Proceedings Article

Optimizing Password Composition Policies

JEREMIAH BLOCKI, Carnegie Mellon University
SARANGA KOMANDURI, Carnegie Mellon University
ARIEL D. PROCACCIA, Carnegie Mellon University
OR SHEFFET, Carnegie Mellon University

A password composition policy restricts the space of allowable passwords to eliminate weak passwords that are vulnerable to statistical guessing attacks. Usability studies have demonstrated that existing password composition policies can sometimes result in weaker password distributions; hence a more *principled* approach is needed. We introduce the *first theoretical model* for optimizing password composition policies. We study the computational and sample complexity of this problem under different assumptions on the structure of policies and on users' preferences over passwords. Our main positive result is an algorithm that – with high probability — constructs almost optimal policies (which are specified as a union of subsets of allowed passwords), and requires only a small number of samples of users' preferred passwords. We complement our theoretical results with simulations using a real-world dataset of 32 million passwords.

Categories and Subject Descriptors: K.4.4 [**Electronic Commerce**]: Security

General Terms: Algorithms, Economics, Security, Theory

Additional Key Words and Phrases: Password composition policy, Sampling, Computational complexity

1. INTRODUCTION

Imagine a web surfer, an online shopper, or a reviewer in a prominent CS and Economics conference who logs on for the first time to a server; so that she can sign up for some service, place a shopping order, or view a list of assigned papers. Such a user registers on the server by choosing a username and picking a password. Naturally, our user's first attempt at picking a password is her favorite combination '`123456`', which the server declines. She then has to pick a password that follows certain guidelines: of suitable length, involving lower- and upper-case letters, with numbers or special characters, etc. Such *password composition policies* defend against the "first line" of attack – guessing attacks by uninformed attackers (attackers with no previous knowledge of the user whose account they are trying to break into).

Password composition policies are a necessity because — without them — user-selected passwords are predictable. Indeed, many unrestricted users would select simple passwords like '`123456`', '`password`' and '`letmein`' [Doel 2012]. Furthermore, this issue is of great

This research was supported in part by the National Science Foundation Science and Technology TRUST, by the National Science Foundation under grants DGE-0903659, CNS-1116776, CCF-1101215 and CCF-1116892, by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273 from the Army Research Office, by the AFOSR MURI on Science of Cybersecurity, by a gift from Microsoft Research and by a NSF Graduate Research Fellowship.
Authors' addresses: J. Blocki, Computer Science Department, Carnegie Mellon University, email: `jblocki@cs.cmu.edu`; S. Komanduri, Human Computer Interaction Institute, Carnegie Mellon University, email: `sarangak@cs.cmu.edu`; A. D. Procaccia, Computer Science Department, Carnegie Mellon University, email: `arielpro@cs.cmu.edu`; O. Sheffet, Computer Science Department, Carnegie Mellon University, email: `osheffet@cs.cmu.edu`.

EC'13, June 16–20, 2013, Philadelphia, USA.

importance to today's economy. Passwords are commonly used in electronic commerce to protect financial assets. In fact, the passwords themselves have financial value. Symantec reported that compromised passwords are sold for between $4 and $30 on the black market [Fossi et al. 2008], and a 2004 Gartner case study [Witty et al. 2004] estimated that it cost a large firm over $17 per password-reset call. Nevertheless, existing password composition policies are typically not principled, and do not necessarily result in less common passwords. For example, studies show that users respond to restrictions in predictable ways [Komanduri et al. 2011], or pick weaker passwords due to user-fatigue [Clair et al. 2006; Kruger et al. 2008].

In this paper, we initiate the *algorithmic* study of password composition policies. Such policies restrict the space of passwords to a subset of allowed passwords, and force each user to pick a password in this subset. Thus, n users induce a distribution over passwords where for a password w, $\Pr[w] = \frac{1}{n}|\{i : i \text{ picks } w\}|$. By declaring different subsets of allowed passwords, different password composition policies induce different distributions. Our work formalizes and addresses the algorithmic problem a server administrator faces when designing a password composition policy; we ask:

> *In what settings can the information about the users' preferences over passwords allow us to design a password composition policy that is* guaranteed *to induce a password distribution as close to uniform as possible?*

We wish to stress at this point that we do not take a cryptographic approach to the problem: we do not design a protocol aimed at amplifying a password's strength, nor do we rely on standard cryptographic assumptions or techniques in designing our password composition policies. Single-factor authentication does not defend against an attacker who learns about the most probable password from an external source. Furthermore, because password systems often allow users multiple attempts in entering their password, an attacker can make a small number of guesses with impunity. Therefore, we instead focus on the design and analysis of algorithms for optimizing the password composition policy's induced distribution over passwords, and in our theoretical results compare the performance of our algorithm to the optimal policy among exponentially many potential policies in the *worst case*.

1.1. Our Model

We study the algorithmic problem of optimizing password composition policies along multiple dimensions: the goal, the user model, and the policy structure.

Goal. We focus on designing a policy that maximizes the minimum-entropy of the resulting password distribution. Specifically, we assume the server deals with n users, each picking a password from some space of passwords $\mathcal{P}$ that respects the server's password composition policy. These n passwords form a distribution over the domain of all allowed passwords and our goal is to minimize the probability of the most likely password. This is a natural goal (see Section 7), as opposed to maximizing the Shannon-entropy of the distribution, which for example is still high even if half the people choose the same password and the other half choose a password uniformly at random from $\mathcal{P}$. From a security standpoint, the minimum entropy represents the fraction of accounts that could be compromised in one guess. For example, an adversary would be able to crack 0.9% of RockYou passwords [Imperva 2010] with only one guess. Alternatively, should the attacker attempt to break into only one account, the minimum entropy represents the likelihood that the account is compromised on the first guess. We also consider a slightly stronger goal of minimizing the fraction of accounts that could be compromised using k guesses, that is, the overall probability of the k most likely passwords [Boztas 1999].

User model. We consider two models for how users select passwords when presented with a password composition policy.

In the *ranking model*, each user has an implicit ranking over passwords, from the most preferred to the least preferred. Given a password policy, each user selects the highest-ranking password among those allowed by the policy. There is a distribution over the space of rankings that determines the fraction of users with each possible ranking. Note that for any password composition policy, such a distribution over rankings induces a distribution over the most preferred *allowed* passwords.

In the *normalization model*, there is a distribution $\mathcal{D}$ over the space of all passwords. This distribution tells us the likelihood that an unrestricted user would select a given password. Given a password composition policy, $\mathcal{D}$ induces a new distribution over the allowed passwords (which can be obtained by normalizing the probabilities under $\mathcal{D}$ of the allowed passwords). When we ban a password the fraction of users that prefer each allowed password grows; the natural interpretation is that users who preferred an allowed password still use that password, but users who preferred a banned password are redistributed among the allowed passwords according to the induced distribution.

As we show, the normalization model is strictly more restrictive than the ranking model: any distribution in the normalization model can be simulated in the ranking model, but there exist hardness results for the ranking model that do not hold for the normalization model.

Policy structure. We consider the best policy that is restricted to manipulation of a given set of *rules* — each rule is simply a predefined subset of potential passwords. These rules are given to us as part of the problem (see Section 7 for a discussion of this point). If we interpret a rule as a subset of banned passwords (e.g., passwords shorter than seven characters), its complement (e.g., passwords of at least seven characters) can be interpreted as a subset of allowed passwords. As such, when we take the union of rules we get either a set of banned passwords (*negative rules*) or allowed passwords (*positive rules*); this is our password composition policy. While the distinction between the two cases may at first seem a mere technicality, it is in fact quite significant due to the following observation. If we ban the union of rules then in order to ban a password that was picked by too many users, we may ban any rule that contains this password. In contrast, if we allow a union of rules then in order to ban this password we must not allow *any* rule that contains it. In other words, when our goal is to discard a password in the negative rules setting, we have multiple ways to do so. When our goal is to discard a password in the positive rules setting, we have only one way to do so — excluding all rules that allow this password. As we shall see, this seemingly small difference leads to a clear separation between the two scenarios in terms of the complexity of designing optimal policies.

We pay special attention to the case where each password has its own singleton rule. In this setting, a policy can be interpreted as a "blacklist" of banned passwords that do not necessarily share common characteristics. Note that when each password has its own singleton rule, it does not matter whether these rules are positive or negative.

