Password Distribution and Its Impacts on Offline Password Guessing Economics

Zhenduo Hou, Ding Wang(✉), Yifei Zhang
College of Cryptology and Cyber Science, Nankai University, Tianjin 300350, China; wangding@nankai.edu.cn
Key Laboratory of Data and Intelligent System Security (NKU), Ministry of Education, Tianjin 300350, China
Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China

***Abstract*—How to fit passwords with a concise distribution model is one of the most fundamental questions in password research. Considerable efforts have been devoted to proposing various fitting methods, including maximum likelihood (ML), least linear square (LLS) and golden section search (GSS), to fit the state-of-the-art password distribution model (i.e., the CDF-Zipf model). In this paper, based on 243.83 million real-world passwords, we, *for the first time*, propose a set of three criteria, i.e., accuracy, stability, and a new completeness criterion, to systematically evaluate these three password distribution fitting methods. We reveal that only GSS satisfies *all* three criteria.**

As a key application of the GSS-fitted CDF-Zipf model, we investigate its practical impacts on offline guessing economics. We propose a generic framework to capture the costs of recommended memory hard functions (MHFs), and *for the first time*, provide concrete empirical evidence from real-world MHF (the leading Argon2i) executions. Armed with these results, we propose two main attacker models, with each catering to attackers' varied guessing capabilities. Within each model, we design a rational attack strategy in the leading profit-driven scenario. Extensive experiments on eight large-scale datasets reveal that MHFs cannot provide sufficient security: Even if the account value is as low as $1, (1) under the idealized attacker model, as high as 14.17%∼100% of passwords will be cracked, and the MHF security has been overestimated by 71.14%∼423.29%; (2) Under the advanced model that incorporates real-world guessing algorithms, 10.25%∼36.82% of passwords will be cracked, and they go beyond the most popular passwords. Besides these attacker models, we also provide approximations for the cracked proportion, allowing security administrators to evaluate password security without performing expensive offline guessing attacks. In all, this work lays a solid foundation for hundreds of password-based applications that employ the CDF-Zipf model.

I. Introduction

Human-chosen passwords are the most widely used authentication on the web. Though they are known to suffer issues in both security (e.g., guessing [77], [82], [87], phishing [59], [70]) and usability (e.g., memorization [39]), and various replacements (e.g., password managers [6], [7], [8] and FIDO2 [3]) have been proposed, passwords remain dominant due to remarkable simplicity, easiness to change and low deployment cost [24], [26], [36], [84]. In particular, password managers and passwordless authentication have not been widely accepted due to usability and security concerns [29], [40], [50], [63]. Hence, passwords will prevail in the foreseeable future.

Though considerable efforts have been made to nudge users towards stronger passwords (e.g., via password strength meters [75]), the current landscape is far from satisfactory: Users tend to choose popular passwords [76] and reuse passwords across sites [57]. It is estimated that 1.01%∼10.44% of users choose the top-10 most popular passwords (e.g., `iloveyou` and `abc123`) [76]. Besides, 72% of young users still reuse passwords despite being aware of the risks [4]. These vulnerable behaviors make password distributions highly skewed, and far from uniform. However, exact password distributions are hard to characterize, forcing password-based applications (e.g., cryptographic protocols and encryption) to rely on the unrealistic assumption that user-chosen passwords are uniform. For instance, a large number of provably secure PAKE protocols (e.g., [10], [32], [47]) assume that "a password is chosen independently and uniformly at random from a dictionary" [47], which can lead to orders of magnitude security overestimates [73]. *This outlines the necessity to accurately characterize password distributions for password-based applications.*

To address this concern, following the idea that passwords follow Zipf's law like human languages [85], Bonneau used the maximum likelihood to fit 70 million passwords. However, he found that passwords were unlikely to be Zipf distributed [25], and Malone and Maher independently reached a similar conclusion [53]. Since the probabilities of unpopular passwords often deviate significantly, they are unlikely to be approximated by empirical frequencies. With this in mind, in 2017, Wang et al. [73] proposed the CDF-Zipf model that fits the cumulative distribution function (CDF), rather than the probability density (mass) function (PDF) as in [25], [53]. They used the golden section search (GSS) fitting method to minimize the Kolmogorov-Smirnov (KS) statistic [33], which is the maximum of CDF deviations between the fitted model and real-world passwords. Their model not only achieves the smallest KS statistic, but also covers the entire dataset. Due to the easiness to use, so far, the CDF-Zipf model has been adopted by about 300 password-based applications across seven domains (e.g., [22], [23], [35], [58], [65], [79], [82])[1].

Guessing attacks are one of the most severe threats to

Network and Distributed System Security (NDSS) Symposium 2027
22-26 March 2027, Seoul, Republic of Korea
ISBN 978-1-970672-09-1
https://dx.doi.org/10.14722/ndss.2027.231286
www.ndss-symposium.org

[1]A list of over 300 studies is shown at https://tinyurl.com/2kpxwjsx .

passwords, and password strengths largely depend on underlying distributions. Offline guessing attacks, fueled by unending data breaches, are particularly concerning since attackers can download and guess passwords without interacting with authentication servers. To resist such attacks, a widely accepted approach is utilizing a "moderately expensive" password hash function [38] to make the computation cost acceptable for users but prohibitive for attackers trying an abnormally large number of guesses (e.g., 10^{14} [38]). The mainstream slow hashes achieve this goal by iterating round functions (e.g., SHA256 for PBKDF2 [45]). However, due to the progress in hardware, the costs of running slow hashes can be reduced drastically by GPUs [34] and even application-specific integrated circuits (ASICs). For instance, after the 2022 LastPass data breach, attackers have successfully extracted over $35 million cryptocurrency by offline guessing users' master passwords [48] despite the LastPass's claim that "cracking users' master passwords is extremely difficult" [71].

To fight against increasingly powerful offline attackers, *memory hard functions* (MHFs) have been proposed [60]. MHFs utilize the property that the cost of extracting/writing data from/into memory is relatively stable across hardware architectures. Among various MHFs, Argon2i [17] (the winner of Password Hashing Competition [1]) has been adopted by a variety of services (see [7], [8], [72]). *To find if attackers' guessing costs are indeed "prohibitive", evaluations of the MHF cost under real-world constraints (e.g., memory configurations) are needed.* However, as far as we know, only Blocki et al. [22] and Ren-Devadas [64] have preliminarily explored this issue, while their results are inconsistent (see Sec. IV-B). In particular, the fundamental issue of how the MHF cost scales with its memory usage has not been well explored, and neither result comes from real-world MHF executions.

Besides, little attention has been paid to how attackers with real-world guessing capabilities should guess. Existing studies (e.g., [15], [19], [22]) mainly focus on the *idealized* model, where the attacker generates a 100% accurate guessing dictionary, and only does not know which password is chosen by a particular user. This is impractical since: (1) Guesses generated by practical guessing algorithms (e.g., Markov [52], FLA [54], and PassLLM [87]) may match *no* hashed password in a breached dataset; (2) Not every password in a breached dataset can be cracked. In this way, prior attack strategies, such as stopping guessing once a new guess is not lucrative [16], [19], [22], may not be applicable in real-world attacks to obtain the maximum profit. Further, it is unknown how security administrators can use such an economic analysis to better protect users' passwords. All this calls for well-grounded and principled research on offline guessing economics.

A. Motivations and challenges

To evaluate whether password hash functions can make attackers' offline guessing costs *prohibitively expensive*, the following key research questions (**RQ**s) need to be addressed:

- **RQ1**: Whether existing password fitting methods are sufficiently sound in terms of key evaluation criteria? How can a fitted password distribution model be accurately and reliably applied in various password-based applications?
- **RQ2**: How to build cost frameworks that can accurately and practically capture attackers' real-world hashing costs, particularly for memory hard functions (MHFs)?
- **RQ3**: What are the optimal attack strategies that rational attackers with realistic guessing capabilities can set in major attack scenarios? How can security administrators evaluate security impacts without cracking actions?

For **RQ1**, there is a lack of systematic evaluations regarding password fitting methods. Firstly, existing studies [22], [42], [73] only focus on the accuracy criterion, *and the results are largely inconsistent.* In 2017, Wang et al. [73] used golden section search (GSS) to fit their proposed CDF-Zipf model, and found that its accuracy outperforms the prior least linear square (LLS) fitted PDF-Zipf [73] and Clauset et al.'s maximum likelihood (ML) [33] fitted models, and can cover the entire dataset. However, in 2018, Blocki et al. [22] proposed using LLS to fit the CDF-Zipf model, and claimed that LLS is more accurate than GSS. To the best of our knowledge, this key and long-standing issue has not been well addressed.

Secondly, besides accuracy, a fitting method should yield reliable CDF-Zipf models/parameters that are sufficiently close to the optimal (yet usually unknown) ones (i.e., completeness); Otherwise, a dataset may be better characterized by other parameters, making fitted results biased. However, only Clauset et al. [33] have explored the completeness of the maximum likelihood (ML) on general power-law distributions, and it is unclear whether ML's completeness holds for passwords. Other password fitting methods, such as GSS [73] and LLS [22], have never been rigorously examined. This is challenging, as it requires in-depth explorations of underlying statistics.

Thirdly, the classic stability criterion, which explores *to what extent fitting results of one dataset can be applied to datasets of different sizes*, has been left unexplored. Since many datasets (e.g., Yahoo [46] and R2Games [62]) have been leaked over once, and passwords of the same language, service and creation policies are distributed in similar shapes [76], fitting results of one dataset can be used to similar/the same datasets of varied sizes. At IEEE S&P'12, Bonneau used ML to fit the 70 million Yahoo dataset (see Table I) with a power-law, found no stability holds, and concluded that passwords are not Zipf-distributed [25]. However, whether other fitting methods possess stability has not been investigated.

For **RQ2**, existing MHF cost frameworks (e.g., [22], [64]) have not been validated by empirical evidence from real-world MHF executions. Particularly, a number of theoretical MHF studies (e.g., [12], [13], [14], [18], [21]) assumed that the overall energy cost increases quadratically with memory usage, which can lead to serious overestimates of the MHF security once such assumptions do not hold. Additionally, the MHF cost comprises multiple components, such as memory reading/writing and round function (e.g., Blake2b in Argon2i [17]) computation, and attackers may use advanced GPUs and ASICs to significantly reduce the computation cost of round functions, so it is crucial to model and quantify different

components of the MHF cost. To do this, the energy cost should be meticulously measured and disentangled.

Upon **RQ1** and **RQ2**, for **RQ3**, existing studies on offline guessing economics are also undesirable. In practice, not every guess in an algorithm-generated guessing dictionary (e.g., Markov [52], PCFG [80], and PassLLM [87]) can match a breached password, and not every password can be cracked. Thus, in major attack scenarios such as profit-driven ones (see [15], [16], [19], [22]), an attacker needs to determine *to what extent she should continue guessing to obtain potential rewards, or stop to avoid further costs.* Besides, it is challenging for security administrators to develop accurate and applicable methods to evaluate security impacts, since such tasks require *estimating* attackers' cracked proportions under varied guessing capabilities *without performing expensive cracking actions*.

B. Our contributions

- **Evaluations of password fitting methods**. We propose a set composed of accuracy, stability, and a new completeness criterion to evaluate fitting methods for password distributions, and examine the mainstream fitting methods: maximum likelihood (ML) [33], least linear square (LLS) [22], and golden section search (GSS) [73]. *We find that only GSS can satisfy all criteria.* For accuracy, we correct the flawed KS statistic 0.0094 (of 70 million Yahoo passwords) in [22] to 0.0871, and get our results confirmed by the authors of [22]. For completeness, we use both rigorous mathematics and extensive experiments to reveal that GSS holds completeness under the KS statistic, so its results are reliable. For stability, our extensive experiments reveal that GSS performs the best. This supports the usage of Wang et al.'s GSS-fitted CDF-Zipf model [73] in over 300 password-based applications.
- **Practical MHF measurements**. We propose an accurate and practical MHF cost framework based on both MHF designs and concrete empirical evidence from real-world MHF executions. We first propose a generic MHF cost framework upon the design of the promising Argon2i [17]. Then, we measure different components (i.e., busy and idle memory energy costs, computation cost, purchase cost) of the MHF cost in real-world Argon2i executions. The empirical evidence validates our generic framework, resolves the inconsistencies between prior Blocki et al.'s [22] and Ren-Devadas's models [64]. Our findings lay a solid foundation for modeling the MHF cost, and thus offline guessing costs in the real world.
- **Analysis of offline guessing economics**. We propose two main attacker models catering to attackers' varied guessing capabilities under the profit-driven attack scenario. Our experiments reveal that MHFs alone cannot provide sufficient security under the recommended settings. Under the idealized $\mathcal{A}_1$ model, the MHF security has been *overestimated* by 71.14%∼423.29% by the prior model in [22]. Under the more advanced $\mathcal{A}_2$ model that incorporates leading guessing algorithms an attacker will still crack 10.25%∼36.82% of passwords. We reveal that our cracked passwords go beyond the most popular (e.g., top-10^3) ones via feature analysis. Additionally, with an estimated number of unmatched guesses, security administrators can approximate the cracked proportion via our attacker models in relatively tight manners without expensive cracking actions. We also discuss insights of our work on more general password-based applications.

TABLE I: Basic information of password datasets (PW=password).

Dataset	Service type	Language	When breached	PW form	Total PW
Yahoo†	Portal	English	May, 2011	Frequency	69,301,337
Dodonew	Ecommerce	Chinese	Dec., 2011	Plaintext	16,258,891
000Webhost	Web hosting	English	Oct., 2015	Plaintext	15,251,073
Rockyou	Social forum	English	Dec., 2009	Plaintext	32,575,499
Tianya	Social forum	Chinese	Dec., 2011	Plaintext	30,901,241
Duelingnetwork	Gaming	English	Mar., 2017	Unsalted MD5	6,487,335
Chegg	Education	English	Apr., 2018	Unsalted MD5	38,997,233
Wishbone‡	Entertainment	English	Jan., 2020	Unsalted MD5	10,196,679
Dailyquiz	Gaming	English	Jan., 2021	Plaintext	12,852,658
Datpiff	Music creation	English	Aug., 2021	Plaintext	7,183,478
Raaga§	Music streaming	Indian	Dec., 2025	Unsalted MD5	3,826,649

† The Yahoo password frequency dataset is published differentially private by Blocki et al. [20], and is available on https://tinyurl.com/3vsjsdfu.
‡ Wishbone was stored in unsalted MD5, and we manage to recover 95.62% of them.
§ The contents of Raaga are mostly in Hindi and other local languages of India. Its passwords were stored in unsalted MD5, and we have recovered 98.30% of them.

II. Preliminaries

A. Datasets

We describe our eleven large-scale datasets in Table I, comprising 243.83 million real-world passwords leaked between 2009 and 2025. These datasets span both early and recent breaches, and are diverse in languages and service types, enabling the representation of users' real-world passwords. For instance, across English-language datasets, the top-10 and top-100 most popular passwords account for 1.63% and 3.29% of those leaked before 2020, compared with 1.27% and 3.13% after 2020, indicating a marginal trend towards adopting more secure passwords as observed in prior studies [23], [25].

For a dataset with N unique and $|D|$ total passwords, we rank passwords from the most to the least popular, and denote $\hat{f}_i$ as the frequency of the i-th unique password in the real-world (i.e., empirical) dataset RealSet. In this way, for passwords $PW_1, PW_2, ...PW_N$, there is $\hat{p}_1 \geq \hat{p}_2 \geq ... \geq \hat{p}_N$, where $\hat{p}_i = \hat{f}_i/|D|$ is the probability density (mass) function (PDF), and the cumulative distribution function (CDF) of RealSet up to rank r is $\hat{P}_r = \sum_{i=1}^{r} \hat{p}_i$. Similarly, for the fitted theoretical model/dataset FitSet, the frequency of the i-th password is f_i, and the CDF is $P_r = \sum_{i=1}^{r} p_i$, with $p_i = f_i/|D|$.

B. Distance metrics between distributions

The Kolmogorov-Smirnov (KS) statistic is the most commonly used statistical distance for measuring the deviation between skewed cumulative distributions [33], and it has been applied in [22], [33], [42], [73], [82]. Formally, the KS statistic between FitSet and RealSet is given as

$$D_{KS} = \max_{1 \leq r \leq N} |P_r - \hat{P}_r|, \tag{1}$$

where $D_{KS} \in [0, 1]$, and a smaller value indicates a more accurate distribution model. Essentially, the KS statistic is the limit of the following generalized normalized $\|D\|_p$ metric

$$\|D\|_p = \Big(\frac{1}{N} \sum_{r=1}^{N} |P_r - \hat{P}_r|^p \Big)^{\frac{1}{p}}, \tag{2}$$

When $p \to \infty$, $\|D\|_\infty$ is reduced to the KS statistic (see Appendix A), and when p=2, $\|D\|_2$ is the *squared mean* of CDF deviation. Hence, mainstream distance metrics can be broadly categorized into finite and infinite $\|D\|_p$ groups.

C. Fitting methods for password distributions

Maximum likelihood. In 2009, Clauset et al. [33] suggested using the maximum likelihood (ML) to fit general power-law distributions in the frequency-frequency coordinate system. In this case, the PDF of power-law has $p(f) \propto f^{-\alpha_0}$, where f is the frequency and $p(f)$ is the PDF of the element occurring f times. Their method first excludes elements with frequencies smaller than $f_{\min}$, then fits the rest in an approximated continuous model, and obtains $\alpha_0 = 1 + \frac{N_{\min}}{\sum_{i=1}^{N_{\min}} \log \frac{f_i}{f_{\min}-1/2}}$, where $N_{\min}$ is the number of elements with $f_i \geq f_{\min}$. We can then derive $s \approx \frac{2-\alpha_0}{1-\alpha_0}$ from α_0 (see details in Appendix A).

