Targeted Guessing Using Multiple Leaked Passwords

Yunkai Zou, Chi Zhang, Yuxuan Wu, Ding Wang
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*—While password reuse attacks have gained considerable attention, only a few studies have examined multi-password reuse attacks, where an attacker exploits the victim's multiple (e.g., at least two) leaked/sister passwords to guess her target password on another account. A key challenge in modeling multi-password reuse attacks is how to characterize users' password reuse behaviors and understand the inherent connections between a user's sister passwords, so as to generate the most effective password guesses when the allowed guess number is small.**

To address this challenge, we, for the first time, show that previous models on multi-password reuse guessing derive most of their advantages over single-password models from an implicit oracle-selection effect, rather than from genuinely modeling cross-password dependencies. Accordingly, we conduct a systematic empirical study to investigate the structural and semantic relationships among multiple leaked passwords from the same user. We further introduce the polynomial loss function during training, and develop PolyGuess, a multi-password guessing model that more accurately captures users' password reuse behaviors. By employing three large-scale datasets with over 3.5 billion real-world passwords, we demonstrate that PolyGuess is effective: Given the victim's one or more (i.e., 1–8) sister passwords (which are denoted as $\mathcal{L}_p$), PolyGuess achieves success rates of 9%–36% (20% on average) in guessing her target password (namely pw_{target}, where $pw_{\text{target}} \notin \mathcal{L}_p$) within 100 guesses, which is 11%–78% (28% on average) higher than its foremost counterpart. We also discuss the implications of our findings for password policies, management, strength estimation and credential checking.

I. Introduction

Text passwords remain the primary authentication method and are unlikely to be fully replaced in the foreseeable future, owing to their simplicity, ease of modification, and low deployment cost [5, 9, 10, 71]. Despite their ubiquity, passwords impose substantial usability burdens that encourage insecure practices. Among these, password reuse remains widespread and serves as one of the most common initial access vectors in real-world breaches [4, 5]. One important factor is the growing burden of password management. Recent NordPass research reports that users manage approximately 120 personal passwords and 67 work-related passwords on average in 2026 [57]. However, earlier studies [16, 19] found that users maintained only about 5.8–6.5 *distinct* passwords, highlighting the ever-intensifying conflicts between the growing number of passwords users manage and their limited repertoire of distinct passwords. As a result, users frequently reuse the same passwords or create slight variations across multiple platforms, prioritizing memorability over security [22, 55].

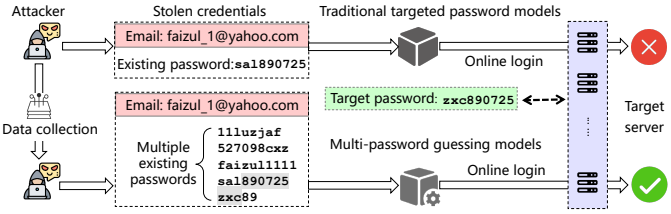


Fig. 1: Comparison between targeted password guessing models that exploit *only one* (see the upper half) and *all* (see the lower half) of the victim's existing/leaked passwords. The latter more naturally and realistically characterizes users' real-world and diverse password reuse behaviors.

The increasing frequency of large-scale password breaches [6, 14, 64] further exacerbates this problem. In January 2026, for instance, a publicly exposed database of approximately 149 million credentials was discovered, with roughly 48 million entries linked to Gmail accounts alone [18]. Critically, such datasets are rarely exploited in isolation. Attackers routinely exploit shared identifiers (e.g., emails) to correlate records across multiple breaches, effectively reconstructing a victim's historical password portfolio [23, 39, 72]. These aggregated datasets in turn enable targeted credential stuffing and credential tweaking attacks, which smartly exploit leaked passwords to compromise accounts across services [39].

Compounding this threat is the scale of multi-password leakage. A SurfShark report [3] reveals that 9.5 billion passwords have been leaked since 2004, compared to 3.7 billion unique email addresses, indicating that the same users are affected by multiple data breaches. Moreover, Pal et al. [39] and Huang et al. [23] provide statistical evidence showing that *23%-42% of users have leaked at least three passwords*. These findings indicate that multi-password leakage is not an edge case but an increasingly prevalent phenomenon affecting a considerable fraction of users. This reality should be explicitly accounted for when designing password guessing models for practical attack scenarios, otherwise we would likely underestimate the threat of real-world targeted guessing.

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.2027.232575
www.ndss-symposium.org

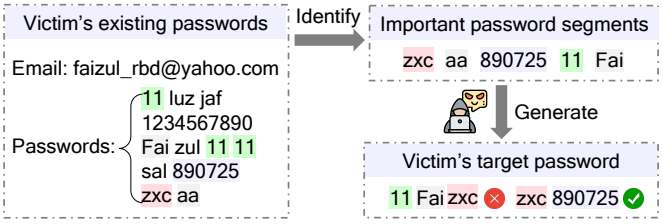


Fig. 2: Example of multi-password guessing. Some password segments (e.g., shared substrings) are highly informative for generating the target password, while others are weakly related and may introduce misleading signals.

A. Motivation and design challenges

Existing studies on password reuse (e.g., [12, 30, 39, 60, 61, 68]) mainly focus on the *single-password* reuse attack, where an attacker exploits *one* leaked password (i.e., a sister password used at one service) of a victim to guess her passwords on other services. While this line of work has been extensively studied, it does not fully capture realistic attack scenarios, in which multiple passwords of the same user have been leaked (see Figure 1 for an example).

Intuitively, the availability of multiple leaked passwords provides richer prior knowledge about a user's password creation behavior. For example, Pearman et al. [45] and Ur et al. [55] find that users tend to select a small number of "base passwords" and derive other passwords by making slight modifications. These characteristics offer greater potential for improving guessing effectiveness. Recent work has begun to explore multi-password guessing, showing that incorporating multiple inputs can significantly improve success rates [23, 65, 72]. However, a fundamental question remains largely unexplored: *why* do multiple passwords help?

Particularly, it is unclear whether improvements in multi-password guessing stem from genuinely modeling cross-password dependencies, or simply from an increased likelihood that one input happens to be highly similar to the target. If the latter dominates, existing multi-password models (e.g., PassLLM-II [72]) may effectively behave as noisy single-password models, implicitly relying on some form of oracle selection rather than well exploiting the multi-password structures and semantics. Understanding this distinction is critical, as it directly affects how models should be designed to reliably benefit from multiple inputs. To systematically investigate this problem, we formulate three research questions:

- **RQ1:** What are the structural and semantic relationships of multiple leaked passwords from the same user?
- **RQ2:** How do existing single-password reuse models behave when applied to multi-password settings?
- **RQ3:** What is the true source of success-rate gains in existing multi-password guessing approaches?

When tackling these questions, we reveal two fundamental challenges in multi-password reuse modeling. First, the leaked passwords of a user can vary substantially in their relevance to the target (see Figure 2 for a toy example): while some share structural or semantic characteristics with the target, the remaining passwords may introduce noise and mislead the model. Prior work on targeted guessing has demonstrated that incorporating irrelevant auxiliary inputs (e.g., employing more types of personally identifiable information) can even *degrade* model effectiveness [60, 61], and this risk is amplified when the informative subset of sister passwords is unknown a priori.

Second, successful password guessing in reuse-based attack scenarios largely depends on relatively subtle but critical transformations (e.g., altering capitalization, or substituting characters) made to existing passwords [12, 60, 61]. In the three datasets analyzed, these transformations are sparse and heterogeneous, whereas copying behaviors (in which large portions of existing passwords are preserved) are more frequent (see Section III-C). While copying is easy for models to learn, it is insufficient for exact-match success: Guessing success hinges on correctly predicting the few decisive positions that differ. Under standard training objectives, the abundance of easy copying positions may dominate the learning signal, leaving mutation positions (target character positions that require edits rather than direct copying from the aligned source password) relatively underweighted.

Both challenges suggest that effectively modeling multi-password reuse requires both (i) robust aggregation of multiple inputs under uncertainty, and (ii) careful allocation of learning signals across different position types. However, existing multi-password models (e.g., Pass2Pass-T [23] and PassLLM-II [72]) primarily rely on input aggregation and do not explicitly address this potential optimization imbalance.

To address both challenges, we first conduct a large-scale empirical study to characterize the multi-password reuse patterns present in three real-world datasets. Within these datasets, password sets exhibit both global diversity and strong local similarity, while exact-match guessing depends disproportionately on sparse but critical transformations. Building upon these findings, we introduce a training objective that aims to rebalance learning signals between frequent copying and sparse transformations, and develop PolyGuess, a multi-password guessing model tailored to these characteristics.

B. Contributions

We summarize our main contributions as follows:

- **A systematic characterization of multi-password reuse behaviors.** We introduce a measurement framework of nine similarity-, structural-, and target-oriented metrics to quantify relationships among passwords linked to the same user. Across the three large-scale datasets studied, we observe *global diversity together with strong local similarity*: Even when a user's passwords are broadly dissimilar, non-trivial pairwise and substring-level overlaps frequently emerge, allowing multiple sister passwords to *collectively* cover a large fraction of the target password.
- **A loss function tailored to multi-password reuse**. We apply a polynomial loss [28] that aims to capture the copy-dominant yet mutation-critical nature of password reuse, rebalancing optimization weight across positions. To the best of our knowledge, this is the first loss tailored to the structural properties of multi-password reuse.

- **Extensive evaluation.** We construct three large-scale mixed datasets, comprising over one billion real-world passwords. Extensive experiments across six representative attack scenarios demonstrate the effectiveness of PolyGuess. Within 1,000 guesses, our PolyGuess outperforms state-of-the-art password models, and particularly, within 100 guesses, it achieves 11%-78% higher success rates than its foremost counterpart.
- **Some insights.** First, much of the gain from multiple inputs in representative existing models (e.g., [65, 72]) can be explained by an implicit oracle-like selection effect: using one oracle-selected source yields guessing success rates close to those obtained with all sister passwords. Second, PolyGuess is more effective at guessing target passwords that are not highly similar to any individual input (Sec. VI-B), suggesting that it captures the relationships among multiple sister passwords. Third, we provide large-scale empirical evidence that tends to invalidate password policies like password history checking and mandatory periodic password changes, and highlights the importance of considering threats arising from attackers with users' multiple leaked passwords when conducting password strength estimation and credential checking.

II. Background and Threat Model

In this section, we present the related work on password reuse, including both single-password reuse attacks and multiple-password reuse attacks.

A. Single-password reuse attacks

In 2014, Das et al. [12] systematically studied cross-site password reuse using large-scale leaked datasets and user surveys. They show that 43%–51% of users directly reuse passwords across sites, while a significant portion of the remaining passwords are generated through simple, highly predictable transformations (e.g., appending digits or symbols). Based on these observations, they propose the first cross-site password guessing algorithm that exploits sister passwords and transformation patterns, achieving substantially higher guessing success rates (e.g., 30% within 100 guesses) than traditional trawling guessing algorithms (e.g., PCFG [63]).

In 2016, Wang et al. [60] proposed TarGuess-II, which for the first time introduced statistical rigor into cross-site guessing, by using probabilistic context-free grammars and Markov n-grams to learn structure-level and segment-level reuse behaviors, respectively. Compared with Das et al.'s algorithm [12], TarGuess-II captures context-dependent transformation behaviors and achieves much higher guessing success rates.

In 2019, Pal et al. [39] proposed Pass2Path, the first work that introduces deep learning techniques into password reuse modeling. Built upon a sequence-to-sequence architecture, Pass2Path models password transformations as a sequence of atomic edit operations (e.g., insertion and deletion), taking a user's sister password as input and generating the corresponding edit sequence. In 2023, Wang et al. [61] pointed out that each atomic edit operation may influence subsequent operations in the password transformation process. They further formulated password reuse as a multi-class classification task with a multi-step decision mechanism, and proposed Pass2Edit [61]. In the same year, Xu et al. [68] proposed PassBERT, pioneering the use of Transformer-based language models for password generation and reuse modeling.

Following this line of research, a series of single-password reuse models have been proposed. Representative approaches include RFGuess-Reuse [62] based on random forests and PointerGuess [65] leveraging pointer networks. More recently, models such as PassLLM-II [72], which uses large language models (LLMs), and KNNGuess [30] based on k-nearest-neighbor reuse retrieval, have further advanced the effectiveness of password reuse attacks.

Intuitively, existing single-password reuse models can be extended to multi-password guessing scenarios. For example, Pal et al. [39] explored a heuristic approach that independently generates guess lists from multiple passwords and merges them in a round-robin manner. While this strategy improves guessing success rates compared to using a single password, it fails to capture the structural and semantic relationships among a user's multiple passwords, reflecting a fundamental limitation of single-password models.

Wang et al. [61] show that dissimilar password pairs can introduce noise and negatively affect model learning. This suggests that, with appropriate preprocessing of the training data (e.g., selecting the most similar password to the target from a user's sister passwords as input), the guessing success rate of such models can be further improved. We validate this through experiments in Section V-D.

B. Multiple password reuse attacks

In 2024, Xiu and Wang [65] introduced MS-PointerGuess, an extension of PointerGuess that adopts a multi-encoder architecture to model two sister passwords simultaneously. While it achieves higher guessing success rates than its single-password counterpart, PointerGuess, its scalability is limited by the varying number of passwords per user. More specifically, the number of encoders must scale with the number of available sister passwords, leading to increased model complexity and reduced computational efficiency.

In 2025, Zou et al. [72] proposed PassLLM-II, which utilizes multiple sister passwords of the same user to predict target passwords across sites, achieving substantially higher guessing success rates than prior models such as Pass2Path [39] and PointerGuess/MS-PointerGuess [65]. More recently, Huang et al. [23] proposed Pass2Pass-T, a Transformer-based model that concatenates multiple leaked passwords from the same user into a single sequence, and applies repetition-aware compression and segmented positional encoding to capture cross-password reuse patterns. With five leaked passwords, the model achieves higher guessing success rates than MS-PointerGuess [65], particularly within the first 1-5 guesses, and generalizes to varying numbers of leaked passwords.

However, both PassLLM-II [72] and Pass2Pass-T [23] lack a systematic analysis of where such gains originate. Moreover, they do not well explore the potential of existing single-password reuse models (e.g., Pass2Path [39] and Pass2Edit [61]) when extended to multi-password scenarios.

C. Password reuse behaviors

Besides password guessing models, a large body of research has examined password reuse from behavioral, measurement, and system-security perspectives, providing important context for understanding real-world threats.

Early studies established the pervasiveness of password reuse and linked it to users' limited ability to manage large password portfolios. In 2007, Florêncio and Herley [16] conducted the first large-scale measurement study, showing that on average, each user maintains about 25 accounts but only 6.5 distinct passwords. Such behaviors are further driven by bounded rationality, where users knowingly trade security for memorability and convenience, and tend to underestimate their own reuse practices [22].

Subsequent research has explored factors influencing reuse behaviors. For example, Pearman et al. [45] showed that password characteristics (e.g., length) and website properties (e.g., category, perceived security level) significantly affect reuse decisions. In addition, studies on password managers reveal mixed effects: while dedicated third-party managers can reduce reuse, a substantial fraction (e.g., approximately 22%-47% among some password-manager-assisted settings) of users still reuse passwords in practice, driven by usability preferences and habitual behaviors [32, 34].

From a security perspective, password reuse has been identified as the key enabler of credential stuffing attacks. For instance, a longitudinal study over two decades found that reused passwords (and their slight modifications) enable successful guessing for a substantial fraction of accounts (e.g., up to 32%), far exceeding common-password guessing strategies [38]. Similarly, Sahin et al. [46] demonstrated that system design choices such as typo-tolerant authentication can further amplify the effectiveness of credential stuffing and password tweaking attacks.

To mitigate these risks, prior work has proposed various defenses, including compromised credential checking services [40], privacy-preserving cross-site detection protocols [53], and password reuse-aware strength meters [39, 58]. More recently, proactive approaches have been explored to explicitly model reuse relationships across websites. For example, in 2024, PassREfinder represents inter-site password reuse as a graph and predicts credential stuffing risks using graph neural networks [26].

D. Threat model

We consider a targeted *online* attacker who has obtained one or more (e.g., $\geq$2) leaked passwords associated with a victim, and submits model-ranked guesses to the authentication interface of another account/service used by that victim. Large-scale breach corpora (e.g., Cit0day [2], COMB [1], and Naz [6]) contain email–password pairs that may be linked through shared identifiers to assemble a set of passwords associated with the same user [39, 61, 68, 72]. The target password is not included in the attacker's known password set.

We evaluate thresholds from 10 to 1,000 guesses to characterize both tightly constrained and sustained low-rate attacks. While online services generally deploy rate limiting and account lockout mechanisms, previous studies show that practical online guessing thresholds can still reach 100 or 1,000 attempts. For example, Wang et al. [59] measured the rate-limiting mechanisms of the Alexa top-10 websites and estimated that each allowed $\geq$3,600 attempts per month (120 per day; see Table 7 in their paper). Without loss of generality, we set the online guessing threshold to 10^3, which is consistent with major online guessing studies [60, 61, 65, 72].

To capture variation in the amount of prior knowledge, we construct attack scenarios with different numbers of available sister passwords (e.g., 1–4 and 5–8). For each user, the attacker exploits all available inputs without knowing in advance which password is most informative for the unseen target.

III. Characterizing intra-user password relationships

To better understand how users manage multiple passwords, we conduct a comprehensive intra-user analysis over users with multiple leaked passwords. More specifically, we aim to characterize the extent to which structural and semantic consistency is preserved (**RQ1**).

