Naturally Rehearsing Passwords[⋆]

Jeremiah Blocki, Manuel Blum, and Anupam Datta

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
jblocki@cs.cmu.edu, mblum@cs.cmu.edu, danupam@cmu.edu

Abstract. We introduce quantitative usability and security models to guide the design of *password management schemes* — systematic strategies to help users create and remember multiple passwords. In the same way that security proofs in cryptography are based on complexity-theoretic assumptions (e.g., hardness of factoring and discrete logarithm), we quantify usability by introducing *usability assumptions.* In particular, password management relies on assumptions about human memory, e.g., that a user who follows a particular rehearsal schedule will successfully maintain the corresponding memory. These assumptions are informed by research in cognitive science and can be tested empirically. Given rehearsal requirements and a user's visitation schedule for each account, we use the total number of extra rehearsals that the user would have to do to remember all of his passwords as a measure of the usability of the password scheme. Our usability model leads us to a key observation: password reuse benefits users not only by reducing the number of passwords that the user has to memorize, but more importantly by increasing the natural rehearsal rate for each password. We also present a security model which accounts for the complexity of password management with multiple accounts and associated threats, including online, offline, and plaintext password leak attacks. Observing that current password management schemes are either insecure or unusable, we present Shared Cues — a new scheme in which the underlying secret is strategically shared across accounts to ensure that most rehearsal requirements are satisfied naturally while simultaneously providing strong security. The construction uses the Chinese Remainder Theorem to achieve these competing goals.

Keywords: Password Management Scheme, Security Model, Usability Model, Chinese Remainder Theorem, Sufficient Rehearsal Assumption, Visitation Schedule.

1 Introduction

A typical computer user today manages passwords for many different online accounts. Users struggle with this task—often forgetting their passwords or

[⋆] This work was partially supported by the NSF Science and Technology TRUST and the AFOSR MURI on Science of Cybersecurity. The first author was also partially supported by an NSF Graduate Fellowship.

adopting insecure practices, such as using the same password for multiple accounts and selecting weak passwords [33,30,39,24]. While there are many articles, books, papers and even comics about selecting strong individual passwords [29,42,35,61,55,27,48,3], there is very little work on *password management schemes*—systematic strategies to help users create and remember multiple passwords—that are both usable and secure. In this paper, we present a rigorous treatment of password management schemes. Our contributions include a formalization of important aspects of a usable scheme, a quantitative security model, and a construction that provably achieves the competing security and usability properties.

Usability Challenge. 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., accessible to an adversary). In contrast, a user's associative memory is private but lossy—if the user does not rehearse a memory it may be forgotten. While our understanding of human memory is incomplete, it has been an active area of research [17] and there are many mathematical models of human memory [37,59,14,40,56]. These models differ in many details, but they all model an associative memory with cue-association pairs: to remember $\hat{a}$ (e.g., a password) the brain associates the memory with a context $\hat{c}$ (e.g., a public hint or cue); such associations are strengthened by rehearsal . A central challenge in designing usable password schemes is thus to create associations that are strong and to maintain them over time through rehearsal. Ideally, we would like the rehearsals to be *natural*, i.e., they should be a side-effect of users' normal online activity. Indeed insecure password management practices adopted by users, such as reusing passwords, improve usability by increasing the number of times a password is naturally rehearsed as users visit their online accounts.

Security Challenge. Secure password management is not merely a theoretical problem—there are numerous real-world examples of password breaches [2,30,20,7,51,12,5,9,8,11,10]. Adversaries may crack a weak password in an *online attack* where they simply visit the online account and try as many guesses as the site permits. In many cases (e.g., Zappos, LinkedIn, Sony, Gawker [12,5,8,7,20,11]) an adversary is able to mount an *offline attack* to crack weak passwords after the cryptographic hash of a password is leaked or stolen. To protect against an offline attack, users are often advised to pick long passwords that include numbers, special characters and capital letters [48]. In other cases even the strongest passwords are compromised via a *plaintext password leak attack* (e.g., [4,9,51,10]), for example, because the user fell prey to a phishing attack or signed into his account on an infected computer or because of server misconfigurations. Consequently, users are typically advised against reusing the same password. A secure password management scheme must protect against all these types of breaches.

Contributions. We precisely define the password management problem in Section 2. A password management scheme consists of a *generator*—a function that outputs a set of public cue-password pairs—and a *rehearsal schedule.* The generator is implemented using a computer program whereas the human user is expected to follow the rehearsal schedule for each cue. This division of work is critical—the computer program performs tasks that are difficult for human users (e.g., generating random bits) whereas the human user's associative memory is used to store passwords since the computer's persistent memory is accessible to the adversary.

Quantifying Usability. In the same way that security proofs in cryptography are based on complexity-theoretic assumptions (e.g., hardness of factoring and discrete logarithm), we quantify usability by introducing *usability assumptions.* In particular, password management relies on assumptions about human memory, e.g., that a user who follows a particular rehearsal schedule will successfully maintain the corresponding memory. These assumptions are informed by research in cognitive science and can be tested empirically. Given rehearsal requirements and a user's visitation schedule for each account, we use the total number of extra rehearsals that the user would have to do to remember all of his passwords as a measure of the usability of the password scheme (Section 3). Specifically, in our usability analysis, we use the *Expanding Rehearsal Assumption (ER)* that allows for memories to be rehearsed with exponentially decreasing frequency, i.e., rehearse at least once in the time-intervals (days) $[1,2)$, $[2,4)$, $[4,8)$ and so on. Few long-term memory experiments have been conducted, but *ER* is consistent with known studies [53,60]. Our memory assumptions are parameterized by a constant σ which represents the strength of the mnemonic devices used to memorize and rehearse a cue-association pair. Strong mnemonic techniques [52,34] exploit the associative nature of human memory discussed earlier and its remarkable visual/spatial capacity [54].

Quantifying Security. We present a game based security model for a password management scheme (Section 4) in the style of exact security definitions [18]. The game is played between a user ($\mathcal{U}$) and a resource-bounded adversary ($\mathcal{A}$) whose goal is to guess one of the user's passwords. Our game models three commonly occurring breaches (online attack, offline attack, plaintext password leak attack).

Our Construction. We present a new password management scheme, which we call Shared Cues, and prove that it provides strong security and usability properties (see Section 5). Our scheme incorporates powerful mnemonic techniques through the use of public cues (e.g., photos) to create strong associations. The user first associates a randomly generated person-action-object story (e.g., Bill Gates swallowing a bike) with each public cue. We use the Chinese Remainder Theorem to share cues across sites in a way that balances several competing security and usability goals: 1) Each cue-association pair is used by many different web sites (so that most rehearsal requirements are satisfied naturally), 2) the total number of cue-association pairs that the user has to memorize is low, 3) each web site uses several cue-association pairs (so that passwords are secure) and 4) no two web sites share too many cues (so that passwords remain secure even after the adversary obtains some of the user's other passwords). We show that our construction achieves an asymptotically optimal balance between these security and usability goals (Lemma 2, Theorem 3).

