Towards a Rigorous Statistical Analysis of Empirical Password Datasets

Jeremiah Blocki*
Purdue University
jblocki@purdue.edu

Peiyuan Liu*
Purdue University
liu2039@purdue.edu

***Abstract*—A central challenge in password security is to characterize the attacker's guessing curve i.e., what is the probability that the attacker will crack a random user's password within the first G guesses. A key challenge is that the guessing curve depends on the attacker's guessing strategy and the distribution of user passwords both of which are unknown to us. In this work we aim to follow Kerckhoffs's principle and analyze the performance of an optimal attacker who knows the password distribution. Let λ_G denote the probability that such an attacker can crack a random user's password within G guesses. We develop several statistically rigorous techniques to upper and lower bound λ_G given N independent samples from the unknown password distribution $\mathcal{P}$. We show that our upper/lower bounds on λ_G hold with high confidence and we apply our techniques to analyze eight large password datasets. Our empirical analysis shows that even state-of-the-art password cracking models are often significantly less guess efficient than an attacker who can optimize its attack based on its (partial) knowledge of the password distribution. We also apply our statistical tools to re-examine different models of the password distribution i.e., the empirical password distribution and Zipf's Law. We find that the empirical distribution closely matches our upper/lower bounds on λ_G when the guessing number G is not too large i.e., $G \ll N$. However, for larger values of G our empirical analysis rigorously demonstrates that the empirical distribution (resp. Zipf's Law) overestimates the attacker's success rate. We apply our statistical techniques to upper/lower bound the effectiveness of password throttling mechanisms (key-stretching) which are used to reduce the number of attacker guesses G. Finally, if we are willing to make an additional assumption about the way users respond to password restrictions, we can use our statistical techniques to evaluate the effectiveness of various password composition policies which restrict the passwords that users may select.**

***Index Terms*—password distribution, password cracking, password dataset, theoretical bounds, statistical analysis, password policies**

I. Introduction

Understanding and characterizing the distribution over user chosen passwords is a key challenge in the field of cybersecurity with important implications towards the development of robust security policies. How expensive does a password hashing algorithm need to be to deter an offline brute-force attacker? Can we strengthen the password distribution by imposing restrictions, e.g., requiring passwords to include numbers and/or capital letters, on the passwords that users can pick? If so are the security gains significant enough to justify the usability costs? Characterizing the attacker's guessing curve is a fundamental challenge which is central to addressing the above questions. In particular, we would like to know the probability that an attacker will crack a random user's password after G attempts as G ranges from small (online password spraying attacker) to large (offline attack). As a concrete motivating application suppose that an organization's current authentication policy locks a user account if there are more than $G = 100$ incorrect login attempts within a 30 day period. The organization is considering adopting a stricter lockout policy which locks down the account after $G = 50$ incorrect logins within 30 days, but will only adopt such a change if the policy change substantially reduces the risk posed by an online attacker. An immediate challenge is that the answer to these questions depends on the distribution $\mathcal{P}$ over user passwords as well as the attacker's guessing strategy both of which are unknown.

One natural attempt to address the above questions is to fix a state-of-the art password cracking model M and assume that the attacker uses this model to generate password guesses. Now our organization might estimate that adopting the stricter lockout policy (resp. doubling the cost of the password hash function) reduces the success rate of an online (resp. offline) password attacker by $\lambda_{G,M} - \lambda_{G/2,M}$ where $\lambda_{G,M}$ denotes the probability that an attacker will crack a randomly sampled password $pwd \leftarrow \mathcal{P}$ within G guesses when using model M. Since the distribution $\mathcal{P}$ is unknown the organization cannot compute $\lambda_{G,M}$ directly. However, it is easy to approximate $\lambda_{G,M}$ empirically given a few samples $S = (s_1, \ldots, s_N)$ from the unknown password distribution $\mathcal{P}$.

One downside to the above approach is that our policy decision is dependent on the specific password cracking model M. Password cracking models can be developed using a variety of different techniques including: Neural Networks [1], Probabilistic Context Free Grammars [2], [3], Markov Models [4]–[7] and sophisticated rule lists [8] used in password cracking software such as John the Ripper (JtR) [9] and Hashcat [10]. Basing policy decisions on a model specific guessing curve $\lambda_{G,M}$ may be unwise as this curve can vary greatly from model to model. In particular, we do not know which model M the attacker will use and it is plausible that the attacker's model will be more sophisticated than any publicly available model. Furthermore, the attacker might continue to improve the cracking model M over time.

In an attempt to address this challenge prior work (e.g., [1], [11]) proposed using the minimum guessing number

*The authors are listed in alphabetical order.

heuristic to approximate the performance of an attacker who may have a more advanced password cracking model with more training data and/or more sophisticated rule lists/dictionaries. Under this heuristic we use multiple different password cracking models (Neural Networks [1], Probabilistic Context Free Grammars [2], [3], Markov Models [4]–[7]) to estimate the attacker's guessing curve. In particular, if a password pwd is cracked within G guesses under *any* of these models then, under the minimum guessing number heuristic, we assume that the real world attacker would also crack this password within G guesses. Our analysis indicates that, even we apply the minimum guessing number heuristic, we can still significantly underestimate the performance of a real world attacker.

Kerckhoffs's principle states one should design systems under the assumption that the attacker will eventually gain full familiarity with them. Applying this principal within the context of password security our organization should assume that the attacker knows the password distribution and consider the guessing curve λ_G of such an attacker when making policy decisions. Here, λ_G denotes the cumulative probability mass of the top G passwords in the (unknown) password distribution $\mathcal{P}$ i.e., the probability that an attacker who knows the distribution can guess a random password $pwd \leftarrow \mathcal{P}$ within G guesses. While a real world attacker may not have perfect knowledge of the password distribution, we should expect that password cracking models will improve over time and performance of the real world attacker may even approach the ideal guessing curve λ_G in the limit. Thus, when making password policy decisions the conservative option is to follow Kerckhoffs's principle and consider the guessing curve of the ideal attacker who knows the password distribution instead of using heuristics to guesstimate the performance of an unknown attacker. An additional advantage of following Kerckhoffs's principle is that the policy recommendations will be stable i.e., they would not need to be reevaluated every time a more sophisticated password cracking model is developed. However, an immediate challenge is that the distribution $\mathcal{P}$ is unknown to us making it impossible to directly compute λ_G.

One common heuristic to estimate λ_G is to use the empirical distribution derived from a breached password dataset e.g., as in [12]–[18]. In particular, given a dataset S of N user passwords we can let $\hat{f}_i$ denote the frequency of the ith most common password pwd in the sample S. Assuming that the samples S were drawn iid from our password distribution $\mathcal{P}$ we can then estimate that an attacker making G guesses per account will crack a random user's password with probability $\hat{\lambda}_G = \sum_{i=1}^{G} \hat{f}_i/N$. A downside to this approach is that the empirical estimate $\hat{\lambda}_G$ can substantially overestimate the true value λ_G. Indeed we will always have $\hat{\lambda}_G = 1$ whenever $G \geq N$ even when the password distribution $\mathcal{P}$ is very strong. For example, suppose that $\mathcal{P}$ is the uniform distribution over strong 56-bit passwords so that an attacker will crack a random user's password with probability at most $\lambda_G = 2^{-56}G$ within G guesses. However, if we use the empirical distribution with a very large sample size $N = 2^{33}$ (larger than the global population) we would still have $\hat{\lambda}_N = 1 \gg 2^{-23} = \lambda_N$. Thus, it may also be undesirable to base policy decisions on the empirical distribution $\hat{\lambda}_G$ — especially in offline attack settings where G can be large.

In this paper we address the following questions: Can we confidently derive accurate upper and lower bounds on λ_G? When (if ever) can we use the empirical distribution $\hat{\lambda}_G$ to accurately model the real distribution? How guess efficient are state-of-the-art password cracking models?

Despite their shortcomings and many attempts to replace them passwords remain entrenched as the dominant form of authentication on the internet and are likely to play a critical role in the foreseeable future because they are easy to use, easy to deploy and users are already familiar with them [19]. Thus, characterizing the distribution λ_G of user chosen passwords will continue to be a important challenge in the field of cybersecurity.

A. Our Contributions

We develop a statistical framework to upper and lower bound the guessing curve (λ_G) of an ideal attacker who knows the password distribution. We stress that we make no a priori assumptions about the shape of the password distribution. Instead, to apply our statistical framework we only require independent samples $S = (s_1, \ldots, s_n)$ from the unknown password distribution $\mathcal{P}$. All of our bounds can be shown to hold with high probability over the randomly selected password samples. In practice, the password samples could either be obtained from prior breached password datasets or (ideally) an organization could adapt prior work [15], [16] to obtain samples from its own users in a secure and privacy preserving manner — see discussion in Section V-D. We apply our techniques to analyze several empirical password datasets and illustrate how the upper/lower bounds can be used to guide policy decisions.

Upper Bounds. We develop two techniques to obtain high confidence upper bounds on the guessing curve of an ideal attacker using the empirical password distribution and linear programming. We first show that λ_G is (with high probability) upper bounded by the empirical estimate $\hat{\lambda}_G$ i.e., with high probability we have $\lambda_G \leq \hat{\lambda}_G + \epsilon$ for a very small constant $\epsilon > 0$. Empirical analysis shows that this upper bound is often tight when G ranges from small to moderately large. However, the upper bound becomes less and less tight as G increases and devolves into the trivial upper bound $\lambda_G \leq 1$ once $G \geq \mathbf{Distinct}(S)$ exceeds the number of distinct passwords in our sample S. Thus, we develop a second approach to upper bound λ_G when G is large using linear programming (LP). Our LP approach adapts techniques of Valiant and Valiant [20] for estimating properties of a distribution when the number of samples is smaller than the support of the distribution. Intuitively, our LP searches for a distribution which maximizes λ_G subject to the constraint that the distribution is "sufficiently consistent" with our sample S. With high probability, the real distribution $\mathcal{P}$ will also be consistent with all of our linear constraints so we will obtain a valid upper bound by maximizing over all consistent distributions. Empirical analysis shows that our LP upper bounds are superior for sufficiently

large G and that we often obtain non-trivial upper bounds even when $G \geq \mathbf{Distinct}(S)$.

Lower Bounds. We develop three techniques to obtain high confidence lower bounds on the guessing curve (λ_G) of an ideal attacker: sampling, model-sampling hybrid and linear programming. Additionally, we show how to adapt prior analysis of Blocki et al. [14] to obtain another high confidence lower bound. Our first sampling technique is a simple algorithm inspired by Good-Turing frequency estimation. The algorithm randomly partitions our sample S into two components D_1 and D_2 of size $N-d$ and d respectively, builds a dictionary $T(D_1, G)$ containing the G most common passwords in D_1, and computes the fraction of passwords in D_2 which appear in this dictionary — we remark that a real world attacker who obtains the samples D_1 can also build the dictionary $T(D_1, G)$ and then use this dictionary to crack other passwords. Empirical analysis demonstrates that the resulting lower bounds are nearly tight for smaller guessing numbers (e.g., $G \leq 10^6$), but the lower bound will plateau at $\approx 1 - \frac{\mathbf{Unique}(S)}{N}$ once $G > \mathbf{Distinct}(S)$, where $\mathbf{Unique}(S)$ counts the number of passwords which appear exactly once in our sample S. We provide two additional techniques to push past this barrier and derive stronger lower bounds when G is large. First, we can adapt the linear program described earlier to instead search for a distribution that *minimizes* λ_G subject to the constraint that the distribution is "sufficiently consistent" with our sample S. Empirical analysis confirms that this approach generates tighter lower bounds when $G > \mathbf{Distinct}(\mathbf{S})$ allowing us to push past the $1 - \frac{\mathbf{Unique}(\mathbf{S})}{N}$ barrier. Finally, we show how a password cracking model M can be used to extend the first lower bound when $G > \mathbf{Distinct}(D_1)$ using a hybrid approach: as before we partition our sample S into two components D_1 and D_2 and then we output the fraction of passwords in D_2 which *either* appear in D_1 or that appear in the top $G - \mathbf{Distinct}(D_1)$ guesses generated by our model M. Empirical analysis shows that this combined bound can improve on our prior lower bounds when G is very large.

Empirical Analysis. We apply our theoretical upper and lower bounds to analyze eight password datasets (i.e. Yahoo!, RockYou, 000webhost, Neopets, Battlefield Heroes, Brazzers, Clixsense, CSDN), and we compare our bounds with state-of-the-art password cracking techniques such as Markov models [4]–[7], Probabilistic Context-free Grammars (PCFG) [2], [3], neural networks [1], John the Ripper (JtR) and Hashcat. We find that our new techniques for lower bounding λ_G significantly improve upon prior work of Blocki et al. [14] e.g., for RockYou dataset with guessing budget $G = 1.3 \times 10^8$ our lower bounds are $\lambda_G \geq 62.64\%$ (sampling) and $\lambda_G \geq 72.70\%$ (linear programming) in comparison to the weaker lower bound $\lambda_G \geq 53.95\%$ obtained from [14]. Our empirical analysis also shows that, for smaller values of G, our upper and lower bounds on λ_G are very close and that the empirical estimate $\hat{\lambda}_G$ is sandwiched between these two values. This provides an answer to our second question i.e., the empirical guessing curve $\hat{\lambda}_G$ closely approximates the real guessing curve as long as the guessing number G is not too large. By contrast, for some larger values of G we can demonstrate with high confidence that the empirical guessing curve $\hat{\lambda}_G$ significantly overestimates λ_G indicating that the empirical distribution should not be used to approximate the real guessing curve in these settings. We also compare our upper/lower bounds to CDF-Zipf curves fit to the empirical dataset [21] and, we identify cases where the CDF-Zipf curve overestimates λ_G by *at least* 12%.

We find that our lower bounds on λ_G are often significantly higher than the guessing curves obtained using state-of-the-art password cracking models [1], [2], [8], [22] indicating that there is still room to develop improved password cracking models which are more guess efficient. For example, an attacker making $G \approx 8.4$ million guesses per account would crack *at most* 14% of passwords in the 000webhost using any of the state-of-the-art password cracking models we analyzed. By contrast, our high confidence[1] lower bounds show that an attacker who knows the password distribution would crack *at least* 39.16% of 000webhost passwords. Similar observations held for other datasets. This provides compelling statistical evidence even the most sophisticated password cracking models still have a large room for improvement in guess efficiency. There has been a push towards designing moderately expensive password hashing algorithms to discourage offline attacks e.g., see [14], [23]–[26]. If password guessing becomes more expensive then attacker will have additional incentive to develop guess efficient models.

Implications for Password Policies. We apply our statistical techniques to quantify the security benefits of adopting a stricter lockout policy and/or increasing the cost of the password hash function by a multiplicative factor b i.e., such that cost of making G/b password guesses after this cost increase is identical to the cost of making G guesses beforehand. In particular, we derive upper and lower bounds on the security benefit $(\lambda_G - \lambda_{G/b})$ of such policy changes for various values of b. We also apply our statistical techniques to analyze several prominent password composition policies. For example, the Yahoo! password frequency corpus S can be divided into two sets S_0 and S_1 representing passwords picked before/after Yahoo! adopted six character minimum policy for user passwords. We can then apply our statistical techniques to compare the distributions of user passwords with and without this restriction. If we are willing to adopt the normalized probabilities model [27], a heuristic assumption about the way user's react to password composition policies, then we can apply our statistical techniques to quantify the performance of arbitrary password composition policies using other password datasets such as 000webhost and RockYou — see discussion and analysis in Section V.

B. Related Work

Password Hashing and Memory Hard Functions: Key-stretching was proposed as early as 1979 [28] as a way to

[1]The probability of an errant lower (or upper) bound can be upper bounded by a small constant δ. In our experiments we tuned our parameters such that $\delta \leq 0.01$.

protect lower-entropy passwords against offline brute force attacks by making the password hash function moderately expensive to compute. Password hashing algorithms such as BCRYPT [29] and PBKDF2 [30] use hash iteration to control guessing costs, but are potentially vulnerable to an attacker who uses FPGAs or Application Specific Integrated Circuits (ASICs) to dramatically reduce guessing costs. Blocki et al. [14] argued that hash iteration alone cannot provide sufficient protection for user password without introducing an unacceptably long delay during the authentication process i.e., minutes. Memory hard functions such as such as scrypt [26], Argon2 [24] or DRSample [25] are designed to force an attacker to allocate large amounts of memory for the duration of computation and are believed to be ASIC resistant. When attempting to tune the cost parameters of our key-stretching algorithm it will be important to understand the password distribution λ_G i.e., a defender might want to compute $\lambda_G - \lambda_{G/2}$ to decide whether or not doubling the cost parameter would substantially reduce the $\%$ of passwords cracked by an offline attacker. We also note that as organizations start to use moderately expensive memory hard password hash functions like scrypt [26], Argon2 [24] or DRSample [25] offline attackers will have stronger incentives to develop guess efficient password cracking models. Thus, it will be desirable to base security decisions on λ_G instead of using a password cracking model.