A. Multi-password dataset construction and statistics

We construct our dataset from three large-scale leaked password corpora. As summarized in Table I, these datasets are Cit0day [2], COMB [1], and Naz [6]. Cit0day is a large-scale aggregated breach dataset containing email address-password pairs compiled from numerous historical data leaks, and is widely used for password security research [41, 66, 67, 69]. COMB (Compilation of Many Breaches) is one of the largest credential compilations, aggregating over 3.2 billion unique email–password pairs from around 252 prior breaches [1]. Naz is a recent large-scale leak dataset involving over 70 million accounts [6], incorporating newer breach data and thus better reflecting contemporary password usage patterns.

Data cleaning. To ensure data quality, we apply a preprocessing pipeline following prior password reuse studies (e.g., [39, 61, 65]). More specifically, we remove records with invalid email formats and filter out passwords containing non-printable, non-ASCII, or corrupted characters. We further discard passwords that are excessively short or long (e.g., shorter than 4 or longer than 30 characters), as well as strings that show abnormal patterns such as long hexadecimal substrings, which are unlikely to correspond to human-generated passwords. In addition, we identify and remove accounts associated with an unusually large number of passwords (e.g., thousands), as these are unlikely to be passwords of real users. Overall, this cleaning procedure removes approximately 0.68%-2.97% of the data, with negligible impact on the overall conclusions.

TABLE I: Details of password datasets leaked from various web services and data cleansing ("PWs" stands for passwords).

Dataset	Language	Leaked Time	Original PWs	Unique PWs	Invalid emails	Invalid PWs	Removed %	After cleaning
Naz	Mixed	Sep. 2023	138,688,172	42,085,995	566,034	375,837	0.68%	137,746,301
COMB	Mixed	Feb. 2021	3,279,064,312	855,833,811	81,542,117	15,718,941	2.97%	3,181,803,254
Cit0day	Mixed	Nov. 2020	158,527,650	63,820,703	1,099,504	2,631,091	2.35%	154,797,055

TABLE II: Statistics of user-count distribution by password number in different multi-source datasets ("PW" stands for password).

# PW per user	Dataset	COMB	Cit0day	Naz
	User accounts	1,434,791,054	92,824,048	33,248,591
2	Original†	228,454,449 (15.92%)	15,429,632 (16.62%)	5,396,162 (16.23%)
	Deduplicated	199,542,328 (13.91%)	4,001,385 (4.31%)	4,826,384 (14.52%)
3	Original	83,902,473 (5.85%)	4,650,823 (5.01%)	2,608,264 (7.84%)
	Deduplicated	64,207,383 (4.48%)	420,363 (0.45%)	1,913,914 (5.76%)
4	Original	39,162,102 (2.73%)	1,878,467 (2.02%)	1,334,455 (4.01%)
	Deduplicated	26,679,297 (1.86%)	98,605 (0.11%)	1,057,715 (3.18%)
5	Original	20,343,699 (1.42%)	916,543 (0.99%)	878,476 (2.64%)
	Deduplicated	12,472,550 (0.87%)	36,933 (0.04%)	656,087 (1.97%)
≥ 6	Original	77,883,371 (5.43%)	2,404,305 (2.59%)	4,520,687 (13.60%)
	Deduplicated	30,684,773 (2.14%)	72,613 (0.08%)	1,947,817 (5.86%)

† "Original" denotes the original statistics, while "Deduplicated" denotes the statistics after removing duplicated passwords per user.

TABLE III: Metric applicability by user password count. ✓: fully applicable; △: applicable with caveats; —: not applicable or degenerate.

Metric	Form†	m=2	m=3	m=4	m=5	$m\geq$6
MeanSim	$\frac{2}{m(m-1)}\sum s_{ij}$	✓	✓	✓	✓	✓
MaxSim	$\max s_{ij}$	✓	✓	✓	✓	✓
VarSim	$\mathrm{Var}(s_{ij})$	—	△	✓	✓	✓
PivotSim	$\frac{1}{m-1}\sum s_{*j}$	—	△	✓	✓	✓
TemplateCons	$1-\frac{H(t)}{\log m}$	✓	✓	✓	✓	✓
LenCV	$\frac{\sigma(\lvert pw\rvert)}{\mu(\lvert pw\rvert)}$	✓	✓	✓	✓	✓
MaxLCSCov	$\max \frac{\lvert\mathrm{LCS}(pw_i,pw_j)\rvert}{\min\lvert pw\rvert}$	✓	✓	✓	✓	✓
MS-PosCopyRate	$\frac{\lvert\mathcal{T}^*\rvert}{\lvert pw_{\text{target}}\rvert}$	✓	✓	✓	✓	✓
MutRatio	$\frac{\mathrm{Lev}(pw_{\text{src}},pw_{\text{target}})}{\lvert pw_{\text{target}}\rvert}$	✓	✓	✓	✓	✓

† m denotes the distinct-password count before target holdout; s_{ij}: pairwise similarity (2-gram cosine) between pw_i and pw_j; $*$ denotes the pivot password (the one with highest average similarity to others); $H(t)$: entropy of template distribution (templates map characters to classes L/D/S); LCS: longest common substring. For target-oriented metrics, one password is randomly held out as pw_{target}, and the remaining $m-1$ passwords form $P\setminus\{pw_{\text{target}}\}$. $\mathcal{T}^*$ denotes the set of target positions supported by at least one sister password via filtered LCS alignment, and pw_{src} denotes the most similar password in $P\setminus\{pw_{\text{target}}\}$ to the target password pw_{target}.

User matching and password aggregation. Since our focus is on relationships among passwords associated with the same user across breach sources, a critical step is linking credential records that likely belong to the same individual. Following Pal et al. [39], we first identify candidate cross-source account groups whose email prefixes match. We then retain a candidate group only if its matched records additionally share at least one identical password. This hybrid criterion combines email-prefix agreement with password overlap to reduce spurious associations. Exact full-email matching provides a strong linkage signal but may miss users who employ different email providers, whereas email-prefix matching can link some accounts missed by full-email matching at the risk of merging unrelated users with identical prefixes. The shared-password requirement thus serves as a conservative cross-check, balancing false merges against missed associations as with [39]. We aggregate the passwords in each retained group into a unified password set.

Multi-password leakage statistics. Table II and Figure 15 (in Appendix) show that multi-password leakage is widespread across all three datasets. We report statistics both before and after per-user password deduplication. The original statistics retain duplicate occurrences of the same password across breaches and thus characterize the raw extent of password exposure, whereas the deduplicated statistics count only distinct passwords and are used in the subsequent set-level relationship analyses. Based on the original datasets, *27.23%–44.32% of users have at least two leaked passwords*. More specifically, 15.92%–16.62% of users have exactly two leaked passwords, 5.01%–7.84% have exactly three, 2.02%–4.01% have exactly four, and 0.99%–2.64% have exactly five; a further 2.59%–13.60% have at least six. Consequently, users with four or more leaked passwords account for 5.60%–20.25% of the population. These results reveal a highly skewed, long-tailed distribution: While users with two or three leaked passwords constitute the largest groups, a non-negligible fraction have substantially larger exposed password portfolios.

B. Metrics for characterizing multi-password relationships

Notation. Let $P=\{pw_1,\ldots,pw_m\}$ denote the set of m sister passwords belonging to user u, including the held-out target.

To quantitatively characterize the structural and semantic characteristics among multiple passwords of the same user, we define nine metrics (see Table III). These metrics are organized into four categories that capture complementary aspects of users' multiple passwords: (i) *similarity-based metrics* that summarize pairwise password similarities (i.e., MeanSim, MaxSim, VarSim, and PivotSim); (ii) *structural metrics* that capture format-level consistency (i.e., TemplateCons and LenCV); (iii) *content-level metrics* that capture substring reuse and local overlap (i.e., MaxLCSCov); and (iv) *target-oriented metrics* that characterize how a target password relates to the remaining passwords under realistic attack scenarios, including multi-source positional support (i.e., MS-PosCopyRate) and single-source transformation cost (i.e., MutRatio).

The first three categories are computed from the complete password set, and the target-oriented metrics are defined by holding out one password as the target and treating the remaining passwords as the available context. As a result, the reliability of pairwise metrics depends on the total number of distinct passwords per user (i.e., m). We therefore specify, for each metric, the minimum m required for meaningful interpretation, and group users by password count ($m=2,3,4,5$ and $m\geq 6$) to ensure comparability.

Note that the aggregated breach corpora do not provide reliable password-creation timestamps, so this holdout is not chronologically ordered. Accordingly, our setting evaluates *password-set completion*: predicting an unseen, held-out member of a user's observed password set from the remaining members. It does not evaluate chronological prediction of a password created by the user at a later time.

In the following, we introduce the intuition behind these metrics and how they relate to multi-password reuse. Formal definitions and details are provided in Appendix B.

Similarity measure. In 2023, Wang et al. [61] demonstrated that 2-gram cosine similarity provides a more effective measure of password similarity than edit distance, as it better captures shared structural patterns in password composition. Therefore, we adopt this metric to quantify the similarity between password pairs. More specifically, we represent each password as a sequence of key-presses on a standard US keyboard using the `word2keypress` transformation (as with [39, 61]). In this representation, each character is mapped to its corresponding keyboard input, where special keys such as `Shift` and `CapsLock` are explicitly encoded.

Based on the resulting key-sequence, we extract character-level 2-gram multisets and represent each password as a vector $\mathbf{v}(pw)$ in the 2-gram space, where each dimension corresponds to a distinct 2-gram and the value denotes its frequency. The similarity between pw_i and pw_j is then computed by:

$$s_{ij} = \cos(\mathbf{v}(pw_i), \mathbf{v}(pw_j)) = \frac{\mathbf{v}(pw_i) \cdot \mathbf{v}(pw_j)}{\|\mathbf{v}(pw_i)\| \cdot \|\mathbf{v}(pw_j)\|}$$

By construction, $s_{ij} \in [0, 1]$: a value of 0 indicates no shared 2-gram structure, while 1 indicates proportional 2-gram frequency vectors.

Similarity-based metrics. We consider two representative statistics: *MeanSim*, which reflects the overall (global) similarity within a password set, and *MaxSim*, which captures the strongest (local) similarity between any pair of passwords. These two metrics distinguish globally diverse password sets from those showing strong localized reuse.

Structural metrics. To examine whether password reuse manifests at the format level, we consider coarse-grained structural patterns, including character-class templates (i.e., letter/digit/symbol) and length variability. These metrics capture whether users maintain consistent structural formats across passwords, independent of specific character choices.

Content-level metrics. Besides coarse-grained structural patterns, we further examine reuse at the content level by measuring contiguous substring overlap between passwords. In particular, *MaxLCSCov* captures the extent to which a long contiguous segment (e.g., a keyword) is preserved across passwords. This metric complements similarity-based measures by explicitly focusing on local content reuse, which may not be fully captured by order-insensitive similarity metrics.

Target-oriented metrics. To better reflect realistic attack scenarios, we further introduce target-oriented metrics by holding out one sister password as the target and treating the remaining ones as available context. Particularly, *MS-PosCopyRate* measures the fraction of target positions that can be explained by combining substrings from multiple source passwords, capturing multi-source positional support. In contrast, *MutRatio* measures the Levenshtein edit distance from the most similar source password to the target, normalized by the target password length.

Robustness to the user-linkage criterion. Before interpreting the measurement results, we examine whether the shared-password requirement in our original user-matching process biases the distributions of the nine metrics. We repeat the complete nine-metric analysis on Naz using exact full-email matching, without requiring any shared password. The resulting distributions across the metrics are highly consistent with those obtained using our original criterion, suggesting that the main observations are not substantially driven by selecting users who already exhibit direct reuse. See Figures 17(a)–17(i) for the corresponding distributions.

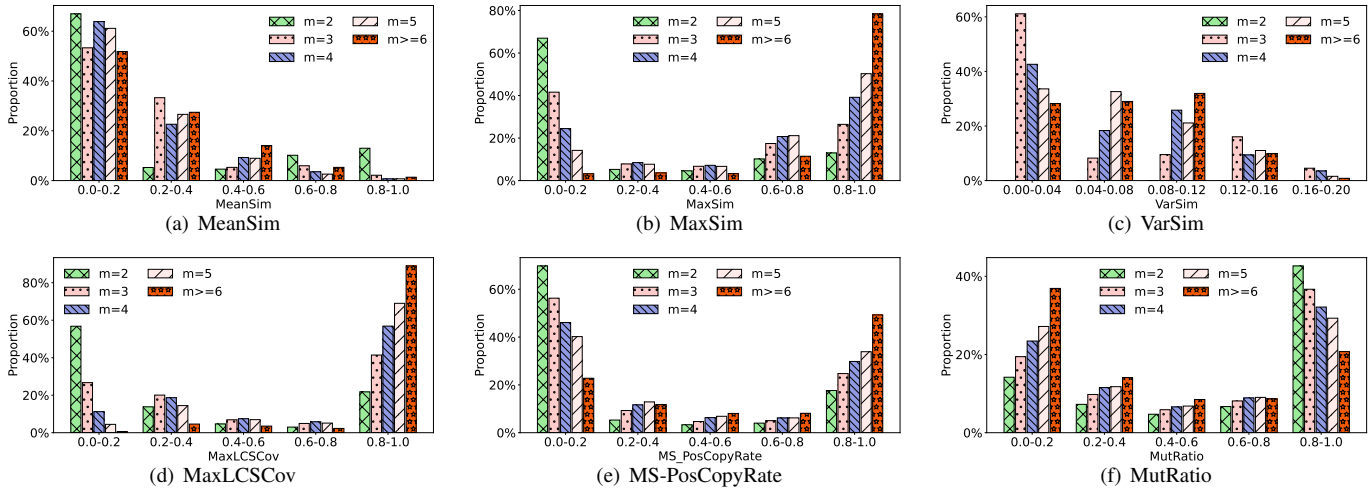
C. Characteristic analysis

We analyze multi-password behaviors on a unified dataset constructed from Cit0day, COMB, and Naz. Detailed analyses are provided in Appendix C. Here, we summarize four key findings. For readability, we group metric scores into coarse bins. Each range corresponds to qualitatively different reuse patterns, which we illustrate with examples in Table VIII.

Scope of analysis. All analyses are conducted on *deduplicated* password sets, where identical passwords within each user are merged and only one instance is retained. The reported statistics therefore capture structural and semantic characteristics among *distinct* passwords, excluding trivial direct reuse and focusing on meaningful transformations relevant to more challenging and security-critical password tweaking. Due to the large scale of the dataset (over 10^8 passwords), we randomly sample 50,000 users per group to compute the reported descriptive statistics. Empirically, we observe that the results stabilize once the sample size exceeds 20,000.

Finding I: Larger password sets increasingly expose strong local reuse despite low global similarity. While global similarity remains low across most password sets (e.g., over $\sim$50% of users have $\text{MeanSim}_u \in [0.0, 0.2]$ even at large m; see Figure 3(a)), the probability of containing at least one highly similar pair rises sharply with m (e.g., $\sim$78% of users have $\text{MaxSim}_u \geq 0.8$ at $m \geq 6$; see Figure 3(b)). Thus, larger password portfolios remain globally diverse while increasingly exposing strong local reuse. This helps explain why existing multi-password models such as PassLLM-II [72] can be effective: with more sister passwords, the model is more likely to receive a highly informative input, implicitly benefiting from a source-selection effect (see Section VI-C). From an attack perspective, this creates a noisy-oracle-like setting in which informative reuse evidence is present but its source is unknown a priori, motivating models that can identify and exploit such evidence without oracle source selection.

Finding II: Similarity heterogeneity increases with password-set size but remains bounded. Across the sampled m-strata, the similarity-variance distribution shifts from



(a) MeanSim (b) MaxSim (c) VarSim

(d) MaxLCSCov (e) MS-PosCopyRate (f) MutRatio

Fig. 3: Key characterization results of multi-password behaviors across users and m-strata. The results show that global similarity remains low (MeanSim), while local reuse (MaxSim), similarity heterogeneity (VarSim), and content-level overlap (MaxLCSCov) increase with m (i.e., the total distinct-password count). Meanwhile, higher MS-PosCopyRate and lower MutRatio indicate that, in larger observed password sets, a greater fraction of target positions can be supported by the available inputs and the closest source tends to require fewer modifications, while exact-match reconstruction still depends on the remaining sparse transformations. Additional metric distributions are provided in Figure 11, and detailed numerical results in Table VII (both in the Appendix).

a low-dominant regime toward greater moderate-variance mass: from $m{=}3$ to $m{\geq}6$, the proportion of users with $\text{VarSim}_u{\in}[0, 0.08]$ decreases from 69% to 57%, while the proportion in $[0.08, 0.16]$ rises from 26% to 42% (see Table VII). High-variance cases ($\text{VarSim}_u \in [0.16, 0.20]$) remain rare ($\leq 5\%$). Thus, larger observed portfolios are associated with more moderate heterogeneity without a corresponding increase in extreme divergence.

Finding III: Local content overlap is more prominent than global structural consistency. High-overlap cases are more prevalent in the larger-m strata (e.g., $\sim$89% of users have $\text{MaxLCSCov}_u \in [0.8, 1.0]$ at $m \geq 6$; see Figure 3(d)), indicating that these sets often contain a pair sharing a long contiguous substring. In contrast, template-level consistency remains low and shows no monotonic trend (see Figure 11(b) in the Appendix). These set-level observations suggest that fine-grained character-level dependencies may provide a more informative modeling signal than coarse global templates, motivating our emphasis on character-level reuse patterns in the subsequent model design.

