x_i \mod 10$ then $f'\left(x_0,\ldots,x_9,x_{10+k_1},\ldots,x_{9+k_1+k_2}\right)=x_{i'}+\sum_{i=10+k_1}^{9+k_1+k_2}x_i \mod 10$ is a linear function. Similarly, if we fix the values of $x_0=\ldots=x_9=c$ then the resulting function

$f'(x_{10},\ldots,x_{9+k_1+k_2}) = c + \sum_{i=10+k_1}^{9+k_1+k_2} x_i \mod 10$ is linear. Thus, $g(f) \le \min\{10, k_1\}$. Now suppose that we don't fix all of the values $x_{10},\ldots,x_{9+k_1} \in \mathbb{Z}_{10}$ and at least one of the variables $x_0,\ldots,x_9$ is not fixed. In this case the resulting function will not be linear. Thus, $g(f) \ge \min\{k_1, 10\}$. We also note that for any $\alpha \in \mathbb{Z}_{10}^{10+k_1+k_2}$ s.t. $H(\alpha) \le k_2$ and $i,t \in \mathbb{Z}_{10}$ that

$$\Pr_{x\sim\mathbb{Z}_{10}^{10+k_1+k_2}}[f(x)=t \mid \alpha\cdot x \equiv i \mod 10] = \Pr_{x\sim\mathbb{Z}_{10}^{10+k_1+k_2}}[f(x)=t] = \frac{1}{10}\ .$$

Therefore,

$$\begin{aligned}
\hat{Q}_{\alpha}^{f,t} &= \mathbb{E}_{x\sim\mathbb{Z}_{10}^{10+k_1+k_2}}\left[Q^{f,t}(x)\chi_\alpha(x)\right] \\
&= \sum_{i=0}^{9}\Pr[\alpha\cdot x\equiv i \mod 10]\,\mathbb{E}_{x\sim\mathbb{Z}_{10}^{10+k_1+k_2}}\left[Q^{f,t}(x)\chi_\alpha(x)\,\middle|\,\alpha\cdot x\equiv i \mod 10\right] \\
&= \sum_{i=0}^{9}\exp\left(\frac{-2\pi i\sqrt{-1}}{10}\right)\Pr[\alpha\cdot x\equiv i \mod 10]\,\mathbb{E}_{x\sim\mathbb{Z}_{10}^{10+k_1+k_2}}\left[Q^{f,t}(x)\,\middle|\,\alpha\cdot x\equiv i \mod 10\right] \\
&= \frac{1}{10}\sum_{i=0}^{9}\exp\left(\frac{-2\pi i\sqrt{-1}}{10}\right)\mathbb{E}_{x\sim\mathbb{Z}_{10}^{10+k_1+k_2}}\left[Q^{f,t}(x)\,\middle|\,\alpha\cdot x\equiv i \mod 10\right] \\
&= 0\ ,
\end{aligned}$$

which implies that $r(f) \ge k_2+1$. Similarly, if we set $\alpha = (\alpha_0,\ldots,\alpha_{9+k_1+k_2})$ such that $\alpha_0 = 1$ and $\alpha_{10+k_1} = \ldots = \alpha_{9+k_1+k_2} = 1$ so that α has hamming weight k_2+1 then we can verify that $\hat{Q}_{\alpha}^{f,t} \neq 0$. ◀

G Security Upper Bounds

G.1 Statistical Algorithms

Theorem 27 demonstrates that our lower bound for statistical algorithms are asymptotically tight for our human computable password schemes f_{k_1,k_2}. In particular, we demonstrate that $m = \tilde{O}\left(n^{(k_2+1)/2}\right)$ queries to 1-MSTAT are sufficient for a statistical algorithm to recover σ.

▶ **Theorem 27.** *For $f = f_{k_1,k_2}$ there is a randomized algorithm that makes $O\left(n^{\max\{1,(k_2+1)/2\}}\log^2 n\right)$ calls to the 1-MSTAT$\left(n^{\lceil r(f_i)/2\rceil}\right)$ oracle and returns σ with probability $1-o(1)$.*