Offline Password Cracking Models and Defenses: Offline password cracking has been studied for decades. Researchers have proposed many password cracking algorithms based on probabilistic password models such as Probabilistic Context-free Grammars (PCFG) [2], [3], Markov models [4]–[7], and neural networks [1]. Monte-Carlo strength estimation [22] is a tool which allows us to efficiently approximate the guessing number of a given without requiring the defender to simulate the full attack. Liu et al. [8] developed tools to estimate guessing numbers for software tools such as John the Ripper (JtR) [9] and Hashcat [10] which are used more frequently by real world attackers. Juels and Rivest [31] suggested the use of honeywords (fake passwords) to help detect offline attacks. Compromised Credential Checking services such as HaveIBeenPwned and Google Password Checkup can be used to help alert users when one of their passwords have been breached [32], [33]. Distributed password hashing e.g., [34] ensures that the information needed to evaluate the password hash function is distributed across multiple servers so that a hacker who breaks into any individual server will not be able to mount an offline attack. Multifactor authentication provides another defense against password cracking attacks [35], [36].

Strengthening the Password Distribution: A large body of research has focused on encouraging (or forcing) users to pick stronger passwords. Password composition policies [37]–[39] require users to pick passwords that comply with particular requirements e.g., passwords must at least one number and one upper case letter or passwords must be at least 8 characters long. Password composition policies often induce a substantial usability burden [37], [40] and can often be counter-productive [27], [39]. Telepathwords [41] is a password meter which encourages users to select stronger passwords by displaying realtime predictions for the next character that the user will type. Bonneau and Schecter [42] introduced the notion of incremental password strengthening where users are continually nudged to memorize one more character of a strong 56-bit password. Another line of work has focused on developing strategies for users to generate stronger passwords e.g., see [43]–[46]. To determine whether a particular intervention (e.g., password generation strategy/composition policy/strength meter/ etc..) strengthened the password distribution we need to characterize the attacker's guessing curve before/after the intervention.

Empirical Distribution: The empirical password distribution has been used to evaluate many password research ideas. Harsha et al. [47] used empirical distributions derived from the LinkedIn and RockYou datasets to quantify the advantage of an attacker after learning the length of the user's password. The empirical password distribution has also been used to tune and evaluate distribution-aware password throttling mechanisms [48], [49] to defend against online attacks, distribution-aware mechanisms to tune relevant cost parameters for password hashing [17], [18], [50], achieve (personalized) password typo correction [12], [13] and evaluate the security of Compromised Credential Checking protocols [33].

Estimating Properties of (Password) Distributions: Valiant and Valiant [20] proposes an approach to accurately estimate key properties of *any* distribution over at most k distinct elements using $N = O(k/\log k)$ independent and identically distributed (i.i.d.) samples. However, we cannot directly apply the results of Valiant and Valiant [20] to bound λ_G in our password setting as the number of distinct passwords k in the support of the password distribution $\mathcal{P}$ is unknown to us and we almost certainly have $k \gg N^2$ even for our largest password datasets. However, we are able to adapt the techniques of Valiant and Valiant [20] when developing the linear program that we use to upper/lower bound λ_G. Bonneau [16] collected a password frequency corpus derived from approximately 7×10^7 Yahoo! passwords and used this corpus to estimate properties of the Yahoo! password distribution. Subsampling was one of the key heuristics used to identify stable statistical estimates e.g., Bonneau found that the value $\hat{\lambda}_{10}$ was stable under subsampling [16] heuristically concluding that $\lambda_{10} \approx \hat{\lambda}_{10}$. By contrast, we provide rigorous statistical techniques to upper/lower bound λ_G directly and precisely bound the probability of an error δ. We also stress that the value $\hat{\lambda}_G$ will not be stable with respect to subsampling for many larger values of $G > 10$ while our techniques can still be applied to obtain rigorous upper/lower bounds on λ_G.

II. Attack Model and Notation

A. Notation

We let $\mathcal{P}$ denote an arbitrary password distribution over passwords $pwd_1, pwd_2, \ldots$ and we let p_i denote the probability of sampling pwd_i. We use $s \leftarrow \mathcal{P}$ to denote a random sample from the distribution and $S = \{s_1, ..., s_N\} \leftarrow \mathcal{P}^N$ to denote a multiset of N independent and identically distributed samples from $\mathcal{P}$. We will occasionally abuse notation and also use

$\mathcal{P}$ to denote the set of passwords $\{pwd_1, pwd_2, \ldots\}$ in the support of the distribution. We assume that the passwords are ordered in descending order of probability such that $p_1 \geq p_2 \geq p_3 \ldots$ and in general $p_i \geq p_{i+1}$. Note that passwords are sampled with replacement so we could have $s_i = s_j$ for $i < j$. It will be convenient to define $f_i^S \doteq |\{j : s_j = pwd_i\}|$ as the number of times pwd_i appears in the sample S — the superscript S may be omitted when the sample set S is clear from context (Note that if $i < j$ we could still have $f_i^S < f_j^S$ even though $p_i > p_j$ since the passwords pwd_i are ordered by probability *not* by their frequency in the sample S.) We will also define $F_i^S \doteq |\{pwd_j : f_j^S = i\}|$ as the number of distinct passwords that appear exactly i times in S and denote $F^S = \left(F_1^S, F_2^S, ..., F_N^S\right)$ as the frequency encoding of our sample S. Under this notation we have $\mathbf{Unique}(S) = F_1^S$, $\mathbf{Distinct}(S) = \sum_{i \leq N} F_i^S$ and $N = \sum_{i \leq N} i \times F_i^S$.

It will be convenient to let L_S denote the list of all distinct passwords in S ordered by frequency f_i^S — ties can be broken in arbitrary order e.g., lexicographic. We also define the set $T(S, G)$ which contains the first G passwords in L_S. If $G \geq \mathbf{Distinct}(S)$ then $T(S, G)$ is simply the set of all distinct passwords in the sample S. Finally, we will use $\texttt{bpdf}(i, N, p) \doteq \binom{N}{i} p^i (1-p)^{N-i}$ to denote the binomial probability density function i.e., if password pwd has probability p and we draw N samples from our password distribution $\mathcal{P}$ then $\texttt{bpdf}(i, N, p)$ denotes the probability that pwd is sampled exactly i times.

B. Attacker Model

We consider an attacker who knows the password distribution $\mathcal{P}$ but does not have additional information about the sampled passwords $S \leftarrow \mathcal{P}^N$. In particular, for each i the attacker knows pwd_i and p_i. For each sampled user password $s_i \in S$ the attacker is given G guesses to crack the password s_i. For an online attacker G will typically be small as an authentication server can lock the account after several consecutive incorrect login attempts. By contrast, G will be much larger for an offline attacker who has stolen the (salted) cryptographic hash of the user's password can check as many passwords as s/he wants by comparing the (salted) cryptographic hash $h^i = H(u_i, s_i)$ with the hash $h_j^i = H(u_i, pwd_j)$ for each $j \leq G$. An offline attacker is limited only by the resources s/he is willing to invest cracking and by the cost of repeatedly evaluating the password hash function.

Whether G is large or small the optimal strategy for the attacker is always to check the G most probable passwords $pwd_1, pwd_2, \ldots, pwd_G$ in the distribution. We use the random variable $\lambda(S, G) \doteq \sum_{i \leq G} f_i^S / N$ to denote the percentage of passwords in S cracked within G guesses. Observe that the expected value of $\lambda(S, G)$ is $\mathbb{E}(\lambda(S, G)) = \sum_{i \leq G} p_i = \lambda_G$. In the next section we will show that the random variable $\lambda(S, G)$ is tightly concentrated around its mean λ_G i.e., except with negligible probability we will have $|\lambda_G - \lambda(S, G)| \leq \epsilon$. In this sense upper/lower bounding λ_G and $\lambda(S, G)$ can be seen as (nearly) equivalent problems.

III. Theoretical Upper/Lower Bounds

In this section we introduce several algorithms to generate high-confidence upper bounds and lower bounds on λ_G given N independent and identically distributed (iid) samples $S = \{s_1, \ldots s_n\}$ from our password distribution $\mathcal{P}$. An upper bound $\mathsf{UB}(S, G)$ (resp. an lower bound $\mathsf{LB}(S, G)$) derived from the sample S holds with confidence $1-\delta$ if $\Pr[\lambda_G \geq \mathsf{UB}(S, G)] \leq \delta$ (resp. $\Pr[\lambda_G \leq \mathsf{LB}(S, G)] \leq \delta$) where the randomness is taken over the selection of $S \leftarrow \mathcal{P}^N$. Before presenting our results, we first introduce the well-known bounded differences inequality [51] (also called McDiarmid's inequality), which will be useful in our proofs.

Theorem 1. *(Bounded Differences Inequality [51]) Suppose that $(X_1, ..., X_n) \in \Omega$ are independent random variables. Let $f : \Omega \to \mathbb{R}$ satisfy the bounded differences property with constants $c_1, ..., c_n$, i.e., for all $i \in \{1, ..., n\}$ and all $x, x' \in \Omega$ that differ only at the i-th coordinate, the output of the function $|f(x) - f(x')| \leq c_i$. Then,*

$$\Pr[f(X_1, ..., X_n) - \mathbb{E}(f(X_1, ..., X_n)) \geq t] \leq \delta,$$
$$\Pr[f(X_1, ..., X_n) - \mathbb{E}(f(X_1, ..., X_n)) \leq -t] \leq \delta,$$

where $\delta = \exp\left(\frac{-2t^2}{\sum_{i=1}^n c_i^2}\right)$.

As an immediate application of McDiarmid's inequality we can prove that, except with negligible probability, we have $|\lambda(S, G) - \lambda_G| \leq \epsilon$ i.e., $\lambda(S, G)$ is tightly concentrated around its mean $\mathbb{E}[\lambda(S, G)] = \lambda_G$ when the sample size N is large enough — see Theorem 2. Thus, one strategy to derive a high confidence upper/lower bound for λ_G is to derive a high confidence upper/lower bound for $\lambda(S, G)$ and we will immediately obtain a high confidence upper/lower bound for λ_G as a corollary. The proof of Theorem 2 is in Appendix E-B. Intuitively, we have $|\lambda(S, G) - \lambda(S', G)| \leq 1/N$ whenever $S = \{s_1, ..., s_i, ..., s_N\}$ and $S' = \{s_1, ..., s_i', ..., s_N\}$ is obtained by swapping out s_i for s_i'. Thus, we can apply McDiarmid's inequality.

Theorem 2. *For any guessing number $G \geq 0$ and any $0 \leq \epsilon \leq 1$ we have:*

$$\Pr[\lambda(S, G) \leq \lambda_G + \epsilon] \geq 1 - \exp\left(-2N\epsilon^2\right) \text{ , and}$$
$$\Pr[\lambda(S, G) \geq \lambda_G - \epsilon] \geq 1 - \exp\left(-2N\epsilon^2\right)$$

where the randomness is taken over the sample set $S \leftarrow \mathcal{P}^N$ of size N.

A. Empirical Distribution as an Upper Bound

In this section we show that we can upper bound λ_G using the empirical distribution $\hat{\lambda}_G$. In particular, we argue that for any sample S we have $\hat{\lambda}_G \geq \lambda(S, G)$. As an immediate corollary we get that $\Pr[\lambda_G > \hat{\lambda}_G + \epsilon] \leq \delta$ where $\delta = \exp\left(-2N\epsilon^2\right)$.

Theorem 3. *For any sample set $S \leftarrow \mathcal{P}^N$ with size N and any $G > 0$ we have $\lambda(S, G) \leq \frac{1}{N} \sum_{i: pwd_i \in T(S,G)} f_i^S$.*

Proof. We observe that $\sum_{i:pwd_i \in T(S,G)} f_i^S = \max_{1 \le j_1 < \cdots < j_G \le |P|} \sum_{i \in \{j_1,\ldots,j_G\}} f_i^S \ge \sum_{i \le G} f_i^S = N\lambda(S,G)$. It follows that $\lambda(S,G) \le \frac{1}{N}\sum_{i \in T(S,G)} f_i^S$ as claimed. □

Recall that $T(S,G)$ is the set of G most frequent passwords in the sample S. Applying Theorem 2 to Theorem 3, we can immediately obtain the upper bound of λ_G as below:

Corollary 4. *For any guessing number $G \ge 0$ and $\epsilon > 0$ we have*

$$\Pr\left[\lambda_G \le \frac{1}{N}\sum\nolimits_{i \in T(S,G)} f_i + \epsilon\right] \ge 1 - \exp\left(-2N\epsilon^2\right),$$

where the randomness is taken over the selection of $S \leftarrow \mathcal{P}^N$.

B. A Lower Bound for $G < N$

In this section, we introduce a new idea to lower bound λ_G. The key idea is to randomly partition our sample S into two datasets $D_1 = \{s_1, ..., s_{N-d}\}$ and $D_2 = \{s_{N-d+1}, ..., s_N\}$ with size $N-d$ and d. We can then construct a dictionary $T(D_1,G)$ containing the top G passwords in D_1 which is used to attack passwords in D_2. In particular, we let $h(D_1,D_2,G) = |\{N-d+1 \le i \le N : s_i \in T(D_1,G)\}|$ denote the number of samples in D_2 that are also in $T(D_1,G)$. Because we can view D_2 as d independent samples from the password distribution $\mathcal{P}$ we have $\mathbb{E}\left[h(D_1,D_2,G)/d\right] \le \lambda_G$ since λ_G denotes the expected fraction of passwords in D_2 that are cracked using the optimal dictionary instead of $T(D_1,G)$. Adding a slack term t/d we can use the Bounded Differences Inequality to argue that $\Pr[\lambda_G \le \frac{1}{d}(h(D_1,D_2,G) - t)] \le \delta$ where $\delta = \exp(-2t^2/d)$.

We remark that $h(D_1,D_2,G)/d$ can also be viewed as the success rate of an attacker who has partial knowledge of the password distribution. In particular, suppose that the attacker has $N-d$ samples D_1 from $\mathcal{P}$ e.g., obtained by cracking the corresponding password hashes or by some other means such as phishing. Then the attacker can construct the dictionary $T(D_1,G)$ and use this dictionary to crack the remaining passwords in D_2. We can view $h(D_1,D_2,G)$ as the number passwords in D_2 that that the attacker would crack within G guesses.

Theorem 5. *For any guessing number $G \ge 1$ and any parameters $0 < d < N$ and $t \ge 0$, we have*

$$\Pr[\lambda_G \ge \frac{1}{d}(h(D_1,D_2,G) - t)] \ge 1 - \exp\left(-2t^2/d\right)$$

where the randomness is taken over the samples $D_1 \leftarrow \mathcal{P}^{N-d}$ and $D_2 \leftarrow \mathcal{P}^d$.

When applying Theorem 5 we can select $d \ll N$ and set $t = \sqrt{(d/2)\ln(1/\delta)}$ to get ensure that $t/d = o(1)$ is small and our probability of error is at most δ. The proof of Theorem 5 is deferred to the full version [52] of the paper. The full version [52] also contains a corollary lower bounding $\lambda(S,G)$ using Theorems 5 and 2.

The lower bound in Theorem 5 is often tight for smaller values of $G \ll N$, but it will plateau at $G = \mathbf{Distinct}(S)$ since the set $T(D_1,G)$ already contains all of the passwords in D_1. When $d \ll N$ and $G = \mathbf{Distinct}(S)$ the lower bound will closely match the Good-Turing estimate $1 - \frac{\mathbf{Distinct}(S)}{N}$ for the total probability mass of all passwords in the set S.

C. An Extended Lower Bound Using Password Models

The lower bound from Section III-B plateau's when $G \ge \mathbf{Distinct}(S)$. Is it possible to derive tighter lower bounds for larger values of G? In this section we show that any password cracking model can be used to derive high confidence lower bounds on λ_G and then show how our prior lower bound can be combined with a password cracking model M to derive tighter bounds.

