On the Economics of Offline Password Cracking

Jeremiah Blocki
Purdue University

Ben Harsha
Purdue University

Samson Zhou
Purdue University

Abstract—**We develop an economic model of an offline password cracker which allows us to make quantitative predictions about the fraction of accounts that a rational password attacker would crack in the event of an authentication server breach. We apply our economic model to analyze recent massive password breaches at Yahoo!, Dropbox, LastPass and AshleyMadison. All four organizations were using key-stretching to protect user passwords. In fact, LastPass' use of PBKDF2-SHA256 with 10^5 hash iterations exceeds 2017 NIST minimum recommendation by an order of magnitude. Nevertheless, our analysis paints a bleak picture: the adopted key-stretching levels provide insufficient protection for user passwords. In particular, we present strong evidence that most user passwords follow a Zipf's law distribution, and characterize the behavior of a rational attacker when user passwords are selected from a Zipf's law distribution. We show that there is a finite threshold which depends on the Zipf's law parameters that characterizes the behavior of a rational attacker — if the value of a cracked password (normalized by the cost of computing the password hash function) exceeds this threshold then the adversary's optimal strategy is *always* to continue attacking until each user password has been cracked. In all cases (Yahoo!, Dropbox, LastPass and AshleyMadison) we find that the value of a cracked password almost certainly exceeds this threshold meaning that a rational attacker would crack all passwords that are selected from the Zipf's law distribution (i.e., most user passwords). This prediction holds even if we incorporate an aggressive model of diminishing returns for the attacker (e.g., the total value of 500 million cracked passwords is less than 100 times the total value of 5 million passwords). On a positive note our analysis demonstrates that memory hard functions (MHFs) such as SCRYPT or Argon2i can significantly reduce the damage of an offline attack. In particular, we find that because MHFs substantially increase guessing costs a rational attacker will give up well before he cracks most user passwords and this prediction holds even if the attacker does not encounter diminishing returns for additional cracked passwords. Based on our analysis we advocate that password hashing standards should be updated to require the use of memory hard functions for password hashing and disallow the use of non-memory hard functions such as BCRYPT or PBKDF2.**

1. Introduction

In the last few years breaches at organizations like Yahoo!, Dropbox, Lastpass, AshleyMadison, LinkedIn, eBay and Adult FriendFinder have exposed over a billion user passwords to offline attacks. Password hashing algorithms are a critical last line of defense against an offline attacker who has stolen password hash values from an authentication server. An attacker who has stolen a user's password hash value can attempt to crack each user's password offline by comparing the hashes of likely password guesses with the stolen hash value. Because the attacker can check each guess offline it is no longer possible to lockout the adversary after several incorrect guesses.

An offline attacker is limited only by the cost of computing the hash function. Ideally, the password hashing algorithm should be moderately expensive to compute so that it is prohibitively expensive for an offline attacker to crack most user passwords e.g., by checking millions, billions or even trillions of password guesses for each user. It is perhaps encouraging that AshleyMadison, Dropbox, LastPass and Yahoo! had adopted slow password hashing algorithms like BCRYPT and PBKDF2-SHA256 to discourage an offline attacker from cracking passwords. In the aftermath of these breaches, the claim that slow password hashing algorithms like BCRYPT [1] or PBKDF2 [2] are sufficient to protect most user passwords from offline attackers has been repeated frequently. For example, LastPass [3] claimed that "Cracking our algorithms [PBKDF2-SHA256] is extremely difficult, even for the strongest of computers." Security experts have made similar claims about BCRYPT e.g., after the Dropbox breach [4] a prominent security expert confidently stated that "all but the worst possible password choices are going to remain secure" because Dropbox had used the BCRYPT hashing algorithm.

Are these strong claims about the security of BCRYPT and PBKDF2 true? Despite all of their problems passwords remain prevalent and are likely to remain entrenched as the dominant form of authentication on the internet for years to come because they are easy to use and deploy, and users are already familiar with them [5], [6], [7]. It is therefore imperative to develop tools to quantify the damages of password breaches, and provide guidance to organizations on how to store passwords. In this work we seek to address the following question:

> Can we quantitatively predict how many user passwords a rational attacker will crack after a breach?

We introduce a game-theoretic model to answer this question and analyze recent data-breaches. Our analysis strongly challenges the claim that BCRYPT and PBKDF2-SHA256 provide adequate protection for user passwords. On the positive side our analysis indicates that more modern password hashing algorithms [8] (e.g., memory hard functions [9]) can provide meaningful protection against offline attackers.

1.1. Contributions

We first develop a new decision-theoretic framework to quantify the damage of an offline attack. Our model generalizes the stackelberg game-theoretic model of Blocki and Datta [10]. A rational password attacker is economically motivated and will quit guessing once his marginal guessing costs exceed his marginal reward. The attacker's marginal reward is given by the probability p_i that the next (ith) password guess is correct times the value of an *additional* cracked password to the adversary e.g., the additional revenue of selling that password on the black market or the expected amount of additional money that could be extorted from this user. Given the average value v of each cracked password for the adversary[1], the cost k of computing the password hash function and the probability distribution $p_1 > p_2 > \ldots$ over user selected passwords, our model allows us to predict exactly how many passwords a rational adversary will crack. Unlike the model of Blocki and Datta [10] we can use our framework to model a setting in which the attacker encounters diminishing returns as we would expect in most (black)markets i.e., the total value of 500 million cracked passwords may be significantly less than 100 times the total value of 5 million passwords.

Second, we present the strongest evidence to date that Zipf's law models the distribution of user selected passwords (with the possible exception of the tail of the distribution). These findings strongly support previous conclusions of Wang and Wang [11]. In particular, we show that Zipf's law closely fits the Yahoo! password frequency corpus. This dataset was collected by Bonneau [12] and later published by Blocki et al. [13]. In contrast to datasets from password breaches the Yahoo! dataset was collected by trusted parties, and is representative of active Yahoo! users (researchers have observed that hacked datasets contain many passwords that appear to be fake [14]). Our sample size, 70 million users, is also more than twice as large as the datasets Wang and Wang[11] used to support their argument that Zipf's law closely models password datasets.

Third, we show that there is a finite threshold $T(.)$ which characterizes the behavior of a rational value v-adversary whenever the distribution over passwords follows Zipf's law. In particular, if the first cracked password has value $v \geq T(.) \times k$ then the adversary's optimal strategy is always to continue guessing until he cracks the user's password. The threshold $T(y, r, a)$ is parameterized Zipf's law parameters y and r and a parameter a representing the rate of password value decay. We remark that, even if Zipf's law fails to model the tail of the password distribution, the threshold $T(y, r, a)$ still provides a useful characterization of the attacker's behavior. In particular, if $(1 - x)\%$ of passwords in a distribution follow Zip's law and the other $x\%$ follow some unknown (possibly uncrackable) distribution then our bounds imply that an attacker will compromise at least $(1 - x)\%$ of user passwords whenever $v \geq T(y, r, a) \times k$.

Fourth, we also derive model independent upper and lower bounds on the fraction of passwords that a rational adversary would crack. While these bounds are slightly weaker than the bounds we can derive using Zipf's law these bounds do not require any modeling assumptions e.g., it is impossible to determine for sure whether or not Zipf's law fits the tail of the password distribution. Interestingly, the lower bounds we derive suggest that state of the art password crackers [15] could still be improved substantially.

Fifth, we apply our framework to analyze recent large scale password breaches including LastPass, AshleyMadison, Dropbox and Yahoo! Our analysis strongly challenges the claim that BCRYPT and PBKDF2-SHA256 provide adequate protection for user passwords. In fact, if the password distribution follows Zipf's law then our analysis indicates that a rational attacker will almost certainly crack 100% of user passwords e.g., unless the value of Dropbox/LastPass/AshleyMadison/Yahoo! passwords is *significantly* less valuable than black market projections [16].

Finally, we derive *model independent* upper and lower bounds on the % of passwords cracked by a rational adversary. These bounds do not rely on the assumption that Zipf's law models the tail of the password distribution[2]. Nevertheless, our predictions are still quite dire e.g., a rational adversary will crack 51% of Yahoo! passwords *at minimum*. Our analysis indicates that, to achieve sufficient levels of protection with BCRYPT or PBKDF2, it would be *necessary* to run these algorithms for well over a second on modern CPU which would constitute an unacceptable authentication delay in many contexts [17]. On a more positive note our analysis suggests that the use of more modern password hashing techniques like memory hard functions *can* provide strong protection against a rational password attacker *without* introducing inordinate delays for users during authentication. In particular, our analysis suggests that it could be possible to reduce the % of cracked passwords below 22.2% without increasing authentication delays to a full second.

1.2. Discussion

In light of our analysis we contend that that there is a clear need to update standards for password storage to provide developers with clear guidance about the importance of using memory hard functions such as SCRYPT [9] or Argon2id [18]. In a recent recent user study Naiakshina et al. [19] asked developers to select a password hash function for a new social networking platform. None of the

[1] More precisely, if there are N users in the dataset and the total value of all N cracked passwords is V then $v = V/N$. When there are diminishing returns for additional cracked passwords the parameter v may be significantly lower than the value of the first cracked password.

[2] Wang and Wang [11] observed that the tails of empirical password datasets are not inconsistent with a Zipf's law distribution. However, we cannot be entirely confident that Zipf's law models the tail of the distribution since, by definition, we do not have many samples for passwords in the tail of the distribution.

developers in this study selected a memory hard function[3] and the strongest password hashing algorithms selected were PBKDF2 with 20,000 hash iterations and BCRYPT with 1,024 iterations. The selection of PBKDF2 with 20,000 hash iterations would be deemed acceptable under 2017 NIST standards [20] — PBKDF2 with at least $10,000$ iterations is presented an acceptable selection for password hashing[4]. In this sense, LastPass' use of PBKDF2-SHA256 with $100,000$ iterations greatly exceeds current NIST standards. Nevertheless, our analysis suggests that even PBKDF2-SHA256 with $100,000$ hash iterations is insufficient to protect a majority a user passwords while memory hard functions such as SCRYPT [9] or Argon2id [18] would provide meaningful protection. In addition to memory hard functions we also advocate for the use of secure distributed password hashing protocols [22], [23], [24] whenever feasible so that an attacker cannot mount an offline attack without breaching *multiple* authentication servers.

2. Economic Model

2.1. Preliminaries

Given a dataset D of N user passwords we use f_i to denote the frequency of the i'th most common password in the dataset and we use pwd_i to denote the i'th most common password in the dataset. We use $p_1, p_2, \ldots$ to denote the actual distribution over passwords $pwd_1, pwd_2, \ldots$. That is p_i is the probability that a random user selects password pwd_i. We use $\hat{p}_i = f_i/N$ to denote an empirical estimate of p_i given a dataset D which was sampled from the real password distribution. We also use $\lambda_i = \sum_{j=1}^{i} p_j$ to denote the cumulative probability of the i most likely passwords. Equivalently, λ_i denotes the probability that an adversary cracks the user's password within the first i guesses.

We say that the probability distribution $p_1 \geq p_2 \ldots$ follows Zipf's law if $p_i = \frac{z}{i^s}$ for some constants s and z. We say that a probability distribution follows a CDF-Zipf distribution if $\lambda_i = y i^r$ for some constants r and y.

Offline Attack. To authenticate users password authentication servers traditionally store salted password hashes. In more detail to authenticate user u the authentication stores a record like the following: $(u, s_u, H(pwd_u|s_u))$. Here, u is the the username and pwd_u is the user's password, s_u is a random string called the salt value used to protect against rainbow table attacks and H is a cryptographic hash function. An adversary who breaches the authentication server will be able to obtain the hash value along with the secret salt value. This adversary can now attempt as many guesses as he desires offline by computing the hashes of likely passwords guesses $H(g_1, s_u), H(g_2, s_u), \ldots$ and comparing these values with the stolen password hash. The attacker is only limited by the resources that he is willing to invest trying to crack the user's password.

Key Questions and Parameters. We aim to address the following questions: How many guesses will our rational adversary attempt? What fraction of the user passwords will an adversary manage to break? The answer to these questions will depend on several factors. How valuable is a cracked password to the adversary? How much does it cost to compute H each time we validate a new password guess? What does the distribution over user passwords look like?

We use v to denote the value of a cracked password to the adversary measured in units of C_H, where H is an underlyng cryptographic hash function like SHA256. We can estimate $v^{\$}$ by looking at black market prices for cracked passwords. For example, Fossi et al. [16] found that the market price for hacked passwords tends to lie in the range $[\$4, \$30]$. A more recent analysis of Yahoo! passwords found that they sell for between \$0.7 and \$1.2 [25] — the drop in price may be due to an increased supply of Yahoo! passwords. Herley and Florencio found that dishonest behavior can significantly inhibit trade on black markets [26]. Thus, these prices may underestimate the true value of a cracked password.