For binary functions $f' : \{0,1\}^k \to \{0,1\}$, Feldman et al. [26] gave a randomized statistical algorithm to find $\sigma' \in \{0,1\}^n$ using just $O\left(n^{r(f)/2}\log^2 n\right)$ calls to the 1-MSTAT $\left(n^{\lceil r(f)/2\rceil}\right)$ oracle. Their main technique is a discrete spectral iteration procedure to find the eigenvector (singular vector) with the largest eigenvalue (singular value) of a matrix M sampled from a distribution $M_{\sigma',p}$ over $\left|X_{\lfloor r(f)/2\rfloor}\right| \times \left|X_{\lceil r(f)/2\rceil}\right|$ matrices. With probability $1-o(1)$ this eigenvector will encode the value $\sum_{i\in C}\sigma'(i) \mod 2$ for each clause $C \in X_{r(f)/2}$. We show that the discrete spectral iteration algorithm of Feldman et al [26] can be extended to recover $\sigma \in \mathbb{Z}_{10}$ when f_{k_1,k_2} is one of our candidate human computable functions.

Discussion

We note that Theorem 27 cannot be extended to arbitrary functions $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$. Consider for example the unique function $f : \mathbb{Z}_{10}^6 \to \mathbb{Z}_{10}$ s.t. $f(x_1,\ldots,x_6) \equiv f'(x_1 \mod 2,\ldots,x_6 \mod 2) \mod 2$ and $f(x_1,\ldots,x_6) \equiv f''(x_1 \mod 5,\ldots,x_6 \mod 5) \mod 5$, where $f' : \mathbb{Z}_2^6 \to \mathbb{Z}_2$ and

$f'' : \mathbb{Z}_5^6 \to \mathbb{Z}_5$. By the Chinese Remainder Theorem instead of picking a secret mapping $\sigma \in \mathbb{Z}_{10}^n$ we could equivalently pick the unique secret mappings $\sigma_1 \in \mathbb{Z}_2^n$ and $\sigma_2 \in \mathbb{Z}_5^n$ s.t $\sigma \equiv \sigma_1 \mod 2$ and $\sigma \equiv \sigma_2 \mod 5$. Now drawing challenge response pairs from the distributions Q_σ^f is equivalent to drawing challenge-response pairs from the distributions $Q_{\sigma_1}^{f'}$ and $Q_{\sigma_2}^{f''}$. Suppose that $f'(x_1, \ldots, x_6) = x_1x_2+x_3+x_4+x_5+x_6 \mod 2$, and $f''(x_1, \ldots, x_6) = x_1$. Then we have $r(f) = \min\left(r(f'), r(f'')\right) = r(f'') = 1$, but $r(f') = 4$. We can find σ_2 using $O\left(n \log^2 n\right)$ calls to 1-MSTAT(n), but to find σ we must also recover σ_1. This provably requires at least $\tilde{\Omega}\left(n^{r(f')/2}\right) = \tilde{\Omega}\left(n^2\right)$ calls to 1-MSTAT $\left(n^2\right)$.

Background

The proof of Theorem 27 relies on the discrete spectral iteration algorithm of [26]. We begin by providing a brief overview of their algorithm. In their setting the secret mapping σ is defined over the binary alphabet $\mathbb{Z}_2^n$. Let $c_1 = \lceil \frac{r(f)}{2} \rceil, c_2 = \lfloor \frac{r(f)}{2} \rfloor$ and let $\delta \in [0,2]\backslash\{1\}$. They use σ to define a distribution over $|X_{c_1}| \times |X_{c_2}|$ matrices $M_{\sigma,\delta,p} = \hat{M}\left(Q_{\sigma,\delta,p}\right) - Jp$, where J denotes the all ones matrix. For $(C_1) \in X_{c_1}$, $(C_2) \in X_{c_2}$ such that $C_1 \bigcap C_2 = \emptyset$ we have

$$\hat{M}\left(Q_{\sigma,\delta,p}\right)\left[\left(C_1\right),\left(C_2\right)\right] = \begin{cases} 1, & \text{with probability } \left(p\left(2-\delta\right)\right) \text{ if } \sum_{j \in C_1 \cup C_2} \sigma\left(j\right) \equiv 0 \mod 2 \\ 1, & \text{with probability } \left(p\delta\right) \text{ if } \sum_{j \in C_1 \cup C_2} \sigma\left(j\right) \not\equiv 0 \mod 2 \\ 0, & \text{otherwise} \end{cases} .$$

Given a vector $x \in \{\pm 1\}^{\left|X_{c_2}\right|}$ (resp. $y \in \{\pm 1\}^{\left|X_{c_1}\right|}$) $M_{\sigma,\delta,p}x$ defines a distribution over vectors in $\mathbb{R}^{\left|X_{c_1}\right|}$ (resp. $M_{\sigma,p}^T y$ defines a distribution over vectors in $\mathbb{R}^{\left|X_{c_1}\right|}$).