Related Work. A distinctive goal of our work is to quantify usability of password management schemes by drawing on ideas from cognitive science and leverage this understanding to design schemes with acceptable usability. We view the results of this paper–employing usability assumptions about rehearsal requirements—as an initial step towards this goal. While the mathematical constructions start from the usability assumptions, the assumptions themselves are empirically testable, e.g., via longitudinal user studies. In contrast, a line of prior work on usability has focused on empirical studies of user behavior including their password management habits [33,30,39], the effects of password composition rules (e.g., requiring numbers and special symbols) on individual passwords [38,22], the memorability of individual system assigned passwords [50], graphical passwords [28,19], and passwords based on implicit learning [23]. These user studies have been limited in duration and scope (e.g., study retention of a single password over a short period of time). Other work [25] articulates informal, but more comprehensive, usability criteria for password schemes.

Our use of cued recall is driven by evidence that it is much easier than pure recall [17]. We also exploit the large human capacity for visual memory [54] by using pictures as cues. Prior work on graphical passwords [28,19] also takes advantage of these features. However, our work is distinct from the literature on graphical passwords because we address the challenge of managing multiple passwords. More generally, usable and secure password management is an excellent problem to explore deeper connections between cryptography and cognitive science.

Security metrics for passwords like (partial) guessing entropy (e.g., how many guesses does the adversary need to crack α-fraction of the passwords in a dataset [41,44,24]? how many passwords can the adversary break with β guesses per account [26]?) were designed to analyze the security of a dataset of passwords from many users, not the security of a particular user's password management scheme. While these metrics can provide useful feedback about individual passwords (e.g., they rule out some insecure passwords) they do not deal with the complexities of securing multiple accounts against an adversary who may have gained background knowledge about the user from previous attacks — we refer an interested reader to the full version [21] of this paper for more discussion.

Our notion of (n, ℓ, γ)-sharing set families (definition 5) is equivalent to Nisan and Widgerson's definition of a (k, m)-design [43]. However, Nisan and Widgerson were focused on a different application (constructing pseudorandom bit generators) and the range of parameters that they consider are not suitable for our password setting in which ℓ and γ are constants. See the full version[21] of this paper for more discussion.

2 Definitions

We use $\mathcal{P}$ to denote the space of possible passwords. A password management scheme needs to generate m passwords $p_1, ..., p_m \in \mathcal{P}$ — one for each account A_i.

Associative Memory and Cue-Association Pairs. Human memory is associative. Competitors in memory competitions routinely use mnemonic techniques (e.g., the method of loci [52]) which exploit associative memory[34]. For example, to remember the word 'apple' a competitor might imagine a giant apple on the floor in his bedroom. The bedroom now provides a context which can later be used as a cue to help the competitor remember the word apple. We use $\hat{c} \in \mathcal{C}$ to denote the cue, and we use $\hat{a} \in \mathcal{AS}$ to denote the corresponding association in a cue-association pair $(\hat{c}, \hat{a})$. Physically, $\hat{c}$ (resp. $\hat{a}$) might encode the excitement levels of the neurons in the user's brain when he thinks about his bedroom (resp. apples) [40].

We allow the password management scheme to store m sets of public cues $c_1, ..., c_m \subset \mathcal{C}$ in persistent memory to help the user remember each password. Because these cues are stored in persistent memory they are always available to the adversary as well as the user. Notice that a password may be derived from multiple cue-association pairs. We use $\hat{c} \in \mathcal{C}$ to denote a cue, $c \subset \mathcal{C}$ to denote a set of cues, and $C = \bigcup_{i=1}^{m} c_i$ to denote the set of all cues — $n = |C|$ denotes the total number of cue-association pairs that the user has to remember.

Visitation Schedules and Rehearsal Requirements. Each cue $\hat{c} \in C$ may have a rehearsal schedule to ensure that the cue-association pair $(\hat{c}, \hat{a})$ is maintained.

Definition 1. *A rehearsal schedule for a cue-association pair* $(\hat{c}, \hat{a})$ *is a sequence of times* $t_0^{\hat{c}} < t_1^{\hat{c}} < ....$ *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. We discuss sufficient rehearsal assumptions in section 3. The length of each interval $\left[t_i^{\hat{c}}, t_{i+1}^{\hat{c}}\right)$ may depend on the strength of the mnemonic technique used to memorize and rehearse a cue-association pair $(\hat{c}, \hat{a})$ as well as i — the number of prior rehearsals. For notational convenience, we use a function $R : C \times \mathbb{N} \to \mathbb{R}$ to specify the rehearsal requirements (e.g., $R(\hat{c}, j) = t_j^{\hat{c}}$), and we use $\mathcal{R}$ to denote a set of rehearsal functions.

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. We do not assume that the exact visitation schedules are known a priori. Instead we 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. Formally,

Definition 2. *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*

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

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

We use rehearsal requirements and visitation schedules to quantify the usability of a password management scheme by measuring the total number of extra rehearsals. 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, $X_{t,\hat{c}}$ denotes the total number of extra rehearsals of the cue-association pair $(\hat{c}, \hat{a})$ during the time interval $[0,t]$. We use $X_t = \sum_{\hat{c} \in C} X_{t,\hat{c}}$ to denote the total number of extra rehearsals during the time interval $[0,t]$ to maintain all of the cue-assocation pairs.

Usability Goal: Minimize the expected value of $E\left[X_t\right]$.

Password Management Scheme. A password management scheme includes a generator $\mathcal{G}_m$ and a rehearsal schedule $R \in \mathcal{R}$. The generator $\mathcal{G}_m\left(k, b, \boldsymbol{\lambda}, R\right)$ utilizes a user's knowledge $k \in \mathcal{K}$, random bits $b \in \{0,1\}^*$ to generate passwords $p_1, ..., p_m$ and public cues $c_1, ..., c_m \subseteq \mathcal{C}$. $\mathcal{G}_m$ may use the rehearsal schedule R and the visitation schedules $\boldsymbol{\lambda} = \langle \lambda_1, ..., \lambda_m \rangle$ of each site to help minimize $E\left[X_t\right]$. Because the cues $c_1, ...c_m$ are public they may be stored in persistent memory along with the code for the generator $\mathcal{G}_m$. In contrast, the passwords $p_1, ...p_m$ must be memorized and rehearsed by the user (following R) so that the cue association pairs (c_i, p_i) are maintained in his associative memory.

Definition 3. *A password management scheme is a tuple* $\langle \mathcal{G}_m, R \rangle$, *where* $\mathcal{G}_m$ *is a function* $\mathcal{G}_m : \mathcal{K} \times \{0,1\}^* \times \mathbb{R}^m \times \mathcal{R} \rightarrow \left(\mathcal{P} \times 2^{\mathcal{C}}\right)^m$ *and a* $R \in \mathcal{R}$ *is a rehearsal schedule which the user must follow for each cue.*

Our security analysis is not based on the secrecy of $\mathcal{G}_m$, k or the public cues $C = \bigcup_{i=1}^m c_i$. The adversary will be able to find the cues $c_1, ..., c_m$ because they are stored in persistent memory. In fact, we also assume that the adversary has background knowledge about the user (e.g., he may know k), and that the adversary knows the password management scheme $\mathcal{G}_m$. The only secret is the random string b used by $\mathcal{G}_m$ to produce $p_1, ..., p_m$.

