Success Rates Doubled with Only One Character: Mask Password Guessing

Yunkai Zou, Ding Wang, Fei Duan
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 traditional whole-password guessing attacks have been extensively studied, few studies have explored mask password guessing, where an attacker has somehow obtained *partial information* of the target victim's password (e.g., length and/or some characters) by exploiting various side-channel attacks (e.g., shoulder surfing, smudge, and keystroke audio feedback).**

To evaluate the threats posed by mask attackers with varied capabilities, we investigate *four major mask guessing scenarios*, each of which is based on different kinds of information exploited by the attacker (e.g., the length of the victim's password and some characters). For the first time, we systematically and comprehensively characterize the impacts of mask guessing that incorporate side-channel priors, personally identifiable information (PII), and previously leaked (sister) passwords, by proposing two password models: neural network-based PassSeq and probability statistics-based Kneser-Ney. Using the maximum likelihood estimation technique, we propose a new guess number estimation method to accurately and efficiently estimate the guess number required against the target password under a given password model. Extensive experiments on 15 large-scale datasets demonstrate the effectiveness of PassSeq and Kneser-Ney. Particularly, within ten guesses: (1) When a trawling attacker knows the character composition (without order) of the victim's 4-digit PIN, the success rate increases by 152% (from 14% to 35%); (2) When a PII-based targeted attacker knows the length of the victim's password, the success rate increases by 47%-82%; and (3) if this targeted attacker further knows *one character* of the victim's password (besides the length), the success rate generally *doubles*, reaching 7%-29% (and these figures will be 33%-73% for a targeted attacker that can exploit the victim's sister password).

To further validate the practicality of our mask guessing models, we collect real-world keystroke audio data from 11 popular keyboards (e.g., Apple, Dell, Lenovo) and replicate attacks where partial password information is inferred via acoustic side channels. Experiments show that our PassSeq significantly boosts the success rates of existing keystroke inference attacks, achieving an *additional* 5.6%-166.7% improvement within 10 guesses. This work highlights that mask password guessing is a damaging threat that deserves more attention.

I. Introduction

Text passwords remain the most widely used authentication method due to their simplicity of use, ease of change, and low deployment cost, and they are likely to maintain the dominant position in the foreseeable future [10], [11], [25], [72]. Accurately modeling password guessability can not only help precisely evaluate the security threats brought by attackers, but also help administrators design and deploy effective security mechanisms (e.g., password strength meters [13], [65] and honeywords [32], [62]) to protect systems and users. A common way to evaluate the strength of a password is to estimate its guess number under a given password guessing model/algorithm/tool [16], [34].

There have been dozens of password models designed for whole-password guessing, such as probabilistic context-free grammar (PCFG [64]), Markov [40], [45] and recurrent neural network (RNN)-based models [43], [49]. However, these models are *not* optimal for mask guessing, where the attacker has further obtained partial information of the victim's password (e.g., some characters and/or length). In general, password models like RFGuess [63], Markov [40], and RNN [43] are trained to predict the next character based on preceding characters in a password (i.e., characters are trained and predicted from a single direction). As a result, they are unable to utilize characters that follow a template to predict preceding masked characters. For example, in the template `**veu4eve*` (corresponding to the password `loveu4ever`, a stylized form of 'love you forever'), where $\star$ represents masked characters, they cannot utilize the following `veu4eve` to determine the probabilities of the first two masks.

Worse still, existing side-channel attacks (e.g., shoulder surfing and smudge attacks) pose significant threats by potentially exposing sensitive parts of the target victim's password. Among these attacks, shoulder-surfing attacks [9], [20], [36] are particularly concerning, as they enable attackers to directly see partial characters and/or the length of the target victim's password. In 2024, Hu et al. [30] systematically investigated password-masking practices across modern authentication systems and found that all evaluated mobile browsers implement *dynamic masking* (i.e., briefly revealing the most recently typed character before re-masking it). This mechanism makes certain characters trivially observable to nearby shoulder-surfers or camera-based side channels. Such momentary exposure on mobile devices directly motivates our mask guessing scenarios, in which attackers exploit short-lived character leaks to substantially increase guessing success rates.

Actually, shoulder-surfing attacks are frequently encountered in real-world scenarios, especially in public or semi-

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.241059
www.ndss-symposium.org

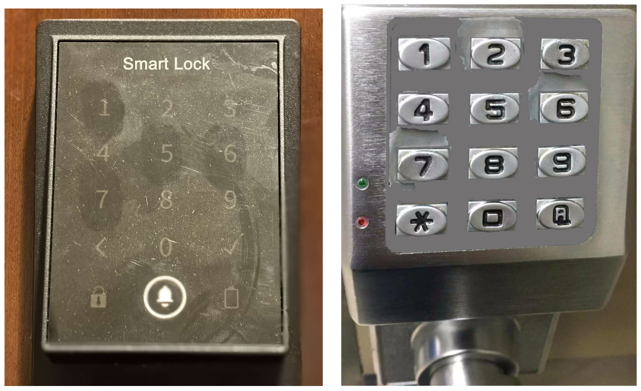


Fig. 1. Access control keypads that leave/leak input traces in the real world. We can clearly see that the left user's password consists of the four digits 1, 5, 6, and 7, while the right user's password contains 2, 6, and 7. Using this side-channel information, the attacker can perform mask password guessing to significantly increase her guessing success rate (see Fig. 6).

public spaces, and have alarmingly high success rates. For instance, Honan [28] reports that 85% of participants admitted being able to observe sensitive information on others' computer screens; the Ponemon Institute [1] finds that 91% of 157 visual hacking attempts were successful in office settings. Large-scale user studies further corroborate these findings: Marques et al. [42] report that about 20% of U.S. adults (and up to 52% of young users) admit to successfully snooping on others' phones within a single year. Aviv et al. [6] establish empirical baselines for mobile shoulder-surfing, showing success rates of 10.8% for 6-digit PINs after a single observation and 26.5% after multiple observations.

Besides shoulder surfing, there are also common and practical threats that motivate our work. For example, smudge attacks [6] allow attackers to determine the character composition of a victim's password by analyzing the traces left/leaked on input devices (see Fig. 1); keystroke audio feedback [7], [8], [12], [71] enables attackers to listen to keystrokes, revealing the probability distribution of individual characters and the password length; Thermal attacks [3], [4], exploiting keyboard heat maps, can reliably identify the last input password character with certainty. Notably, Alotaibi et al. [4] illustrate, through an example decoding path (Fig. 11), that the per-state decoding confidence along the inferred path can follow a monotonic increasing trend with position, with later positions showing higher conditional probabilities; Bicycle attacks [22], which monitor encrypted traffic between users and authentication servers, can infer the exact password length. Particularly, Harsha et al. [22] reveal that $>84\%$ of the Alexa Top-100 pages are vulnerable to password-length leakage.

If the attacker knows some characters and the exact length of the victim's password *with certainty*, she can directly perform template-based mask guessing. Alternatively, if the attacker has partial information about some characters (e.g., the probability distribution of characters obtained through keystrokes or the composition of characters obtained through smudges), she can incorporate this distribution (derived from acoustic side-channel analysis) as prior knowledge into the mask-based model to further improve the guessing success rate based solely on sound. In Sec. IV-F, we demonstrate that *our proposed password model, when integrated with acoustic-derived priors, can significantly boost the guessing success rates of existing acoustic side-channel attacks (e.g., [2], [21]) by 6%-167% within 10 guesses* (see Table X). This is achieved using keystroke data collected from 11 real-world keyboards (e.g., Apple, Lenovo, and Dell laptops).

Despite the increasing deployment of multi-factor authentication (MFA), password-only authentication remains prevalent. Recent reports indicate that merely 27%-34% of small and medium-sized enterprises have adopted MFA [55], and that about 60% of users still rely on passwords for their personal accounts [41]. Moreover, widely used dynamic masking interfaces (particularly on mobile browsers [30]) briefly display the most recently typed character before re-masking it, creating routine opportunities for partial or momentary character exposure. Besides UI-level leakage, attackers today can easily acquire identical keyboards, collect training data, and leverage side-channel inference [12], [21]. All this makes mask guessing attacks increasingly practical. Our study therefore provides timely insights into how modern systems behave under partial leakage and how improved modeling of masked passwords can enhance password strength evaluation, inform adaptive throttling and rate-limiting mechanisms, and support the design of more robust user interfaces.