If $r(f)$ is even then the largest eigenvalue of $\mathbb{E}\left[M_{\sigma,\delta,p}\right]$ has a corresponding eigenvector $x^* \in \{\pm 1\}^{X_{r(f)/2}}$, where for $C_i \in X_{r(f)/2}$ we have $x^*\left[C_i\right] = 1$ if $\sum_{j \in C_i} \sigma(j) \equiv 1 \mod 2$; otherwise $x^*\left[C_i\right] = -1$ (if $r(f)$ is odd then we consider the top singular value instead). Feldman et al [26] use discrete spectral iteration to find x^* (or $-x^*$). Given x^* it is easy to find σ using Gaussian Elimination.
The discrete spectral iteration algorithm of Feldman et al [26] starts with a random vector $x^0 \in \{0,1\}^{\left|X_{c_2}\right|}$. They then sample $x^{i+1} \sim M_{\sigma,p}x^i$ and execute a normalization step to ensure that $x^{i+1} \in \{0,1\}^{\left|X_{k_2}\right|}$. When $r(f)$ is odd, power iteration has two steps: draw a sample $y^i \sim M_{\sigma,\delta,p}x^i$ and sample from the distribution $x^{i+1} = M_{\sigma,\delta,p}^T y^i$. They showed that $O\left(\log\left|X_{r(f)}\right|\right)$ iterations suffice to recover σ whenever $p = \frac{K \log\left|X_{r(f)}\right|}{(\delta-1)^2\sqrt{\left|X_{r(f)}\right|}}$, and that for a vector $x \in \{0,1\}^{\left|X_{k_2}\right|}$ (resp. $y \in \{\pm 1\}^{\left|X_{k_1}\right|}$) it is possible to sample from $M_{\sigma,\delta,p}x$ (resp. $M_{\sigma,\delta,p}^T y$) using $O\left(1/p\right)$ queries to 1-MSTAT $\left(\left|X_{c_1}\right|\right)$.

Our Reduction

The proof of Theorem 27 uses a reduction to the algorithm of Feldman et al[26].

Proof of Theorem 27 (sketch). Given a mapping $\sigma \in \mathbb{Z}_d^n$ and a number $i \in \mathbb{Z}_d$ we define a mapping $\sigma_i \in \mathbb{Z}_2^n$ where

$$\sigma_i(j) = \begin{cases} 1, & \textbf{if} \sigma\left(j\right) = i \\ 0, & \text{otherwise} \end{cases} .$$

Clearly, to recover σ it is sufficient to recover σ_i for each $i \in \mathbb{Z}_d$. Therefore, to prove Theorem 27 it suffices to show that given $x \in \{\pm 1\}^{\left|X_{k_2}\right|}$ (resp. $y \in \{\pm 1\}^{\left|X_{k_1}\right|}$) we can sample from the distribution $M_{\sigma_i,\delta,p}x$ (resp. $M^T_{\sigma_i,\delta,p}y$) using $O(1/p)$ queries to 1-MSTAT $\left(\left|X_{\lceil r(f)/2\rceil}\right|\right)$ for each $i \in \{0,\ldots,d-1\}$, where 1-MSTAT uses the distribution Q^f_σ. In general, this will not possible for arbitrary functions f. However, Lemma 28 shows that for our candidate human computable functions $f_{1,3}, f_{2,2}$ we can sample from the distributions $M_{\sigma_i,\delta,p}x$ (resp. $M^T_{\sigma_i,\delta,p}y$). The proof of Lemma 28 is similar to the proof of [26, Lemma 10]. ◀

▶ **Lemma 28.** *Given vectors $\vec{x} \in \{\pm1\}^{|X_{c_1}|}, \vec{y} \in \{\pm1\}^{|X_{c_2}|}$ we can sample from $M_{\sigma_i,\delta,p}x$ and $M^T_{\sigma,\delta,p}y$ using $O\left(n^{(k_2+1)/2}\log^2 n\right)$ calls to the 1-MSTAT$\left(n^{\lceil r(f)/2\rceil}\right)$ oracle for $f = f_{k_1,k_2}$.*

The proof of Lemma 28 relies on Fact 29.