1.2. Our Results

As we noted above, a password composition policy induces a distribution over most preferred passwords (in both user models). We study algorithms that *sample* these distributions — algorithms that repeatedly query random users and ask them to choose a password constrained by some policy, and then output the a good policy for the empirical sample of users. Our goal is therefore twofold: (i) to show that having sufficiently many samples (i.e., sufficiently many users queried) guarantees that w.h.p the best policy for the empirical sample is good for all users; and (ii) exhibit algorithms that find an optimal (or close-to-optimal) policy for a given sample. Clearly, we want our sample size to be "small". In

Table I: Summary of Complexity Results.

	Ranking Model		**Normalization Model**	
	Constant k	**Large k**	**Constant k**	**Large k**
Singleton rules	P	NP-Hard (Thm 3.4) APX-Hard w/ UGC (Thm 3.5)	P	P (Thm 4.1)
Positive rules	P (Thm 3.2)	NP-Hard	P	NP-Hard (Thm 4.4)
Negative rules	$n^{1/3}$-approx is NP-hard (Thm 3.6)	NP-Hard	NP-Hard (Thm 4.2)	NP-Hard

particular, since the size of the space of all passwords $\mathcal{P}$ — which we denote by N — is typically very large (e.g., $\mathcal{P}$ can include all passwords that are no longer than 32 ASCII characters), we wish to get a bound on the sample size that is independent of N.

For the ease of exposition, we discuss goal (ii) before goal (i). I.e., we first (Sections 3 and 4) study the problem in a simpler setting where the preferences of all users are given to us as input; and only then (Section 5) we introduce an algorithm that samples users' preferences. Also for the ease of exposition, we first discuss algorithms where $\mathcal{P}$ is a part of the input, so they are allowed to run in time polynomial in N. This is motivated by the fact that computational complexity of problems in this setting informs their study in the sampling setting — it is hopeless to design efficient sampling algorithms for problems that are computationally hard. (Efficient sampling algorithms are applicable only to computationally tractable problems.)

Table I summarizes our complexity results. The parameter k refers to our optimization target: minimizing the likelihood of the k most likely passwords. Some results are direct corollaries of others — using the fact that singleton rules are a special case of positive rules and the fact that the normalization model is a special case of the ranking model (see Section 2). Looking at the table one immediately notices a clear separation between negative rules and positive rules: optimization using the latter is much easier.

We therefore focus on positive rules in our attempt to design an efficient sampling algorithm. Our main result is the best one could hope for in this setting. We design an algorithm that works in the more general ranking model, and finds a policy whose entropy is ϵ-close to optimal with probability $1 - \delta$, for any given $\epsilon, \delta > 0$. The required number of samples is polynomial in $1/\epsilon$, $\log(1/\delta)$, and the number of positive rules m. We can assume that m is small, because each rule corresponds to a subset of passwords that can be concisely described to users.

These results can be applied in a practical setting, and we show this through simulated sampling experiments using natural rules and a large dataset of real passwords. The experimental results provide evidence for the difficulty of the negative rules setting: we search all combinations of rules to find the optimal policy and then attempt to discover this policy by making decisions both randomly and with a heuristic. In the negative rules setting, neither approach succeeded at finding the optimal policy after hundreds of iterations at various sample sizes, and average-case performance did not improve with sample size. In the positive rules setting, the average-case performance of our efficient algorithm improved with sample size and, with a moderate sample size, found policies that were either optimal or very close to optimal.

1.3. Related Work

It has been repeatedly demonstrated that users tend to select easily guessable passwords [Imperva 2010; Doel 2012; Bonneau 2012] and NIST recommends that organizations "should

also ensure that other trivial passwords cannot be set," to thwart potential attackers [Scarfone and Souppaya 2009]. Unfortunately, this task is more difficult than it might appear at first. Policies were initially developed without empirical data to support them, since such data was not available to policy designers [Burr et al. 2006]. When hackers leaked the RockYou dataset to the Internet, both researchers (and attackers) suddenly had access to password data, leading to many insights into true passwords [Weir et al. 2010]. However, recent research analyzing leaked datasets from non-English speakers, notably Hebrew and Chinese-language websites, shows that trivial password choices can vary between contexts, making a simple blacklist approach ineffective [Bonneau and Xu 2012]. This means that, depending on the context, a policy based on leaked password data might provide no security guarantee, and it has ethical issues as well.

To combat this issue, researchers have turned to a sampling approach. Bonneau [2012] added a system for sampling to the Yahoo! password infrastructure. This system allows one to gain empirical data about the frequency distribution of passwords without revealing the passwords themselves. Such approaches provide a way of gathering empirical data about passwords while maintaining the anonymity of users. Our algorithms could be used in conjunction with such an infrastructure to optimize policies.

Komanduri et al. [2011] studied the effectiveness of several basic password composition policies by using Amazon's Mechanical Turk to conduct a large scale user study. They found that people often respond to restrictions in predictable ways (e.g., if the password needs to contain a capital letter users might tend to capitalize the first letter of a password) and provide very general recommendations for password composition policies. However, no theoretical model has been proposed for studying the password composition problem.

Schechter et al. [2010] suggest using a popularity oracle to prevent individual passwords that have been used too frequently from being selected by new users. They also proposed using the count-min sketch data structure [Cormode and Muthukrishnan 2005] to build such a popularity oracle. Malone and Maher [2012] suggest a similar system using a Metropolis-Hastings scheme to force an approximately uniform distribution on passwords. Usability results on the effectiveness of dictionary checks [Komanduri et al. 2011] suggest that such policies would be very frustrating since the policy is hidden from users behind an oracle. In contrast, we seek to construct optimal policies from combinations of rules that are visible to the user and can be described in natural language.

This consideration of users is important to electronic commerce, even where security is concerned. Florencio and Herley [2010] studied the economic factors that drive institutions to adopt strict password composition policies and find that they often value the user experience over security. An e-mail provider like Yahoo! might adopt simple composition policies because a frustrated user could easily switch to Gmail, while universities are free to adopt strict policies because users cannot switch easily.

2. A MODEL OF PASSWORD COMPOSITION POLICIES

We use $\mathcal{P}$ to denote the space of all possible passwords. $N = |\mathcal{P}|$ is used to denote the total number of passwords. We denote the number of users by n.

A password composition policy may be specified in terms of rules. A rule is a subset of passwords $R \subseteq \mathcal{P}$ (e.g., the set of all passwords with more than seven characters). We use $R_1, ..., R_m$ to denote a list of rules that may be active or inactive. We consider two schemes.

— *Positive Rules:* A password w is allowed if and only if it is allowed by some active positive rule. Formally, a password composition policy $\mathcal{A}_S = \bigcup_{i \in S} R_i$ is specified by a set $S \subseteq [m] = \{1, ..., m\}$ of active rules. In this setting rules should consist of sets of passwords which we expect to be strong (e.g., R_i might be the set of all passwords longer than 10 characters, or the set of all passwords that use both upper and lowercase letters, or the set of all passwords that do not include a dictionary word).

— *Negative Rules:* A password w is allowed if and only if it is not contained in any active negative rule. Formally, a solution $\mathcal{A}_S = \left\{w \in \mathcal{P} \,\middle|\, w \notin \bigcup_{i \in S} R_i\right\}$ is given by a subset $S \subseteq [m]$ of active rules. A negative rule should consist of passwords that we expect to be weak (e.g., R_i might be the set of all passwords without an uppercase letter, or the set of all passwords shorter than 6 characters, or the set of all passwords that include a dictionary word).

We also consider the special case of *singleton rules*, where our rules are $\{w_1\}, \ldots, \{w_N\}$. Equivalently, we are allowed to ban or allow any individual password.

We use $\Pr[w \mid \mathcal{A}]$ to denote the probability of a password w given composition policy $\mathcal{A}$. For $w \notin \mathcal{A}$ we have $\Pr[w \mid \mathcal{A}] = 0$. Given a set $W \subseteq \mathcal{A}$ we will also use $\Pr[W \mid \mathcal{A}] = \sum_{w \in W} \Pr[w \mid \mathcal{A}]$. We use $p\,(k, \mathcal{A}) = \max_{W \subseteq \mathcal{A}: |W| = k} \Pr[W \mid \mathcal{A}]$ to denote the probability of the k most popular passwords. Intuitively, $p\,(k, \mathcal{A})$ represents the probability that an adversary can successfully guess a password using k attempts. To avoid cumbersome notation we sometimes use $p_1 = p\,(1, \mathcal{A})$ to denote the probability of the most popular password. Similarly, we use p_2 (resp., p_k) to denote the probability of the second (resp., k'th) most popular password.