Least linear square (LLS) treats the CDF of passwords as $P_r = C \cdot r^s$ (C and s are parameters [73]), i.e., $\log P_r = s \cdot \log r + \log C$, which is a straight line under the log-log scale. In this method, *fitted C and s values explicitly depend on log-ranks and log-CDFs of data in the real-world dataset* RealSet. At IEEE S&P'18, Blocki et al. [22] used the LLS fitting method to find C and s, and resulted in a line with real-world CDF data in adjacency of it (see details in Appendix A).

Golden section search. Wang et al. [73] employed golden section search (GSS) to better fit password distributions. Essentially, it searches for the parameter values that can minimize *the KS statistic* between the fitted dataset FitSet sampled from the corresponding CDF-Zipf distribution (characterized by the searched C and s) and the RealSet (see Alg. 4 in Appendix A), i.e., $\min\limits_{(C,s)} D_{KS}$. Here, frequencies in FitSet are positive integers and ranked in descending order, so FitSet is represented numerically by ranked frequencies. Other metrics like normalized $\|D\|_p$ in Eq. 2 can also be integrated with GSS by taking $\min\limits_{(C,s)} \|D\|_p$. We will explore this in Sec. III.

D. System model

We mainly focus on offline password attacks, where an attacker downloads and guesses passwords *without interacting with the authentication server*. We describe them as follows.

Password authentication system. Storing passwords in the salted-hash form $(U, H(PW_U||Salt_U), Salt_U)$ is the standard practice of keeping passwords safe on the server. $Salt_U$ (the user U's salt) is a short random string (at least 32 bits, as recommended by NIST SP 800-63B-4 [67]), and $H(\cdot)$ is a collision-resistant hash that can either be a slow hash (e.g., PBKDF2 [45]) or a memory hard function (MHF, e.g., Argon2i [17]). When the user U logs in with PW', the server compares the salted hash of the user's input with the one stored on the server: $H(PW'||Salt_U) = H(PW_U||Salt_U)$ means $PW' = PW_U$, and the user U can log in. As a result, for an attacker who has obtained the salted hash file, she needs to hash guesses $PW_1, PW_2, \ldots$ (generated by guessing algorithms like Markov [52] and FLA [54]) with each user's salt $Salt_U$, and compares whether $H(PW_i||Salt_U) \stackrel{?}{=} H(PW_U||Salt_U)$.

Based on whether an attacker can generate a sufficiently accurate guessing dictionary, we explore two types of attacker models. The first is the widely-adopted (e.g., [15], [22], [35], [65]) idealized $\mathcal{A}_1$ model, where an attacker generates a 100% accurate dictionary, and only does not know which password is chosen by a particular user. This model is useful, as it can be applied to PINs comprised of a fixed number (e.g., 4 and 6) of digits. The second $\mathcal{A}_2$ model is more practical, as it incorporates algorithmically generated guessing dictionaries (e.g., Markov [52], FLA [54], and PCFG [80]): Not all guesses can match a salted hash in the target dataset, and not all salted hashes can be recovered. We will explore them in Sec. V.

III. Evaluation Criteria of Fitting Methods for Password Distributions

We propose accuracy, completeness, and stability criteria to systematically evaluate the mainstream ML [33], LLS [22], and GSS [73] fitting methods for the widely used CDF-Zipf password distribution model [73]. We reveal that *only* GSS satisfies *all* three criteria under the KS statistic. To our knowledge, we are the first to address this fundamental issue (i.e., **RQ1**). We mainly focus on the CDF-Zipf model [73], and others (e.g., stretched-exponential [42]) are similar.

A. Accuracy evaluation

Accuracy is the most fundamental criterion to determine whether fitted models can well characterize real-world password distributions. We explore both KS statistics used in prior studies (see [22], [33], [42], [73], [82]) and the general normalized metrics represented by $\|D\|_2$ (see their formulas in Sec. II-B), and find that GSS outperforms ML and LLS.

For the KS statistic (i.e., $p \to \infty$, see Eq. 1), we first compare GSS [73] with ML [33]. Particularly, we replicate the experiments[2], and the results validate Wang et al.'s claim that GSS outperforms ML in terms of accuracy (see Table II) [73]. We next compare GSS with LLS, and find that Blocki et al.'s claim that LLS outperforms GSS in accuracy [22] is invalid. For instance, even using their fitting results C=0.0211 and s=0.2166 [22] of the examined Yahoo dataset as the input of LLS, the output D_{KS}=0.0978 is *nine times* larger than D_{KS}=0.0094 in [22]. Three members of our team have independently implemented LLS as Blocki et al. [22] instruct, and get the same discrepancy. We guess that such discrepancy comes from errors in their programming code [22]. *We have reported this issue to the authors, and they have confirmed our guess* and said "I made a mistake somewhere in my previous calculations. Seeing that you guys have the same numbers as me now convinces me...". Besides, in Table II, none of the D_{KS} values from LLS are smaller than GSS. Hence, we are confident that GSS leads to smaller KS statistics than LLS.

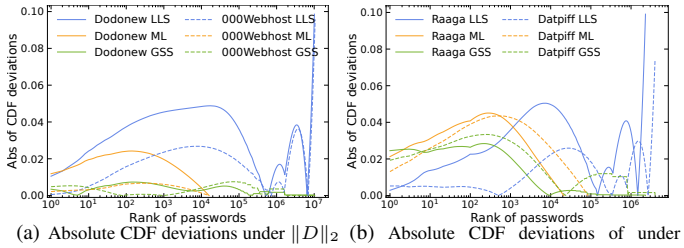
For normalized $\|D\|_p \in [0,1]$ with finite $p \in \mathbb{Z}^+$, GSS is also more accurate. We use $\|D\|_2 = \left(\frac{1}{N}\sum_{r=1}^{N}(P_r - \hat{P}_r)^2\right)^{\frac{1}{2}}$ as a typical example. More specifically, since ML [33] and

[2] Both we and Wang et al. [73] use the codes at https://pypi.org/project/powerlaw/1.5/ given in [33] to fit power law on password datasets.

TABLE II: Fitting results of the CDF-Zipf password distribution model under ML [33], LLS [22] and GSS [73] fitting methods.

Dataset	ML parameters†					LLS parameters				GSS parameters					
	Coverage	D_{KS}	$\|D\|_2$§	$f_{\min}$	s	D_{KS}	$\|D\|_2$§	C	s	D_{KS}	C	s	$\|D\|_2$§	C	s
Yahoo	45.83%	0.1568	0.0477	5	0.0374	0.0871	0.0411	0.0193	0.2225	0.0408	0.0332	0.1809	0.0146	0.0397	0.1666
Dodonew	15.15%	0.0245	0.0094	36	0.2853	0.0986	0.0439	0.0040	0.3365	0.0048	0.0193	0.2123	0.0009	0.0180	0.2174
000Webhost	8.75%	0.0078	0.0031	23	0.2975	0.0958	0.0411	0.0011	0.4145	0.0044	0.0057	0.2827	0.0015	0.0065	0.2741
Rockyou	49.60%	0.1440	0.0402	5	0.0576	0.0713	0.0328	0.0303	0.2078	0.0446	0.0381	0.1861	0.0144	0.0496	0.1656
Tianya	50.37%	0.0690	0.0161	5	0.0905	0.0669	0.0305	0.0359	0.1990	0.0226	0.0621	0.1556	0.0019	0.0882	0.1310
Duelingnetwork	18.74%	0.0137	0.0032	15	0.2464	0.0721	0.0300	0.0056	0.3373	0.0078	0.0135	0.2587	0.0014	0.0173	0.2406
Chegg	17.47%	0.0170	0.0065	6	0.2077	0.1317	0.0604	0.0007	0.4112	0.0081	0.0082	0.2360	0.0005	0.0114	0.2138
Wishbone	21.71%	0.0179	0.0052	16	0.2142	0.0918	0.0408	0.0066	0.3154	0.0152	0.0183	0.2294	0.0010	0.0261	0.2035
Dailyquiz	30.87%	0.0390	0.0146	10	0.2205	0.0455	0.0179	0.0102	0.2906	0.0215	0.0145	0.2554	0.0107	0.0178	0.2400
Datpiff	30.55%	0.0441	0.0161	7	0.1908	0.0743	0.0319	0.0088	0.3064	0.0211	0.0169	0.2464	0.0051	0.0228	0.2233
Raaga	25.59%	0.0453	0.0163	10	0.1195	0.0991	0.0449	0.0099	0.3081	0.0160	0.0288	0.2118	0.0015	0.0370	0.1917

† ML fits passwords with frequencies $f_i \geq f_{\min}$ in the frequency-frequency coordinate system to find α_0, and converts it to $s \approx \frac{2-\alpha_0}{1-\alpha_0}$ (see details at Appendix A). We omit C since it depends on s. We compute the KS statistic D_{KS} in the rank-frequency system common in password studies [22], [73], [77], [78], [80].
‡ The value **0.0094** taken verbatim from Blocki et al.'s work [22] is flawed and the correct D_{KS}=0.0871 from our implementation of LLS is *8 times* theirs.
§ $\|D\|_2$ denotes the normalized 2-norm metric given by Eq. 2. Since ML and LLS do not incorporate concrete distance metrics, their fitted parameters will not change when $\|D\|_2$ distance is used. For GSS, new C and s values are obtained if $\|D\|_2$ is used as the new optimization goal (see Line 14 of Alg. 4 in Appendix A).



(a) Absolute CDF deviations under $\|D\|_2$ for Dodonew and 000Webhost datasets.

(b) Absolute CDF deviations of under $\|D\|_2$ for Datpiff and Raaga datasets.

Fig. 1: Absolute CDF deviations of ML [33], LLS [22] and GSS [73] when GSS is fitted under $\|D\|_2=\left(\frac{1}{N}\sum_{i=1}^{N}(P_r-\hat{P}_r)^2\right)^{\frac{1}{2}}$ (see Eq. 2). For most passwords, GSS outperforms ML and LLS in terms of accuracy.

LLS [22] involve no concrete metric, their fitted parameters will not change, and we can compute $\|D\|_2$ via C and s values in Table II. For GSS, we change the goal from minimizing D_{KS} to $\|D\|_2$ in Alg. 4 and leave other steps unchanged.

The absolute CDF deviations are shown in Fig 1. We find that $\|D\|_2$ obtained from GSS ranges from 0.0005∼0.0146, which are 26.74%∼94.62% and 40.28%∼99.11% smaller than those from ML and LLS, respectively. These results are reasonable because: (1) ML aims to find the most likely power-law distribution on the *frequency-frequency coordinate system* with the observed data [33], which differs from the GSS's goal of minimizing CDF deviations on the *rank-frequency one*; (2) LLS is equivalent to minimizing the normalized $\|\log \tilde{D}_2\|=(\sum_{i=1}^{N}(\log P_r - \log \hat{P}_r)^2)^{\frac{1}{2}}$ under the log-log scale, rather than our investigated normalized $\|D\|_2$ under the normal scale. We also note that the metric needs to be normalized into $[0,1]$, or the CDF deviation can be as large as 0.80.

B. Completeness evaluation

Beyond accuracy, completeness evaluates the reliability of a fitted model by measuring how closely fitted parameters approach the optimum (usually unknown) under a given model. As an intrinsic property, completeness means that parameters obtained from fitting a real-world dataset and its corresponding fitted model are statistically indistinguishable. Otherwise, a password dataset may be more accurately characterized by parameter values other than the fitted ones.

We use the CDF-Zipf model as an example. In practice, practitioners often do not know C and s values in prior, and a fitting method (e.g., ML, LLS or GSS) only gives results without telling if they are correct. Ideally, (C', s') obtained from fitting the fitted dataset FitSet should be *sufficiently close* to the original (C, s) from fitting the real-world dataset RealSet, i.e., $C' \approx C$ *and* $s' \approx s$, and completeness is evaluated by this measurement. As an intrinsic property independent of whether C and s are known, $(C', s') \approx (C, s)$ also means $(C, s) \approx (C^*, s^*)$ for the unknown optimum (C^*, s^*). Otherwise, if completeness does not hold, even if a fitting method gives (C, s), the dataset might be characterized by substantially different $(\tilde{C}, \tilde{s})$, making the fitted C and s unreliable.

As revealed by Clauset et al. [33], ML holds completeness. Our experiments on the eight large-scale datasets also validate this, with the relative difference of the parameter s being acceptably low as 0.09%∼9.10%. Hence, we mainly focus on LLS and GSS. We develop rigorous mathematical theories and employ extensive experiments to demonstrate that GSS holds completeness, and explain why LLS does not.

Theoretical analysis. Theorem 1 lays the foundation of the GSS completeness. We provide detailed proofs under the KS statistic (see Appendix B), while others remain open questions. First, we leverage Chebyshev Alternation Theorem [61] that is only applicable for the KS statistic to prove the existence of the optimal parameters (see Lemmas 1 and 2), then show that a golden search converges to the optimum (see Lemma 3). We next incorporate the effects of random samplings, and show that a fitted model characterized by any given (C, s) (i.e., the known optimum) leads to statistically indistinguishable empirical means in GSS fittings (see Theorem 1), i.e., $\overline{C'} \approx \overline{C}$ and $\overline{s'} \approx \overline{s}$. Since the standard deviations are usually small (e.g., about 10^{-4} [42]), it is natural to derive that $C' \approx C$ and $s' \approx s$.

Theorem 1. *Suppose* (C, s) *and* (C', s') *are obtained from using GSS to fit the real-world* RealSet *and the fitted model* FitSet *characterized by* (C, s). *For any* $\epsilon > 0$, *in the GSS fitting (see Alg. 4 in Appendix A), there exist* $\lim_{m,n\to\infty} \Pr(|\overline{C'} - \overline{C}| < \epsilon) = 1$ *and* $\lim_{m,n\to\infty} \Pr(|\overline{s'} - \overline{s}| < \epsilon) = 1$ *for a sufficiently large iteration number* m, *where* $(\overline{C'}, \overline{s'})$ *and* $(\overline{C}, \overline{s})$ *are empirical means of* n *trials.*

Proof. Based on Lemmas 2 and 3 in Appendix B, for each $(C(\{u\}), s(\{u\}))$ obtained from fitting RealSet under the set of random seed $\{u_i\}_{i=1}^{|D|}$, since $(C(\{u\}), s(\{u\}))$ is the optimal minimizer, there are $\lim_{m\to\infty} C'_m(\{\omega\}, \{u\}) = C(\omega)$ and $\lim_{m\to\infty} s'_m(\{\omega\}, \{u\}) = s(\{\omega\})$, where $C'_m(\{\omega\})$ and $s'_m(\{\omega\})$ denote C' and s' values in the m-th iteration under another fixed set of random seed $\{\omega_i\}_{i=1}^{|D|}$ when approximating $(C(\{u\}), s(\{u\}))$. By taking expectation on both sides, we

TABLE III: Completeness evaluation results of LLS and GSS.‡

Dataset	LLS completeness				GSS completeness			
	C'	s'	$RD(C)$	$RD(s)$	C'	s'	$RD(C)$	$RD(s)$
Yahoo	0.0118	0.2602	**39.04%**	**16.94%**	0.0332	0.1809	**0.06%**	**0.02%**
Dodonew	0.0024	0.3820	**39.81%**	**13.52%**	0.0193	0.2121	**0.20%**	**0.06%**
000Webhost	0.0006	0.4599	**37.29%**	**10.96%**	0.0057	0.2824	**0.15%**	**0.04%**
Rockyou	0.0202	0.2406	**33.25%**	**15.82%**	0.0381	0.1861	**0.06%**	**0.02%**
Tianya	0.0246	0.2302	**31.57%**	**15.59%**	0.0621	0.1555	**0.08%**	**0.04%**
Duelingnetwork	0.0038	0.3768	**32.74%**	**11.71%**	0.0133	0.2595	**0.36%**	**0.09%**
Chegg	0.0004	0.4644	**45.71%**	**12.92%**	0.0082	0.2360	**0.18%**	**0.05%**
Wishbone	0.0041	0.3587	**37.37%**	**13.74%**	0.0183	0.2292	**0.28%**	**0.08%**
Dailyquiz	0.0072	0.3224	**29.09%**	**10.93%**	0.0146	0.2554	**0.17%**	**0.04%**
Datpiff	0.0059	0.3454	**33.29%**	**12.73%**	0.0169	0.2464	**0.17%**	**0.05%**
Raaga	0.0063	0.3530	**35.91%**	**14.56%**	0.0286	0.2126	**0.30%**	**0.09%**

‡ RD means the relative difference between the predetermined C and s (parameter values obtained from fitting RealSet), and C' and s' obtained from fitting FitSet.

can obtain $\mathbb{E}[\lim_{m\to\infty} C'_m]$=$\mathbb{E}[C]$ and $\mathbb{E}[\lim_{m\to\infty} s'_m]$=$\mathbb{E}[s]$, and thus the following based on the law of large numbers,

$$\lim_{m,n\to\infty} \Pr(|\overline{C'}-\overline{C}|<\epsilon) = \Pr(|\overline{C'}-\mathbb{E}[C']+\mathbb{E}[C]-\overline{C}|<\epsilon)$$
$$\geq \lim_{m,n\to\infty} \Pr(|\overline{C'}-\mathbb{E}[C']|<\frac{\epsilon}{2}) + \Pr(|\mathbb{E}[C]-\overline{C}|<\frac{\epsilon}{2}) - 1 = 1, \quad (3)$$

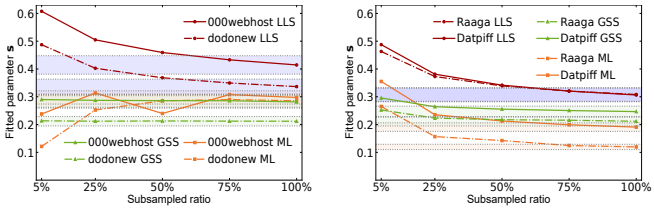
where the subscript m is omitted for conciseness. For the parameter s, the case is similar. This concludes the proof. □

In contrast, LLS lacks completeness since its log-log scale enlarges small deviations of popular passwords and produces large deviations for unpopular ones. More concretely, deviations in FitSet from P_r=$C\cdot r^s$ will propagate to C' and s' since they are determined by $\log r$ and $\log P_r$ (see Eq. 16 in Appendix A) of FitSet. To show this, we suppose the fitted CDF as P'_r=$C\cdot r^s$+ϵ_r with ϵ_r as the deviation, use the Taylor series $\log P'_r$–$\log C\cdot r^s$=$\frac{\epsilon_r}{C\cdot r^s}$–$\frac{\epsilon_r^2}{2(C\cdot r^s)^2}$+$O(\epsilon_r^3)$. Hence, for popular passwords (i.e., $C\cdot r^s\approx$0), the log difference is enlarged by the log-log scale; For unpopular passwords, ϵ_r is often large (e.g., $\epsilon_r\in[0.1231, 0.2455]$ when $P_r\approx$1). These make (C', s') obtained from fitting FitSet different from the original (C, s). More details are shown in Appendix B.