A. Design challenges

In mask guessing, attackers can additionally exploit partial character information from the target password (compared with traditional whole-password guessing), and a given password template (e.g., `**veu4eve*`) usually indicates that the attacker further knows the length and pattern/structure/composition of the target password. This exemplary subtlety partially implies that *the exploration of mask guessing looks deceptively simple, but actually, it is rather challenging*. The following explains why.

Firstly, when applied to mask guessing scenarios, most state-of-the-art password models are unable to assign a probability to the missing character of a template efficiently. As briefly mentioned in [48], [50] and in-depth investigated in this work, character-level generative models like RFGuess [63], FLA [43] and Markov [40], [45] imply a causal order between password characters. This assumption asserts that causality flows in a single and specific direction during the generation process, that is, from the beginning to the end of the string. As a result, these password models are good at assigning probabilities to the following characters by the preceding characters but *not vice versa*. While training companion models with two directions may seem reasonable, it introduces two independent probabilistic models: the forward and reverse-trained models. This approach necessitates careful consideration of additional probabilistic smoothing, which can be complex and non-trivial.

PCFG-based password models [57], [64] are designed for token-based probabilistic modeling. They can only instantiate the basic structures (L/D/S) of passwords with substrings and cannot infer the missing positions of a password character by character, so these models cannot be directly employed for mask guessing. Deep generative models like PassGAN [26] and CPG [50] have inherent limitations. More specifically, they cannot assign a probability to the generated password, preventing them from sorting the guesses in an optimal order. Thus, a new technical route for mask guessing is needed, but how to design password models to overcome the identified weaknesses has not been systematically explored.

Secondly, it is inherently impractical to directly apply the existing password strength evaluation method, i.e., the Monte Carlo (MC) method [16], to estimate a password's guess number for mask guessing due to the accuracy vs. efficiency trade-off. Since the way of enumerating large-scale guesses is computationally intensive, previous works [37], [43], [67] have commonly employed the MC method to estimate the guess number of a password under a given probabilistic password model. Its basic idea is to sample from the given password model, i.e., generating random passwords according to the probabilities assigned by the model, and a more accurate result can simply be obtained by increasing the sample size.

However, when applied to mask guessing, the MC method [16] does not provide a guarantee that the randomly sampled passwords will necessarily match a given template (e.g., `**veu4eve*`). It needs to filter out a large number of samples that do not conform to the password template. This may result in an insufficient sample size, which does not guarantee the reliability of the evaluation results.

B. Our contributions

In summary, our contributions are four-fold.

- **Two mask password models**. We propose a Seq2Seq-based password model, namely PassSeq, which can accurately capture the impact of leaked characters on the overall security of the whole password and is applicable to various mask guessing scenarios. *For the first time*, we introduce the Kneser-Ney smoothing technique [35] into password guessing, and propose the Kneser-Ney password model, well mitigating the long-standing issues of overfitting and data sparseness.
- **New guess number evaluation method.** We adjust the generation mechanism of existing character-level autoregressive password models (e.g., Markov [40] and Backoff [40]) from traditional whole-password guessing to mask guessing, and propose a new guess number estimation method based on the Maximum Likelihood Estimation (MLE) technique for mask password guessing. With MLE, we can accurately and efficiently evaluate the password strength under each given template.
- **Extensive evaluation.** By conducting an extensive review of existing side-channel attacks on passwords, we model four different types of realistic mask attackers. The results demonstrate the severe dangers posed by various mask attackers. Particularly, when a targeted attacker knows just *one character* of the victim's password, the guessing success rate generally doubles within 1-20 guesses (see Figs. 5(a)-5(d)). Furthermore, within 10 guesses, our PassSeq can boost the effectiveness of existing keystroke acoustic-based side-channel attacks, achieving up to a 167% increase in guessing success rates (see Table X).
- **Some insights.** We provide a practical application and valuable insights. Specifically, our PassSeq can serve as a reliable password strength meter. Interestingly, mask attackers exploiting PII can achieve a 6%-14% higher guessing success rate when the first character of the target password is exposed, compared to when a randomly chosen character is exposed (see Figs. 5(a)-5(d)).

II. Related work and Background

In this section, we first introduce major password guessing models and then present our *threat model.*

A. Password guessing attacks

This work divides password guessers in existing research into three types according to different technical routes.

Type-1 research mainly employs *empirical knowledge and heuristic insights* to design various transformation rules (namely mangling rules), such as insertion, reversal, capitalization, and leet (e.g., `password`$\rightarrow$`passw0rd`). In 1979, Morris and Thompson [44] pioneered heuristic transformation rules to generate variants of words in the dictionary, and performed password guessing. Besides, there exist some open source password guessing tools (e.g., John the Ripper [46] and Hashcat [23]), taking the mangling rule techniques one step further by leveraging GPUs to automate the guesses at scale. While these heuristics are reasonably successful in practice, they are primarily applied in an ad hoc manner, rather than being constructed from a principled approach.

Type-2 research gradually moves away from the heuristic stage of relying on whimsical ideas and enters the scientific stage based on reliable *probabilistic statistical models*. In 2005, Narayanan and Shmatikov [45] proposed a password model based on the Markov chain, which estimates password probability from every two adjacent characters. In 2014, Ma et al. [40] introduced a number of normalization (e.g., End-symbol) and smoothing (e.g., Backoff) techniques on the Markov chain to overcome the data sparseness and overfitting problems, enabling password generation without being restricted by fixed-length substrings.

In 2009, Weir et al. [64] designed a password guessing model based on probabilistic context-free grammars (PCFG). PCFG [64] views a password as the concatenation of several independent strings of different lengths and character types (i.e., letters, digits, and symbols), corresponding to a specific password structure. Thus, it can generate password structure templates with grammatical rules obtained by statistics, and then fill them with character strings in the training dictionary. Since then, a series of improved PCFG-based methods have been proposed one after another. For example, in 2014, Veras et al. [57] discovered semantic strings in passwords through dictionary matching and classified them into a special category, strengthening the ability to analyze password semantics;

TABLE I

COMPARISON OF MAJOR MASK GUESSING LITERATURE (GAN/WAE=GENERATIVE ADVERSARIAL NETWORK/WASSERSTEIN AUTOENCODER).

Model / Paper	Password length	Probabilistic per-position	Character composition	PII/Reuse-based	Core modeling idea	Notes / Distinction
CPG [50] (Pasquini et al., S&P'21)	Required	Fixed p=0.5	✗	✗	Representation learning via GAN/WAE latent smoothness	These models primarily extend trawling password generation to conditional/*fixed* template-based settings, but lack explicit probabilistic modeling for masked positions and considerations for unknown PW lengths. While Xu et al. [68] and Yang-Wang [69] consider PII/reuse-based scenarios, they are both designed for *whole*-password guessing rather than mask guessing.
PassBERT [68] (Xu et al., Security'23)	Required	Fixed p=0.5	✗	✗	Bi-directional Transformer (pre-train and fine-tune)	
RankGuess-MASK [69] (Yang and Wang, S&P'25)	Required	Fixed p=0.5	✗	✗	Adversarial ranking with offline reinforcement learning	
PassSeq and Kneser-Ney (This work)	Unknown L considered	Fixed p=0.5; Side channel-aware	✓	✓	Seq2Seq with attention and Kneser–Ney smoothing	We quantify four mask guessing scenarios that incorporate side-channel priors, PII, and previously leaked (sister) passwords.

In 2015, Houshmand et al. [29] added the keyboard and multiword patterns to PCFG to enhance its ability to crack weak passwords. In 2022, Wang et al. [58] investigated an identification method for password segments and introduced the ReSeg-PCFG model. Their study revealed that the number of segments significantly influences password security. In 2024, Huang et al. [31] integrated two prominent password guessing models, PCFG [64] and the Markov-based OMEN [18], with Hashcat to fully leverage GPU acceleration, significantly improving the guessing efficiency of both models.

Type-3 research is mainly based on *machine/deep learning*, taking more advantage of large-scale data to address the data sparseness and overfitting problems existing in traditional statistical password models. In 2017, Melicher et al. [43] introduced long short-term memory networks [27] to password guessing and proposed FLA (Fast, lean, and accurate). Like Markov [40], FLA is also trained to generate the next character of a password given the preceding characters. The difference is that FLA is able to automatically assign a small (but non-zero) probability to each character in the alphabet, instead of using smoothing techniques. Recently, Pasquini et al. [49] proposed a FLA-based universal password model, which relies on auxiliary data (e.g., email) to build a pre-trained model for the target group of users. In addition, there are several leading targeted password models, such as TarGuess [61], which can exploit victims' PIIs, and Pass2Path [47], which focuses on credential stuffing/tweaking attacks.