Finding IV: Exact reconstruction depends on sparse but decisive mutation positions. In the smaller-m strata, the MutRatio_u distribution is polarized between low- and high-mutation cases (see Figure 3(f)); in the larger-m strata, low mutation ratios are more prevalent, meaning that the held-out target more often has a closely related source in the observed set. Multi-source positional coverage is also higher in the larger-m strata (see Figure 3(e)), which is consistent with different inputs offering complementary positional evidence. Since copied positions dominate these alignments while exact reconstruction still depends on the remaining transformations, mutation positions are sparse yet disproportionately important. This imbalance motivates training objectives that aim to prevent abundant copying positions from dominating the learning signal and instead allocate sufficient emphasis to the mutation positions that determine exact-match success.

Statistical validation. Using the full Naz dataset, we apply Cochran–Armitage trend tests to representative set-level conditions derived from our findings. All tested conditions show statistically significant monotonic trends across m-strata ($p < 0.01$), providing statistical support for the observed associations between password-set size and the corresponding reuse metrics (see Appendix A).

Sensitivity to service risk. Password choice may depend strongly on perceived account risk: users may invest more effort in protecting high-value services (e.g., banking) than low-sensitivity applications [45]. Since the three public datasets do not provide reliable service context for each password, we cannot directly stratify passwords by the value of the service on which they were used. As a practical but imperfect proxy, we identify a subset of COMB accounts associated with government-related (e.g., `.gov`) or bank-related email domains and compare their distributions across the nine metrics with the aggregate distributions in our main analysis. The metric distributions are generally consistent, suggesting that the observed copying and transformation patterns are also present among accounts potentially associated with higher-value organizations. See Figures 17(j)–17(r) in the Appendix for the nine-metric distributions of this proxy subset.

Scope and external validity. Our findings characterize multi-password copying and transformation patterns observed across three large-scale leaked-credential datasets. Since the provenance, collection context, user populations, and service context of these public datasets cannot be fully verified, it remains unclear whether the observed patterns generalize beyond the datasets and associated populations studied. Confirming that interpretation requires controlled user studies that directly examine how users construct passwords under different service-

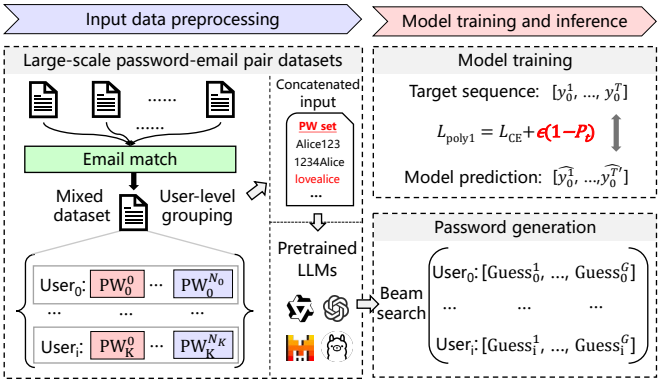


Fig. 4: Overview of the PolyGuess framework. Multiple passwords associated with the same user are first grouped and serialized into structured inputs. A pretrained language model then performs conditional password generation and is optimized with the Poly-1 loss, which strengthens the learning signal in the intermediate-confidence regime. At inference time, beam search generates a ranked set of password guesses for each targeted user.

risk contexts. We leave this as an important next step.

Implications for model design. The above findings translate into two design requirements. First, the noisy-oracle structure requires a model to identify informative reuse signals from multiple inputs without knowing in advance which sister password is most relevant to the target. Second, the copy-dominant yet transformation-critical structure calls for a training objective that aims to prevent abundant easy-copy positions from overshadowing sparse but decisive mutation positions. These requirements motivate our concatenation-based input representation and Poly-1 loss, respectively.

IV. PolyGuess: A Multi-Password Guessing Model

In this section, we develop PolyGuess (see Figure 4) by following the implications of Section III-C. We first validate the measurement-derived training implication through a position-level analysis of PassLLM-II [72]. We then formalize multi-password conditional generation for aggregating noisy inputs and finally derive Poly-1 to emphasize the sparse mutation positions that constitute the prediction bottleneck.

A. From measurement findings to training evidence

Section III-C shows that copying accounts for most target positions, whereas sparse transformations are disproportionately important for exact matches. To determine whether this measurement-level imbalance is also a learning bottleneck, we conduct a fine-grained position-level analysis using the strong multi-password baseline PassLLM-II [72]. The statistics are computed on samples successfully cracked within 1,000 guesses, using the model's top-1 character predictions. For each prediction, we align the target with its most similar source and categorize positions as *copy* (direct matches) or *mutation* (requiring edit operations). For example, in `password123` → `Password!23`, `assword` and `23` are copy positions, whereas the initial capitalization (`p`→`P`) and substitution (`1`→`!`) are mutation positions.

The gap is pronounced: PassLLM-II [72] achieves 93.15% accuracy on copy positions but only 1.71% on mutation positions. While mutation positions account for just 14.10% of characters, they contribute 70.22% of prediction errors. Thus, the sparse transformations identified by our measurement study are also the principal position-level bottleneck of an existing multi-password model. This evidence motivates a training objective that introduces a confidence-dependent correction to the learning signal, with the aim of improving predictions at difficult but informative mutation positions.

B. Multi-password conditional generation

Multi-password reuse guessing poses a practical challenge: useful information is unevenly distributed across inputs, and the most informative password is unknown *a priori*. As shown in Section III-C, password sets show global diversity with strong local reuse, requiring models to identify which inputs are useful for predicting the target.

However, our empirical results show that explicit input selection is unnecessary. PassLLM-II [72] already performs well without selection (see Figure 10). Moreover, under an oracle setting (where the most similar password is selected), using only a single input achieves guessing success rates comparable to using all inputs. This suggests that such LLM-based multi-password models can use information from relevant passwords even when other inputs are less similar to the target.

Based on this observation, we adopt a simple concatenation-based paradigm that uses all available (leaked) passwords as input without selection. We further find that guessing success rate is largely insensitive to input ordering (see Appendix G), indicating robust information aggregation. This design enables effective integration of complementary reuse patterns while naturally handling input uncertainty.

We formulate multi-password reuse guessing as a conditional sequence generation problem. Let

$$\mathcal{L}_p = (pw_1, pw_2, \ldots, pw_k), \qquad y = (y_1, y_2, \ldots, y_T), \quad (1)$$

where $\mathcal{L}_p$ denotes a k-entry input sequence (duplicates allowed), with target $y = pw_{k+1}$.

We concatenate $\mathcal{L}_p$ into a single sequence:

$$X = pw_1 \ \langle \text{SEP} \rangle \ pw_2 \ \langle \text{SEP} \rangle \cdots \ \langle \text{SEP} \rangle \ pw_k. \quad (2)$$

We build on a lightweight causal language model (0.5B-parameter Qwen), following PassLLM-II [72]. The generation objective is:

$$p_\theta(y \mid \mathcal{L}_p) = \prod_{t=1}^{T} p_\theta(y_t \mid X, y_{<t}). \quad (3)$$

Training uses teacher forcing with ground-truth prefixes, and inference proceeds autoregressively until termination.

C. Polynomial loss design

From position-level evidence to loss design. The measurement findings and the PassLLM-II [72] analysis establish the same imbalance at two levels: copying positions are abundant and easy to predict, whereas mutation positions are sparse, error-prone, and decisive for exact-match success. Standard

cross-entropy assigns the same functional form to every token. Given the predominance of copying positions, the cumulative learning signal may be dominated by these positions, leaving sparse mutation positions relatively underemphasized. Thus, we seek a confidence-aware objective that introduces additional learning signals at intermediate-confidence positions, which may include difficult but informative mutation positions.

A difficulty-aware reweighting strategy: focal loss [31]. A natural way to address this imbalance is to reweight tokens by prediction difficulty. In this spirit, focal loss introduces a modulating factor to emphasize hard-to-predict tokens:

$$\mathcal{L}_{\text{Focal}}^{(t)} = -(1-P_t)^{\gamma}\log P_t. \tag{4}$$

This formulation suppresses the contribution of well-classified tokens and shifts relative emphasis toward harder predictions. This may reduce the influence of easy copying positions. In our experiments, focal loss improves multi-password guessing success rates (Figure 5). However, the hardest target positions may also correspond to cases where the available sister-password context provides weak, conflicting, or highly uncertain evidence. Consequently, overemphasizing these positions may be less effective than strengthening the learning signal at intermediate-confidence positions while still addressing the mutation-related prediction bottleneck identified above.

Polynomial view of cross-entropy. To better understand how to reshape the learning signal, we revisit cross-entropy from a polynomial perspective, following the Taylor expansion framework of Leng et al. [28]. Their polynomial loss addresses class-frequency imbalance in image classification, whereas we focus on structural imbalance in multi-password reuse: copying positions are abundant and easy to predict, whereas sparse mutation positions are decisive for exact-match success. More specifically, when a model is well-trained, $P_t{\approx}1$ at copying positions (which dominate the sequence), making $x{=}1$ a natural expansion point. Expanding $f(x){=}-\log x$ around $x{=}1$ and substituting $x{=}P_t$ yields

$$-\log P_t = \sum_{j=1}^{\infty} \frac{1}{j}(1-P_t)^j. \tag{5}$$

This representation decomposes the loss into residual-based components with different sensitivities to prediction confidence. Cross-entropy can be viewed as a weighted polynomial over the residual $(1-P_t)$ with fixed coefficients $1/j$, which motivates a more flexible loss design with tunable coefficients.

Generalized polynomial loss. Based on the above observation, we adopt a generalized polynomial formulation:

$$\mathcal{L}_{\text{Poly}}^{(t)} = \sum_{j=1}^{\infty} \alpha_j (1-P_t)^j. \tag{6}$$

When $\alpha_j = 1/j$, it reduces exactly to standard cross-entropy. To preserve its stability while enabling task adaptivity, we parameterize $\alpha_j = 1/j + \epsilon_j$, which gives

$$\mathcal{L}_{\text{Poly}}^{(t)} = \mathcal{L}_{\text{CE}}^{(t)} + \sum_{j=1}^{\infty} \epsilon_j (1-P_t)^j. \tag{7}$$

Poly-1 approximation and final loss. The polynomial view also reveals how different correction orders redistribute the optimization signal. For an additional term $\epsilon_j(1-P_t)^j$, the magnitude of its correction to the true-class logit gradient is proportional to $jP_t(1-P_t)^j$, which is maximized at $P_t = 1/(j{+}1)$. The first-order correction therefore contributes its largest additional gradient around intermediate confidence ($P_t = 0.5$), whereas higher-order corrections progressively shift their emphasis toward lower-confidence predictions. This behavior is well aligned with our setting: our position-level analysis identifies mutation positions, which are substantially harder to predict than copying positions, as the main prediction bottleneck, while extremely uncertain predictions may also reflect weak or conflicting evidence from the available inputs. Accordingly, we use the first-order correction as a simple confidence-aware approximation and empirically compare it with higher-order alternatives in Figure 5:

$$\epsilon_1 = \epsilon, \qquad \epsilon_j = 0 \quad (j > 1), \tag{8}$$

which leads to the final per-token loss

$$\mathcal{L}_{\text{ours}}^{(t)} = -\log P_t + \epsilon(1-P_t), \tag{9}$$

where $\epsilon > 0$ is a tunable hyperparameter. The sequence-level loss and overall training objective are

$$\mathcal{L}_{\text{ours}} = \sum_{t=1}^{T} \left[-\log P_t + \epsilon(1-P_t)\right], \tag{10}$$

and

$$\min_{\theta} \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} \left[-\log P_t^{(i)} + \epsilon\left(1-P_t^{(i)}\right)\right], \tag{11}$$

respectively, where $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ denotes the training set. In practice, we evaluate $\epsilon \in \{0.5, 1, 2\}$ and find that $\epsilon = 1$ provides the best overall effectiveness (see Figure 5).

Gradient analysis. We derive the gradient with respect to logit $z_{t,k}$. Let $q_{t,k}$ denote the predicted probability assigned to token k at position t. Using $\partial P_t/\partial z_{t,k} = P_t(1[k=y_t] - q_{t,k})$, we obtain

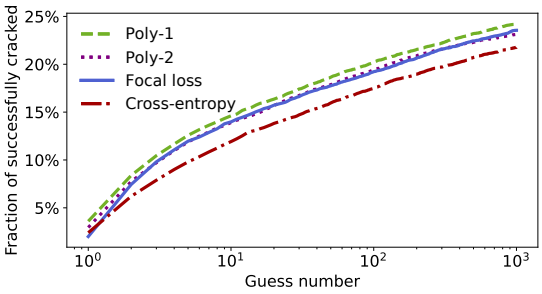
$$\frac{\partial \mathcal{L}_{\text{ours}}^{(t)}}{\partial z_{t,k}} = \left(q_{t,k} - 1[k=y_t]\right)(1+\epsilon P_t). \tag{12}$$

The loss rescales the standard cross-entropy gradient by the factor $(1+\epsilon P_t)$, preserving its direction while introducing an additional confidence-dependent correction. Equivalently, the additional Poly-1 term contributes a true-class gradient correction whose magnitude is proportional to $P_t(1-P_t)$; this correction is largest at intermediate confidence and vanishes toward both $P_t \to 0$ and $P_t \to 1$.

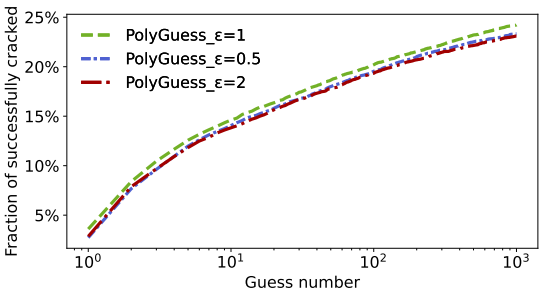
For comparison, focal loss [31], $-(1-P_t)^{\gamma}\log P_t$, rescales the cross-entropy gradient by

$$(1-P_t)^{\gamma} - \gamma P_t (1-P_t)^{\gamma-1}\log P_t.$$

This factor approaches 1 as $P_t \to 0$ and approaches 0 as $P_t \to 1$, thereby suppressing easy predictions and shifting



(a) Comparison of loss functions.



(b) Effect of the hyperparameter ϵ.

Fig. 5: Comparison of loss functions and hyperparameter sensitivity under the same architecture and training setup. Poly-1 consistently achieves the highest guessing success rates. The effectiveness remains stable across different ϵ values, with $\epsilon = 1$ achieving the best overall results.

relative emphasis toward harder ones. Poly-1 instead adds its strongest correction in the intermediate-confidence regime, providing a confidence-dependent correction that may better address the mutation-related prediction bottleneck identified in our position-level analysis, consistent with the empirical results in Figure 5.

Effect of loss design. To evaluate the impact of loss design, we compare standard cross-entropy (PassLLM-II [72]), focal loss [31], and two variants of our polynomial loss (Poly-1 and Poly-2) under the same model architecture. As shown in Figure 5, Poly-1 consistently achieves the highest success rates at all tested numbers of guesses, followed by Poly-2 and focal loss. Both focal loss and polynomial loss outperform cross-entropy, indicating that reweighting the learning signal is beneficial in multi-password settings. Poly-1 further outperforms focal loss, suggesting that shifting relative emphasis broadly toward harder predictions is less effective in our setting than concentrating the strongest additional correction at intermediate confidence. The lower performance of Poly-2 further suggests that shifting the correction toward lower-confidence predictions provides limited additional benefit.

V. Experimental Setup

In this section, we introduce the baseline models and describe how to effectively adapt existing single-password reuse models to the multi-password setting. Then, we present our experimental results.

A. State-of-the-art models for comparison

We comprehensively compare PolyGuess with state-of-the-art password models. For a fair comparison, we ensure

TABLE IV: Experimental setups for multi-password guessing scenarios†.

#	Dataset	k-password guessing	Train samples	Avg. # PWs per user	Test samples	Avg. # PWs per user
1	COMB-mix	$k\in[1,4]$	200,000	2.6	20,000	2.5
2		$k\in[5,8]$	200,000	7.2	20,000	7.0
3	Cit0day-mix	$k\in[1,4]$	200,000	2.5	20,000	2.3
4		$k\in[5,8]$	200,000	7.0	20,000	7.0
5	Naz-mix	$k\in[1,4]$	200,000	2.8	20,000	2.8
6		$k\in[5,8]$	200,000	7.2	20,000	7.2

† k counts known input entries. Raw training samples retain duplicates, including input–target matches, before model-specific preprocessing; test inputs may contain duplicates but must differ from the target. "Unseen" denotes a held-out test target, rather than a password created later.

that all models work on the same training and test sets, and manage to obtain/use their codes shared/open-sourced by the original authors. For all model parameters, we follow the best recommendations. Below, we briefly describe these models.

KNNGuess. It is a single-password guessing model proposed by Li and Wang [30] at NDSS'26. It augments a Transformer-based architecture with a k-nearest neighbors retrieval mechanism to better capture password reuse behaviors. We use the default configuration: a 6-layer encoder-decoder with embedding size 512, the Adam optimizer with learning rate 5×10^{-4}, and retrieval parameters $k = 32$ and $T = 300$.

PassLLM-II. It is a reuse-based guessing model proposed by Zou et al. [72]. It leverages a LoRA fine-tuned decoder-only LLM to generate passwords conditioned on either a single sister password or multiple sister passwords, and employs dynamic beam search for efficient inference. We adopt the authors' default setting with a Qwen2.5-0.5B base model.

TarGuess-II. This model is a PCFG [63]-based password guessing model proposed by Wang et al. at ACM CCS'16 [60]. TarGuess-II characterizes users' password reuse behaviors at two levels (segment- and character-level). To capture users' vulnerability to choosing popular passwords, TarGuess-II incorporates a popular password list when generating a guess list for each user. The parameter settings follow the default configurations provided in the authors' original code.