Experiments. We use the data in Table II as the predetermined C and s, and explore completeness under KS statistic and normalized D_2 metrics. For example, in the Yahoo-based experiment under the KS statistic, we set C=0.0193, s=0.2225 for LLS, and C=0.0332, s=0.1809 for GSS, then generate FitSet characterized by C and s via LLS and GSS (see Algs. 3 and 4 in Appendix A). We next fit FitSet with the same fitting method to obtain C' and s', and measure the relative differences $RD(C)$=$|C'-C|/C$, and $RD(s)$=$|s'-s|/s$. If both $RD(C)$ and $RD(s)$ are small (e.g., <1%), the fitting method is considered to be complete (i.e., holds completeness).

Table III shows that GSS holds completeness but LLS does *not* under the KS statistic, validating our theoretical analysis. More specifically, the maximum $RD(C)$ and $RD(s)$ from GSS are 0.36%. In contrast, both $RD(C)$ and $RD(s)$ from LLS exceed 10%, and even the two smallest RD from LLS ($RD(s)$=10.93% of Dailyquiz and 10.96% of 000Webhost) are over 30 times the largest GSS ones. Similarly, when GSS aims to minimize $\|D\|_2$ in Eq. 2, $RD(C)$ and $RD(s)$ are 0.14%∼0.89% and 0.04%∼0.34%, respectively, strengthening GSS's completeness (see more details in Appendix B).

Remark. Completeness essentially differs from accuracy. First, for CDF-Zipf [73], ML largely holds completeness but not accuracy (see Table II). Second, for other models with varied accuracy, GSS holds completeness while LLS does not. For instance, on the one hand, for the inaccurate exponential model (whose KS statistic is 0.1135∼0.3892) [42], the relative differences of α under LLS and GSS are 37.76%∼69.94% and 0.01%∼0.07%, respectively. On the other hand, for the more accurate stretched-exponential model [42], whose KS statistic is 38.65%∼66.70% lower than that of CDF-Zipf on our eight out of eleven datasets, GSS is complete while LLS is not.



(a) s stability of Dodonew & 000webhost. (b) s stability of Datpiff & Raaga.

Fig. 2: Stability of the parameter s fitted on subsets with ML [33], LLS [22] and GSS [73] under the KS statistic. Stabilized s (with relative difference about 8%) are in the shadow area (orange for ML, blue for LLS, and green for GSS).

C. Stability evaluation

Stability measures the difference between fitted parameter values of the entire and subsampled datasets. This criterion determines *to what extent fitting results of one dataset can be applied to the same dataset of different sizes.* Since many password files have been breached more than once (e.g., [46], [62]), and password datasets of the same language, service type, and creation policy are distributed similarly [73], [76], stability is of practical importance. We compare the stability of the mainstream ML [33], LLS [22], and GSS [73] fitting methods. To do this, we randomly subsample (without replacement) the entire dataset of ratios 5%, 25%, 50%, 75%, and 100%, and fit them under the KS statistic metric. We mainly focus on the parameter s, while C is similar: *If the relative difference between values obtained from fitting subsampled and the entire dataset is small (e.g., no more than about 8%), a fitting method is regarded stabilized.* In this case, the fitted model characterized by parameters obtained from fitting a subsampled dataset can still maintain relatively high accuracy.

Results in Fig. 2 show that GSS performs the best. Specifically, if the KS statistic of the original dataset has $D_{KS}\leq$0.01 (e.g., Dodonew and 000Webhost), GSS can largely stabilize even when the subsample ratio is 5%, and GSS can stabilize when the ratio is 50%. In particular, when using such C and s values to characterize the original dataset, D_{KS} ranges from 0.0049∼0.0788 (resp. 0.1023∼0.2296) when GSS is (resp. is not) stabilized. In contrast, LLS stabilizes only at a 75% subsample ratio, but the KS statistic still reaches 0.0546∼0.1039. For ML, the relative difference varies substantially, from 0.87%∼42.92% at the 75% subsample ratio to 17.64%∼402.95% at 5%. This

TABLE IV: Evaluation criteria of ML [33], LLS [22] and GSS [73] under the KS statistic metric.

Method‡ \ Criterion†	Accuracy	Completeness	Stability
ML [33]	×	✓	×
LLS [22]	×	×	×
GSS [73]	✓	✓	✓

† Criterion represents the evaluation criteria, i.e., accuracy, completeness, and stability.
‡ Method represents the fitting methods, i.e., Clauset et al.'s ML [33], Blocki et al.'s LLS [22] and Wang et al.'s GSS [73].

invalidates Blocki et al.'s claim [22] that LLS would stabilize on large datasets, and explains why subsampling datasets based on fixed sizes (e.g., one million) is undesirable.

Summary of fitting methods. Evaluation results on accuracy, completeness, and stability reveal that only Wang et al.'s GSS [73] can satisfy *all* three criteria (see Table IV) under the KS statistic. *Based on these desirable properties, particularly completeness that can be proved under the KS statistic, unless stated otherwise, we mainly use the GSS-fitted CDF-Zipf model and the KS statistic.* We also note that the GSS-fitted results in Tables II and III are slightly different from those in [42]. This is expected because statistical randomness and enhanced search (e.g., more iterations) can affect the fitted results.

D. Utilization of GSS-fitted CDF-Zipf model

We now explore how the GSS-fitted CDF-Zipf model should be applied, which is critical for over 300 password-based applications using Zipf models (see them at https://tinyurl.com/2kpxwjsx). A natural way is to use the fitted FitSet. This is reasonable, since the dataset size only needs to exceed 10^6 (which holds for our datasets in Table I) to make the statistical randomness sufficiently small [42], and thus keep the GSS-fitted CDF-Zipf model (i.e., FitSet) accurate.

As an explicit expression can further facilitate security analysis, one natural idea is using the explicit P_r=$C{\cdot}r^s$. However, due to the sampling nature and the rank operation in GSS (see Algs. 3 and 4 in Appendix A), this expression may be undesirable for unpopular passwords (e.g., with frequency$\leq$5): The number of unique passwords n=$\lceil(\frac{1}{C})^{\frac{1}{s}}\rceil$ is much larger than N of the real-world RealSet (e.g., n=$4.68N$ in Yahoo). To address this issue, *we find to what extent the explicit expression P_r=$C{\cdot}r^s$ can well approximate the fitted distribution*.

We compare the KS statistic between P_r=$C{\cdot}r^s$ (with the set being ExplSet) and the CDF of RealSet, with that between P_r=$C{\cdot}r^s$ and the CDF of FitSet. We find that first, for the top-10^5 passwords, D_{KS}^{ER} ranges from 0.0044 to 0.0446, so using P_r=$C{\cdot}r^s$ is acceptably accurate. Besides, the KS statistic (denoted as D_{KS}^{EF}) between ExplSet and FitSet is as small as 0.0008$\sim$0.0166. Second, for larger ranks (e.g., top-10^6), both D_{KS}^{ER} and D_{KS}^{EF} are much larger and can be close and exceed 0.1, so using P_r=$C{\cdot}r^s$ may be inaccurate. Thus, it is suitable to use P_r=$C{\cdot}r^s$ for about the top-10^5 passwords.

Comparison with an alternative. We discuss the pros and cons of using the parametric GSS-fitted CDF-Zipf model [73] and Blocki-Liu's non-parametric method [23]. Conceptually, the CDF-Zipf model [73] characterizes a password distribution with C and s, enabling concise usage for downstream applications. In comparison, Blocki–Liu's method [23] computes per-rank high-confidence CDF bounds, yielding statistical guarantees via concentration inequalities and linear programming, which might be a drawback in terms of ease of use.

Further, the computational gap is also substantial. Fitting a CDF-Zipf model is more efficient than computing statistical bounds. Our preliminary experiments show that GSS can fit the Yahoo dataset in about 22 mins; In comparison, Blocki–Liu's method [23] is more computationally intensive and needs about 22 hours to solve linear programming for 100 different ranks. Hence, the GSS-fitted CDF-Zipf model is more suitable for general security analysis of password-based applications, while Blocki–Liu's method is more suitable for tasks requiring rigorous statistics, e.g., measuring subtle differences between male and female chosen passwords (see Sec. V-B of [23]).

IV. Framework and Measurements of Memory Hard Functions (MHFs)

Besides distributions, password hashing costs also impact offline guessing attacks. To answer **RQ2**, we propose a generic cost framework for memory hard functions (MHFs) and, *for the first time*, validate it with real-world executions. Our framework mainly focuses on direct MHF execution costs, enabling both attackers and security administrators to more readily estimate hashing costs in real-world scenarios.

A. Cost frameworks

MHFs. We propose a MHF cost framework built on the leading Argon2i [17] (the winner of Password Hashing Competition [1]). Our cost framework is generic, and can be extended to other MHFs (e.g., [14], [21]) with similar configurations.

A MHF is modeled as a directed acyclic graph that has τ iteration(s) of memory, and each contains σ memory units (measured in KB, e.g., σ=1GB=2^{20}KB). For a user, an Argon2i execution involves two processes. First, in each of the total σ steps, Argon2i sequentially computes Blake2b (whose cost is C_{H_C}), temporarily places the output in the cache, and writes it into one *empty* memory unit (whose cost is C'_{H_B}). Second, Argon2i iterates the above process for τ iteration(s). At each step of an iteration, Argon2i firstly extracts the memory data to cache (whose cost is C''_{H_B}), computes Blake2b with extracted data as the input, and overwrites the output into one memory unit. In this design, four types of energy costs exist: (1) $C_{H_C}(\tau{+}1)\sigma$ computation cost; (2) $C'_{H_B}(\tau{+}1)\sigma$ busy memory cost from writing Blake2b data from the cache into memory; (3) $C''_{H_B}\tau\sigma$ busy memory cost from extracting memory data to cache; (4) $C_{H_I}(\sum_{i=1}^{\sigma} i{+}\tau\sigma^2)$ idle memory cost from holding static memory data. Generally, $C'_{H_B}{\approx}C''_{H_B}$, i.e., the energy cost is the same for extracting and writing data from/into memory, so we can simplify as C'_{H_B}=C''_{H_B}=$\frac{1}{2}C_{H_B}$. As a summation of cost (1)$\sim$(4), the Argon2i energy cost is

$$k_E \approx \Big(C_{H_C}(\tau+1) + C_{H_B}\frac{2\tau+1}{2}\Big)\sigma + C_{H_I}\frac{2\tau+1}{2}\sigma^2. \quad (4)$$

This reveals that in practical usages when the σ and τ are given, how the energy cost k_E scales with σ depends on C_{H_B}, C_{H_C}, and C_{H_I}. However, many prior studies (e.g., [13], [14], [22]) implicitly assumed that $C_{H_B}{\approx}C_{H_I}$, and claimed that k_E scales quadratically with σ in almost all cases. We will use concrete empirical evidence to invalidate this in Sec. IV-B.

An attacker should also consider how memory purchase cost (whose unit is C_{H_P}) contributes to the MHF cost via area-time complexity essential to memory hardness (see [12], [13], [14], [18], [21]). More specifically, well-designed attack algorithms can reduce memory hardness through more efficient memory

TABLE V: Argon2i energy cost on our Ubuntu 22.04 computer with energy measured in Joules (J).†

Memory iteration τ	Memory usage σ=1GB		Memory usage σ=2GB		Memory usage σ=4GB		Memory usage σ=8GB		Memory usage σ=16GB		Memory usage σ=32GB	
	Memory cost (busy+idle)	Computation cost	Memory cost (busy+idle)	Computation cost	Memory cost (busy+idle)	Computation cost	Memory cost (busy+idle)	Computation cost	Memory cost (busy+idle)	Computation cost	Memory cost (busy+idle)	Computation cost
1	2.86 —	93.34 —	5.36 —	193.60 —	11.64 —	400.32 —	24.88 —	808.32 —	46.96 —	1682.10 —	97.14 —	3421.19 —
2	4.70 1.84	139.80 46.46	9.05 3.69	289.27 95.67	18.68 7.04	593.27 192.94	41.06 16.18	1218.63 410.32	78.45 31.50	2534.16 852.06	158.19 61.05	5231.38 1810.19
3	6.50 1.80	186.63 46.83	12.83 3.78	385.64 96.37	25.94 7.26	786.75 193.48	58.02 16.96	1626.62 407.99	110.05 31.59	3405.70 871.54	224.87 66.68	7099.46 1868.08
4	8.49 1.99	233.32 46.69	16.53 3.70	480.95 95.31	33.22 7.28	978.45 191.71	75.94 17.93	2033.57 406.95	145.33 35.28	4239.66 833.96	294.44 69.58	8984.21 1884.75
5	10.36 1.87	280.26 46.94	20.18 3.64	576.48 95.53	40.76 7.54	1177.74 199.29	92.34 16.40	2445.95 412.38	177.12 31.79	5135.29 895.63	355.76 61.32	10708.50 1724.29

† The energy costs (measured in Joules) of memory and computation. The second sub-column in each memory cost (resp. computation cost) records the cost of each iteration. These values are obtained by subtracting the cost values of τ from those of $\tau+1$. It is clear that the energy cost mostly scales linearly with both the memory iteration τ and memory usage σ.

usage. This cost scales quadratically with σ, and linearly with τ (see Appendix D) [21], so an attacker's cost is

$$k{=}k_E+\frac{k_P}{Aq}{=}\Big(C_{H_C}(\tau+1)+C_{H_B}(\frac{2\tau+1}{2})\Big)\sigma+\Big(C_{H_I}+\frac{C_{H_P}}{Aq}\Big)\frac{2\tau+1}{2}\sigma^2, \quad (5)$$

where A_q denotes the MHF attack quality under pebbling attacks, measuring the memory-hardness reduction achievable by well-designed parallel implementations/attacks (e.g., $A_q{\approx}5$ [13], [14], [21]). Unlike C_{H_B}, which is relatively stable across architectures and resistant to GPU/ASIC reduction [14], [17], [60], C_{H_C} can be further reduced (see Sec. IV-C).

Slow hashes. We use PBKDF2 [45] (with SHA256 as the round function) as an example, so the energy cost is $k_E{=}C_{H_C}{\cdot}\tau$, where τ is the iteration number (e.g., $\tau{=}10^7$), and C_{H_C} is the computation cost of the round function.

B. MHF memory cost measurements

Memory energy cost. We now validate our generic MHF cost framework and measure the real-world memory energy costs via Argon2i executions. *To the best of our knowledge, only Blocki et al. [22] and Ren-Devadas [64] have explicitly measured memory energy costs, but they are both flawed. On the one hand, Blocki et al. [22] did not distinguish between busy memory cost and idle memory cost, i.e., assuming* $C_{H_B}{\approx}C_{H_I}$. They also used the flawed computation cost $C_{H_C}{=}\$7.00{\times}10^{-15}$ from rough observations of the Bitcoin network in 2013, and assumed that the cost was equal to the reward [27]. However, Bitcoin prices are mainly determined by supply and demand, and have little relation with its cost. On the other hand, Ren and Devadas [64] conjectured $C_{H_B}{\gg}C_{H_I}$, but have provided no empirical evidence from real-world MHF executions. These inconsistencies highlight the need for a systematic measurement of the memory cost in real-world MHF executions. *As far as we know, we are the first to deal with this long-standing dispute with concrete evidence.*

Measurement setups. We use the Argon2i code shared by the authors of [17] (see https://github.com/P-H-C/phc-winner-argon2) and mainly use Ubuntu 22.04 (with Intel Xeon(R) Gold 6230R CPU and 256GB DDR4). Particularly, we select Intel PCM [43] to monitor energy consumption (in Joules) for real-time measurements inside CPUs and caches. *Given that single-thread execution is more concise and more user-friendly (e.g., no need to slice memory into lanes [17]), we run Argon2i 100 times in single-thread mode for each parameter configuration and take the average as the results.*

We close all unrelated programs and only execute Argon2i to minimize the impacts of irrelevant factors. To further rule out the impacts of the operating system, for each fixed σ (e.g., σ=1GB), we measure the busy and idle memory costs under memory iterations $\tau{\in}\{1,...,5\}$. We then subtract the energy costs of τ+1 by τ (e.g., τ=3 and 2) to ensure that the energy costs are mainly from Argon2i. Next, we measure the memory cost under memory usages $\sigma{\in}\{1\text{GB},\ldots,16\text{GB},32\text{GB}\}$.