We consider two user models that determine how users choose passwords under a given password composition policy.

— *The ranking model:* A ranking is simply a permutation of $\mathcal{P}$, which represents a user's password preferences. It can be represented using an ordered list $\ell_i = w_{1,i}, ..., w_{N,i}$; user i prefers password $w_{j,i}$ to $w_{j+1,i}$ for all j. The ranking ℓ_i naturally tells us which password i will pick under any composition policy $\mathcal{A}$. Specifically, i will use password $w_{\mathcal{A},i} = w_{j,i}$ where $j = \text{argmin}\{t : w_{t,i} \in \mathcal{A}\}$. Given a distribution $\mathcal{D}$ over rankings, we have

$$\Pr\left[w \,|\mathcal{A}\right] = \Pr_{\ell_i \sim \mathcal{D}}\left[w_{\mathcal{A},i} = w\right] \ .$$

— *The normalization model:* Let $\mathcal{D}$ be an initial distribution over $\mathcal{P}$, and let $\Pr[w] = \Pr_{x \sim \mathcal{D}}[w = x]$. If we select the composition policy $\mathcal{A}$ then the probabilities of all $w \in \mathcal{A}$ are simply re-normalized so that

$$\forall w \in \mathcal{P}, \mathcal{A} \subseteq \mathcal{P}, \Pr\left[w \,|\mathcal{A}\right] = \frac{\Pr\left[w\right]}{\Pr\left[\mathcal{A}\right]} \ .$$

Clearly it holds for both models that the probability of an allowed password monotonically increases as one bans more passwords. Formally, for all $w \in \mathcal{A}$ and $B \subseteq \mathcal{P}$ such that $w \notin B$ we have

$$\Pr\left[w \,|\mathcal{A}\right] \leq \Pr\left[w \,|\mathcal{A} \backslash B\right] \ . \tag{1}$$

Another important observation is that for our purposes the ranking model is more general than the normalization model. Indeed, we argue that a distribution $\mathcal{D}$ over passwords in the normalization model induces an equivalent distribution over rankings. To generate the most highly ranked password, draw a password w_1 from $\mathcal{D}$. Next, let $\mathcal{A}_1 = \mathcal{P} \setminus \{w_1\}$, and draw the next most preferred password w_2, where $w_2 = w$ with probability $\Pr[w \,|\mathcal{A}_1]$. In the following round we ban w_2 to obtain a policy $\mathcal{A}_2$, and so on, until all passwords have been banned.

Given $k \in \mathbb{N}$, our goal is to find $S \subseteq [m]$ such that $p\,(k, \mathcal{A}_S) \leq p\,(k, \mathcal{A}_{S'})$ for all $S' \subseteq [m]$. When $k = 1$ this goal is equivalent to maximizing the minimum entropy. If $p\,(k, \mathcal{A}_S) \leq c \cdot p\,(k, \mathcal{A}_{S'}) + \epsilon$ for all $S' \subseteq [m]$ then we say that S is a (c, ϵ)-approximation. To simplify notation we sometimes use c-approximation instead of $(c, 0)$-approximation.

3. RANKING MODEL: COMPLEXITY RESULTS

In this section we consider the complexity of finding the optimal password composition policy in the more general ranking model when the organization is given complete informa-

tion about users' preferences. Specifically, the organization is given the rankings $\ell_1, ..., \ell_n$ of every user.

Our first result is for the positive rules setting. Given positive rules $R_1, ..., R_m$ we show that $p(k, \mathcal{A}_S)$ can be computed efficiently for constant values of k (see Theorem 3.2). In fact, for the special case $k = 1$ we present a very simple algorithm that suffices. Both algorithms can be easily extended to the less general normalization model. Our algorithms are based on three simple ideas: (1) Reduced Preference Lists — each preference list ℓ_i can be efficiently reduced to a short (length $\leq m$) preference list $\hat{\ell}_i$. (2) Guess and Check — start by guessing the 'structure' of the optimal solution and find the resulting solution. (3) Iterative Elimination — find the most popular password w and eliminate all positive rules that contain w. Our sampling algorithms are based on the same core ideas.

Unfortunately, the picture is different in the negative rules even when k is a constant. Given negative rules $R_1, ..., R_m$ we show that it is hard to even $n^{1/3}$-approximate $p(1, \mathcal{A}_S)$. Also, for non-constant values of k we show that it is hard to compute $p(k, \mathcal{A}_S)$ in the singleton rules setting, which immediately implies hardness in both the positive rules setting and in the negative rules setting. Given a stronger complexity assumption known as the Unique Games Conjecture [Khot 2002] it is also hard to c_0-approximate $p(k, \mathcal{A}_S)$ in the singleton rules setting for some constant c_0. However, our hardness results do not rule out the possibility of a c-approximation for a larger constant c.

3.1. Positive Rules: Efficient Algorithm for Constant k

We first show that $p(k, \mathcal{A}_S)$ can be computed efficiently for constant values of k in the positive rules setting. In this section the organization is given positive rules $R_1, ..., R_m$ as well as preference lists $\ell_1, ..., \ell_n$. We assume that the organization can efficiently query the preference lists (e.g., given $S \subseteq [m]$ the organization can efficiently find $\ell_i(\mathcal{A}_S)$ — user i's preferred password given policy $\mathcal{A}_S$).

We elaborate on the key algorithmic ideas listed above. First, we can efficiently reduce each preference list ℓ_i to a list $\hat{\ell}_i$ of at most m passwords (Claim 3.1). While the reduced list $\hat{\ell}_i$ is much shorter than ℓ_i it is still sufficient to determine user i's preferred password given policy $\mathcal{A}_S$ for any $S \subseteq [m]$. We use $\hat{\mathcal{P}}$ to denote the reduced space of potential passwords.

Algorithm 1 Reduce

Input:
Preference List: ℓ
Positive Rules: $R_1, ..., R_m$
Initialize: $i \leftarrow 0$, $S_0 \leftarrow [m]$, $\hat{\ell} \leftarrow$ empty ranking.
while $S_i \neq \emptyset$ **do**
  Let w be $\ell(\mathcal{A}_{S_i})$.
  $\hat{\ell} \leftarrow \langle \hat{\ell}, w \rangle$ ▷ 'Append' the current most preferred password to $\hat{\ell}$
  $S_{i+1} \leftarrow S_i \setminus \{j \mid w \in R_j\}$ ▷ Deactivate all rules that contain w
  $i \leftarrow i + 1$
return $\hat{\ell}$

CLAIM 3.1. *Algorithm 1 makes at most m queries to ℓ and m^2 membership queries and outputs a reduced preference list $\hat{\ell}$ over at most m passwords such that for every $S \subseteq [m]$ it holds that $\hat{\ell}(\mathcal{A}_S) = \ell(\mathcal{A}_S)$.*

PROOF. Clearly, the algorithm's main loop iterates at most m times because for each i we eliminate at least one rule (e.g., $|S_{i+1}| < |S_i|$), so the bound on queries and the length

of $\hat{\ell}$ are immediate. (Because we assume that we can query ℓ efficiently Algorithm 1 is also efficient.) By construction we have $\hat{\ell}(S_i) = \ell(S_i)$ for each S_i. Fix any $S \subseteq [m]$. Let S_i be such that $S \subseteq S_i$ yet $S \not\subseteq S_{i+1}$ and let w_i be the most preferred word in ℓ out of all words in $\bigcup_{j \in S_i} R_j$. If it is the case that $w_i \in \bigcup_{j \in S} R_j$, then w_i is the most preferred word in S too and we're done. Otherwise, $w_i \in \bigcup_{j \in S_i \setminus S} R_j$ which means that removing the set $\{j \in S_i \ : \ w_i \in R_j\}$ creates a set S_{i+1} s.t. $S \subseteq S_{i+1}$, contradiction. □

Second, the "guess and check" idea means that our algorithm starts by guessing what the optimal solution looks like (e.g., what the k most popular passwords will be in the optimal solution and what the probability of the k'th most popular password is). There are at most $(mn)^{O(k)}$ potential solutions to brute-force try. As we show, for each solution, it is easy to figure out which sets must be eliminated.