Password hash functions are often constructed from an underlying cryptographic hash function H. For example, PBKDF2-SHA256 simply iterates the SHA256 hash function multiple times. We use k to denote the cost of a computing the final password hash function — once again measured in units of C_H. We use $v^{\$} = v \times C_H$ (resp. $k^{\$} = k \times C_H$) to denote the value (resp. cost) in USD given an estimate of C_H.

2.2. Rational Adversary

We model a rational adversary who has obtained the salted password hash of a user's password. Our model generalizes the stackelberg game-theoretic framework of Blocki and Datta [10] by introducing a parameter $0 \leq a \leq 1$ which models diminishing returns. We assume that adversary knows the password distribution $p_1, p_2, \ldots$ as well as the corresponding passwords $pwd_1, pwd_2, \ldots$. However, the adversary does not know which password the user selected.

Attacker Game. We model password cracking using a single-shot game. In the game we sample a random password pwd from the password distribution $\Pr[pwd_i] = p_i$. The adversary picks a threshold $t \geq 0$. The threshold t specifies an ordered list $L(t) = pwd_1, \ldots, pwd_t$ of the t most likely passwords. If the real password is contained in the list of adversary guesses, $pwd \in L(t)$, then the adversary receives a payment of v and we charge the adversary $j \cdot k$, where j is the index of the correct password guess $pwd = pwd_j$. If the real password is not contained in the list $pwd_1, \ldots, pwd_t$ of adversary guesses then the adversary receives no payment ($v = 0$) and the adversary is charged $t \cdot k$. Notice that $t = 0$ corresponds to the strategy in which the adversary gives up without guessing, and $t = \infty$

[3] On a positive note the authors did find that priming developers about the importance password security resulted in the selection of stronger password hashing algorithms.

[4] An upgrade from $1,000$ iterations as the minimal acceptable number of hash iterations for PBKDF2 in an older 2010 NIST standard [21].

corresponds to the strategy in which the adversary never quits. Observe that $\lambda_t = \sum_{j=1}^{t} p_j$ denotes the fraction of user passwords that are cracked by a threshold t adversary.

About the Attacker. In our analysis we consider an attacker that is

(1) **Informed:** The attacker knows the password distribution $p_1, p_2 \ldots$ and the associated passwords $pwd_1, pwd_2 \ldots$. However, the attacker does not know which password a particular user u selected.
(2) **Untargeted:** The attacker does not have personal knowledge about the user that can be exploited to improve the guessing attack.
(3) **Rational:** The attacker is economically motivated, and will stop attacking the user once marginal guessing costs exceed the marginal guessing rewards.

Discussion. Our attacker model captures the most common types of password attacks. It is generally reasonable to assume that the attacker knows the password distribution — possibly excluding of the tail of the distribution. In particular, previous password breaches provide plenty of training data for the attacker and it is reasonable to assume that password cracking models will continue to improve as attackers obtain more and more training data from future password breaches. We focus on an untargeted attacker in our analysis. However, we stress that our model may also be useful when considering a targeted attacker with background knowledge of the user (e.g., name, birthdate, hobbies etc...). In particular, let p_i denote the probability that a targeted adversary's i'th guess is correct. Wang et al. observed that a targeted distribution over user passwords $p_1, p_2 \ldots$ still seems to follow Zipf's law [27].

Rational Attacker Behavior. If the adversary chooses a threshold t then his expected guessing costs are

$$C(t) = t\left(1 - \sum_{j=1}^{t} p_j\right) k + k \sum_{j=1}^{t} j \cdot p_j \ .$$

Similarly, his expected reward is

$$R(t) = v \left(\sum_{j=1}^{t} p_j\right)^{a}$$

where the parameter $0 \leq a \leq 1$ allows us to model diminishing returns for the attacker as he obtains additional cracked passwords. For example, let $t_{1\%}$ (resp. $t_{2\%}$) be given such that $p_1 + \ldots + p_{t_{1\%}} = 0.01$ ($p_1 + \ldots + p_{t_{2\%}} = 0.02$) then for $a < 1$ we have $R(t_{2\%}) = 2^a R(t_{1\%}) < 2 \times R(t_{1\%})$ even though an adversary cracks twice as many passwords by increasing his threshold from $t_{1\%}$ to $t_{2\%}$.

Diminishing Returns: We note that the original model of Blocki and Datta [10] is a special case of our model when $a = 1$ (no diminishing returns). There are a number of reasons why an attacker may encounter diminishing returns ($a < 1$) for additional cracked passwords. First, if the attacker plans to sell the passwords on the black market then basic economics suggests that increasing the supply of cracked passwords is likely to drive down prices. In the case of a large breach like Yahoo! (500 million passwords) it is conceivable the number of available passwords on the black market might quickly increase by two orders of magnitude. Second, the more user accounts that are hacked/actively exploited the more likely it is that the original breach will be detected. If the breach is detected then an organization can ask (or require) users to change their passwords or require two-factor authentication, which will reduce the value of each cracked password[5].

Interpreting model parameter v: We note that we have $v = R(\infty)$, where $R(\infty) \times N$ denotes the total value of a completely cracked password dataset of size N. Thus, the parameter v denotes the average value of a cracked password given that all password have been cracked. We can estimate this parameter v based on black market sales data. For example, suppose that we know that $R(t_{1\%}) = \$4 \times 1\%$ e.g., from equilibrium black market prices when only 1% of cracked passwords are on the market. In this case we can extrapolate

$$v = R(\infty) = R(t_{100\%}) = 100^a R(t_{1\%}) = \frac{\$4}{100^{1-a}} \ . \quad (1)$$

Rational Attacker Behavior: Formally, the rational adversary will select the threshold t^* maximizing his overall utility

$$t^* = \arg\max_t \left(R(t) - C(t)\right) \ .$$

Intuitively, a rational adversary should stop guessing if the marginal cost of one more password guess exceeds the marginal benefit of that guess. Thus, we will have $MC(t^*) = C(t^*) - C(t^*-1) \approx MR(t^*) = R(t^*) - R(t^*-1)$. The marginal cost of increasing the threshold from $t-1$ to t is

$$MC(t) = C(t) - C(t-1) = k\left(1 - \sum_{j=1}^{t-1} p_j\right) \ . \quad (2)$$

Intuitively, the attacker pays an extra cost k to hash pwd_t if and only if the first $t-1$ guesses are incorrect. Similarly, the attacker's marginal revenue is $MR(t) = R(t) - R(t-1)$ when $a = 1$ we have $MR(t) = v \times p_t$ otherwise

$$MR(t) = v\left(\left(\sum_{j=1}^{t} p_j\right)^{a} - \left(\sum_{j=1}^{t-1} p_j\right)^{a}\right) \times p_t \ . \quad (3)$$

Note that λ_{t^*} denotes the expected fraction of passwords compromised by an rational attacker. Given a specific assumption about the password distribution (e.g., Zipf's law) we can derive bounds on λ_{t^*}.

Competition. We do not attempt to directly model the behavior of an adversary who faces competition from other password crackers. Many breaches (e.g., Yahoo!, LinkedIn,

[5] However, the cracked passwords arguably still have significant value after the breach is detected for two reasons. First, many users will not update their passwords unless they are required to do so. Second, many of the users that do update their passwords may do so in a predictable way [28]. Third, many users will have the same password for other accounts.

Dropbox) remained undetected for several years. In these cases it may be reasonable to assume that the password cracker faced no competition. However, competition certainly could occur in the event that the breach is public (e.g., Ashley Madison). In an extremely competitive setting (e.g., password for a cryptocurrency wallet) only the first attacker to crack the password will be rewarded[6]. Such competition would decrease the expected reward for each cracked password and could potential reduce the total % of passwords cracked by *each individual* attacker.

However, from the defender's point of view the goal is to minimize the % of passwords that are cracked by *any* attacker. Thus, we can argue that competition will have a minimal impact on the total % of cracked passwords. In particular, even in an extremely competitive setting where only the first attacker to find the password is rewarded we still have

$$\mathbf{CompCrack}(v,a) \geq \min_{0\leq p\leq 1} \max\{\mathbf{Cracked}(pv,a), 1-p\} \ .$$

Here $\mathbf{CompCrack}(v)$ (resp. $\mathbf{Cracked}(v)$) denotes the % of passwords that are cracked by some attacker when the value of a password is v and attackers face competition (resp. do not face competition). This follows because the expected reward for attacker when faced with competition is at least $R_{comp}(t) \geq p_{first} \times R(t)$ where p_{first} is the probability that no competing attacker managed to crack the password already. If p_{first} is small then the marginal rewards will also be small so the attacker may quit earlier, but in this case it is likely that another attacker has already compromised the account $(1-p_{first})$.

Defender Actions. The value λ_{t^*} will depend on k, v as well as the underlying password distribution $p_1 \geq p_2 \geq \ldots$. The goal of key-stretching is to increase k so that we can reduce λ_{t^*}, the fraction of compromised accounts, in the event of an authentication server breach. However, the defender is constrained by server workload and by authentication times. In particular, the number of sequential hash iterations (τ) is bounded by usability constraints as users may be unhappy if they need to wait a long time to authenticate e.g., it would at least a second to compute PBKDF2-SHA256 with $\tau = 10^7$ hash iterations on a modern CPU [29]. Similarly, the total workload k is similarly bounded by workload constraints e.g., the authentication server must be able to handle all of the authentication requests even during traffic peaks. If the value v is sufficiently large (in proportion to the cost k of a password guess) then a rational attacker will crack every password $\lambda_{t^*} = 1$. In this case we say that all of the key-stretching effort was useless against a value v rational adversary.

Password hashing algorithms like BCRYPT, PBKDF2 and SCRYPT have parameters that control the running time (number of hash iterations) τ and total cost k of computing the password hash function. Thus, the cost k of computing PBKDF2 or BCRYPT is $k = \tau \times C_H$, where C_H denotes the cost of computing the underlying hash function (e.g., SHA256 or Blowfish). We will treat C_H as a unit of measurement when we report the cost k and write $k = \tau$ for the BCRYPT and PBKDF2 functions. Given an estimate of C_H in USD we will use $k^{\$} = k \times C_H$ to denote the cost of computing the password hash function in USD.

Intuitively, a memory hard function is a function whose computation requires large amounts of memory. One of the key advantages of a memory hard function is that cost k potentially scales with τ^2 instead of τ making it possible to increase costs without introducing intolerable authentication delays. An ideal memory hard function runs in time τ and requires τ blocks of memory to compute. Thus, the Area x Time (AT) complexity of computing the Memory Hard Function scales with τ^2 because the adversary must allocate τ blocks of memory for τ units of time. In particular, we use $k = \tau \times C_H + \tau^2 \times C_{mem}$ to model the approximate cost of computing a memory hard function which iteratively makes τ calls to the underlying hash function H and requires τ blocks of memory. By contrast, the AT complexity of BCRYPT and PBKDF2 is just $k = \tau$ since these functions can be computed with a single block of memory. Here, C_{mem} is a constant representing the core memory-area ratio. That is the area of one block of memory on chip divided by the the area of a core evaluating H on chip. In this paper we use the estimate $C_{mem} \approx 1/3000$ as in [30], [31] though we stress that our analysis could be easily repeated with different parameter choices.

Model Limitations. To keep exposition simple we do not attempt to incorporate any model of equilibrium prices for cracked passwords on the black market and instead assume that the value of a cracked password $v^{\$}$ is static for all users. A targeted adversary may have higher valuations for specific user passwords e.g., celebrities, politicians. Similarly, an attacker who floods a black market with cracked passwords may drive equilibrium prices down. Our primary findings would not be altered in any significant way by including such a model unless equilibrium prices drop by 1–2 orders of magnitude [16]. We also remark that our intention is to model an untargeted economically motivated attacker and not a nation state focused on cracking the passwords of a particular person of interest. However, it may still be reasonable to believe that a nation state attacker will be largely be constrained by economic considerations (e.g., expected value of additional intelligence gained by cracking the password versus expected cost to crack password).

3. Yahoo! Passwords follow Zipf's Law

Zipf's law states that the frequency of an element in a distribution is related to its rank in the distribution. There are two variants of Zipf's law for passwords: PDF-Zipf and CDF-Zipf. In the CDF-Zipf model we have $\lambda_t = \sum_{j=1}^{t} p_i = y \cdot t^r$, where the constants y and r are the CDF-Zipf parameters. In the PDF-Zipf model we have $f_i = \frac{C}{i^s}$, where s and C are the PDF-Zipf parameters. Normalizing by N the number of users we have $p_i = \frac{z}{i^s}$, where $z = \frac{C}{N}$.

[6] However, we remark that in many instances attackers may unknowingly "share" the benefit of a cracked account. For example, an attacker who cracks a password may not actually change the password since such an action would alert the legitimate user of the breach.

Wang et al. [32] previously found that password frequencies tend to follow PDF-Zipf's law if the tail of the password distribution (e.g., passwords with frequency $f_i < 5$) is dropped. Wang and Wang [11] subsequently found that CDF-Zipf's model is superior in that the CDF-Zipf fits were more stable than PDF-Zipf fits and that the CDF-Zipf fit performed better under Kolmogorov-Smirnov (KS) tests. Furthermore, the CDF-Zipf model can fit the entire password distribution (e.g., without excluding passwords with frequency $f_i < 5$). These claims were based on analysis of several smaller password datasets ($N \leq 32.6$ million users) which were released by hackers.