▶ **Fact 29.** *For each $j,t \in \mathbb{Z}_{10}$ we have*

$$\Pr_{\left(x_0,\ldots,x_{9+k_1+k_2}\right)\sim\mathbb{Z}_{10}^{10+k_1+k_2}}\left[x_t + \sum_{i=10+k_1}^{10+k_1+k_2} x_i \equiv j \,\middle|\, f_{k_1,k_2}\left(x_0,\ldots,x_{9+k_1+k_2}\right) \equiv j \mod 10\right] = \frac{\left(\frac{9}{10}\left(\frac{1}{10}\right)+\frac{1}{10}\right)\left(\frac{1}{10}\right)}{\left(\frac{1}{10}\right)} = \frac{19}{100}\ ,$$

and

$$\Pr_{\left(x_0,\ldots,x_{9+k_1+k_2}\right)\sim\mathbb{Z}_{10}^{10+k_1+k_2}}\left[x_t + \sum_{i=10+k_1}^{10+k_1+k_2} x_i \equiv j \,\middle|\, f_{k_1,k_2}\left(\sigma\left(x_0,\ldots,x_{9+k_1+k_2}\right)\right) \not\equiv j \mod 10\right] = \frac{\left(\frac{9}{10}\left(\frac{1}{10}\right)+\frac{1}{10}(0)\right)\left(\frac{1}{10}\right)}{\left(\frac{1}{10}\right)} = \frac{9}{100}\ .$$

Proof of Lemma 28. Given a value $j \in \mathbb{Z}_{10}$ and a value $i \in \mathbb{Z}_{10}$ we let $x^i_j \in \{0,1\}$ be the indicator variable for the event $x_j = i$. By Fact 29 it follows that

$$\begin{aligned}&\Pr_{\left(x_0,\ldots,x_{k_1+k_2+9}\right)\sim\mathbb{Z}_{10}^{k_1+k_2+10}}\\ &\left[x^i_0 + x^i_{9+k_1} + \ldots + x^i_{7+k_1+c_1} \equiv 1 \mod 2 \,\middle|\, f_{k_1,k_2}\left(\sigma\left(x_0,\ldots,x_{9+k_1+k_2}\right)\right) \equiv ic_1 \mod 10\right]\\ \neq\ &\Pr_{\left(x_0,\ldots,x_{k_1+k_2+9}\right)\sim\mathbb{Z}_{10}^{k_1+k_2+10}}\\ &\left[x^i_0 + x^i_{9+k_1} + \ldots + x^i_{7+k_1+c_1} \equiv 1 \mod 2 \,\middle|\, f_{k_1,k_2}\left(\sigma\left(x_0,\ldots,x_{9+k_1+k_2}\right)\right) \not\equiv ic_1 \mod 10\right]\ .\end{aligned}$$

Now for f_{k_1,k_2} we define the function $h^{i,+} : X_{k_1+k_2+10} \times \mathbb{Z}_{10} \to X_{c_1} \cup \{\perp\}$ as follows

$$\begin{aligned}&h^{i,+}\left(x_0,\ldots,x_{9+k_1+k_2}, f_{k_1,k_2}\left(\sigma\left(x_0,\ldots,x_{9+k_1+k_2}\right)\right)\right)\\ &\qquad= \begin{cases}\left(x_0, x_{9+k_1},\ldots,x_{7+k_1+c_1}\right) & \text{if} f_{k_1,k_2}\left(\sigma\left(x_0,\ldots,x_{13}\right)\right) \equiv ic_1 \mod 10\\ \perp & \text{otherwise.}\end{cases}\ .\end{aligned}$$

Intuitively, given a clause $C_1 \in X_{c_1}$ the probability that $h^{i,+}$ returns C_1 is greater if $\sum_{j\in C_1}\sigma_i(j) \equiv 1 \mod 2$.