Memory energy cost results are shown in Table V: They confirm our energy cost framework of Eq. 4, and reveal that the overall memory cost (busy and idle) scales linearly with the memory usage σ, not quadratically as in [12], [13], [22]. This invalidates the prior cost framework $k{=}k_E{=}C_{H_C}(\frac{\sigma^2}{3000}+\sigma)$ [22], suggesting that security for MHFs relying on quadratic cost assumptions (e.g., [12], [13], [16]) is overestimated.

More specifically, firstly, *the memory cost scales almost linearly with* σ *for each fixed* τ. For example, the cost at σ=16GB is 16.98∼17.54 times the cost at σ=1GB, which reveals that the busy memory cost is significantly larger than the idle one as Ren-Devadas's conjecture [64], i.e., $C_{H_B}{\gg}C_{H_I}$ instead of $C_{H_B}{\approx}C_{H_I}$ that leads to quadratic costs. Secondly, the computation cost scales linearly with σ, and is over 20 times the memory cost. Thirdly, for a fixed memory usage σ, the memory energy cost in the first memory iteration (i.e., τ=1) is about 1.5 times that in other iterations. Hence, the energy cost of filling an empty memory unit is one-half of extracting and writing memory data, substantiating $C'_{H_B}{=}C''_{H_B}{=}\frac{1}{2}C_{H_B}$.

To further quantify the C_{H_B} and C_{H_I} values, we build a system of linear equations (with six equations and two variables) based on Eq. 4 and our results in Table V:

$$\begin{cases} 3.5(2^{20}C_{H_B}+2^{40}C_{H_I})=6.50\text{J}, \\ \cdots\cdots\cdots\cdots \\ 3.5(2^{24}C_{H_B}+2^{48}C_{H_I})=110.05\text{J}, \\ 3.5(2^{25}C_{H_B}+2^{50}C_{H_I})=224.87\text{J}. \end{cases} \quad (6)$$

To dwarf the impact of irrelevant factors (e.g., the operating system) that may cause undesirable inaccuracy, we use the last two equations with the largest memory usages to solve C_{H_B} and C_{H_I}. We obtain $C_{H_B}{=}1.83{\times}10^{-6}$J and $C_{H_I}{=}2.42{\times}10^{-15}$J, where C_{H_B} is about 3.57 times the state-of-the-art Ren-Devadas's 0.5nJ/Byte (i.e., $5.12{\times}10^{-7}$J) obtained from writing/extracting a SHA256 data into/from the memory units (not actual executions of MHFs) [64]. Further, for the recommended τ=3 and σ=1GB setting [17], the busy memory cost 6.72J is over 700 times the idle cost 0.0093J. In this case, the memory usage in an Argon2i execution needs to be *128GB per MHF* to make the busy and idle costs comparable. This substantiates Ren-Devadas's conjecture that the busy memory cost is orders of magnitude larger than the idle one [64], so the idle memory cost can be omitted.

Memory purchase cost. We consider the memory purchase cost, and use the typical 32GB Samsung DDR4 units as the benchmark. To transform the energy cost from Joule to U.S. dollar, we use the electricity price of about $\$0.15/Kwh$[3] in

[3]The actual electricity price may change with locations and time.

TABLE VI: Computation costs of SHA256 and Blake2b round functions in PBKDF2 (slow hash) and Argon2i (MHF).†

Round function	Guessing hardware	Power	Price	Hash rate‡	Amortized purchase	Energy	C_{H_C}
SHA256	Bitman S19 Pro [2]	3,250W	$9,130	110TH/s	$1.32e-18	$1.23e-18	**$2.55e-18**
	GeForce RTX 4090§	450W	$1600	21.9GH/s	$1.16e-15	$8.54e-16	**$2.00e-15**
	Bitcoin Miner [27]*	—	—	—	—	$7.00e-15	**$7.00e-15**
Blake2b	StrongU STU-U2 [66]	1,600W	$900	7.0TH/s	$2.04e-18	$9.52e-18	**$1.16e-17**
	GeForce RTX 4090§	450W	$1600	13.1GH/s	$1.94e-15	$1.44e-15	**$3.38e-15**

† C_{H_C} consists of two parts: the energy cost and the amortized computation hardware purchase cost (with the hardware lifespan being 2 years [21]).
§ GeForce RTX 4090 is the GPU on our Ubuntu computer.
‡ H/s means hash per second, 1GH=10^9H, 1TH=10^{12}H, and $ means U.S. dollar.
∗ Blocki et al. [22] used $\$7.00\times10^{-5}$ from the Bitcoin network in 2013 [27].

the United States [5]. Particularly, we amortize the memory purchase cost per 1KB of Blake2b output, and obtain C_{H_P}=$\$4.63\times10^{-21}$ (see details in Appendix D). This shows that to make the purchase cost comparable with the energy cost, the memory usage needs to be $\geq$8GB per MHF for users, causing serious usability issues (i.e., taking about 8s in authentication). Based on this, *we conclude that the total MHF cost should scale linearly, rather than quadratically as claimed in [12], [13], [22] with practical memory usage.*

Limitation. First, our memory cost measurement data still has 3.52%$\sim$12.25% fluctuations. A possible improvement is to use customized external devices such as PowerInsight [49], but their usage requires specialized skills. Second, attackers can execute in multi-thread modes, which may incur extra costs such as memory management and internal communications. Still, we find that linear scaling of MHF energy costs holds in such models. For instance, with 4 threads, memory energy cost increases by 2.04 (resp. 3.82) when σ grows from 1GB to 2GB (resp. 4GB). Despite this, we are the first to empirically investigate the key memory costs of MHFs in their executions.

C. MHF computation cost investigations

In attacks, specialized hardware such as GPUs and even ASICs might be used to accelerate round function computations. While practical GPU and ASIC accelerations of MHFs demand considerable engineering effort (e.g., cache management), heuristic round function designs can be vulnerable to such accelerations, thereby significantly reducing attack costs. Hence, we still investigate energy and purchase costs of state-of-the-art cracking hardware, assuming a two-year hardware duration as in [21]. Results are shown in Table VI.

The data corroborate the MHF design idea that the memory reading/writing operations can mitigate the computation efficiency of ASICs. More concretely, although both GPUs and ASICs can reduce the Blake2b computation cost compared with general-purpose CPUs, an attacker's cost is close, e.g., $\$2.84\times10^{-7}$ for ASIC vs. $\$2.98\times10^{-7}$ for GPU when Argon2i is executed with the recommended τ=3 and σ=1GB [17]. In contrast, when the slow hash PBKDF2 [45] is executed with the maximum 10^7 iterations [27], an attacker's overall cost from using ASIC is only 0.13% of that from using GPU ($\$2.55\times10^{-18}$ vs. $\$2.00\times10^{-15}$). *Thus, when evaluating an attacker's MHF cost, we do not distinguish between GPUs and ASICs, and use the data obtained from GPUs as the baseline.*

Remark. Offline guessing economics rely on these estimates. For example, if attackers can run Argon2i more efficiently on GPUs and ASICs, the total attack cost drops, posing greater security threats. We also note that using GPU-measured Argon2i costs (e.g., via Hashcat [9]) may not capture real-world threats: Such sequential executions can omit pebbling attacks (e.g., [12], [13], [14], [18], [21]) that reduce memory hardness, as well as cost reductions from GPUs and ASICs. Thus, estimates based on GPU-measured costs can overestimate the MHF security, and we instead adopt conservative assumptions with separated memory and computation costs.

V. Analysis of Offline Guessing Economics

Upon answering **RQ1** and **RQ2**, we evaluate their impacts on offline guessing economics to answer **RQ3**. We present our main assumptions about our attacker models. First, we omit the costs of fitting password distributions and other preprocessing costs (see detailed discussions in Sec. V-D), as they are often orders of magnitude smaller than hashing costs. Second, we assume that an attacker faces little competition. This is reasonable because passwords are often kept secret before being well-exploited [56], and attackers often do not know others before cracked datasets are made public.

In this work, we mainly focus on the profit-driven attack scenario as in [15], [19], [22], where a rational attacker $\mathcal{A}$ aims to maximize her profit (i.e., reward minus cost). The reward is obtained by selling cracked passwords/accounts (e.g., v=$1 per account [86]), and the cost mainly stems from hashing guesses. *For the maximum profit, $\mathcal{A}$ needs to devise a strategy before offline guessing attacks to determine when she should stop hashing a new guess before the expected reward is less than the cost.* With some extent of notation abuse, we denote the target dataset (a multiset) as RealSet, and $\mathcal{A}$'s guessing dictionary (a set) as DictSet. We suppose that $p_1\geq p_2\geq\ldots$ are the probability density (mass) distributions of $PW_1, PW_2, \ldots\in$RealSet, and $0<\alpha\leq1$ measures the diminishing returns.

A. Theories for idealized $\mathcal{A}_1$ model

We first explore the idealized $\mathcal{A}_1$ model (see Sec. II-D), where an attacker generates a 100% accurate dictionary, and only does not know which password is chosen by a particular user (see [15], [22], [35], [65]). Since each guess in DictSet matches a salted hash in RealSet, her reward and cost are

$$\begin{aligned} R(r) &= v(\textstyle\sum_{i=1}^{r} p_i)^{\alpha} \\ C(g) &= k\cdot g(1-\textstyle\sum_{i=1}^{g} p_i) + k\textstyle\sum_{i=1}^{g} i\cdot p_i, \end{aligned} \tag{7}$$

where v is the value of a cracked account, and k is the hashing cost (see Sec. IV). Since each guess in DictSet can match at least a (salted-hash) password in RealSet, $g\equiv r$, and $\mathcal{A}_1$ will stop at r=r^* once her expected marginal reward is smaller than the marginal cost, i.e., $R(r^*)$–$R(r^*$–$1)<C(r^*)$–$C(r^*$–$1)$. With this premise, the following Theorem 2 provides a generalized approach for $\mathcal{A}_1$ to compute the cracked proportion, without specifying password distributions (e.g., the CDF-Zipf model [73]). The detailed proof is shown in Appendix C.

Theorem 2. *Let v be the account value, k be the cost per guess, $0<\alpha\leq1$ the diminishing return, and P_r, p_r be the CDF and PDF of the password distribution. If there exists*

TABLE VII: Cracked proportions of top-10^5 passwords using Markov [52], PCFG [80], FLA [54], and PassLLM [87] with 10^8 guesses.†

Guessing algorithm	Top-10^5 password cracked proportion							
	Dodonew	Webhost	Rockyou	Tianya	Wishbone	Dailyquiz	Datpiff	Raaga
Markov [52]	**95.27%**	**82.34%**	**98.78%**	**98.54%**	**97.70%**	**93.04%**	**97.46%**	**91.93%**
PCFG [80]	**85.49%**	54.77%	**98.79%**	**92.77%**	**94.14%**	60.84%	71.84%	71.38%
FLA [54]	**82.07%**	71.89%	**95.88%**	**95.99%**	**95.06%**	**96.42%**	**98.31%**	**94.45%**
PassLLM [87]	**92.05%**	75.86%	**97.91%**	**97.26%**	**96.65%**	**92.08%**	**95.57%**	**91.13%**

† For each dataset, a random 50% is chosen as the training set, with the other half being the test set. Due to Zipf's distribution of passwords [73], 70.90%∼82.42% unique passwords in the test set are not in the training set. Webhost denotes 000Webhost.

$$1 > \frac{v}{k}p_1^{\alpha} \text{ for } r{=}1, \text{ or } 1 > \frac{v}{k}\min_{2\le t\le r}\frac{\alpha\cdot p_t}{(1-P_{t-1})P_{t-1}^{1-\alpha}} \text{ for } r{\ge}2, \quad (8)$$

then the profit-driven attacker will stop at the r–1-th unique password in RealSet *without trying to crack the r-th one.*

However, not every guess in DictSet can match a salted hash in RealSet in practical attacks: Particularly, the number g of tried guesses in DictSet is often significantly larger than that r of unique cracked passwords in RealSet (e.g., 10^8 guesses for 10^5 passwords), so the $\mathcal{A}_1$ model may not hold in practice.

B. Theories for more practical $\mathcal{A}_2$ model

To address the practicality issue of $\mathcal{A}_1$, we propose the $\mathcal{A}_2$ attacker model that generates DictSet via practical guessing algorithms (e.g., Markov [52], PCFG [80], and FLA [54]). Since $\mathcal{A}_2$ cannot know whether the next popular password in RealSet will be cracked, for simplicity, we suppose that passwords in RealSet are cracked successively for $\mathcal{A}_2$, i.e., $PW_r{\in}$RealSet will be cracked after PW_{r-1}. In this way, no uncracked passwords exist in $\mathcal{A}_2$'s view. This is reasonable because a DictSet with 10^8 guesses can crack overwhelming proportions of top-10^5 passwords in RealSet (see Table VII).

We now simulate how $\mathcal{A}_2$ conducts offline guessing attacks, and *the key challenge is to determine to what extent she should stop guessing when faced with unmatched guesses.* We denote g_r as the cumulative number of $\mathcal{A}_2$'s hashed guesses to crack $PW_r{\in}$RealSet (with $g_r{\ge}r$, not $g_r{\equiv}r$ as in the $\mathcal{A}_1$ model), and inductively get the reward and cost. More specifically, we suppose that $\mathcal{A}_2$ has cracked $PW_1, \ldots, PW_{r-1}{\in}$RealSet by hashing the top-g_{r-1} guesses in DictSet, and aims to crack $PW_r{\in}$RealSet by hashing $PW_{g_{r-1}+1}, \ldots, PW_g{\in}$DictSet. Since each $PW{\in}$RealSet is assumed to be cracked, the reward is the same as Eq. 7, and the cost of hashing g guesses is

$$C(g) = C(g_{r-1}) + k(g - g_{r-1})(1 - P_{r-1}). \quad (9)$$

Accordingly, by calculating the marginal reward and the marginal cost from Eq. 9, we can derive Theorem 3 for $\mathcal{A}_2$. Since the case of r=1 is trivial, we mainly focus on $r{\ge}2$.

Theorem 3. *Suppose the profit-driven $\mathcal{A}_2$ has cracked $PW_1,\ldots, PW_{r-1}{\in}$*RealSet *by hashing the top-g_{r-1} guesses in* DictSet. *When she tries to crack $PW_r{\in}$*RealSet*, if*

$$\Delta_r = (g - g_{r-1}) > \frac{v}{k}\min_{2\le t\le r}\frac{\alpha\cdot p_t}{(1-P_{t-1})P_{t-1}^{1-\alpha}} = \frac{v}{k}Det(r) \quad (10)$$

holds, she will not guess the $r^*{=}r$-th unique password, and obtain the nearly optimal profit (compared with $\mathcal{A}_1$).

We first explain Theorem 3. For $PW_r{\in}$RealSet, Δ_r in the left-hand side of Eq. 10 comes from hashing guesses in the guessing dictionary, and records the number of unmatched guesses plus one $\mathcal{A}_2$ needs to check (i.e., checked guesses). If there is no unmatched guess in cracking PW_r, the top-g_{r-1}+1 guess can match PW_r, i.e., Δ_r=1. For the right-hand side of Eq. 10, $Det(r)$ (resp. k) is obtained from the fitted CDF-Zipf model in Sec. III (resp. cost in Sec. IV), and can be precomputed in advance. Due to the Zipf law [73], there is $P_r{\approx}P_{r-1}$ for moderately large r (where guessing typically stops), so minor changes in the denominator of Eq. 10 will not significantly affect the overall results (see Sec. V-C).

We then show how $\mathcal{A}_2$ can devise her strategy by following Theorem 3. For each target $PW_r{\in}$RealSet, $\mathcal{A}_2$ should calculate $\frac{v}{k}Det(r)$ as *the maximum number of allowed guesses*, and compare with the corresponding Δ_r from the ongoing offline attack. Once there is $\Delta_r{>}\frac{v}{k}Det(r)$, the guess stops. This ensures that $\mathcal{A}_2$'s marginal hashing cost in cracking PW_r will not exceed her expected marginal reward, with at most $v{\cdot}p_r$ (if α=1) extra loss in cracking PW_r, leading to the nearly optimal profit. The strategy is shown in Alg. 1.

Algorithm 1: $\mathcal{A}_2$'s strategy in profit-driven scenario.