Example Password Management Schemes. Most password suggestions are too vague (e.g.,"pick an obscure phrase that is personally meaningful to you") to satisfy the precise requirements of a password management scheme — formal security proofs of protocols involving human interaction can break down when humans behave in unexpected ways due to vague instructions [46]. We consider the following formalization of password management schemes: (1) **Reuse Weak** — the user selects a random dictionary word w (e.g., from a dictionary of

20, 000 words) and uses $p_i = w$ as the password for every account A_i. (2) **Reuse Strong** — the user selects four random dictionary words (w_1, w_2, w_3, w_4) and uses $p_i = w_1 w_2 w_3 w_4$ as the password for every account A_i. (3) **Lifehacker** (e.g., [3]) — The user selects three random words (w_1, w_2, w_3) from the dictionary as a base password $b = w_1 w_2 w_3$. The user also selects a random derivation rule d to derive a string from each account name (e.g., use the first three letters of the account name, use the first three vowels in the account name). The password for account A_i is $p_i = bd(A_i)$ where $d(A_i)$ denotes the derived string. (4) **Strong Random and Independent** — for each account A_i the user selects four fresh words independently at random from the dictionary and uses $p_i = w_1^i w_2^i w_3^i w_4^i$. Schemes (1)-(3) are formalizations of popular password management strategies. We argue that they are popular because they are easy to use, while the strongly secure scheme **Strong Random and Independent** is unpopular because the user must spend a lot of extra time rehearsing his passwords. See the full version [21] of this paper for more discussion of the security and usability of each scheme.

3 Usability Model

People typically adopt their password management scheme based on usability considerations instead of security considerations [33]. Our usability model can be used to explain why users tend to adopt insecure password management schemes like **Reuse Weak**, **Lifehacker**, or **Reuse Strong**. Our usability metric measures the extra effort that a user has to spend rehearsing his passwords. Our measurement depends on three important factors: rehearsal requirements for each cue, visitation rates for each site, and the total number of cues that the user needs to maintain. Our main technical result in this section is Theorem 1 — a formula to compute the total number of extra rehearsals that a user has to do to maintain all of his passwords for t days. To evaluate the formula we need to know the rehearsal requirements for each cue-association pair as well as the visitation frequency λ_i for each account A_i.

Rehearsal Requirements. If the password management scheme does not mandate sufficient rehearsal then the user might forget his passwords. Few memory studies have attempted to study memory retention over long periods of time so we do not know exactly what these rehearsal constraints should look like. While security proofs in cryptography are based on assumptions from complexity theory (e.g., hardness of factoring and discrete logarithm), we need to make assumptions about humans. For example, the assumption behind CAPTCHAs is that humans are able to perform a simple task like reading garbled text [58]. A rehearsal assumption specifies what types of rehearsal constraints are sufficient to maintain a memory. We consider two different assumptions about sufficient rehearsal schedules: Constant Rehearsal Assumption (CR) and Expanding Rehearsal Assumption (ER). Because some mnemonic devices are more effective than others (e.g., many people have amazing visual and spatial memories [54]) our assumptions are parameterized by a constant σ which represents the strength of the mnemonic devices used to memorize and rehearse a cue association pair.

Constant Rehearsal Assumption (CR): The rehearsal schedule given by $R(\hat{c}, i) = i\sigma$ is sufficient to maintain the association $(\hat{c}, \hat{a})$.

CR is a pessimistic assumption — it asserts that memories are not permanently strengthened by rehearsal. The user must continue rehearsing every σ days — even if the user has frequently rehearsed the password in the past.

Expanding Rehearsal Assumption (ER): The rehearsal schedule given by $R(\hat{c}, i) = 2^{i\sigma}$ is sufficient to maintain the association $(\hat{c}, \hat{a})$.

ER is more optimistic than CR — it asserts that memories are strengthened by rehearsal so that memories need to be rehearsed less and less frequently as time passes. If a password has already been rehearsed i times then the user does not have to rehearse again for $2^{i\sigma}$ days to satisfy the rehearsal requirement $\left[2^{i\sigma}, 2^{i\sigma+\sigma}\right)$. ER is consistent with several long term memory experiments [53],[17, Chapter 7], [60] — we refer the interested reader to full version[21] of this paper for more discussion. We also consider the rehearsal schedule $R(\hat{c}, i) = i^2$ (derived from [15,57]) in the full version — the usability results are almost indentical to those for ER.

Visitation Schedules. Visitation schedules may vary greatly from person to person. For example, a 2006 survey about Facebook usage showed that 47% of users logged in daily, 22.4% logged in about twice a week, 8.6% logged in about once a week, and 12% logged in about once a month[13]. We use a Poisson process with parameter λ_i to model the visitation schedule for site A_i. We assume that the value of $1/\lambda_i$ — the average inter-visitation time — is known. For example, some websites (e.g., gmail) may be visited daily ($\lambda_i = 1/1$ day) while other websites (e.g., IRS) may only be visited once a year on average (e.g., $\lambda_i = 1/365$ days). The Poisson process has been used to model the distribution of requests to a web server [47]. While the Poisson process certainly does not perfectly model a user's visitation schedule (e.g., visits to the IRS websites may be seasonal) we believe that the predictions we derive using this model will still be useful in guiding the development of usable password management schemes. While we focus on the Poisson arrival process, our analysis could be repeated for other random processes.

We consider four very different types of internet users: very active, typical, occasional and infrequent. Each user account A_i may be visited daily (e.g., $\lambda_i = 1$), every three days ($\lambda_i = 1/3$), every week (e.g. $\lambda_i = 1/7$), monthly ($\lambda_i = 1/31$), or yearly ($\lambda_i = 1/365$) on average. See table 1 to see the full visitation schedules we define for each type of user. For example, our very active user has 10 accounts he visits daily and 35 accounts he visits annually.

Table 1. 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

Table 2. $E\left[X_{365}\right]$: Extra Rehearsals over the first year for both rehearsal assumptions.
B+D: **Lifehacker**
SRI: **Strong Random and Independent**

Assumption	CR ($\sigma = 1$)		ER ($\sigma = 1$)	
Schedule/Scheme	B+D	SRI	B+D	SRI
Very Active	≈ 0	$23,396$	.023	420
Typical	.014	$24,545$	.084	456.6
Occasional	.05	$24,652$	.12	502.7
Infrequent	56.7	$26,751$	1.2	564

Extra Rehearsals. Theorem 1 leads us to our key observation: cue-sharing benefits users both by (1) reducing the number of cue-association pairs that the user has to memorize and (2) by increasing the rate of natural rehearsals for each cue-association pair. For example, a active user with 75 accounts would need to perform 420 extra-rehearsals over the first year to satisfy the rehearsal requirements given by ER if he adopts **Strong Random and Independent** or just 0.023 with **Lifehacker** — see table 2. The number of unique cue-association pairs n decreased by a factor of 75, but the total number of extra rehearsals $E[X_{365}]$ decreased by a factor of $8,260.8 \approx 75 \times 243$ due to the increased natural rehearsal rate.

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

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

Theorem 1 follows easily from Lemma 1 and linearity of expectations. Each cue-association pair $(\hat{c}, \hat{a})$ is rehearsed naturally whenever the user visits *any* site which uses the public cue $\hat{c}$. Lemma 1 makes use of two key properties of Poisson processes: (1) The natural rehearsal schedule for a cue $\hat{c}$ is itself a Poisson process, and (2) Independent Rehearsals - the probability that a rehearsal constraint is satisfied is independent of previous rehearsal constraints.