Given a vector $x \in \{\pm1\}^{|X_{c_1}|}$ we query our 1-MSTAT $(|X_{c_1}|+1)$ oracle $\lceil 10/p\rceil$ times with the function $h^{i,+}$ to sample from $M_{\sigma_i,\delta,p}x$. Let $q_1,\ldots,q_{\lceil 10/p\rceil} \in X_{c_1}$ denote the responses and let $x\left[q_j\right]$ denote the value of the vector x at index q_j. We observe that for some $\delta \neq 1$ we have

$$M_{\sigma_i,\delta,p}x[C] \sim \sum_{\substack{j\in\lceil 10/p\rceil\\ q_j\neq\perp}} x\left[q_j\right] - p\sum_{C'\in X_{c_1}} x[C']\ ,$$

for every $C \in X_{c_2}$. ◀

Algorithm 4: GaussianAttack

```
input : Clauses $C_1, \ldots, C_m \sim X_k$, and labels $f(\sigma(C_1)), \ldots, f(\sigma(C_m))$.
forall $S \in X_{g(f)}$, $\alpha \in \mathbb{Z}_d^{g(f)}$ do
    LC $\leftarrow \emptyset$ ;
    // LC is the set of linear constraints extracted
    forall $C \in \{C_1, \ldots, C_m\}$ do
        LC $\leftarrow$ LC $\bigcup$ TryExtract $(C, f(\sigma(C)), S, \alpha)$ ;
        if $|$LC$| \geq n$ then
            $\sigma' \leftarrow$ LinearSolve $(LC)$ ;
            if $\forall i \in [m].\ f(\sigma'(C_i)) = f(\sigma(C_i)) \in C$ then
                return $\sigma'$
            end
        end
    end
end
```

Open Question

Can we precisely characterize the functions $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$ for which we can efficiently recover σ after seeing $\tilde{O}\left(n^{r(f)/2}\right)$ challenge-response pairs? Feldman et al. [26] gave a statistical algorithm that recovers the secret mapping whenever $d = 2$ after making $\tilde{O}\left(n^{r(f)/2}\right)$ queries to 1-MSTAT $\left(n^{r(f)/2}\right)$. While we show that the same algorithm can be used to recover σ after making $\tilde{O}\left(n^{r(f)/2}\right)$ queries to 1-MSTAT $\left(n^{r(f)/2}\right)$ in our candidate human computable password schemes with $d = 10$, we also showed that these results do not extend to all functions $f : \mathbb{Z}_d^k \to \mathbb{Z}_d$.

G.2 Gaussian Elimination

Most known algorithmic techniques can be modeled within the statistical query framework. Gaussian Elimination is a notable exception. As an example consider the function $f(x_1, \ldots, x_7) = x_1 + \ldots + x_7 \mod 10$ (in this example $r(f) = 7$ and $g(f) = 0$). Our previous results imply that any statistical algorithm would need to see at least $m = \tilde{\Omega}\left(n^{7/2}\right)$ challenge response pairs $(C, f(\sigma(C)))$ to recover σ. However, it is trivial to recover σ from $O(n)$ random challenge response pairs using Gaussian Elimination. In general, consider the following attacker shown in algorithm 4, which uses Gaussian Elimination. Algorithm 4 relies on the subroutine **TryExtract** $(C, f(\sigma(C)), S, \alpha)$, which attempts to extract a linear constraint from $(C, f(\sigma(C)))$ under the assumption that $\sigma(S) = \alpha$. We assume **TryExtract** $(C, f(\sigma(C)), S, \alpha)$ returns $\emptyset$ if it cannot extract a linear constraint. For example, if we assume that $\sigma(1) = 4$ and $\sigma(2) = 8$ and let $C = (i_0, i_1, ..., i_9, 1, 2, i_{10}, i_{11})$ (with $i_j \in [n] \setminus \{1, 2\}$) then we have $f_{2,2}(\sigma(C)) = \sigma\left(i_{4+8 \mod 10}\right) + \sigma(i_{10}) + \sigma(i_{11}) \mod 10$. In this case, **TryExtract** $(C, f(\sigma(C)), \{1, 2\}, \{4, 8\})$ would return the constraint $f(\sigma(C)) = \sigma(i_2) + \sigma(i_{10}) + \sigma(i_{11}) \mod 10$. However, **TryExtract** $(C, f(\sigma(C)), \{i_0, 2\}, \{4, 8\})$ would return $\emptyset$.

Fact 30 says that an attacker needs at least $m = \tilde{\Omega}\left(n^{1+g(f)}\right)$ challenge-response pairs to recover σ using Gaussian Elimination. This is because the probability that **TryExtract** $(C, f(\sigma(C))\, S, \alpha)$ extracts a linear constraint is at most $O\left(\left(\frac{|S|}{n}\right)^{-g(f)}\right)$, which is $O\left(n^{-g(f)}\right)$ for $|S|$ constant. The adversary needs $O(n)$ linearly independent constraints to run Gaussian Elimination. If the adversary can see at most $\tilde{O}\left(n^{s(f)}\right)$ examples neither approach (Statistical Algorithms or Gaussian Elimination) can be used to recover σ.