Input : Guessing dictionary DictSet, real-world dataset RealSet, account value v, and cost of hashing password k.
Output: Cracked set CrackSet and cracked proportion $\lambda(g)$.

begin
  CrackSet=$\emptyset$, CrackAccount=$\emptyset$;
  $r = |\text{CrackSet}| + 1$, $g = 0$; /* r records the number of unique passwords cracked in RealSet, and g records the cumulative number of checked guesses in DictSet to crack PW_r.*/
  while RealSet\CrackSet $\neq\emptyset$ **do**
    $CheckCount(r)$=0; /* Number of checked guesses.*/
    $Det(r)$= $\min_{2\le t\le r}\frac{\alpha\cdot p_t}{(1-P_{t-1})P_{t-1}^{1-\alpha}}$ for $r{\ge}2$ and $Det(1){=}p_1^{\alpha}$ for r=1; /* Maximum number of allowed guesses. */
    while DictSet\CrackSet$\neq\emptyset \bigwedge U{\notin}$CrackAccount **do**
      $H_{g_iU}{=}Hash(PW_g||Salt_U)$, g++;
      if $H_{g_iU}{\in}$RealSet **then**
        $PW_U{=}PW_g$; /* PW_g is a correct guess.*/
        CrackSet.$Append(PW_g)$, $\lambda(g)$+=p_r, $CheckCount(r)$++, r++;
        CrackedAccount.$Append(U)$;
      else
        $CheckCount(r)$++;
        if $CheckCount(r){>}\frac{v}{k}Det(r)$ **then**
          Stop guess; /* Guess number $\le\frac{v}{k}Det(r)$.*/
Output: Cracked set CrackSet, and cracked proportion $\lambda(g)$.

The attacker model accounting for uncracked passwords (denoted as $\mathcal{A}_2'$) can become less practical to implement. Specifically, P_{r-1} in the reward cost terms are replaced by $\sum_{i=1}^{r-1}\delta_i p_i$, where δ_i=1 if $PW_i{\in}$RealSet$\cap$DictSet and δ_i=0 otherwise. Thus, Eq. 10 in Theorem 3 becomes

$$(g - g_{r-1}) > \frac{v}{k}\min_{2\le t\le r^*}\frac{\alpha\cdot p_t}{(1-\sum_{t=1}^{r-1}\delta_t p_t)(\sum_{t=1}^{r-1}\delta_t p_t)^{1-\alpha}} \quad (11)$$

Unlike Eq. 10, Eq. 11 is hard to compute in advance. First, the prior cracked proportion $\sum_{i=1}^{r-1}\delta_i{\cdot}p_i$ is unknown without actual cracking actions. Second, even if $\sum_{i=1}^{r-1}\delta_i{\cdot}p_i$ is known, $\mathcal{A}_2'$ still has to estimate the rank r^* of the password PW_{r^*} she will crack (e.g., assuming $r^*{=}r$). These make the $\mathcal{A}_2'$ model difficult for security administrators who do not perform cracking to use. Fortunately, since guessing dictionaries largely cover the top-10^5 passwords (see Table VII), uncracked passwords can be reasonably ignored with minimal impacts.

TABLE VIII: Cracked proportion under idealized $\mathcal{A}_1$ and more practical $\mathcal{A}_2$ models incorporating Markov [52], PCFG [80], FLA [54] and PassLLM [87].

Dataset	Idealized guessing†		Markov guessing algorithm [52]‡			PCFG guessing algorithm [80]‡			FLA guessing algorithm [54]‡			PassLLM guessing algorithm [87]‡		
	Blocki model [22]	Our $\mathcal{A}_1$ model	Our $\mathcal{A}_2$ model‡	Difference‡	Estimated§ cracked interval	Our $\mathcal{A}_2$ model‡	Difference‡	Estimated§ cracked interval	Our $\mathcal{A}_2$ model‡	Difference‡	Estimated§ cracked interval	Our $\mathcal{A}_2$ model‡	Difference‡	Estimated§ cracked interval
Yahoo	17.64%	33.10%	—	—	—	—	—	—	—	—	—	—	—	—
Dodonew	11.15%	30.56%	16.61%	0.11%	14.20%∼19.32%	17.61%	1.91%	15.13%∼20.38%	11.01%	1.89%	8.84%∼13.38%	10.57%	0.85%	8.40%∼12.93%
000Webhost	5.80%	22.13%	7.35%	0.42%	5.24%∼9.63%	6.02%	2.06%	3.95%∼8.25%	4.84%	0.13%	2.80%∼7.02%	5.29%	0.14%	3.24%∼7.48%
Rockyou	25.64%	45.73%	28.88%	0.64%	25.75%∼32.49%	33.47%	0.22%	29.90%∼37.63%	18.51%	0.07%	16.03%∼21.25%	17.68%	0.89%	15.23%∼20.39%
Tianya	28.14%	48.16%	32.50%	0.22%	29.38%∼36.02%	36.82%	2.23%	33.40%∼40.83%	23.81%	0.05%	21.22%∼26.70%	20.21%	0.36%	17.76%∼22.89%
Duelingnetwork	19.11%	100.0%	—	—	—	—	—	—	—	—	—	—	—	—
Chegg	2.96%	14.17%	—	—	—	—	—	—	—	—	—	—	—	—
Wishbone	16.70%	53.92%	18.28%	0.04%	15.70%∼21.25%	19.72%	0.42%	17.03%∼22.81%	12.64%	0.56%	10.37%∼15.18%	12.22%	0.02%	9.95%∼14.71%
Dailyquiz	20.85%	52.39%	21.78%	0.68%	18.84%∼25.29%	21.34%	5.11%	18.41%∼24.84%	12.94%	0.81%	10.59%∼15.57%	10.25%	0.43%	8.05%∼12.70%
Datpiff	22.01%	100.0%	22.61%	2.02%	19.62%∼26.18%	23.60%	5.36%	20.51%∼27.28%	14.74%	0.16%	12.31%∼17.48%	11.56%	0.93%	9.30%∼14.06%
Raaga	27.06%	100.0%	23.40%	0.04%	20.56%∼26.67%	23.60%	0.17%	20.72%∼26.89%	20.55%	0.01%	17.87%∼23.55%	18.15%	1.42%	15.63%∼21.00%

† $\mathcal{A}_1$ can generate a 100% accurate guessing dictionary, and only does not know which password is adopted by a particular user. Blocki et al.'s model [22] assumes k=k_E=$C_{H_C}(\frac{\sigma^2}{3000}+\sigma)$ with C_{H_C}=7×10^{-15} and σ=2^{20}KB=1GB. Our $\mathcal{A}_1$ model is built on the GSS-fitted CDF-Zipf model [73] justified in Sec. III, and the real-world MHF data in Sec. IV. Yahoo, Duelingnetwork, and Chegg datasets are only explored under the $\mathcal{A}_1$ model since they do not contain plaintext passwords.

‡ The more practical and advanced $\mathcal{A}_2$ attacker uses leading Markov [52], PCFG [80], FLA [54] and PassLLM [87] algorithms to generate guessing dictionaries, and unmatched guesses exist. The "Difference" column records the difference in cracked proportions with and without considering uncracked passwords.

§ The estimated cracked interval is obtained from Eq. 12, which estimates $\mathcal{A}_2$'s cracked proportion when Δ_r is not exactly known. Even with some extent of inaccuracy, this interval still covers the actual cracked proportion with relatively tight lower and upper bounds.

Approximation of cracked proportion. We now show how security administrators can use the $\mathcal{A}_2$ model to evaluate password security in the view of guessing economics. Once the expected checked guesses Δ_r=g–g_{r-1} is estimated, e.g., by comparing the guessing dictionary with the known training set, they can approximate the cracked proportion without cracking actions. Formally, we can obtain Corollary 1 as follows.

Corollary 1. *If passwords are accurately characterized by the CDF-Zipf model [73] with the explicit P_r=Cr^s (see Sec. III-D), $Det(r)$ is non-increasing, and P_{r-1}≈P_r holds when guessing stops, then the cracked proportion against $\mathcal{A}_2$ knowing the number of checked guesses Δ_r should fall into*

$$F_1(r)=\left(\frac{1}{1+\frac{v\cdot s}{k\Delta_r(r-1)}}\right)^{\frac{1}{\alpha}}<P_r<\frac{1}{1+(\frac{\alpha\cdot v\cdot s}{k\cdot\Delta_r r})^{\frac{1}{\alpha}}}=F_2(r). \quad (12)$$

From Corollary 1, a security administrator with estimated Δ_r can approximate the cracked proportion in a simple yet effective way without performing guessing attacks. More specifically, she can solve P_r=$F_1(r)$ and P_r=$F_2(r)$, and bound the cracked proportion with $[\max\{F_1(r_1),F_2(r_1)\},\min\{F_1(r_2),F_2(r_2)\}]$. To ensure this simplicity, it is also necessary to exclude uncracked passwords. Otherwise, P_r in Eqs. 9 and 10 becomes $\sum_{i=1}^{r-1}\delta_i{\cdot}p_i$, where the cracked proportion $\sum_{i=1}^{r-1}\delta_i p_i$ cannot be estimated in prior. The detailed proof is shown in Appendix C.

C. Empirical evaluation results

Experimental setups. We adopt the GSS-fitted CDF-Zipf model under the KS statistic metric (see Table II) to characterize password distributions. We set v=\$1, which is the lowest account price in [86], and adopt α=0.8 to measure diminishing returns as in [22]. For MHFs, we use Argon2i with the recommended memory iteration τ=3 and usage σ=1GB [17]; For slow hashes, we use PBKDF2 [45] with the number of SHA256 iterations being 10^7 as in [22], [27]. Our results are mainly obtained under these concrete setups.

For the idealized $\mathcal{A}_1$ model, we use all eleven datasets in Table I. For the more practical $\mathcal{A}_2$ model, we use the datasets with plaintext passwords in Table VII to generate guessing dictionaries. Following [74], [78], we randomly split each dataset into two halves: We use one half as the training set to generate 10^8 guesses and treat the other as the test set. This "half-half" practice does not violate the requirement that training and test sets should be different, since the Zipf distribution [73] makes that 70.90%∼82.42% of unique passwords different. We also do not remove overlapped passwords as in [74], [77], [78], [87]. Since two halves are equal-sized, parameters obtained on the training set can accurately characterize the test set.

Next, we train our password guessing algorithms. For Markov, we choose the order-3 (i.e., 4-gram), and use Laplace smoothing and EndSymbol normalization as suggested in [52]; For PCFG [80][4], we divide a password into letters, digits, and special symbol segments; For FLA, we use the same settings as in [54]; For PassLLM [87], we use the distilled version with 0.5 billion parameters since it can efficiently guess passwords without noticeably impairing guessing performance.

Results of $\mathcal{A}_1$ and $\mathcal{A}_2$ models. Table VIII shows cracked proportions. Under the $\mathcal{A}_1$ model, MHFs are less secure than the prior work [22] assumed: 14.17%∼100% of passwords will be cracked under our idealized $\mathcal{A}_1$ model built on the accurate GSS-fitted CDF-Zipf model and MHF cost measurements, which is a 71.14%∼423.29% overestimate relative to prior 2.96%∼28.14% given by Blocki et al.'s model [22].

Results of the $\mathcal{A}_2$ model in Table VIII are obtained from implementing Alg. 1. Overall, these guessing algorithms are effective, as cracked proportions can reach 10.25%∼36.82% (except 000Webhost that requires passwords to have both letters and digits, and character length ≥6). Even if unmatched guesses exist, on seven out of eight datasets with plaintext passwords, cracked proportions can be larger than those from Blocki et al.'s $\mathcal{A}_1$ model [22], strengthening the insecurity of MHFs. Additionally, the difference in cracked proportions is generally small when considering uncracked passwords, with the differences being as small as 0.01%∼5.36%. This is reasonable, since there are at most 13 successive uncracked passwords before guesses stop, and the GSS-fitted CDF-Zipf model [73] is relatively accurate with the KS statistic being 0.44%∼4.46%. *This substantiates our assumption in the $\mathcal{A}_2$ model that ignoring uncracked passwords is acceptable.* We also note that, regardless of attacker models, setting $P_{t-1}^{1-\alpha}$ as $P_t^{1-\alpha}$ in the denominator of Eqs. 10 and 11 has almost no impact on the cracked proportion, with differences smaller

[4]See the PCFG code at https://github.com/lakiw/pcfg_cracker .

than 0.02% for $\mathcal{A}_1$ and $\mathcal{A}_2$. This is reasonable, since when guessing stops, the difference in $Det(r)$ is as small as 0.058%.

We further investigate the features of cracked passwords. First, the profit-driven $\mathcal{A}_2$ makes 0.7∼99 million guesses, and cracks about 1,582∼62,339 unique passwords, exceeding the top-10^3 most popular ones. As a result, there can be 12.40%∼69.80% (except 000Webhost) of cracked passwords having frequencies no more than 20, while all top-10^3 ones have higher frequencies. Second, the cracked passwords also extend beyond common patterns. For instance, in Chinese datasets (i.e., Dodonew and Tianya), where users are prone to using digits [76], the proportion of consecutive digits and repetitions is 32.38%∼49.72% (resp. 20.74%∼31.16%) for the top-10^3 (resp. top-10^4) passwords, while it drops to about 17.10%∼24.76% for the Markov-cracked passwords. Third, cracked passwords are structurally more complex. In English datasets (except 000Webhost), where users are more likely to use letters [76], the proportion of mixed-type passwords (i.e., those containing at least two of letters, digits, and symbols) reaches 18.51%∼43.11% among PCFG-cracked passwords, compared with 12.82%∼37.38% among top-10^4 ones.

Fig. 3 shows how, under the log-log scale, password distributions and hashing costs jointly impact password security against $\mathcal{A}_2$ using Markov [52]. Each curve represents the maximum number of allowed guesses (i.e., the right-hand side of Eq. 10), and it is impacted by $Det(r)$ depending on the password distribution and k depending on the hashing cost k. These curves are parallel: The slope is about $-1+s\alpha$, and the y-intercepts are mainly impacted by k, C and α. Besides, the curves representing ASIC and GPU are almost overlapping, justifying our usage of GPU computation data (see Sec. IV-C). The shadow area gives the number of checked guesses for each password in cracking (i.e., the left-hand side of Eq. 10): When $\mathcal{A}_2$ stops guessing, there is $5 \leq \Delta_r \leq 59$, which does not intersects with $\Delta_r = y = 1$ as in the $\mathcal{A}_1$ model.

Table VIII also validates Corollary 1. Overall, even if Δ_r is not exactly estimated, the estimated interval can largely cover $\mathcal{A}_2$'s cracked proportion, and the length of the estimated interval ranges from 4.22%∼7.73%. For instance, when $\Delta_r \in [4, 12]$, the estimated interval is 14.20%∼19.32% for Dodonew against Markov [52], which tightly covers $\mathcal{A}_2$'s cracked proportion 16.61%. This enables our economic model to accurately estimate the threat of offline guessing attacks.

Impacts of users' password behaviors. We examine this by using the widely deployed Zxcvbn password strength meter (PSM) [81][5] as a concrete example. More specifically, we suppose that weak passwords scored 0 and 1 out of 5 will not be allowed after the Zxcvbn PSM is enforced. In this case, the fitted C decreases by 69.17%∼93.28% while s increases by 27.00%∼73.84%, *meaning popular passwords become less concentrated and their frequencies are less skewed.* Consequently, the marginal reward decreases while the marginal cost increases, so a profit-driven attack would stop earlier. Our evaluation results in Table IX confirm this: (1) For $\mathcal{A}_1$, the cracked proportion after filtering drops to 13.99%∼54.48% (except Raaga), a reduction of 7.53%∼45.52%; In particular, passwords of frequency one in Datpiff remain uncracked once Zxcvbn is enforced; (2) For $\mathcal{A}_2$ using guessing algorithms, the cracked proportion drops to 2.00%∼21.77%, a reduction of 0.04%∼26.74%. Only Dailyquiz shows slight security degradation, possibly because PassLLM [87] generates guesses that better match filtered passwords. *Overall, it is recommended to combine MHFs with defenses that make password distributions less skewed to enhance the overall password security.*

[5]See the Zxcvbn code at https://pypi.org/project/zxcvbn/ .

TABLE IX: Reduced cracked proportion of datasets filtered by Zxcvbn [81].†

Dataset	$\mathcal{A}_1$ reduced	$\mathcal{A}_2$ reduced			
		Markov [52]	PCFG [80]	FLA [54]	PassLLM [87]
Dodonew	12.65%	6.89%	10.34%	4.75%	3.60%
000Webhost	7.53%	2.93%	4.02%	2.56%	2.76%
Rockyou	31.73%	17.58%	26.74%	16.45%	15.00%
Tianya	27.93%	18.63%	25.57%	16.09%	11.95%
Wishbone	32.54%	7.70%	13.92%	8.60%	7.58%
Dailyquiz	9.24%	3.11%	5.95%	0.04%	–2.52%
Datpiff	45.52%	0.84%	15.83%	7.45%	5.90%
Raaga	0.00%	4.05%	11.55%	3.17%	1.62%

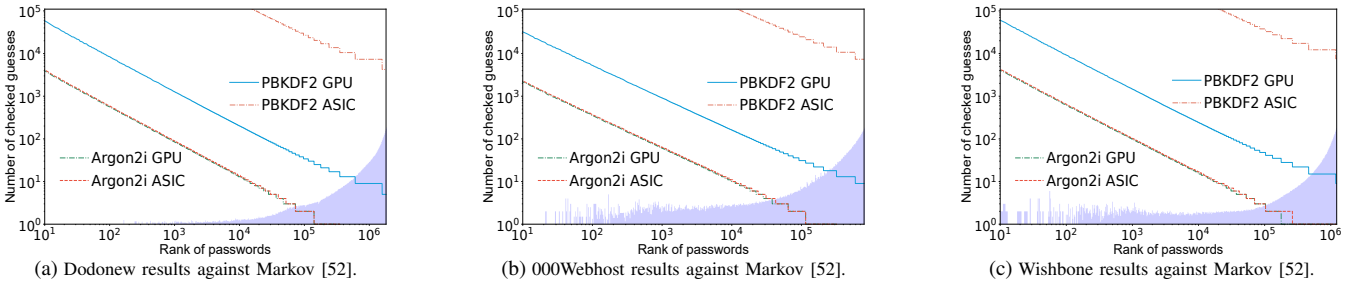
† The reduced cracked proportion is computed as the difference between the cracked proportions of the Zxcvbn-filtered and the original datasets in Table VIII.