Let $M(D_1,G)$ be the set of top G password guesses output by an attack model M trained on D_1. We now follow the same approach as before and partition S into two sets $D_1 = \{s_1, ..., s_{N-d}\}$, $D_2 = \{s_{N-d+1}, ..., s_N\}$. Let $h'_M(D_1,D_2,G)$ be the number of passwords cracked in D_2 by making guesses in $M(D_1,G)$. We can prove a generalized lower bound of λ_G — see Theorem 6 below.

Theorem 6. *For any guessing number $G > 0$ and any parameters $0 < d < N$, $t \ge 0$ we have:*

$$\Pr[\lambda_G \ge \frac{1}{d}(h'_M(D_1,D_2,G) - t)] \ge 1 - \exp\left(-2t^2/d\right)$$

where the randomness is taken over the sample S of size N.

Proof. Since $p_1, p_2, ..., p_i, ...$ are sorted in decreasing order, we have $\mathbb{E}(h'_M(D_1,D_2,G)) = d \times \sum_{i:pwd_i \in M(D_1,G)} p_i \le d \times \sum_{i \le G} p_i = d \times \lambda_G$. Using Theorem 1, we have:

$$\Pr[h'_M(D_2,G) \le d \sum_{i:pwd_i \in M(D_1,G)} p_i + t] \ge 1 - \exp\left(-2t^2/d\right)$$

$$\Rightarrow \Pr[\lambda_G \ge \frac{1}{d}(h'_M(D_2,G) - t)] \ge 1 - \exp\left(-2t^2/d\right)$$

where the last line follows since $\lambda_G \ge \sum_{i:pwd_i \in M(D_1,G)} p_i$. □

As an immediate corollary of Theorem 6 we can also lower bound $\lambda(S,G)$ — see the full version [52]. As a more useful corollary given any model M we can define a hybrid attack model $M^*(D_1,G)$ which, given a dataset D_1, first constructs a dictionary $T(D_1,G)$ containing the top G passwords in D_1. When trying to crack a new password $pwd \leftarrow \mathcal{P}$ the model $M^*(D_1,G)$ begins by checking each of the passwords in this dictionary $T(D_1,G)$. If $G > \mathbf{Distinct}(D_1)$ and pwd does not appear in the dictionary $T(D_1,G)$ then the model proceeds to generate the remaining $G' = G - \mathbf{Distinct}(D_1)$ guesses using the model M — we can optionally omit guesses which already appear in our training dataset D_1. Note that for $G \le \mathbf{Distinct}(D_1)$ we have $h'_{M^*}(D_1,D_2,G) = h(D_1,D_2,G)$ where $h(D_1,D_2,G)$ counts the number of samples in D_2 that appear in the top G samples from D_1. For $G \ge \mathbf{Distinct}(D_1)$ the function $h'_{M^*}(D_1,D_2,G)$ counts the number of samples in D_2 that either (1) appear in D_1 or (2) appear in $M(D_1,G')$. Thus, at minimum we always have $h'_{M^*}(D_1,D_2,G) \ge \max\{h(D_1,D_2,G), h'_M(D_1,D_2,G')\}$ where $G' = G - \mathbf{Distinct}(D_1)$. Intuitively, the lower bound

will be at least as good as our prior approach from Theorem 5 and at least as good as the model M.

Corollary 7. *Let M be a password cracking model and let parameters $G, d > 0$, $t > 0$ be given then*

$$\Pr[\lambda_G \geq \frac{1}{d}(h'_{M^*}(D_1, D_2, G) - t)] \geq 1 - \delta$$

where $\delta = \exp\left(-2t^2/d\right)$ and the randomness is taken over the set S of size N.

We can also view the lower bound from Corollary 7 as denoting the success rate of a hybrid attacker i.e., an attacker who has obtained a cracked/leaked passwords D_1 and runs the hybrid attack described above will crack $h'_{M^*}(D_1, D_2, G)$ of the remaining passwords in D_2 within G guesses.

D. Upper And Lower Bounds Using Linear Programming

In this section we propose a different approach to generate upper and lower bounds using linear programming which is inspired by work of Valiant and Valiant [20]. As we will see in our empirical analysis these LP bounds tend to be tighter when the guessing number is large though the bounds are slightly worse when the guessing number G is smaller i.e., $G \ll \mathbf{Distinct}(S)$.

Intuitively, we derive our upper (resp. lower) bounds by designing a linear program to find a distribution $\mathcal{P}'$ that maximizes (resp. minimizes) λ'_G subject to various consistency constraints derived from our sample $S \leftarrow \mathcal{P}^n$. To simplify our exposition it is helpful to begin with a simplifying assumption that the probability distribution $\mathcal{P}$ can be represented as a histogram $h_1, \ldots, h_\ell$ over a finite probability mesh $X = \{x_1, \ldots, x_\ell\}$ i.e., there are exactly h_i passwords in the support of the distribution which have probability exactly x_i. We will later show how this simplifying assumption can be removed.

The values h_i are our unknown variables in our linear program. We first note that any valid probability histogram must satisfy the constraints that $h_1 x_1 + \ldots + h_\ell x_\ell = 1$ and $0 \leq h_i$. More significantly, we adapt ideas from Good-Turing Frequency estimation to argue that the known value $P_i^S \doteq \frac{(i+1)F_{i+1}^S}{N-i}$ will be close to the (unknown value) $\sum_{j=1}^{\ell} h_j x_j \cdot \texttt{bpdf}(i, N, x_j)$ with high probability i.e., we can compute P_i^S and add linear constraints on the variables $h_1, \ldots, h_\ell$ to ensure that $P_i^S \approx \sum_{j=1}^{\ell} h_j x_j \cdot \texttt{bpdf}(i, N, x_j)$. We include this constraint for each value of $i \leq i'$ where i' is a parameter that will be selected later. We remark that the original password distribution $\mathcal{P}$ will be consistent with each of these constraints with high probability. Thus, with high probability, minimizing (resp. maximizing) λ'_G subject to the relevant constraints yields a lower (resp. upper) bound on λ_G. We can remove our simplifying assumption that $\mathcal{P}$ is consistent with a finite probability mesh by (1) adding a new variable p which intuitively represents the probability mass of all passwords in the distribution with probability $\leq x_\ell$, (2) ensuring that the probability mesh is sufficiently fine-grained that every probability value $1 \geq p \geq x_\ell$ is close to some point on the mesh, and (3) adding slack terms to each constraint to ensure that the probability histogram for $\mathcal{P}$ is still consistent with all of the constraints even after rounding probability values to fit the mesh.

We remark that if the dataset S was not sampled independently from some (unknown) distribution $\mathcal{P}$ then it is possible that there will be *no* feasible solution to our linear program. Thus, as a side-benefit our linear program can allow us to identify datasets which are inconsistent with our iid assumption. As a concrete example, suppose that the dataset S was sampled by picking $s_1, \ldots, s_{N/2} \leftarrow \mathcal{P}$ independently and then duplicating the last $N/2$ passwords i.e., $s_{N/2+i} = s_i$ for $i \leq N/2$. The resulting dataset would (likely) be correctly rejected by our linear program since it is not iid.

1) A Linear Programming Task with Idealized Settings: We first describe our linear program in the idealized setting where we assume that our (unknown) password distribution $\mathcal{P}$ is consistent with a finite probability mesh. In particular, we will fix a finite probability mesh $X_\ell = \{x_1, \ldots, x_\ell\}$ with $x_1 > x_2 > \cdots x_\ell$ and assume that for all passwords pwd_i we have $p_i \in X$ i.e., the mesh contains every probability value in our distribution. This allows us to view the original distribution $\mathcal{P}$ as a histogram $H = h_1, \ldots, h_\ell$ where h_i denotes the number of items in the support of $\mathcal{P}$ which occur with probability x_i. We use $h_1, \ldots, h_\ell \geq 0$ as variables in our linear program (relaxing the natural constraint that h_i is an integer). Now the linear constraint $\sum_j h_j x_j = 1$ encodes the requirement that our probabilities sum to 1.

Given our sample S of size N recall that F_i^S denotes the number of distinct passwords that appear exactly i times in our sample S. Thus, the expected value of F_i^S is $\sum_{j=1}^{\ell} h_j \texttt{bpdf}(i, N, x_j)$ and $\sum_{j=1}^{l} h_j \times x_j \times \texttt{bpdf}(i, N, x_j)$ is the expected probability mass of all items that were sampled exactly i times. Adapting ideas from Good-Turing Frequency estimation we can argue that (whp) $\sum_{j=1}^{l} h_j \times x_j \times \texttt{bpdf}(i, N, x_j)$ will be close to $P_i^S = \frac{(i+1)F_{i+1}^S}{N-i}$.

In particular, Lemma 1 shows that for each frequency i we will have $P_i^S - \epsilon_{2,i} - \frac{i+1}{N-i} \leq \sum_{j=1}^{\ell} h_j \times x_j \times \texttt{bpdf}(i, N, x_j) \leq P_i^S + \epsilon_{2,i}$ with high probability. Thus, we will include this linear constraint for each $i \leq i'$ where i' is a parameter that can be tuned. Intuitively, increasing i' adds additional constraints to our linear program which reduces the feasible region and can only improve the upper/lower bound. However, increasing i' can also decrease our confidence δ since we need to ensure that $\mathcal{P}$ is consistent with *all* of the constraints that we generate to argue that the upper/lower bounds are valid.

Intuitively, if we search for a distribution that maximizes (resp. minimizes) λ_G we obtain an upper bound (resp. lower bound) since the real distribution $\mathcal{P}$ will be one of the feasible solutions (whp). The remaining challenge is to encode λ_G as a linear objective function using the variables $h_1, \ldots, h_\ell$. If we happened to know the integers c, idx such that $G = c + \sum_{j<idx} h_j$ and $0 \leq c \leq h_{idx}$ then we easily could encode $\lambda_G = c \cdot x_{idx} + \sum_{j<idx} h_i \cdot x_i$ as a linear objective function. However, we cannot compute c or idx a priori since the values $h_1, \ldots, h_\ell$ are unknown. We deal with this challenge

by introducing a separate linear program for each possible value of idx and by adding a new variable c along with the constraints that $0 \leq c \leq h_{idx}$ and $c = G - \sum_{j<idx} h_j$. Letting y^*_{idx} denote the value of optimal solution to our LP with the parameter idx we can combine these solutions to get our final upper/lower bound i.e., $y^* = \max\{y^*_{idx} : idx \leq \ell\}$ (resp. $y^* = \min\{y^*_{idx} : idx \leq \ell\}$) when computing our upper bound (resp. lower bound) where each y^*_{idx} is the value we obtain when maximizing (resp. minimizing) the corresponding LP.

Our linear program $\texttt{LP1}(G, b, X_\ell, F^S, idx, i', \boldsymbol{\epsilon_2})$ is shown below (Linear Programming Task 1). The inputs include the guessing budget G, the probability mesh $X_\ell = \{x_1, \ldots, x_\ell\}$, the set $F^S = \{F^S_1, \ldots, F^S_N\}$, a bit $b \in \{-1, 1\}$ which indicates whether we are looking for an upper or lower bound, the (guessed) value idx and parameters i' and $\boldsymbol{\epsilon_2} = \{\epsilon_{2,0}, \ldots, \epsilon_{2,i'}\}$ related to the consistency constraints.

Linear Programming Task 1:
$\texttt{LP1}(G, b, X_\ell, F^S, idx, i', \boldsymbol{\epsilon_2})$
Input Parameters: G, b, $X_\ell = \{x_1, \ldots, x_\ell\}$, $F^S = \{F^S_1, \ldots, F^S_N\}$, idx, i', $\boldsymbol{\epsilon_2} = \{\epsilon_{2,0}, \ldots, \epsilon_{2,i'}\}$
Variables: $h_1, \ldots, h_\ell, c$
Objective: $\min\left(b \times (\sum_{j<idx} h_j \times x_j + c \times x_{idx})\right)$
Constraints:
1) $\sum_{j<idx} h_j + c = G$
2) $\forall 0 \leq i \leq i'$, $\frac{(i+1)F^S_{i+1}}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} \leq \sum_{j=1}^{\ell} h_j \times x_j \times \texttt{bpdf}(i, N, x_j) \leq \frac{(i+1)F^S_{i+1}}{N-i} + \epsilon_{2,i}$
3) $\sum_{j=1}^{\ell} h_j \times x_j = 1$
4) $0 \leq c \leq h_{idx}$

The following lemma indicates that constraint (2) holds with high probability over the selection of $S \leftarrow \mathcal{P}^N$ when we select the parameters i' and $\boldsymbol{\epsilon_2} = \{\epsilon_{2,0}, \ldots, \epsilon_{2,i'}\}$ properly.

Lemma 1. *For any $i \geq 0$ and $0 \leq \epsilon_{2,i} \leq 1$, we have $\frac{(i+1)F^S_{i+1}}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} \leq \sum_j h_j \times x_j \times \texttt{bpdf}(i, N, x_j) \leq \frac{(i+1)F^S_{i+1}}{N-i} + \epsilon_{2,i}$ with probability at least $1 - 2 \times \exp\left(\frac{-2(N-i)^2\epsilon^2_{2,i}}{N(i+1)^2}\right)$ where the probability is taken over the selection of our sample $S \leftarrow \mathcal{P}^N$.*

The proof of Lemma 1 is in Appendix E-A. Using Lemma 1 and conclusions we present above, we can show that the upper/lower bounds of λ_G derived from our linear program hold with high-confidence. In particular, for any constant $\delta > 0$ the parameters $\epsilon_{2,i}$ can be tuned such that, except with probability δ, we have the lower bound (resp. upper bound) $\lambda_G \geq \min_{1 \leq idx \leq \ell} \texttt{LP1}(X_\ell, F^S, idx, G, 1, i', \boldsymbol{\epsilon_2})$ (resp. $\lambda_G \leq \max_{1 \leq idx \leq \ell} \texttt{LP1}(X_\ell, F^S, idx, G, -1, i', \boldsymbol{\epsilon_2})$. See Theorem 10 in Appendix E-A for a formal statement.

2) Intermediate Step: Linear Programming with Countably Infinite Probability Mesh: In this section we show how to relax the assumption that the real password distribution $\mathcal{P}$ is consistent with a finite probability mesh and replace it with a slightly weaker assumption that $\mathcal{P}$ is consistent with a *countably infinite* mesh $X = \{x_1, x_2, ..., x_\ell, x_{\ell+1}, ...\}$ with $x_1 > x_2 > x_3 > \cdots$ and $\lim_{i \to \infty} x_i = 0$. As before we assume that for all passwords pwd_i in the support of $\mathcal{P}$ the probability $p_i \in X$ of sampling this password lies in the mesh X. Suppose that $H = (h_1, h_2, \ldots)$ is a histogram encoding of $\mathcal{P}$ and let $p = \sum_{i \geq \ell+1} x_i h_i$ be the total probability mass of all passwords in $\mathcal{P}$ with probability smaller than x_ℓ. Our key idea to ensure that our linear programs remain finite is to introduce a new variable for p and eliminate the variables h_j for each $j > \ell$. For example, we now have the constraints that $p + x_1 h_1 + \ldots + x_\ell h_\ell = 1$ and that $0 \leq p \leq 1$.

Lemma 1 still applies so (whp) we have $(i+1)F^S_{i+1}/(N-i) - \epsilon_{2,i} - \frac{i+1}{N-i} \leq \sum_{j=1}^{\infty} x_j h_j \texttt{bpdf}(i, N, x_j) \leq (i+1)F^S_{i+1}/(N-i) + \epsilon_{2,i}$ i.e., the expected probability mass of all items that appear i times in our sample $S \leftarrow \mathcal{P}^N$ ($i \geq 0$) is still $\sum_{j=1}^{\infty} x_j h_j \texttt{bpdf}(i, N, x_j)$. However, we cannot add these constraints to our linear program since we do not have variables for h_j when $j > \ell$. Instead, we rely on the following observations (1) The function $f(x) = \texttt{bpdf}(0, N, x)$ is monotonically decreasing over the domain $[0, 1]$ so we have $\texttt{bpdf}(0, N, x_\ell) < \texttt{bpdf}(0, N, x_j)$ whenever $j > \ell$. Thus, we can bound the partial sum $\sum_{j=\ell+1}^{\infty} x_j h_j \texttt{bpdf}(0, N, x_j)$ as follows $p \geq \sum_{j=\ell+1}^{\infty} x_j h_j \texttt{bpdf}(0, N, x_j) \geq p \cdot \texttt{bpdf}(0, N, x_\ell)$. (2) For $i > 0$ the function $f_i(x) = \texttt{bpdf}(i, N, x)$ is monotonically increasing over the domain $[0, 1/N]$. Thus, for $i > 0$ we have $\texttt{bpdf}(i, N, x_\ell) > \texttt{bpdf}(i, N, x_j)$ whenever $x_j < x_\ell < \frac{1}{N}$. In this case we can bound the partial sum $\sum_{j=\ell+1}^{\infty} x_j h_j \texttt{bpdf}(i, N, x_j)$ as follows $p \cdot \texttt{bpdf}(i, N, x_\ell) \geq \sum_{j=\ell+1}^{\infty} x_j h_j \texttt{bpdf}(0, N, x_j) \geq 0$.