In 2016 Yahoo! allowed the release of a differentially private list of password frequencies for users of their services [13]. We refer an interested reader to [12], [13] for additional details about how the Yahoo! data was collected and how it was perturbed to preserve differential privacy. The Yahoo! dataset is superior to other datasets in that it offers the largest sample size $N = 70$ million and the dataset was collected and released by trusted parties. We show that the Yahoo! dataset is also well modeled by CDF-Zipf's law. Our analysis comprises the strongest evidence to date of Wang and Wang's premise [11] that password distributions follow CDF-Zipf's law due to the advantages of the Yahoo! dataset. We focus on the CDF-Zipf's law model in this section since it can fit the entire password distribution [11]. We also verified that the Yahoo! dataset is also well modeled by PDF-Zipf's law if we drop passwords with frequency $f_i < 5$ like Wang et al. [32], but we omit this analysis from the submission due to lack of space.

The rest of this section is structured as follows: First, in section 3.1 we discuss the advantages of using the Yahoo! dataset over leaked datasets like RockYou. In 3.2 we show that the noise that was added to preserve differential privacy will have a negligibly small impact on CDF-Zipf fittings. In section 3.3 we use subsampling to show that the CDF-Zipf fittings for Yahoo! converge to a stable solution. Finally, in section 3.4 we present the CDF-Zipf fitting for the entire Yahoo! dataset.

3.1. On Ecological Validity

The Yahoo! frequency corpus offers many advantages over breached password datasets such as RockYou or Tianya.

- The Yahoo! password frequency corpus is based on 70 million Yahoo! passwords — more than twice as large as any of the breached datasets analyzed by Wang and Wang [11].
- The records were collected in a trusted fashion. No infiltration, hacking, tricks, or general foul play was used to obtain any of this data. There was no ulterior motive behind collecting these passwords other than to provide valuable data in a way that can be used for scientific research. By contrast, it is possible that hackers strategically omit (or inject) password data before they release a breached dataset like RockYou or Tianya! Why should we trust rogue hackers to provide researchers with representative password data?

List Version	y	σ_y
RockYou Standard	0.0288	
RockYou Diff. Private	0.0302	$1.348 * 10^{-6}$
	r	σ_r
RockYou Standard	0.2108	
RockYou Diff. Private	0.2077	$2.94 * 10^{-6}$
	R^2	σ_{R^2}
RockYou Standard	0.9687	
RockYou Diff. Private	0.9681	$6.50 * 10^{-7}$

TABLE 1: Impact of Differential Privacy on CDF Fit

- Breached password datasets often contain many passwords/ accounts that look suspiciously fake. In 2016 Yang et al [14] suggested that such passwords can be removed with DBSCAN [33]. Cleansing operations ended up removing a reasonable portion of the dataset (e.g., 5 million passwords were removed from RockYou's data). With the Yahoo! data such cleansing is not needed, as it was collected in a manner that ensured collected passwords were in use. Previous work that has been done on Zipf distributions in breached password datasets [11] did not perform any sort of sanitizing step on the data. It is unclear how such operations would affect the Zipf law fit.
- The information is released in a responsible way that preserves users' privacy. The differential privacy mechanism means that even with the released data it is not possible to determine any new information about Yahoo's users that an adversary would not be able to obtain anyways.
- Data from the Yahoo! password frequency corpus ultimately is derived from the passwords of active Yahoo! users who were logging in during the course of the study as opposed to passwords from throwaway accounts that have been long forgotten.

3.2. On the Impact of Differential Privacy on CDF-Zipf Fits

The published Yahoo! password frequency lists were perturbed to ensure differential privacy. Before attempting to fit this dataset using Zipf's law we seek to answer the following question: Does this noise, however small, affect our CDF-Zipf fitting process in any significant way? We claim that the answer is no, and we offer strong empirical evidence in support of this claim. In particular, we took the RockYou dataset ($N \approx 32.6$ million users) and generated 30 different perturbed versions of the frequency list by running the (ϵ, δ)-differentially private algorithm of Blocki et al. [13]. We set $\epsilon = 0.25$, the same value that was used to collect the Yahoo! dataset that we analyze. For each of these perturbed frequency lists we compute a CDF-Zipf law fit using linear least squares regression. To apply Linear Least Squares regression we apply logarithms to the

Sample Size (Millions)	y	r	R^2
15	0.00949	0.2843	0.9542
30	0.01321	0.2544	0.9531
45	0.01592	0.2384	0.9529
60	0.01810	0.2277	0.9530
Full	0.02112	0.2166	0.9544

TABLE 2: Yahoo! CDF-Zipf with Sub-sampling

CDF-Zipf equation $\lambda_t = y \cdot t^r$ to obtain a linear equation $\log \lambda_t = \log y + r \log t$.

Our results, shown in Table 1, strongly suggest that the differential privacy mechanism does not impact the parameters y and r in a CDF-Zipf fitting in any significant way. In particular, the parameters y and r we obtain from fitting the original data with a CDF-Zipf model are virtually indistinguishable from the parameters we obtain by fitting on one of the perturbed datasets. Similarly, differential privacy does not affect the R^2 value of the CDF-Zipf fit. Here, R^2 measures how well the linear regression models the data (R^2 values closer to 1 indicate better fittings). Thus, one can compute CDF-Zipf's law parameters for the Yahoo! data collected by [13] and [12] without worrying about the impact of the (ϵ, δ)-differentially private algorithm used to perturb this dataset. We also verified that the noise added to the Yahoo! dataset will also have a negligible affect on the parameters s and z in a PDF-Zipf fitting.

3.3. Testing Stability of CDF-Zipf Fit via Subsampling

There are two primary ways to find a CDF-Zipf fit: Golden Section Search (GSS) and Linear Least Squares (LLS). Wang et al. [11] previously found that CDF-Zipf fits stabilize more quickly with GSS than with LLS. This was particularly important because the largest dataset they tested had size $\approx 3 \times 10^7$. In this section we test the stability of LLS by subsampling from the much larger Yahoo! dataset. In particular, we subsample (without replacement) datasets of size 15 million, 30 million, 45 million and 60 million and use LLS to compute the CDF-Zipf parameters y and r for each subsampled dataset. Our results are shown in table 2 graphically in Figure 1. While the CDF-Zipf fit returned by LLS does take longer to stabilize our results indicate that it does eventually stabilize at larger (sub)sample sizes (e.g., the Yahoo! dataset).

We also found that the PDF-Zipf parameters s and z stabilize before $N = 7 \times 10^7$ samples.

3.4. Fitting the Yahoo! data set with CDF-Zipf

We used both LLS regression and GSS to obtain separate CDF-Zipf fittings for the Yahoo! dataset. The results, shown in table 3 and graphically in Figure 6 showed that both methods produce high quality fittings. In addition to the parameters y and r we report R^2 values and Kolmogorov-Smirnov (KS) distance. The KS test can be thought of as the largest distance between the observed discrete distribution $F_n(x)$ and the proposed theoretical distribution $F(x)$. Formally,

$$D_{KS} = sup\,|F_n(x) - F(x)|$$

Intuitively, smaller D_{KS} values (resp. larger R^2 values) indicates better fits.

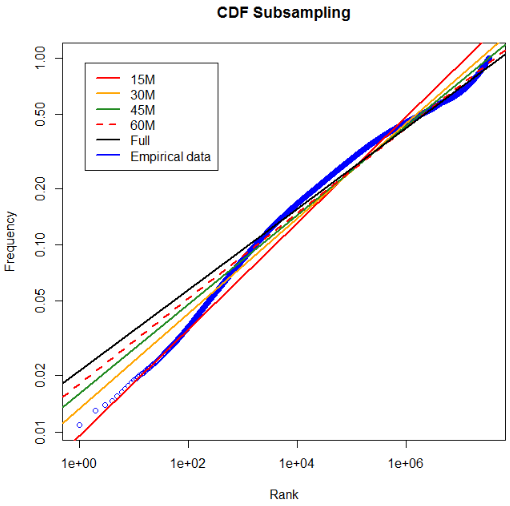


Fig. 1: Yahoo! CDF-Zipf Subsampling

Method	y	r	R^2	KS
LLS	0.0211	0.2166	0.9544	0.0094328
GSS	0.03315	0.1811	0.9498	0.022282

TABLE 3: Yahoo! CDF-Zipf Test Results

3.4.1. Discussion. Both LLS and GSS produce high quality CDF-Zipf fittings (e.g., $R^2 = 0.9544$) for the Yahoo! dataset. LLS regression outperforms the golden section search under both R^2 and Kolmogorov-Smirnov (KS) tests. Wang and Wang [11] had previously adopted golden section search because the results stabilized quickly. While this was most likely the right choice for smaller password datasets like RockYou, our analysis in the previous section suggest that LLS eventually produces stable solutions when the sample size is large (e.g., $N \geq 60$ million samples) as it is in the Yahoo! dataset. Thus, in the remainder of the paper we use the CDF-Zipf parameters $y = 0.0211$ and $= 0.2166$ from LLS regression. We stress that the decision to use the CDF-Zipf parameters from LLS instead of the parameters returned by GSS does not affect our findings in any significant way.

We remark that LLS is also more efficient computationally. While we were able to run GSS to find a CDF-Zipf fit for the Yahoo! dataset ($N \approx 7 \times 10^7$), running GSS on a dataset of $N = 1$ billion passwords (e.g., the size of the most recent Yahoo! breach [34]) would be difficult if not intractable. By contrast, LLS could still be used to find a CDF-Zipf fitting and our analysis suggests that the fit would be superior.

Dataset	y	r	$T(y,r,1)$	$T(y,r,0.8)$
RockYou	0.0374	0.1872	1.70×10^7	2.04×10^7
000webhost	0.0059	0.2816	3.67×10^7	4.27×10^7
Battlefield	0.0103	0.2949	2.37×10^6	2.77×10^6
Tianya	0.0622	0.1555	2.28×10^7	2.76×10^7
Dodonew	0.0194	0.2119	4.92×10^7	5.87×10^7
CSDN	0.0588	0.1486	7.63×10^7	9.24×10^7
Mail.ru	0.0252	0.2182	8.75×10^6	1.04×10^7
Gmail	0.0210	0.2257	1.14×10^7	1.36×10^7
Flirtlife.de	0.0346	0.2916	4.44×10^4	5.19×10^4
Yahoo!	0.0211	0.2166	2.25×10^7	2.69×10^7

TABLE 4: CDF-Zipf threshold $T(y,r,a) < \nu/k$ at which adversary cracks 100% of passwords for $a \in \{1, 0.8\}$.

4. Analysis of Rational Adversary Model for Zipf's Law

In this section, we show that there is a finite threshold $T(y,r,a)$ which characterizes the behavior of a rational offline adversary when user passwords follow CDF-Zipf's law with parameters y and r i.e., $\lambda_i = yi^r$. In particular, Theorem 1 gives a precise formula for computing this threshold $T(y,r,a)$[7]. If $\nu/k \geq T(y,r,a)$ then a rational value ν adversary will proceed to crack all user passwords as marginal guessing rewards will *always* exceed marginal guessing costs for a rational attacker. In Table 4 we use this formula to explicitly compute $T(y,r,a)$ for the Yahoo! dataset as well as for nine other password datasets analyzed by Wang and Wang [11].

We note that we choose to focus on CDF-Zipf's law in this section as it is believed to be better than PDF-Zipf models. However, we stress that similar bounds can be derived using PDF-Zipf's law though we omit these results from the submission for lack of space.

Theorem 1. Let k denote the cost of attempting a password guess. If

$$\frac{\nu}{k} \geq T(y,r,a) = \max_{t \leq Z}\left(\frac{1 - y(t-1)^r}{y^a(ra)t^{ra-1}}\right)$$

where

$$Z = \left\lceil \left(\frac{1}{y}\right)^{1/r} \right\rceil + 1$$

then a value ν rational attacker will crack 100% of passwords chosen from a Zipf's law distribution with parameters y and s.