Cross-website guessing. We also preliminarily explore cross-website guessing attacks where the training and test sets come from different datasets. For English (resp. Chinese) services, we train on Rockyou (resp. Tianya) and test on Wishbone and Dailyquiz (resp. Dodonew) datasets. Our results show that the cracked proportion remains noticeable, which ranges from 20.53%∼30.44% with PassLLM [87] performing the best. This is expected because even in the intra-website case, the Zipf distribution ensures that 70.90%∼82.42% of unique passwords differ between training and test sets. Additionally, since many datasets (e.g., [46], [62]) have been leaked multiple times, focusing on intra-website guessing is largely reasonable.

D. Limitation and discussion

Dependence on specific settings. Our conclusion that MHFs alone do not provide sufficient security is based on the widely used CDF-Zipf password distribution model [73] (see Sec. III), the profit-driven scenario (see Sec. V) along with its reward (i.e., \$1) and MHF cost parameters (see Sec. IV). Similar analyses under other settings can accordingly be conducted.

More specifically, first, for password distributions, the stretched-exponential model [42] might be incorporated as an alternative to the widely used CDF-Zipf model [73]. Since it provides better accuracy on eight of our eleven datasets (see Sec. III-B), it may improve our economic analysis. Second, for the MHF cost, our results are based on the assumption that memory costs are relatively stable across architectures and hard to reduce on GPUs/ASICs [14], [17], [60]. Fluctuations in cost measurements, hardware and electricity prices may affect the exact values, but our framework remains applicable. Moreover, more efficient implementations, such as improved pebbling attacks [12], [13], [14], [18], [21] and more advanced memory technologies, may further reduce password security against offline attacks. Consequently, higher account values (e.g., those with financial assets [48]) will motivate profit-driven attackers to crack more accounts. Third, attack scenarios beyond the profit-driven case also exist. For example,



(a) Dodonew results against Markov [52]. (b) 000Webhost results against Markov [52]. (c) Wishbone results against Markov [52].

Fig. 3: How $\mathcal{A}_2$ using the Markov guessing algorithm [52] should hash guesses in the profit-driven scenario. Each curve represents the maximum number of allowed guesses with a given hash function (PBKDF2 [45] or Argon2i [17]) under our setting and cracking hardware (ASIC or GPU in Table VI). The shadow area is the actual number of checked guesses determined by Markov [52]. The attacker will stop guessing once the curve enters the shadow area.

in the fixed-budget scenario, an attacker keeps guessing until the budget is exhausted, yielding a larger cracked proportion.

Impacts of additional factors. In terms of costs, our economic model focuses on the per-guess hashing cost, while omitting several real-world factors attackers might face in full campaigns. First, dataset acquisition is largely negligible. Breached datasets are often sold for less than \$0.001 per account (e.g., [28], [31]), which is far below the assumed \$1 account value [86], and some are freely available (see [11], [30]). Second, preprocessing costs are relatively small. Fitting datasets in Table I with GSS on our Ubuntu 22.04 platform consumes about 5∼8 orders of magnitude less energy than hashing passwords with slow hashes (τ=10^7 as in [22], [27]) or MHFs (τ=3, σ=1 GB as in [17]). For training and guess generation, even the time-consuming PassLLM, requiring about 22 ($\approx$12+$10^8/(3000\times3600)$) hours on an RTX 4090 GPU (see Table 7 of [87]), yields an amortized per-guess cost 2∼3 orders of magnitude smaller than hashing. An attacker can also reuse one dictionary across similar datasets (e.g., same language and service types) to further reduce this overhead. Infrastructure costs such as cooling can increase electricity consumption by a factor of 1.09∼1.55 [41], but it can be mitigated by locating hardware in regions with cheaper electricity.

Besides, reward factors such as monetization uncertainty can also affect guessing economics. The realized value of a cracked account may fall below its expectation. In this case, attackers can mitigate such risks by mixing high-value (e.g., financial) accounts with low-value (e.g., portal) ones and selling them in batches. Besides, attackers can also monetize accounts through spam and phishing [56]. For instance, in 2025, an attacker used a compromised small-business email account, impersonated a professional service, and obtained its customers' Microsoft credentials [68]. We leave detailed analysis as future work.

VI. Some Insights

Existing password-based applications. Since the CDF-Zipf model [73] has been adopted by over 300 password-related studies (see at https://tinyurl.com/2kpxwjsx), our work can improve their security evaluations. For example, in detecting credential tweaking attacks [58], our justification on the GSS-fitted CDF-Zipf model can lead to selecting a more effective blocklist to prevent extractions of users' passwords from popular variants (e.g., extracting `P@ssword` from `password`). Besides, our study can also be extended to security evaluations of industrial systems. By jointly modeling password distributions and hashing costs, security administrators can evaluate offline-guessing resistance for real-world deployments. For instance, password managers (e.g., KeePassXC [6], NordPass [7], and Bitwarden [8]) use Argon2i hashed master passwords in encrypting password vaults. Such evaluations are particularly important when vaults store high-value assets like financial accounts (e.g., the LastPass breach [48]).

Online guessing economics are also of practical concern. Rate-limiting mechanisms (e.g., CAPTCHAs and lockout [51], [75]) can thwart login attempts, driving attackers towards the more efficient targeted guessing where guesses differ across victims. As Wang et al. [77] reveal, Zipf's law also exists in such settings: For a given targeted algorithm (e.g., TarGuess), the cracked proportion per account after n guesses follows P'_n=$C{\cdot}n^s$, with C and s depending on the target dataset (e.g., $C\in[0.001, 0.01]$ and $s\in[0.15, 0.30]$ for Dodonew, see Fig. 13(k) of [77]). Online guessing costs can involve CAPTCHA solving (e.g., \$0.0004∼\$0.0029 per solve, see [44], [55], [83]), IP rental to evade blocking (e.g., \$50∼\$300 per month [37]), and lockout time (e.g., one hour after 5∼20 failed logins [75]). In this way, the security of online guessing economics can be analyzed in a similar approach.

VII. Conclusion

In this work, we have tackled offline guessing economics in a principled approach. We first systematically evaluate the mainstream maximum likelihood (ML) [33], least linear square (LLS) [22] and golden section search (GSS) [73] fitting methods for the CDF-Zipf password distribution model, and reveal that GSS significantly outperforms others in terms of the critical accuracy, completeness, and stability criteria. This lays a theoretical foundation for using the CDF-Zipf model in over 300 password-based applications across seven domains.

As a key application, we have proposed a practical cost framework for MHFs (represented by Argon2i [17]) and *for the first time*, used empirical data from Argon2i execution to measure its memory costs. With accurate MHF cost data, we have analyzed the offline guessing security under two attacker models in the major profit-driven attack scenario, each reflecting different guessing capabilities. Our experiment results show that, contrary to Blocki et al.'s conclusion [22], even with recommended settings, MHFs do not offer sufficient security in both idealized and more practical attack models, leaving noticeable proportions of passwords vulnerable. We

also showed how to approximate the proportion of cracked passwords without performing offline guessing attacks. Considering the ubiquity of human-chosen passwords, we believe that our study on the fundamental password distribution issue is of practical significance, and will trigger research interest in measuring how password distributions can impact the security evaluations of various password-based applications.

Ethical considerations

Although our datasets in Table I are publicly available and have been widely used in prior studies [22], [23], [52], [54], [77], [80], [87], passwords are essentially sensitive, and using them can bring risks to affected users. We acknowledge such risks, and recognize that the users whose data were exposed in these breaches have not consented to use their data in research, nor have they had any opportunity to consent. Despite their publicity, their usage still raises important ethical considerations, and we process them with caution.

More specifically, we take the following four precautions: (1) We do not further disseminate breached datasets to other parties, and only use them for this study; (2) Our datasets are stored and processed on computers unconnected to the Internet; (3) We treat each account as confidential, and mainly report aggregated statistics such as frequencies to minimize the impacts on affected users; (4) We delete all sensitive information after our analysis. Through these precautions, our work meets the key ethical guidelines [69]. We have also obtained approvals from our center's IRB. Overall, our study benefits the academic community by introducing a new understanding of offline guessing economics, thereby facilitating security administrators to take proper countermeasures.

Acknowledgment

The authors are grateful to the anonymous reviewers for their invaluable comments. Ding Wang is the corresponding author. The authors would like to thank Yuanwei Ji for the helpful discussion. This research was in part supported by the National Natural Science Foundation of China under Grant Nos. 62602302 and 62572259, and by the Tianjin Natural Science Foundation Project under Grant No. 25JCJQJC00230.

References

[1] *Password hashing competition and our recommendation for hashing passwords: Argon2*, April 2019, https://password-hashing.net/.

[2] *Bitcoin miner S19 pro*, May 2020, https://www.asicminervalue.com/miners/bitmain/antminer-s19-pro-110th.

[3] *Simpler, stronger authentication: solving the world's password problem*, July 2023, https://fidoalliance.org/.

[4] *Gen Z's Password Fatigue Finds 72% of Digital Natives Reuse the Same Password Across Accounts*, April 2025, https://www.businesswire.com/news/home/20250423085569/en/.

[5] *Global electricity prices*, 2025, https://www.globalpetrolprices.com/USA/electricity_prices/.

[6] *KeePassXC: User Guide*, Oct. 2025, https://keepassxc.org/docs/KeePassXC_UserGuide.

[7] *NordPass Business Whitepaper*, Oct. 2025, https://assets.nordpass.com/nordpass-business-whitepaper.pdf.

[8] *Bitwarden Security Whitepaper*, 2026, https://bitwarden.com/help/bitwarden-security-white-paper.

[9] *Hashcat: Advanced password recovery*, Oct. 2026, https://hashcat.net/hashcat/.

[10] M. Abdalla, F. Benhamouda, and P. MacKenzie, "Security of the J-PAKE password-authenticated key exchange protocol," in *Proc. IEEE S&P 2015*, pp. 571–587.

[11] L. Abrams, *Havenly discloses data breach after 1.3M accounts leaked online*, Aug. 2020, https://www.bleepingcomputer.com/news/security/havenly-discloses-data-breach-after-13m-accounts-leaked-online/.

[12] J. Alwen and J. Blocki, "Efficiently computing data-independent memory-hard functions," in *Proc. CRYPTO 2016*, pp. 241–271.

[13] J. Alwen and J. Blocki, "Towards practical attacks on Argon2i and Balloon hashing," in *Proc. IEEE EuroS&P 2017*, pp. 142–157.

[14] J. Alwen, J. Blocki, and B. Harsha, "Practical graphs for optimal side-channel resistant memory-hard functions," in *Proc. ACM CCS 2017*.

[15] W. Bai and J. Blocki, "DAHash: Distribution aware tuning of password hashing costs," in *Proc. FC 2021*, pp. 382–405.

[16] W. Bai, J. Blocki, and M. H. Ameri, "Cost-asymmetric memory hard password hashing," in *Proc. SCN 2022*, pp. 21–44.

[17] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: New generation of memory-hard functions for password hashing and other applications," in *Proc. IEEE EuroS&P 2016*, pp. 292–302.

[18] A. Biryukov and D. Khovratovich, "Tradeoff cryptanalysis of memory-hard functions," in *Proc. ASIACRYPT 2015*, pp. 633–657.

[19] J. Blocki and A. Datta, "CASH: A cost asymmetric secure hash algorithm for optimal password protection," in *Proc. IEEE CSF 2016*.

[20] J. Blocki, A. Datta, and J. Bonneau, "Differentially private password frequency lists," in *Proc. NDSS 2016*, pp. 1–15.

[21] J. Blocki, B. Harsha, S. Kang, S. Lee, L. Xing, and S. Zhou, "Data-independent memory hard functions: New attacks and stronger constructions," in *Proc. CRYPTO 2019*, pp. 573–607.

[22] J. Blocki, B. Harsha, and S. Zhou, "On the economics of offline password cracking," in *Proc. IEEE S&P 2018*, pp. 35–53.

[23] J. Blocki and P. Liu, "Towards a rigorous statistical analysis of empirical password datasets," in *Proc. IEEE S&P 2023*, pp. 607–625.

[24] J. Bonneau, C. Herley, P. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE S&P 2012*, pp. 553–567.

[25] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proc. IEEE S&P 2012*, pp. 538–552.

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

[27] J. Bonneau and S. Schechter, "Toward reliable storage of 56-bit keys in human memory," in *Proc. USENIX SEC 2014*, pp. 607–623.

[28] Brinztech, *Brinztech Alert: Database of SecureTeen is on Sale (1.4 Million Records)*, Nov. 2025, https://www.brinztech.com/breach-alerts/brinztech-alert-database-of-secureteen-is-on-sale-1-4-million-records/.

[29] P. A. Cabarcos and P. Mayer, "The more accounts I use, the less I have to think': A longitudinal study on the usability of password managers for novice users," in *Proc. SOUPS 2025*, pp. 351–369.

[30] Catalin Cimpanu, *Dating app MobiFriends silent on security breach impacting 3.6 million users*, May 2020, https://www.zdnet.com/article/dating-app-mobifriends-silent-on-security-breach-impacting-3-6-million-users/.

[31] Catalin Cimpanu , *Hacker selling 40 million user records from popular Wishbone app*, May 2020, https://www.zdnet.com/article/hacker-selling-40-million-user-records-from-popular-wishbone-app/.

[32] S. A. Chaudhry, A. Irshad, K. Yahya, N. Kumar, M. Alazab, and Y. B. Zikria, "Rotating behind privacy: An improved lightweight authentication scheme for cloud-based IoT environment," *ACM Trans. Internet Technol.*, vol. 21, no. 3, pp. 1–19, 2021.

[33] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, vol. 51, no. 4, pp. 661–703, 2009.

[34] S. Croley, *RTX 4090 v6.2.6. benchmark*, Oct. 2022, https://gist.github.com/Chick3nman/32e662a5bb63bc4f51b847bb422222fd.

[35] Y. Duan, D. Wang, and Y. Fu, "Security analysis of master-password-protected password management protocols," in *Proc. IEEE S&P 2025*.

[36] B. Eitel, *Will passwords become a thing of the past?*, Mar. 2023, https://www.idsalliance.org/will-passwords-become-a-thing-of-the-past/.

[37] M. Eriksson, *Best Residential Proxies 2026*, April 2026, https://www.privateproxyguide.com/best-rotating-residential-proxies/.

[38] D. Florêncio, C. Herley, and P. van Oorschot, "An administrator's guide to Internet password research," in *Proc. USENIX LISA 2014*, pp. 44–61.

[39] X. Gao, Y. Yang, C. Liu, C. Mitropoulos, J. Lindqvist, and A. Oulasvirta, "Forgetting of passwords: Ecological theory and data," in *Proc. USENIX SEC 2018*, pp. 221–238.