We can use the above observations to update Constraint (2) in LP1. Similarly, we can replace Constraint (3) in LP1 with $\sum_{j=1}^{\ell} h_j x_j = 1 - p$. We call this updated linear program LP1a. Since this is only an intermediate step, we defer the full description of LP1a and associated theorems to the full version [52]. We briefly remark that for $i > 0$ (resp. $i = 0$) we have $\lim_{x \to 0} \texttt{bpdf}(i, N, x) = 0$ (resp. $\lim_{x \to 0} \texttt{bpdf}(0, N, x) = 1$) so the upper/lower bounds on our partial sums are essentially tight and we are not substantially loosening our constraints in LP1a.

3) Final LP: Eliminating Ideal Mesh Assumptions: In this section, we present our final linear program to bound λ_G without making any idealized assumptions about the distribution $\mathcal{P}$. Following [20], we fix a particular mesh $X_{\ell,q} = \{x_1, \ldots, x_\ell\}$ where we have $x_i = q \cdot x_{i+1} = q^{\ell-i} x_\ell$ for each $1 \leq i < \ell$. Note that the parameter $q > 1$ determines how fine-grained our mesh values are. We will pick x_ℓ to be suitably small (e.g., $x_\ell = 10^{-4} N^{-1}$) and set $\ell = \left\lfloor \frac{\ln(\frac{1}{x_\ell})}{\ln q} \right\rfloor + 1$ so that $x_1 \approx 1$.

For the purposes of analysis consider a histogram encoding of the real distribution $\mathcal{P}$ i.e., $H^r = \{h^r_1, h^r_2 \ldots\}$ where the mesh $X^r = \{x^r_1, x^r_2, \ldots\} = \{\Pr[pwd] : pwd \in \mathcal{P}\}$ is defined using the exact probabilities in the distribution. Of course the mesh X^r and the associated histogram H^r are unknown so we cannot simply evaluate the formula above. However, it is helpful to imagine rounding each point x^r_i to a value $\texttt{Round}(x^r_i) \in X_{\ell,q}$ and defining the rounded

histogram $h_i = \sum_{j:\texttt{Round}(x_j^r)=x_i} h_j^r$ accordingly. Supposing that $G = c + \sum_{i=1}^{idx'} h_i^r$ for some idx' and $c \le h_{idx'}^r$ we also have $G = c + \sum_{i=1}^{idx} h_i$ for some idx and $c \le h_i$. Assuming that we always round down, i.e., $\texttt{Round}(x_i^r) \le x_i^r$, then we have $\lambda_G = c' \cdot x_{idx'}^r + \sum_{j=1}^{idx'-1} x_i^r h_i^r \ge c \cdot \texttt{Round}(x_{idx'}^r) + \sum_{j=1}^{idx'-1} \texttt{Round}(x_i^r) h_i^r \ge c \cdot x_{idx} + \sum_{j=1}^{idx-1} h_i x_i$.

Intuitively, to obtain our lower bound we relax all of our constraints from LP1a to ensure that our rounded histogram is still consistent even still hold (whp) even if $\mathcal{P}$ does not precisely fit the mesh $X_{\ell,q}$. For example, we can replace the exact constraint that $\sum_{j=1}^{\ell} h_j^r \times \texttt{Round}(x_i^r) = 1 - p$ with an approximate version $\frac{1-q}{q} \le \sum_{j=1}^{\ell} h_j^r \times \texttt{Round}(x_i^r) \le 1 - p$ which the rounded histogram will still satisfy. The approach to define our linear program for the upper bound is similar with the key difference that we round up instead of down i.e., $\texttt{Round}(x_i^r) \ge x_i^r$.

The linear program LPlower to lower bound λ_G is shown below. We add slack terms with parameters $q, \epsilon_{\mathbf{3}}, \hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}}$ to ensure that the rounded histogram is consistent with all of the constraints. The linear program LPupper to upper bound λ_G is similar to LPlower. We show it in Appendix A.

Linear Programming Task 2:
$\texttt{LPlower}(G, X_\ell, F^S, idx, i', \epsilon_{\mathbf{2}}, \epsilon_{\mathbf{3}}, \hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}})$
Input Parameters: G, $X_\ell = \{x_1, ..., x_\ell\}$, $F^S = \{F_1^S, ..., F_N^S\}$, idx, i', $\epsilon_{\mathbf{2}} = \{\epsilon_{2,0}, \ldots, \epsilon_{2,i'}\}$, $\hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}} = \{\hat{x}_{\epsilon_{3,0}}, \ldots, \hat{x}_{\epsilon_{3,i'}}\}$, $\epsilon_{\mathbf{3}} = \{\epsilon_{3,i} = \frac{1}{q^{i+1}} \left(\frac{1-\hat{x}_{\epsilon_{3,i}}}{1-q\hat{x}_{\epsilon_{3,i}}}\right)^{N-i} - 1, 0 \le i \le i'\}$
Variables: $h_1, ..., h_\ell, c, p$
Objective: $\min \left(\sum_{j<idx} h_j \times x_j + c \times x_{idx}\right)$
Constraints:

1) $\sum_{j<idx} h_j + c = G$
2) $\forall 0 \le i \le i'$:
   a) for $i = 0$, $\frac{1}{q^{i+1}}(\frac{(i+1)F_{i+1}^S}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} - p) \le \sum_{j=1}^{\ell} h_j \times x_j \times \texttt{bpdf}(i, N, x_j) \le (1 + \epsilon_{3,i})(\frac{(i+1)F_{i+1}^S}{N-i} + \epsilon_{2,i} - p \times \texttt{bpdf}(i, N, qx_\ell)) + \texttt{bpdf}(i, N, \hat{x}_{\epsilon_{3,i}})$
   b) for $1 \le i \le i'$, $\frac{1}{q^{i+1}}(\frac{(i+1)F_{i+1}^S}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} - p \times \texttt{bpdf}(i, N, qx_\ell)) \le \sum_{j=1}^{\ell} h_j \times x_j \times \texttt{bpdf}(i, N, x_j) \le (1 + \epsilon_{3,i})(\frac{(i+1)F_{i+1}^S}{N-i} + \epsilon_{2,i}) + \texttt{bpdf}(i, N, \hat{x}_{\epsilon_{3,i}})$
3) $\frac{1-p}{q} \le \sum_{j=1}^{\ell} h_j \times x_j \le 1 - p$
4) $0 \le c \le h_{idx}$

(**Note:** we consider $idx = 1, 2, ..., \ell + 1$. When $idx = \ell + 1$, we define $h_{\ell+1} = G$ and $x_{\ell+1} = 0$.)

Theorem 8 shows that the upper/lower bounds we obtain hold with high confidence — due to space limitations the formal proof is deferred to the full version [52].

Theorem 8. *Given an unknown password distribution $\mathcal{P}$ for any $G \ge 0$, integer $i' \ge 0$, $\epsilon_{\mathbf{2}} = \{\epsilon_{2,0}, \ldots, \epsilon_{2,i'}\} \in [0,1]^{i'+1}$, $\hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}} = \{\hat{x}_{\epsilon_{3,0}}, \ldots, \hat{x}_{\epsilon_{3,i'}}\}$, $\epsilon_{\mathbf{3}} = \{\epsilon_{3,i} = \frac{1}{q^{i+1}}(\frac{1-\hat{x}_{\epsilon_{3,i}}}{1-q\hat{x}_{\epsilon_{3,i}}})^{N-i} - 1, 0 \le i \le i'\}$, we have:*

$$\Pr\left[\lambda_G \ge \min_{1 \le idx \le l+1} \texttt{LPlower}(X_{l,q}, F^S, idx, G, i', \epsilon_{\mathbf{2}}, \epsilon_{\mathbf{3}}, \hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}})\right] \ge 1 - \delta$$

$$\Pr\left[\lambda_G \le \max_{1 \le idx \le l+1} \texttt{LPupper}(X_{l,q}, F^S, idx, G, i', \epsilon_{\mathbf{2}}, \epsilon_{\mathbf{3}}, \hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}})\right] \ge 1 - \delta$$

where $\delta = 2 \times \sum_{0 \le i \le i'} \exp\left(\frac{-2(N-i)^2\epsilon_{2,i}^2}{N(i+1)^2}\right)$ and the randomness is taken over the sample $S \leftarrow \mathcal{P}^N$ of size N.

As a corollary of Theorem 8 we can also lower bound $\lambda(S, G)$ — see the full version [52].

E. A Lower Bound for $G \ge N$ Derived from Existing Work

In this section we prove a lower bound using an idea from prior work. We use this bound as a comparison in Section IV showing that our bounds perform very well.

Fixing arbitrary parameters $L \ge 1$ and $j \ge 2$ Blocki et al. [14] proposed the formula $f(S, L) \doteq \frac{1}{N} \sum_{i:f_i^S \ge j} f_i^S - \frac{N}{(j-1)!L^{j-1}}$ as a lower bound for the expected number of passwords cracked by a rational attacker[2] when the value of a cracked password is at least $v > NL$. In Theorem 9 we prove that the same formula $f(S, L)$ can be used to derive high confidence lower bounds for λ_{NL} (fixing $G = NL$) as below:

Theorem 9. *Given a sample set $S \leftarrow \mathcal{P}^N$ with size N, for any $L \ge 1, t \ge 0, 0 \le \epsilon \le 1$, any integer $j \ge 2$, the expected percentage of passwords cracked by a perfect knowledge attacker making $G = N \times L$ guesses is bounded as below:*

$$\Pr[\lambda_G \ge f(S, L) - t/N - \epsilon] \ge 1 - \delta$$

where $\delta = \exp\left(-2t^2/(Nj^2)\right) + \exp\left(-2N\epsilon^2\right)$, $f(S, L) \doteq \frac{1}{N} \sum_{i:f_i^S \ge j} f_i^S - \frac{N}{(j-1)!L^{j-1}}$ and the randomness is taken over the sample set $S \leftarrow \mathcal{P}^N$.

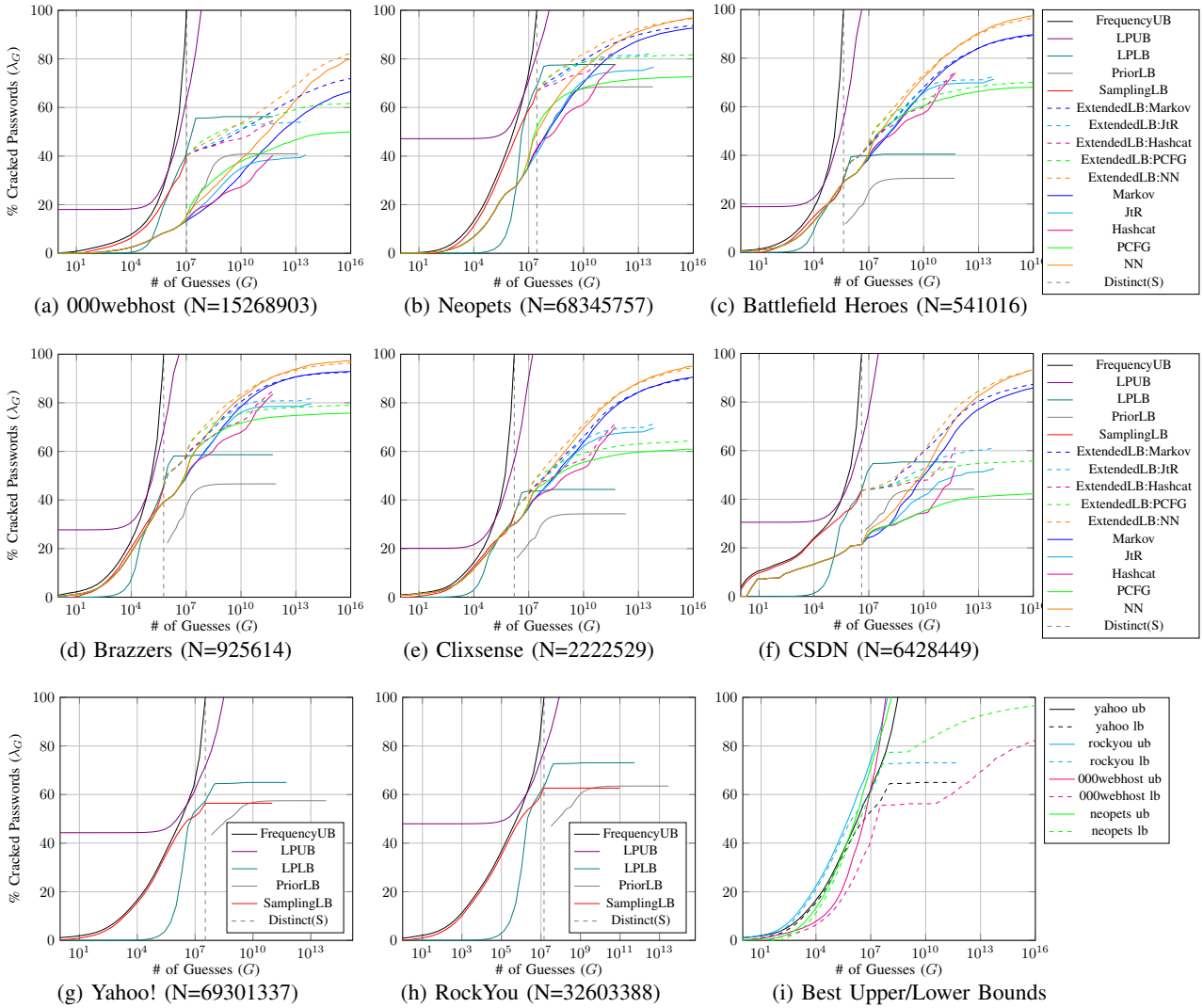
We leave the proofs of Theorem 9 to the full version [52] as the analysis closely follows Blocki et al. [14] and McDiarmid's inequality, and is not the main focus of this paper.

This lower bound is derived only as a comparison to the new approaches we proposed above. Empirical analysis in Section IV demonstrates our new techniques (Corollary 7 and Theorems 5 and 8) yield superior lower bounds.

IV. Empirical Analysis

In this section we apply our techniques to upper/lower bound λ_G to analyze several empirical password datasets. We compare these bounds to guessing curves generated by state of the art password cracking models.

[2]Intuitively, a rational password cracker will continue checking passwords as long as the marginal reward $p_i v$ of checking the next password pwd_i exceeds the marginal guessing cost. Assuming the cost to check each password guess is at most 1 then as $p_i v \ge 1$ the attacker will continue guessing and check the next password pwd_i.



(a) 000webhost (N=15268903) (b) Neopets (N=68345757) (c) Battlefield Heroes (N=541016)

(d) Brazzers (N=925614) (e) Clixsense (N=2222529) (f) CSDN (N=6428449)

(g) Yahoo! (N=69301337) (h) RockYou (N=32603388) (i) Best Upper/Lower Bounds

Fig. 1: 000webhost, Neopets, Battlefield Heroes, Brazzers, Clixsense, CSDN, Yahoo! and RockYou Guessing Curves, and Best Bounds

A. *Datasets*

We use eight empirical password datasets in our analysis: Yahoo!, RockYou, 000webhost, Neopets, Battlefield Heroes, Brazzers, Clixsense and CSDN — for the last six datasets we use the sanitized versions prepared by Liu et al. [8]. Table II in Appendix C provides basic information about each dataset. N represents the total size of the dataset and **# Distinct** represents the number of distinct passwords after eliminating duplicates (i.e. $\mathbf{Distinct}(S)$). Similarly, **# Unique** represents the number of passwords that appear exactly once in S (i.e. $\mathbf{Unique}(S)$). In our analysis we view each dataset S as representing N iid samples from an unknown password distribution. Good-Turing frequency estimation tells us that the total probability mass of unseen passwords is approximately $1 - \sum_{pwd \in S} \Pr[pwd] \approx \frac{\mathbf{Unique}(S)}{N}$ which means for $G = \mathbf{Distinct}(S)$ we have $\lambda_G \geq \sum_{pwd \in S} \Pr[pwd] \approx \frac{N - \mathbf{Unique}(S)}{N}$. Thus, for Yahoo! (resp. CSDN) we should have $\lambda_G \geq 0.575$ (resp. $\lambda_G \geq 0.443$).