In 2021, Pasquini et al. [50] successfully mitigated the training collapse problem of PassGAN [26]. They achieved this by implementing a form of stochastic smoothing over the representation of strings, leading to a substantial improvement in the performance of GAN-based approaches. On this basis, they constructed two guessing frameworks, i.e., conditional password guessing (CPG) and dynamic password guessing (DPG). They also showed that existing password models that are designed for traditional whole-password guessing (e.g., PCFG [64], Markov [45], and FLA [43]) can be used for mask guessing by filtering the generated guesses through templates, but their guessing success rates are not high. Moreover, this approach would generate a lot of redundant guesses that do not conform to the template, which might be inefficient.

In 2023, Xu et al. [68] introduced pre-training and fine-tuning techniques for password guessing, and proposed PassBERT, a framework based on Transformer. The authors first pre-trained a password model with the masked language modeling objective using large-scale datasets. Then, they designed attack-specific fine-tuning approaches to tailor the pre-trained password model to three attack scenarios.

In 2025, Yang and Wang [69] proposed RankGuess, a password guessing framework based on adversarial ranking. By formulating the guessing task as a Markov Decision Process, RankGuess demonstrates improved performance over existing models across three major guessing scenarios (Trawling, PII-based, and Mask guessing scenarios). Recently, several large language model-based password models (e.g., PassGPT [51], PagPassGPT [53], and PassLLM [73]) have been proposed. However, these models are focused on traditional whole-password guessing, rather than mask guessing attackers that can exploit the victim's partial password information (e.g., length and/or some characters) through side-channel cues.

Summary. To the best of our knowledge, only a few studies have explicitly addressed the mask password guessing scenario, including the conditional password guessing (CPG) framework [50], PassBERT [68], and RankGuess-MASK [69]. However, these works focus mainly on canonical settings, where part of the password is already known or structurally fixed. *They overlook more realistic and practical variants*, such as side-channel attacks (see Fig. 10) that provide probabilistic information about character-level distributions (which we summarize and compare in Table I). Fundamentally, mask password guessing can be viewed as a problem of probability allocation/smoothing over missing characters. This raises an important question: Could traditional statistical smoothing techniques (e.g., Backoff [40]) be more suitable than complex neural architectures in this context? Furthermore, are there more effective smoothing techniques that better align with the nature of mask guessing? These questions remain largely unexplored in existing literature.

B. Threat model

We introduce a formal attacker model for real-world mask guessing scenarios, grounded in the partial information that can be exploited by the attacker. Based on the characteristics and granularity of the available information, we identify four

TABLE II
SUMMARY OF REPRESENTATIVE SIDE-CHANNEL CHARACTERISTICS FOR CONTEXTUALIZING OUR EVALUATED MASK-GUESSING SCENARIOS.[†]

Attack types and base success rates / prevalence / practical realism	Key experimental settings of representative studies	Experimental setup in this work
Shoulder-surfing [5], [42] ① Using video recordings of participants unlocking devices, Aviv et al. [5] report shoulder-surfing success rates of 34.9%/10.8% and 56.7%/26.5% for 4-digit/6-digit PINs after one and multiple observations. ② An anonymity-preserving survey by Marques et al. [42] finds that about 31% of participants admit to having looked through someone else's phone without permission within the past year. When weighted to the U.S. adult population, this corresponds to ∼20%. ③ Dynamic masking[‡] in major Android/iOS browsers makes partial character exposure a realistic threat and directly motivates our work [30].	① Aviv et al. [5] investigated shoulder-surfing attacks by showing 1,173 participants pre-recorded videos of users entering 4- and 6-digit PINs from different viewing angles. ② Using anonymous online list experiments via Google Consumer Surveys, Marques et al. [42] measured how often people looked through others' phones without permission ($N \approx 2,000$). ③ Hu et al. [30] investigated the password masking practice deployed by the Google CrUX Top-1K websites and major desktop and mobile browsers.	We consider nine partial password-leakage cases covering scenarios where the attacker exploits: (a) personally identifiable information (PII) of the target, (b) previously leaked passwords of the same user, and (c) no related information. Specifically, these include: (1) PII-based scenarios with length leakage; (2) PII-based scenarios (length + one char leak); (3) Reuse-based scenarios with length leakage; (4) Reuse-based scenarios (length + one char); (5) Trawling scenarios (the first character leak); (6) Trawling scenarios (length + 1st char leak); (7) Trawling scenarios (50% chars leak); (8) Trawling scenarios (50% chars + length); (9) Trawling scenarios with length leakage. Results for the PII/reuse-based guessing scenarios appear in Fig. 5, and results for the trawling guessing scenarios are in Table VII and Fig. 7.
Bicycle attack (length-leak) [22] Harsha et al. [22] show that at least 84% of Alexa Top-100 websites fail to properly pad password fields, making them vulnerable to password-length leakage attacks.	Password lengths are inferred from TLS request sizes of Alexa Top-100 websites (e.g., Gmail), revealing the exact length via plaintext–ciphertext size correlation [22].	The risks introduced by password length leakage are quantitatively evaluated in our constructed scenarios (1), (3), and (9) above; see Fig. 5 for PII and Reuse-based guessing scenarios and Fig. 7 for the corresponding trawling scenarios.
Keystroke audio feedback [21] Under a threat model where an attacker records keystrokes using a nearby smartphone or through a remote voice call (e.g., Zoom), Harrison et al. [21] extract mel-spectrogram features and train a neural classifier that achieves up to 95% *character-level* accuracy for keystrokes.	Data were collected on a MacBook Pro 16-inch (2021), with 36 alphanumeric keys (0–9, a–z) each pressed 25 times, while keystroke audio was recorded in stereo at 44.1 kHz and 32-bit resolution using an iPhone 13 mini placed ∼17 cm from the keyboard [21].	We follow the threat model of Harrison et al. [21] and use per-character probability distributions inferred from keystroke audio as priors for our PassSeq. Note that Harrison et al. [21] collected repeated presses of the same key rather than actual passwords. To construct a more realistic guessing scenario, we collected over 1,500 keystroke samples of real PIN entries from 16 users across 11 popular keyboards (see Table X).
Thermal residue [4] Alotaibi et al. [4] show that, using AI-driven analysis of thermal images taken after password entry, an attacker can achieve success rates ranging from 92% (6-symbol) to 55% (16-symbol). When images are captured shortly after entry, attackers achieve 86% at 20s, 76% at 30s, and 62% at 60s.	Data were collected from 21 participants who entered passwords of 6, 8, and 12-16 characters on a Microsoft Wired Keyboard 600. An optris PI 450 thermal camera was used to capture thermal snapshots at 20s, 30s, and 60s [4].	We construct per-position disclosure probabilities inspired by the position-dependent recovery probabilities observed in ThermoSecure's decoding process [4] (see Fig. 11). We define a heuristic six-position disclosure profile, $[0.17, 0.20, 0.25, 0.33, 0.50, 1.00]$, capturing the monotonic increase in recovery probabilities observed in thermal-residue attacks (see Fig. 4).
Smudge attacks [14] Within 20 attempts, the smudge supported-guessing attack achieves ≈74% success rate with clean residue and ≈32% with noisy residue [14].	Smudge traces were extracted from smartphone photos (using Samsung Galaxy S4 with train/test = 219/93, N = 12); evaluated unlock, call, and social-app scenarios [14].	Smudges (see Fig. 1 for example) and thermal residue (e.g., [3]) reveal the set of characters composing the password but not their exact positions. We model this scenario using PIN data in Fig. 6 and quantify the incremental risk introduced by such partial composition leakage.

[†] See Table II in the full version for a more detailed summary, available at http://wangdingg.weebly.com/publications.html.
[‡] All major mobile browsers on Android/iOS implement dynamic masking (where a recently typed character is unmasked, but previously typed characters are masked), allowing shoulder-surfing attackers to observe characters during password entry.

generalized categories of attackers. Each attacker is characterized by information K, which constrains an effective guessing space $\mathcal{G}_K$ and informs the guessing strategy π_K.

General setting. Let $\mathcal{P}$ denote the password space and $\text{pw}^* \in \mathcal{P}$ the ground-truth password of length L. The attacker attempts to guess pw^* given partial prior knowledge K. We define the constrained candidate space as:

$$\mathcal{G}_X = \{\text{pw} \in \mathcal{P} \mid \text{pw satisfies } K\}\,, \tag{1}$$

and the goal is to construct a guessing policy π_X that maximizes the success rate under a given guessing budget.