Algorithm 2 GuessAndCheck

Input:
Preference Lists: $\ell_1, ..., \ell_n$
Positive Rules: $R_1, ..., R_m \subseteq \mathcal{P}$
Integer k
Initialize: $Candidates \leftarrow \emptyset$ ▷ Candidate Solutions
for $i = 1 \to n$ **do**
  $\hat{\ell}_i \leftarrow Reduce\,(\ell_i, R_1, ..., R_m)$
$\hat{\mathcal{P}} \leftarrow \bigcup_{i=1}^{n} \hat{\ell}_i$. ▷ Reduced Password Space
for all (G, p) with $G \subseteq \hat{\mathcal{P}}$ s.t. $|G| = k$ and $p \in \{1/n, 2/n, ..., 1\}$ **do**
  $S_{G,p} \leftarrow [m]$
  while $S_{G,p} \neq \emptyset$ and $\exists w \in \left(\hat{\mathcal{P}} \setminus G\right) \cap \mathcal{A}_{S_{G,p}}$ s.t $\Pr\left[w \,\middle|\, \mathcal{A}_{S_{G,p}}\right] > p$ **do**
    $S_{G,p} \leftarrow S_{G,p} \setminus \{j \mid w \in R_j\}$ ▷ Ban w because it is inconsistent with guess
  if $\Pr\left[w \,\middle|\, \mathcal{A}_{S_{G,p}}\right] \leq p$ for all $w \in \left(\mathcal{A}_{S_{G,p}} \setminus G\right)$ **then**
    $Candidates \leftarrow Candidates \cup \{S_{G,p}\}$
return $\arg\min_{(G,p) \in Candidates} p\left(k, \mathcal{A}_{S_{G,p}}\right)$

THEOREM 3.2. *Algorithm 2 runs in time polynomial in n^k, m^k and outputs a set of positive rules $S \subseteq [m]$ of positive rules such that*

$$p\left(k, \mathcal{A}_S\right) \leq p\left(k, \mathcal{A}_{S'}\right)$$

for every other set $S' \subseteq [m]$.

PROOF. It is evident that the running time of the algorithm is $\text{poly}(n^k, m^k)$ since we only have $O((nm)^k)$ potential solutions to try.

Let $\mathcal{A}_{S^*}$ denote an optimal solution and let G^* denote the k most popular passwords in this solution. Suppose we start with the correct guess ($G = G^*$ and p is the probability of the k'th most popular password), then we claim that our algorithm must produce the optimal solution. In particular, we maintain the invariant that $\mathcal{A}_{S^*} \subseteq \mathcal{A}_{S_{G,p}}$ until we converge to the optimal solution. Clearly, this is true initially — before we have eliminated any passwords.

Suppose that the invariant holds and that our algorithm bans a password $w \in \mathcal{P} \setminus G$ by deactivating all rules in $S_{G,p}$ that contain w. Then by the definition of our algorithm we must have $\Pr\left[w \,\middle|\, \mathcal{A}_{S_{G,p}}\right] > p$. If $w \in \mathcal{A}_{S^*}$ then by Equation (1) we have

$$\Pr\left[w \,|\, \mathcal{A}_{S^*}\right] \geq \Pr\left[w \,\middle|\, \mathcal{A}_{S_{G,p}}\right] > p \ ,$$

which contradicts the choice of G. Therefore $w \notin \mathcal{A}_{S^*}$, so all rules that contain it are deactivated in $\mathcal{A}_{S^*}$ and the invariant still holds. By definition Algorithm 2 terminates when every password $w \in A_{S_{G,p}} \setminus G$ has probability at most p. Because our invariant still holds we can apply Equation (1) again to get

$$\Pr\left[G \,\middle|\, \mathcal{A}_{S_{G,p}}\right] \leq \Pr\left[G \,|\, \mathcal{A}_{S^*}\right] = p\left(k, \mathcal{A}_{S^*}\right) \ .$$

Hence, $A_{S_{G,p}}$ is an optimal solution. □

For the special case $k = 1$ the simple algorithm IterativeElimination (Algorithm 3) suffices. The basic idea is very simple: iteratively eliminate the most popular password w by deactivating all positive rules that contain w. We repeat this process until no passwords remain. We claim that one of the solutions along the way was the optimal solution.

Algorithm 3 IterativeElimination

Input:
Preference Lists: $\ell_1, ..., \ell_n$
Positive Rules: $R_1, ..., R_m \subseteq \mathcal{P}$
Initialize: $S_0 \leftarrow [m]$, $i \leftarrow 0$
while $S_i \neq \emptyset$ **do**
  $w\left(S_i\right) \leftarrow \arg\max\left\{\Pr\left[w \mid \mathcal{A}_{S_i}\right] \mid w \in \mathcal{A}_{S_i}\right\}$ ▷ $w\left(S_i\right)$ is most popular allowed pwd
  $S_{i+1} \leftarrow S_i \setminus \left\{j \mid w\left(S_i\right) \in R_j\right\}$ ▷ Deactivate all rules that contain $w\left(S_i\right)$
  $i \leftarrow i + 1$
return S_{i^*} where $i^* \leftarrow \arg\min_i p\left(1, \mathcal{A}_{S_i}\right)$

THEOREM 3.3. *Algorithm 3 outputs a set of positive rules $S \subseteq [m]$ such that*

$$\forall S' \subseteq [m] \ , \quad p\left(1, \mathcal{A}_S\right) \leq p\left(1, \mathcal{A}_{S'}\right) \ .$$

PROOF. Let T denote the optimal policy. Clearly if $T = [m]$ then our algorithm returns $S^* = T$ because that is the first set we try. Otherwise, $T \subsetneq [m]$. Let S be the last set our algorithm considers that has the property that $T \subseteq S$. Again, if $T = S$, our algorithm returns S. Let $w(T)$ be the most popular word in $\mathcal{A}_T$, and because of optimality $\Pr[w(T) \mid \mathcal{A}_T] \leq \Pr[w(S) \mid \mathcal{A}_S]$.

Now, because we modify S to not contain T in the next iteration, then the most popular word in S, $w(S)$ has to belong to some rule R_j where $j \in T$. Therefore $w(S) \in \bigcup_{j \in T} R_j$, and by the definition, the most popular word in $\mathcal{A}_T$ satisfies $\Pr[w(T) \mid \mathcal{A}_T] \geq \Pr[w(S) \mid \mathcal{A}_T]$.

But observe, because $w(S) \in \bigcup_{j \in T} R_j$, we must have that $w(S)$ is at least as popular in T. Indeed, if ℓ is a preference list where we disallowed $\mathcal{P} \setminus \bigcup_{j \in S} R_j$ and the most preferred word is $w(S)$, then as long as we disallow more words but keep allowing $w(S)$ the word $w(S)$ remains at the top of the list. Therefore, $\Pr[w(S) \mid \mathcal{A}_T] \geq \Pr[w(S) \mid \mathcal{A}_S]$. Combining together all inequalities we get $\Pr[w(T) \mid \mathcal{A}_T] = \Pr[w(S) \mid \mathcal{A}_S]$, which means our algorithm returns $S^* = S$. □

3.2. Singleton Rules: Hardness for Large k

Now we turn our attention to the problem of optimizing $p\left(k, \mathcal{A}_S\right)$ for large values of k. Theorem 3.4 says that unless $P = NP$ no polynomial time algorithm can compute $p\left(k, \mathcal{A}_S\right)$ even with singleton rules. If we are willing to make the Unique Games Conjecture (UGC) [Khot 2002] then it is hard to even c_0-approximate $p\left(k, \mathcal{A}_S\right)$ for some constant c_0. These results immediately imply hardness in both the positive and negative rules setting because these settings are a generalization of the singleton rules setting.