▶ **Fact 30.** *Algorithm 4 needs to see at least $m = \tilde{\Omega}\left(n^{1+g(f)}\right)$ challenge-response pairs to recover σ.*

Remark G.2 explores the tradeoff between the adversary's running time and the number of challenge-response pairs that an adversary would need to see to recover σ using Gaussian elimination. In particular the adversary can recover σ from $\tilde{O}\left(n^{1+g(f)/2}\right)$ challenge-response pairs if he is willing to increase his running time by a factor of $d^{\sqrt{n}}$. In practice, this attack may be reasonable for $n \leq 100$ and $d = 10$, which means that it may be beneficial to look for candidate human computable functions f that maximize $\min\{r(f)/2, 1 + g(f)/2\}$ instead of $s(f)$ whenever $n \leq 100$.

▶ Remark. If the adversary correctly guesses value of $\sigma(S)$ for $|S| = n^{\epsilon}$ then he may be able to extract a linear constraint from a random example with probability $\Omega(1/n^{(1-\epsilon)g(f)})$. The adversary would only need $\tilde{O}\left(n^{1+(1-\epsilon)g(f)}\right)$ examples to solve for σ, but his running time would be proportional to $d^{\epsilon n}$ – the expected number of guesses before he is correct.

H Rehearsal Model

In this section we review the usability model of Blocki et al. [13]. Their usability model estimates the 'extra effort' that a user needs to expend to memorize and rehearse all of his secrets for a password management scheme. In this section we use $(\hat{c}, \hat{a})$ to denote a cue-association pair, and we use the variable t to denote time (days). In our context $(\hat{c}, \hat{a})$ might denote the association between a letter (e.g., 'e') and the secret digit associated with that letter (e.g., $\sigma(e)$). If the user does not rehearse an association $(\hat{c}, \hat{a})$ frequently enough then the user might forget it. Their are two main components to their usability model: rehearsal requirements and visitation schedules. Rehearsal requirements specify how frequently a cue-association pair must be used for a user to remember the association. Visitation schedules specify how frequently the user authenticates to each of his accounts and rehearses any cue-association pairs that are linked with the account.

H.1 Rehearsal Requirements

Blocki et al. [13] introduce a rehearsal schedule to ensure that the user remembers each cue-association pair.

▶ **Definition 31.** [13] A rehearsal schedule for a cue-association pair $(\hat{c}, \hat{a})$ is a sequence of times $t_0^{\hat{c}} < t_1^{\hat{c}} < \ldots$. For each $i \geq 0$ we have a *rehearsal requirement*, the cue-association pair must be rehearsed at least once during the time window $\left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right) = \{x \in \mathbb{R} \mid t_i^{\hat{c}} \leq x < t_{i+1}^{\hat{c}}\}$.

A rehearsal schedule is *sufficient* if a user can maintain the association $(\hat{c}, \hat{a})$ by following the rehearsal schedule. The length of each interval $\left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right)$ may depend on the strength of the mnemonic techniques used to memorize and rehearse a cue-association pair $(\hat{c}, \hat{a})$ as well as i – the number of prior rehearsals [54, 52].

Expanding Rehearsal Assumption [13]: The rehearsal schedule given by $t_i^{\hat{c}} = 2^{i\mathbf{s}}$ is sufficient to maintain the association $(\hat{c}, \hat{a})$, where $\mathbf{s} > 0$ is a constant.

Table 6 Visitation Schedules - number of accounts visited with frequency λ (visits/days)

Schedule | λ	$\frac{1}{1}$	$\frac{1}{3}$	$\frac{1}{7}$	$\frac{1}{31}$	$\frac{1}{365}$
Very Active	10	10	10	10	35
Typical	5	10	10	10	40
Occasional	2	10	20	20	23
Infrequent	0	2	5	10	58

H.2 Visitation Schedules

Suppose that the user has m accounts $A_1, \ldots, A_m$. A visitation schedule for an account A_i is a sequence of real numbers $\tau_0^i < \tau_1^i < \ldots$, which represent the times when the account A_i is visited by the user. Blocki et al. [13] do not assume that the exact visitation schedules are known a priori. Instead they model visitation schedules using a random process with a known parameter λ_i based on $E\left[\tau_{j+1}^i - \tau_j^i\right]$ – the average time between consecutive visits to account A_i.