Type I: Template attackers ($\mathcal{A}_{\text{template}}$). These attackers possess an exact mask template PT, e.g., `**veu4eve*`, where known positions are revealed and the others remain masked. The corresponding candidate set is:

$$\mathcal{G}_{\text{template}} = \{\text{pw} \in \mathcal{P} \mid \text{pw}[i] = \text{PT}[i]\ \forall i \text{ s.t. } \text{PT}[i] \neq *\}\,, \tag{2}$$

where c_i is the character at position i in the password, and $*$ is the masked character. Such templates can be obtained from shoulder-surfing (e.g., [20], [36]) and bicycle attacks [22].

In response to this attack scenario, Pasquini et al. [50], Xu et al. [68], and Yang and Wang [69] have respectively proposed the CPG, PassBERT, and RankGuess password models. These works employ various password templates to generate large-scale password guesses that conform to the templates, and then evaluate their password models on specific (template-conformed) test sets (that are different from the training sets). While all three studies highlight that some real-world side-channel attacks could obtain partial password information, they do not clearly articulate which specific parts of a password can be obtained under which types of attack scenarios.

To comprehensively evaluate the practical risks of the mask scenario, we extensively review existing studies that can obtain partial password information and further classify the mask scenarios to better align with real-world situations. Still, this basic mask scenario (i.e., the $\mathcal{A}_{\text{template}}$ attacker) will serve as a benchmark for comparing the advancements of the models proposed in this paper (see Table VII).

Type II: Distributional or compositional attackers ($\mathcal{A}_{\text{stat}}$). These attackers acquire statistical or compositional side-channel information about the target victim's password. One form of leakage is a positional character distribution $D = \{D_i\}_{i=1}^{L}$, where each D_i denotes the distribution over possible characters at position i. Specifically,

$$D_i(c) = \Pr(\text{pw}^*[i] = c), \tag{3}$$

which denotes the probability that character c appears at position i of the target password pw^*. Such distributions are typically derived from thermal residue on keyboards (e.g., [3], [4]) or keystroke audio feedback (e.g., [7], [12], [71]). Another common form is an unordered character multiset $M \subseteq \Sigma^*$, as revealed by smudge attacks on touchscreens (see Fig. 1), which disclose the set of characters used in the password without any positional information. In both cases, the attacker's guessing strategy involves either maximizing the joint likelihood of characters under D or enumerating permutations of M that form valid password candidates.

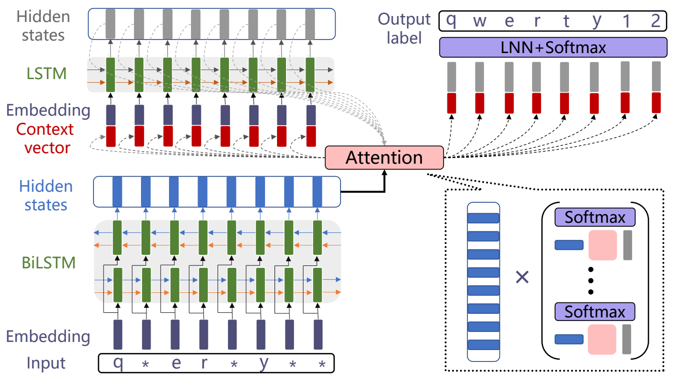


Fig. 2. The network architecture of our Seq2Seq. Here we show a high-level example of the architecture with one layer of LSTM/BiLSTM.

Type III: Contextual prior attackers ($\mathcal{A}_{\text{context}}$). These attackers enhance traditional mask guessing by incorporating user-specific prior knowledge into the attack process. Such contextual priors include personally identifiable information (PII), denoted as $\mathcal{I}$ (e.g., the victim's name, birthday, or email address), as well as previously leaked passwords, denoted as $\mathcal{R}$. By conditioning guesses on $\mathcal{I}$ or adapting patterns from $\mathcal{R}$, these attackers can significantly narrow the search space and increase the guessing success rate.

Type IV: Uncertain-length template attackers ($\mathcal{A}_{\text{length}}$). These attackers obtain partial template information PT but cannot determine the exact password length. Instead, a length candidate set $\mathcal{L} \subseteq \mathbb{N}$ is given. The attacker must consider multiple candidate lengths in parallel:

$$\mathcal{G}_{\text{length}} = \bigcup_{\ell \in \mathcal{L}} \{\text{pw} \in \mathcal{P} \mid |\text{pw}| = \ell \text{ and pw satisfies PT}\}\,. \tag{4}$$

Table II provides representative side-channel attacks (e.g., shoulder surfing [5], [42], keystroke audio feedback [12], [21], and thermal residue [3], [4]) and their key characteristics (including baseline success rates, prevalence, practical feasibility, and typical experimental settings) which contextualize our evaluated mask-guessing scenarios. The corresponding attacker capabilities are listed separately in Table III.

III. Our two new sequence password models: PassSeq and Kneser-Ney

We first elaborate on our PassSeq model, then introduce the Kneser-Ney smoothing technique [35] to password guessing, and finally present a new method for estimating the guess number required for mask guessing.

A. Our PassSeq password model

Mask guessing scenarios can be modeled as *character-level sequence-to-sequence language modeling tasks* that leverage prior information, such as PII (if available) and template sequences, to predict target password sequences. The Seq2Seq model, with its encoder-decoder architecture [54], offers significant structural flexibility, making it well-suited for this task. More specifically, the encoder can be composed of

TABLE III
ATTACKER CAPABILITIES CONSIDERED IN THIS WORK.

Mask attacker types	Password length	Character information with 100% certainty	Character information with some probability[†]	Character compositions[‡]	PII*	Leaked passwords*	Existing literature	Experimental results
$\mathcal{A}_{\text{template}}$	✓	✓					[50], [68], [69]	Table VII
$\mathcal{A}_{\text{stat}}$	✓		✓	✓			This work	Fig. 4, Fig. 6, Table X
$\mathcal{A}_{\text{context}}$	✓	✓			✓	✓	This work	Fig. 5
$\mathcal{A}_{\text{length}}$		✓					This work	Fig. 5, Fig. 7

[†] Keystroke audio feedback (e.g., [7], [8], [12]) or thermal residual (e.g., [3], [4]) can infer the likelihood of password characters (with position determined).
[‡] Smudge attacks (e.g., [6] and Fig. 1) are a type of side-channel attack that can infer the composition of password characters (with position undetermined).
*PII=Personally Identifiable Information. Besides the partial information of the victim's password (e.g., password length and/or some characters), the mask attacker can exploit the victim's name, birthday, and previously leaked passwords to further enhance their guessing success rates.

bidirectional LSTM [27], capturing bidirectional information of the password templates. This is particularly useful for templates where the continuity of local n-grams is disrupted (e.g., `⋆2⋆45⋆as⋆`). As we qualitatively analyze in Section IV-B, bidirectional encoding more effectively accommodates such templates than traditional n-gram smoothing. Further, the decoder structure ensures that passwords are generated character by character in an *autoregressive* manner from left to right (see equation 5), aligning better with the typical human behavior when creating and entering passwords. This also results in the model outperforming the PassBERT model [68] in mask scenarios. While transformer architectures [56] also perform well in bidirectional sequence tasks, the simplicity of the Seq2Seq model aligns with Occam's razor principle, making it a pragmatic choice for our application.

Our Seq2Seq network architecture is shown in Fig. 2. The encoder converts the input password character tokens to a representation vector and then feeds it to the decoder to generate output tokens based on a target vocabulary. The initial Seq2Seq model was proposed by Sutskever et al. [54], and originally applied to the English-French translation task. In 2015, Luong et al. [39] employed an attention mechanism, which makes use of *all* the encoder outputs and thereby improves the model performance. Thus, we also employ the attention mechanism to build our neural network.

When applying Seq2Seq for mask guessing scenarios, the password template PT (e.g., `q⋆er⋆y⋆⋆`) is encoded as the input of the model, and the corresponding password pw (i.e., `qwerty12`) is used as the output label. We define the probability of predicting the complete password pw$(=c_0, ..., c_n)$ given the corresponding template PT as

$$\Pr(\mathrm{pw}|\mathrm{PT}) = \prod_{\mathrm{i}=1}^{\mathrm{n}} \Pr(\mathrm{c_i}|\mathrm{c_0}, ..., \mathrm{c_{i-1}}, \mathrm{PT}). \tag{5}$$

In the training phase, we use cross-entropy to quantify the discrepancy between the outputs and the corresponding labels. In the generation phase, we can use beam search or our proposed mask search algorithm (see Algorithm 1) to generate guesses. We call the resulting model PassSeq, and put the details of the model structure in Table XIX of the full version, available at http://wangdingg.weebly.com/publications.html). We note that Pal et al. [47] and Xiu and Wang [66] have respectively proposed leading password reuse models based on Seq2Seq. However, their model architectures, training/generation paradigms, and core application scenarios are all different from our PassSeq.