THEOREM 3.4. *Unless $P = NP$ there is no $\text{poly}(k, n, N)$-algorithm that gets as input an arbitrary set of n preference-lists $\ell_1, ..., \ell_n$ over $\mathcal{P}$ and an integer k, and outputs the optimal $p(k, \mathcal{A})$ in the singleton rules setting.*

PROOF. We prove the theorem using a reduction from the Vertex-Cover problem. Given a graph G over g vertices and e edges and an integer t, we first define

$$\mathcal{P} = \{w_u \ : \ u \in V(G)\} \cup \{w_{u,v} \ : \ (u, v) \in E(G)\}$$

and observe that $|\mathcal{P}| = g + e$. We also construct the following $n = 2e$ preference-lists, where for every edge $(u, v) \in E(G)$ we have the two lists:

$$\ell_{u,v} = w_u, w_{u,v}, \dots$$
$$\ell_{v,u} = w_v, w_{u,v}, \dots$$

where the choice of passwords below position 2 is arbitrary, but both rankings must be identical from position 2 onwards. Finally, we set $k = g + e - t - 1$.

Given a policy $\mathcal{A} \subseteq \mathcal{P}$, we denote all banned words as $\mathcal{B} = \mathcal{P} \setminus \mathcal{A}$. We denote by $L_{\mathcal{B}}$ as the set of words that at least one user ranks first after banning all words in $\mathcal{B}$. Observe, $L_{\emptyset} = \{w_u \ : \ u \in V(G)\}$. Using this notation, we show this reduction indeed proves NP-hardness.

First, suppose G has a vertex cover C of size $\leq t$. Then by banning all passwords $\mathcal{B} = \{w_v \ : \ v \in C\}$ we now have $L_{\mathcal{B}} = \mathcal{P} \setminus \mathcal{B}$, because for every $(u, v) \in E(G)$ either w_u or w_v are banned, so the word $w_{u,v}$ appears at the top of at least one of the two lists $\{\ell_{u,v}, \ell_{v,u}\}$. Therefore, the n preference-lists induce a distribution whose support contains $g + e - |\mathcal{B}| \geq g + e - t$ words, thus $p(g + e - t - 1, \mathcal{A}) < 1$.

Conversely, suppose all vertex covers of G are of size at least $t + 1$. Let $\mathcal{A}$ be any set of banned words. Clearly, if $|\mathcal{B}| \geq t+1$ then the distribution induced by the n preferences-lists has support of size at most $g+e-t-1$, which means that $p(g+e-t-1, \mathcal{B}) = 1$. Otherwise, $|\mathcal{B}| \leq t$, and we denote the set of vertices $C = \{v : w_v \in \mathcal{B}\}$. Observe, since any vertex cover of G must contain $\geq t+1$ vertices, then there has to be at least $t+1-|C|$ edges that C does not cover (since we can always complete C to a vertex cover by adding one vertex from each uncovered edge). Therefore, there have to be at least $t + 1 - |C|$ words that do not appear at the top of any preference list. We conclude that the distribution induced by the n preference-lists has a support of size at most

$$|L_{\mathcal{B}}| = g - |C| + e - (t + 1 - |C|) \leq g + e - t - 1$$

thus $p(g + e - t - 1, \mathcal{A}) = 1$. □

From the same reduction described in Theorem 3.4 we get UGC-hardness of approximation. While there are sub-exponential time algorithms to solve the Unique Games problem [Arora et al. 2010], there are no known polynomial time algorithms. Many famous approximation hardness results are based on the Unique Games Conjecture (e.g., $2 - \epsilon$ hardness for vertex cover [Khot and Regev 2008]). Our reduction relies on a result in [Austrin et al. 2011], which says that vertex cover is hard to approximate up to a (say) 1.5-factor even on bounded degree graphs. Because we start with a bounded degree graph we can argue that each password in our reduction appears at the top of at most d preference-lists for some constant d. See the full version of this paper [Blocki et al. 2013] for a formal proof.

THEOREM 3.5. *There exists a constant $c > 1$ such that it is UGC-hard for a $\text{poly}(n, N, k)$-time algorithm to c-approximate the optimal $p(k, \mathcal{A})$ in the singleton rules setting and the rankings model.*

3.3. Negative Rules: Hardness of Approximation for $k = 1$

We next turn to negative rules, where we show that the problem is extremely difficult even for $k = 1$. Though the proof appears in the full version of this paper [Blocki et al. 2013], it is quite interesting and we encourage the reader to take a look.

THEOREM 3.6. *Let $\epsilon > 0$. Unless $P = NP$ there is no polynomial time algorithm (in N, n, m) that approximates $\min_{S\subseteq[m]} p(1, \mathcal{A}_S)$ to a factor of $n^{1/3-\epsilon}$ in the negative rules setting and the rankings model.*

4. NORMALIZATION MODEL: COMPLEXITY RESULTS

In this section we focus on complexity results for the normalization model. Here the structure of the input to our problem is a bit different: For each password $w \in \mathcal{P}$ we are given the probability $\Pr[w]$ that w is selected by a random user when $\mathcal{A} = \mathcal{P}$. Note that now we can give the distribution explicitly because it requires N numbers (whereas a distribution over rankings requires $N!$ numbers). This distribution induces a distribution over $\mathcal{P}$ for any password composition policy $\mathcal{A}$ by normalizing probabilities, as explained in Section 2.

Because the normalization model is a special case of the ranking model our algorithms for the ranking model can also be applied in the normalization model. The question is whether or not the hardness results carry over.

We first consider the singleton rules setting with large k, and show that that we can compute $\arg\min_{\mathcal{A}\subseteq\mathcal{P}} p(k, \mathcal{A})$ in polynomial time in N (Theorem 4.1). This result separates the normalization model from the ranking model (e.g., compare Theorems 4.1 and 3.4). However, it does not extend to the positive rules setting. In fact, we show that optimizing $p(k, \mathcal{A}_S)$ is NP-Hard when k is a parameter (Theorem 4.4).

With negative rules $R_1, ..., R_m$ we show that it is hard to c_0-approximate $\arg\max_{S\subseteq[m]} p(1, \mathcal{A}_S)$ (Theorem 4.2). However, we cannot rule out the possibility of an efficient c-approximation algorithm for some constant c in the normalization model (recall that Theorem 3.6 ruled out the possibility of a c-approximation algorithm in the ranking model for any c).

4.1. Singleton Rules: Efficient Algorithm for large k

We present SortAndOptimize — an efficient algorithm to optimize $p(k, \mathcal{A})$ in the singleton rules setting for *any* value of k. The key intuition behind our algorithm is that if $w_1 \in \mathcal{P}$ is the most likely password then w_1 will remain the most likely allowed password unless we ban it — a property that does not hold in the rankings model. A formal proof of Theorem 4.1 can be found in the full version of this paper [Blocki et al. 2013].

THEOREM 4.1. *For every k, Algorithm 4 computes $\arg\min_{\mathcal{A}} p(k, \mathcal{A})$ in the singleton rules setting of the normalized probabilities model, in time $O(N \log(N))$.*

Algorithm 4 SortAndOptimize

Input:
Password space $\mathcal{P}$ and a probability distribution over $\mathcal{P}$.
Integer k.
Sort the words in $\mathcal{P}$ from highest to lowest probability, $w_1, w_2, \ldots, w_N$.
return the set $\mathcal{A}_i = \{w_j \ : \ j \geq i\}$, where i minimizes the ratio

$$p(k, \mathcal{A}_i) = \frac{\sum_{i\leq j\leq i+k} \Pr[w_j]}{\sum_{j\geq i} \Pr[w_j]}$$

4.2. Negative Rules: Hardness for $k = 1$

We next prove an inapproximability result that is somewhat weaker than the one that we obtained for the more general ranking model.

THEOREM 4.2. *There exists some constant $c_0 > 1$ such that unless $NP = BPP$ no polynomial time algorithm (in n, N, m) can c_0-approximate $\min_{S\subseteq[m]} p(1, \mathcal{A}_S)$ in the negative rules setting and the normalization model.*

We will require the following construction; the proof is given in the full version of this paper [Blocki et al. 2013].

LEMMA 4.3. *Fix m and s such that $m \geq s$. There exists a domain D of size $\Theta(s^2 \log(m))$ and a family of m sets, $F_1, F_2, \ldots, F_m \subseteq D$, such that each set in the family contains $\frac{|D|}{2s}$ elements, and for every $C \subseteq [m]$ of size $|C| \leq s$, we have that the size of the union $\left|\bigcup_{i\in C} F_i\right| \geq \frac{|D|}{2s}\frac{|C|}{4}$. This domain can be constructed in randomized $\mathrm{poly}(s, m)$ time.*

That is, each set in this family contains exactly the same fraction of the domain, and furthermore — any union of $|C| \leq s$ sets has the property that its cardinality is proportional to $\Omega(|C|)|F_i|$.