Lemma 1. *Let* $S_{\hat{c}} = \{i \mid \hat{c} \in c_i\}$ *and let* $\lambda_{\hat{c}} = \sum_{i \in S_{\hat{c}}} \lambda_i$ *then the probability that the cue* $\hat{c}$ *is not naturally rehearsed during time interval* $[a, b]$ *is* $\exp\left(-\lambda_{\hat{c}}\left(b - a\right)\right)$.

4 Security Model

In this section we present a game based security model for a password management scheme. The game is played between a user ($\mathcal{U}$) and a resource bounded adversary ($\mathcal{A}$) whose goal is to guess one of the user's passwords. We demonstrate how to select the parameters of the game by estimating the adversary's amortized cost of guessing. Our security definition is in the style of the exact security definitions of Bellare and Rogaway [18]. Previous security metrics (e.g., min-entropy, password strength meters) fail to model the full complexity of the password management problem (see the full version [21] of this paper for more discussion). By contrast, we assume that the adversary knows the user's password management scheme and is able to see any public cues. Furthermore, we assume that the adversary has background knowledge (e.g., birth date, hobbies) about the user (formally, the adversary is given $k \in \mathcal{K}$). Many breaches occur because the user falsely assumes that certain information is private (e.g., birth date, hobbies, favorite movie)[6,49].

Adversary Attacks. Before introducing our game based security model we consider the attacks that an adversary might mount. We group the adversary attacks into three categories: *Online Attack* — the adversary knows the user's ID and attempts to guess the password. The adversary will get locked out after s incorrect guesses (strikes). *Offline Attack* — the adversary learns both the cryptographic hash of the user's password and the hash function and can try many guesses $q_{\$B}$. The adversary is only limited by the resources B that he is willing to invest to crack the user's password. *Plaintext Password Leak Attack* — the adversary directly learns the user's password for an account. Once the adversary recovers the password p_i the account A_i has been compromised. However, a secure password management scheme should prevent the adversary from compromising more accounts.

We model online and offline attacks using a guess-limited oracle. Let $S \subseteq [m]$ be a set of indices, each representing an account. A guess-limited oracle $O_{S,q}$ is a blackbox function with the following behavior: 1) After q queries $O_{S,q}$ stops answering queries. 2) $\forall i \notin S$, $O_{S,q}(i,p) = \perp$ 3) $\forall i \in S$, $O_{S,q}(i,p_i) = 1$ and 4) $\forall i \in S, p \neq p_i$, $O_{S,q}(i,p) = 0$. Intutively, if the adversary steals the cryptographic password hashes for accounts $\{A_i \mid i \in S\}$, then he can execute an offline attack against each of these accounts. We also model an online attack against account A_i with the guess-limited oracle $O_{\{i\},s}$ with $s \ll q$ (e.g., $s = 3$ models a three-strikes policy in which a user is locked out after three incorrect guesses).

Game Based Definition of Security. Our cryptographic game proceeds as follows:
Setup: The user $\mathcal{U}$ starts with knowledge $k \in \mathcal{K}$, visitation schedule $\boldsymbol{\lambda} \in \mathbb{R}^m$, a random sequence of bits $b \in \{0,1\}^*$ and a rehearsal schedule $R \in \mathcal{R}$. The user runs $\mathcal{G}_m(k, b, \boldsymbol{\lambda}, R)$ to obtain m passwords $p_1, ..., p_m$ and public cues $c_1, ..., c_m \subseteq \mathcal{C}$ for accounts $A_1, ..., A_m$. The adversary $\mathcal{A}$ is given k, $\mathcal{G}_m$, $\boldsymbol{\lambda}$ and $c_1, ..., c_m$.
Plaintext Password Leak Attack: $\mathcal{A}$ adaptively selects a set $S \subseteq [m]$ s.t $|S| \leq r$ and receives p_i for each $i \in S$.
Offline Attack: $\mathcal{A}$ adaptively selects a set $S' \subseteq [m]$ s.t. $|S'| \leq h$, and is given blackbox access to the guess-limited offline oracle $O_{S',q}$.
Online Attack: For each $i \in [m] - S$, the adversary is given blackbox access to the guess-limited offline oracle $O_{\{i\},s}$.
Winner: $\mathcal{A}$ wins by outputting (j, p), where $j \in [m] - S$ and $p = p_j$.

We use $\mathbf{AdvWins}(k, b, \boldsymbol{\lambda}, \mathcal{G}_m, \mathcal{A})$ to denote the event that the adversary wins.

Definition 4. *We say that a password management scheme $\mathcal{G}_m$ is (q, δ, m, s, r, h)-secure if for every $k \in \mathcal{K}$ and adversary strategy $\mathcal{A}$ we have*

$$\Pr_b \left[\mathbf{AdvWins}\left(k, b, \boldsymbol{\lambda}, \mathcal{G}_m, \mathcal{A}\right) \right] \leq \delta \ .$$

Discussion: Observe that the adversary cannot win by outputting the password for an account that he already compromised in a plaintext password leak. For example, suppose that the adversary is able to obtain the plaintext passwords for $r = 2$ accounts of his choosing: p_i and p_j. While each of these breaches is arguably a success for the adversary the user's password management scheme cannot be blamed for any of these breaches. However, if the adversary can use this information to crack any of the user's other passwords then the password management scheme can be blamed for the additional breaches. For example, if our adversary is also able to use p_i and p_j to crack the cryptographic password hash $h(p_t)$ for another account A_t in at most q guesses then the password management scheme could be blamed for the breach of account A_t. Consequently, the adversary would win our game by outputting (t, p_t). If the password management scheme is $(q, 10^{-4}, m, s, 2, 1)$-secure then the probability that the adversary could win is at most 10^{-4} — so there is a very good chance that the adversary will fail to crack p_t.

Economic Upper Bound on q. Our guessing limit q is based on a model of a resource constrained adversary who has a budget of $\$B$ to crack one of the user's passwords. We use the upper bound $q_B = \$B/C_q$, where $C_q = \$R/f_H$ denotes the amortized cost per query (e.g., cost of renting ($\$R$) an hour of computing time on Amazon's cloud [1] divided by f_H — the number of times the cryptographic hash function can be evaluated in an hour.) We experimentally estimate f_H for SHA1, MD5 and BCRYPT[45] — more details can be found in the full version [21] of this paper. Assuming that the BCRYPT password hash function [45] was used to hash the passwords we get $q_B = B\left(5.155 \times 10^4\right)$ — we also consider cryptographic hash functions like SHA1, MD5 in the full version[21] of this paper. In our security analysis we focus on the specific value $q_{\$10^6} = 5.155 \times 10^{10}$ — the number of guesses the adversary can try if he invests $\$10^6$ to crack the user's password.

Sharing and Security. In section 3 we saw that sharing public cues across accounts improves usability by (1) reducing the number of cue-association pairs that the user has to memorize and rehearse, and (2) increasing the rate of natural rehearsals for each cue-association pair. However, conventional security wisdom says that passwords should be chosen independently. Is it possible to share public cues, and satisfy the strong notion of security from definition 4? Theorem 2 demonstrates that public cues can be shared securely provided that the public cues $\{c_1, \ldots, c_m\}$ are a (n, ℓ, γ)-sharing set family. The proof of theorem 2 can be found in the full version of this paper [21].