B. Our Kneser-Ney password model

Traditional statistical password models (e.g., Markov [45]) generally rely on smoothing techniques to address the data sparsity issue inherent in n-gram modeling. In 2014, Ma et al. [40] incorporated the backoff smoothing method [33] into the Markov password model [45], resulting in the widely used Backoff password model. While effective in some scenarios, backoff smoothing tends to assign overly simplistic probabilities in the absence of high-order n-gram matches, limiting its generalization ability in sparse or unseen contexts. To better address these limitations, we explore the use of Kneser-Ney (KN) smoothing, a technique that has shown remarkable robustness in natural language modeling but remains underexplored in password modeling. To motivate this choice and highlight the shortcomings of traditional approaches, we provide a toy example in Appendix A of the full version to illustrate the differences between MLE, backoff smoothing, and Kneser-Ney smoothing in character-level prediction.

Model formulation. In practice, KN smoothing has emerged as one of the most theoretically grounded and empirically robust smoothing methods in language modeling, owing to its ability to balance count discounting and context diversity [24], [35], [52]. Given the analogous challenges in character-level password modeling (e.g., extreme sparsity and high contextual variability), we incorporate KN smoothing as a principled and effective solution within our probabilistic framework. Let the password substring be $pw_i^j = c_i c_{i+1} \cdots c_j$. The conditional probability of the next character c_i given context $\mathrm{pw}_{i-n+1}^{i-1}$ is:

$$\begin{aligned}\Pr_{\mathrm{KN}}(c_i \mid \mathrm{pw}_{i-n+1}^{i-1}) = {} & \frac{C(\mathrm{pw}_{i-n+1}^{i}) - D(C(\mathrm{pw}_{i-n+1}^{i}))}{\sum_{c_i} C(\mathrm{pw}_{i-n+1}^{i})} \\ & + \gamma(\mathrm{pw}_{i-n+1}^{i-1}) \cdot \Pr_{\mathrm{KN}}(c_i \mid \mathrm{pw}_{i-n+2}^{i-1})\end{aligned} \tag{6}$$

The discount function $D(x)$ is:

$$D(x) = \begin{cases} 0, & \text{if } x = 0 \\ x - \frac{n_1}{n_1 + 2n_2} \cdot \frac{n_{x+1}}{n_x}(x+1), & \text{if } x = 1, 2, 3 \\ D(3), & \text{if } x > 3 \end{cases} \tag{7}$$

where n_x is the number of n-gram substrings with frequency x. The backoff weight $\gamma(\cdot)$ is:

$$\begin{aligned}\gamma(\mathrm{pw}_{i-n+1}^{i-1}) = {} & \frac{D(1)N_1(\mathrm{pw}_{i-n+1}^{i-1}\cdot) + D(2)N_2(\mathrm{pw}_{i-n+1}^{i-1}\cdot)}{\sum_{c_i} C(\mathrm{pw}_{i-n+1}^{i})} \\ & + \frac{D(3)N_{3+}(\mathrm{pw}_{i-n+1}^{i-1}\cdot)}{\sum_{c_i} C(\mathrm{pw}_{i-n+1}^{i})}\end{aligned} \tag{8}$$

Here, N_1, N_2, and N_{3+} denote the number of distinct continuations of pw_{i-n+1}^{i-1} appearing exactly once, twice, and three or more times, respectively.

C. Our guess number estimation method

Considering that enumerating large-scale guesses is computationally intensive, previous works (e.g., [17], [37], [43], [67]) generally employ the Monte Carlo (MC) method [16] to estimate a password's guess number (given a password probabilistic model) if the guesses are in descending order of likelihood. However, MC is not suitable for mask guessing, since it lacks an efficient way to ensure that the sampled passwords match a given template. To address this limitation, we consider modifying the original MC estimation as follows:

$$C_{\Delta} = \sum_{\text{pw}\in\Theta} \begin{cases} \frac{1}{n\cdot p(\text{pw})} & \text{if } p(\text{pw}) > p(\text{pw}^*), \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Here, Θ is a sample of size N from the password space Γ, and C_{Δ} estimates the number of passwords more likely than pw*. By constraining pw to also match a given password template PT*, we can refine the estimation to match mask guessing scenarios. However, this adaptation is often impractical. Even with 10^7 samples from a trained Markov model [40], the average number of passwords matching a specific template remains low (e.g., 0-16), far below the recommended $N = 10^4$, leading to unreliable estimation.

Instead, we propose using Maximum Likelihood Estimation (MLE) to estimate password strength in mask guessing. Given that the number of passwords matching a template PT* is finite and known (denoted n), and that m of them satisfy $p(\text{pw}) > p(\text{pw}^*)$, we aim to estimate m.

To this end, we perform t independent sampling trials. In each trial, we uniformly sample N passwords from the set of candidates matching PT*, and count how many of them satisfy $p(\text{pw}) > p(\text{pw}^*)$. Let x_i denote the number of such passwords in the i-th trial. The MLE of p is then computed as:

$$\tilde{p} = \frac{1}{tN}\sum_{i=1}^{t} x_i. \tag{10}$$

This estimator is unbiased, and its variance satisfies $\text{Var}(\tilde{p}) = \mathcal{O}(1/(tN))$, indicating rapid convergence as the number of trials and sample size increase.

Full derivation and theoretical analysis of MLE (including variance and convergence rate) can be found in Appendix A.

Intuitively, MLE follows a principle similar to the *mark-recapture* estimation: by uniformly sampling n passwords from N total passwords (within a given template) and observing m passwords that are stronger than the target password, we can estimate its rank as $M{=}mN/n$. Repeating this process multiple times (e.g., 4-10) can produce an (approximately) unbiased estimate of the target password's rank, thereby aligning with the attacker's ranking strategy.

In practice, MLE is particularly suitable when estimating large-scale guess numbers for a given password under a given template (e.g., $\geq 10^8$ guesses). We provide a brief analysis of the computational overhead introduced by MLE in Section IV-F, highlighting its significant efficiency advantage over directly generating and ranking the entire candidate space to determine the guess number. All large-scale experimental results reported in Table X of the full version are obtained using our MLE-based estimation method.

IV. Experiments

Now we first elaborate on our password datasets and then fairly/comprehensively evaluate our proposed PassSeq and Kneser-Ney with their foremost counterparts (i.e., Markov [40], Backoff [40], Hashcat [23], PCFG [64], OMEN [18], CPG [50], PassBERT [68], and RankGuess [69]) in typical mask guessing scenarios.

A. Our datasets

Datasets. We introduce 15 real-world datasets with over 3.4 billion passwords (see Table IV), including eight from English sites, six from Chinese sites, and one mixed (COMB, containing passwords from over 50 countries/regions). Among them, three datasets (i.e., 12306, Rootkit, and Clixsense) are associated with various PIIs (e.g., name, email, and birthday). For PII or reuse-based guessing scenarios, we obtain another two PII datasets (see Table V) and four password-pair datasets by matching email (see Table VI). These datasets are compromised by hackers or leaked by insiders, and have been publicly available for some time. Following the data cleaning method employed by [40], [60], [69], we remove non-password strings in the original dataset, including headers, descriptions, footnotes, and non-ASCII strings. We also remove passwords with length>30, because they are likely to be constructed by password managers or just junk information.

B. Template-based mask guessing scenarios

We compare our PassSeq and Kneser-Ney with eight state-of-the-art password guessers in typical mask guessing scenarios (i.e., the type-I attacker $\mathcal{A}_{\text{template}}$), and they are PCFG [64], OMEN [18], Markov [40], Backoff [40], Hashcat [23], CPG [50], PassBERT [68], and RankGuess [69]. The reasons for selecting these password models are as follows: PassBERT [68], CPG [50] and RankGuess [69] are state-of-the-art password models *specifically designed for* typical mask guessing scenarios; PCFG [64] remains the most influential structure-based guesser; Hashcat's mask mode is a widely used attack method among real-world attackers; Markov/OMEN/Backoff are leading character-level autoregressive models that rely on statistical smoothing techniques. Particularly, we adapt Markov and Backoff to mask guessing scenarios using our proposed generation algorithm (see Algorithm 1).