PROOF OF THEOREM 4.2. We reduce from Set-Cover — one of the classic NP-Complete problems [Karp 1972]. We are given sets $S_1, ..., S_m \subseteq U$, universe $U = \{1, ..., g\}$, and an integer $t \leq m$, and we are asked whether there is a set $C \subseteq [m]$ of size $\leq t$ such that $U = \bigcup_{i\in C} S_i$.

It is a known fact that there exist Set-Cover instances, with (g, m, t) all polynomially dependent of each other, that are hard to approximate to a factor of $c \ln n$ [Alon et al. 2006]. That is, on this particular family of instances, it is NP-hard to distinguish whether there exists a cover of size t or all covers have size $(1 - \epsilon)c \cdot t \ln n$.

We now describe the reduction. Given a (g, m, t)-Set Cover instance, we set $s = c \cdot t \ln g = \Theta(t \ln t)$ and construct a domain D and m sets $F_1, F_2, \ldots, F_m \subseteq D$ as in Lemma 4.3. We then create the following password-banning instance. First $\mathcal{P}$ is the union of D with additional disjoint g words denoted $w_1, ..., w_g$. Now, for each set S_i in the Set-Cover we add a rule R_i where $R_i = \{w_j\}_{j\in S_i} \cup F_i$. Finally, we set the words' probabilities as follows. Fixing some arbitrarily small $\delta > 0$, we set for every i the probability $\Pr[w_i] = \frac{1-\delta}{g}$, and for every $x \in D$ we set the probability $\Pr[x] = \frac{\delta}{|D|}$.

Without loss of generality we can assume that $|D| \geq 100g$ (because, for example, we can take $100g$ copies of the original D). Therefore, any policy that bans all of $\{w_1, w_2, \ldots w_g\}$ yet leaves a constant (say $> 1/10$) fraction of D has $p_1 \leq 10/|D|$, whereas any policy that keeps even one of the words in $\{w_1, w_2, \ldots, w_g\}$ has $p_1 \geq 1/(2g)$. Therefore, if the Set-Cover instance has a cover of size $\leq s = \Theta(t \ln g)$, then a c_0-approximation of the optimal banning-policy must find a cover for $\{w_1, w_2, \ldots, w_g\}$. We will assume from now on that our Set-Cover instance is such that it has a cover of size $\leq s$. (Indeed, if $s > t \log(t)$ then the instance is no longer NP-hard, since the greedy algorithm must return a cover of size $> t \log(t)$ which causes us to deduce that the optimal cover must have size $> t$.)

So now, suppose our Set-Cover instance has a cover of size t. Then the respective union of rules bans every password in $\{w_1, w_2, \ldots, w_g\}$ and no more than $\frac{t}{2s}|D|$ words of D (we get an upper bound by multiplying the size of each set by the number of sets). This leaves a collection of $\left(1 - \frac{t}{2s}\right)|D|$ equally likely words, so $p_1 = \left(1 - \frac{t}{2s}\right)^{-1}|D|^{-1} = (1 - O(1/\log(g)))^{-1}|D|^{-1} = (1 + o(1))|D|^{-1}$. In contrast, if all covers of our Set-Cover instance have size $s' \geq c \cdot t \ln(g)$ (where, because we assume some cover has size $\leq s$, we have $s' \leq s$,) then any collection of rules that bans all words in $\{w_1, w_2, \ldots, w_g\}$ must also ban at least $\frac{s'}{8s}|D|$ words out of D. This leaves at most $(1 - \Omega(1))|D|$ words in D and

so $p_1 \geq (1-\Omega(1))^{-1}|D|^{-1}$. Denoting the latter constant as c_0^{-1}, we have that any $c_0 - \epsilon$ approximation of the optimal banning-policy indicates the existence of a cover of cardinality $< c \cdot t \ln(g)$. $\square$

4.3. Positive Rules: Hardness of Approximation for Large k

While we can show that it is possible to optimize $p(k, \mathcal{A})$ in the singleton rules setting our result does not extend to the more general positive rules setting. We are able to show that it is NP-Hard to compute $\arg\min_{S\subseteq[m]} p(k, \mathcal{A}_S)$. However, our reduction does not imply approximation hardness so we cannot rule out the existence of a PTAS.

THEOREM 4.4. *Unless $P = NP$ there is no polynomial time algorithm (in N, m, n) which outputs $\arg\min_{S\subseteq[m]} p(k, \mathcal{A}_S)$ in the positive rules setting and the normalization model.*

The theorem's proof is relegated to the full version of this paper [Blocki et al. 2013].

5. EFFICIENT SAMPLING ALGORITHMS

In a sense, our complexity results are not "realistic", and in particular in the ranking model our positive algorithmic results assume access to each user's full preferences. Moreover, some algorithms are allowed to run in polynomial time in the number of passwords N, which can be huge. In this section we use our complexity results as guidelines in the design of practical sampling algorithms.

In more detail, we are given oracle access to rules $R_1, ..., R_m$ (e.g., we can ask whether or not a password $w \in R_i$) and we are allowed to sample from the distribution induced by the password composition policy $\mathcal{A}_S$ for any $S \subseteq [m]$. Less formally, a sample is equivalent to asking a random user what her favorite password is given the current policy.

We will work in the more general ranking model, so there is essentially only one positive result we can build on: Theorem 3.2, a polynomial time algorithm for constant k in the positive rules setting. When adapting this algorithm to the sampling setting, we cannot expect it to work perfectly due to the inherent uncertainty of this domain. Instead we expect the algorithm to find an ϵ-optimal password composition policy with probability at least $1-\delta$, for any given ϵ and δ. Crucially, the number of samples must not depend on the number of passwords N, and must have a polynomial dependence on the other parameters.

Formally, we let $S^* \subseteq [m]$ denote the optimal collection of positive rules to activate (for all $S \subseteq [m]$, $p(1, \mathcal{A}_{S^*}) \leq p(1, \mathcal{A}_S)$). Our goal is to find a $(1, \epsilon)$-approximation $S \subseteq [m]$ to $p(1, \mathcal{A}_{S^*})$, that is, S such that $p(1, \mathcal{A}_S) \leq p(1, \mathcal{A}_{S^*}) + \epsilon$, with probability $1 - \delta$.

We first present Algorithm 5 that achieves our goal for $k = 1$; this algorithm is an adaptation of Algorithm 3.

THEOREM 5.1. *Algorithm 5 runs in polynomial time in $m, 1/\epsilon, 1/\delta$, requires $O\left(m \log(m/\delta)/\epsilon^2\right)$ samples and returns a $(1, \epsilon)$-approximation $S \subseteq \{1, ..., m\}$ of $p(1, \mathcal{A}_{S^*})$ with probability at least $1 - \delta$.*

PROOF. Let

$$BAD_i = \left\{ \exists w \in \mathcal{A}_{S_i} \;\middle|\; \left| \frac{s_w}{s} - \Pr[w \mid \mathcal{A}_{S_i}] \right| \geq \epsilon/2 \right\} ,$$

denote the event that our probability estimates are off during iteration i. Claim 5.2 bounds the probability of any bad event. The proof of Claim 5.2 can be found in the full version of this paper [Blocki et al. 2013]. The proof involves bucketing the passwords based on their probability, applying Chernoff Bounds to upper bound the probability of a bad estimate for our passwords in each bucket, and repeatedly applying union bounds.

CLAIM 5.2. $\Pr[\exists i, BAD_i] \leq \delta$.