Pass2Path. It is a seq2seq-based password model proposed by Pal et al. [39] in 2019. Pass2Path creatively models the password reuse process as a "password-to-path" task, where the model inputs the existing password and predicts the corresponding edit operation sequence. Consistent with [39], the learning rate is 0.0003, the layer number of RNN is 3, the hidden unit is 128, and the dropout probability is 0.4.

Pass2Edit. Proposed by Wang et al. [61], Pass2Edit employs a "multi-step decision" mechanism for generating edit operations. Pass2Edit takes both the original existing password and the intermediate password produced thus far as inputs to predict the next edit operation. The predicted operation is then applied to the intermediate password to produce the input for the next timestep. Consistent with [61], we use a 3-layer GRU and a 2-layer fully connected network. The learning rate is 0.001 and the dropout rate is 0.4.

PassBERT. It is a unified and pioneering password guessing framework proposed by Xu et al. [68] at USENIX Security'23,

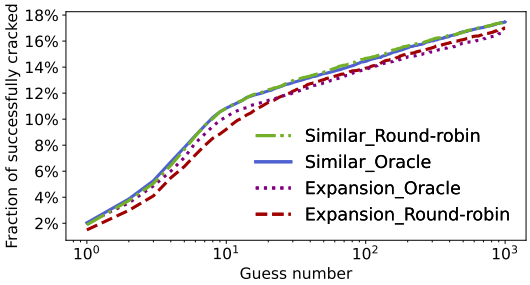


Fig. 6: Training and generation strategies for multi-password adaptation based on a single-password reuse model. Using KNNGuess [30] as a representative single-password guessing model, we compare different adaptation strategies in multi-password settings along two dimensions: training strategy (i.e., similarity-based selection vs. pairwise expansion) and generation strategy (i.e., round-robin merging vs. oracle selection).

which supports three password guessing scenarios, including password reuse attacks. It utilizes the BERT model to predict the edit operation sequence based on each user's existing password. Consistent with [68], we use 2 attention heads, a learning rate of 10^{-5}, the Adam optimizer, a dropout rate of 0.1, and a sequence length of 32/34.

PointerGuess / MS-PointerGuess. PointerGuess, proposed by Xiu and Wang [65] at USENIX Security'24, is a targeted password guessing model that reformulates conditional password generation as a combination of copying characters from existing passwords and generating new characters. Building upon PointerGuess, the authors further proposed MS-PointerGuess, which extends the framework to the multi-password setting by incorporating a multi-encoder architecture to jointly model multiple existing passwords. However, due to the substantial computational overhead introduced by additional encoders, the original paper only implements a two-encoder design for the two-password setting. To ensure a fair comparison, we strictly follow the original design and adopt the same two-encoder architecture in our experiments. Following [65], we also use the recommended parameter settings, including $\epsilon = 1 \times 10^{-12}$ for smoothing, the attention mechanism, a learning rate of 0.001, and a dropout rate of 0.5.

Pass2Pass-T. It is a multi-password targeted guessing model proposed by Huang et al. [23] in 2025. Pass2Pass-T employs a Transformer-based architecture to directly generate a target password from multiple leaked passwords of the same user. More specifically, the model concatenates multiple input passwords into a single sequence, and further introduces repetition-aware compression and segment-wise positional encoding to better capture multi-password characteristics. Following [23], we use 8 Transformer layers with embedding size 512 and 8 attention heads, trained using Adam with a learning rate of 1×10^{-4} and a dropout rate of 0.1.

B. Adapting single-password models

To answer **RQ2**, we now establish how existing single-password reuse models (e.g., Pass2Path [39] and Pass2Edit [61]) can be adapted to multi-password guessing scenarios. Our goal is to construct strong and fair baselines that fully exploit the capabilities of single-password models under realistic multi-password scenarios.

Existing password reuse models can be broadly categorized into two types: (1) *sequence generation models*, which take a source password as input and directly generate the target password (e.g., PointerGuess [65], PassLLM-II [72], and KNNGuess [30]); and (2) *edit-based models*, which predict a sequence of transformation operations from the source password to the target (e.g., Pass2Path [39] and Pass2Edit [61]). Sequence generation models can be extended to multi-password settings via input concatenation (as adopted by Pass2Pass-T [23] and PassLLM-II [72]), whereas edit-based models are not naturally suited for multi-source inputs.

Training strategies. Given multiple available passwords per user, we consider two representative strategies for constructing training data. *(i) Pairwise expansion* pairs each available password with the target, generating multiple training instances per user. *(ii) Similarity-based selection* retains only the most similar password to the target, focusing on the most informative transformation signal.

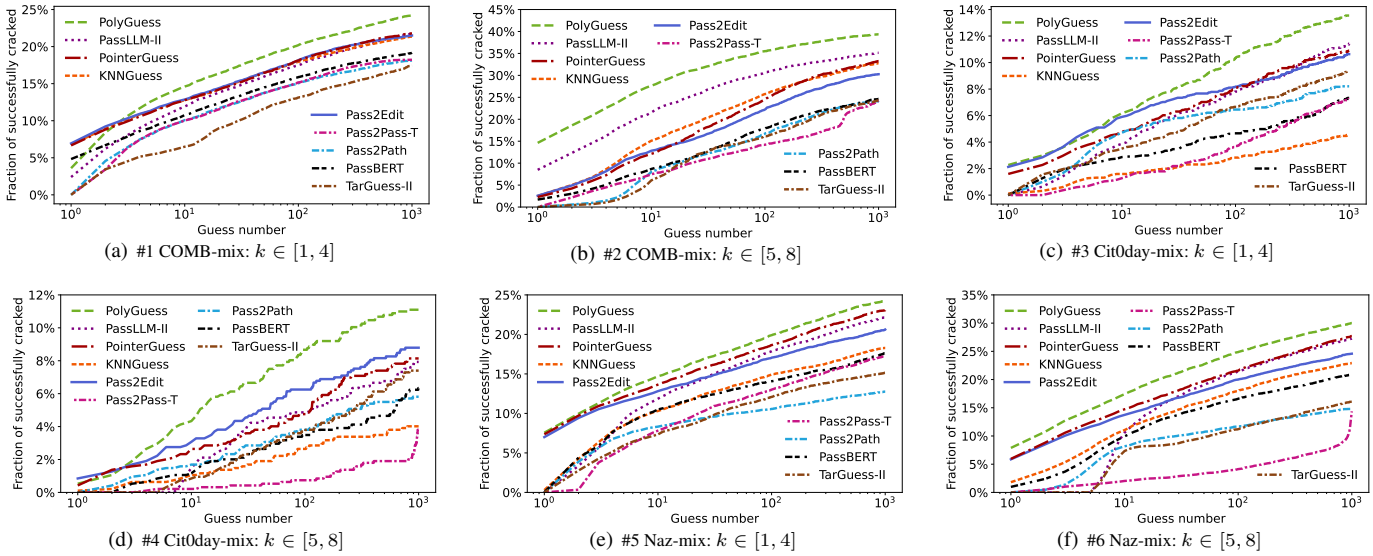
Generation strategies. At inference time, since the target password is unknown, all available passwords can be used as independent inputs. We consider two strategies. *(i) Round-robin merging* generates candidate lists from each input password and interleaves them, ensuring balanced utilization of multiple sources. *(ii) Oracle selection* assumes access to the most similar password to the target and uses it as the sole input, serving as an ideal upper bound where the model always receives the most informative source.

As shown in Figure 6 (here we take KNNGuess [30] as an illustrative example), two observations emerge. First, using all available passwords with round-robin merging performs comparably to oracle selection, suggesting that it is an effective and practical strategy for exploiting multiple inputs without explicit selection. Second, similarity-based selection consistently outperforms pairwise expansion during training. This is because, while pairwise expansion increases the amount of training data, it may introduce weakly related pairs that add noise and degrade learning quality [61]. In contrast, similarity-based selection emphasizes high-quality supervision by focusing on the strongest source-target relationship.

Based on these observations, we adopt similarity-based selection for training and round-robin merging for generation as our default configuration for adapting single-password models. As we will demonstrate in subsequent large-scale experiments (see Figure 7), this configuration provides a strong baseline, enabling single-password models (e.g., PointerGuess [65] and Pass2Edit [61]) to achieve success rates significantly higher than existing multi-password models (e.g., Pass2Pass-T [23] and MS-PointerGuess [65]).

C. Attack scenarios design

We construct several datasets by matching user accounts across large-scale password leaks using email identifiers. More specifically, we use three widely used aggregated datasets (i.e., COMB, Cit0day, and Naz) and derive three mixed datasets



(a) #1 COMB-mix: $k \in [1, 4]$ (b) #2 COMB-mix: $k \in [5, 8]$ (c) #3 Cit0day-mix: $k \in [1, 4]$

(d) #4 Cit0day-mix: $k \in [5, 8]$ (e) #5 Naz-mix: $k \in [1, 4]$ (f) #6 Naz-mix: $k \in [5, 8]$

Fig. 7: Guessing effectiveness across six multi-password scenarios. For each scenario, the training configuration follows Table IV, and the test setting is specified in the sub-caption. All six scenarios use $(pw_1, \ldots, pw_k) \rightarrow pw_{k+1}$, where $(pw_1, \ldots, pw_k)$ is the test input and pw_{k+1} is the target; the indices do not represent password-creation order. Our PolyGuess achieves the best overall guessing success rates across the evaluated settings.

(i.e., COMB-mix, Cit0day-mix, and Naz-mix), each containing multiple passwords associated with the same matched user across aggregated credential or breach records.

For the main experiments in Figure 7, training samples retain duplicates, including input–target matches, before model-specific preprocessing. Test inputs may contain duplicates, but none matches the target. We consider two input-count regimes: (i) small portfolios (k = 1-4), and (ii) larger portfolios (k = 5-8). This setting reflects practical attack scenarios with varying levels of prior knowledge. We restrict $k \leq 8$ for three reasons: (1) users with more than eight leaked passwords are rare in the dataset (see Figure 15); (2) adapting single-password models to multiple inputs increases inference cost because candidates are generated for each input and then merged; and (3) prior work such as Pass2Pass-T [23] similarly limits input size to control model input length. Following standard practice in password reuse modeling [61, 65, 72], we sample 200K training instances and at least 20K test instances per dataset (at the user level) to ensure stable evaluation. The full experimental configurations are summarized in Table IV.

Breach-ordered evaluation. We further evaluate PolyGuess in a breach-ordered setting, using passwords leaked in earlier breaches to guess a target password leaked in a later breach (see Appendix E). The guessing success rates of PolyGuess are only slightly lower than those under random hold-out evaluation, suggesting a limited impact of breach chronology in this evaluation. We leave a systematic study of temporal password evolution to future work.

D. Experimental results

Here we briefly analyze the experimental results of Scenarios #1∼#6 and provide an overall analysis of the results. As with [39, 60, 61, 65, 68], we employ the guess-number graph to evaluate the performance (i.e., guessing success rate) of different password models.

Overall analysis. Figure 7 and Table X present the guessing success rates of different models under multi-password guessing scenarios. Overall, PolyGuess consistently outperforms all baselines within 10^1–10^3 guesses. More specifically, when allowed 100 guesses, the average success rates of our PolyGuess across all scenarios are 16.81% for $k \in [1, 4]$ and 23.02% for $k \in [5, 8]$ (where k denotes the number of sister passwords available per user), compared to 14.38% and 18.99% for PassLLM-II [72], 14.94% and 16.97% for PointerGuess [65], and 14.47% and 16.17% for Pass2Edit [61]. When the guess number increases to 1,000, the average success rates of PolyGuess further rise to 20.66% and 26.83%, while the corresponding figures are 18.46% and 23.45% for PassLLM-II [72], 18.58% and 23.03% for PointerGuess [65], and 17.58% and 21.21% for Pass2Edit [61]. Particularly, PolyGuess outperforms its foremost counterpart (i.e., PassLLM-II [72]) by 11.42%-78.40% (28.2% on average) and 9.20%-39.97% (16.8% on average) within 100 and 1,000 guesses, respectively.

In particular, the lower success rates on Cit0day are consistent with its markedly different post-deduplication password-portfolio distribution: only a small fraction of its users have multiple distinct passwords (see Table II).

Since PolyGuess and PassLLM-II [72] differ primarily in the loss formulation, the consistent gains across these controlled comparisons suggest that the Poly-1 loss helps the model make better use of the available training evidence, particularly at critical mutation positions. To better understand this advantage, we examine the passwords additionally cracked by PolyGuess in Section VI-B.

RQ2: Effectiveness of single-password models in multi-password settings. Interestingly, in some cases, single-

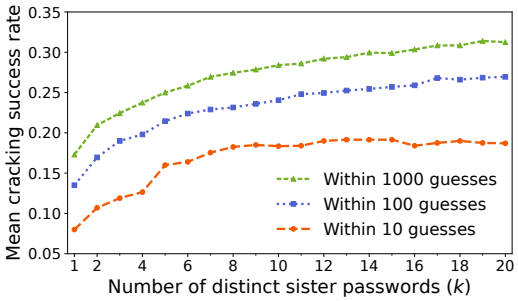


Fig. 8: Mean success rates on COMB and Naz versus the number of distinct sister passwords. Each curve averages the two datasets with equal weights.

password models (e.g., PointerGuess [65] and Pass2Edit [61]) using similarity-based selection during training and round-robin merging during generation achieve success rates comparable to the leading multi-password model PassLLM-II [72], and even substantially outperform Pass2Pass-T [23] and MS-PointerGuess [65] (see Figure 13). This observation indicates that simply incorporating multiple passwords as input does not necessarily translate into improved effectiveness. Instead, it suggests that existing multi-password models (e.g., PassLLM-II [72]) may not fully exploit the structural and semantic dependencies across multiple passwords. We will further investigate this phenomenon and analyze the source of effectiveness in multi-password guessing models in Section VI-C.

Effect of the number of sister passwords. We evaluate PolyGuess on 1,000 fixed targets in each of COMB and Naz (Cit0day does not contain enough users with 20 distinct sister passwords). For each target, we increase the number of distinct sister passwords used as input from $k = 1$ to $k = 20$, adding one password at each step. Four checkpoints trained on the same users cover k = 1–4, 5–10, 11–15, and 16–20. Training and test users do not overlap. Figure 8 shows that the mean success rate within 1,000 guesses increases from 25.00% at $k = 5$ to 28.40% at $k = 10$, but rises by only 0.90 percentage points from $k = 16$ to $k = 20$. This smaller increase can be partly explained by how rarely later inputs reveal new parts of the target. We measure target coverage using exact four-character matches. For example, if a known sister password contains `blue` from the target `blue4729`, another containing `4729` covers a previously uncovered part. From $k = 16$ to $k = 20$, adding one sister password increases target coverage in 0.4%–1.3% of cases in either dataset.

VI. Further Analysis

We analyze PolyGuess in terms of model efficiency, the characteristics of additionally cracked passwords, and the sources of gains in existing multi-password guessing models.

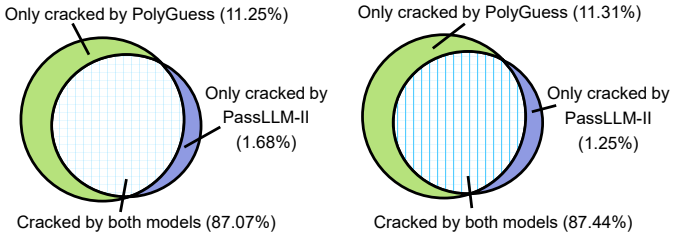
A. Model efficiency

We report the training time, inference speed, and model size of all models in Table V. The performance gap in password generation speed mainly stems from differences in model architectures, model sizes during inference, and generation strategies. In particular, PolyGuess and PassLLM-II [72] employ Qwen-based LLM architectures. Although LoRA fine-tuning introduces only a small number of trainable parameters, inference still requires the entire Qwen backbone, resulting in higher computational overhead than lightweight models such as KNNGuess [30] and the RNN-based decoder used by Pass2Edit [61]. However, in targeted online guessing, generation speed is generally not the primary bottleneck because the attack is constrained by account-level interactions, including login attempts and rate limits. To further improve the generation efficiency of PolyGuess, a promising direction is to investigate architectural optimizations tailored to password guessing, such as simplifying computationally intensive modules while preserving guessing success rates; we leave this direction for future work.

TABLE V: Running time of different password models.[†]

Attack model	Training	Testing	Speed[‡] (pw/s)	Size*
PolyGuess	01:35	09:27	250∼260	21.0 MB
PassLLM-II [72]	00:33	08:54	270∼280	21.0 MB
Pass2Pass-T [23]	00:21	09:52	240∼250	38.0 MB
MS-PointerGuess [65]	00:57	00:05	9,000∼9,500	4.3 MB
KNNGuess [30]	01:02	03:57	1,000∼1,100	189 MB
PointerGuess [65]	00:25	00:28	9,500∼10,000	2.3 MB
PassBERT [68]	02:08	00:51	5,000∼5,200	85.2 MB
Pass2Edit [61]	00:14	02:26	1,600∼2,000	10.9 MB
Pass2Path [39]	00:52	01:24	2,500∼2,600	56.7 MB
TarGuess-II [60]	00:01	00:24	11,000∼11,500	84.4 MB

[†]All running data are measured from Scenario #5, and the format of the training/testing time is "hour:minute". The parameters of all password models are consistent with the previous experiments in Sec. V-A.
[‡]Speed denotes the number of guesses generated per second.
*For PolyGuess and PassLLM-II [72], size denotes the LoRA adapter weights only, excluding the pretrained base model.