We make sure the eight approaches work on the same training (i.e., 80% Rockyou as with [50]) and test sets, and manage to use/obtain their codes shared/open-sourced by the original authors. For all model parameters, we follow the best recommendations. The specific setup is as follows.

- **PassSeq**. For the built-in Seq2Seq model, we set the context length to 16, the hidden size, and the embedding size to 256 and 128, respectively. See Table XIX (in the full version) for detailed parameters/model structure.

TABLE IV

BASIC INFORMATION ABOUT OUR 12 PASSWORD DATASETS (PW=PASSWORD).

Dataset	Language	Service type	Leaked time	Original size	Non-PWs/Non-ASCII	PW length>30	Removed	After cleaning	Remaining
Post Millennial	English	News outlet	May 2024	38,902	0	31	0.08%	38,871	99.92%
Wishbone	English	Mobile app	Jan. 2020	10,092,037	798	250	0.01%	10,090,989	99.99%
000Webhost	English	Web hosting	Oct. 2015	15,299,907	69,606	1,131,721	7.76%	14,098,263	92.24%
LinkedIn	English	Job hunting	July 2012	54,656,615	0	17,157	0.03%	54,639,458	99.97%
Yahoo	English	Portal	July 2012	453,391	10,709	1	2.37%	442,681	97.63%
Rockyou	English	Social forum	Dec. 2009	32,603,387	149,971	2,068	0.47%	32,451,348	99.53%
Taobao	Chinese	E-commerce	Feb. 2016	15,072,418	0	86	0.00%	15,072,332	100.00%
Sohu	Chinese	Portal	Oct. 2015	14,755,046	147,429	50	1.00%	14,607,567	99.00%
126	Chinese	Email	Dec. 2011	6,392,568	0	599	0.00%	6,391,969	100.00%
CSDN	Chinese	Programmer forum	Dec. 2011	6,426,872	0	125	0.00%	6,428,285	100.00%
Dodonew	Chinese	E-commerce	Dec. 2011	16,282,276	4,975	12,070	0.10%	16,265,231	99.90%
COMB	Mixed	Mixed	Feb. 2021	3,279,064,312	14,948,181	10,576,452	0.78%	3,253,539,679	99.22%

TABLE V

BASIC INFORMATION OF OUR PII DATASETS.

Dataset	Language	Items num	Types of PII useful for this work
12306	Chinese	129,303	Email, User name, Name, Birthday, Phone
Dodonew-PII	Chinese	80,758	Email, User name, Name, Birthday, Phone
000Webhost-PII	English	79,580	Email, User name, Name, Birthday
Rootkit	English	69,418	Email, User name, Name, Birthday
ClixSense	English	2,222,045	Email, User name, Name, Birthday

TABLE VI

BASIC INFORMATION OF OUR PASSWORD REUSE DATASETS.[†]

Language	Training set setup	Size (pairs)	Test set setup	Size (pairs)
Chinese	CSDN→126	97,915	CSDN→12306	12,635
Chinese	CSDN→12306	12,635	CSDN→126	97,915
English	000Web.→Clixsense	150,273	000Web.→Yahoo	36,936
English	000Web.→ClixSense	150,273	000Web.→LinkedIn	231,452

[†] $A \rightarrow B$ means that a user's password at service A can be exploited by an attacker to help attack this user's account at service B. 000Web.=000Webhost.

- **PCFG**. We use the latest open-source implementation of the PCFG cracker [64] and follow the default grammar extraction and generation pipeline.
- **OMEN**. We use the public release of OMEN [18] and execute it with its recommended default configuration.
- **Kneser-Ney**. We set the order to 7 (i.e., 8-gram) based on our experimental results (see Fig. 3).
- **Markov/Backoff**. For Markov, we set the order to 3, and adopt the Laplace and End symbol technique recommended by [40]. For Backoff, we set the count threshold to 10 as recommended by [40].
- **Hashcat**. We employ the mask attack mode with the given password templates (Markov optimization).
- **CPG**. We use the source code of CPG [50] (i.e., PasswordAE) with the default parameters.
- **PassBERT**. PassBERT [68] provides two pre-trained variants: one on natural language and one on password data. We use the latter due to its higher cracking rates.
- **RankGuess-MASK**. We run the authors' provided source code with the recommended hyperparameters [69]. See our full version for details.

At IEEE S&P'21, Pasquini et al. [50] evaluated existing password models (e.g., Markov [40] and FLA [43]) in mask guessing scenarios by filtering the generated guesses through templates. However, they also revealed that this solution has two main drawbacks: (1) It's costly and storage-demanding; (2) It's difficult for such models to generate relatively low-probability guesses. For a fair comparison, we first closely follow the experimental setup in [50], [68], [69] (e.g., the

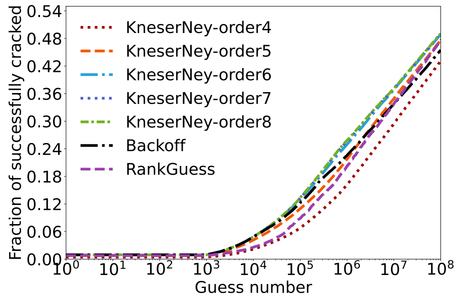


Fig. 3. We evaluate our Kneser-Ney password model with different n-gram orders (n=4, 5, 6, 7, 8), alongside Backoff [40] and RankGuess [69] baselines. Results show that the guessing success rate of Kneser-Ney is slightly higher when n=7/8. Therefore, we set n=7 for all subsequent experiments.

following template classification method is exactly the same as [50], [68], [69]). Then, we propose a generation algorithm (see Algorithm 1) and adapt the existing character-level autoregressive password models (including Markov [40], [45] and Backoff [40]) for mask guessing.

More specifically, we first construct a test set consisting of password templates, where the character set includes 94 printable ASCII characters (excluding space) and a wildcard/mask symbol $*$. Each password template (PT) is extracted from a randomly sampled password in a validation set V. We use LinkedIn as the validation set and retain only passwords with length $\leq$30, resulting in 6×10^6 unique passwords.

For each template, we uniformly sample a password pw from V and independently mask each character in pw with probability p=0.5 (the rationale for this choice is discussed later). We then retain only the templates that contain at least four observable characters and at least five wildcards, following [50]. Finally, we extract all passwords in V that conform to a given PT, forming the candidate set V^{PT}. Depending on the size of V^{PT}, the templates are divided into four classes.

- T_{common}={PT$|$ $\forall$PT, $|V^{\text{PT}}| \in [1,000, 1,500]$}
- T_{uncommon}={PT$|$ $\forall$PT, $|V^{\text{PT}}| \in [50, 150]$}
- T_{rare}={PT$|$ $\forall$PT, $|V^{\text{PT}}| \in [10, 15]$}
- $T_{\text{super-rare}}$={PT$|$ $\forall$PT, $|V^{\text{PT}}| \in [1, 5]$}

Each class contains 30 templates. This setup corresponds to the $\mathcal{A}_{\text{template}}$ attacker, that is, the attacker knows the length and some characters (with certainty) of the target password. Essentially, template-based mask guessing inherently relies on the mask probability p to generate the corresponding password templates. Choosing a value of p that is too high

TABLE VII

PROPORTION OF AVERAGE MATCHED PASSWORDS OVER THE BIASED PASSWORDS TEST-SET DIVIDED INTO FOUR CLASSES (RG=RANKGUESS).†

Templates class	Markov-d [40]	Markov-m	Backoff-d [40]	Backoff-m	Hashcat [23]	PCFG [64]	OMEN [18]	CPG [50]	RG-MASK [69]	PassBERT [68]	KN-m	PassSeq
Common [1,000-1,500]	0.2806	0.6457	0.4753	0.8179	0.2803	0.6104	0.3337	0.6312	0.6551	0.4548	**0.8366**	0.7760
Uncommon [50-150]	0.1041	0.4624	0.4720	0.8038	0.1832	0.6291	0.2123	0.6539	0.5616	0.3538	**0.8823**	0.7957
Rare [10-15]	0.1709	0.6162	0.5273	0.8879	0.1823	0.8441	0.2590	0.6601	0.7193	0.4633	**0.9244**	0.8947
Super-Rare [1-5]	0.0580	0.2830	0.1656	0.4094	0.1066	0.1902	0.0285	0.2783	0.3147	0.1931	0.4270	**0.4879**