One of the password datasets (Yahoo! [15], [16]) is actually a differentially private frequency list and does not include plaintext passwords. For this dataset we can still apply our techniques to upper/lower bound λ_G, but we cannot compare our bounds with empirical password cracking models. While the dataset was slightly perturbed to satisfy differential privacy, Blocki et al. [15] showed that the L1 distortion is minimal i.e., the additional error term is $O(1/\sqrt{N})$.

The Independent Samples Assumption: In our analysis we assume that the dataset S was sampled iid from some unknown distribution $\mathcal{P}$. As we noted previously our linear program allows us to reject datasets S which are clearly inconsistent with our iid assumption. In particular, if the linear program is infeasible this indicates that there is no password distributions $\mathcal{P}$ consistent with the dataset S. This might occur if a large fraction of the dataset was duplicated[3] For example, the description of the LinkedIn frequency corpus [47] indicates

[3]When $s_1, s_2 \leftarrow \mathcal{P}$ are sampled independently it is always possible that $s_1 = s_2$. By contrast, if we sample $s_1 \leftarrow \mathcal{P}$ and then simply fix $s_2 = s_1$ without re-sampling then we would say that the record s_2 was duplicated e.g., a user registers for an account with password s_1 and later registers for a second account with the same password.

that that the dataset contains $177,500,189$ passwords, but only $164,590,819$ unique e-mail addresses [53]. This means that there are more passwords than unique users and it is possible that many of the entries in the LinkedIn frequency corpus are duplicates. We confirmed that the LinkedIn frequency corpus is not iid using our linear program. The linear program was infeasible indicating that we can reject the independent samples hypothesis for this dataset. By contrast, with every other dataset our linear program found feasible solutions. While we cannot absolutely guarantee that our LP rejects every dataset S which is not iid these results increase our confidence that this assumption is a reasonable approximation of reality.

Ethical Considerations. Many of the password datasets we analyze contain stolen passwords that were subsequently leaked on the internet and the usage of this data raises important ethical considerations. We did not crack any new passwords as part of our analysis and the breached datasets are already publicly available. Thus, our usage of the data does not pose any additional risk to users.

B. Password Cracking Models

We use the empirical attack results in Liu et al. [8] to compare with our bounds for 000webhost, Neopets, Battlefield Heroes, Brazzers, Clixsense and CSDN datasets and to generate an extended lower bound using Corollary 7. Liu et al. [8] evaluate 10 password cracking models on the six datasets based on Markov model, PCFG, neural networks, Hashcat, and JtR techniques. In our analysis, we focus on the best performing ones of each password cracking technique: the neural network model (denote as **NN**) in Melicher et al. [1], the Markov 4-gram model (denote as **Markov**) in Dell'Amico and Filippone [22], the original PCFG model [2] (denote as **PCFG**), the extended JtR [8] (denote as **JtR**), and Hashcat (denote as **Hashcat**) implemented in Liu et al. [8]. Some other models (e.g. Markov backoff model [22] and the PCFG model in Komanduri [54]) in Liu et al. [8] may outperform **Markov** or **PCFG** in some ranges of G, but for every value of G their performance is worse than at least one of the models we selected.

C. Evaluation

For each password dataset S we generate upper/lower bounds on λ_G using our results from Section III and compare the upper/lower bounds with the guessing curves derived from password cracking models. Our upper/lower bounds and empirical attack results for 000webhost, Neopets, Battlefield Heros, Brazzers, Clixsense, and CSDN, Yahoo!, and RockYou are shown in Figures 1(a)-(h). The vertical dashed gray line in each subfigure shows $\mathbf{Distinct}(S)$ of the dataset S.

In these figures we use $\texttt{FrequencyUB}(S,G)$ (resp. $\texttt{LPUB}(S,G)$) to denote the upper bound obtained from Corollary 4 (resp. Theorem 8) with password dataset S. Similarly, we use $\texttt{SamplingLB}(S,G)$ (resp. $\texttt{LPLB}(S,G)$) to denote the lower bound obtained by applying Theorem 5 (resp. Theorem 8). For comparison we also plot $\texttt{PriorLB}(\texttt{S},\texttt{G},\texttt{j})$ which denotes the lower bound from Theorem 9 based on results of Blocki et al. [14] — specifically we set $\texttt{PriorLB}(S,G) = \max_{2\le j\le 1000} \texttt{PriorLB}(\texttt{S},\texttt{G},\texttt{j})$ where $\texttt{PriorLB}(\texttt{S},\texttt{G},\texttt{j})$ is the lower bound we obtain after fixing the parameter j in Theorem 9. Two of the lower bounds LBUB and LPLB require us to solve linear programs as a subroutine. We used Gurobi 9.0.1 [55] as our linear programming solver.

For each of the upper/lower bounds there is an error term δ which upper bounds the probability that our bound is wrong. The error term δ will depend on our choice of parameters. For example, in Theorem 9 (resp. Theorem 5) the parameters t,ϵ (resp. t,d) determines $\delta = \exp(-2t^2/(Nj^2)) + \exp(2N\epsilon^2)$ (resp. $\delta = \exp(-2t^2/d)$). Briefly, we always select parameters such that $\delta \le 0.01$. For example, to generate $\texttt{SamplingLB}(S,G)$ we set $d = 2.5\times 10^4$ and $t \ge \sqrt{d\ln(1/\delta)/2}$ in Theorem 5 i.e., we randomly partition S into $D_1 \in \mathcal{P}^{N-d}$ and $D_2 \in \mathcal{P}^d$ and return our lower bound $(h(D_1,D_2,G)-t)/d$ where $h(D_1,D_2,G)$ counts the number of passwords in D_2 that are top G passwords in D_1. See Appendix D for concrete details on how we specify all of these relevant parameters for other upper/lower bounds.

We compare our upper/lower bounds with empirical password guessing curves derived from state of the art password cracking models. Specifically, we compare with the guessing curves generated by Liu et al. [8]. In particular, for each password dataset S they first subsample $25,000$ passwords to obtain a smaller dataset D_2. Then for each password $pwd \in D_2$ and password cracking model M they compute a guessing number $g_{M,pwd}$ for that password (often using Monte Carlo strength estimation [22]). Finally, for a guessing bound G we can estimate that the model will crack $\tilde{\lambda}_{G,M} = |\{pwd \in D_2 : g_{M,pwd} \le G|\,/|D_2|$ passwords. We can also apply Corollary 7 to derive a new lower bound on λ_G by combining the results from our model M with our sampling based lower bound $\texttt{SamplingLB}(S,G)$ — we use $\texttt{ExtendedLB}(S,G,M)$ to denote the extended lower bound and highlight these lower bounds using dotted lines in Figure 1. The guessing curves generated by Liu et al. [8] were generated with a subsample of size $d = 25000$. We used the same value of d when applying the lower bounds $\texttt{SamplingLB}(S,G)$ and $\texttt{ExtendedLB}(S,G,M)$ as this is already sufficient to achieve error bound $\delta \le 0.01$.

D. Discussion

Figure 1 shows that when G is small our upper bound $\texttt{FrequencyUB}(S,G)$ and lower bound $\texttt{SamplingLB}(S,G)$ are very close to each other. For example, when $G \le 262144$ (resp. $G \le 1.048576\times 10^6$) the difference between the upper bound FrequencyUB and the lower bound SamplingLB for Yahoo! dataset is smaller than 1.407% (resp. 2.483%). As long as $\texttt{FrequencyUB}(S,G) - \texttt{SamplingLB}(S,G)$ is small the empirical distribution $\hat{\lambda}_G$ will give us an accurate approximation of the guessing curve λ_G from the real (unknown) password distribution. However, as G becomes large the gap $\texttt{FrequencyUB}(S,G) - \texttt{SamplingLB}(S,G)$ begins to widen e.g., for the Yahoo! dataset when $G = 1.6777216\times 10^7$ we have $\texttt{FrequencyUB}(S,G) - \texttt{SamplingLB}(S,G) = 22.769\%$. While

$\texttt{FrequencyUB}(S,G)$ and lower bound $\texttt{SamplingLB}(S,G)$ give the best upper/lower bounds for smaller G we can see that the bounds reach plateau once $G \geq \mathbf{Distinct}(S)$ e.g., for the Yahoo! dataset $\texttt{SamplingLB}(S,G)$ and $\texttt{FrequencyUB}(S,G)$ remain constant for all $G \geq 3.3885218 \times 10^7$. Once $G \geq \mathbf{Distinct}(S)$ we need new ideas upon the lower bound $\lambda_G \geq \frac{N-\mathbf{Unique}(S)}{N}$ or the upper bound $\lambda_G \leq 1$.

Our linear programming bounds LPUB and LPLB push past the $G \leq \mathbf{Distinct}(S)$ barrier and allow us to obtain tighter upper and lower bounds even when $G \geq \mathbf{Distinct}(S)$. The linear programming bounds (LPUB and LPLB) are worse when G is small e.g., for Yahoo! dataset when $G = 262144$ FrequencyLB and SamplingLB tightly bound λ_G as $34.14\% \leq \lambda_G \leq 35.55\%$ while the LP bounds are $3.58\% \leq \lambda_G \leq 45.66\%$. However, as G increases we find that the linear programming approach yields significantly tighter bounds e.g., for the Yahoo! dataset when $G = 6.7108864 \times 10^7$ our LP bounds show that $61.76\% \leq \lambda_G \leq 77.72\%$ while our frequency and sampling based bounds $56.41\% \leq \lambda_G \leq 100\%$ are much less tight. In such cases when $\hat{\lambda}_G > \texttt{LPUB}(S,G)$ the empirical distribution *should not* be used to estimate the real guessing curve.

Similar to $\texttt{SamplingLB}(S,G)$ our linear programming lower bound $\texttt{LPLB}(S,G)$ also plateaus when G is sufficiently large, but it plateaus at a higher value e.g., 64.97% for Yahoo! dataset. We also note that both of our lower bounds SamplingLB and LPLB dramatically outperform the lower bound PriorLB based on prior work of Blocki et al. [14].

Are Password Cracking Models Guess Efficient? Figure 1 demonstrates empirical cracking models are often much less guess efficient than an attacker who knows the distribution. In particular, if $\tilde{\lambda}_{G,M}$ denotes the percentage of passwords cracked by model M withing G guesses and $\tilde{\lambda}_{G,M} < \max\{\texttt{SamplingLB}(S,G), \texttt{LPLB}(S,G)\}$ then we can be confident that a perfect knowledge attacker would crack more passwords. For example, Figure 1a (resp. Figure 1d) shows that an attacker making $G = 8390551$ (resp. $G = 2097152$) guesses would crack *at most* 14% (resp. 42.14%) of 000webhost (resp. Brazzers) passwords using any password cracking model. By contrast, our lower bounds indicate that an attacker who knows the password distribution will crack *at least* 39.16% (resp. 58.05%) of 000webhost (resp. Brazzers) passwords. These results indicate that even state of the art password cracking models can be improved substantially. One policy implication of this finding is that higher levels of key stretching may be necessary to protect hashed passwords against offline brute-force attacks.

Reducing the Uncertain Region: The lower bound ExtendedLB extends our sampling based approach with empirical password cracking models. The lower bound eventually improves on SamplingLB and LPLB for sufficiently large G. For example, for Brazzers dataset when $G = 10^{16}$ our model based lower bound using neural network attack results ($\texttt{ExtendedLB} : \texttt{NN}$) shows $\lambda_G \geq 96.34\%$ while the best of our other lower bounds is only 58.62%.

Figure 1i plots the best upper/lower bounds (denoted as ub/lb in the figure) for the Yahoo!, RockYou, 000webhost, and Neopets datasets — to avoid overcrowding we plot the best upper/lower bounds for Battlefield Heroes, Brazzers, Clixsense, and CSDN datasets in Appendix C. In particular, we plot $\mathbb{UB}(S,G) = \min\{\texttt{LPUB}(S,G), \texttt{FrequencyUB}(S,G)\}$ (solid curves) and $\mathbb{LB}(S,G) = \max\{\texttt{LPLB}(S,G), \texttt{SamplingLB}(S,G), \texttt{ExtendedLB}: \texttt{NN}(S,G)\}$ (dotted curves). Notice that the lower bound appears to initially plateau before it starts to increase again e.g., for 000webhost dataset the lower bound plateaus at 55.55% when $G = 3.35544 \times 10^7$, barely increases when $3.35544 \times 10^7 \leq G \leq 3.43597 \times 10^{10}$ and then starts to significantly increase again when $G \geq 3.43597 \times 10^{10}$. The initial plateau point is occurs when the lower $\texttt{LPLB}(S,G)$ plateaus and once the empirical guessing curves "catch up" the curve starts to increase again. Each point on the best bounds in Figure 1i hold with probability no less than 0.98 (the union bound of the probabilities that each bound holds).

The real (unknown) guessing curve λ_G lies somewhere in between $\mathbb{LB}(S,G)$ and $\mathbb{UB}(S,G)$. While our work substantially tightens the gap $\mathbb{UB}(S,G) - \mathbb{LB}(S,G)$, there is still an uncertain region between the solid/dotted curves. We conjecture that improved password cracking models may be able to tighten this gap.

The Impact of Sample Size: We use Yahoo! dataset ($N = 69301337$) as an example to analyze the impact of the sample size on the quality of our lower/upper bounds. We generate four subsampled Yahoo! datasets with sample size $N = 10^4, 10^5, 10^6, 10^7$ respectively and generate the best upper/lower bounds for each subsampled dataset (as we did for the original Yahoo! dataset in Figure 1i). We use the same parameter settings as we use for the original Yahoo! dataset, except that for $N = 10^4$ we set $d = 2500$ instead of 25000 when generating $\texttt{SamplingLB}(S,G)$ using Theorem 5. We plot the best upper and lower bounds for different sample sizes in Figure 2. As expected the upper/lower bounds improve as the sample size increases and when $N = 10^7$ the bounds are reasonably close to those obtained from the original dataset. On the negative side when the sample size is just $N = 10^4$ the upper/lower bounds diverge rapidly. Thus, for smaller password datasets such as those collected from a user study our upper/lower bounds may not be particularly useful. This may justify the continued use of password cracking models as a heuristic when analyzing smaller password datasets though we still need to be cautious when drawing conclusions as state-of-the-art password cracking models dramatically under-perform in comparison to an attacker who knows the password distribution. Developing statistical techniques to rigorous compare two password distributions with only a few samples remains an important open research challenge.