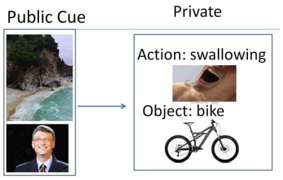
Definition 5. *We say that a set family $\mathcal{S} = \{S_1, ..., S_m\}$ is (n, ℓ, γ)-sharing if (1) $|\bigcup_{i=1}^{m} S_i| = n$, (2)$|S_i| = \ell$ for each $S_i \in \mathcal{S}$, and (3) $|S_i \cap S_j| \leq \gamma$ for each pair $S_i \neq S_j \in \mathcal{S}$.*

Theorem 2. *Let $\{c_1, \ldots, c_m\}$ be a (n, ℓ, γ)-sharing set of m public cues produced by the password management scheme $\mathcal{G}_m$. If each $a_i \in \mathcal{AS}$ is chosen uniformly at random then $\mathcal{G}_m$ satisfies (q, δ, m, s, r, h)-security for $\delta \leq \frac{q}{|\mathcal{AS}|^{\ell - \gamma r}}$ and any h.*

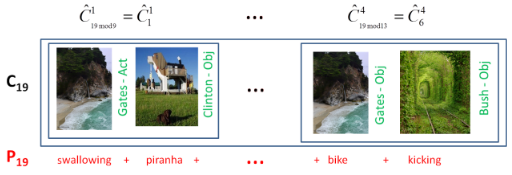
Discussion: To maintain security it is desirable to have ℓ large (so that passwords are strong) and γ small (so that passwords remain strong even after an adversary compromises some of the accounts). To maintain usability it is desirable to have n small (so that the user doesn't have to memorize many cue-association pairs). There is a fundamental trade-off between security and usability because it is difficult to achieve these goals without making n large.

For the special case $h = 0$ (e.g., the adversary is limited to online attacks) the security guarantees of Theorem 2 can be further improved to $\delta \leq \frac{sm}{|A|^{\ell - \gamma r}}$ because the adversary is actually limited to sm guesses.

5 Our Construction



(a) PAO Story with Cue


$\hat{C}^1_{19 \bmod 9} = \hat{C}^1_1$
...
$\hat{C}^4_{19 \bmod 13} = \hat{C}^4_6$
C_{19}
Gates - Act
Clinton - Obj
Gates - Obj
Bush - Obj
P_{19}
swallowing + piranha + ... + bike + kicking

(b) Account A_{19} using Shared Cues with the $(43, 4, 1)$-sharing set family **CRT** $(90, 9, 10, 11, 13)$.

Fig. 1.

We present Shared Cues— a novel password management scheme which balances security and usability considerations. The key idea is to strategically share cues to make sure that each cue is rehearsed frequently while preserving strong security goals. Our construction may be used in conjunction with powerful cue-based mnemonic techniques like memory palaces [52] and person-action-object stories [34] to increase σ — the association strength constant. We use person-action-object stories as a concrete example.

Person-Action-Object Stories. A random person-action-object (PAO) story for a person (e.g., Bill Gates) consists of a random action $a \in \mathcal{ACT}$ (e.g., swallowing) and a random object $o \in \mathcal{OBJ}$ (e.g., a bike). While PAO stories follow a very simple syntactic pattern they also tend to be surprising and interesting because the story is often unexpected (e.g., Bill Clinton kissing a piranha, or Michael Jordan torturing a lion). There is good evidence that memorable phrases tend to use uncommon combinations of words in common syntactic patterns [31]. Each cue $\hat{c} \in \mathcal{C}$ includes a person (e.g., Bill Gates) as well as a picture. To help the user memorize the story we tell him to imagine the scene taking place inside the picture (see Figure 1a for an example). We use algorithm 2 to automatically generate random PAO stories. The cue $\hat{c}$ could be selected either with the user's input (e.g., use the name of a friend and a favorite photograph) or automatically. As long as the cue $\hat{c}$ is fixed before the associated action-object story is selected the cue-association pairs will satisfy the independence condition of Theorem 2.

5.1 Constructing (n, ℓ, γ)-sharing set families

We use the Chinese Remainder Theorem to construct nearly optimal (n, ℓ, γ)-sharing set families. Our application of the Chinese Remainder Theorem is different from previous applications of the Chinese Remainder Theorem in cryptography (e.g., faster RSA decryption algorithm [32], secret sharing [16]). The inputs $n_1, ..., n_\ell$ to algorithm 1 should be co-prime so that we can invoke the Chinese Remainder Theorem — see Figure 1b for an example of our construction with $(n_1, n_2, n_3, n_4) = (9, 10, 11, 13)$.

Algorithm 1. CRT $(m, n_1, ..., n_\ell)$

```
Input: $m$, and $n_1, ..., n_\ell$.
for $i = 1 \to m$ do
    $S_i \leftarrow \emptyset$
    for $j = 1 \to \ell$ do
        $N_j \leftarrow \sum_{i=1}^{j-1} n_j$
        $S_i \leftarrow S_i \cup \{(i \mod n_j) + N_j\}$
return $\{S_1, \ldots, S_m\}$
```

Algorithm 2. CreatePAOStories

```
Input: $n$, random bits b, images $I_1, ..., I_n$, and names $P_1, ..., P_n$.
for $i = 1 \to n$ do
    $a_i \overset{\$}{\leftarrow} \mathcal{ACT}$, $o_i \overset{\$}{\leftarrow} \mathcal{OBJ}$    %Using random bits b
%Split PAO stories to optimize usability
for $i = 1 \to n$ do
    $\hat{c}_i \leftarrow ((I_i, P_i, \text{'}Act'\text{'}), (I_{i+1 \mod n}, P_{i+1 \mod n}, \text{'}Obj'\text{'}))$
    $\hat{a}_i \leftarrow (a_i, o_{i+1 \mod n})$
return $\{\hat{c}_1, \ldots, \hat{c}_n\}, \{\hat{a}_1, \ldots, \hat{a}_n\}$
```

Lemma 2 says that algorithm 1 produces a (n, ℓ, γ)-sharing set family of size m as long as certain technical conditions apply (e.g., algorithm 1 can be run with any numbers $n_1, ..., n_\ell$, but lemma 2 only applies if the numbers are pairwise co-prime.).

Lemma 2. *If the numbers $n_1 < n_2 < \ldots < n_\ell$ are pairwise co-prime and $m \leq \prod_{i=1}^{\gamma+1} n_i$ then algorithm 1 returns a $(\sum_{i=1}^{\ell} n_i, \ell, \gamma)$-sharing set of public cues.*

Proof. Suppose for contradiction that $|S_i \bigcap S_k| \geq \gamma + 1$ for $i < k < m$, then by construction we can find $\gamma + 1$ distinct indices $j_1, ..., j_{\gamma+1} \in$ such that $i \equiv k \mod n_{j_t}$ for $1 \leq t \leq \gamma + 1$. The Chinese Remainder Theorem states that there is a unique number x^* s.t. (1) $1 \leq x^* < \prod_{t=1}^{\gamma+1} n_{j_t}$, and (2) $x^* \equiv k \mod n_{j_t}$ for $1 \leq t \leq \gamma + 1$. However, we have $i < m \leq \prod_{t=1}^{\gamma+1} n_{j_t}$. Hence, $i = x^*$ and by similar reasoning $k = x^*$. Contradiction!