Algorithm 5 SampleAndEliminate

Positive Rules: $R_1, ..., R_m$
Input: ϵ, δ
Initialize: $S_0 \leftarrow [m], i \leftarrow 0$
$s \leftarrow \frac{100}{\epsilon^2} \log\left(\frac{4m}{\epsilon\delta}\right)$
while $S_i \neq \emptyset$ **do**
  Sample: Draw samples $w_1, ..., w_s$ according to the distribution $\Pr[w \mid \mathcal{A}_{S_i}]$
  $W \leftarrow \{w_1, ..., w_s\}$
  $s_w \leftarrow |\{j \mid w_j = w\}|$ for each $w \in W$.
  $w^* \leftarrow \arg\max\{s_w \mid w \in W\}$ ▷ w^* is the most frequently sampled password
  $\hat{p}_i \leftarrow \frac{s_{w^*}}{s}$ ▷ $\hat{p}_i$ is our estimation of $\Pr[w^* \mid \mathcal{A}_{S_i}]$
  if $\hat{p}_i \leq \epsilon/2$ **then return** S_i ▷ The current solution is already sufficiently good
  else
    $S_{i+1} \leftarrow S_i - \{j \mid w^* \in S_j\}$ ▷ Deactivate all rules that contain w^*
    $i \leftarrow i + 1$
return S_{i^*} where $i^* = \arg\max\{\hat{p}_j | j \leq m\}$.

For the rest of the analysis we assume that no bad event occurs. Let $p^* = \min_{S \subseteq [m]} p(1, \mathcal{A}_S)$ and suppose that $A_{S^*} \subseteq A_{S_i}$. Clearly, this is true when $i = 0$. If $\hat{p}_i \geq \epsilon/2 + p^*$ then $\Pr[w^* \mid \mathcal{A}_{S^*}] \geq \Pr[w^* \mid \mathcal{A}_{S_i}] > p^*$ so that $w^* \notin A_{S^*}$. Hence, $A_{S^*} \subseteq A_{S_{i+1}}$ and the property is maintained for at least one more iteration. If instead $\hat{p}_i < \epsilon/2 + p^*$ then we have $\hat{p}_{i^*} \leq \hat{p}_i \leq p^* + \epsilon/2$ so for each $w \in \mathcal{A}_{S_{i^*}}$ we have $\Pr[w \mid \mathcal{A}_{S_{i^*}}] \leq p^* + \epsilon$. We conclude that the solution S_{i^*} is a $(1, \epsilon)$-approximation. □

We next explain how to extend Algorithm 2 to $(1, \epsilon)$-approximate the optimal $p(k, \mathcal{A}_S)$ for any constant k.

THEOREM 5.3. *There is an algorithm which runs in polynomial time (in $m, 1/\epsilon, \delta$), takes a polynomial number of samples, and returns a $(1, \epsilon)$-approximation $S \subseteq [m]$ of $p(k, \mathcal{A}_{S^*})$ with probability at least $1 - \delta$.*

PROOF SKETCH. To extend Algorithm 2 to $(1, \epsilon)$-approximate $p(k, \mathcal{A}_S)$ for constant k we need one more idea. We cannot simply obtain a reduced password space $\hat{P}$ by reducing preference lists because we can only sample from our distribution. Notice that for any $S \subseteq [m]$ such that $i \in S$ we have $\Pr[w \mid \mathcal{A}_S] \leq \Pr\left[w \mid \mathcal{A}_{\{i\}}\right]$ so to obtain a $(1, \epsilon)$-approximation it is sufficient to limit our attention to passwords in the following set

$$\hat{P} = \left\{ w \;\middle|\; \exists i, \Pr\left[w \;\middle|\; \mathcal{A}_{\{i\}} \geq \frac{\epsilon}{k}\right] \right\} .$$

We can obtain a superset of $\hat{P}$ by sampling. For each positive rule R_i we draw s independent samples from the distribution $\mathcal{A}_{\{i\}}$ and set

$$T_i = \left\{ w \;\middle|\; \frac{s_w}{s} > \frac{\epsilon}{2k} \right\} .$$

Intuitively, a password w is included in T_i if and only if our estimated probability is sufficiently large. Let $T = \bigcup_i T_i$. For a sufficiently large sample size $s = O(poly(m, k, 1/\epsilon, 1/\delta))$ we can apply Chernoff Bounds to argue that with probability $1 - \delta$ (1) $|T|$ is small, i.e., $O(poly(m, k, 1/\epsilon, 1/\delta))$, and (2) $T \supset \hat{P}$. □

6. EXPERIMENTS

To demonstrate how our ideas could apply in a real-world scenario, we simulated runs of Algorithm 5 by sampling with replacement from the RockYou leaked password set [Imperva

2010]. The set contains over 32 million passwords with a frequency distribution similar to that of many other password sets [Bonneau 2012]. Note that all results presented here are limited by the dataset and assume the normalization model. Working in the normalization model is crucial because we cannot ask the RockYou users for their preferred password under a specific policy; an initial distribution over $\mathcal{P}$ — which is available to us — is sufficient though, because it induces a distribution for any policy $\mathcal{A}$.

We selected 21 positive rules that mirror commonly used password composition rules that are used in practice, and looked at sample sizes s of 100, 500, 1000, 5000, and 10000. The rules included length requirements, character class requirements, combinations of requirements, a dictionary check, etc. (See the full version [Blocki et al. 2013] for a complete listing of the rules we selected.) For each run with a particular value of s, the algorithm returns a policy $\mathcal{A}_S$ for which we can measure $p\,(1, \mathcal{A}_S)$ in the original dataset and compare with the optimal $p\,(1, \mathcal{A}_{S^*})$, determined from running Algorithm 3 on the original dataset. We performed 500 runs for each of the five values of s.

To gain an understanding of how policies based on negative rules perform, we took the complement of the 21 positive rules selected above to get 21 negative rules. We then determined the optimal negative rules policy by calculating $S^* = \arg\min_{S \subseteq [m]} p\,(1, \mathcal{A}_S)$ via brute-force. This was required because we have no equivalent to Algorithm 3 for negative rules. With this baseline in hand, we designed two naïve algorithms, similar in spirit to Algorithm 5. There are multiple ways to discard a password in the negative rules setting, and one algorithm makes this decision randomly while the other bans the smallest subset as determined from the current sample. Again, 500 runs were performed for each $s \in \{100, 500, 1000, 10000, 50000\}$.

6.1. Baselines

We examined several baselines for comparison with our algorithm. Table II shows these baselines, the probability of the most frequent password in the resulting policy, and the optimal policy as a union or intersection of rules (for clarity, the complement of the union of negative rules is shown as the intersection of positive rules).

As shown in Table II from the means across policies, randomly selecting a policy from the power set of rules can be worse than having no policy. The "one rule maximum" baseline was selected because, if decided based on sampling, only m distributions need be sampled. Our efficient algorithm requires the same amount of sampling, but can find the optimal policy over $S \subseteq [m]$ rather than $S \in \{1, ..., m\}$. Also of interest is the optimal policy with negative rules, which is over 3x better than the optimal policy with positive rules. However, as shown in the following section, the performance of our sampling algorithms with negative rules was far worse than in the positive rules setting.

Table II: Baseline probabilities for the RockYou dataset

Baseline	$\boldsymbol{p\,(1, \mathcal{A}_S)}$	$\boldsymbol{S}$
Mean across negative rules policies	1.3×10^{-2}	
Mean across positive rules policies	1.0×10^{-2}	
All passwords allowed (no policy)	9.2×10^{-3}	
One positive rule ($S \in \{1, ..., m\}$)	6.8×10^{-4}	8 chars, 1 upper, 1 digit
Optimal policy with positive rules	4.4×10^{-4}	14 chars OR 2 symbols OR 8 chars, 1 upper, 1 digit
Optimal policy with negative rules	1.4×10^{-4}	10 chars AND 2 digits AND 1 symbol AND 1 lowercase AND not in dictionary

Table III: Performance of Sampling Algorithms with Positive Rules

Sample Size	mean $p(1, \mathcal{A}_S)$	min $p(1, \mathcal{A}_S)$	% Optimal
100	6.8×10^{-3}	1.2×10^{-3}	
500	9.7×10^{-4}	$\mathbf{4.4\times10^{-4}}$	2%
1000	9.5×10^{-4}	$\mathbf{4.4\times10^{-4}}$	10%
5000	6.0×10^{-4}	$\mathbf{4.4\times10^{-4}}$	14%
10000	5.7×10^{-4}	$\mathbf{4.4\times10^{-4}}$	19%

Table IV: Performance of Sampling Algortihms with Negative Rules

	Random Decision		Ban Smallest	
Sample Size	mean $p(1, \mathcal{A}_S)$	min $p(1, \mathcal{A}_S)$	mean $p(1, \mathcal{A}_S)$	min $p(1, \mathcal{A}_S)$
100	6.8×10^{-3}	1.2×10^{-3}	7.2×10^{-3}	2.3×10^{-3}
500	4.4×10^{-3}	6.3×10^{-4}	9.0×10^{-3}	2.3×10^{-3}
1000	4.3×10^{-3}	4.5×10^{-4}	8.6×10^{-3}	2.3×10^{-3}
5000	6.3×10^{-3}	4.5×10^{-4}	9.2×10^{-3}	9.2×10^{-3}
10000	7.2×10^{-3}	4.5×10^{-4}	9.2×10^{-3}	9.2×10^{-3}

6.2. Performance

In the positive rules setting (see Table III), the algorithm performed extremely well even at moderate sample sizes. The average policy selected with $s = 500$ was almost 10x better than having no policy. At $s = 1000$, the optimal policy was found 10% of the time (50 out of 500 times).