A rehearsal requirement $\left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right)$ can be satisfied naturally if the user visits a site A_j that uses the cue $\hat{c}$ ($\hat{c} \in c_j$) during the given time window. Here, c_j denote the set of cue-association pairs that the user must remember when logging into account A_j. Formally,

▶ **Definition 32.** [13] We say that a rehearsal requirement $\left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right)$ is *naturally satisfied* by a visitation schedule $\tau_0^i < \tau_1^i < \ldots$ if $\exists j \in [m], k \in \mathbb{N}$ s.t $\hat{c} \in c_j$ and $\tau_k^j \in \left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right)$. We use

$$ER_{t,\hat{c}} = \left|\left\{ i \;\middle|\; t_{i+1}^{\hat{c}} \le t \wedge \forall j,k. \left(\hat{c} \notin c_j \vee \tau_k^j \notin \left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right)\right)\right\}\right| \ ,$$

to denote the number of rehearsal requirements that are not naturally satisfied by the visitation schedule during the time interval $[0, t]$.

▶ **Example.** Consider the human computable function $f_{2,2}$ from section 3, and suppose that the user has to compute $f_{2,2}\left(\sigma\left(C_i\right)\right)$ to authenticate at account A_j, where $C_i = (x_0, \ldots, x_{13})$. When the user computes $f_{2,2}$ he must rehearse the associations $(x_{10}, \sigma(x_{10}))$, $(x_{11}, \sigma(x_{11}))$, $(x_{12}, \sigma(x_{12}))$, $(x_{13}, \sigma(x_{13}))$ and $(x_i, \sigma(x_i))$ where $i = (\sigma(x_{10}) + \sigma(x_{11}) \mod 10)$. Thus $c_j \supset \{x_i, x_{10}, x_{11}, x_{12}, x_{13}\}$. When user authenticates he naturally rehearses each of these associations in c_j.

If a cue-association pair $(\hat{c}, \hat{a})$ is not rehearsed naturally during the interval $\left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right)$ then the user needs to perform an extra rehearsal to maintain the association. Intuitively, $ER_{t,\hat{c}}$ denotes the total number of extra rehearsals of the cue-association pair $(\hat{c}, \hat{a})$ during the time interval $[0, t]$, and $ER_t = \sum_{\hat{c} \in C} ER_{t,\hat{c}}$ denotes the total number of extra rehearsals during the time interval $[0, t]$ to maintain all of the cue-association pairs. Thus, a smaller value of $E\left[ER_t\right]$ indicates that the user needs to do less extra work to rehearse his secret mapping.

Poisson Arrival Process

The visitation schedule for each account A_j is given by a Poisson arrival process with parameter λ_j, where $1/\lambda_j = E\left[\tau_{j+1}^i - \tau_j^i\right]$ denotes the average time between consecutive visits to account A_j.

Table 7 $\mathbb{E}[ER_{365}]$: Extra Rehearsals over the first year to remember $\sigma : \{1, \ldots, n\} \to \mathbb{Z}_{10}$ in our scheme with $f_{2,2}$ or $f_{1,3}$. Compared with Shared Cues schemes SC-0,SC-1 and SC-2[13].

	Our Scheme ($\sigma \in \mathbb{Z}_{10}^n$)			Shared Cues		
User	$n = 100$	$n = 50$	$n = 30$	SC-0	SC-1	SC-2
Very Active	0.396	0.001	≈ 0	≈ 0	3.93	7.54
Typical	2.14	0.039	≈ 0	≈ 0	10.89	19.89
Occasional	2.50	0.053	≈ 0	≈ 0	22.07	34.23
Infrequent	70.7	22.3	6.1	≈ 2.44	119.77	173.92

Table 8 Single-Digit Challenge Layout. Given a random mapping σ from letters to digits the user can compute $f_{2,2}(\sigma(`C'))$ by executing the following steps (1) Recall $\sigma(`A')$ – the number associated with the letter A, (2) Recall $\sigma(`B')$, (3) Compute $i = \sigma(`A') + \sigma(`B') \mod 10$ – without loss of generality suppose that $i = 8$, (4) Find the letter at index $i$ – 'M' if $i = 8$, (5) Recall $\sigma(`M')$ (6) Recall $\sigma(`C')$ (7) Compute $j = \sigma(`M') + \sigma(`C') \mod 10$ (8) Recall $\sigma(`D')$ (9) Return $j + \sigma(`D')$ mod 10.