(a) Scenarios when $k \in [1, 4]$ (b) Scenarios when $k \in [5, 8]$

Fig. 9: The overlap between the passwords cracked by our PolyGuess and those cracked by PassLLM-II [72] (within 1,000 guesses).

B. Characteristics of additionally cracked passwords

We analyze the characteristics of passwords that are successfully cracked by PolyGuess but missed by PassLLM-II [72] under the same multi-password input setting.

Overlap of cracked passwords. Figure 9 shows that the two models share a large fraction of cracked passwords, accounting for 87.07% when $k \in [1, 4]$ and 87.44% when $k \in [5, 8]$ (where k denotes the number of sister passwords available per user). In particular, PolyGuess consistently cracks more additional passwords than PassLLM-II [72] (i.e., 11.25% vs. 1.68% for $k \in [1, 4]$, and 11.31% vs. 1.25% for $k \in [5, 8]$). We provide representative examples of passwords additionally cracked by PolyGuess in Table IX and further analyze the characteristics of these additional passwords below.

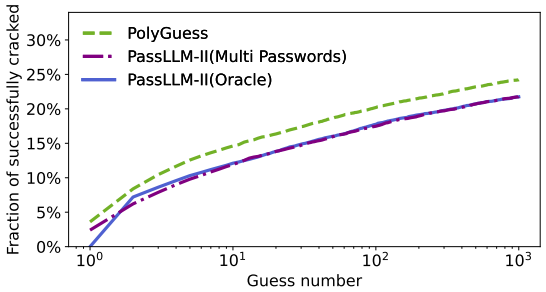


Fig. 10: Comparison of generation strategies at test time. We evaluate PassLLM-II [72] under two settings: (i) oracle selection, which uses the most similar single sister password as input, and (ii) the original PassLLM-II setting, which uses all available sister passwords as input.

Similarity to input passwords. We measure the maximum 2-gram cosine similarity between each cracked password and the set of input passwords. Passwords additionally cracked by PolyGuess show significantly lower (maximum) similarity scores (average: 0.7057) compared to those cracked by both models (average: 0.8435), indicating that PolyGuess can crack targets even without highly similar inputs.

Beyond single-source similarity. We further examine whether a cracked password is a near-duplicate of any individual input password. Among passwords additionally cracked by PolyGuess, 26.13% have similarity below 0.6 to every individual input, compared with 6.31% among passwords cracked by both models. Thus, these additional targets are less often explained by a highly similar single source. This difference is consistent with PolyGuess making use of complementary evidence across inputs.

Low-confidence positions. We analyze token-level prediction confidence via the difficulty $d_t = -\log p_\theta(y_t \mid X, y_{<t})$, where larger values indicate lower confidence. Passwords additionally cracked by PolyGuess show substantially higher difficulty (0.8154 vs. 0.4905). Moreover, 26.52% of tokens in these passwords fall into the low-confidence regime ($d_t > 1.0$), compared to 7.10% for passwords cracked by both models, corresponding to $p(y_t) < e^{-1}$.

C. Source of gains in multi-password guessing

To answer **RQ3**, we use PassLLM-II [72] as a representative multi-password model and conduct a controlled comparison among three strategies: (i) an oracle single-password setting that selects the input most similar to the held-out target, (ii) the original PassLLM-II with concatenated multi-password input, and (iii) our PolyGuess.

For representative multi-password models, oracle-selected single-password input approaches multi-password input. As shown in Figure 10, the oracle single-password setting of PassLLM-II [72] achieves a success rate close to that of its concatenated multi-password setting. We observe the same qualitative pattern for MS-PointerGuess [65], using PointerGuess with oracle source selection as the reference. These results suggest that an important source of the gains in existing multi-password models is an implicit oracle-selection effect. While such oracle selection is not deployable, this observation is consistent with our set-level finding that larger password portfolios are more likely to contain a highly similar source (see the Finding-I in Section III-C).

PolyGuess improves over both PassLLM-II [72] settings. PolyGuess consistently outperforms PassLLM-II with either the oracle-selected single source or concatenated multi-password input. Since PolyGuess employs the same underlying LLM architecture and concatenated-input paradigm while primarily changing the training objective, this result supports the narrower conclusion that Poly-1 improves the model's use of the available multi-password context, with gains consistent with better prediction of critical mutation positions.

These results distinguish two effects. For PassLLM-II [72], much of the benefit of multiple passwords can be explained by the increased chance of exposing a highly informative source, whereas PolyGuess provides additional gains consistent with improved optimization of difficult mutation positions under the same multi-password context. This distinction suggests that future evaluations should separate gains due to source availability from those due to model or training design.

VII. Security Implications

For password policies. To restrict exact password reuse, many systems check password history. For example, Google [21] prohibits passwords previously used on the same account. The password policies of some high-profile services even explicitly specify the number of previous passwords checked: AWS Builder ID prohibits reuse of the last three passwords [8], while PayPal Braintree lists "Can't be one of the last 4 previously used passwords" as a requirement for new Control Panel passwords [44]. Salesforce likewise states that users "can't reuse their last three passwords" under its default policy for new organizations [47], while its Government Cloud Plus secure configuration guide recommends prohibiting reuse of the 24 most recent passwords [48].

Checking a longer password history rules out more previously used passwords but has two drawbacks: (1) the server needs to store more password records, in plaintext, hashed, or salted-hash form, which would expose more credential information about the user if the server were compromised; (2) our results reveal risks beyond *exact* reuse: multiple *non-identical* sister passwords can collectively cover substantial portions of a target password (see Section III-C), and guessing success rates generally increase with the increase of available sister passwords (see Figure 8). Thus, it is of limited benefit to prohibit exact reuse, and instead, password policies at security-critical systems can check whether a newly chosen password is predictable from multiple other passwords belonging to the same user (e.g., through proactive use of password leakage detection services such as MIGP [40]).

For password expiration, China's Ministry of Science and Technology (MOST) *mandates* password changes every six months [37]. Similarly, Salesforce uses a default password expiration period of "90 days" [47]. Frequent password changes may encourage users to repeatedly modify familiar passwords [7]. At CCS 2010, Zhang et al. [70] demonstrated

this risk by guessing replacement passwords from expired ones. Our study further shows that attackers can achieve higher guessing success rates by jointly exploiting relationships among multiple leaked passwords than by relying on a single leaked password. These findings complement prior advice on password changes [7, 70] and provide concrete empirical evidence that supports standard bodies like NIST and NCSC which discard mandatory periodic password changes [52, 54].

Password policies addressing reuse risks should also preserve usability. Users may modify passwords they already remember to reduce the effort of managing multiple accounts [7, 17], because familiar knowledge can help people remember new information [15]. Accordingly, the burden of creating and memorizing new passwords is an important consideration when designing effective defenses against multi-password guessing. Prior research has offered reasonable guidance for balancing security and usability in password policy design: Komanduri et al. [27] and Shay et al. [50] showed that policies emphasizing password length can offer a better balance between strength and usability than traditional password complexity requirements; Tan et al. [51] further supported combining password length requirements with a minimum strength threshold based on guessability estimates.

For password replacement, Gerchanovsky et al. [20] demonstrated the potential of LLM-assisted suggestions, which can improve resistance to online guessing even when the attacker knows both the original password and the tool's implementation, with comparable password recall to the traditional approach after one week. Our results suggest that such mechanisms could also be evaluated against attackers who jointly exploit *multiple* leaked sister passwords to assess whether their security benefits still persist.

Reducing the burden of password management. Given users' limited effort budgets (e.g., energy and time) in managing multiple accounts [17], password managers [13, 24] can help reduce this burden. Simply storing and autofilling existing user-created passwords reduces memorization and entry effort, but does not remove any predictable relationships among passwords. Replacing such passwords with independently generated random ones can reduce the relationships exploited by multi-password guessing. NIST requires password verifiers to allow the use of password managers and autofill [52]. For users hesitant to adopt a password manager, a practical starting point is to use it for less important accounts and devote some of the effort saved to protecting more important accounts, as suggested in [17, 49].

Applications to strength estimation and credential checking. Prior work has evaluated password strength in terms of guessability using trawling guessing models (e.g., [25, 35, 42, 56]), and single-password reuse models have been adapted for this purpose (e.g., [30, 39, 65]). PolyGuess extends this approach to settings where multiple passwords from the same user are available. A password manager could employ PolyGuess to assess whether a new password can be guessed from the user's other stored passwords within a given number of guesses and warn the user accordingly. PolyGuess could also complement compromised-credential checking services such as MIGP [40], which checks exact reuse and variants generated from individual leaked passwords. Given multiple passwords reliably associated with the same user, PolyGuess could generate candidate variants to identify predictable relationships that pairwise checks in MIGP [40] may miss. Such assessments could be performed locally or through appropriately designed privacy-preserving protocols that keep plaintext passwords on the client.

Metric-informed defensive directions. Our measurements suggest two complementary usages of the proposed metrics defined in Section III-B. First, password managers could use MaxSim, MaxLCSCov, MS-PosCopyRate, and MutRatio to assess the relationship between a candidate password and the user's existing passwords, identifying potentially predictable modifications. Second, MeanSim, PivotSim, VarSim, TemplateCons, and LenCV could characterize the structure of a user's password portfolio and inform risk assessment. When multiple leaked passwords are reliably linked to the same user, these metrics could provide important signals for prioritizing post-leak notifications and other protective measures.

VIII. Conclusion

This paper systematically investigates the multi-password reuse threat, where smart attackers exploit the victim's multiple existing passwords to guess her target password on another account. We propose PolyGuess, a multi-password reuse model that effectively characterizes the structural and semantic relationships among a user's passwords. Extensive experiments demonstrate that PolyGuess significantly outperforms leading password models. We hope that our work can inform both future targeted-guessing models and reuse-aware defenses.

Acknowledgement

The authors are deeply grateful to the shepherd and anonymous reviewers for their invaluable comments. Ding Wang is the corresponding author. This research was in part supported by the National Natural Science Foundation of China under Grants Nos. 62502238 and 62572259, and by the China Postdoctoral Science Foundation under Grant No. 2025M771596, and by the Tianjin Natural Science Foundation Project under Grant No. 25JCJQJC00230.

Ethical considerations

While our password datasets have been publicly available on the Internet or the dark web and widely used in prior research [11, 12, 29, 33, 36, 39, 41, 43, 67], they remain sensitive private data. To ensure no additional harm is caused to users, we take two precautions: (1) We process these datasets in an isolated environment without internet access and delete all sensitive data (e.g., email address and corresponding passwords) upon completion of our experiments; (2) We report only aggregated statistical results. We treat each account with the utmost confidentiality to prevent any new risks to the individuals involved. We hope this work advances understanding of multi-password reuse and informs defenses.

References

[1] *COMB data breach: what it means, and how to protect yourself*, Feb. 2021, https://1password.com/blog/what-comb-means-for-you-and-your-business.

[2] *Data Breach Saga: What You Need To Know About The Cit0day Data Leak*, Mar 2021, https://www.bitdefender.com/en-us/blog/hotforsecurity/data-breach-saga-what-you-need-to-know-about-the-cit0day-data-leak.

[3] *North Americans are the most affected by password leaks*, Sep 2023, https://surfshark.com/research/chart/leaked-passwords.

[4] *2024 Data Breach Investigations Report*, May 2024, https://www.verizon.com/business/resources/reports/2024-dbir-data-breach-investigations-report.pdf.

[5] *2025 Data Breach Investigations Report*, Apr 2025, https://www.verizon.com/business/resources/reports/2025-dbir-data-breach-investigations-report.pdf.

[6] *Naz.API Leaks Data from Over 70 Million Accounts*, Apr 2025, https://www.cyberqp.ai/post/naz-api-leaks-data-from-over-70-million-accounts.

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

[8] Amazon Web Services, *Change your AWS Builder ID password*, https://docs.aws.amazon.com/signin/latest/userguide/change-password-builder-id.html.

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

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

[11] W. Corrias, F. De Gaspari, D. Hitaj, and L. V. Mancini, "MAYA: Addressing inconsistencies in generative password guessing through a unified benchmark," in *Proc. IEEE S&P 2026*, pp. 2957–2976.

[12] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *Proc. NDSS 2014*, pp. 1–15.

[13] Y. Duan, D. Wang, and Y. Fu, "Security analysis of master-password-protected password management protocols," in *Proc. IEEE S&P*, 2025, pp. 701–719.

[14] M. Durr, *Data breach exposes logins, passwords for 184M Apple, Facebook, Google users and more*, May 2025, https://www.mlive.com/news/2025/05/data-breach-exposes-logins-passwords-for-184m-apple-facebook-google-users-and-more.html.

[15] K. A. Ericsson, W. G. Chase, and S. Faloon, "Acquisition of a memory skill," *Science*, vol. 208, no. 4448, pp. 1181–1182, 1980.

[16] D. Florêncio and C. Herley, "A large-scale study of web password habits," in *Proc. WWW 2007*, pp. 657–666.

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

[18] J. Fowler, *149M Logins and Passwords Exposed Online Including Financial Accounts, Instagram, Facebook, Roblox, Dating Sites, and More*, January 2026, https://www.expressvpn.com/research/149m-infostealer-data-exposed.

[19] 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.

[20] A. Gerchanovsky, L. Bauer, and M. K. Reiter, "MindReader: Using LLMs to encourage memorable and secure password replacement," *arXiv preprint arXiv:2607.12148*, 2026.

[21] Google, *Create a strong password & a more secure account*, https://support.google.com/accounts/answer/32040?hl=en&format=embed.

[22] A. Hanamsagar, S. S. Woo, C. Kanich, and J. Mirkovic, "Leveraging semantic transformation to investigate password habits and their causes," in *Proc. CHI 2018*, pp. 1–12.

[23] F. Huang, J. Yang, H. Cheng, W. Li, and P. Wang, "Personalized password guessing via modeling multiple leaked credentials of the same user," in *Proc. ESORICS 2025*, pp. 224–242.

[24] A. Infantino, M. M. Ali, K. Solomos, and J. Polakis, "Vault Raider: Stealthy UI-based attacks against password managers in desktop environments," in *Proc. NDSS*, 2026, pp. 1–18.

[25] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *Proc. IEEE S&P 2012*, pp. 523–537.

[26] J. Kim, M. Song, M. Seo, Y. Jin, and S. Shin, "PassREfinder: Credential stuffing risk prediction by representing password reuse between websites on a graph," in *Proc. SP 2024*, pp. 1385–1404.

[27] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman, "Of passwords and people: Measuring the effect of password-composition policies," in *Proc. ACM CHI*, 2011, pp. 2595–2604.

[28] Z. Leng, M. Tan, C. Liu, E. D. Cubuk, J. Shi, S. Cheng, and D. Anguelov, "PolyLoss: A polynomial expansion perspective of classification loss functions," in *Proc. ICLR 2022*.

[29] L. Li, B. Pal, J. Ali, N. Sullivan, R. Chatterjee, and T. Ristenpart, "Protocols for checking compromised credentials," in *Proc. ACM CCS 2019*, pp. 1387–1403.

[30] Z. Li and D. Wang, "Targeted password guessing using k-nearest neighbors," in *Proc. NDSS 2026*, pp. 1–18.

[31] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. ICCV 2017*, pp. 2999–3007.

[32] S. G. Lyastani, M. Schilling, S. Fahl, M. Backes, and S. Bugiel, "Better managed than memorized? studying the impact of managers on password strength and reuse," in *Proc. USENIX SEC 2018*, pp. 203–220.

[33] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Proc. IEEE S&P 2014*, pp. 689–704.

[34] P. Mayer, C. W. Munyendo, M. L. Mazurek, and A. J. Aviv, "Why users (don't) use password managers at a large educational institution," in *Proc. USENIX SEC 2022*, pp. 1849–1866.

[35] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *Proc. ACM CCS 2013*, pp. 173–186.

[36] 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.

[37] Ministry of Science and Technology of the People's Republic of China, *Government Service Platform: Frequently Asked Questions (in Chinese)*, Aug. 2022, https://fuwu.most.gov.cn/html/bszx/qtl/20220825/123124359.html.

[38] A. Nisenoff, M. Golla, M. Wei, J. Hainline, H. Szymanek, A. Braun, A. Hildebrandt, B. Christensen, D. Langenberg, and B. Ur, "A two-decade retrospective analysis of a university's vulnerability to attacks exploiting reused passwords," in *Proc. USENIX SEC 2023*, pp. 5127–5144.

[39] 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.

[40] B. Pal, M. Islam, M. S. Bohuk, N. Sullivan, L. Valenta, T. Whalen, C. Wood, T. Ristenpart, and R. Chatterjee, "Might i get pwned: A second generation compromised credential checking service," in *Proc. USENIX SEC 2022*, pp. 1831–1848.

[41] D. Pasquini, G. Ateniese, and C. Troncoso, "Universal neural-cracking-machines: Self-configurable password models from auxiliary data," in *Proc. IEEE S&P 2024*, pp. 1365–1384.

[42] D. Pasquini, M. Cianfriglia, G. Ateniese, and M. Bernaschi, "Reducing bias in modeling real-world password strength via deep learning and dynamic dictionaries," in *Proc. USENIX SEC 2021*, pp. 821–838.

[43] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, "Improving password guessing via representation learning," in *Proc. IEEE S&P 2021*, pp. 1382–1399.

[44] PayPal Braintree, *Managing Users and Roles*, https://developer.paypal.com/braintree/articles/control-panel/users-roles/managing-users-roles.

[45] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's go in for a closer look: Observing passwords in their natural habitat," in *Proc. ACM CCS 2017*, pp. 295–310.