In the negative rules setting (see Table IV), however, neither algorithm found the optimal policy. The "Ban Smallest" heuristic, when faced with a choice between multiple subsets that contain the most likely password, decides to ban the smallest available subset, disrupting the space the least. This might seem like an intuitively good choice but, in fact, it fails to find a better policy than the empty set at large sample sizes. The randomized algorithm does better (it cannot actually do worse) but still has much worse average case performance than using our efficient algorithm with positive rules.

7. DISCUSSION

We conclude by discussing some key points.

Where do the rules comes from? Throughout the paper we have assumed that the rules (whether positive or negative) are given as part of the input; it is not up to us to find these rules. Our experiments indicate that a collection of intuitive and practical rules can already give very good results on real data. However, the question of deciding which rules should be added to our collection is outside the scope of this paper. Much like the problem of feature selection, it is an interesting problem with real-life implications, which we suspect will be very difficult in practice.

Alternate policy goals. Our goal [Boztas 1999] has been to minimize $p(k, \mathcal{A}_S)$. Intuitively, $p(k, \mathcal{A}_S)$ represents the probability that an adversary with no background knowledge can successfully guess the password of a randomly selected user in k tries. A small value of k optimizes security guarantees against an online guessing attack in which the adversary is locked out after k failed attempts to login. A much larger value of k (e.g., 2^{32}) is necessary to optimize security against an adversary who has obtained the cryptographic hash of a password and is able to mount a brute-force dictionary attack [Seeley 1989]. However, the optimal solutions for $p(1, \mathcal{A}_S)$ and $p\left(2^{32}, \mathcal{A}_S\right)$ might be completely different. One stronger goal that we might hope to achieve is to optimize both goals simultaneously. More formally, can we find a policy $S \subseteq [m]$ such that for every $S' \subseteq [m]$ and every $k \leq N$ we have $p(k, \mathcal{A}_S) \leq c \cdot p(k, \mathcal{A}_{S'})$ for some constant c? Unfortunately, the answer is no. For any

constant c this universal approximation goal is impossible to satisfy in the ranking model — see Theorem B.1 in the full version of this paper [Blocki et al. 2013].

Other natural goals include α-work factor [Pliam 2000] and a refinement called α-guesswork [Bonneau 2012] (e.g., maximize the total number of guesses needed to compromise α-fraction of the accounts). While α-guesswork is an useful metric to analyze the security of 70 million Yahoo passwords [Bonneau 2012], it may not be a desirable optimization goal for the organization because it might allow the adversary to crack up to $\alpha - \epsilon$-fraction of the accounts with relatively few guesses.

Another interesting direction is to account for an adversary with basic background information about the user (e.g., e-mail address, username, birthday). It may not always be realistic to assume that the adversary has no background knowledge because the adversary can often easily obtain some background knowledge about a user by searching for publicly available information on the internet. One approach might be to design a rule R to specify different passwords for different users (e.g., the set of passwords that contain the username or birthday of the user).

Open Questions. While we were able to prove several hardness results about finding the optimal password composition policy in the negative rules setting, it is possible that these hardness results could be circumvented by making mild (hopefully realistic) assumptions about the underlying password distribution or the rules $R_1, ..., R_m$. Are there efficient algorithms to optimize $p(k, \mathcal{A}_S)$ in the negative rules setting given realistic assumptions? It is also possible that mild realistic assumptions could be used to circumvent the impossibility result of Theorem B.1 [Blocki et al. 2013], and design a universal approximation algorithm.

There are also several interesting technical questions that remain open:

(1) Normalization model with negative rules: Can we efficiently c-approximate $p(1, \mathcal{A}_{S^*})$ for any constant c? Is there a sub-exponential algorithm (in m) to compute $p(1, \mathcal{A}_{S^*})$?
(2) Ranking model with positive rules: Can we efficiently c-approximate $p(k, \mathcal{A}_{S^*})$ for some constant c when k is a parameter?

The future. There is a real need for a principled approach to optimizing password composition policies. We have taken a first step in this direction by providing an intuitive theoretical model and showing that it leads to algorithms that perform well on real data. We can only hope that our work will spark a fundamentally new interaction between theory and practice in passwords research.

REFERENCES

Alon, N., Moshkovitz, D., and Safra, S. 2006. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms 2,* 2, 153–177.

Arora, S., Barak, B., and Steurer, D. 2010. Subexponential algorithms for unique games and related problems. In *Proc. of FOCS*. 563–572.

Austrin, P., Khot, S., and Safra, M. 2011. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory of Computing 7,* 1.

Blocki, J., Komanduri, S., Procaccia, A. D., and Sheffet, O. 2013. Optimizing password composition policies. *CoRR abs/1302.5101.*

Bonneau, J. 2012. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Proc. of Oakland*. 538–552.

Bonneau, J. and Xu, R. 2012. Character encoding issues for web passwords. In *Web 2.0 Security & Privacy.*

Boztas, S. 1999. Entropies, guessing, and cryptography. Technical report, Department of Mathematics, Royal Melbourne Institute of Technology.

Burr, W. E., Dodson, D. F., and Polk, W. T. 2006. Electronic authentication guideline. *NIST Special Publication 800-63*.

Clair, L., Johansen, L., Enck, W., Pirretti, M., Traynor, P., McDaniel, P., and Jaeger, T. 2006. Password exhaustion: Predicting the end of password usefulness. *Proc. of ICISS*, 37–55.

Cormode, G. and Muthukrishnan, S. 2005. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms 55,* 1, 58–75.

Doel, K. 2012. Scary logins: Worst passwords of 2012 and how to fix them. Retrieved 1/21/2013.

Florêncio, D. and Herley, C. 2010. Where do security policies come from. In *Proc. of SOUPS*. 10.

Fossi, M., Johnson, E., Turner, D., Mack, T., Blackbird, J., McKinney, D., Low, M. K., Adams, T., Laucht, M. P., and Gough, J. 2008. Symantec report on the undergorund economy. Retrieved 1/8/2013.

Imperva. 2010. Consumer password worst practices. Retrived 1/22/2013.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum, 85–103.

Khot, S. 2002. On the power of unique 2-prover 1-round games. In *Proc. of STOC*. 767–775.

Khot, S. and Regev, O. 2008. Vertex cover might be hard to approximate to within 2-ε. *Journal of Computer and System Sciences 74,* 3, 335–349.

Komanduri, S., Shay, R., Kelley, P., Mazurek, M., Bauer, L., Christin, N., Cranor, L., and Egelman, S. 2011. Of passwords and people: measuring the effect of password-composition policies. In *Proc. of CHI*. 2595–2604.

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

Malone, D. and Maher, K. 2012. Investigating the distribution of password choices. In *Proc. of WWW*. 301–310.

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

Scarfone, K. and Souppaya, M. 2009. NIST special publication 800-118: Guide to enterprise password management (draft).

Schechter, S., Herley, C., and Mitzenmacher, M. 2010. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *Proc. of HotSec*. 1–8.

Seeley, D. 1989. Password cracking: A game of wits. *Communications of the ACM 32,* 6, 700–703.

Weir, M., Aggarwal, S., Collins, M., and Stern, H. 2010. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proc. of CCS*. 162–175.

Witty, R., Brittain, K., and Allen, A. 2004. Justify identity management investment with metrics. Gartner Group report.