Example: Suppose that we select pairwise co-prime numbers $n_1 = 9, n_2 = 10, n_3 = 11, n_4 = 13$, then $\mathbf{CRT}(m, n_1, \ldots, n_4)$ generates a $(43, 4, 1)$-sharing set family of size $m = n_1 \times n_2 = 90$ (i.e. the public cues for two accounts will overlap in at most one common cue), and for $m \leq n_1 \times n_2 \times n_3 = 990$ we get a $(43, 4, 2)$-sharing set family.

Lemma 2 implies that we can construct a (n, ℓ, γ)-sharing set system of size $m \geq \Omega\left((n/\ell)^{\gamma+1}\right)$ by selecting each $n_i \approx n/\ell$. Theorem 3 proves that we can't hope to do much better — any (n, ℓ, γ)-sharing set system has size $m \leq O\left((n/\ell)^{\gamma+1}\right)$. We refer the interested reader to the full version[21] of this paper for the proof of Theorem 3 and for discussion about additional (n, ℓ, γ)-sharing constructions.

Theorem 3. *Suppose that $\mathcal{S} = \{S_1, ..., S_m\}$ is a (n, ℓ, γ)-sharing set family of size m then $m \leq \binom{n}{\gamma+1} \Big/ \binom{\ell}{\gamma+1}$.*

5.2 Shared Cues

Our password management scheme —Shared Cues— uses a (n, ℓ, γ)-sharing set family of size m (e.g., a set family generated by algorithm 1) as a hardcoded input to output the public cues $c_1, ...c_m \subseteq \mathcal{C}$ and passwords $p_1, ..., p_m$ for each account. We use algorithm 2 to generate the underlying cues $\hat{c}_1, \ldots, \hat{c}_n \in \mathcal{C}$ and their associated PAO stories. The computer is responsible for storing the public cues in persistent memory and the user is responsible for memorizing and rehearsing each cue-association pair $(\hat{c}_i, \hat{a}_i)$.

We use two additional tricks to improve usability: (1) Algorithm 2 splits each PAO story into two parts so that each cue $\hat{c}$ consists of *two* pictures and *two* corresponding people with a label (action/object) for each person (see Figure 1b). A user who sees cue $\hat{c}_i$ will be rehearsing both the i'th and the $i+1$'th PAO story, but will only have to enter one action and one object. (2) To optimize usability we use GreedyMap (Algorithm 4) to produce a permutation $\pi : [m] \rightarrow [m]$ over the public cues — the goal is to minimize the total number of extra rehearsals by ensuring that each cue is used by a frequently visited account.

Algorithm 3. *SharedCues* $[S_1, \ldots, S_m,]$ $\mathcal{G}_m$

```
Input: $k \in \mathcal{K}$, $b$, $\lambda_1, ..., \lambda_m$, Rehearsal Schedule $R$.
$\{\hat{c}_1, \ldots, \hat{c}_n\}, \{\hat{a}_1, \ldots, \hat{a}_n\} \leftarrow$ CreatePAOStories $(\mathbf{n}, \mathbf{I_1}, ..., \mathbf{I_n}, \mathbf{P_1}, \ldots, \mathbf{P_n})$
for $i = 1 \rightarrow m$ do
    $c_i \leftarrow \{\hat{c}_j \mid j \in S_i\}$, and $p_i \leftarrow \{\hat{a}_j \mid j \in S_i\}$.
% Permute cues
$\pi \leftarrow GreedyMap(m, \lambda_1, ..., \lambda_m, c_1, \ldots, c_m, R, \sigma)$
return $(p_{\pi(1)}, c_{\pi(1)}), \ldots, (p_{\pi(m)}, c_{\pi(m)})$
User: Rehearses the cue-association pairs $(\hat{c}_i, \hat{a}_i)$ by following the rehearsal schedule $R$.
Computer: Stores the public cues $c_1, ..., c_m$ in persistent memory.
```

Once we have constructed our public cues $c_1, ..., c_m \subseteq \mathcal{C}$ we need to create a mapping π between cues and accounts $A_1, ..., A_m$. Our goal is to minimize the total number of extra rehearsals that the user has to do to satisfy his rehearsal requirements. Formally, we define the **Min-Rehearsal** problem as follows:
Instance: Public Cues $c_1, ..., c_m \subseteq \mathcal{C}$, Visitation Schedule $\lambda_1, ..., \lambda_m$, a rehearsal schedule R for the underlying cues $\hat{c} \in C$ and a time frame t.
Output: A bijective mapping $\pi : \{1, ..., m\} \rightarrow \{1, ..., m\}$ mapping account A_i to public cue $S_{\pi(i)}$ which minimizes $E[X_t]$.
Unfortunately, we can show that **Min-Rehearsal** is NP-Hard to even approximate within a constant factor. Our reduction from Set Cover can be found in the full version[21] of this paper. Instead GreedyMap uses a greedy heuristic to generate a permutation π.

Theorem 4. *It is NP-Hard to approximate* **Min-Rehearsal** *within a constant factor.*

Algorithm 4. GreedyMap

```
Input: $m, \lambda_1, ..., \lambda_m, c_1, \ldots, c_m$, Rehearsal Schedule $R$ (e.g., CR or ER with parameter $\sigma$).
Relabel: Sort $\lambda$'s s.t $\lambda_i \geq \lambda_{i+1}$ for all $i \leq m-1$.
Initialize: $\pi_0(j) \leftarrow \perp$ for $j \leq m$, $UsedCues \leftarrow \emptyset$.
%$\pi_i$ denotes a partial mapping $[i] \rightarrow [m]$,for $j > i$, the mapping is undefined (e.g., $\pi_i(j) = \perp$). Let $S_k = \{\hat{c} \mid \hat{c} \in c_k\}$.
for $i = 1 \rightarrow m$ do
    for all $j \in [m] - UsedCues$ do
        $\Delta_j \leftarrow \sum_{\hat{c} \in S_j} E\left[X_{t,\hat{c}} \middle| \lambda_{\hat{c}} = \lambda_i + \sum_{j : \hat{c} \in S_{\pi_{i-1}(j)}} \lambda_j\right] - E\left[X_{t,\hat{c}} \middle| \lambda_{\hat{c}} = \sum_{j : \hat{c} \in S_{\pi_{i-1}(j)}} \lambda_j\right]$
% $\Delta_j$: expected reduction in total extra rehearsals if we set $\pi_i(i) = j$
    $\pi_i(i) \leftarrow \arg\max_j \Delta_j$, $UsedCues \leftarrow UsedCues \cup \{\pi_i(i)\}$
return $\pi^m$
```

5.3 Usability and Security Analysis

We consider three instantiations of Shared Cues: SC-0, SC-1 and SC-2. SC-0 uses a $(9,4,3)$-sharing family of public cues of size $m = 126$ — constructed by taking all $\binom{9}{4} = 126$ subsets of size 4. SC-1 uses a $(43,4,1)$-sharing family of public cues of size $m = 90$ — constructed using algorithm 1 with $m = 90$ and $(n_1, n_2, n_3, n_4) = (9, 10, 11, 13)$. SC-2 uses a $(60,5,1)$-sharing family of public cues of size $m = 90$ — constructed using algorithm 1 with $m = 90$ and $(n_1, n_2, n_3, n_4, n_5) = (9, 10, 11, 13, 17)$.