Attacker with Partial Knowledge of the Password Distribution: The results in Figure 1 indicate that a perfect-knowledge attacker will often substantially outperform state-of-the-art password models. However, the password cracking models do not require perfect knowledge of the password distribution and it is possible that our lower-bounds on λ_G might overestimate the performance of a real-world attacker. As discussed previously we can also view the lower bound

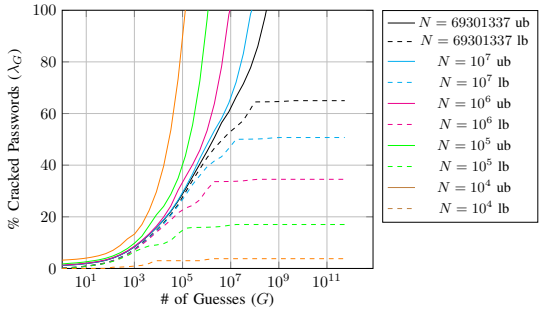


Fig. 2: The Impact of Sample Size on Yahoo! Dataset

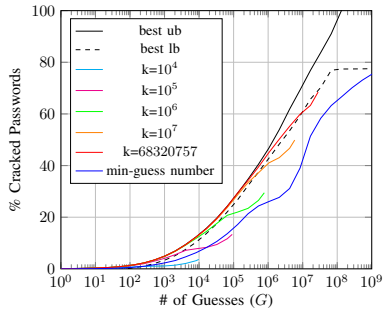


Fig. 3: Attacker's Knowledge (Neopets)

from Theorem 5 as lower bounding the performance of an attacker with partial knowledge of the distribution. Supposing that the attacker has obtained k independent samples $D_1 = \{s_1, \ldots, s_k\} \leftarrow \mathcal{P}$ (e.g., $s_1, \ldots, s_k$ might represent passwords that the attacker has already cracked) from our unknown distribution $\mathcal{P}$ the attacker can build a dictionary $T(D_1, G)$ containing the top G passwords from D_1 and use this dictionary to help crack any remaining passwords. In Figure 3 we evaluate the performance of this partial knowledge attacker for $k \in \{10^4, 10^5, 10^6, 10^7, N-d\}$ where $d = 25,000$ denotes the size of our holdout test set D_2. Figure 3 plots the results for the Neopets dataset with similar plots for 000webhost, Yahoo! and RockYou are deferred to the full version [52] of the paper. The figure shows that as the attacker obtains (cracks) more password samples (k) the performance of the hybrid (partial knowledge) attacker continues to improve and gradually approaches our theoretical lower bounds e.g., fixing the guessing number $G = 524288$ a hybrid attacker building a dictionary from 10^7 samples will crack 10.696% more user passwords than an attacker with 10^6 samples. For comparison, we also plot the guessing curve derived using the minimum guessing number (**min-guess number**) heuristic. Intuitively, we heuristically assume that each particular password will be cracked within G guesses if the password was cracked by *any* of the five password cracking models (see Figure 1b) within G guesses. The minimum guessing number was proposed as a heuristic to model the performance of a real-world attacker [11] who may have more sophisticated dictionaries and rule-lists than publicly available models. Our experimental results indicate that our hybrid partial knowledge attacker will quickly start to outperform the minimum guessing number heuristic and to approach our lower bound from Theorem 5.

V. Applications to Password Policies

In this section we illustrate how our statistical techniques can be used to help guide password policies.

A. Tuning Password Hash Cost Parameters

We first consider the problem of tuning the cost parameter of a password hash function. An offline attacker who has stolen the (salted) cryptographic hash of the user's password can check as many passwords as s/he wants by computing the (salted) cryptographic hashes of likely password guesses to see if they match the stolen hash value. An offline attacker is limited only by the resources s/he is willing to invest cracking and by the cost of repeatedly evaluating the password hash function. Ideally a password hash function should be moderately expensive to compute so that it is impractical for an offline attacker to check millions or billions of password guesses. However, the function cannot be too expensive as the organization must also evaluate the hash function every time a user attempts to login.

Suppose that our organization is considering doubling the cost of the hash function. Assuming that the attacker's resources remain constant this would reduce the number of passwords that the offline attacker can check from G to $G/2$ and the probability that the attacker cracks the user's password would decrease from λ_G to $\lambda_{G/2}$. However, doubling the hash cost parameter will require the organization to invest additional computing resources to handle user authentications. Thus, the organization will only make the change if the security benefits are substantial enough.

We can use our statistical techniques to upper/lower bound the security gain $\lambda_G - \lambda_{G/2}$ (resp. $\lambda_G - \lambda_{G/2^x}$) when increasing hashing costs by a factor of 2 (resp. 2^x). In particular, if $\mathbb{UB}(S, G)$ (resp. $\mathbb{LB}(S, G)$) denotes our upper/lower bounds on λ_G then we know that

$$\mathbb{LB}(S, G) - \mathbb{UB}(S, G/2^x) \leq \lambda_G - \lambda_{G/2^x} \leq \mathbb{UB}(S, G) - \mathbb{LB}(S, G/2^x) \ .$$

Figure 4 plots our upper/lower bounds on $\lambda_G - \lambda_{G/2^x}$ for $x \in \{1, 2, 3, 4, 5\}$ for the Neopets dataset — similar results for Yahoo!, RockYou and 000webhost are deferred to the full version [52]. For example, suppose that the attacker was originally able to attempt $G = 1.67 \times 10^7$ password guesses before we increase our guessing costs by 2^5. The figure indicates that this policy change will reduce the number of cracked passwords by *at least* 25.2% and *at most* 37.7%. If reducing the potential damage of an offline attack by 25.2% (resp. 37.7%) is (resp. is not) worth the additional costs then the organization should (resp. should not) increase the hash cost parameter. We remark that an organization that transitions away from a password hash function like PBKDF2 or BCRYPT to a modern memory hard function like scrypt [26], Argon2 [24] or DRSample [25] will substantially increase hashing costs.

We remark that upper/lower bounding $\lambda_G - \lambda_{G/b}$ can be useful when defending against online password spraying attacks. Suppose that our organization is considering adopting a stricter version of its lockout policy where an account is locked whenever there are at least G/b (as opposed to G) incorrect

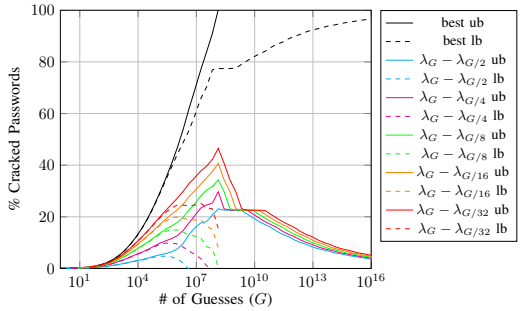


Fig. 4: Tuning Password Hash Cost Parameters (Neopets)

login attempts within a 30 day window. For reference, NIST Authentication Guidelines [56] suggests limiting the number of incorrect login attempts to $G = 100$ within a 30 day window. Adopting a stricter lockout policy comes with a usability cost so our organization would only adopt the policy if the security benefits ($\lambda_G - \lambda_{G/b}$) are substantial enough e.g., with $G = 128$ in Figure 4 we have $0.491\% \leq \lambda_G - \lambda_{G/2} \leq 0.579\%$.

B. Comparing Password Distributions

The Yahoo! frequency corpus [15], [16] allows us to compare password distributions along several dimensions: gender, account tenure and composition policy (before/after) the adoption of a six character minimum restriction. The comparison results are shown in Figure 7 in Appendix C.

Measuring Shifts in The Password Distributions over Time We generated upper and lower bounds for Yahoo! passwords with account tenure of 5-10 years and account tenure below 1 year as shown in Figure 7a. We found statistically significant evidence for Bonneau's claim [16] that there is "a weak trend towards improvement over time, with more recent accounts having slightly stronger passwords". For example, when our attacker makes $G = 8192$ guesses, at least 16.86% old account passwords (5-10 years) will be guessed while at most 14.40% new account passwords (0-1 year) will be guessed (confidence: 98%). One caveat is that the weak trend towards stronger passwords may be partially explained by Yahoo!'s adoption of the six-character minimum requirement.

Measuring the Effect of Gender on Password Strength Bonneau [16] previously found that (self-reported) gender had a small/split effect on password security with "male-chosen passwords being slightly more vulnerable to online attack and slightly stronger against offline attack." We are able to provide statically significant justification for these claims using our upper/lower bounds as shown in Figure 7c. First, we can provide statistically significant evidence that female passwords are more resistant to an attacker making $G \leq 128$ guesses. For example, when $G = 32$ we find that the attacker will crack at least 2.776% (resp. at most 2.283%) of male (resp. female) passwords. For $512 \leq G \leq 1048576$ the conclusions are reversed e.g., when $G = 131072$ we find that the attacker will crack at most 30.050% (resp. at least 32.219%) of male (resp. female) passwords. (Note: When $G > 1048576$ the upper/lower bounds start to diverge preventing us from making definitive comparisons. Also, when $G = 256$ the upper/lower bounds for male/female passwords are very close).

Measuring The Effect of Password Requirements At Registration We generated upper/lower bounds for Yahoo! passwords that were selected with (and without) a six character minimum restriction as shown in Figure 7b. Bonneau [16] previously concluded that the change made "almost no difference" in security against online guessing while slightly increasing the resistance to offline attacks. By contrast, we find statistically significant evidence that an online attacker making at most $G = 8$ guesses will crack more passwords picked under the six character minimum restriction. For example, when an online attacker makes $G = 8$ guesses, we can say (with 98% confidence) that the attacker will crack between 1.393% to 1.610% (resp. 1.645% to 1.880%) of passwords picked without the restriction (resp. with the restriction). The findings are reversed for an attacker making $256 \leq G \leq 524288$ guesses yielding statistically significant evidence for Bonneau's finding that passwords picked under the six-character minimum restriction are more resistant to offline attacks. (Note: for larger values of G the upper/lower bounds begin to diverge preventing us from making a definitive comparison).

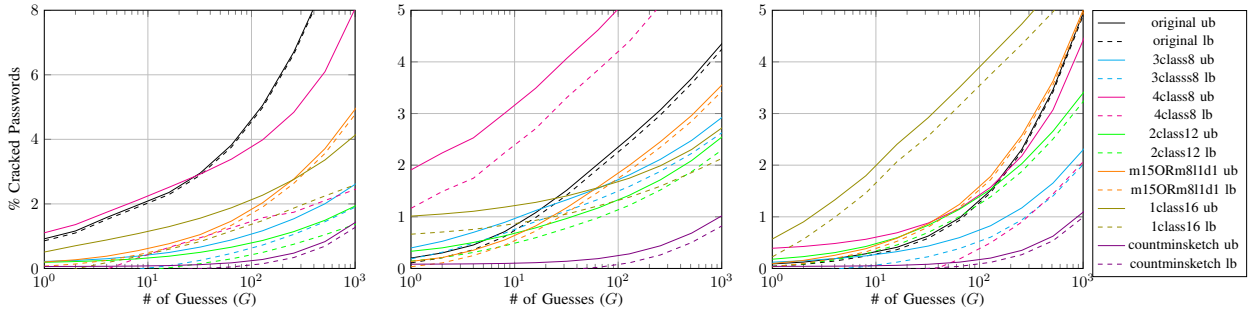
C. Password Composition Policies

We now turn our attention to the problem of identifying secure password composition policies which yield password distributions that are more resistant to online (resp. offline) password cracking attacks. One immediate challenge we face is that, with the exception of the Yahoo! frequency corpus, none of the available password datasets record the passwords that each user would have selected in response to (additional) restrictions. One potential remedy would be to run a large user study to ask users what password they would select under a variety of restrictions i.e., obtaining samples from each of the resulting distributions for each password composition policy. However, as we previously discussed one limitation of our statistical techniques is that we need a reasonably large number of samples from each distribution before we would be able to draw meaningful comparisons e.g., see Figure 2. In practice, the number of participants N (e.g., $N \leq 10^4$) in any user study would be heavily constrained by practical budget considerations and would be too small to allow us to draw meaningful comparisons using our statistical techniques.

We address the challenges above by following [27] and making a heuristic assumption about the way that users respond to password restrictions. Suppose that we are given a predicate `Allowed` describing our password composition policy i.e., $\texttt{Allowed}(pwd) = 1$ if and only if users are allowed to select the password pwd. If $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) denotes the probability distribution before (resp. after) adopting the composition policy then the normalized probabilities model [27] says that for each password pwd with $\texttt{Allowed}(pwd) = 1$ we have

$$\Pr_{x \leftarrow \mathcal{P}_2}[x = pwd] = \Pr_{x \leftarrow \mathcal{P}_1}[x = pwd \mid \texttt{Allowed}(x) = 1] \ .$$

Intuitively, we can imagine that each user utilizes rejection sampling and repeatedly samples passwords from $\mathcal{P}_1$ until



(a) RockYou (b) 000webhost (c) Neopets

Fig. 5: Comparison of Password Composition Policies

s/he finds one that is consistent with the composition policy. While this heuristic assumption may not perfectly model how users respond to password restrictions, the assumption allows us to apply our statistical techniques to compare candidate password composition policies and identify the most promising candidates for further evaluation. In particular, if the password dataset S_1 denotes N iid samples from the distribution $\mathcal{P}_1$ then we can filter S_1 by removing passwords that are incompatible with the composition policy to obtain $S_2 = \{pwd \in S \ : \ Allowed(pwd) = 1\}$. Now S_2 can be viewed as $N' \leq N$ iid samples from $\mathcal{P}_2$. Thus, we can apply our statistical techniques to obtain upper/lower bounds ($\mathbb{UB}(S_2, G)$ and $\mathbb{LB}(S_2, G)$) for the new distribution. In our analysis we primarily focus on smaller guessing numbers G to determine whether or not the new distribution is more/less resistant to online cracking.

We analyze a wide variety of candidate PCPs. In general, we use $x$class$y$ to denote the policy which requires the user to pick a password with *at least* y characters total coming from *at least* x distinct character classes (lowercase letters, uppercase letters, digits, special characters). For example, 4class8 requires passwords that are at least 8 characters long and include at least one lowercase, one uppercase, one digit and one special character. We also consider the PCP used by Github.com (at least 15 characters OR at least 8 characters including a number and a lower case letter). Finally, we consider a proposal of Schechter et al. [57] to use a count-min-sketch data structure to estimate the frequency of each user password and ban passwords whose (estimated) frequency is above a threshold $r \times N$. We instantiated this policy with the differentially private count-min-sketch as described in [48] which adds laplace noise to each cell in the count-sketch data-structure to preserve ϵ-differential privacy. Since the count-min-sketch is trained based on sampled passwords S we adopted the following approach to ensure that the comparison with other PCPs is fair. 1) We randomly partition the dataset S into two equal size datasets S_{train} and S_{test} of size $N/2$. 2) The dataset S_{train} is used to construct the noisy count-min-sketch (depth=5, width $=10^6$, 20MB of space) using the differential privacy parameter $\epsilon = 0.1$ and then discarded. 3) We define the password policy $\texttt{Allowed}(pwd) = 1$ if and only the estimated frequency of pwd is *at most* $r \times N/2$ — we fixed $r = 10^{-5}$ in our experiments below and explore different threshold parameters $r \in \{10^{-4}, 10^{-5}, 10^{-6}\}$ in the full version [52]. Because the count-sketch was trained only using passwords from the dataset S_{train} we can filter S_{test} to obtain fresh samples that are consistent with the composition policy.

Figure 5 shows the upper and lower bounds of different PCPs as well as the upper/lower bounds of the original dataset (**original ub**, **original lb**) of the three datasets. To avoid clutter we omitted PCPs (e.g. $1class8$, $2class8$, $1class12$) that performed similarly to others. Since we are focusing primarily on online attacks, we enlarge the plots and only show $1 \leq G \leq 1000$ in this figure. Note: to get tighter bounds with small G, we set the parameter d in `SamplingLB` to be $N/2$ and set the error rate ϵ of `FrequencyUB` to be 0.01. We leave the full plots with $G > 1000$ in the full version [52].

An interesting finding is that no single PCP, excluding count-min-sketch, performs well across all datasets and parameter ranges. For example, 1.6% of the 000webhost passwords that comply with the policy are $P@ssw0rd$ so applying the $4class8$ rule actually increases the percentage of cracked passwords as shown in Figure 5b. Similarly, the $1class16$ policy performs particularly poorly on the Neopets dataset. On the positive side the count-min sketch PCP universally performed well across all three datasets and for all guessing parameters G. For example, when $G = 128$ applying the count-min sketch PCP to the 000webhost dataset reduces the percentage of cracked passwords from at least 2.434% to *at most* 0.283%. For comparison, applying the $4class8$ (resp. $1class16$) *increases* (resp. *decreases*) the percentage of cracked passwords to at least 4.395% (resp. at least 1.379%). Our analysis supports the proposal of Schechter et al. [57] to base password composition policies on (estimated) password frequency with the caveat that our analysis in this section does depend on a heuristic assumption about how password composition policies impact the distribution. We also remark that when the guessing number G is low that it is more likely that a real world attacker will closely resemble a perfect knowledge attacker. In particular, the attacker only needs to know the top G passwords and we expect that each of these passwords will occur relatively frequently making them the easiest to learn.

D. Discussion: Obtaining Password Samples

Before applying our statistical techniques we implicitly assume that our organization has been able to obtain independent

samples $S = (s_1, \ldots, s_N)$ from the unknown distribution $\mathcal{P}$ over user passwords. While our organization can always rely on a breached password dataset such as 000webhost or RockYou, it is unlikely that the password samples will be perfectly representative of the current user password distribution e.g., due to varying demographic factors, language, culture, account value and password restrictions. Ideally, an organization would obtain the samples $S = (s_1, \ldots, s_N)$ from *its own* users before applying our statistical framework. While the collection of such a sample raises security and privacy concerns, we stress that we do not require plaintext passwords in order to apply our statistical techniques. Bonneau [16] already developed an efficient framework to securely collect an anomymized password frequency list and Blocki et al. [15] developed an efficient differentially private algorithm that was used to publish a differentially private version of Yahoo! frequency corpus that Bonneau collected ($N \approx 70$ million users). An organization could adapt this same framework to securely and privately collect a password frequency corpus from its own users and then apply our statistical techniques to characterize the attacker's guessing curve.