[46] S. Sahin and F. Li, "Don't forget the stuffing! revisiting the security impact of typo-tolerant password authentication," in *Proc. CCS 2021*, pp. 252–270.

[47] Salesforce, *Set Password Policies*, https://help.salesforce.com/s/articleView?id=sf.admin_password.htm&language=en_US&type=5.

[48] Salesforce, *Configure Security Settings and Monitor Compliance in Government Cloud*, May 2026, https://help.salesforce.com/s/articleView?id=ind.government_cloud_configure_security_settings_and_monitor_compliance.htm&language=en_US&type=5.

[49] S. Schechter, *Before you use a password manager*, Jun. 2019, https://stuartschechter.org/posts/before-you-use-a-password-manager/.

[50] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, "Designing password policies for strength and usability," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 4, pp. 13:1–13:34, 2016.

[51] J. Tan, L. Bauer, N. Christin, and L. F. Cranor, "Practical recommendations for stronger, more usable passwords combining minimum-strength, minimum-length, and blocklist requirements," in *Proc. ACM CCS*, 2020, pp. 1407–1426.

[52] D. Temoshok, J. L. Fenton, Y.-Y. Choong, N. Lefkovitz, A. Regenscheid, R. Galluzzo, and J. P. Richer, *NIST SP 800-63B-4: Digital Identity Guidelines: Authentication and Authenticator Management*, Jul. 2025, https://pages.nist.gov/800-63-4/sp800-63b.html.

[53] K. Thomas, J. Pullman, K. Yeo, A. Raghunathan, P. G. Kelley, L. Invernizzi, B. Benko, T. Pietraszek, S. Patel, D. Boneh *et al.*, "Protecting accounts from credential stuffing with password breach alerting," in *Proc. USENIX SEC 2019*, pp. 1556–1571.

[54] UK National Cyber Security Centre, *The problems with forcing regular password expiry*, Oct. 2016, https://www.ncsc.gov.uk/blog-post/problems-forcing-regular-password-expiry.

[55] B. Ur, F. Noma, J. Bees, S. M. Segreti, R. Shay, L. Bauer, N. Christin, and L. F. Cranor, "I added '!' at the end to make it secure: Observing password creation in the lab," in *Proc. SOUPS 2015*, pp. 123–140.

[56] B. Ur, S. M. Segreti, L. Bauer, N. Christin, L. F. Cranor, S. Komanduri, D. Kurilova, M. L. Mazurek, W. Melicher, and R. Shay, "Measuring real-world accuracies and biases in modeling password guessability," in *Proc. USENIX SEC 2015*, pp. 463–481.

[57] K. Viezelyte, *Juggling Security: How Many Passwords Does the Average Person Have in 2026?*, May 2026, https://nordpass.com/blog/how-many-passwords-does-average-person-have/.

[58] D. Wang, D. He, H. Cheng, and P. Wang, "fuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars," in *Proc. IEEE/IFIP DSN 2016*, pp. 595–606.

[59] 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.

[60] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM CCS 2016*, pp. 1242–1254.

[61] D. Wang, Y. Zou, Y.-A. Xiao, S. Ma, and X. Chen, "PASS2EDIT: A multi-step generative model for guessing edited passwords," in *Proc. USENIX SEC 2023*, pp. 983–1000.

[62] D. Wang, Y. Zou, Z. Zhang, and K. Xiu, "Password guessing using random forest," in *Proc. USENIX SEC 2023*, pp. 965–982.

[63] 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.

[64] D. Winder, *16 Billion Apple, Facebook, Google And Other Passwords Leaked*, Jun 2025, https://www.forbes.com/sites/daveywinder/2025/06/20/16-billion-apple-facebook-googl

e-passwords-leaked---change-yours-now/.
[65] K. Xiu and D. Wang, "PointerGuess: Targeted password guessing model using pointer mechanism," in *Proc. USENIX SEC 2024*, pp. 5555–5572.
[66] M. Xu, W. Han, J. Yu, J. Liu, X. Zhang, Y. Lin, and J. S. Dong, "On the account security risks posed by password strength meters," in *Proc. AsiaCCS 2025*, pp. 1582–1599.
[67] 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.
[68] M. Xu, J. Yu, X. Zhang, C. Wang, S. Zhang, H. Wu, and W. Han, "Improving real-world password guessing attacks via bi-directional transformers," in *Proc. USENIX SEC 2023*, pp. 1001–1018.
[69] M. Xu, S. Zhang, K. Zhang, H. Zhang, J. Zhang, J. Yu, L. Cheng, and W. Han, "Using parallel techniques to accelerate PCFG-based password cracking attacks," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 4, pp. 3483–3496, 2025.
[70] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: An algorithmic framework and empirical analysis," in *Proc. ACM CCS 2010*, pp. 176–186.
[71] V. Zimmermann, "From the quest to replace passwords towards supporting secure and usable password creation," Ph.D. dissertation, Technical University of Darmstadt, Germany, 2021.
[72] Y. Zou, M. An, and D. Wang, "Password guessing using large language models," in *Proc. USENIX SEC 2025*, pp. 7799–7818.

Appendix

A. Statistical validation of set-level trends

To further confirm that the observed set-level patterns are not due to random variation, we perform Cochran–Armitage trend tests on Naz for representative binary conditions derived from our key findings. These conditions correspond to high-similarity, high-overlap, heterogeneous, and low-transformation regimes: (i) $\text{MaxSim}_u \geq 0.8$, (ii) $\text{VarSim}_u > 0.04$, (iii) $\text{MaxLCSCov}_u \geq 0.8$, (iv) $\text{MS-PosCopyRate}_u \geq 0.8$, and (v) $\text{MutRatio}_u \leq 0.2$. For each condition, we test whether its prevalence changes monotonically across m-strata, thereby capturing systematic changes in these set-level statistics as the observed password portfolio grows. Table VI reports the corresponding Z-statistics, p-values, and trend directions. All tested conditions show statistically significant increasing trends across m-strata ($p<0.01$). These results provide quantitative support for the set-level reuse trends observed in Naz and reported in Section III-C.

TABLE VI: Cochran–Armitage trend tests on the full Naz dataset.

Condition	Z-statistic	p-value	Trend
$\text{MaxSim}_u \geq 0.8$	31.96	< 0.01	↑
$\text{VarSim}_u > 0.04$	21.94	< 0.01	↑
$\text{MaxLCSCov}_u \geq 0.8$	30.19	< 0.01	↑
$\text{MS-PosCopyRate}_u \geq 0.8$	12.73	< 0.01	↑
$\text{MutRatio}_u \leq 0.2$	7.73	< 0.01	↑

TABLE VII: Distribution of different set-level reuse metrics across m-strata.

Metric	Range	Proportion (%)				
		m=2	m=3	m=4	m=5	$m\geq$6
MeanSim	0.0–0.2	66.99	53.32	63.86	61.15	51.83
	0.2–0.4	5.26	33.29	22.63	26.60	27.43
	0.4–0.6	4.59	5.30	9.28	8.96	14.08
	0.6–0.8	10.17	5.95	3.53	2.59	5.31
	0.8–1.0	13.00	2.14	0.70	0.70	1.35
MaxSim	0.0–0.2	66.99	41.59	24.46	14.21	3.21
	0.2–0.4	5.26	7.81	8.45	7.69	3.62
	0.4–0.6	4.59	6.71	7.18	6.69	3.27
	0.6–0.8	10.17	17.44	20.74	21.14	11.45
	0.8–1.0	13.00	26.45	39.17	50.28	78.44
VarSim†	0.00–0.04	–	61.12	42.63	33.61	28.25
	0.04–0.08	–	8.25	18.35	32.61	29.00
	0.08–0.12	–	9.53	25.81	21.11	31.95
	0.12–0.16	–	16.04	9.39	11.03	9.90
	0.16–0.20	–	4.53	3.51	1.56	0.83
PivotSim†	0.0–0.2	–	47.93	35.96	32.14	25.47
	0.2–0.4	–	17.88	38.40	38.81	34.28
	0.4–0.6	–	23.86	17.64	20.86	22.75
	0.6–0.8	–	6.41	6.47	6.88	14.75
	0.8–1.0	–	3.92	1.53	1.32	2.76
TemplateCons	0.0–0.2	88.55	75.57	63.37	88.21	76.75
	0.2–0.4	0.00	0.00	31.15	4.49	19.94
	0.4–0.6	0.00	22.76	4.99	6.07	2.38
	0.6–0.8	0.00	0.00	0.00	0.92	0.53
	0.8–1.0	11.46	1.67	0.50	0.32	0.41
LenCV	0.0–0.2	78.10	61.01	53.25	45.56	29.13
	0.2–0.4	18.10	32.12	37.04	43.00	54.99
	0.4–0.6	3.30	5.99	8.41	9.64	10.67
	0.6–0.8	0.44	0.82	1.19	1.67	5.09
	0.8–1.0	0.07	0.05	0.10	0.12	0.07
MaxLCSCov	0.0–0.2	56.79	26.71	11.21	4.45	0.65
	0.2–0.4	13.79	20.06	18.66	14.46	4.61
	0.4–0.6	4.66	6.88	7.46	6.97	3.52
	0.6–0.8	2.99	4.93	5.81	5.12	2.28
	0.8–1.0	21.77	41.43	56.86	69.00	88.94
MS-PosCopyRate	0.0–0.2	69.75	56.27	46.10	40.20	22.78
	0.2–0.4	5.29	9.27	11.63	12.88	11.79
	0.4–0.6	3.32	4.68	6.31	6.87	8.06
	0.6–0.8	4.01	5.04	6.16	6.19	8.13
	0.8–1.0	17.64	24.74	29.81	33.88	49.25
MutRatio‡	0.0–0.2	14.19	19.44	23.45	27.18	36.91
	0.2–0.4	7.25	9.78	11.53	11.77	14.09
	0.4–0.6	4.72	5.88	6.59	6.81	8.48
	0.6–0.8	6.69	8.12	8.89	9.03	8.64
	0.8–1.0	42.65	36.69	32.14	29.29	20.74

† VarSim and PivotSim are not reported for m=2.
‡ For MutRatio, only the $[0, 1]$ range is displayed.

B. Metric definitions

We now provide detailed definitions of the nine metrics and describe the multi-password characteristics they capture.

Mean pairwise similarity. The MeanSim metric captures the overall level of within-user similarity. High values (e.g., $\geq$0.5) indicate greater structural and compositional consistency, whereas low values (e.g., $\leq$0.3) indicate low pairwise similarity. *Applicable for all groups ($m \geq 2$).*

$$\text{MeanSim}_u = \frac{2}{m(m-1)} \sum_{i<j} s_{ij}.$$

Maximum pairwise similarity. MaxSim detects *local* reuse. Even when a user's passwords are globally diverse, a high

MaxSim (e.g., $\geq$0.8) indicates that at least one high-similarity pair exists, suggesting localized reuse through minor modification. *Applicable for: all groups ($m \geq 2$).*

$$\text{MaxSim}_u = \max_{i<j} s_{ij}.$$

Similarity variance. VarSim measures similarity heterogeneity within a password set. Near-zero variance indicates that pairwise similarities are nearly equal. A high variance indicates mixed behavior, where some pairs show high similarity while others remain low-similarity. *Applicable for: $m \geq 3$.*

$$\text{VarSim}_u = \text{Var}\big(\{s_{ij}\}_{i<j}\big).$$

Pivot similarity. Pearman et al. [45] and Ur et al. [55] show that users tend to select a small number of "base passwords/elements" and derive other passwords via slight modifications. To capture such hub-like reuse patterns in our data, we introduce the metric of pivot similarity. First, we define the *pivot password* as the most central password:

$$pw^* = \arg\max_{pw_i} \frac{1}{m-1} \sum_{j \neq i} s_{ij}.$$

Then, PivotSim is the average similarity between pw^* and all other passwords:

$$\text{PivotSim}_u = \frac{1}{m-1} \sum_{j \neq *} s_{*j}.$$

A high PivotSim (e.g., $\geq$0.5) indicates high average similarity to the other passwords and may suggest a hub-like reuse structure. When $m = 2$, PivotSim reduces to a single pairwise score and is therefore not reported. *Applicable for: $m \geq 3$.*

Template consistency. To capture structural regularity at the format level (independent of specific character choices), we measure template consistency based on character class patterns. Each password is mapped to a structural *template* by replacing every character with its character-class label: `L` (letter), `D` (digit), or `S` (symbol). For example, `alice2024!` $\rightarrow$ `LLLLLDDDDS`. Template consistency is measured by the normalized entropy of the template distribution:

$$\text{TemplateCons}_u \;=\; 1 - \frac{H(\{t_i\})}{\log m},$$

where H is the Shannon entropy of the multiset of templates $\{t_1, \ldots, t_m\}$, normalized by $\log m$ to ensure the result lies in $[0, 1]$ regardless of m. A value of 1 means all passwords share a single template; a value of 0 means every password has a distinct template. This formulation is stable across all group sizes and permits direct cross-group comparison. *Applicable for: all groups ($m \geq 2$).*

Length variability. LenCV complements TemplateCons by capturing length-level structural regularity. It is defined as the coefficient of variation of password lengths:

$$\text{LenCV}_u \;=\; \frac{\sigma(\{|pw_i|\})}{\mu(\{|pw_i|\})},$$

where σ and μ are the standard deviation and mean of the password lengths, respectively. A low LenCV (e.g., $\leq$0.2) indicates that all passwords have similar lengths, while a high LenCV (e.g., $\geq$0.4) suggests substantial variation in length across a user's passwords. For $m = 2$, LenCV remains interpretable as a normalized measure of the length difference between the two passwords; we therefore apply it to all groups. *Applicable for: all groups ($m \geq 2$).*

Longest common substring coverage. To capture contiguous substring reuse (an aspect not fully reflected by the order-insensitive 2-gram cosine similarity), we compute the longest common substring (LCS) coverage between password pairs:

$$\text{LCSCov}_{ij} \;=\; \frac{|\text{LCS}(pw_i,\, pw_j)|}{\min(|pw_i|,\, |pw_j|)}.$$

The user-level statistic is the maximum over all pairs:

$$\text{MaxLCSCov}_u \;=\; \max_{i<j} \text{LCSCov}_{ij}.$$

LCSCov captures content-level substring reuse: the extent to which a shared character sequence (e.g., a keyword or number string) is preserved across passwords. Unlike 2-gram cosine similarity, LCS requires contiguous character matches, making it more sensitive to the retention of a recognizable core substring. MaxLCSCov $\geq$0.5 means that at least half of the shorter password is preserved as a contiguous segment. *Applicable for: all groups ($m \geq 2$).*

Multi-source positional copy rate (MS-PosCopyRate). We measure how much of the target password can be explained by *combining* substrings from multiple sister passwords. Intuitively, this metric captures the fraction of target positions that are supported by at least one source password.

For each user, we randomly hold out one password as pw_{target}. For each $pw_i \in P \setminus \{pw_{\text{target}}\}$, we compute an LCS-based alignment with pw_{target}, and extract all matched substrings. To reduce noise, we retain only contiguous matches of length ≥ 2.

Let $\hat{\mathcal{A}}_i$ denote the resulting set of aligned position pairs. We then define the set of supported target positions as:

$$\mathcal{T}^* = \{t \mid \exists pw_i \in P \setminus \{pw_{\text{target}}\},\ \exists\,(s,t) \in \hat{\mathcal{A}}_i\}.$$

The user-level metric is:

$$\text{MS-PosCopyRate}_u = \frac{|\mathcal{T}^*|}{|pw_{\text{target}}|}.$$

In contrast to MaxLCSCov, which reflects the longest match from a single password pair, MS-PosCopyRate aggregates coverage across all sister passwords, capturing how multiple sources jointly contribute to the construction of the target.

Mutation ratio (MutRatio). Unlike MS-PosCopyRate, which measures multi-source positional coverage, MutRatio measures single-source edit distance normalized by the target password length. We select the password $pw_{\text{src}} \in P \setminus \{pw_{\text{target}}\}$ most similar to the target and define:

$$\text{MutRatio} = \frac{\text{Lev}(pw_{\text{src}}, pw_{\text{target}})}{|pw_{\text{target}}|}.$$

This metric captures the normalized edit cost from the

selected source password to the target. Its distribution reveals the extent to which users vary in single-source reusability, ranging from near-complete inheritance to substantial independent modification behavior. Note that MutRatio is not theoretically bounded by 1.0 because the denominator is the target password length. When the source password is longer than the target, the edit distance may exceed the target length.

MS-PosCopyRate and MutRatio provide complementary target-oriented perspectives. MS-PosCopyRate measures how much of the target can be collectively supported by sister passwords, whereas MutRatio measures the normalized edit cost from the selected source password to the target. A high MS-PosCopyRate with a low MutRatio indicates that the target is largely supported by passwords with limited modification, reflecting strong compositional reuse across multiple sources. *Applicable for: all groups ($m \geq 2$).*

C. Detailed analysis of multi-password characteristics

Global similarity patterns (MeanSim). Figure 3(a) shows the distribution of MeanSim_u across different m-strata. For all groups, the majority of users fall into the lowest similarity bin (0.0–0.2), accounting for $\sim$67% at m=2 and remaining above $\sim$50% even at $m\geq$6, indicating that low global similarity is the dominant pattern regardless of the number of passwords. A high-similarity tail ($\text{MeanSim}_u \geq 0.6$, combining the 0.6–0.8 and 0.8–1.0 bins) is most prominent at small m (e.g., up to $\sim$23% for m=2) but diminishes as m increases, suggesting that incorporating additional passwords introduces more diverse elements into the set, reducing average similarity. Overall, these results indicate that users' leaked password sets are typically globally dissimilar.