Proof : Suppose a password frequency distribution follows Zipf's Law, for some parameters $0 < r < 1$ and y, so that $\lambda_n = yn^r$. Since the marginal revenue is $MR(n) = \nu(\lambda_t^a - \lambda_{t-1}^a)$ and the marginal cost is $MC(n) = k\left(1 - \sum_{n=1}^{t} p_n\right)$, a rational adversary can be assumed to continue attacking as long as $MR(n) \geq MC(n)$. Therefore, the attacker will not quit as long as

$$\nu(\lambda_t^a - \lambda_{t-1}^a) \geq k\left(1 - \sum_{n=1}^{t} p_n\right)$$
$$\nu(y^a t^{ra} - y^a(t-1)^{ra}) \geq k(1 - y(t-1)^r)$$

In particular, the attacker will not quit as long as

$$\frac{\nu}{k} \geq \frac{1 - y(t-1)^r}{y^a t^{ra} - y^a(t-1)^{ra}}\ .$$

Notably, if $\frac{\nu}{k} \geq \max_t\left(\frac{1-y(t-1)^r}{y^a t^{ra} - y^a(t-1)^{ra}}\right)$ for all t, then a rational adversary will eventually crack *all* passwords. Since $y^a t^{ra} - y^a(t-1)^{ra} = \int_{t-1}^{t} y^a(ra)x^{ar-1}\,dx$ we have $y^a(ra)(t-1)^{ar-1} \leq y^a t^{ra} - y^a(t-1)^{ra} \leq y^a(ra)t^{ra-1}$ and

$$\max_t\left(\frac{1 - y(t-1)^r}{y^a t^{ra} - y^a(t-1)^{ra}}\right) \geq \max_t\left(\frac{1 - y(t-1)^r}{y^a(ra)t^{ra-1}}\right)\ .$$

Let $f(t) = \left(\frac{1-y(t-1)^r}{y^a(ra)t^{ra-1}}\right)$. We note that if, $\forall t \geq Z, f'(t) \leq 0$ then for any value of t exceeding Z it will always be true that $\frac{\nu}{k} \geq 0$, and thus an adversary will be expected to crack all passwords. Then it follows that

$$f'(t) = \frac{(t-1)^{r-1}t^{1-ar}y^{1-a}}{a} + \frac{(1-ar)t^{-ar}y^a(1-(t-1)^r y)}{ar},$$

then $f'(t) \leq 0$ if and only

$$\frac{(1-ar)y^{-a}(1-(t-1)^r y)}{art^{ar}} \leq \frac{y^{1-a}t(t-1)^{r-1}}{at^{ar}}$$
$$(1-ar)(1-(t-1)^r y) \leq yt(t-1)^{r-1}r$$
$$(1-ar) \leq y(t-1)^{r-1}((t-1)(1-ar)+tr)$$

Since $(t-1)(1-ar) \leq (t-1)(1-ar)+tr$, then the last expression certainly holds true if $(1-ar) \leq y(t-1)^{r-1}(t-1)(1-ar)$ or equivalently, $\frac{1}{y} \leq (t-1)^r$. Thus, by setting $Z = \left\lceil 1 + \left(\frac{1}{y}\right)^{1/r} \right\rceil$, it follows that $f'(t) \leq 0$ for all $t > Z$. □

5. Analysis of Previous Password Breaches

In this section, we apply our economic model to analyze the consequences of recent password breaches and the impact of defenses that could have been adopted.

5.1. Breaches

We focus on the following breaches in our analysis:

5.1.1. Yahoo!. Attackers stole password hashes for 500 million Yahoo! users in 2014, though the breach was unknown to the general public until 2016 [35]. While Yahoo!

[7] We remark that when $a = 1$ it is possible to derive a closed form expressing for the threshold $T(y,r,a)$.

used BCRYPT to hash passwords [8], they have not publicly specified the number of hash iterations τ that they used. However, we do have empirical password frequency data from 70 million Yahoo! users which allowed us to derived CDF-Zipf parameters $y = 0.0211$ and $r = 0.2166$ for Yahoo! passwords. Thus, we can predict the % of cracked passwords for different values of τ that Yahoo! might have chosen.

5.1.2. Dropbox. Attackers stole password hashes for ≈ 68.7 million Dropbox users though the breach was unknown to the general public until 2016 [4]. Dropbox used BCRYPT at level 8 (i.e., $\tau = 2^8 = 256$ hash iterations) to hash passwords. We don't have empirical password data from Dropbox users from which we can derive Zipf's law parameters y and r. However, we have Zipf's law parameters for many other datasets such as RockYou, Tianya, CSDN and Yahoo! allowing us to predict how many passwords a value v adversary would crack if, say, Dropbox passwords and RockYou passwords have similar strength. Arguably, Dropbox passwords could be quite valuable as they are often used to protect sensitive data.

5.1.3. AshleyMadison. Attackers stole nearly 40 million AshleyMadison password hashes [36] in 2015 and released the stolen data publicly a month later. AshleyMadison primarily used BCRYPT at level 12 ($\tau = 2^{12} = 4,096$ hash iterations) to hash passwords [37]. However, CynoSure Prime noticed that some passwords were effectively protected with MD5 instead of BCRYPT due to an implementation error. CynoSure Prime managed to crack approximately 11 million of these MD5 hashes in just 10 days [36], though it has been claimed that most of the passwords protected by BCRYPT are uncrackable [37]. Similar to Dropbox, we do not have Zipf's law parameters for AshleyMadison users. However, it is plausible to believe that these parameters are comparable to the parameters derived from other datasets such as Yahoo! or RockYou!

5.1.4. LastPass. LastPass was using PBKDF2-SHA256 with $\tau = 10^5$ rounds of iteration when they were breached in 2015. Similar to AshleyMadison and Dropbox breaches we don't currently have Zipf's law parameters for LastPass passwords though we can still predict how many passwords would be breached under the assumption that these passwords have similar strength to passwords in other datasets like RockYou or Yahoo! Arguably master passwords will be more valuable to an attacker than regular passwords as a master password will unlock multiple user accounts. On the other hand previous research [12] has not found a clear correlation between password strength and account value.

Estimating v. As described in Section 2 the value v represents the value per password when all passwords are released on the market. Thus, although the actual black market prices may vary with supply, the parameter v is fixed. Our estimate of this value parameter will depend on the current black market price, and model parameter a (diminishing returns). In Table 5 we show various estimates of v obtained from multiple estimates of black market password prices. These estimates include measurements from Fossi [16] and more recent estimates from [25], which finds that Yahoo! passwords go for 0.70-1.20 USD on the black market. To obtain the estimates in Table 5, we assume that the black market prices were observed when just 1% of the passwords were on the market. This allows us to esimate the value v if all passwords were to be released using equation 1. We remark that the difference between the two estimates [25] and [16] may be explained due to additional black market supply. We view $a = 0.8$ as substantial diminishing returns e.g., the marginal revenue decreases by a factor of $1/3$ when the attacker compromises all accounts. An interesting direction for future work may be to estimate the parameter a from a longitudinal study of black markets.

Translating between v and $v^{\$}$. Bonneau and Schechter [29] observed that in 2013, Bitcoin miners were able to perform approximately 2^{75} SHA-256 hashes in exchange for bitcoin rewards worth about \$257M. Correspondingly, one can estimate the cost of evaluating a SHA-256 hash to be approximately $C_H = \$7 \times 10^{-15}$. Alternatively, the cost can be viewed as the economic opportunity cost of evaluating each hash function (for instance, renting a botnet or computing on a cloud platform.) Because Bitcoin mining is almost exclusively performed on application specific integrated circuits (ASICs) the above cost analysis implicitly assumes that the attacker is willing to fabricate an an ASIC to evaluate PBKDF2-SHA256 or BCYRPT. We contend that this is a plausible scenario for a rational attacker, since fabrication costs would amortize over the number of user accounts being attacked (e.g., 500+ million). Furthermore, we note that an attacker who is not willing to pay to fabricate an ASIC could obtain similar performance gains using a field programmable gate array (FPGA).

5.2. Results

In section 4 we showed that, if passwords follow CDF-Zipf's law with parameters y and r, and $v/k \geq T(y, r, a)$ then a rational adversary will crack 100% of user passwords. Figure 2(a) plots $v = k \times T(y, r, 0.8)$ for various thresholds from Table 4 including Yahoo! and RockYou. Thus, for a point (v, τ) lying on the blue line, a value v rational adversary will crack 100% of Yahoo! passwords when he can compute the hash function at cost $k = \tau$. Note that $\tau = k$ for hash functions like BCRYPT and PBKDF2 — the ones used by Yahoo!, Dropbox, AshleyMadison and LastPass. For reference, Figure 2(a) includes the actual values of τ selected by AshleyMadison, Dropbox and LastPass as well as the value $\tau = 10^7$. Bonneau and Schechter estimated that SHA256 can be evaluated 10^7 times in 1 second on

[8] An earlier 2013 Yahoo! breach affected approximately 1 billion Yahoo! users [34]. We focus on the 2014 breach because the breach occurred after Yahoo! upgraded their password hashing algorithm from MD5 to BCRYPT. We note that any negative findings about the 2014 breach will certainly extend to the earlier breach since a weaker hashing algorithm was involved.

$R(t_{1\%})$ (USD)	a = 0.8	a = 0.9	a = 1.0
0.70	0.28	0.44	0.70
1.20	0.48	0.76	1.20
4.00	1.59	2.52	4.00
30.00	11.94	18.93	30.00

TABLE 5: v conversion chart

a modern CPU [38]. Thus, 10^7 upper bounds the value of τ that one could select without delaying authentication for more than 1 second when using PBKDF2-SHA256.

The plots predict that, unless we set $\tau \gg 10^7$, the adversary will crack 100% of passwords in almost every instance. In particular, the levels of key-stretching performed by Dropbox, AshleyMadison and even Lastpass are all well below the thresholds necessary to protect Yahoo!, RockYou or CSDN passwords.

Figure 2(b) is similar to Figure 2(a) except that we rescale to y axis to show $v^{\$}$, given monetary estimations of computation cost and password values, so that we can focus on the number of hash iterations necessary to simply avoid all passwords being cracked.

While we do not have CDF-Zipf parameters for other breaches such as AshleyMadison, Dropbox, or LastPass, we do have the value $\tau = k$ for each of these breaches. Figure 2(c) plots $v = k \times T(y, r, 0.8)$ only this time we hold k constant and allow $T(y, r, 0.8)$ to vary. For example, in the black line we fix $k = \tau = 10^5$ since LastPass used PBKDF2-SHA256 with $\tau = 10^5$ hash iterations and allow $T(y, r, 0.8)$ to vary. The vertical lines represent the thresholds $T(y, r, 0.8)$ we derive from CDF-Zipf's law fits for RockYou, Tianya and Yahoo! Table 4 shows the value of $T(y, r, 0.8)$ obtained from 10 different password datasets. Observe that in all of cases we had $T(y, r, 0.8) \leq 7.64 \times 10^7$. As in Figure 2(b) the y-axis in Figure 2(c) is scaled to show the value $v^{\$}$ in USD (estimated). Thus, if Dropbox (resp. AshleyMadison/LastPass) passwords have comparable strength to Yahoo! passwords (resp. Tianya, RockYou) then a rational adversary would crack 100% of these passwords. Indeed, Figure 2(c) shows that unless the thresholds $T(y, r, a)$ for Dropbox/LastPass/AshleyMadison are significantly larger than the previously observed thresholds, a rational adversary would be compelled to crack all passwords, given the range of password values. For example, even if the threshold $T(y, r, a)$ for Dropbox exceeds the threshold for Yahoo! by four orders of magnitude then the adversary will still crack 100% of these passwords.

5.3. Discussion

Figures 2(a), 2(b) and 2(c) paint a grim picture. PBKDF2 and BCRYPT most likely provide dramatically insufficient protection for most AshleyMadison, Dropbox, Yahoo! and LastPass users — even if we used the lowest estimation of the value parameter v from Table 5 ($v^{\$} = 0.28$ USD) and we assume that the attacker faces substantial diminishing returns ($a = 0.8$) for additional cracked passwords. Furthermore, it would not have been possible to provide sufficient protection for users using PBKDF2 or BCRYPT without introducing intolerable authentication delays (≥ 1 second).

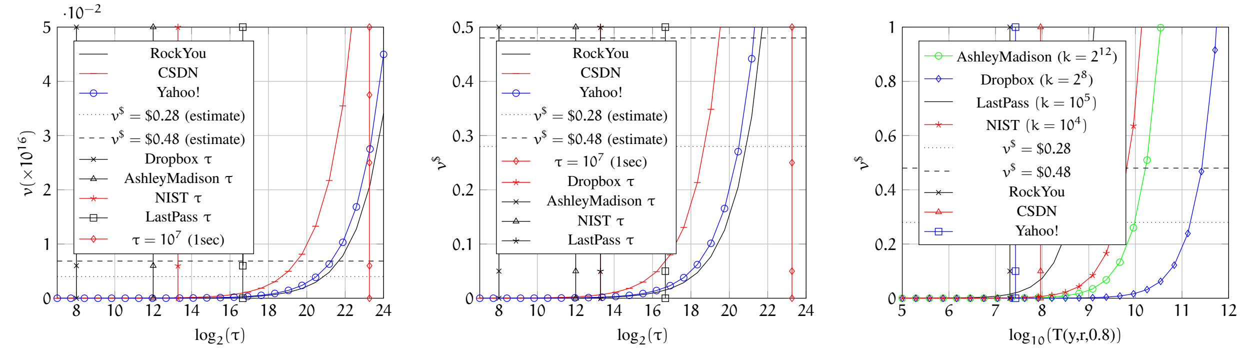
Our analysis assumes that the password distribution truly follows CDF-Zipf's law. While previous research (e.g., [32], [11] and our own results in Section 3) strongly supports the hypothesis that *most* of the password distribution follows Zipf's law, it is not possible to definitively state that the tail of the password distribution does not follow Zipf's law since each of the passwords in the tail were (by definition) observed with low frequency. We stress that even if CDF-Zipf's law does not fit the tail of the password distribution that $T(y, r, a)$ still characterizes adversary behavior. For example, suppose that the $(100 - x)\%$ of passwords follow a Zipf's law distribution with parameters y, r while $x\%$ of passwords in the tail of the password distribution do not. In this case, whenever $v/k \geq T(y, r, a)$ we a rational adversary will crack *at least* $(100-x)\%$ of the user's passwords which follow Zipf's Law.