[40] M. Grafnetter, *Pass-the-Passkey Family of Attacks*, July 2026, https://specterops.io/wp-content/uploads/sites/3/2026/08/Pass-the-Passkey_A4_v2.pdf.
[41] Hanwha Data Centers, *Energy Efficiency Solutions for Data Centers: 2025 and Beyond*, Oct. 2025, https://www.hanwhadatacenters.com/blog/energy-efficiency-solutions-for-data-centers-2025-and-beyond/.
[42] Z. Hou and D. Wang, "New observations on Zipf's law in passwords," *IEEE Trans. Inform. Foren. Secur.*, vol. 18, pp. 517–532, 2023.
[43] I. Inc., *Intel performance counter monitor - A better way to measure CPU utilization*, Nov. 2022, https://www.intel.com/content/www/us/en/developer/articles/technical/performance-counter-monitor.html.
[44] D. K, *6 Best CAPTCHA Solving Tools (2026): Pricing, Speed & Accuracy Compared*, April 2026, https://spyderproxy.com/blog/6-best-captcha-solving-tools.
[45] B. Kaliski, "PKCS # 5: Password-based cryptography specification version 2.1," Tech. Rep., Jan. 2017.
[46] M. Kan, *Yahoo execs botched its response to 2014 breach, investigation finds*, Mar. 2017, https://www.csoonline.com/article/560433/yahoo-exec s-botched-its-response-to-2014-breach-investigation-finds.html.
[47] J. Katz, R. Ostrovsky, and M. Yung, "Efficient and secure authenticated key exchange using weak passwords," *J. ACM*, vol. 57, no. 1, pp. 1–41, 2009.
[48] R. Lakshmanan, *LastPass 2022 Breach Led to Years-Long Cryptocurrency Thefts, TRM Labs Finds*, Dec. 2025, https://thehackernews.com/2025/12/lastpass-2022-breach-led-to-years-long.html.
[49] J. H. Laros, P. Pokorny, and D. DeBonis, "Powerinsight-a commodity power measurement capability," in *Proc. IGCC 2013*, pp. 1–6.
[50] L. Lassak, E. Pan, B. Ur, and M. Golla, "Why aren't we using passkeys? Obstacles companies face deploying FIDO2 passwordless authentication," in *Proc. USENIX SEC 2024*, pp. 7231–7248.
[51] B. Lu, X. Zhang, Z. Ling, Y. Zhang, and Z. Lin, "A measurement study of authentication rate-limiting mechanisms of modern websites," in *Proc. ACSAC 2018*, pp. 89–100.
[52] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE S&P 2014*, pp. 689–704.
[53] D. Malone and K. Maher, "Investigating the distribution of password choices," in *Proc. WWW 2012*, pp. 301–310.
[54] 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 *Proc. USENIX SEC 2016*, pp. 175–191.
[55] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: CAPTCHAs-Understanding CAPTCHA-solving services in an economic context," in *Proc. USENIX SEC 2010*.
[56] J. Onaolapo, E. Mariconti, and G. Stringhini, "What happens after you are pwnd: Understanding the use of leaked webmail credentials in the wild," in *Proc. IMC 2016*, pp. 65–79.
[57] B. Pal, T. Daniel, R. Chatterjee, and T. Ristenpart, "Beyond credential stuffing: Password similarity models using neural networks," in *Proc. IEEE S&P 2019*, pp. 417–434.
[58] B. Pal, M. Islam, M. Sanusi, N. Sullivan, L. Valenta, T. Whalen, C. Wood, T. Ristenpart, and R. Chattejee, "Might I get pwned: A second generation compromised credential checking service," in *Proc. USENIX SEC 2022*, pp. 1831–1848.
[59] P. Peng, C. Xu, L. Quinn, H. Hu, B. Viswanath, and G. Wang, "What happens after you leak your password: Understanding credential sharing on phishing sites," in *Proc. ACM ASIACCS 2019*, pp. 181–192.
[60] C. Percival. 2009., *Stronger key derivation via sequential memory-hard functions*, 2009, https://tools.ietf.org/html/rfc7914.
[61] M. J. D. Powell, *Approximation theory and methods*. Cambridge University press, 1981.
[62] S. Ragan, *R2Games compromised again, over one million accounts exposed*, April 2017, https://www.csoonline.com/article/561335/r2games-compromised-again-over-one-million-accounts-exposed.html.
[63] T. Reittinger and G. Pernul, "No password, no problem? A large-scale field study of passkey adoption and usage," in *Proc. IEEES&P 2026*, pp. 3607–3626.
[64] L. Ren and S. Devadas, "Bandwidth hard functions for ASIC resistance," in *Proc. TCC 2017*, pp. 466–492.
[65] S. Salamatian, W. Huleihel, A. Beirami, A. Cohen, and M. Médard, "Why botnets work: Distributed brute-force attacks need no synchronization," *IEEE Trans. Inform. Foren. Secur.*, vol. 14, no. 9, pp. 2288–2299, 2019.
[66] StrongU Inc, *StrongU STU*, 2018, https://www.asicminervalue.com/miners/strongu/stu-u2.
[67] D. Temoshok, J. Fenton, Y.-Y. Choong, N. Lefkovitz, A. Regenscheid, and J. Richer, "NIST 800-63B-4 digital identity guidelines: Authentication and lifecycle management," National Institute of Standards and Technology, Tech. Rep., July 2025.
[68] The Proofpoint Threat Research Team, *Microsoft OAuth App Impersonation Campaign Leads to MFA Phishing*, July 2025, https://www.proofpoint.com/us/blog/threat-insight/microsoft-oauth-app-impersonation-campaign-leads-mfa-phishing.
[69] D. R. Thomas, S. Pastrana, A. Hutchings, R. Clayton, and A. R. Beresford, "Ethical issues in research using datasets of illicit origin," in *Proc. IMC 2017*, pp. 445–462.
[70] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki *et al.*, "Data breaches, phishing, or malware? Understanding the risks of stolen credentials," in *Proc. ACM CCS 2017*, pp. 1421–1434.
[71] K. Toubba, *Notice of recent security incident*, Dec. 2022, https://blog.lastpass.com/2022/12/notice-of-recent-security-incident/.
[72] Victor, *Raising the bar in security with Argon2*, Aug. 2023, https://tuta.com/blog/best-encryption-with-kdf.
[73] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inform. Foren. Secur.*, vol. 12, no. 11, pp. 2776–2791, 2017.
[74] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "A security analysis of honeywords," in *Proc. NDSS 2018*, pp. 1–16.
[75] D. Wang, X. Shan, Q. Dong, Y. Shen, and C. Jia, "No single silver bullet: Measuring the accuracy of password strength meters," in *Proc. USENIX SEC 2023*, pp. 947–964.
[76] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: Understanding passwords of Chinese web users," in *Proc. USENIX SEC 2019*, pp. 1537–1555.
[77] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM CCS 2016*.
[78] D. Wang, Y. Zou, Z. Zhang, and K. Xiu, "Password guessing using random forest," in *Proc. USENIX SEC 2023*, pp. 965–982.
[79] K. C. Wang and M. K. Reiter, "Detecting stuffing of a user's credentials at her own accounts," in *Proc. USENIX SEC 2020*, pp. 2201–2218.
[80] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. IEEE S&P 2009*, pp. 391–405.
[81] D. Wheeler, "zxcvbn: Low-budget password strength estimation," in *Proc. USENIX SEC 2016*, pp. 157–173.
[82] M. Xu, C. Wang, J. Yu, J. Zhang, K. Zhang, and W. Han, "Chunk-level password guessing: Towards modeling refined password composition representations," in *Proc. ACM CCS 2021*, pp. 5–20.
[83] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang, "Yet another text captcha solver: A generative adversarial network based approach," in *Proc. ACM CCS 2018*, pp. 332–348.
[84] V. Zimmermann, "From the quest to replace passwords towards supporting secure and usable password creation," Ph.D. dissertation, 2021, technische Universität.
[85] G. K. Zipf, *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books, 2016.
[86] M. Zoltan, *Dark web price index 2023*, April 2023, https://www.privacyaffairs.com/dark-web-price-index-2023/.
[87] Y. Zou, M. An, and D. Wang, "Password guessing using large language models," in *Proc. USENIX SEC 2025*, pp. 7799–7818.

APPENDIX

A. Detailed explanations of preliminaries

Relationship between the KS statistic and normalized $\|D\|_p$. Let $x_r{=}\frac{1}{N}\sum_{i=1}^{r}|P_r{-}\hat{P}_r|$, where N denotes the number of unique elements. Without loss of generality, we assume $0{\le}x_1{\le}x_2\le\ldots\le x_N$. When $x_N{>}0$ and $p{\ge}1$, there exist

$$\|x\|_p = x_N\left(\sum_{r=1}^{N}\left(\frac{x_r}{x_N}\right)^p\right)^{1/p} = x_N\Big(1+\sum_{r=1}^{N-1}\left(\frac{x_r}{x_N}\right)^p\Big)^{1/p} \qquad (13)$$
$$x_N\cdot 1^{1/p} \le \|x\|_p \le x_N\cdot(N\cdot 1)^{1/p} = x_N\cdot N^{1/p}.$$

As $p{\to}\infty$, by squeeze theorem we get $\lim_{p\to\infty}\|x\|_p{=}x_N$, so $\|D\|_\infty$ is the KS statistic D_{KS}. This completes the proof.

We next present Clauset et al.'s maximum likelihood (ML) [33], Blocki et al.'s least linear square (LLS) [22], and Wang et al.'s golden section search (GSS) [73] fitting methods, and how they characterize password distributions.

Maximum Likelihood (ML). In 2009, Clauset et al. [33] studied power-law distributions in various fields (e.g., physics, biology and computer science), and suggested using the maximum likelihood (ML) fitting method. Particularly, for popular passwords with frequency $f_i \geq f_{\min}$ (e.g., $f_{\min}$=5), they proposed the approximated probability density (mass) function (PDF) and the complementary cumulative distribution function (CCDF) of the power-law as follows

$$\begin{cases} p(f) = (1-\alpha_0)\left(f_{\min}-\frac{1}{2}\right)^{1-\alpha_0}\left(f-\frac{1}{2}\right)^{-\alpha_0}, \\ P_c(f) = \left(\frac{f_{\min}-\frac{1}{2}}{f-\frac{1}{2}}\right)^{\alpha_0-1}. \end{cases} \tag{14}$$

By treating $\mathcal{L}=\sum_{f\geq f_{\min}} \log p(f)$ and solving $\frac{\mathrm{d}\,\mathcal{L}}{\mathrm{d}\,\alpha_0}=0$, their method gives $\alpha_0=1+\frac{N_{\min}}{\left(\sum_{i=1}^{N_{\min}} \log \frac{f_i}{f_{\min}-1/2}\right)}$, where $N_{\min}$ is the number of elements with frequencies $f_i \geq f_{\min}$ [33].

We next show how to use Clauset et al.'s method [33] to generate the fitted dataset FitSet. In general, FitSet is obtained from randomly sampling $N_{\min}$ numbers from the fitted power-law, with each number being the frequency of an element (e.g., password in this work). Particularly, for $N_{\min}$ uniformly distributed numbers $\{u_r\}_{r=1}^{N_{\min}} \in (0,1]$, there is

$$f_r = P_c^{-1}(u_r) = \lfloor (f_{\min}-\frac{1}{2}) u_r^{\frac{1}{1-\alpha_0}} + \frac{1}{2} \rfloor, \tag{15}$$

and ranking them in descending order of frequency leads to FitSet containing elements with $f_i \geq f_{\min}$. Eq. 15 also reveals that the frequency has $f_r \propto r^{\frac{1}{1-\alpha_0}}$, so the parameter s of CDF-Zipf has $s \approx \frac{2-\alpha_0}{1-\alpha_0}$. The process is shown in Alg. 2 as follows.

Algorithm 2: Maximum likelihood (ML) fitting.

Input : Real-world password dataset RealSet.
Output: Fitted parameters α_0 and s, and fitted FitSet [33].

begin
  $\alpha_0 = 1 + \frac{N_{\min}}{\left(\sum_{i=1}^{N_{\min}} \log \frac{f_i}{f_{\min}-1/2}\right)}$; /*Fit with ML [33] method. $N_{\min}$ is the number of unique passwords with $f \geq f_{\min}$.*/
  $s=\frac{2-\alpha_0}{1-\alpha_0}$;**for** $i=1$ **to** $N_{\min}$ **do**
    $u=U(0,1]$; /*Generate random real numbers in $(0,1]$.*/.
    $f=P_c^{-1}(u_i)=\lfloor (f_{\min}-\frac{1}{2}) u^{\frac{1}{1-\alpha_0}} + \frac{1}{2} \rfloor$; /*Generate frequencies following fitted power-law.*/
  for $\langle r, f_r \rangle \in$ Temp **do**
    FitSet.*Append*(f_r);
  Rank FitSet in descending order;
Output: Fitted parameters α_0 and s, and fitted model FitSet.

Least Linear Square (LLS). Under the log-log scale, Blocki et al. [22] used LLS to fit the cumulative distribution function (CDF) of passwords in the rank-frequency coordinate system, and used the explicit expression $P_r=C\cdot r^s$ to characterize the real-world password distribution [22]. In this method, C and s values fitted via LLS depend on log-ranks and log-CDFs of real-world RealSet. Formally, by denoting $x_r=\log r$ and $y_r=\log P_r$ under the log-log scale, we can obtain

$$\begin{cases} s = \frac{\sum_{r=1}^{N}(\log r-\overline{\log r})(\log P_r-\overline{\log P_r})}{\sum_{r=1}^{N}(\log r-\overline{\log r})^2}, \\ C = \exp(\overline{\log P_r} - s\cdot\overline{\log r}), \end{cases} \tag{16}$$

where $\overline{x}$ denotes the mean. This allows us to explain why LLS is not complete, and we will present details in Appendix B.

After obtaining C and s, the fitted model FitSet, which contains a total of $|D|$ passwords, can be similarly constructed via randomly sampling $|D|$ elements from $P_r=C\cdot r^s$. These sampled frequencies are further ranked in descending order. We summarize the process in Alg. 3 as follows.

Algorithm 3: Least linear square fitting [22].

Input : Real-world password dataset RealSet.
Output: Parameters C and s, KS statistic D_{KS} and fitted model FitSet.

begin
  for $r=1$ **to** N **do**
    $x_r=\log r$, $y_r=\log P_r$;
  $\bar{x}=\frac{1}{N}\sum_{r=1}^{N} x_r$, $\bar{y}=\frac{1}{N}\sum_{r=1}^{N} y_r$, $s=\frac{\sum_{r=1}^{N}(x_r-\bar{x})(y_r-\bar{y})}{\sum_{r=1}^{N}(x_r-\bar{x})^2}$, $C=\exp(\bar{y}-s\cdot\bar{x})$; /* Using LLS to fit the password distribution with a straight line under the log-log scale.*/
  for $j=1$ **to** $|D|$ **do**
    $u_j = U(0,1]$; /*Generate random seeds in $(0,1]$.*/;
    $r = \lceil (u_j/C)^{1/s} \rceil$; /* Sampling from the CDF-Zipf model with $P_r=C\cdot r^s$ to construct the fitted model FitSet. */
    Count the frequency of the r as f_r;
  for $\langle r, f_r \rangle \in$ Temp **do**
    FitSet.*Append*(f_r);
  Rank FitSet in descending order;
Output: Fitted parameter results C and s and fitted model FitSet.

Golden section search (GSS) [73] treats fitting a password distribution as a statistical optimization, which minimizes the KS statistic between FitSet and RealSet via iteratively and respectively searching for the optimal C and s values. Specifically, data in FitSet are obtained from randomly sampling $|D|$ elements from $P_r=C\cdot r^s$, and ranking them in descending order of frequency. Through a two-dimensional golden section search, GSS contracts each parameter's search interval by a factor $(\sqrt{5}-1)/2$ per iteration. Note that when the distance between the fitted FitSet and the real-world RealSet is changed into normalized $\|D\|_p=\left(\frac{1}{N}\sum_{r=1}^{N}|P_r-\hat{P}_r|^p\right)^{\frac{1}{p}}$ with finite $p\in\mathbb{Z}^+$ (see Eq. 2), the GSS fitting process can work accordingly by changing the minimization goal from D_{KS} to $\|D\|_p$. We summarize this process in Alg. 4 as follows.

B. Detailed explorations of completeness

We first prove GSS's completeness. Since C and s obtained from GSS are essentially random variables, for simplicity, we denote the statistics under one set of seed $\{u_i\}_{i=1}^{|D|}$ as $C(\{u\})$ and $s(\{u\})$. To begin with, we exclude statistical randomness (i.e., the seed $\{u\}$ is fixed) and prove the existence of the optimal C and s. We then explain that GSS can approximate the optimum for a unimodal objective, and finally incorporate statistical randomness. To do this, we present Chebyshev Alternation Theorem (CAT) [61] as the basis in Lemma 1.

Lemma 1. ***(CAT** [61]). Let H be a d-dimensional subspace $C[a,b]$ of continuous functions satisfying the Haar condition: For every nonzero $h(x)\in H$, it has at most d–1 zeros. Then, for any $g(x)\in C[a,b]$, there exists a unique approximation $h(x)^*\in C[a,b]$ that minimizes its KS statistic with $g(x)$, i.e.,*

$$\max_{x\in[a,b]} |g(x)-h^*(x)| = \min_{h(x)\in H} \max_{x\in[a,b]} |g(x)-h(x)|. \tag{17}$$

Besides, $h^*(x)\in H$ is the optimal approximation of $g(x)\in$

Algorithm 4: Golden section search (GSS) fitting [73].

Input : Real-world password dataset RealSet.
Output: KS statistic D_{KS}, parameters C and s, and fitted model FitSet.

begin
  $(C_0, s_0) = \text{LLS}(\text{CDF}(\text{RealSet}))$; /*Initial searching points.*/
  $C = C_0, s = s_0$;
  for i_C=1 **to** m **do**
    for i_s=1 **to** m **do**
      C=$\text{GS}_1(\frac{C_0}{d_0}, C_0 \times d_0, C)$; /* One-dimensional golden search on C, d_0 is the interval length. */
      for j=1 **to** $|D|$ **do**
        s=$\text{GS}_1(\frac{s_0}{d_0}, s_0 \times d_0, s)$; /* Golden search on s. */
        $u_j = U(0,1), r = \lceil (u_j/C)^{1/s} \rceil$; /* Sampling from P_r=$C \cdot r^s$ to construct the fitted model FitSet. */
        Count the frequency of the r as f_r;
      for $\langle r, f_r \rangle \in$ Temp **do**
        FitSet.*Append*(f_r);
      Rank FitSet in descending order;
      $D_{KS} = \max |\text{CDF}(\text{FitSet}) - \text{CDF}(\text{RealSet})|$;
Output: D_{KS}, C and s, and FitSet.

$C[a,b]$ if and only if $e^*(x)$=$g(x)$–$h^*(x)$ has at least d+1 alternated points $x_1<\ldots<x_{d+1}$ in the interval $[a,b]$ that satisfy $|e^*(x_i)|$=$|e^*(x_{i+1})|$ and $e^*(x_i)$=–$e^*(x_{i+1})$.

The above Lemma 1 lays the foundation for the existence of optimal approximation and reveals the criterion. We now use it to prove the existence of optimal C and s that characterize the GSS-fitted CDF-Zipf model in approximating a password distribution. Formally, we have the following Lemma 2.