Our usability results can be found in table 3 and our security results can be found in table 4. We present our usability results for the very active, typical, occasional and infrequent internet users (see table 1 for the visitation schedules) under both sufficient rehearsal assumptions CR and ER. Table 3 shows the values of $E[X_{365}]$ — computed using the formula from Theorem 1 — for SC-0, SC-1 and SC-2. We used association strength parameter $\sigma = 1$ to evaluate each password management scheme — though we expect that σ will be higher for schemes like Shared Cues that use strong mnemonic techniques [1].

Table 3. $E[X_{365}]$: Extra Rehearsals over the first year for SC-0,SC-1 and SC-2

Assumption	CR ($\sigma = 1$)			ER ($\sigma = 1$)		
Schedule/Scheme	SC-0	SC-1	SC-2	SC-0	SC-1	SC-2
Very Active	≈ 0	$1,309$	$2,436$	≈ 0	3.93	7.54
Typical	≈ 0.42	$3,225$	$5,491$	≈ 0	10.89	19.89
Occasional	≈ 1.28	$9,488$	$6,734$	≈ 0	22.07	34.23
Infrequent	≈ 723	$13,214$	$18,764$	≈ 2.44	119.77	173.92

Our security guarantees for SC-0,SC-1 and SC-2 are illustrated in Table 4. The values were computed using Theorem 2. We assume that $|\mathcal{AS}| = 140^2$ where $\mathcal{AS} = \mathcal{ACT} \times \mathcal{OBJ}$ (e.g., their are 140 distinct actions and objects), and that the adversary is willing to spend at most $\$10^6$ on cracking the user's passwords (e.g., $q = q_{\$10^6} = 5.155 \times 10^{10}$). The values of δ in the $h = 0$ columns were computed assuming that $m \leq 100$.

Discussion: Comparing tables 3 and 2 we see that **Lifehacker** is the most usable password management scheme, but SC-0 compares very favorably! Unlike **Lifehacker**, SC-0 provides provable security guarantees after the adversary phishes one account — though the guarantees break down if the adversary can also execute an offline attack. While SC-1 and SC-2 are not as secure as **Strong Random and Independent** — the security guarantees from **Strong Random and Independent** do not break down even if the adversary can recover *many* of the user's plaintext passwords — SC-1 and SC-2 are far more usable than **Strong Random and Independent**. Furthermore, SC-1 and SC-2 do provide very strong security guarantees (e.g., SC-2 passwords remain secure against offline attacks even after an adversary obtains two plaintext passwords for accounts

[1] We explore the effect of σ on $E[X_{t,c}]$ in the full version[21] of this paper.

Table 4. Shared Cues $(q_{\$10^6}, \delta, m, s, r, h)$-Security: δ vs h and r using a (n, ℓ, γ)-sharing family of m public cues

Offline Attack?	$h = 0$			$h > 0$		
(n, ℓ, γ)-sharing	$r = 0$	$r = 1$	$r = 2$	$r = 0$	$r = 1$	$r = 2$
$(n, 4, 3)$ (e.g., SC-0)	2×10^{-15}	0.011	1	3.5×10^{-7}	1	1
$(n, 4, 1)$ (e.g., SC-1)	2×10^{-15}	4×10^{-11}	8×10^{-7}	3.5×10^{-7}	0.007	1
$(n, 5, 1)$ (e.g., SC-2)	1×10^{-19}	2×10^{-15}	4×10^{-11}	1.8×10^{-11}	3.5×10^{-7}	0.007

of his choosing). For the very active, typical and occasional user the number of extra rehearsals required by SC-1 and SC-2 are quite reasonable (e.g., the typical user would need to perform less than one extra rehearsal per month). The usability benefits of SC-1 and SC-2 are less pronounced for the infrequent user — though the advantage over **Strong Random and Independent** is still significant.

6 Discussion and Future Work

We conclude by discussing future directions of research.
Sufficient Rehearsal Assumptions: While there is strong empirical evidence for the Expanding Rehearsal assumption in the memory literature (e.g., [60]), the parameters we use are drawn from prior studies in other domains. It would be useful to conduct user studies to test the Expanding Rehearsal assumption in the password context, and obtain parameter estimates specific to the password setting. We also believe that user feedback from a password management scheme like Shared Cues could be an invaluable source of data about rehearsal and long term memory retention.
Expanding Security over Time: Most extra rehearsals occur soon after the user memorizes a cue-association pair — when the rehearsal intervals are still small. Is it possible to start with a password management scheme with weaker security guaratnees (e.g., SC-0), and increase security over time by having the user memorize additional cue-association pairs as time passes?
Human Computable Passwords: Shared Cues only relies on the human capacity to memorize and retrieve information, and is secure against at most $r = \ell/\gamma$ plaintext password leak attacks. Could we improve security (or usability) by having the user perform simple computations to recover his passwords? Hopper and Blum proposed a 'human authentication protocol' — based on the noisy parity problem — as an alternative to passwords [36], but their protocol seems to be too complicated for humans to execute. Could similar ideas be used to construct a secure human-computation based password management scheme?

References

1. Amazon ec2 pricing, `http://aws.amazon.com/ec2/pricing/` (retrieved October 22, 2012)

2. Cert incident note in-98.03: Password cracking activity (July 1998), http://www.cert.org/incident_notes/IN-98.03.html (retrieved August 16, 2011)
3. Geek to live: Choose (and remember) great passwords (July 2006), http://lifehacker.com/184773/geek-to-live--choose-and-remember-great-passwords (retrieved September 27, 2012)
4. Rockyou hack: From bad to worse (December 2009), http://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/ (retrieved September 27, 2012)
5. Oh man, what a day! an update on our security breach (April 2010), http://blogs.atlassian.com/news/2010/04/oh_man_what_a_day_an_update_on_our_security_breach.html (retrieved August 18, 2011)
6. Sarah palin vs the hacker (May 2010), http://www.telegraph.co.uk/news/worldnews/sarah-palin/7750050/Sarah-Palin-vs-the-hacker.html (retrieved September 9, 2012)
7. Nato site hacked (June 2011), http://www.theregister.co.uk/2011/06/24/nato_hack_attack/ (retrieved August 16, 2011)
8. Update on playstation network/qriocity services (April 2011), http://blog.us.playstation.com/2011/04/22/update-on-playstation-network-qriocity-services/ (retrieved May 22, 2012)
9. Apple security blunder exposes lion login passwords in clear text (May 2012), http://www.zdnet.com/blog/security/apple-security-blunder-exposes-lion-login-passwords-in-clear-text/11963 (retrieved May 22, 2012)
10. Data breach at ieee.org: 100k plaintext passwords (September 2012), http://ieeelog.com/ (retrieved September 27, 2012)
11. An update on linkedin member passwords compromised (June 2012), http://blog.linkedin.com/2012/06/06/linkedin-member-passwords-compromised/ (retrieved September 27, 2012)
12. Zappos customer accounts breached (January 2012), http://www.usatoday.com/tech/news/story/2012-01-16/mark-smith-zappos-breach-tips/52593484/1 (retrieved May 22, 2012)
13. Acquisti, A., Gross, R.: Imagined communities: awareness, information sharing, and privacy on the facebook. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 36–58. Springer, Heidelberg (2006)
14. Anderson, J., Matessa, M., Lebiere, C.: Act-r: A theory of higher level cognition and its relation to visual attention. Human-Computer Interaction 12(4), 439–462 (1997)
15. Anderson, J.R., Schooler, L.J.: Reflections of the environment in memory. Psychological Science 2(6), 396–408 (1991)
16. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Transactions on Information Theory 29(2), 208–210 (1983)
17. Baddeley, A.: Human memory: Theory and practice. Psychology Pr. (1997)
18. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
19. Biddle, R., Chiasson, S., Van Oorschot, P.: Graphical passwords: Learning from the first twelve years. ACM Computing Surveys (CSUR) 44(4), 19 (2012)
20. Biddle, S.: Anonymous leaks 90,000 military email accounts in latest antisec attack (July 2011), http://gizmodo.com/5820049/anonymous-leaks-90000-military-email-accounts-in-latest-antisec-attack (retrieved August 16, 2011)