VI. Conclusion

We introduced several statistical techniques to upper and lower bound λ_G the performance of a password cracker who knows the distribution from which passwords were sampled. Our upper/lower bounds hold with high confidence and can be derived from a password dataset even when the real password distribution is unknown to us. We applied our technique to analyze several large empirical password datasets. Our analysis demonstrates that the empirical guessing curve closely matches the real guessing curve as long as G is not too large, and highlights that state-of-the-art password cracking models are often far less guess efficient than a perfect knowledge attacker. For example, with the 000webhost dataset the state-of-the-art password cracking models indicate the attacker making 8338608 guesses would crack any password with probability 14% while our lower bound shows the probability is at least 39.16% with high confidence (99%). This shows that even the most sophisticated password cracking models still have a large room for improvement. We also demonstrate how to apply our theoretical bounds to determine whether or not particular password interventions (e.g., key-stretching, imposing restrictions on the passwords users can pick) yield effective defenses. While our results significantly narrows the uncertain region for λ_G, there are regions where our best upper/lower bounds diverge significantly. Reducing this gap will help us to better understand the distribution over user chosen passwords and is an important challenge for future research.

Acknowledgments

This research was supported in part by the National Science Foundation under awards CNS #1755708 and CNS #2047272, a gift from Protocol Labs, and by a Purdue Big Ideas award. We would like to thank anonymous reviewers for constructive feedback which helped us to improve the presentation of this paper.

References

[1] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks," in *USENIX Security 2016*, T. Holz and S. Savage, Eds. USENIX Association, Aug. 2016, pp. 175–191.

[2] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *2009 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2009, pp. 391–405.

[3] R. Veras, C. Collins, and J. Thorpe, "On semantic patterns of passwords and their security impact," in *NDSS 2014*. The Internet Society, Feb. 2014.

[4] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *ACM CCS 2005*, V. Atluri, C. Meadows, and A. Juels, Eds. ACM Press, Nov. 2005, pp. 364–372.

[5] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from Markov models," in *NDSS 2012*. The Internet Society, Feb. 2012.

[6] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2014, pp. 689–704.

[7] M. Dürmuth, F. Angelstorf, C. Castelluccia, D. Perito, and A. Chaabane, "Omen: Faster password guessing using an ordered markov enumerator," in *International Symposium on Engineering Secure Software and Systems*. Springer, 2015, pp. 119–132.

[8] E. Liu, A. Nakanishi, M. Golla, D. Cash, and B. Ur, "Reasoning analytically about password-cracking software," in *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2019, pp. 380–397.

[9] "John the ripper," https://www.openwall.com/john/, accessed March 15, 2021.

[10] "Hashcat," https://hashcat.net/hashcat/, accessed March 15, 2021.

[11] B. Ur, S. M. Segreti, L. Bauer, N. Christin, L. F. Cranor, S. Komanduri, D. Kurilova, M. L. Mazurek, W. Melicher, and R. Shay, "Measuring real-world accuracies and biases in modeling password guessability," in *USENIX Security 2015*, J. Jung and T. Holz, Eds. USENIX Association, Aug. 2015, pp. 463–481.

[12] R. Chatterjee, A. Athayle, D. Akhawe, A. Juels, and T. Ristenpart, "pASSWORD tYPOS and how to correct them securely," in *2016 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2016, pp. 799–818.

[13] R. Chatterjee, J. Woodage, Y. Pnueli, A. Chowdhury, and T. Ristenpart, "The TypTop system: Personalized typo-tolerant password checking," in *ACM CCS 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM Press, Oct. / Nov. 2017, pp. 329–346.

[14] J. Blocki, B. Harsha, and S. Zhou, "On the economics of offline password cracking," in *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 853–871.

[15] J. Blocki, A. Datta, and J. Bonneau, "Differentially private password frequency lists," in *NDSS 2016*. The Internet Society, Feb. 2016.

[16] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *2012 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2012, pp. 538–552.

[17] J. Blocki and A. Datta, "CASH: A cost asymmetric secure hash algorithm for optimal password protection," in *CSF 2016Computer Security Foundations Symposium*, M. Hicks and B. Köpf, Eds. IEEE Computer Society Press, 2016, pp. 371–386.

[18] W. Bai and J. Blocki, "Dahash: Distribution aware tuning of password hashing costs," *arXiv preprint arXiv:2101.10374*, 2021.

[19] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *2012 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2012, pp. 553–567.

[20] G. Valiant and P. Valiant, "Estimating the unseen: Improved estimators for entropy and other properties," *Journal of the ACM (JACM)*, vol. 64, no. 6, pp. 1–41, 2017.

[21] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[22] M. Dell'Amico and M. Filippone, "Monte Carlo strength evaluation: Fast and reliable password checking," in *ACM CCS 2015*, I. Ray, N. Li, and C. Kruegel, Eds. ACM Press, Oct. 2015, pp. 158–169.

[23] J.-P. A. et al., "Password hashing competition," 2015, https://password-hashing.net/.

[24] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: new generation of memory-hard functions for password hashing and other applications," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 292–302.

[25] J. Alwen, J. Blocki, and B. Harsha, "Practical graphs for optimal side-channel resistant memory-hard functions," in *ACM CCS 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM Press, Oct. / Nov. 2017, pp. 1001–1017.

[26] C. Percival, "Stronger key derivation via sequential memory-hard functions," 2009.

[27] J. Blocki, S. Komanduri, A. Procaccia, and O. Sheffet, "Optimizing password composition policies," in *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, ser. EC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 105–122. [Online]. Available: https://doi.org/10.1145/2482540.2482552

[28] R. Morris and K. Thompson, "Password security: A case history," *Communications of the ACM*, vol. 22, no. 11, pp. 594–597, 1979.

[29] N. Provos and D. Mazieres, "Bcrypt algorithm," in *USENIX*, 1999.

[30] B. Kaliski, "Pkcs# 5: Password-based cryptography specification version 2.0," 2000.

[31] A. Juels and R. L. Rivest, "Honeywords: making password-cracking detectable," in *ACM CCS 2013*, A.-R. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM Press, Nov. 2013, pp. 145–160.

[32] K. Thomas, J. Pullman, K. Yeo, A. Raghunathan, P. G. Kelley, L. Invernizzi, B. Benko, T. Pietraszek, S. Patel, D. Boneh, and E. Bursztein, "Protecting accounts from credential stuffing with password breach alerting," in *USENIX Security 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, Aug. 2019, pp. 1556–1571.

[33] L. Li, B. Pal, J. Ali, N. Sullivan, R. Chatterjee, and T. Ristenpart, "Protocols for checking compromised credentials," in *ACM CCS 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM Press, Nov. 2019, pp. 1387–1403.

[34] A. Everspaugh, R. Chatterjee, S. Scott, A. Juels, and T. Ristenpart, "The pythia PRF service," in *USENIX Security 2015*, J. Jung and T. Holz, Eds. USENIX Association, Aug. 2015, pp. 547–562.

[35] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung, "Fourth-factor authentication: Somebody you know," in *ACM CCS 2006*, A. Juels, R. N. Wright, and S. De Capitani di Vimercati, Eds. ACM Press, Oct. / Nov. 2006, pp. 168–178.

[36] M. Mannan and P. C. van Oorschot, "Leveraging personal devices for stronger password authentication from untrusted computers," *J. Comput. Secur.*, vol. 19, no. 4, p. 703–750, Dec. 2011.

[37] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor, "Encountering stronger password requirements: user attitudes and behaviors," in *Proceedings of the Sixth Symposium on Usable Privacy and Security*, ser. SOUPS '10. New York, NY, USA: ACM, 2010, pp. 2:1–2:20. [Online]. Available: http://doi.acm.org/10.1145/1837110.1837113

[38] J. Campbell, W. Ma, and D. Kleeman, "Impact of restrictive composition policy on user password choices," *Behaviour & Information Technology*, vol. 30, no. 3, pp. 379–388, 2011.

[39] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of passwords and people: measuring the effect of password-composition policies," in *CHI*, 2011, Conference Proceedings, pp. 2595–2604. [Online]. Available: http://dl.acm.org/citation.cfm?id=1979321

[40] A. Adams and M. A. Sasse, "Users are not the enemy," *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, 1999.

[41] S. Komanduri, R. Shay, L. F. Cranor, C. Herley, and S. E. Schechter, "Telepathwords: Preventing weak passwords by reading users' minds," in *USENIX Security 2014*, K. Fu and J. Jung, Eds. USENIX Association, Aug. 2014, pp. 591–606.

[42] J. Bonneau and S. E. Schechter, "Towards reliable storage of 56-bit secrets in human memory," in *USENIX Security 2014*, K. Fu and J. Jung, Eds. USENIX Association, Aug. 2014, pp. 607–623.

[43] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," *IEEE Security & privacy*, vol. 2, no. 5, pp. 25–31, 2004.

[44] W. Yang, N. Li, O. Chowdhury, A. Xiong, and R. W. Proctor, "An empirical study of mnemonic sentence-based password generation strategies," in *ACM CCS 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM Press, Oct. 2016, pp. 1216–1229.

[45] J. Blocki, M. Blum, and A. Datta, "Naturally rehearsing passwords," in *ASIACRYPT 2013, Part II*, ser. LNCS, K. Sako and P. Sarkar, Eds., vol. 8270. Springer, Heidelberg, Dec. 2013, pp. 361–380.

[46] J. Blocki, S. Komanduri, L. F. Cranor, and A. Datta, "Spaced repetition and mnemonics enable recall of multiple strong passwords," in *NDSS 2015*. The Internet Society, Feb. 2015.

[47] B. Harsha, R. Morton, J. Blocki, J. Springer, and M. Dark, "Bicycle attacks considered harmful: Quantifying the damage of widespread password length leakage," *Computers & Security*, vol. 100, p. 102068, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820303412

[48] J. Blocki and W. Zhang, "Dalock: Distribution aware password throttling," *arXiv preprint arXiv:2005.09039*, 2020.

[49] Y. Tian, C. Herley, and S. Schechter, "Stopguessing: Using guessed passwords to thwart online guessing," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 576–589.

[50] W. Bai, J. Blocki, and B. Harsha, "Information signaling: A counter-intuitive defenseagainst password cracking," *arXiv preprint arXiv:2009.10060*, 2020.

[51] C. McDiarmid, "On the method of bounded differences," *Surveys in combinatorics*, vol. 141, no. 1, pp. 148–188, 1989.

[52] J. Blocki and P. Liu, "Towards a rigorous statistical analysis of empirical password datasets," *CoRR*, vol. abs/2105.14170, 2021. [Online]. Available: https://arxiv.org/abs/2105.14170

[53] R. Redman, "Linkedin revisited – full 2012 hash dump analysis," https://blog.korelogic.com/blog/2016/05/19/linkedin_passwords_2016, May 19, 2016.

[54] S. Komanduri, "Modeling the adversary to evaluate password strength with limited samples," *Ph.D. dissertation*, 2016.

[55] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021. [Online]. Available: http://www.gurobi.com

[56] P. Grassi and J. Fenton, "Nist sp800-63b: Digital authentication guideline," Technical report, NIST, Reston, VA, 2016. https://pages.nist.gov/800-63-3/sp800-63b.html, Tech. Rep., 2016.

[57] S. Schechter, C. Herley, and M. Mitzenmacher, "Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks," in *5th USENIX Workshop on Hot Topics in Security (HotSec 10)*. Washington, DC: USENIX Association, Aug. 2010.

[58] D. Malone and K. Maher, "Investigating the distribution of password choices," in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 301–310.

[59] D. Wang and P. Wang, "On the implications of zipf's law in passwords," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 111–131.

[60] N. Cubrilovic, "Rockyou hack: From bad to worse," https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/, December 15, 2009.

[61] X. Yang, "Chinese internet suffers the most serious user data leak in history," https://blogs.forcepoint.com/security-labs/chinese-internet-suffers-most-serious-user-data-leak-history, December 26, 2011.

[62] D. Goodin, "13 million plaintext passwords belonging to webhost users leaked online," https://arstechnica.com/information-technology/2015/10/13-million-plaintext-passwords-belonging-to-webhost-users-leaked-online/, October 28, 2015.

[63] J. Cox, "Another day, another hack: Tens of millions of neopets accounts," https://motherboard.vice.com/en_us/article/ezpvw7/neopets-hack-another-day-another-hack-tens-of-millions-of-neopets-accounts, May 5, 2016.

[64] J. Walker, "Lulzsec over, release battlefield heroes data," https://www.rockpapershotgun.com/2011/06/26/lulzsec-over-release-battlefield-heroes-data/, June 26, 2011.

[65] J. Cox, "Nearly 800,000 brazzers porn site accounts exposed in forum hack," https://motherboard.vice.com/en_us/article/vv7pgd/nearly-800000-brazzers-porn-site-accounts-exposed-in-forum-hack, September 5, 2016.

[66] D. Goodin, "6.6 million plaintext passwords exposed as site gets hacked to the bone," https://arstechnica.com/information-technology/2016/09/plaintext-passwords-and-wealth-of-other-data-for-6-6-million-people-go-public/, September 13, 2016.

APPENDIX A
THE LINEAR PROGRAM LPUPPER

Below we show the linear program LPupper that is used to generate the upper bound ($\texttt{LPUB}(S,G)$) in Theorem 8.

Linear Programming Task 3:
$\texttt{LPupper}(G, X_l, F^S, idx, i', \epsilon_{\mathbf{2}}, \epsilon_{\mathbf{3}}, \hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}})$
Input Parameters: G, $X_l = \{x_1, ..., x_l\}$, $F^S = \{F_1^S, ..., F_N^S\}$, idx, i', $\epsilon_2 = \{\epsilon_{2,0}, \ldots, \epsilon_{2,i'}\}$, $\hat{\mathbf{x}}_{\epsilon_{\mathbf{3}}} = \{\hat{x}_{\epsilon_{3,0}}, \ldots, \hat{x}_{\epsilon_{3,i'}}\}$, $\epsilon_{\mathbf{3}} = \{\epsilon_{3,i} = \frac{1}{q^{i+1}} \left(\frac{1-\hat{x}_{\epsilon_{3,i}}}{1-q\hat{x}_{\epsilon_{3,i}}}\right)^{N-i} - 1, 0 \le i \le i'\}$
Variables: $h_1, ..., h_l, c, p$
Objective: $\max\left(\sum_{j<idx} h_j \times x_j + c \times x_{idx}\right)$
Constraints:

1) $\sum_{j<idx} h_j + c = G$
2) $\forall 0 \le i \le i'$:
   a) for $i = 0$, $\frac{1}{1+\epsilon_{3,i}}(\frac{(i+1)F_{i+1}^S}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} - p - \texttt{bpdf}(i, N, q\hat{x}_{\epsilon_{3,i}})) \le \sum_{j=1}^{l} h_j \times x_j \times \texttt{bpdf}(i, N, x_j) \le q^{i+1}(\frac{(i+1)F_{i+1}^S}{N-i} + \epsilon_{2,i} - p \times \texttt{bpdf}(i, N, x_l))$
   b) for $1 \le i \le i'$, $\frac{1}{1+\epsilon_{3,i}}(\frac{(i+1)F_{i+1}^S}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} - p \times \texttt{bpdf}(i, N, x_l) - \texttt{bpdf}(i, N, q\hat{x}_{\epsilon_{3,i}})) \le \sum_{j=1}^{l} h_j \times x_j \times \texttt{bpdf}(i, N, x_j) \le q^{i+1}(\frac{(i+1)F_{i+1}^S}{N-i} + \epsilon_{2,i})$
3) $1 - p \le \sum_{j=1}^{l} h_j \times x_j \le q \times (1-p)$
4) $0 \le c \le h_{idx}$

(**Note:** we consider $idx = 1, 2, ..., l+1$. When $idx = l+1$, we define $h_{l+1} = G$ and $x_{l+1} = x_l$.)