5.4. Memory Hard Functions

Memory hard functions potentially provide a way of increasing computation cost without drastically increasing computation time. As the name suggests memory hard functions require a large amount of memory to evaluate. Thus, the cost of purchasing/renting hardware for password cracking, approximated by a functions Area x Time (AT) complexity, can be substantial for an attacker. Specifically, AT complexity of SCRYPT [9], scales quadratically with the number of time steps [39]. Thus, as discussed in Section 2, we estimate $k^{\$} = \tau C_H + \tau^2 C_{mem}$, where $C_H \approx \$7 \times 10^{-15}$ [29] and $C_{mem} \approx \frac{C_H}{3000}$ as in [30], [31].

In the last section we assumed that the attacker faced aggressive diminishing marginal returns for additional cracked passwords and we used the lowest possible estimations of adversary value finding that an attacker still cracks 100% of passwords from a Zipf's law distribution. By contrast, in this section we operate under the conservative assumptions that the attacker does not face diminishing returns and we use the larger estimations of adversary value in our analysis. Nevertheless, we find that the use of MHFs can substantially reduce the % of cracked passwords.

Figure 3 plots $v^{\$}$ (estimate) versus the minimum value of τ necessary to prevent a rational attacker from cracking 100% of passwords. For example, the blue line predicts that if Yahoo! had adopted memory hard functions with only $\tau = 2^{20}$ iterations (0.1 seconds) then a value \$30 adversary will not crack all passwords selected from a CDF-Zipf's law distribution with the parameters $y = 0.0211$ and $r = 0.2166$, the parameters for our CDF-Zipf's fit for Yahoo! passwords. By contrast, Yahoo! would need to set $\tau = 2^{26}$ (≈ 7 seconds) when using a function like PBKDF2 or BCRYPT just to ensure that the adversary does not crack 100% of passwords when $a = 1$.



(a) $v/k = T(y,r,0.8)$ for RockYou, CSDN and Yahoo!

(b) $v^\$$ vs. τ for $v = k \times T(y,r,0.8)$.

(c) $v^\$$ versus $T(y,r,0.8)$ when $v = k \times T(y,r,0.8)$, at fixed values of k

Fig. 2: $(a = 0.8)$

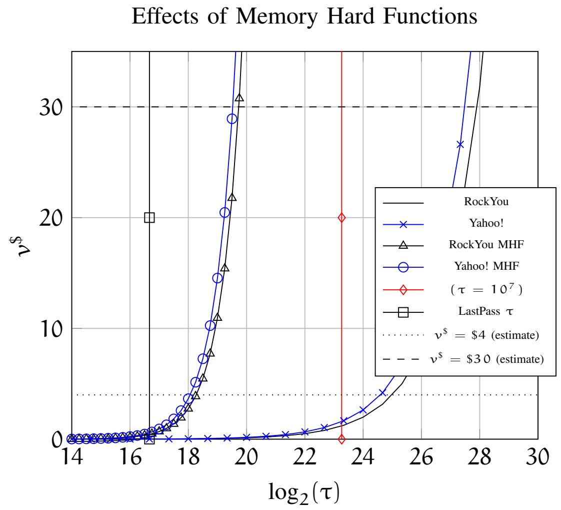


Fig. 3: Memory Hard Functions: $v^\$$ vs τ when $v = k \times T(y,r,1)$ using thresholds $T(y,r,1)$ for RockYou and Yahoo! $k = \tau C_H + \tau^2 C_{mem}$ for MHFs and $k = C_H \times \tau$ otherwise.

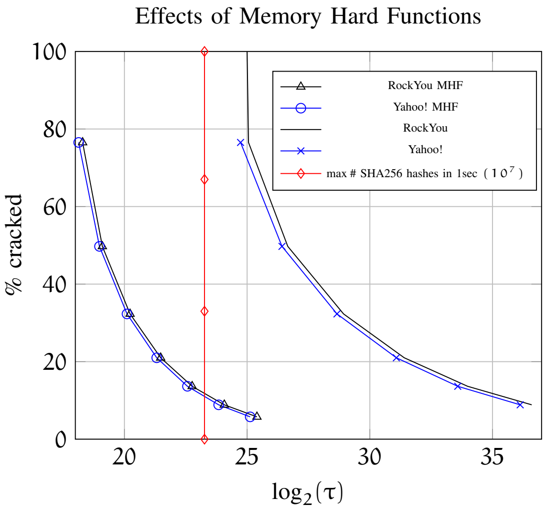


Fig. 4: Memory Hard Functions: % cracked by value $v = \$4$ adversary against MHF with running time parameter τ.

Figure 3 predicts that MHFs prevent a rational adversary from cracking *all* passwords from a Zipf's law distribution. Of course, if the adversary still cracks 99.9% of passwords then this result would not be particularly exciting. Figure 4 plots % cracked passwords vs. τ against a value $v^\$ = \$4$ adversary. These plots provide an optimistic outlook for MHFs. For example, the plots predict that we can significantly reduce the % of cracked passwords (easily below 20%) with out introducing unacceptably long authentication delays when passwords follow a Zipf's law distribution. By contrast, the plots predict that we would need to set $\tau \approx 2^{32}$ (400+ seconds) to achieve the same result using PBKDF2 or BCRYPT when $a = 1$.

6. Model Independent Analysis

In this section we derive model-independent upper and lower bounds on the % of users whose passwords would be cracked by a rational adversary. The advantage of a model independent analysis is that the bounds we derive apply even if we do not make any assumptions about the shape of the password distribution. As we observed previously it is not possible to definitively claim that the tail of the password distribution follows Zipf's law — even if the tail of the distribution is not known to be *inconsistent* with Zipf's law [32], [11]. The disadvantage of a model independent analysis is that the bounds we are able to derive may not always be tight as the bounds we may be able to derive using specific modeling assumptions e.g., Zipf's law. In this section we assume for the sake of simplicity that $a = 1$ i.e., the marginal value of each additional cracked password remains constant.

Suppose that we are given N independent samples

$pwd^1, \ldots, pwd^N \leftarrow \mathcal{X}$ from an (unknown) distribution $\mathcal{X}$. As before, we will let f_i denote the number of users who chose password pwd_i in a dataset and without loss of generality assume that these frequencies are sorted so that $f_i \geq f_{i+1}$. We can use f_i to obtain an estimate $\hat{p}_i = \frac{f_i}{N}$ for p_i, the true probability that that a random user selects the password pwd_i. While we do have $\hat{p}_i \geq \hat{p}_{i+1}$ we stress that we may no longer assume that $p_i \geq p_{i+1}$ since our empirical value $\hat{p}_i$ (resp. $\hat{p}_{i+1}$) may over/under estimate the true probability p_i.

6.1. Lower Bound

Theorem 2 lower bounds the number of passwords that will be cracked by a rational adversary in expectation. The expectation is taken over N passwords sampled from $\mathcal{X}^N$.

Theorem 2. If $\frac{V}{k} \geq NL$ and $a = 1$ then a rational adversary will crack at least

$$\sum_{i:f_i \geq j} f_i - \frac{N}{(j-1)!L^{j-1}}$$

user passwords, in expectation.

The proof of Theorem 2 is in appendix A. The proof begins with the observation that a password $pwd^t = pwd_i$ will certainly be cracked by a value V adversary if $p_i \geq \frac{1}{NL}$. We then introduce the notion of a (j, L)-*bad overestimate*. In particular, a (j, L)-bad overestimate for pwd^t occurs when $p_i < \frac{1}{NL}$ but $f_i \geq j$. If we have $f_i \geq j$ then either $p_i \geq \frac{1}{NL}$ and the password will be cracked, or we have a (j, L)-bad overestimate for the password pwd^t. We can then show that $\frac{N}{(j-1)!L^{j-1}}$ upper bounds the expected number of passwords pwd^t with (j, L)-bad overestimates.

6.2. Upper Bound

In contrast to Theorem 2 , Theorem 3 upper bounds the % of passwords that we expect an attacker to compromise.

Theorem 3. If $\frac{V}{k} \leq NL\left(1 - \frac{1+\epsilon}{N}\sum_{i=1}^{t} f_i\right)$ then, except with probability $\exp\left(-\frac{\epsilon^2 N \sum_{i=1}^{t} p_i}{2}\right)$, a rational adversary will crack at most $\sum_{i:f_i > j} f_i + \mu(N, L, j)$ user passwords where $\mu(N, L, j) =$

$$\sum_{i:0<f_i \leq j} f_i \sum_{\ell=0}^{j-1} \binom{N-1}{\ell} \left(\frac{1}{NL}\right)^{\ell} \left(\frac{NL-1}{NL}\right)^{N-\ell-1} .$$

The proof of Theorem 3 is in appendix A. Briefly, we apply Chernoff bounds to show that, if $\frac{V}{k} \leq NL\left(1 - \frac{1+\epsilon}{N}\sum_{i=1}^{t} f_i\right)$, then with high probability the number of user passwords in our dataset that a rational adversary cracks is at most

$$\sum_{i:f_i \geq j} f_i + \sum_{i} f_i \times C_i .$$

Here, C_i denotes the event that we have a (j, L)-bad underestimate for the password pwd_i. We then separately upper bound the sum $\sum_i f_i \times C_i$ to obtain the bound in Theorem 3.

6.3. Applications

Theorems 2 and 3 allows us to derive different upper and lower bounds by plugging in different values of j and L. For example, by increasing j we decrease the term $\frac{N}{(j-1)!L^{j-1}}$ in Theorem 2, but we also decrease the sum $\sum_{i:f_i \geq j} f_i$. Increasing (resp. decreasing) L is equivalent to assuming the adversary has a higher (resp. lower) value for cracked passwords, which intuitively allows us to establish higher lower bounds (resp. smaller upper bounds) on the percentage of passwords cracked.

6.3.1. Lower Bounds. Applying Theorem 2 we can derive specific lower bounds for each of the datasets studied by [11] as well as for the Yahoo! frequency corpus. For most datasets we obtain our lower bound by setting $j = 2$ and $L = 10$. For the Yahoo! and RockYou datasets we obtained better lower bounds by setting $j = 3$ and $L = 10$. The result appears below:

Dataset	Unique PWs	Total PWs	$\frac{V}{k}$	% cracked
RockYou	14,326,970	32,581,870	3.2582×10^8	46.03
000webhost	10,583,709	15,251,073	1.5251×10^8	20.60
Battlefield	417,453	542,386	5.4239×10^6	13.03
Tianya	12,898,437	30,901,241	3.0901×10^8	48.26
Dodonew	10,135,260	16,258,891	1.6259×10^8	27.66
CSDN	4,037,605	6,428,277	6.4283×10^7	27.19
Mail.ru	2,954,907	4,932,688	4.9327×10^7	30.01
Gmail	3,132,028	4,929,090	4.9291×10^7	26.46
Flirtlife.de	115,589	343,064	3.3406×10^6	56.04
Yahoo!	2.94×10^7	7×10^7	7×10^8	51

Remark: When $j = 1$ we have $\sum_{i:f_i \geq j} f_i - \frac{N}{(j-1)!L^{j-1}} = N - N = 0$ meaning that Theorem 2 provides no lower bound on the % of cracked passwords. At first glance this may appear to be a shortcoming of the theorem. However, we observe that it is impossible to obtain better lower bounds without making assumptions about the password distribution. In particular, let $\mathcal{X}_1$ (resp. $\mathcal{X}_2$) be the uniform distribution over a set of 2^{3n} (resp. 2^{8n}) passwords. Observe that $\mathcal{X}_1$ and $\mathcal{X}_2$ can induce dramatically different rational attacker behavior (e.g., if the value of a password is $2^{3n}k$, the adversary will crack 100% of passwords if the true password distribution is $\mathcal{X}_1$ and 0% of passwords if the true distribution is $\mathcal{X}_2$). However, if we draw $N = 2^n$ samples from $\mathcal{X}_1$ and $\mathcal{X}_2$, then the frequency lists for the two password distributions will be indistinguishable ($f_1 = f_2 = \ldots = f_N = 1$) by birthday bounds ($N \ll 2^{1.5n}$).