† The evaluation metric is the average guessing success rate across 30 password templates within each class. The guessing success rate for each template is calculated as $N_{\text{intersection}}/|V^{\text{PT}}|$, where $N_{\text{intersection}}$ represents the intersection between the candidate passwords and the passwords in the validation set (i.e., V^{PT}). "-d" means default mode: We directly generate 10^9 candidate passwords using the original/default generation methods of these models (These generated passwords generally include a significant number that does not match the provided template), and then calculate the guessing success rate; "-m" means mask mode: We use Algorithm 1 to generate 10^6 candidate passwords that naturally match each provided template.

or too low may bias the evaluation of different models in mask guessing scenarios. More specifically, a *high* probability (e.g., p=0.8, as in `ab******`) produces too many masked positions and makes the task closely resemble whole-password guessing, while a *low* probability (e.g., p=0.2, as in `aaaaaaa**`) results in an extremely small search space that can be nearly exhausted by simple enumeration (generally within 10^6 guesses, e.g., the total space of a template with three masked characters is only 94^3). Therefore, we adopt p=0.5, which is consistent with prior work (e.g., CPG [50], PassBERT [68], and RankGuess-MASK [69]) and provides a balanced and realistic evaluation setting. In addition, we introduce three masking strategies and report their comparative results in Appendix C of the full version. Overall, masking strategies have only a marginal effect on guessing success rates, and p=0.5 is a simple yet effective setting for our study.

Note that Hashcat [23], CPG [50], PassBERT [68] and RankGuess-MASK [69] can be directly applied to mask password guessing scenarios. We employ each approach to pre-generate 10^6 passwords for each template. For existing sequence models (i.e., Markov [40], OMEN [18], and Backoff [40] and PCFG [64]), we first enable them to pre-generate 10^9 passwords in descending order of probability. Then we filter out passwords that conform templates PTs from these guesses, and match V^{PT} to obtain the guessing success rates. We call this approach the default mode of a sequence model (i.e., "-d" in Table VII). While these models are not inherently designed for mask guessing, they can be better applied to this attack scenario with proper adjustments. We call this approach the mask mode of a sequence model (i.e., "-m" in Table VII).

Algorithm 1 illustrates how to adapt the generation mechanism of sequence models from traditional whole-password guessing (default mode) to template-based guessing (mask mode). The core idea is to perform depth-first search (DFS) under template and probability constraints. For example, given a password template `123*5*`, each trained sequence model M (including our PassSeq and Kneser-Ney) provides a character distribution for each wildcard position $\star$. The constrained DFS procedure for generating guesses that naturally match a given template (e.g., `123*5*`) generally consists of four steps. First, we set a minimum probability threshold $\mathcal{T}$ and initialize DFS from left to right. Second, if the character at the current position is known (e.g., `5` in `123*5*`), we directly append it to the current prefix `s`(=`1234`). Third, if the position corresponds to a wildcard $\star$, we enumerate candidate characters `c` whose conditional probability satisfies

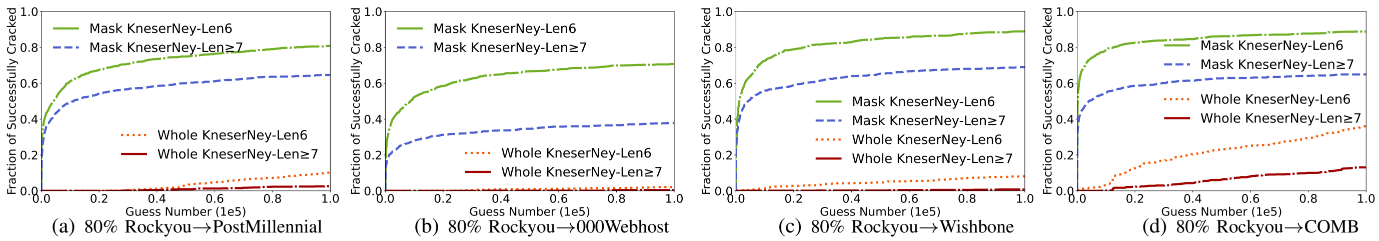
Algorithm 1: PW generation for mask guessing.

```
Input: Template PT, Model, Threshold $\mathcal{T}$, CharSet.
Output: Generate passwords pw that match the given template.
DFS(pw, p, i) : /* p is the probability of password prefix; i
    is the $i$-th position of pw; i < len(pw); i initialize to 0 */
if $i = (PT.length - 1)$ then
    return pw
if $PT[i] = *$ then
    for $c \in CharSet$ do
        pw+ = c
        if $Model.calculate(pw) < \mathcal{T}$ then
            continue
        else
            p = Model.calculate(pw)
            DFS(pw, p, i + 1) /*Search from the next position.*/
else
    pw+ = PT[i] /* Add the unmasked character directly. */
    if $Model.calculate(pw) < \mathcal{T}$ then
        return
    else
        p = Model.calculate(pw)
        DFS(pw, p, i + 1)
```

$\Pr_{\text{M}}(\text{s}||\text{c}) > \mathcal{T}$. Fourth, we recursively repeat the above process until the end of the password template is reached.

Results. Table VII reports the comparison results across four template classes. We observe that, when adapted to the mask guessing scenario, traditional statistical models (e.g., Markov [40] and Backoff [40]) achieve substantial improvements in guessing capability (see the difference between the "-d" and "-m" columns). Surprisingly, Backoff-m outperforms state-of-the-art neural models that specifically tailored for mask guessing (including CPG [50], PassBERT [68], and RankGuess-MASK [69]), across all template categories. This demonstrates that employing statistical smoothing to estimate character-level probabilities is a promising and effective strategy for tackling the mask password guessing problem.

In particular, our Kneser-Ney model (KN-m) achieves the highest guessing success rates among all baseline models (excluding PassSeq) across the four template classes. It is especially effective in the *rare* category, where data sparsity at higher-order n-gram levels poses significant modeling challenges and smoothing becomes crucial. While slightly behind KN-m and Backoff-m on the first three higher-frequency template categories, our PassSeq demonstrates its greatest strength in the most challenging *super-rare* class. Specifically, PassSeq achieves a success rate of 0.4879, outperforming the strongest neural baseline RankGuess-MASK [69] (0.3147) by a relative margin of 55.1%, and our enhanced Backoff-m



(a) 80% Rockyou→PostMillennial (b) 80% Rockyou→000Webhost (c) 80% Rockyou→Wishbone (d) 80% Rockyou→COMB

Fig. 4. Mask password guessing under probabilistic character leakage (i.e., the type-2 attacker $\mathcal{A}_{\text{stat}}$). We adopt a thermal-residue-based leakage model in which per-position leakage follows a front-to-back increasing key probability profile, inspired by Alotaibi et al. [4] (see Fig. 11 and Table I of [4]). The positional leakage probabilities are set to $[0.17, 0.20, 0.25, 0.33, 0.50, 1.00]$ for length-6 passwords and are extended to other lengths (i.e., Len$\geq$7 in the figure) via linear interpolation. For each dataset, we randomly sample 1,000 passwords and construct corresponding probabilistic templates. Each point (X, Y) in the figure represents that, at X guesses, a proportion Y of password templates have their original passwords successfully recovered.

model (0.4094) by 19.2%. These results highlight PassSeq's robustness in addressing extremely sparse password templates, where most prior models fail to generalize effectively.

To qualitatively compare bidirectional encoding (PassSeq) with n-gram smoothing (Kneser-Ney), we examine the templates from Table VII for which PassSeq achieves fewer average guesses than Kneser-Ney. Results show a general trend: PassSeq performs better than Kneser-Ney when masked positions are irregularly distributed, widely separated, or disrupt the continuity of local character sequences (e.g., `⋆am⋆⋆bon⋆`; see more examples in Table XX of the full version). Such templates are generally longer and contain multiple fragmented masked blocks, which introduce long-range dependencies that n-gram smoothing cannot effectively capture. PassSeq, benefiting from bidirectional encoding, aggregates contextual information across distant unmasked segments and remains effective even when local substrings are heavily interrupted.

Further, we have set up some general scenarios to show the impact of exposing/masking *different numbers of characters* on password security. Results (see Table X in the full version) show that disclosing just one *additional* character can *double* the attacker's guessing success rate (e.g., a 121.9% increase within 10^5 guesses on the COMB dataset using PassSeq).

Since PassSeq, Kneser-Ney, and our improved Backoff baseline (Backoff-m) consistently rank among the top three in guessing success rates and outperform state-of-the-art mask-guessing models such as RankGuess-MASK [69], PassBERT [68], and CPG [50], we mainly employ these three models in the following experiments to *quantitatively evaluate the threat posed by different classes of mask-guessing attackers*.