APPENDIX B
ZIPF'S LAW IN PASSWORDS

Zipf's law [21], [58], [59] has been proposed as reasonable model for the password distribution λ_G. For example, CDF Zipf's law estimates that $\lambda_G \approx yG^r$ where the constant $y, r > 0$ are tuned based on the sample S. Several papers [14], [21], [59] find that CDF Zipf's law closely fits all known empirical password distributions. However, there is no theoretical guarantee that the estimate yG^r is close to λ_G. we compare our upper/lower bounds with the CDF-Zipf curves using the parameters y, r from [59] for the datasets RockYou, Battlefield Heroes, 000webhost, CSDN and [14] for the more recent Yahoo! dataset [15], [16]. The plots and parameter settings for Rockyou, Yahoo! and CSDN datasets are shown in Figure 6 and Table I. The comparison results for 000webhost and Battlefield Heroes are similar to the other three datasets. We leave their plots and parameter settings in the full version [52]. In all of the plots the CDF-Zipf plot (green) is close to our best upper bound (red). For the Battlefield, CSDN and 000webhost datasets the CDF-Zipf plot (green) lies in between our best upper bound (red) and our best lower bound (blue) indicating that the curve yG^r is consistent with our statistical bounds. For the RockYou and Yahoo! datasets the CDF-Zipf plots (green) often lie above the red upper bound e.g., for the RockYou (resp. Yahoo!) dataset when $G = 33554432$ (resp. $G = 134217728$) we have $yG^r \ge 0.96$ (resp. $yG^r \ge 0.98$) while our upper bounds imply that $\lambda_G \le 0.87$ (resp. $\lambda_G \le 0.86$). In such cases we can confidently state that the CDF-Zipf curve overestimates λ_G.

TABLE I: CDF Zipf's Law Parameters [14], [59]

Dataset (S)	y	r
Yahoo! [15]	0.03315	0.1811
RockYou [60]	0.037433	0.187227
CSDN [61]	0.058799	0.148573

TABLE II: Basic Information for Password Datasets

Dataset (S)	**# Passwords** (N)	**# Distinct**	**# Unique**
Yahoo! [15]	69301337	33895873	29452171
RockYou [60]	32603388	14344391	11884632
000webhost [62]	15268903	10592935	9006529
Neopets [63]	68345757	27987227	21509860
Battlefield Heroes [64]	541016	416130	373549
Brazzers [65]	925614	587934	491136
Clixsense [66]	2222529	1628577	1455585
CSDN [61]	6428449	4037749	3581824

APPENDIX C
MISSING FIGURES AND TABLES

The basic information of the eight datasets used in this paper is in Table II.

Figure 7 shows the comparison results of Yahoo! frequency corpus discussed in Section V-B.

We have shown the best upper/lower bounds for Yahoo!, RockYou, 000webhost, and Neopets in Figure 1i in the main body of the paper. Here we plot the best upper/lower bounds for Battlefield Heroes, Brazzers, Clixsense, and CSDN datasets in Figure 8.

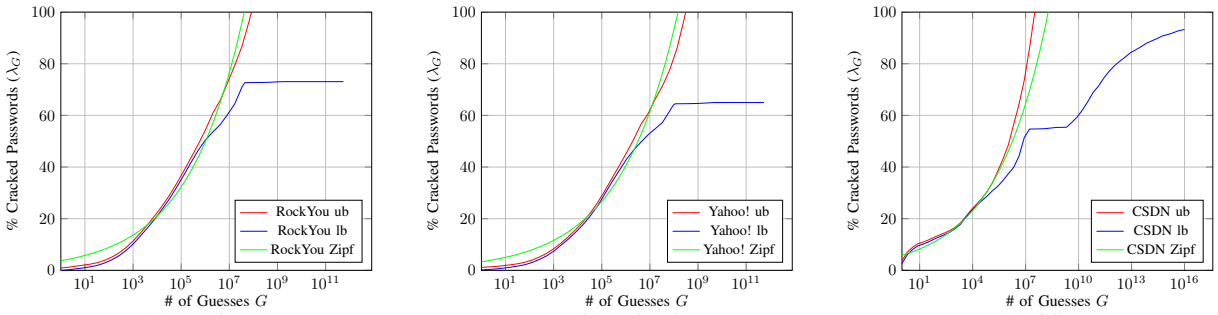
APPENDIX D
PARAMETER SETTING

To generate high-confidence theoretical bounds of λ_G and $\lambda(S,G)$, we assign the parameters values to guarantee every bound in Section III holds with probability *at least* 0.99.

Recall that the parameter ϵ in Corollary 4, Theorem 5, Corollary 7, Theorem 9 bound the difference between λ_G and $\lambda(S,G)$. We fix this parameter as $\epsilon = \epsilon_1 = \left(\frac{\ln(\delta_1)}{-2N}\right)^{\frac{1}{2}}$ where we set $\delta_1 = 0.00009$ for all bounds of λ_G and $\lambda(S,G)$. Then the upper bound in Corollary 4 ($\texttt{FrequencyUB}(S,G)$) holds with probability at least $1 - \delta_1 \ge 0.99$.

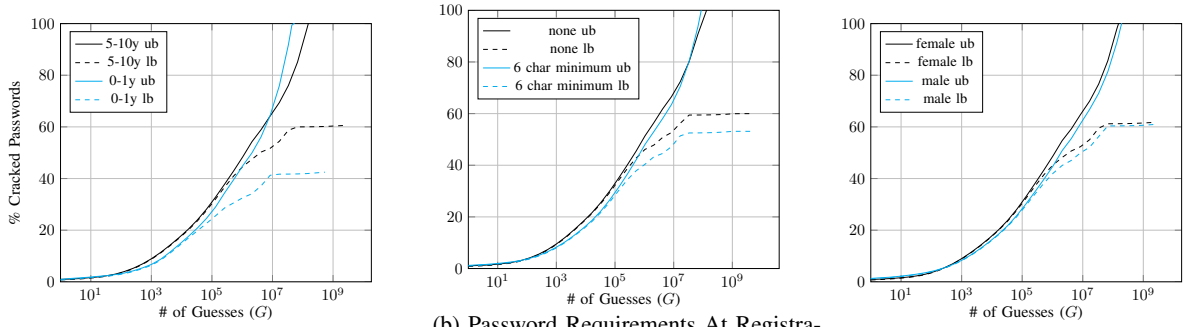
We denote $\delta_{2,j} = \exp\left(\frac{-2t^2}{Nj^2}\right)$ in Theorem 9. We set $\delta_{2,j} = 0.01 - \delta_1$ (i.e. $t = \left(\frac{Nj^2 \ln(\delta_{2,j})}{-2}\right)^{\frac{1}{2}}$) such that the lower bound in Theorem 9 ($\texttt{PriorLB}(\texttt{S}, \texttt{G}, \texttt{j})$) holds with probability at least $1 - \delta_1 - \delta_{2,j} \ge 0.99$ for any $j \ge 2$.

We denote $\delta_3 = \exp\left(\frac{-2t^2}{d}\right)$ in both Theorem 5 and Corollary 7. We set $d = 25000$ and $\delta_3 = 0.01 - \delta_1$ (i.e. $t = \left(\frac{d \ln(\delta_3)}{-2}\right)^{\frac{1}{2}}$) such that the lower bound in Corollary 7 ($\texttt{ExtendedLB}(S,G)$) extends the lower bound in Theorem 5 ($\texttt{SamplingLB}(S,G)$), and both of them hold with probability at least $1 - \delta_3 \ge 0.99$.



(a) RockYou (b) Yahoo! (c) CSDN

Fig. 6: Comparison of CDF Zipf's Law Guessing Curves and Our Best Upper/Lower Bounds



(a) Account Tenure (b) Password Requirements At Registration (c) Gender (self-reported)

Fig. 7: Analysis on Yahoo! Corpus

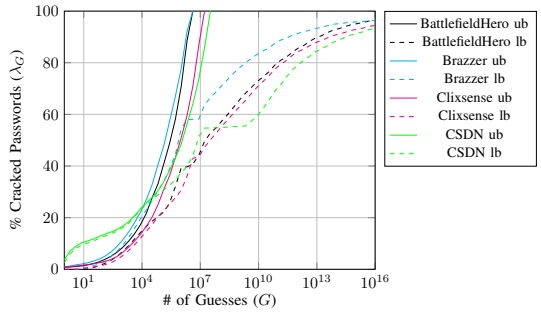


Fig. 8: Best Upper/Lower Bounds for Battlefield Heroes, Brazzers, Clixsense, and CSDN

We denote $\delta_{4,i} = \exp\left(\frac{-2(N-i)^2\epsilon_{2,i}^2}{N(i+1)^2}\right)$ in Theorem 8. we set $q = 1.002$, $i' = 4$, $\{\delta_{4,0}, \delta_{4,1}, \delta_{4,2}, \delta_{4,3}, \delta_{4,4}\} = \{0.00009, 0.000165, 0.00175, 0.00175, 0.0012\}$, and $\{\hat{x}_{\epsilon_{3,0}}, \hat{x}_{\epsilon_{3,1}}, \hat{x}_{\epsilon_{3,2}}, \hat{x}_{\epsilon_{3,3}}, \hat{x}_{\epsilon_{3,4}}\} = \{7.0/N, 11.0/N, 14.0/N, 16.3/N, 18.5/N\}$. Note that $\epsilon_{2,i} = \left(\frac{N(i+1)^2\ln(\delta_{4,i})}{-2(N-i)^2}\right)^{\frac{1}{2}}$ and $\epsilon_{3,i} = \frac{1}{q^{i+1}}\left(\frac{1-\hat{x}_{\epsilon_{3,i}}}{1-q\hat{x}_{\epsilon_{3,i}}}\right)^{N-i} - 1$ for any $0 \le i \le i'$. Both of the upper and lower bounds in Theorem 8 ($\texttt{LPUB}(S,G)$ and $\texttt{LPLB}(S,G)$) hold with probabilities at least $1 - \sum_{i=0}^{i'} \delta_{4,i} \ge 0.99$.

APPENDIX E
MISSING PROOFS

A. Missing Proofs of LP Bounds in Section III-D

Reminder of Lemma 1. *For any $i \ge 0$ and $0 \le \epsilon_{2,i} \le 1$, we have $\frac{(i+1)F_{i+1}^S}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} \le \sum_j h_j \times x_j \times \texttt{bpdf}(i,N,x_j) \le \frac{(i+1)F_{i+1}^S}{N-i} + \epsilon_{2,i}$ with probability at least $1 - 2\times\exp\left(\frac{-2(N-i)^2\epsilon_{2,i}^2}{N(i+1)^2}\right)$ where the probability is taken over the selection of our sample $S \leftarrow \mathcal{P}^N$.*

Proof of Lemma 1. We first prove that $\sum_j h_j \times x_j \times \texttt{bpdf}(i,N,x_j)$ can be bounded using $\frac{i+1}{N-i}\mathbb{E}(F_{i+1}^S)$ as below:

$$\begin{aligned}
&\sum_j h_j \times x_j \times \texttt{bpdf}(i,N,x_j)\\
=&\sum_j h_j \times x_j \times \frac{N!}{(N-i)!i!}x_j^i(1-x_j)^{N-i}\\
=&\frac{i+1}{N-i}\sum_j h_j \times (1-x_j) \times \texttt{bpdf}(i+1,N,x_j)\\
=&\frac{i+1}{N-i}\mathbb{E}(F_{i+1}^S) - \frac{i+1}{N-i}\sum_j h_j \times x_j \times \texttt{bpdf}(i+1,N,x_j)
\end{aligned}$$

Since $0 \le \frac{i+1}{N-i}\sum_j h_j \times x_j \times \texttt{bpdf}(i+1,N,x_j) \le \frac{i+1}{N-i}$, we have $\frac{i+1}{N-i}\mathbb{E}(F_{i+1}^S) - \frac{i+1}{N-i} \le \sum_j h_j \times x_j \times \texttt{bpdf}(i,N,x_j) \le \frac{i+1}{N-i}\mathbb{E}(F_{i+1}^S)$.

Next we will prove that when i is small, F_{i+1}^S is highly concentrated on $\mathbb{E}(F_{i+1}^S)$. We define $Y_1, ..., Y_N$ to be N independent password sample random variables, and consider $F_{i+1}^S = h(Y_1, ..., Y_N)$ to be a function that outputs the number of distinct passwords that appear exact $i+1$ times among all N samples in the sample set S. Then we have

$$\sup_{y_1,...,y_i,...,y_N,y_i'} |h(y_1,...,y_i,...,y_N) - h(y_1,...,y_i',...,y_N)| \leq 1.$$

Using Theorem 1, for any $t_1 \geq 0$ we have:

$$\Pr[h(Y_1,...,Y_N) \geq \mathbb{E}(F_{i+1}^S) - t_1] \geq 1 - \exp\left(-2t_1^2/N\right)$$
$$\Pr[h(Y_1,...,Y_N) \leq \mathbb{E}(F_{i+1}^S) + t_1] \geq 1 - \exp\left(-2t_1^2/N\right)$$

Denote $t_1 = \frac{N-i}{i+1}\epsilon_{2,i}$. Since $h(Y_1,...,Y_N) = F_{i+1}^S$ we have $\frac{(i+1)F_{i+1}^S}{N-i} - \epsilon_{2,i} - \frac{i+1}{N-i} \leq \sum_j h_j \times x_j \times \texttt{bpdf}(i,N,x_j) \leq \frac{(i+1)F_{i+1}^S}{N-i} + \epsilon_{2,i}$ with probability at least $1 - 2 \times \exp\left(\frac{-2(N-i)^2\epsilon_{2,i}^2}{N(i+1)^2}\right)$.

Theorem 10. *Given a probability distribution $\mathcal{P}$ which is consistent with a finite mesh $X_l = \{x_1, \ldots, x_l\}$ for any $G \geq 0$, any $i' \geq 0$, any $\epsilon_{\mathbf{2}} = \{\epsilon_{2,0}, \ldots, \epsilon_{2,i'}\} \in [0,1]^{i'+1}$, we have:*

$$\Pr\left[\lambda_G \geq \min_{1 \leq idx \leq l} \texttt{LP1}(X_l, F^S, idx, G, 1, i', \epsilon_{\mathbf{2}})\right] \geq 1 - \delta$$
$$\Pr\left[\lambda_G \leq \max_{1 \leq idx \leq l} |\texttt{LP1}(X_l, F^S, idx, G, -1, i', \epsilon_{\mathbf{2}})|\right] \geq 1 - \delta$$

where $\delta = 2 \times \sum_{0 \leq i \leq i'} \exp\left(\frac{-2(N-i)^2\epsilon_{2,i}^2}{N(i+1)^2}\right)$ and the randomness is taken over the sample $S \leftarrow \mathcal{P}^N$ of size N.

B. Other Missing Proofs

Reminder of Theorem 2. *For any guessing number $G \geq 0$ and any $0 \leq \epsilon \leq 1$ we have:*

$$\Pr[\lambda(S,G) \leq \lambda_G + \epsilon] \geq 1 - \exp\left(-2N\epsilon^2\right) \text{ , and}$$
$$\Pr[\lambda(S,G) \geq \lambda_G - \epsilon] \geq 1 - \exp\left(-2N\epsilon^2\right)$$

where the randomness is taken over the sample set $S \leftarrow \mathcal{P}^N$ of size N.

Proof of Theorem 2. Recall that the samples $s_1, s_2, ..., s_N$ in S are N independent random variables sampled from the real password distribution $\mathcal{P}$. For any two sample set $S = \{s_1, ..., s_i, ..., s_N\}$ and $S' = \{s_1, ..., s_i', ..., s_N\}$ that only differs on one sample s_i and s_i', we have $N \times |\lambda(S,G) - \lambda(S',G)| \leq 1$. Therefore, using Theorem 1, for any parameter $0 \leq \epsilon \leq 1$ we have:

$$\Pr[\lambda(S,G) \geq \lambda_G + \epsilon] \leq \exp\left(-2N\epsilon^2\right)$$
$$\Rightarrow \Pr[\lambda(S,G) \leq \lambda_G + \epsilon] \geq 1 - \exp\left(-2N\epsilon^2\right)$$
$$\Pr[\lambda(S,G) \leq \lambda_G - \epsilon] \leq \exp\left(-2N\epsilon^2\right)$$
$$\Rightarrow \Pr[\lambda(S,G) \geq \lambda_G - \epsilon] \geq 1 - \exp\left(-2N\epsilon^2\right)$$