21. Blocki, J., Blum, M., Datta, A.: Naturally rehearsing passwords. CoRR abs/1302.5122 (2013)
22. Blocki, J., Komanduri, S., Procaccia, A., Sheffet, O.: Optimizing password composition policies
23. Bojinov, H., Sanchez, D., Reber, P., Boneh, D., Lincoln, P.: Neuroscience meets cryptography: designing crypto primitives secure against rubber hose attacks. In: Proceedings of the 21st USENIX Conference on Security Symposium, pp. 33–33. USENIX Association (2012)
24. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: 2012 IEEE Symposium on Security and Privacy (SP), pp. 538–552. IEEE (2012)
25. Bonneau, J., Herley, C., van Oorschot, P.C., Stajano, F.: The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In: IEEE Symposium on Security and Privacy, pp. 553–567. IEEE (2012)
26. Boztas, S.: Entropies, guessing, and cryptography. Department of Mathematics, Royal Melbourne Institute of Technology, Tech. Rep 6 (1999)
27. Brand, S. Department of defense password management guideline
28. Brostoff, S., Sasse, M.: Are Passfaces more usable than passwords: A field trial investigation. In: People and Computers XIV-Usability or Else: Proceedings of HCI, pp. 405–424 (2000)
29. Burnett, M.: Perfect passwords: selection, protection, authentication. Syngress Publishing (2005)
30. Center, I.: Consumer password worst practices. Imperva (White Paper) (2010)
31. Danescu-Niculescu-Mizil, C., Cheng, J., Kleinberg, J., Lee, L.: You had me at hello: How phrasing affects memorability. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers, vol. 1, pp. 892–901. Association for Computational Linguistics (2012)
32. Ding, C., Pei, D., Salomaa, A.: Chinese remainder theorem. World Scientific (1996)
33. Florencio, D., Herley, C.: A large-scale study of web password habits. In: Proceedings of the 16th International Conference on World Wide Web, pp. 657–666. ACM (2007)
34. Foer, J.: Moonwalking with Einstein: The Art and Science of Remembering Everything. Penguin Press (2011)
35. Gaw, S., Felten, E.W.: Password management strategies for online accounts. In: Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS 2006, pp. 44–55. ACM, New York (2006)
36. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
37. Kohonen, T.: Associative memory: A system-theoretical approach. Springer, Berlin (1977)
38. Komanduri, S., Shay, R., Kelley, P., Mazurek, M., Bauer, L., Christin, N., Cranor, L., Egelman, S.: Of passwords and people: measuring the effect of password-composition policies. In: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems, pp. 2595–2604. ACM (2011)
39. Kruger, H., Steyn, T., Medlin, B., Drevin, L.: An empirical assessment of factors impeding effective password management. Journal of Information Privacy and Security 4(4), 45–59 (2008)
40. Marr, D.: Simple memory: a theory for archicortex. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, 23–81 (1971)
41. Massey, J.: Guessing and entropy. In: Proceedings of the 1994 IEEE International Symposium on Information Theory, p. 204. IEEE (1994)

42. Monroe, R.: Xkcd: Password strength, `http://www.xkcd.com/936/` (retrieved August 16, 2011)
43. Nisan, N., Wigderson, A.: Hardness vs randomness. Journal of Computer and System Sciences 49(2), 149–167 (1994)
44. Pliam, J.O.: On the incomparability of entropy and marginal guesswork in brute-force attacks. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 67–79. Springer, Heidelberg (2000)
45. Provos, N., Mazieres, D.: Bcrypt algorithm
46. Radke, K., Boyd, C., Nieto, J.G., Brereton, M.: Towards a secure human-and-computer mutual authentication protocol. In: Proceedings of the Tenth Australasian Information Security Conference (AISC 2012), vol. 125, pp. 39–46. Australian Computer Society Inc. (2012)
47. Rasch, G.: The poisson process as a model for a diversity of behavioral phenomena. In: International Congress of Psychology (1963)
48. Scarfone, K., Souppaya, M.: Guide to enterprise password management (draft). National Institute of Standards and Technology 800-188 6, 38 (2009)
49. Schechter, S., Brush, A., Egelman, S.: It's no secret. measuring the security and reliability of authentication via 'secret' questions. In: 2009 30th IEEE Symposium on Security and Privacy, pp. 375–390. IEEE (2009)
50. Shay, R., Kelley, P., Komanduri, S., Mazurek, M., Ur, B., Vidas, T., Bauer, L., Christin, N., Cranor, L.: Correct horse battery staple: Exploring the usability of system-assigned passphrases. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, p. 7. ACM (2012)
51. Singer, A.: No plaintext passwords. The Magazine of Usenix & Sage 26(7) (November 2001) (retrieved August 16, 2011)
52. Spence, J.: The memory palace of Matteo Ricci. Penguin Books (1985)
53. Squire, L.: On the course of forgetting in very long-term memory. Journal of Experimental Psychology: Learning, Memory, and Cognition 15(2), 241 (1989)
54. Standingt, L.: Learning 10,000 pictures. Quarterly Journal of Experimental Psychology 5(20), 7–22 (1973)
55. Stein, J.: Pimp my password. Time, 62 (August 29, 2011)
56. Valiant, L.: Memorization and association on a realistic neural model. Neural Computation 17(3), 527–555 (2005)
57. van Rijn, H., van Maanen, L., van Woudenberg, M.: Passing the test: Improving learning gains by balancing spacing and testing effects. In: Proceedings of the 9th International Conference of Cognitive Modeling (2009)
58. Von Ahn, L., Blum, M., Hopper, N., Langford, J.: Captcha: Using hard ai problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 646–646. Springer, Heidelberg (2003)
59. Willshaw, D., Buckingham, J.: An assessment of marr's theory of the hippocampus as a temporary memory store. Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences 329(1253), 205 (1990)
60. Wozniak, P., Gorzelanczyk, E.J.: Optimization of repetition spacing in the practice of learning. Acta Neurobiologiae Experimentalis 54, 59–59 (1994)
61. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password memorability and security: Empirical results. IEEE Security & Privacy 2(5), 25–31 (2004)