Lemma 2. *Let P_r=$\sum_{i=1}^{r} f_i/|D|$ and $\hat{P}_r$=$\sum_{i=1}^{r} \hat{f}_i/|D|$ be the real and fitted CDFs (see Sec. II). For a fixed random seed (omitted for simplicity), there exists a unique pair (C,s) that minimizes the KS statistic D_{KS}=$\max_r |P_r$–$\hat{P}_r|$. Besides, there are at least three integer ranks $r_1<r_2<r_3$ and a real number $D^{\mathbb{R}^+}_{\text{KS}} \in [0,1]$ that satisfy $D^{\mathbb{R}^+}_{\text{KS}} \le |P_{r_i}$–$\hat{P}_{r_i}| \le D^{\mathbb{R}^+}_{\text{KS}} + \varepsilon$ with ε=$(f_{r_1}$+$\hat{f}_{r_1})/|D|$ and alternated signs for P_r–$\hat{P}_r$.*

Proof. Without ambiguity, we extend the domains of the complementary CDFs (CCDFs) of real-world and fitted distributions from integers to real numbers, and regulate the closed interval as $[1, |D|]$, where $|D|$ is the total number of passwords. Conceptually, the continuous function of a CCDF can be constructed via at most $|D|$–1 straight lines connecting two consecutive integers. Formally, we define $g(x)$ as

$$g(x) = \begin{cases} 1 - \hat{P}_x & x \in \mathbb{Z}^+ \\ 1 - \hat{P}_{\lfloor x \rfloor} + (P_{\lfloor x \rfloor} - P_{\lfloor x+1 \rfloor})(x - \lfloor x \rfloor) & x \notin \mathbb{Z}^+. \end{cases} \tag{18}$$

For $h(x) \in H$ characterized by C and s, it is a two-dimensional function and has at most one zero-point on the interval $[1, |D|]$. Since it can be similarly defined in a similar approach to the fitted CDF P_r, we omit it here. Based on Lemma 1, there exists $h^*(x) \in H$ that minimizes $e(x)$=$\max_{x \in [1,M]} |g(x)$–$h(x)|$ across $h(x) \in H$, and $x_1<x_2<x_3 \in [1,M] \subseteq \mathbb{R}^+$ that make $e^*(x)$=$g(x)$–$h^*(x)$ satisfy $e^*(x_i)$=–$e^*(x_{i+1})$ for $i \in \{1,2,3\}$. In this case, we take r_i=$\arg\max_{\{\lfloor x_i \rfloor, \lfloor x_i \rfloor + 1\}} |e(x)|$, i.e., the integer in adjacency of x_i that makes $|e(x)|$ larger, and have

$$\begin{aligned} D_{KS} &\le \max_{r_i \in \{r_1, r_2, r_3\}} |P_{r_i} - \hat{P}_{r_i}| = |g(r_i) - h^*(r_i)| \\ &\le |g(r_i) - g(x_i)| + |g(x_i) - h^*(x_i)| + |h^*(x_i) - h^*(r_i)| \\ &\le |e(x_i)| + (\hat{f}_{r_1} + f_{r_1})/|D| = D^{\mathbb{R}^+}_{KS} + \varepsilon \end{aligned} \tag{19}$$

where $D^{\mathbb{R}^+}_{KS}$ is the KS statistic on $[1, M]$. Clearly, when x_i=r_i, there is D_{KS}=$D^{\mathbb{R}^+}_{KS}$. Further, since x_i=$\arg\max_{x \in [1,M]} |g(x)$–$h^*(x)|$, $e^*(x)$ has no zeros between $\lfloor x_i \rfloor$ and $\lfloor x_i$+1$\rfloor$, so the signs of $e^*(x_i)$ and $e^*(r_i)$ are the same, and the sign of P_{r_i} – $\hat{P}_{r_i}$ is alternated. This concludes the proof. □

The above contents ensure the existence of optimal parameters that minimize the KS statistic. For normalized $\|D\|_p$ with $p \in \mathbb{Z}^+$ (see Eq. 2), the solution can be more complicated, and we leave them as open questions. The next Lemma 3 shows that GSS can accurately approximate the optimal solution.

Lemma 3. *Let e: $[x_{\min}, x_{\max}] \to \mathbb{R}$ be a deterministic unimodal function with a unique minimizer x_0=$\arg\min_x \{e(x) | x \in [x_{\min}, x_{\max}]\}$. A one-dimensional golden search can generate sub-intervals $[a_m, b_m]$ with a_0=$x_{\min}$, b_0=$x_{\max}$ and b_{m+1}–a_{m+1}=$\varphi\,(b_m$–$a_m)$ where φ=$(\sqrt{5}$–$1)/2$. Then $x_0 \in [a_m, b_m]$ for all m and $\lim_{m \to \infty} a_m$=$\lim_{k \to \infty} b_m$=x_0.*

Proof. Based on golden search, for any iteration number $m \in \mathbb{N}$, there exist $x_0 \in [a_m, b_m]$, $[a_{m+1}, b_{m+1}] \subseteq [a_m, b_m]$ and b_m–a_m=$(x_{\max}$–$x_{\min})\,\varphi^m$. By leveraging the nested intervals theorem of real numbers, $\bigcap_{m=1}^{\infty} [a_m, b_m]$ is a single point set and only x_0 can fall into it. This concludes the proof. □

Based on these results, Theorem 1 in Sec. III-B can hold.

We next explain why LLS [22] fails to hold completeness. For a given (C,s), we suppose P'_r is the CDF of the fitted model FitSet, and P_r=$C \cdot r^s$ is the explicit expression. Similarly, (C', s') is obtained from fitting FitSet with LLS. In this setting, there exists P'_r=$C \cdot r^s$+ϵ_r, with ϵ_r being the deviation caused by random sampling. By the Taylor series, we have

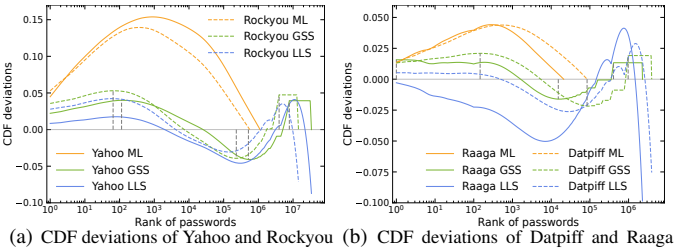
$$\log P'_r - \log C \cdot r^s = \frac{\epsilon_r}{C \cdot r^s} - \frac{\epsilon_r^2}{2(C \cdot r^s)^2} + O(\epsilon_r^3) = \eta_r + O(\epsilon_r^3). \tag{20}$$

We integrate with Eq. 16, and find the relative deviations have

$$\begin{cases} RD(s) = \left| \frac{s'-s}{s} \right| \approx \left| \frac{\sum_{r=1}^{N} (\log r - \overline{\log r})(\eta_r - \overline{\eta_r})}{s \sum_{r=1}^{N} (\log r - \overline{\log r})^2} \right|, \\ RD(C) = \left| \frac{C-C'}{C} \right| \approx \left| \exp\left(\overline{\eta_r} - \overline{\log r} \cdot (s' - s) \right) - 1 \right|. \end{cases} \tag{21}$$

As a result, for popular passwords with small $C \cdot r^s \approx 0$, η_r and $\overline{\eta_r}$ are enlarged by P_r=$C \cdot r^s$ in the denominator; For unpopular passwords with $C \cdot r^s \approx 1$, due to ranking operations (see Alg. 3) in constructing the fitted model FitSet, ϵ_r and thus η_r are large. Thus, neither $RD(C)$ nor $RD(s)$ can approximate 0.

We also conduct additional experiments to validate the above analysis. Fig. 4 shows how the CDF deviation (with no absolute value) between the fitted model and the real-world RealSet, i.e., e_r=P_r–$\hat{P}_r$. For GSS, at least three points can make $e_r \approx D_{KS}$, and the signs of e_r are alternated as Lemma 2 predicts. For instance, for Yahoo, the absolute CDF deviation reaches its maximum when the rank is $r \in \{112, 6.21 \times 10^5, 7.89 \times 10^6\}$, with signs positive, negative, and positive, respectively; In contrast, no such phenomena exist for ML [33] and LLS [22], strengthening the completeness of GSS. For LLS, Table X shows the detailed comparison of $RD(C)$, $RD(s)$, and the right-hand side of Eq. 21. First, the Taylor series in Eq. 20 is accurate, with a maximal relative



(a) CDF deviations of Yahoo and Rockyou datasets under the KS statistic.

(b) CDF deviations of Datpiff and Raaga datasets under the KS statistic.

Fig. 4: CDF deviations of ML [33], LLS [22] and GSS [73] when GSS is fitted under the KS statistic (see Eq. 1). For the GSS-fitted results, there are at least three ranks that make CDF deviations equal/near the KS statistic, and the signs of these deviations are alternated as Lemma 2 predicts.

difference of 0.73%∼3.29%. Second, the $RD(C)$ and $RD(s)$ values from Sec. III-B closely match their approximations in Eq. 21, with errors as small as 0.07%∼4.56%. This strengthens our explanation of why the LLS is not complete.

C. Details of offline guessing theories in Sec. V

Theorem 2. *Let v be the account value, k be the cost per guess, $0<\alpha\leq 1$ the diminishing return, and P_r, p_r be the CDF and PDF of the password distribution. If there exists*

$$1 > \frac{v}{k}p_1^{\alpha} \text{ for } r{=}1, \text{ or } 1 > \frac{v}{k}\min_{2\leq t\leq r}\frac{\alpha\cdot p_t}{(1-P_{t-1})P_{t-1}^{1-\alpha}} \text{ for } r{\geq}2, \quad (22)$$

then the profit-driven attacker will stop at the r–1-th unique password in RealSet *without trying to crack the r-th one.*

Proof. Under the profit-driven model (Sec. V-A), the attacker continues guessing if the marginal reward exceeds the marginal cost, that is, the attacker guesses as long as $v(P_r^{\alpha}-P_{r-1}^{\alpha})\geq k(1-P_{r-1})$ for all $r\geq 1$, i.e.,

$$v\cdot p_1^{\alpha}\geq k \text{ for } r=1, \quad \text{and} \quad \frac{v}{k}\geq \max_{2\leq t\leq r}\frac{1-P_{t-1}}{P_t^{\alpha}-P_{t-1}^{\alpha}} \text{ for } r\geq 2. \quad (23)$$

By taking the inverse of Eq. 23, we can obtain the condition for the attacker to stop guessing, i.e.,

$$1 > \frac{v}{k}p_1^{\alpha} \quad \text{for } r{=}1, \quad \text{or} \quad 1 > \frac{v}{k}\min_{2\leq t\leq r}\frac{P_t^{\alpha}-P_{t-1}^{\alpha}}{1-P_{t-1}} \text{ for } r\geq 2. \quad (24)$$

Due to $0<\alpha\leq 1$ and $P_t{=}P_{t-1}{+}p_t$ for $t\geq 2$, the expression $x^{\alpha-1}$ is decreasing, and we can obtain the following by applying the Lagrange mean-value theorem with a $\xi\in(P_{t-1},P_t)$,

$$\frac{P_t^{\alpha}-P_{t-1}^{\alpha}}{1-P_{t-1}}=\frac{\alpha\xi^{\alpha-1}p_t}{1-P_{t-1}}\leq\frac{\alpha\, p_t}{(1-P_{t-1})P_{t-1}^{1-\alpha}}, \quad (25)$$

where the equality holds when $\alpha{=}1$. In this way, once $1{>}\frac{v}{k}p_1^{\alpha}$ or $1>\frac{v}{k}\min_{2\leq t\leq r}\frac{\alpha\cdot p_t}{(1-P_{t-1})P_{t-1}^{1-\alpha}}$ can hold, so can Eq. 24 and the attacker stops guessing. This concludes the proof. □

Corollary 1. *If passwords are accurately characterized by the CDF-Zipf model [73] with the explicit $P_r{=}Cr^s$ (see Sec. III-D), $Det(r)$ is non-increasing, and $P_{r-1}{\approx}P_r$ holds when guessing stops, then the cracked proportion against $\mathcal{A}_2$ knowing the number of checked guesses Δ_r should fall into*

$$F_1(r)=\Big(\frac{1}{1+\frac{v\cdot s}{k\Delta_r(r-1)}}\Big)^{\frac{1}{\alpha}}<P_r<\frac{1}{1+(\frac{\alpha\cdot v\cdot s}{k\cdot\Delta_r r})^{\frac{1}{\alpha}}}=F_2(r). \quad (26)$$

TABLE X: Results on explaining why LLS fails to achieve completeness.

Dataset	Max RD in Eq. 20†	Parameter C			Parameter s		
		$RD(C)$‡	Approx. $RD(C)$‡	Error analysis	$RD(s)$‡	Approx. $RD(s)$‡	Error analysis
Yahoo	**1.01%**	39.03%	39.07%	**0.10%**	16.94%	16.78%	**0.98%**
Dodonew	**1.80%**	39.83%	39.96%	**0.32%**	13.52%	13.23%	**2.21%**
000Webhost	**2.69%**	37.31%	37.56%	**0.68%**	10.97%	10.60%	**3.45%**
Rockyou	**0.73%**	33.26%	33.29%	**0.08%**	15.83%	15.71%	**0.72%**
Tianya	**0.64%**	31.58%	31.61%	**0.07%**	15.60%	15.50%	**0.62%**
Duelingnetwork	**1.37%**	32.81%	32.91%	**0.31%**	11.73%	11.53%	**1.69%**
Chegg	**3.29%**	45.71%	46.03%	**0.67%**	12.93%	12.36%	**4.56%**
Wishbone	**1.53%**	37.41%	37.52%	**0.33%**	13.76%	13.51%	**2.21%**
Dailyquiz	**0.79%**	29.14%	29.18%	**0.15%**	10.94%	10.85%	**0.89%**
Datpiff	**1.17%**	33.35%	33.43%	**0.23%**	12.75%	12.57%	**1.41%**
Raaga	**1.64%**	35.84%	35.94%	**0.30%**	14.53%	14.26%	**1.85%**

† Max RD in Eq. 20 denotes the relative difference between the left- and right-hand sides of Eq. 20 that measures the accuracy of the Taylor series close to $C{\cdot}r^s$.
‡ $RD(C)$ and $RD(s)$ are relative differences of C and s (see Table III), and Approx denotes the result obtained via the right-hand side of Eq. 21.

Proof. When $\mathcal{A}_2$ stops, there is $\Delta_r{=}\frac{v}{k}Det(r)$. For the left-hand side, since $Det(r)$ and p_r are non-increasing, we have

$$\begin{aligned}\Delta_r &< \frac{v}{k}\frac{\alpha\cdot s\cdot\frac{C(r-1)^s}{r-1}}{(1-P_{r-1})P_{r-1}^{1-\alpha}}=\frac{\alpha\cdot v\cdot s}{k(r-1)}\frac{P_{r-1}}{(1-P_{r-1})P_{r-1}^{1-\alpha}}\\ &=\frac{\alpha\cdot v\cdot s}{k(r-1)}\frac{P_{r-1}^{\alpha}}{1-P_{r-1}}<\frac{\alpha\cdot v\cdot s}{k(r-1)}\frac{P_{r-1}^{\alpha}}{1-P_{r-1}^{\alpha}}\end{aligned} \quad (27)$$

Thus, we can obtain $P_r{>}P_{r-1}{>}\Big(\frac{1}{1+\frac{\alpha\cdot v\cdot s}{k\Delta_r(r-1)}}\Big)^{\frac{1}{\alpha}}$. Now, to prove the right-half of the inequality, we have

$$\begin{aligned}\Delta_r &> \frac{v}{k}\frac{\alpha\cdot s\cdot\frac{Cr^s}{r}}{(1-P_{r-1})P_{r-1}^{1-\alpha}}>\frac{\alpha\cdot v\cdot s}{k\cdot r}\frac{P_r}{(1-P_{r-1})P_r^{1-\alpha}}\\ &\approx\frac{\alpha\cdot v\cdot s}{k\cdot r}\frac{P_r^{\alpha}}{1-P_r}>\frac{\alpha\cdot v\cdot s}{k\cdot r}\Big(\frac{P_r}{1-P_r}\Big)^{\alpha}\end{aligned} \quad (28)$$

Hence, we can obtain $P_r<\frac{1}{1+(\frac{\alpha\cdot v\cdot s}{k\cdot\Delta_r r})^{\frac{1}{\alpha}}}$. This ends the proof. □

D. Memory purchase costs

In terms of MHF purchase cost, we use the 32GB DDR4 memory (about \$36[6]) as a typical example to estimate the amortized memory purchase cost whose unit is C_{H_P}. We first explain why the amortized memory purchase cost scales quadratically with the memory usage σ based on Blocki et al.'s work [21]. Suppose the lifespan of the memory is 2 years as in [21], and τ=3 and σ=1GB in [17]. Under this setting, Argon2i takes about 1s in its execution [17], so the amortized cost for 1GB memory in 1s is given as

$$\frac{\$36}{32\times 2\times 365\times 24\times 3600}=\$1.78\times 10^{-8}. \quad (29)$$

When σ=2GB, two terms in the denominator will change. First, the memory purchase cost is amortized to 2GB, so 32 is reduced to 16. Second, it takes 2s to run Argon2i for σ=2GB, so 3600 is reduced to 1800. Hence, the result should be $\$7.13\times 10^{-8}$, which is 4 times the cost of $\$1.78\times 10^{-8}$. *In this way, the memory purchase cost scales quadratically with the memory usage σ.* We also note that, in the first process of writing memory data (see Sec. IV-A), each memory unit (i.e., 1KB) stores and holds data sequentially, so the amortized memory is $\sum_{i=1}^{\sigma} i/\sigma{\approx}\sigma/2$; In the following memory iteration(s), Argon2i always occupies σKB. Hence, for τ=3 and σ=1GB=2^{20}KB, there is $C_{H_P}{\cdot}(7/2){\cdot}\ 2^{40}{=}\$1.78{\times}10^{-8}$, and we have $C_{H_P}{=}\$4.63{\times}10^{-21}$.

[6]See at Amazon http://tinyurl.com/384eyta2 . The actual price may change from time to time due to multiple non-technical factors.