Local similarity patterns (MaxSim). Figure 3(b) shows the distribution of MaxSim_u across m-strata. For m=2, the distribution is heavily concentrated in the lowest bin (0.0–0.2, $\sim$67%), indicating that most password pairs show low similarity when only two passwords are available. As m increases, the distribution shifts toward higher similarity ranges, with the upper bins becoming increasingly dominant. In particular, the proportion of users with at least one highly similar pair ($\text{MaxSim}_u \geq 0.8$) rises steadily with m, reaching $\sim$78% at $m\geq$6. These results show that while most pairwise relationships are weak on average (see Figure 3(a)), the probability of observing at least one highly similar pair increases rapidly with m. This contrast demonstrates that low global similarity does not preclude the presence of strong local reuse patterns.

Similarity heterogeneity (VarSim). For m=3, the distribution is heavily concentrated in the lowest bin (0.00–0.04, $\sim$61%), indicating largely homogeneous similarity levels across password pairs (see Figure 3(c)). As m increases, the distribution shifts away from this low-variance regime toward intermediate variance ranges. In particular, the lowest-bin proportion decreases steadily (to $\sim$43% at m=4, $\sim$34% at m=5, and $\sim$28% at $m\geq$6). At m=5, the 0.04–0.08 bin accounts for $\sim$33%, slightly below the lowest bin ($\sim$ 34%). For $m\geq$6, the 0.08–0.12 bin is the largest within this stratum, accounting for $\sim$32% (see Table VII). The distribution remains dispersed rather than collapsing into a single high-variance mode. Notably, the high-variance tail (0.16–0.20) remains small ($\sim$1% at $m\geq$6), indicating that the shift is driven by the emergence of password sets with moderate similarity variance rather than extreme outliers. Under the $\text{VarSim}_u > 0.04$ criterion, the proportion of such users increases steadily from $\sim$39% at m=3 to $\sim$72% at $m\geq$6. These results indicate that larger password sets tend to show structured heterogeneity: a mixture of locally similar password pairs and globally dissimilar ones, rather than uniformly high or low similarity.

Hub-centered similarity structure (PivotSim). Figure 11(a) shows the distribution of PivotSim_u for $m\geq$3. For m=3, the distribution is concentrated in the lowest range (0.0–0.2, $\sim$48%), indicating that even the most central password has low average similarity to the others for these users. As m increases, the lowest bin shrinks steadily ($\sim$36% at m=4, $\sim$32% at m=5, $\sim$25% at $m\geq$6), while the intermediate range (0.2–0.4) expands and becomes the most prominent region from m=4 onward ($\sim$38%, $\sim$39%, $\sim$34%, respectively). The high-similarity tail ($\geq$0.6) remains small across all m ($\leq$18%), indicating that pivot passwords are typically only partially similar to the rest of the set, rather than near-identical to all others. Interestingly, the 0.6–0.8 range becomes more pronounced at $m\geq$6 ($\sim$15%), suggesting that stronger hub-like relationships may emerge more frequently in larger password sets. These trends suggest that larger password portfolios are increasingly likely to organize around a loose central reference, consistent with prior findings that users derive passwords from a few base elements [45, 55]. Notably, $\sim$20%–$\sim$35% of users across m-strata exhibit $\text{MeanSim}_u < 0.4$ but $\text{PivotSim}_u \geq 0.4$, showing that such centralized structures are not captured by global similarity alone and suggesting that multi-password models should account for such within-set structural heterogeneity.

Structural template patterns (TemplateCons). Figure 11(b) shows the distribution of TemplateCons_u across different m-strata. For all groups, the distribution is strongly skewed toward the lowest bin (0.0–0.2), accounting for the majority of users (e.g., $\sim$89% at m=2 and $\sim$77% at $m\geq$6). High-consistency values are rare beyond m=2, indicating that users seldom maintain a consistent character-class template across multiple passwords. Overall, these results suggest that password reuse is not driven by global structural consistency. Instead, compared with the strong substring overlap observed in MaxLCSCov, reuse behavior is primarily captured at the content level rather than at the coarse structural template level.

Length variability (LenCV). Figure 11(c) shows the distribution of LenCV_u across different m-strata. For small m, the distribution is heavily concentrated in the lowest range (0.0–0.2), accounting for $\sim$78% of users at m=2, indicating highly consistent password lengths. As m increases, the mass gradually shifts toward higher variability ranges: the proportion in the lowest bin decreases steadily (from $\sim$78% at m=2 to $\sim$29% at $m\geq$6), while the intermediate range (0.2–0.4) becomes dominant (reaching $\sim$55% at $m\geq$6). Using $\text{LenCV}_u > 0.2$ as a threshold for noticeable variation, the proportion of such users increases markedly with m (from

TABLE VIII: Interpretable examples for metric ranges ($m = 3$). Ranges are consistent with Table VII.

Metric	Range	Passwords ($m = 3$)	Explanation
MeanSim	0.0–0.2	`apple123`, `Qx!7Lm2`, `football9`	All pairs are unrelated, yielding very low average similarity across the entire set.
	0.2–0.4	`alexsmith`, `alexjones`, `marksmith`	One pair shares a prefix while the others remain weak.
	0.4–0.6	`alex123`, `alex4567`, `alexabcd`	All pairs share partial overlap, producing a moderate overall similarity level.
	0.6–0.8	`alexjon123`, `alexjon456`, `alexjonxyz`	Strong shared prefixes across pairs lead to relatively high similarity.
	0.8–1.0	`alex123`, `alex124`, `alex125`	Nearly identical strings yield consistently high similarity across all pairs.
MaxSim	0.0–0.2	`apple123`, `Qx!7Lm2`, `football9`	No pair shares meaningful similarity across characters or structure.
	0.2–0.4	`alex123`, `mark123`, `hello456`	Only weak overlap exists in at most one pair, resulting in low maximum similarity.
	0.4–0.6	`alex123`, `alex999`, `abc456`	One pair exhibits moderate similarity while others remain weakly related.
	0.6–0.8	`BlueSky9900`, `BlueSky8800`, `test123`	One pair is strongly similar with clear shared structure.
	0.8–1.0	`BlueSky99`, `BlueSky98`, `apple123`	One pair is nearly identical, resulting in very high maximum similarity.
VarSim	0.00–0.04	`alex123`, `alex124`, `alex125`	All pairwise similarities are nearly identical, leading to extremely low variance.
	0.04–0.08	`alex12`, `alex99`, `john12`	Slight variation across pairs produces a small but non-zero variance.
	0.08–0.12	`alex1234`, `alex1299`, `pass456`	Mixed similarity levels across pairs lead to moderate variance.
	0.12–0.16	`alex123`, `alex124`, `Qx!7Lm2`	One highly similar pair and others dissimilar produce high variance.
	0.16–0.20	`alex1234`, `alex1235`, `Qx!7Lm2`	Strong disparity across pairs results in very high variance.
PivotSim	0.0–0.2	`apple123`, `Qx!7Lm2`, `football9`	No password is similar to others, so no meaningful pivot structure exists.
	0.2–0.4	`alex123`, `bank456`, `alex999`	One password shows weak similarity to others, forming a limited pivot.
	0.4–0.6	`alex123`, `alex456`, `bank123`	One password is moderately central among the set.
	0.6–0.8	`alexjon123`, `alexjon456`, `alexjonxyz`	A pivot password emerges with strong similarity to others.
	0.8–1.0	`alex123`, `alex124`, `alex125`	One password is almost identical to others, forming a pivot.
TemplateCons	0.0–0.2	`apple123`, `Qx!7Lm2`, `abc!789`	Passwords follow different templates, resulting in very low structural consistency.
	0.2–0.4	-	For $m = 3$, TemplateCons $\in \{0, 0.421, 1\}$, so the ranges are unreachable.
	0.4–0.6	`apple123`, `hello456`, `abc!789`	Partial sharing of templates leads to moderate consistency.
	0.6–0.8	-	For $m = 3$, TemplateCons $\in \{0, 0.421, 1\}$, so the ranges are unreachable.
	0.8–1.0	`abc123`, `def456`, `ghi789`	All passwords follow the same character-class template, yielding high consistency.
LenCV	0.0–0.2	`apple123`, `hello456`, `world789`	All passwords have similar lengths, resulting in minimal variation.
	0.2–0.4	`apple123`, `hello45678`, `world7`	Length differences are moderate across the set.
	0.4–0.6	`short1`, `longpassword123`, `mid123`	Noticeable variation in length produces moderate dispersion.
	0.6–0.8	`a1`, `longpass1234`, `pass`	Large differences in length produce substantial variability.
	0.8–1.0	`a`, `longpassword12`, `abc`	Extreme length differences lead to very high variability.
MaxLCSCov	0.0–0.2	`apple123`, `Qx!7Lm2`, `football9`	No pair shares any meaningful contiguous substring.
	0.2–0.4	`alex123`, `all999`, `pass456`	Only short substrings are shared, resulting in weak overlap.
	0.4–0.6	`alex123`, `alex999`, `abc456`	Moderate-length substrings are shared between some pairs.
	0.6–0.8	`alex123`, `myalex1`, `pass456`	Long substrings are shared, indicating strong content reuse.
	0.8–1.0	`alex123`, `alex124`, `alex125`	Nearly identical substrings dominate, yielding very high overlap.
MS-PosCopyRate	0.0–0.2	{`apple123`, `Qx!7Lm2`} → `zebra@99`	Sister passwords provide almost no positional coverage of the target.
	0.2–0.4	{`alex123`, `pass456`} → `alk999`	Only a small portion of the target is covered by available inputs.
	0.4–0.6	{`alex123`, `pass456`} → `alex999`	Only one source contributes meaningful aligned substrings.
	0.6–0.8	{`alex123`, `123b`} → `alex123bobby`	Most but not all target positions are covered; partial suffix remains unsupported.
	0.8–1.0	{`alex123`, `123alex`} → `alex123alex`	Nearly all positions are covered by combining multiple passwords.
MutRatio	0.0–0.2	{`summer2023`, `pass456`} → `summer2024`	The closest password differs only slightly, requiring minimal modification.
	0.2–0.4	{`blueSky99`, `pass456`} → `blueSky2024`	Moderate edits are required to reach the target password.
	0.4–0.6	{`blueSky99`, `abc123`} → `blueSky@2024`	Multiple edits are needed, reflecting moderate transformation cost.
	0.6–0.8	{`apple129`, `pass456`} → `alexblue9`	Significant modification is required due to limited similarity.
	0.8–1.0	{`abc123`, `xyz`} → `BlueSky2024`	Almost complete transformation is needed, as no close source exists.

$\sim$22% at m=2 to over $\sim$70% at $m\geq$6). Overall, these results indicate that password length choices become increasingly diverse as the number of passwords grows, although most users still fall within low-to-moderate variability ranges.

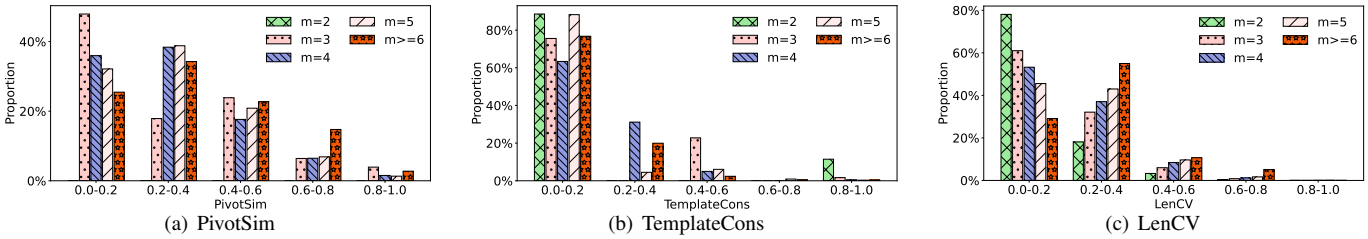
Contiguous substring overlap (MaxLCSCov). Figure 3(d) shows the distribution of MaxLCSCov$_u$ across different m-strata. For m=2, the distribution is relatively dispersed, with a majority of users in the lowest bin (0.0–0.2, $\sim$57%) but a non-negligible fraction already showing high overlap (0.8–1.0, $\sim$22%). As m increases, the distribution shifts sharply toward higher overlap ranges. In particular, the proportion of users in the highest bin (0.8–1.0) increases rapidly, from $\sim$22% at m=2 to $\sim$69% at m=5, and reaches $\sim$89% at $m\geq$6, becoming the dominant regime. Meanwhile, the lowest bin shrinks dramatically to only $\sim$1% at large m. These results indicate that users' password sets frequently contain at least one pair sharing a long contiguous substring, and that such local content reuse becomes highly prevalent as the number of passwords grows. In contrast, TemplateCons shows low consistency in character-class templates. These results suggest that password reuse is primarily driven by local content preservation rather than global structural consistency.

Multi-source positional support (MS-PosCopyRate). Figure 3(e) shows the distribution of MS-PosCopyRate$_u$ across different m-strata. As m increases, the mass shifts steadily toward higher coverage ranges. In particular, the highest bin (0.8–1.0) grows from $\sim$18% at m=2 to $\sim$49% at $m\geq$6, while the lowest bin decreases to $\sim$23%. These results indicate that positional coverage improves substantially with m, suggesting that multiple passwords collectively explain an increasing fraction of the target structure.

Mutation ratio (MutRatio). Figure 3(f) shows a polarized distribution of MutRatio, with mass concentrated toward the lower (0.0–0.2) and higher (0.8–1.0) ranges. However, the

TABLE IX: Representative examples of passwords cracked by PolyGuess but not by PassLLM-II. Categories are defined based on observable transformations.

Category	Transformation	Sister passwords	Target password
Concatenation / Recombination	Substring concatenation	ruslan, dogoda, 41'atline1, 410041, 23322332	ruslandogoda
	Partial extraction	lwwzq7, Lwwzq7, 98774682852zlata, 98774682852, 9877468285	98774682z
Formatting / Symbolic Variation	Hyphen insertion	dyqkv, dy-qkv, dy1qkv	dy1-qkv
	Symbol substitution	sandy23, 9901065, SANDY**, 2011_23	2011*23
	Structured extension	9069347256, 12354kd@, 12354Kd@, 12354kd, 12354Kd, 906934725	12354kd@mail.ru
Semantic / Identity Transformation	Name substitution	batman31, frame25557	superman31
Pattern Extension / Mutation	Keyboard pattern extension	26081994, 1q2w3e, 1o2l3e4g, 100775, 1O2l3e4g	1q2w3e4r5t
	Substitution of repeated digits	000999, 35104811z, kingofgame, 03004647, 112233	777888
	Character rearrangement	Fivgw6j76Z, Fivgw6j76Z123, Fivgw6j76Zabc123	Z67j6wgviF
	Initial-character substitution	0j32ubnv, aj32ubnv, lj32ubnv	mj32ubnv
Multi-source Composition	Name + number composition	guznova_galina, guznova_galin, 15270430, 27150430, 270387	galina15270430
	Identity composition	genius, andreev, alesha08, alesha123123o, alesha, aleshacreative14041971	aleshaandreev
	Date + name composition	angel01, a3g48bq1ke2y, 122412, 1208f2008, A3g48bq1ke2y, 12082008	12082008angel



(a) PivotSim (b) TemplateCons (c) LenCV

Fig. 11: Additional characterization results of multi-password behaviors. PivotSim captures hub-centered reuse structures, TemplateCons measures format-level consistency, and LenCV reflects password-length variability across a user's password set. Results for the other six metrics are shown in Figure 3.

TABLE X: Guessing success rates of different models under multi-password guessing scenarios.