6.3.2. Upper Bounds. Similarly, we may use Theorem 3 to derive model independent upper bounds on the percentage of Yahoo! passwords cracked by a rational adversary as shown in Figure 5. As Figure 5 shows we could potentially use memory hard functions to reduce the % of cracked passwords to ≈ 20% without increasing authentication time

V/k	10^8	5×10^7	10^7	5×10^6	10^6
% cracked	100	99	61.38	56.53	52.42
V/k		5×10^5	10^5	5×10^4	10^4
% cracked		42.64	37.46	26.30	22.24

TABLE 6: Model Independent Upper Bound % cracked

past 1 second. This is particularly, impressive when one considers that an attacker only needs a single guess to achieve success rate 1%!

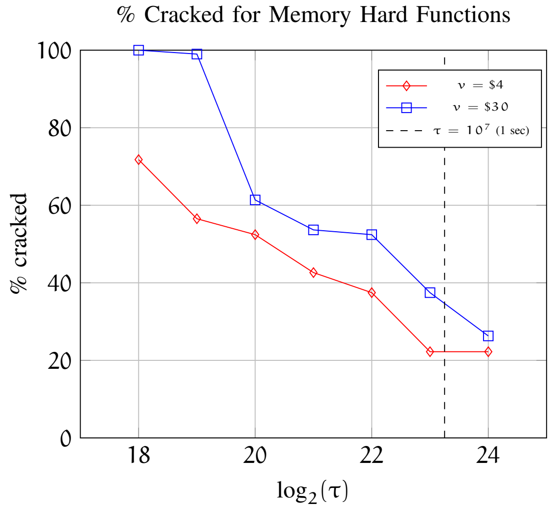


Fig. 5: Memory Hard Functions: % cracked by value $v^{\$} \in \{\$4, \$30\}$ adversary against an ideal MHF with running time parameter τ.

7. Related Work

7.0.1. Password Cracking. The issue of offline password cracking has been known for decades [40]. Password cracking tools have improved steadily as researchers have explored probabilistic password models [41], Probabilistic Context Free Grammars for passwords [42], [43], [44], Markov chain models [45], [46], [47], [48] and even neural networks [49]. Attackers may also use public resources (e.g., quotes from the Internet Movie Database or project Gutenberg to crack sentence based passwords [50], [51]) as well as 'training data' from previous breaches at companies like RockYou or Tianya to improve cracking algorithms. Improved password cracking tools make it all the more crucial to develop secure tools for key-stretching (e.g., data-independent MHFs) to minimize the number of guesses an attacker can try. Allodi has studied the economics of the black market for certain attacks and malware, which may be useful in understanding how password cracking markets may work [52].

7.0.2. Improving Password Strength. Efforts to encourage (or force) users to select stronger passwords have shown limited success [53], [54], [55], [56], [57], [58] and often induce high usability costs [59]. Users can be encouraged to select stronger passwords by providing feedback during the password creation policy (e.g., [60], [61], [62]) or by providing clear instructions for the user to follow when creating passwords [50], [63]. Another extensive line of research explored the use of password composition policies in which a user is required to select a password satisfying certain requirements e.g., contains numbers and/or capital letters [53], [54], [55], [56], [58], [64]. Password composition policies also introduce a high usability cost [57], [65], [66], [59], and they typically do not increase password strength significantly. In fact, sometimes these policies result in weaker user passwords [67], [54]. Similarly, password strength meters often provide inconsistent feedback [61], [62] and they often fail to persuade users to select strong passwords.

Another line of research has focused on helping users to generate and remember passwords. One prominent suggestion is to turn a phrase or a sentence into a password. It has been claimed that these passwords are as strong as random ones [50], [51], and this has been promoted by NIST and by security experts such as Bruce Schneier [68]. However, subsequent research indicates that these suggestions are less secure than previously believed [69], [70]. Another line of research seeks to develop and promote secure and usable strategies for password management when the user needs to create and remember multiple passwords [71], [72], [73], [74]. However, all of these schemes require a motivated user. Bonneau and Schechter [29] and Blocki et al. [63] showed that users are capable of memorizing higher entropy secrets (e.g., 56 bits) by following spaced repetition schedules.

7.0.3. Other Defenses Against Offline Attacks. If an organization has multiple authentication servers then they could distribute storage and/or computation of the password hashes across multiple servers [75], [22], [23], [24]. Juels and Rivest [76] proposed storing the hashes of fake passwords (honeywords) and using a second auxiliary server to detect authentication attempts with honeywords (alerting the organization that an breach has occurred). The expensive requirement to purchase and maintain extra servers may prevent widespread adoption of these proposals. Even if these defenses were adopted there is still a clear need to use secure key-stretching mechanisms — an adversary who breaches both servers can still mount an offline attack. Another line of research has sought to include the solution(s) to hard artificial intelligence problems in the password hash so that an offline attacker needs human assistance to verify each password guess [77], [78], [79]. These solutions increase user workload during authentication e.g., by requiring the user to solve a CAPTCHA puzzle [77], [79].

7.0.4. Modeling the Distribution of User Selected Passwords. Malone and Kevin initially explored the feasibility of modeling the distribution of user password choices using Zipf's law [80]. Wang et al. [32] and Wang and Wang [11] continued this line of work by providing improved techniques to fit Zipf's law parameters to a dataset. Bonneau [12] took a different approach: collect and analyze a massive password frequency corpus with permission from Yahoo! The Yahoo! dataset was recently released using a differentially private algorithm [13]. We elaborate on Zipf's law and the Yahoo! frequency corpus at length in the body of the paper.

7.1. Key-Stretching

Key-stretching was proposed as early as 1979 [40] with the goal of protecting lower-entropy secrets like passwords against offline attacks by making it economically infeasible for an offline attacker to try millions or billions of guesses. Traditionally key stretching has been performed using hash iteration e.g., PBKDF2 [2] and BCRYPT [1]. However, password hash functions like PBKDF2 and BCRYPT require minimal memory to evaluate and thus passwords protected by these hash functions are highly vulnerable to attackers with customized hardware [81]. Memory hard functions (MHFs), first explicitly introduced by Percival [9], are a promising tool for constructing an ideal key-stretching function. MHFs are motivated by the observation that the cost of storing/retrieving items from memory is relatively constant across different computer architectures. At a high level a memory hard function is moderately expensive to compute and most of the costs associated with computing the function are memory related (e.g., storing/retrieving items from memory). Ideally we want the Area x Time complexity of computing a MHF to scale with τ^2, where τ denotes the running time on a standard PC. Intuitively, to compute the MHF once the attacker must dedicate τ blocks of memory for τ time steps, which ensures that the cost of computing the function is equitable across different computer architectures (memory on an ASIC is still expensive). By contrast, Area x Time complexity to compute BCRYPT or PBKDF2 is simply τ. Recall that we want to increase costs quickly to minimize delay during authentication. If costs scale with τ^2 then we can rapidly drive up costs, and if computation requires memory then an adversary will not be able to significantly reduce guessing costs by constructing an ASIC. Almost all of the entrants to the recent Password Hashing Competition (PHC) [8] claimed some form of memory-hardness.

7.1.1. Data (In)dependent Memory Hard Functions. There is a type of MHF called a data-independent MHF (iMHF) which is designed to be resistant to side-channel attacks such as cache timing [82], [83]. These functions have a data access pattern independent of the input. Multiple attacks have been shown in several iMHFs [30], [84], [31], [85], [86], [87], [88]. Data dependent MHFs such as SCRYPT [9] have the previously mentioned side-channel vulnerabilities. Even so, SCRYPT has been found to be optimally memory hard in respect to AT complexity [39], [89].

8. Discussion

Our economic analysis decisively shows that traditional key-stretching tools like PBKDF2 and BCRYPT fail to provide adequate protection for user passwords, while memory hard functions do provide meaningful protection against offline attackers. It is time for organizations to upgrade their password hashing algorithms and adopt modern key-stretching such as memory hard functions [9], [8]. Alternatively, could a creative organization adapt customized Bitcoin mining rigs for use in password authentication? For example, the Antminer S9 [81], currently available on Amazon for approximately $\$3,000$, is capable of computing SHA256 14 trillion times per second. If the organization stored salted and peppered [90], [10] password hash values $u, s_u, SHA256(pwd_u|s_u|p_u)$ then it could potentially use the Antminer S9, or a similar Bitcoin mining rig, to validate a password by quickly enumerating over a (very) large space of secret pepper values p (briefly, a secret salt value that is not stored which even an honest party must brute force).

While our analysis demonstrates that the use of memory hard functions can significantly reduce the fraction of cracked passwords, the damage of an offline attack may still be significant. Thus, we recommend that organizations adopt distributed password hashing [75], [22], [23], [24] whenever feasible so that an attacker who only breaches one authentication server will not be able to mount an offline attack. Furthermore, we recommend that organizations take additional measures to mitigate the affect of an authentication server breach. Solutions might include mechanisms *detect* password breaches through the use of honey accounts or honey passwords[76], multi-factor authentication and fraud detection/correction algorithms to prevent suspicious/harmful behavior [91].

While solid options for password hashing and key-derivation exist [9], [8], [18], [87] the reality is that many organizations and developers select suboptimal password hashing functions [92], [19]. Thus, there is a clear need to provide developers with clear guidance about selecting secure password hash functions. On a positive note recent 2017 NIST guidelines do *suggest* the use of memory hard functions. However, NIST guidelines still allows for the user of PBKDF2 with just $10,000$ hash iterations. Based on our analysis we advocate that password hashing standards should be updated to require the use of memory hard functions for password hashing and disallow the use of non-memory hard functions such as BCRYPT or PBKDF2. It may be expedient for policy makers to audit and/or penalize organizations that fail to follow appropriate standards for password hashing.

We recommend that users primarily focus on selecting passwords that are strong enough to resist targeted online attacks [27] as there is a often a vast gap between the required entropy to resist online and offline attacks [7]. Extra user effort to memorize a high entropy password might be completely wasted if an organization adopts poor password hashing algorithms like SHA1, MD5 [36] or the identity function [92]. This effort would likely be more productively spent on trying to reduce password reuse [72].

9. Acknowledgments

We would like the thank the reviewers for their insightful comments. We would also like to thank Ding Wang for sharing code for computing Zipf fittings. The work was supported by the National Science Foundation under NSF Awards #1649515 and #1704587. Ben Harsha was partially supported by a Intel Graduate Research Assistantship through CERIAS at Purdue. The opinions expressed in this paper are those of the authors and do not necessarily reflect those of the National Science Foundation or Intel.

References

[1] N. Provos and D. Mazieres, “Bcrypt algorithm.” USENIX, 1999.

[2] B. Kaliski, “Pkcs# 5: Password-based cryptography specification version 2.0,” 2000.

[3] L. Breech, “Lastpass security notice,” https://blog.lastpass.com/2015/06/lastpass-security-notice.html/ (retrieved 11/10/2016), 2015.

[4] D. Meyer, “How to check if you were caught up in the dropbox breach,” http://fortune.com/2016/08/31/dropbox-breach-passwords/ (retrieved 11/10/2016).

[5] 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 *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 553–567.

[6] C. Herley and P. C. van Oorschot, “A research agenda acknowledging the persistence of passwords,” *IEEE Security & Privacy*, vol. 10, no. 1, pp. 28–36, 2012.

[7] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, “Passwords and the evolution of imperfect authentication,” *Communications of the ACM*, vol. 58, no. 7, pp. 78–87, 2015.

[8] J.-P. A. et al., “Password hashing competition,” 2015, https://password-hashing.net/.

[9] C. Percival, “Stronger key derivation via sequential memory-hard functions,” in *BSDCan 2009*, 2009.

[10] J. Blocki and A. Datta, “CASH: A cost asymmetric secure hash algorithm for optimal password protection,” in *IEEE 29th Computer Security Foundations Symposium*, 2016, pp. 371–386.

[11] D. Wang and P. Wang, “On the implications of zipf’s law in passwords,” in *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security*, 2016, pp. 111–131.

[12] J. Bonneau, “The science of guessing: analyzing an anonymized corpus of 70 million passwords,” in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 538–552.

[13] J. Blocki, A. Datta, and J. Bonneau, “Differentially private password frequency lists,” in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016*, 2016.

[14] W. Yang, N. Li, I. M. Molloy, Y. Park, and S. N. Chari, “Comparing password ranking algorithms on real-world password datasets,” 2016, pp. 69–90.

[15] 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 *Proceedings of USENIX Security*, 2016.

[16] M. Fossi, E. Johnson, D. Turner, T. Mack, J. Blackbird, D. McKinney, M. K. Low, T. Adams, M. P. Laucht, and J. Gough, “Symantec report on the underground economy,” November 2008, retrieved 1/8/2013.

[17] R. B. Miller, “Response time in man-computer conversational transactions,” in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 1968, pp. 267–277.

[18] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: New generation of memory-hard functions for password hashing and other applications,” in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 2016, pp. 292–302, http://dx.doi.org/10.1109/EuroSP.2016.31.

[19] A. Naiakshina, A. Danilova, C. Tiefenau, M. Herzog, S. Dechand, and M. Smith, “”why do developers get password storage wrong”,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’17, 2017.