A	B	C	D
0	E	5	J
1	F	6	K
2	G	7	L
3	H	8	M
4	I	9	N

Evaluating Usability

Blocki et al. [13] prove the following theorem. Given a sufficient rehearsal schedule and a visitation schedule, Theorem 33 predicts the value of ER_t, the total number of extra rehearsals that a user will need to do to remember all of the cue-association pairs required to reconstruct all of his passwords.

▶ **Theorem 33.** *[13] Let* $i_{\hat{c}}* = \left(\arg\max_x t_x^{\hat{c}} < t\right) - 1$ *then*

$$E[ER_t] = \sum_{\hat{c} \in C} \sum_{i=0}^{i_{\hat{c}}*} \exp\left(- \left(\sum_{\substack{j \ s.t. \\ \hat{c} \in c_j}} \lambda_j \right) \left(t_{i+1}^{\hat{c}} - t_i^{\hat{c}} \right) \right)$$

We use the formula from Theorem 33 to obtain the usability results in Table 7. To evaluate this formula we need to be given the rehearsal requirements, a visitation schedule (λ_i) for each account A_i and a set of public challenges $\vec{C_i} \in (X_{14})^{10}$ for each account A_i. The rehearsal requirements are given by the Expanding Rehearsal Assumption [13] (we use the same association strength parameter $\mathbf{s} = 1$ as Blocki et al. [13]), and the visitation schedules for each user are given in Table 6. We assume that each password is 10 digits long and that the challenges $\vec{C_i} \in (X_{14})^{10}$ are chosen at random by Algorithm 2. Notice that each time the user responds to a single digit challenge he rehearses the secret mapping at five locations (see discussion in Section 3.1). Because the value of $\mathbb{E}[ER_{365}]$ depends on the particular password challenges that we generated for each account, we ran Algorithm 2 and computed the resulting value $\mathbb{E}[ER_{365}]$ one-hundred times. The values in Table 7 represent the mean value of $\mathbb{E}[ER_{365}]$ across all hundred instances.

I Sum of k-Mins

In the basic Hopper-Blum [34] Human Identification Protocol the user memorizes a subset $S \subseteq [n]$ of $k = |S|$ secret indices. A single digit challenge consisted of a vector $x \in \mathbb{Z}_{10}^n$ of n digits and the user responded by with the $\mod 10$ sum of the digits at $k \leq n$ secret locations plus an error term e

$$\sum_{i \in S} x_i \mod 10 + e \ .$$

Typically, the user will set $e = 0$, but occasionally the user is supposed to respond with a completely random digit instead of the correct response (e.g., so that the adversary cannot simply use Gaussian Elimination to find S). Thus, the human user must occasionally generate random numbers in his head to execute the Hopper-Blum protocol. This is potentially problematic because humans are not good at consciously generating random numbers [53, 27, 42]. In fact, hard learning problems like noisy parity might even be easy to learn when humans are providing the source of noise.

Hopper and Blum [34] also proposed a deterministic human identification protocol call sum of k-mins. In this protocol the user memorized k secret pairs (i, j) of indices $S \subseteq [n]^2$. As before a single digit challenge consists of a vector $x \in \mathbb{Z}_{10}^n$ of n digits. However, now the response to the challenge is deterministic

$$\sum_{(i,j) \in S} \min\{x_i x_j\} \mod 10 \ .$$

We observe that for any constant k the protocol is not secure against polynomial time attackers who have seen $O\big(k \cdot \log n\big)$ examples. The adversary can simply enumerate all possible sets S of k pairs and cross out the ones that are inconsistent with the challenge-response pairs he has already seen. Even for larger k (e.g., greater human work) Hopper and Blum [34] observed that the protocol was not secure against an adversary who has seen $O(n^2)$ examples. To see this observe that we can create an indicator variable $y_{i,j}$ for each pair (i, j). Each challenge response pair (x, r) yields a linear constraint

$$\sum_{(i,j)} y_{i,j} \min\{x_i, x_j\} = r \mod 10 \ .$$