Experiment setup		PolyGuess	Pass2Pass -T [23]	KNN Guess [30]	PassLLM -II [72]	Pointer Guess [65]	Pass2Edit [61]	PassBERT [68]	Pass2Path [39]	TarGuess -II [60]
Scenario: $pw_1 \sim pw_k \rightarrow pw_{k+1}$	Guess #									
#1 COMB-mix: $k \in [1, 4]$	10	**14.61%**	10.02%	12.79%	11.92%	12.78%	12.86%	10.76%	10.08%	6.49%
	100	**20.20%**	15.16%	17.94%	17.48%	18.13%	18.17%	15.88%	15.04%	13.12%
	1,000	**24.22%**	18.22%	21.36%	21.76%	21.78%	21.50%	19.16%	18.13%	17.46%
#2 COMB-mix: $k \in [5, 8]$	10	**27.63%**	7.42%	15.10%	21.53%	12.13%	12.82%	8.67%	8.00%	6.07%
	100	**35.53%**	14.20%	25.74%	30.62%	24.57%	22.25%	17.93%	16.74%	16.13%
	1,000	**39.36%**	24.15%	32.76%	35.15%	33.24%	30.27%	24.65%	24.15%	24.30%
#3 Cit0day-mix: $k \in [1, 4]$	10	**6.18%**	1.27%	1.59%	3.86%	4.79%	5.90%	2.87%	4.70%	3.51%
	100	**10.32%**	3.71%	2.83%	7.81%	8.06%	8.19%	4.66%	6.45%	6.66%
	1,000	**13.55%**	7.25%	4.54%	11.43%	10.89%	10.63%	7.33%	8.21%	9.33%
#4 Cit0day-mix: $k \in [5, 8]$	10	**4.33%**	0.21%	1.06%	1.48%	2.64%	3.07%	1.16%	1.69%	0.85%
	100	**8.67%**	0.74%	2.64%	4.86%	4.76%	6.25%	3.49%	3.81%	3.70%
	1,000	**11.10%**	3.81%	4.02%	7.93%	8.14%	8.79%	6.34%	5.81%	7.40%
#5 Naz-mix: $k \in [1, 4]$	10	**14.70%**	7.82%	10.31%	11.87%	13.65%	12.82%	10.50%	8.32%	7.34%
	100	**19.90%**	13.01%	14.91%	17.86%	18.64%	17.05%	14.08%	10.57%	12.09%
	1,000	**24.22%**	17.26%	18.29%	22.18%	23.06%	20.61%	17.65%	12.76%	15.15%
#6 Naz-mix: $k \in [5, 8]$	10	**17.41%**	2.03%	11.04%	10.58%	14.73%	13.74%	9.94%	8.18%	7.33%
	100	**24.86%**	4.13%	17.96%	21.49%	21.59%	20.00%	16.59%	11.70%	11.20%
	1,000	**30.02%**	14.37%	22.93%	27.27%	27.70%	24.58%	20.90%	14.87%	16.07%

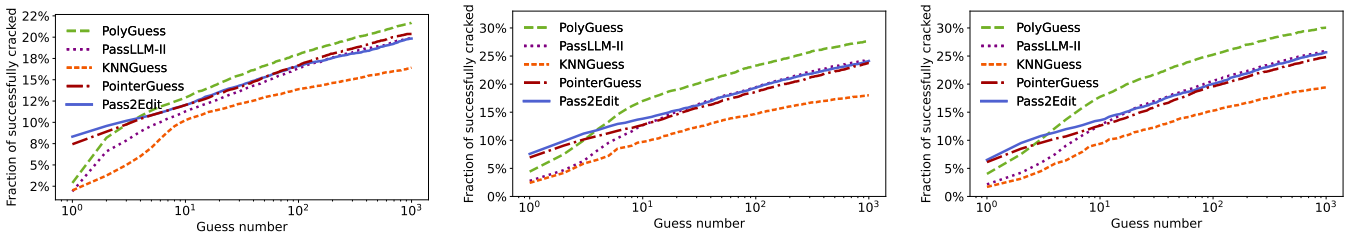
† The notation $pw_1 \sim pw_4 \rightarrow pw_5$ means that the attacker uses four known passwords of a user to guess the fifth unseen password.

dominant region varies across m-strata. The distribution is clearly polarized: for m=2, the highest displayed bin (0.8–1.0) accounts for $\sim$43%, while the lowest bin (0.0–0.2) accounts for $\sim$14%. As m increases, the mass shifts toward lower mutation ratios: the 0.0–0.2 bin rises to $\sim$37% at $m \geq 6$, while the 0.8–1.0 bin decreases to $\sim$21%. This indicates that with more available sister passwords per user, it becomes increasingly likely that the most similar source password requires only minimal modification to reach the target.

Overall, MS-PosCopyRate and MutRatio suggest a structured yet imbalanced generation process. A high MS-PosCopyRate indicates that a large fraction of target positions can be supported by available passwords collectively, while a low MutRatio implies that few edit operations are needed relative to the target password length.

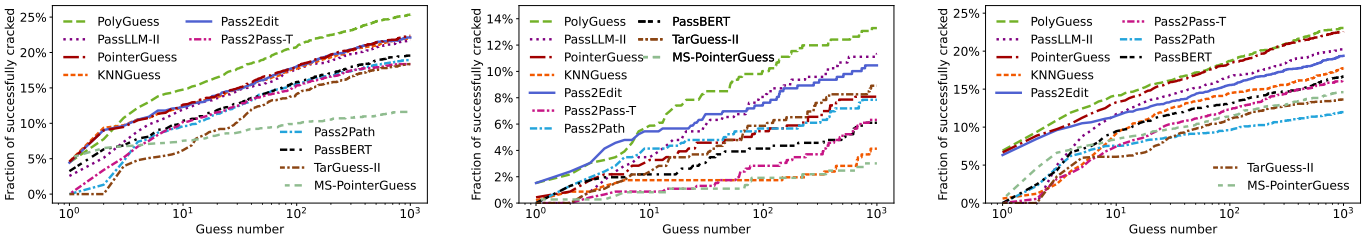
D. Supplementary experimental results

Figure 12 presents the success rates of PolyGuess compared with four representative baselines, including PointerGuess [65], Pass2Edit [61], KNNGuess [30], and PassLLM-II [72], under settings with a fixed number of input passwords. The results show that PolyGuess generally outperforms all baselines even in the single-password setting (k=1), demonstrating that its advantage is not solely dependent on multi-



(a) COMB: $pw_1 \rightarrow pw_2$. (b) COMB: $(pw_1, pw_2, pw_3) \rightarrow pw_4$. (c) COMB: $(pw_1, pw_2, pw_3, pw_4) \rightarrow pw_5$.

Fig. 12: Success-rate comparison under fixed-size multi-password guessing settings ($k = 1, 3, 4$). We compare PolyGuess with the strongest baselines, including PointerGuess [65], Pass2Edit [61], KNNGuess [30], and PassLLM-II [72]. As k increases, the advantage of PolyGuess becomes more pronounced.



(a) COMB: $(pw_1, pw_2) \rightarrow pw_3$ (b) Cit0day: $(pw_1, pw_2) \rightarrow pw_3$ (c) Naz: $(pw_1, pw_2) \rightarrow pw_3$

Fig. 13: Comparison of models in two-password setting. This enables a fair comparison with MS-PointerGuess [65], which is specifically designed for the two-password scenario. Results for other fixed-size settings are provided in Fig. 12. Across all datasets, PolyGuess generally outperforms its counterparts.

password inputs. As the number of available passwords increases, the success-rate gap further widens, indicating that PolyGuess can more effectively exploit additional contextual information compared to existing methods.

Table X reports the exact guessing success rates corresponding to Figure 7 for 10, 100, and 1,000 guesses. These results provide a detailed numerical comparison of all evaluated models across varying input sizes. At 10 guesses, PolyGuess consistently outperforms PassLLM-II [72] across all scenarios, with an average improvement of ∼65%. The gains are particularly pronounced in more challenging settings (e.g., Cit0day and Naz datasets with larger k), where the improvement can exceed 60% and even reach over 190% in certain cases. These results indicate that PolyGuess is especially effective when only a small number of guesses is allowed.

Figure 13 presents the comparison under the fixed two-password setting across different datasets. Even in this constrained scenario, PolyGuess generally achieves higher guessing success rates. Notably, its advantage is particularly significant on more challenging datasets such as Cit0day, where overall success rates are lower. Compared to MS-PointerGuess [65], which is specifically designed for the two-password setting, PolyGuess still maintains a clear lead, demonstrating its robustness and effectiveness. These results indicate that the improvements of PolyGuess are not solely due to scaling to more inputs, but also stem from its ability to model transformations and reuse patterns.

These results suggest that the advantage of our PolyGuess does not arise solely from increased model capacity or access to more inputs. Instead, the improvements are consistently observed across different datasets, numbers of guesses, and

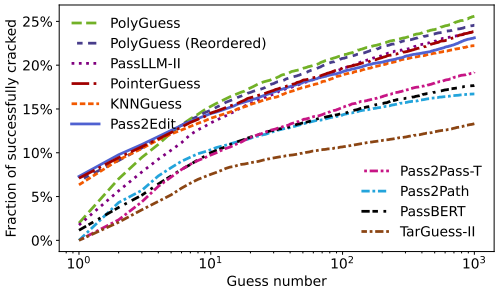


Fig. 14: Guessing success rates under breach-date-constrained and random hold-out evaluations. "PolyGuess (Reordered)" uses passwords from earlier breaches to guess a target from a later breach. "PolyGuess" and the baseline curves show the original random hold-out results for reference.

input settings, suggesting that Poly-1 provides a more effective training objective for predicting targets that require sparse but critical transformations.

E. Evaluation with breach-date constraints

To complement the random hold-out evaluation, we conduct a supplementary experiment using six datasets with publicly available breach dates: LinkedIn, 000Webhost, Twitter, Yahoo, Neopets, and ClixSense. Passwords leaked in earlier breaches serve as inputs, and a password leaked in a later breach serves as the target. As shown in Figure 14, our PolyGuess under this breach-date constraint (labeled "PolyGuess (Reordered)") achieves slightly lower success rates than under random hold-out evaluation, with the two curves remaining close across the tested guess numbers. This comparison suggests a small effect of breach chronology on guessing success rates, while PolyGuess remains effective when the available input pass-

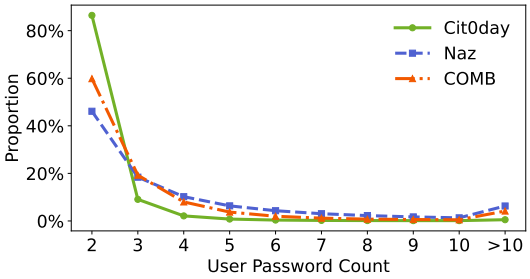


Fig. 15: Distribution of the number of leaked passwords per user across three mixed datasets (i.e., Cit0day [2], COMB [1], and Naz [6]).

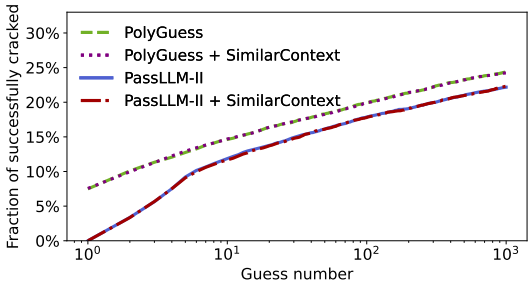


Fig. 16: Effect of centrality-based input reordering (SimilarContext) on guessing success rates. This input reordering results in nearly identical guessing success rates for both our PolyGuess and PassLLM-II [72].

words are leaked before the target breach. We leave a systematic study of inter-breach intervals and password evolution to future work, including the use of reliable creation timestamps to examine transformation direction.

F. Case study: additionally cracked passwords

Table IX presents representative examples of passwords cracked by our PolyGuess but not by PassLLM-II [72]. These cases show different ways of combining and modifying content from the input passwords, beyond simple reuse from a single source password.

First, some examples involve *compositional reuse*, where the target password is constructed by combining substrings from multiple inputs (e.g., `ruslan` + `dogoda` → `ruslandogoda`, or multi-source combinations such as names, dates, and numeric fragments). Such combinations cannot be fully explained by any individual source password. Second, we observe various *symbolic and structural transformations*, including hyphen insertion, symbol substitution, and structured extensions (e.g., `2011_23` → `2011*23`, or adding domain-like suffixes). These transformations require modeling fine-grained edits beyond direct copying. Third, we observe *sequence-level transformations*, including keyboard pattern continuation, substitution of repeated-digit sequences, and character rearrangement (e.g., `000999` → `777888`, or reordering alphanumeric segments).

These examples highlight that passwords additionally cracked by PolyGuess involve multi-source composition and non-trivial transformations, which are difficult to capture using single-source or copy-dominant generation strategies. This provides qualitative evidence that PolyGuess better exploits complementary signals across multiple inputs.

Overall, these results suggest that the Poly-1 loss can improve guessing success rates when the available input passwords differ in their relevance to the target. The case study illustrates the types of targets additionally cracked by our PolyGuess, while the controlled comparisons with PassLLM-II [72] provide quantitative evidence of the benefit of changing the training objective.

G. Effect of input password ordering

We investigate whether the ordering of input passwords affects the performance of multi-password guessing models.

The underlying language model adopts RoPE (Rotary Position Embedding), whose attention mechanism shows a distance-dependent decay effect: as the relative distance between tokens increases, their interaction strength may attenuate. This suggests that if informative passwords are placed far from the decoding position, their contribution might be weakened. Since the target password is unavailable at test time, we cannot directly rank passwords by similarity to the target. Instead, we adopt centrality-based input reordering (SimilarContext) as a proxy, assuming that the password most similar to others best captures the user's stable password behavior. Concretely, we compute pairwise similarities among input passwords and place the most central one closest to the decoding position to prioritize the most representative signal.

For any two passwords pw_A and pw_B, we define their similarity using the longest common substring length (LCS):

$$\mathrm{Sim}(pw_A, pw_B) = \frac{2 \cdot \mathrm{LCS}(pw_A, pw_B)}{|pw_A| + |pw_B|}, \tag{13}$$

The centrality score of $pw_i \in \mathcal{L}_p$ for $k \geq 2$ is:

$$S_{\mathrm{cen}}(pw_i) = \frac{1}{k-1} \sum_{j=1,\, j\neq i}^{k} \mathrm{Sim}(pw_i, pw_j). \tag{14}$$

We sort the passwords in ascending order of centrality:

$$S_{\mathrm{cen}}(pw'_1) \leq S_{\mathrm{cen}}(pw'_2) \leq \cdots \leq S_{\mathrm{cen}}(pw'_k), \tag{15}$$

and construct the reordered sequence:

$$\mathcal{L}'_p = (pw'_1, pw'_2, \ldots, pw'_k), \tag{16}$$

so that the most central password is placed closest to the decoding position, where its contribution is maximized. For $k = 1$, the input remains unchanged.

As shown in Figure 16, reordering input passwords has negligible impact on guessing success rates, indicating that both PolyGuess and PassLLM-II [72] are largely insensitive to input ordering. One plausible explanation is that the models can effectively use information from different input passwords, even when an informative password is located relatively far from the decoding position. This interpretation is compatible with the distributed positional matches measured in Section III-C, where useful information is spread across multiple inputs rather than concentrated in one.

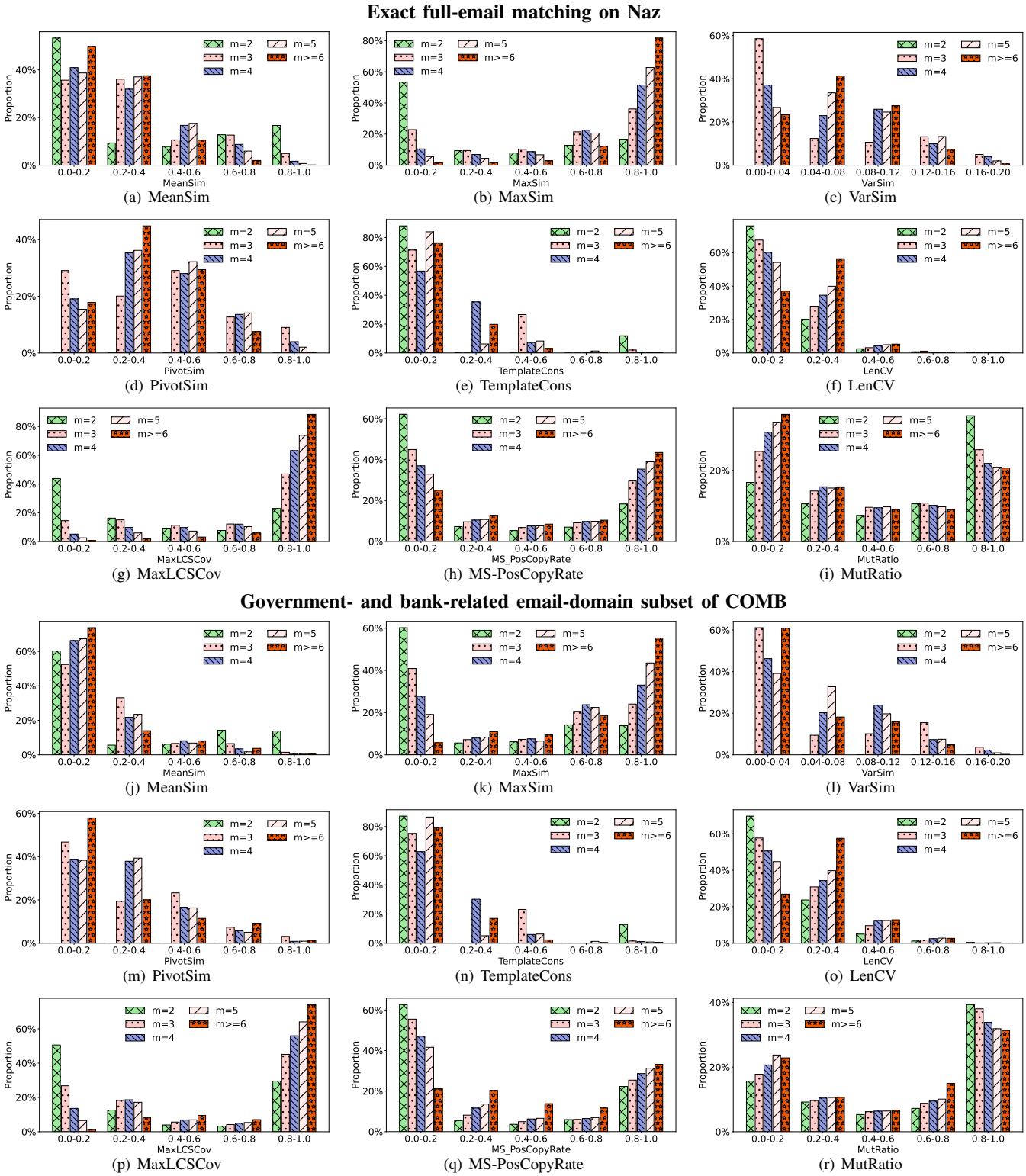


Fig. 17: Robustness to user-linkage criteria and sensitivity to service-risk proxies across nine reuse metrics. Figures 17(a)–17(i) show the Naz distributions under full-email matching, without requiring a shared password. Figures 17(j)–17(r) show the COMB distributions for accounts with government- or bank-related email domains, used as an imperfect proxy for higher-value organizational contexts. In both groups, the metrics are ordered row-wise as MeanSim, MaxSim, VarSim, PivotSim, TemplateCons, LenCV, MaxLCSCov, MS-PosCopyRate, and MutRatio. The broadly consistent distributions support our observations.