[20] P. A. Grassi, E. M. Newton, R. A. Perlner, A. R. Regenscheid, W. E. Burr, J. P. Richer, N. B. Lefkovitz, J. M. Danker, Y.-Y. Choong, K. Greene *et al.*, “Digital identity guidelines: Authentication and lifecycle management,” *Special Publication (NIST SP)-800-63B*, 2017.

[21] M. S. Turan, E. B. Barker, W. E. Burr, and L. Chen, “Sp 800-132. recommendation for password-based key derivation: Part 1: Storage applications,” 2010.

[22] J. Camenisch, A. Lysyanskaya, and G. Neven, “Practical yet universally composable two-server password-authenticated secret sharing,” in *Proceedings of the 2012 ACM conference on Computer and Communications Security*. ACM, 2012, pp. 525–536.

[23] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart, “The pythia prf service,” in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 547–562.

[24] R. W. F. Lai, C. Egger, D. Schröder, and S. S. M. Chow, “Phoenix: Rebirth of a cryptographic password-hardening service,” in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 899–916.

[25] M. Stockley, “What your hacked account is worth on the dark web,” Aug 2016.

[26] C. Herley and D. Florêncio, “Nobody sells gold for the price of silver: Dishonesty, uncertainty and the underground economy,” *Economics of information security and privacy*, pp. 33–53, 2010.

[27] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, “Targeted online password guessing: An underestimated threat,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1242–1254.

[28] Y. Zhang, F. Monrose, and M. K. Reiter, “The security of modern password expiration: an algorithmic framework and empirical analysis,” in *ACM CCS 10: 17th Conference on Computer and Communications Security*, E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, Eds. Chicago, Illinois, USA: ACM Press, Oct. 4–8, 2010, pp. 176–186.

[29] J. Bonneau and S. E. Schechter, “Towards reliable storage of 56-bit secrets in human memory,” in *Proceedings of the 23rd USENIX Security Symposium*, 2014, pp. 607–623.

[30] A. Biryukov and D. Khovratovich, “Tradeoff cryptanalysis of memory-hard functions,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2014, pp. 633–657.

[31] J. Alwen and J. Blocki, “Efficiently computing data-independent memory-hard functions,” in *Advances in Cryptology CRYPTO’16*. Springer, 2016.

[32] X. H. Ding Wang, Gaopeng Jian and P. Wang, “Zipfs law in passwords,” Cryptology ePrint Archive, Report 2014/631, 2014, http://eprint.iacr.org/2014/631.

[33] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *KDD*, 1996, pp. 226–231.

[34] K. Conger, “Yahoo discloses hack of 1 billion accounts,” 2016, retrieved 1/13/2017.

[35] M. Snider and E. Weise, “500 million yahoo accounts breached,” http://www.usatoday.com/story/tech/2016/09/22/report-yahoo-may-confirm-massive-data-breach/90824934/, USA Today, 2016, retrieved 5, Dec. 2016.

[36] CynoSurePrime, “How we cracked millions of ashley madison bcrypt hashes efficiently,” http://cynosureprime.blogspot.com/2015/09/how-we-cracked-millions-of-ashley.html (retrieved 11/10/2016), 2015.

[37] M. Stockley, “What ashley madison got right,” https://nakedsecurity.sophos.com/2015/08/31/what-ashley-madison-got-right/ (retrieved 12/5/2016), 2015.

[38] J. Bonneau and S. Schechter, “”toward reliable storage of 56-bit keys in human memory”,” in *Proceedings of the 23rd USENIX Security Symposium*, August 2014.

[39] J. Alwen, B. Chen, C. Kamath, V. Kolmogorov, K. Pietrzak, and S. Tessaro, “On the complexity of scrypt and proofs of space in the parallel random oracle model,” Cryptology ePrint Archive, Report 2016/100, 2016, http://eprint.iacr.org/.

[40] R. Morris and K. Thompson, “Password security: A case history,” *Communications of the ACM*, vol. 22, no. 11, pp. 594–597, 1979, http://dl.acm.org/citation.cfm?id=359172.

[41] A. Narayanan and V. Shmatikov, “Fast dictionary attacks on passwords using time-space tradeoff,” in *Proceedings of ACM CCS*, 2005, Conference Proceedings, pp. 364–372, http://dl.acm.org/citation.cfm?id=1102168.

[42] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, “Password cracking using probabilistic context-free grammars,” in *IEEE Symposium on Security and Privacy*, 2009, Conference Proceedings, pp. 391–405, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5207658.

[43] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, “Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms,” in *IEEE Symposium on Security and Privacy*, 2012, Conference Proceedings, pp. 523–537.

[44] R. Veras, C. Collins, and J. Thorpe, “On the semantic patterns of passwords and their security impact,” in *Network and Distributed System Security Symposium (NDSS’14)*, 2014.

[45] C. Castelluccia, M. Dürmuth, and D. Perito, “Adaptive password-strength meters from Markov models,” in *Proceedings of NDSS*, 2012, Conference Proceedings.

[46] C. Castelluccia, A. Chaabane, M. Dürmuth, and D. Perito, “When privacy meets security: Leveraging personal information for password cracking,” *arXiv preprint arXiv:1304.6584*, 2013.

[47] J. Ma, W. Yang, M. Luo, and N. Li, “A study of probabilistic password models,” in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, 2014, pp. 689–704.

[48] 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 *Proceedings of the 24th USENIX Security Symposium*. USENIX, Aug. 2015, http://www.ece.cmu.edu/ lbauer/papers/2015/usenix2015-guessing.pdf.

[49] 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 Symposium*, 2016, pp. 175–191.

[50] J. Yan, A. Blackwell, R. Anderson, and A. Grant, “The memorability and security of passwords: some empirical results,” *Technical Report-University Of Cambridge Computer Laboratory*, p. 1, 2000.

[51] ——, “Password memorability and security: Empirical results,” *IEEE Security and Privacy*, vol. 2, no. 5, pp. 25–31, Sep. 2004.

[52] L. Allodi, “Economic factors of vulnerability trade and exploitation: Empirical evidence from a prominent russian cybercrime market,” *ACM CCS ’17*.

[53] 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.

[54] 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, http://dl.acm.org/citation.cfm?id=1979321.

[55] 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, http://doi.acm.org/10.1145/1837110.1837113.

[56] J. M. Stanton, K. R. Stam, P. Mastrangelo, and J. Jolton, “Analysis of end user security behaviors,” *Comput. Secur.*, vol. 24, no. 2, pp. 124–133, Mar. 2005.

[57] P. G. Inglesant and M. A. Sasse, “The true cost of unusable password policies: Password use in the wild,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’10. New York, NY, USA: ACM, 2010, pp. 383–392, http://doi.acm.org/10.1145/1753326.1753384.

[58] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, “Can long passwords be secure and usable?” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’14. New York, NY, USA: ACM, 2014, pp. 2927–2936, http://doi.acm.org/10.1145/2556288.2557377.

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

[60] S. Komanduri, R. Shay, L. F. Cranor, C. Herley, and S. Schechter, “Telepathwords: Preventing weak passwords by reading users’ minds,” in *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 591–606, https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/komanduri.

[61] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, “How does your password measure up? the effect of strength meters on password creation,” in *Proceedings of USENIX Security Symposium*, 2012, Conference Proceedings.

[62] X. de Carné de Carnavalet and M. Mannan, “From very weak to very strong: Analyzing password-strength meters,” in *Network and Distributed System Security Symposium (NDSS 2014)*. Internet Society, 2014.

[63] J. Blocki, S. Komanduri, L. F. Cranor, and A. Datta, “Spaced repetition and mnemonics enable recall of multiple strong passwords,” in *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[64] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, “Designing password policies for strength and usability,” *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 4, p. 13, 2016.

[65] M. Steves, D. Chisnell, A. Sasse, K. Krol, M. Theofanos, and H. Wald, “Report: Authentication diary study,” National Institute of Standards and Technology (NIST), Tech. Rep. NISTIR 7983, 2014.

[66] D. Florêncio, C. Herley, and P. C. Van Oorschot, “An administrator’s guide to Internet password research,” in *Proceedings of the 28th USENIX Conference on Large Installation System Administration*, ser. LISA’14, 2014, pp. 35–52.

[67] J. Blocki, S. Komanduri, A. Procaccia, and O. Sheffet, “Optimizing password composition policies,” in *Proceedings of the fourteenth ACM conference on Electronic commerce*. ACM, 2013, pp. 105–122.

[68] B. Schneier, “Choosing secure passwords,” 2014, https://www.schneier.com/blog/archives/2014/03/choosing_secure_1.html.

[69] C. Kuo, S. Romanosky, and L. F. Cranor, “Human selection of mnemonic phrase-based passwords,” in *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006, pp. 67–78.

[70] W. Yang, N. Li, O. Chowdhury, A. Xiong, and R. W. Proctor, “An empirical study of mnemonic sentence-based password generation strategies,” 2016.

[71] J. Blocki, M. Blum, and A. Datta, “Naturally rehearsing passwords,” in *Advances in Cryptology-ASIACRYPT 2013*. Springer, 2013, pp. 361–380.

[72] D. Florêncio, C. Herley, and P. C. van Oorschot, “Password portfolios and the finite-effort user: Sustainably managing large numbers of accounts.” in *USENIX Security*, 2014, pp. 575–590.

[73] M. Blum and S. S. Vempala, “Publishable humanly usable secure password creation schemas,” in *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.

[74] J. Blocki, M. Blum, and A. Datta, “Toward human computable passwords,” *Innovations in Theoretical Computer Science, ITCS 2017*, 2017.

[75] J. G. Brainard, A. Juels, B. Kaliski, and M. Szydlo, “A new two-server approach for authentication with short secrets.” in *USENIX Security*, vol. 3, 2003, pp. 201–214.

[76] A. Juels and R. L. Rivest, “Honeywords: Making password-cracking detectable,” in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2013.

[77] R. Canetti, S. Halevi, and M. Steiner, *Mitigating Dictionary Attacks on Password-Protected Local Storage*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 160–179, http://dx.doi.org/10.1007/11818175_10.

[78] J. Blocki, M. Blum, and A. Datta, “Gotcha password hackers!” in *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*. ACM, 2013, pp. 25–34.

[79] J. Blocki and H.-S. Zhou, *Designing Proof of Human-Work Puzzles for Cryptocurrency and Beyond*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 517–546, http://dx.doi.org/10.1007/978-3-662-53644-5_20.

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

[81] A. D. Europe, “Antminer s9,” http://www.antminerdistribution.com/antminer-s9/.

[82] D. J. Bernstein, “Cache-Timing Attacks on AES,” http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[83] C. Forler, S. Lucks, and J. Wenzel, “Catena: A memory-consuming password scrambler.” *IACR Cryptology ePrint Archive*, vol. 2013, p. 525, 2013.

[84] J. Alwen and V. Serbinenko, “High Parallel Complexity Graphs and Memory-Hard Functions,” in *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, ser. STOC ’15, 2015, http://eprint.iacr.org/2014/238.

[85] J. Alwen and J. Blocki, “Towards Practical Attacks on Argon2i and Balloon Hashing,” in *Proceedings of the 2nd IEEE European Symposium on Security and Privacy (EuroS&P 2017)*. IEEE, 2017, pp. 142–157, http://eprint.iacr.org/2016/759.

[86] J. Blocki and S. Zhou, “On the depth-robustness and cumulative pebbling cost of Argon2i,” in *TCC 2017: 15th Theory of Cryptography Conference, Part I*, ser. Lecture Notes in Computer Science, Y. Kalai and L. Reyzin, Eds., vol. 10677. Baltimore, MD, USA: Springer, Heidelberg, Germany, Nov. 12–15, 2017, pp. 445–465.

[87] J. Alwen, J. Blocki, and B. Harsha, “Practical graphs for optimal side-channel resistant memory-hard functions,” in *ACM CCS 17: 24th Conference on Computer and Communications Security*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. Dallas, TX, USA: ACM Press, Oct. 31 – Nov. 2, 2017, pp. 1001–1017.

[88] J. Alwen, J. Blocki, and K. Pietrzak, “Depth-robust graphs and their cumulative memory complexity,” in *Advances in Cryptology – EUROCRYPT 2017, Part II*, ser. Lecture Notes in Computer Science, J. Coron and J. B. Nielsen, Eds., vol. 10211. Paris, France: Springer, Heidelberg, Germany, May 8–12, 2017, pp. 3–32.

[89] J. Alwen, B. Chen, K. Pietrzak, L. Reyzin, and S. Tessaro, “Scrypt is Maximally Memory-Hard,” in *Advances in Cryptology-EUROCRYPT 2017*. Springer, 2017, http://eprint.iacr.org/2016/989.

[90] U. Manber, “A simple scheme to make passwords based on one-way functions much harder to crack,” *Computers & Security*, vol. 15, no. 2, pp. 171–176, 1996.

[91] D. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto, “Who are you? A statistical approach to measuring user authenticity,” in *ISOC Network and Distributed System Security Symposium – NDSS 2016*. San Diego, CA, USA: The Internet Society, Feb. 21–24, 2016.

[92] J. Bonneau and S. Preibusch, “The password thicket: technical and market failures in human authentication on the web,” in *Proceedings of the Ninth Workshop on the Economics of Information Security (WEIS)*, Jun. 2010, http://weis2010.econinfosec.org/papers/session3/weis2010_bonneau.pdf.

Appendix

Reminder of Theorem 2. *If* $\frac{V}{k} \geq NL$ *and* $a = 1$ *then a rational adversary will crack at least*

$$\sum_{i:f_i \geq j} f_i - \frac{N}{(j-1)!L^{j-1}}$$

user passwords, in expectation.

Proof of Theorem 2:

We observe that a user password $pwd^t = pwd_i$ will be certainly cracked if $V \times \Pr[pwd] = Vp_i \geq k$ since the marginal cost of including an extra guess pwd in the dictionary is at most k. Thus, the adversary will compromise at least $\sum_{i:p_i \geq k/V} f_i$ accounts. The problem with this lower bound is that we need to know $p_i = \Pr[pwd_i]$ for each password pwd_i to compute it. However, the values p_i are unknown if we do not make assumptions about the shape of the password distribution. However, we can lower bound this quantity. In particular, we say that the estimate $\hat{p}_i = f_i/N$ is a (j, L)-*bad overestimate* if $p_i < \frac{1}{NL}$, but $f_i \geq j$. Let B_i be an indicator random variable for the that $\hat{p}_i$ is (j, L)-bad. Then the sum $\sum_i f_i \times B_i$ computes the total number of users whose password got a (j, L)-bad overestimate. The proof now follows from Claims 4, 5 and 6. Claim 4 lower bounds the fraction of cracked passwords in terms of the events B_i.

Claim 4. If $\frac{V}{k} \geq NL$ then the number of user passwords in our dataset that a rational adversary cracks is at least

$$\sum_{i:f_i \geq j} f_i - \sum_i f_i \times B_i \ .$$

Proof : Suppose that a user selects a password pwd_i with $p_i \geq \frac{1}{NL}$. Since $Vp_i > k \geq \max_t MC(t)$ the marginal reward of guessing pwd_i always exceeds the marginal cost. Thus, a rational attacker *must* eventually guess pwd_i. However, if $p_i < \frac{1}{NL}$ then either $f_i < j$ or $\hat{p}_i$ is a (j, L)-bad overestimate of p_i. Let S denote the set of users who picked a password i such that $f_i \geq j$ and let $T \subseteq S$ denote the set of users whose password got a (j, L)-bad overestimate. Any user in the set $S \backslash T$ will be compromised eventually. Thus, at least $|S \backslash T| = \sum_{i: f_i \geq j} f_i - \sum_i f_i \times B_i$ since $|T| = \sum_i f_i \times B_i$ and $|S| = \sum_{i: f_i \geq j} f_i$. □

Claim 5 bounds the probability of the event B_i — that is the probability that we observe password pwd_i, with $p_i < \frac{1}{NL}$, at least $f_i \geq j$ times conditioned on the event that we observe pwd_i at least once.

Claim 5. $\Pr[B_i] \leq \frac{1}{(j-1)!L^{j-1}}$

Proof : We first observe that

$$\Pr[B_i] \leq \binom{N}{j-1} p_i^{j-1} \ .$$

Recall that for an event which is (j, L)-bad, we have that by definition, $p_i < \frac{1}{NL}$. Thus

$$\begin{aligned} \binom{N}{j-1} p_i^{j-1} &= \frac{N!}{(j-1)!(N-j+1)!} p_i^{j-1} \\ &\leq \frac{N^{j-1}}{(j-1)!} p_i^{j-1} \leq \frac{1}{(j-1)!L^{j-1}} . \end{aligned}$$

□

Finally, Claim 6 shows that we cannot have too many bad overestimates.

Claim 6. $\sum_i f_i \times \mathbb{E}[B_i] \leq \frac{N}{(j-1)!L^{j-1}}$

Proof : Follows immediately from Claim 5 by substituting $\frac{1}{(j-1)!L^{j-1}}$ for $\mathbb{E}[B_i]$ in the above sum. □

□

Reminder of Theorem 3. *If* $\frac{V}{k} \leq NL\left(1 - \frac{1+\epsilon}{N} \sum_{i=1}^t f_i\right)$ *then, except with probability* $\exp\left(-\frac{\epsilon^2 N \sum_{i=1}^t p_i}{2}\right)$*, a rational adversary will crack at most* $\sum_{i: f_i > j} f_i + \mu(N, L, j)$ *user passwords where* $\mu(N, L, j) =$

$$\sum_{i: 0 < f_i \leq j} f_i \sum_{\ell=0}^{j-1} \binom{N-1}{\ell} \left(\frac{1}{NL}\right)^{\ell} \left(\frac{NL-1}{NL}\right)^{N-\ell-1} \ .$$

Proof of Theorem 3: Given N independent samples $pwd^1, \ldots, pwd^N \leftarrow \mathcal{X}$ we use $\mathsf{Pop}_t = \{pwd_1, \ldots, pwd_t\}$ to denote the t most common passwords from these samples and let X_i be an indicator variable for the event $pwd^i \in \mathsf{Pop}_t$.

Claim 7.

$$\sum_{i=1}^{N} X_i \leq \sum_{i=1}^{t} f_i .$$

Proof : Let $pwd_1, \ldots, pwd_t, \ldots$ denote the list of observed passwords ordered by observed frequency. Let $i_1 > \ldots > i_t$ be given such that $\mathsf{Pop}_t = \{pwd_{i_1}, \ldots, pwd_{i_t}\}$. Now we have

$$\sum_{j=1}^{N} X_j = \sum_{j=1}^{t} f_{i_j} \leq \sum_{j=1}^{t} f_j \ .$$

□

.

Claim 8. We have

$$\sum_{i=1}^{t} p_i \leq \frac{(1+\epsilon)}{N} \sum_{i=1}^{N} X_i$$

except with probability

$$\mathbf{Pr}\Big[\sum_{i=1}^{N} X_i \leq \frac{N}{1+\epsilon} \sum_{i=1}^{t} p_i\Big] \leq \exp\left(-\frac{\epsilon^2 N \sum_{i=1}^t p_i}{2}\right) .$$

Proof : Since $\mathbf{Pr}\Big[X_i = 1\Big] = \sum_{i=1}^t p_i$, then $\mathbf{E}\Big[\sum_{i=1}^N X_i\Big] = N \sum_{i=1}^t p_i$. Then applying Chernoff bounds,

$$\mathbf{Pr}\Big[\sum_{i=1}^{N} X_i \leq \frac{N}{1+\epsilon} \sum_{i=1}^{t} p_i\Big] \leq \exp\left(-\frac{\epsilon^2 N \sum_{i=1}^t p_i}{2}\right) .$$

□

Claim 9. With high probability,

$$MC(t) \geq \left(1 - \frac{1+\epsilon}{N} \sum_{i=1}^{t} f_i\right) k.$$

Proof : By Claims 7 and 8,

$$\sum_{i=1}^{t} p_i \leq \frac{1+\epsilon}{N} \sum_{i=1}^{t} f_i.$$

The proof follows from the observation that $MC(t) = \left(1 - \sum_{i=1}^{t-1}\right) k$ and $p_t \geq 0$. □

Now, we define $\hat{p}_i = f_i/N$ as a (j, L)-*bad underestimate* if $p_i > \frac{1}{NL}$, but $f_i \leq j$. Then define C_i as the indicator variable for the event that $\hat{p}_i$ is a (j, L)-bad underestimate and $f_i \geq 1$.

Claim 10. If $\frac{V}{k} \leq NL\left(1 - \frac{1+\epsilon}{N} \sum_{i=1}^t f_i\right)$ then the number of user passwords in our dataset that a rational adversary cracks is at most

$$\sum_{i: f_i \geq j} f_i + \sum_i f_i \times C_i \ .$$

Proof : Suppose that a user selects a password pwd_i with $p_i \leq \frac{1}{NL}$. Since $Vp_i < NL\left(1 - \frac{1+\epsilon}{N} \sum_{i=1}^t f_i\right) k \leq MC(t)$ the marginal reward of guessing pwd_i never exceeds the marginal cost. Thus, a rational attacker *never* chooses to guess pwd_i. If $p_i > \frac{1}{NL}$ then either $f_i > j$ or $\hat{p}_i$ is a (j, L)-bad underestimate of p_i. Let S denote the set of users who picked a password i such that $f_i > j$ and let $T \subseteq S$ denote the set of users whose password got a (j, L)-bad underestimate. Only the users in the set $S \cup T$ may be compromised eventually. Thus, at most $|S \cup T| \leq \sum_{i:f_i \geq j} f_i + \sum_i f_i \times C_i$ since $|T| = \sum_{i:f_i \leq j} f_i \times C_i$ and $|S| = \sum_{i:f_i > j} f_i$. □

Then the following immediately holds, noting that there can be at most NL passwords which are (j, L)-bad underestimates:

Corollary 11. If $\frac{V}{k} \leq NL\left(1 - \frac{1+\epsilon}{N}\sum_{i=1}^{t} f_i\right)$ then the number of user passwords in our dataset that a rational adversary cracks is at most

$$\sum_{i:f_i \geq j} f_i + \sum_{i}^{i+NL} f_i \ .$$

Claim 12.

$$\Pr\left[C_i \mid f_i \geq 1\right] \leq \sum_{k=0}^{j-1} \binom{N-1}{\ell} \left(\frac{1}{NL}\right)^{\ell} \left(\frac{NL-1}{NL}\right)^{N-\ell-1}$$

Proof : Recall that for $C_i = 1$, we require $p_i > \frac{1}{NL} \geq \frac{j}{N}$ but $f_i \leq j$. Then for $j < N/2$,

$$\begin{aligned}\Pr\left[C_i \mid f_i \geq 1\right] &= \sum_{k=0}^{j-1} \binom{N-1}{\ell} p_i^{\ell} (1-p_i)^{N-\ell-1} \\ &\leq \sum_{k=0}^{j-1} \binom{N-1}{\ell} \left(\frac{1}{NL}\right)^{\ell} \left(\frac{NL-1}{NL}\right)^{N-\ell-1}\end{aligned}$$

□

Claim 13. $\sum_{i:0<f_i\leq j} f_i \times \mathbb{E}\left[C_i \mid f_i \geq 1\right]$.

Proof : Follows immediately from Claim 12 by substituting into $\mathbb{E}\left[C_i \mid f_i \geq 1\right]$ in the above sum. □

□

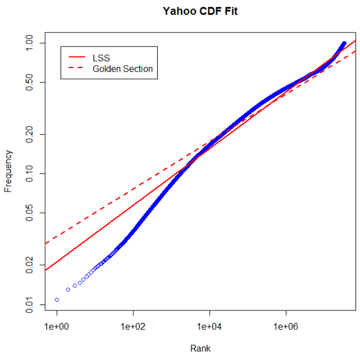
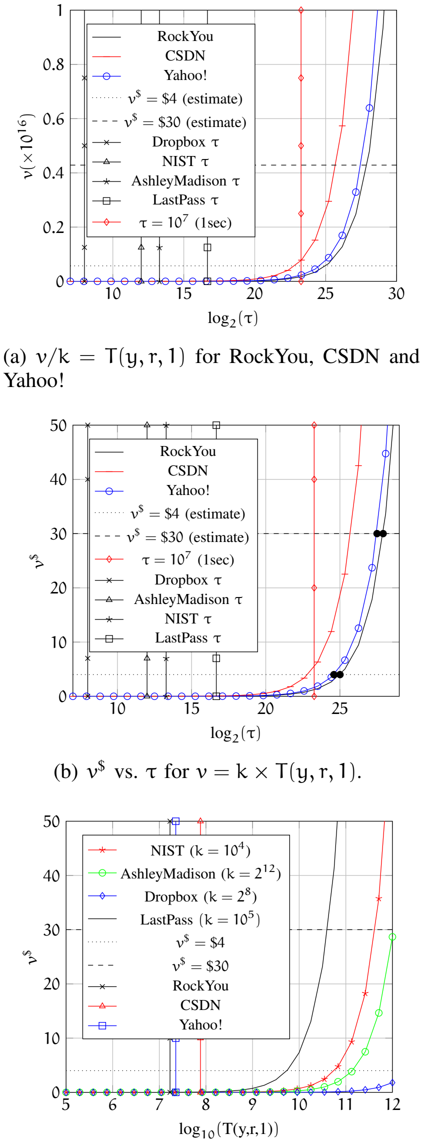


Fig. 6: Yahoo! CDF-Zipf Fittings



(a) $v/k = T(y, r, 1)$ for RockYou, CSDN and Yahoo!

(b) $v^{\$}$ vs. τ for $v = k \times T(y, r, 1)$.

(c) $v^{\$}$ versus $T(y, r, 1)$ when $v = k \times T(y, r, 1)$, at fixed values of k

Fig. 7: No Diminishing Returns $(a = 1)$