C. Distributional mask guessing scenarios

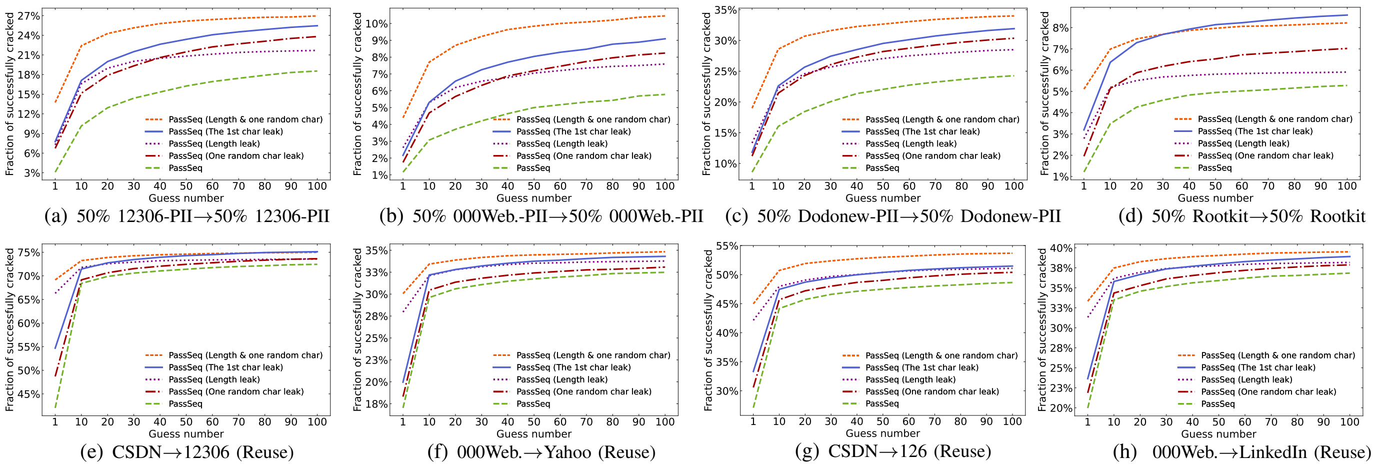
Considering that some side-channel attacks (e.g., thermal residue [3]) may not reveal any particular password character with 100% certainty, we set up two mask attack scenarios in which the attacker acquires statistical or compositional side-channel information about the target password.

Thermal attackers. To model probabilistic character leakage in mask guessing attacks, we construct per-position character disclosure probabilities inspired by empirical observations from prior thermal-residue attack studies. More specifically, we adopt a six-position recognizability profile, $[0.17, 0.20, 0.25, 0.33, 0.50, 1.00]$, inspired by the monotonically increasing position-wise decoding probabilities illustrated in Fig. 11 and Table I of ThermoSecure [4], which reflect progressively stronger confidence in character ordering as the decoding proceeds. We use these values as heuristic position-wise disclosure priors for character leakage. We then linearly interpolate these values to any password length L to obtain a position-dependent probability sequence q_i. Each q_i is clipped to $[0, 1]$ and perturbed with Gaussian noise $\mathcal{N}(0, 0.05^2)$ to account for real-world variability (e.g., surface material). Experimental results show that the guessing success rates of disclosing characters (with thermal-residue-based leakage probabilities) are 2.47-33.67 times higher than those of not disclosing any password character (i.e., the Whole Kneser-Ney lines in Fig. 4). Fig. 14 (in the full version) additionally considers other leakage probabilities that follow the increasing-probability trend observed in thermal-residue attacks [3], [4], and the conclusions still hold.

Smudge attackers. In practice, smudges in keypads (see Fig. 1) can reveal the target victim's character composition information (without order). We evaluate the impact of this threat on real-world PINs using the Amitay-4-digit dataset, which consists of 204,432 human-chosen 4-digit PINs collected by Daniel Amitay with the iOS application "Big Brother Camera Security" in 2011. Specifically, we assume that $\mathcal{A}_{\text{stat}}$ knows the digits composing the target 4-digit PIN through keystroke traces but does not know their exact positions (which is also aligned with the experimental setup in [12]). To model this, we modify the PassSeq model to limit the search space to the known digits of the target user when generating guesses. We then calculate the guessing success rate in this scenario and compare it to the scenario where the attacker has no prior knowledge. Fig. 6 shows that, for a *single guess*, the success rate of $\mathcal{A}_{\text{stat}}$ is 255% higher than that of an attacker without any prior knowledge. When up to 10 guesses are allowed, the success rate increases by 152%, reaching 35%.

D. Contextual prior mask guessing scenarios

In the above, we have only considered the threat posed by mask attackers who possess partial information about the target victim's password, without leveraging the target victim's PII or previously leaked passwords. Yet, in reality, a large fraction of users (e.g., 37%-51% in [61]) build passwords with their own PII, and tend to reuse their existing passwords (e.g.,



(a) 50% 12306-PII→50% 12306-PII (b) 50% 000Web.-PII→50% 000Web.-PII (c) 50% Dodonew-PII→50% Dodonew-PII (d) 50% Rootkit→50% Rootkit

(e) CSDN→12306 (Reuse) (f) 000Web.→Yahoo (Reuse) (g) CSDN→126 (Reuse) (h) 000Web.→LinkedIn (Reuse)

Fig. 5. Mask guessing scenarios with PII/existing leaked password (i.e., the $\mathcal{A}_{\text{context}}$ attacker). The training and testing setup is recommended by [61]. On both Chinese and English datasets, disclosing password length and one password character significantly improves the attacker's guessing success rate. In Figs. 5(a)-5(d), with a single guess, the leak of password length and one character *doubles* the attacker's success rates (see Table VIII for the confidence statistics of the success-rate improvement), increasing by at least 165%±24% (95% CI: 124%-223%). See Table XIV in the full version for specific success-rate figures.

TABLE VIII
CONFIDENCE STATISTICS OF SUCCESS-RATE IMPROVEMENTS (**ONE CHARACTER+LENGTH LEAKAGE** VS. BASELINE) UNDER DIFFERENT SCENARIOS AND GUESS BUDGETS[†].

Scenarios	Top-1 (Mean±Std, 95% CI)	Top-10 (Mean±Std, 95% CI)	Top-100 (Mean±Std, 95% CI)
(a) 50% 12306 → 50% 12306	3.4150 ± 0.6312 (2.4405, 4.9938)	1.2331 ± 0.2207 (0.9289, 1.7605)	0.4893 ± 0.0665 (0.3752, 0.6356)
(b) 50% 000Webhost → 50% 000Webhost	2.7961 ± 1.1789 (1.3225, 5.9781)	1.5555 ± 0.3672 (0.9554, 2.3875)	0.8666 ± 0.1425 (0.5727, 1.1932)
(c) 50% Dodonew → 50% Dodonew	1.6532 ± 0.2439 (1.2416, 2.2259)	0.8900 ± 0.1036 (0.7060, 1.0787)	0.4029 ± 0.0500 (0.2974, 0.4872)
(d) 50% Rootkit → 50% Rootkit	3.4139 ± 1.0793 (1.3747, 5.9562)	0.9676 ± 0.2131 (0.6201, 1.3814)	0.5276 ± 0.1115 (0.2428, 0.7561)

[†]All values in the table are relative improvement ratios. Each result is obtained by performing 30 rounds of sampling with replacement over the entire test set, with 1,000 target users per round. Cells highlighted in gray correspond to scenarios where the success rates doubled compared to the baseline.

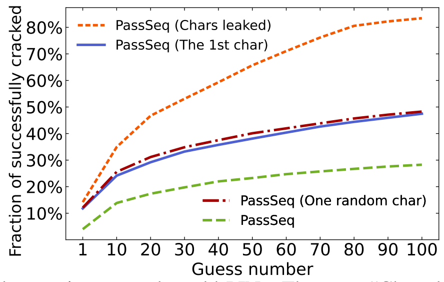


Fig. 6. Mask guessing on real-world PINs. The term "Chars leaked" means that the $\mathcal{A}_{\text{stat}}$ attacker knows the digit composition of the target victim's PIN but does not know the exact positions of the digits. PassSeq is trained on 50% Amitay-4-digit dataset, and is tested on the rest 50%. "The 1st char/One random char" means the 1st digit/a random digit of the victim is leaked.

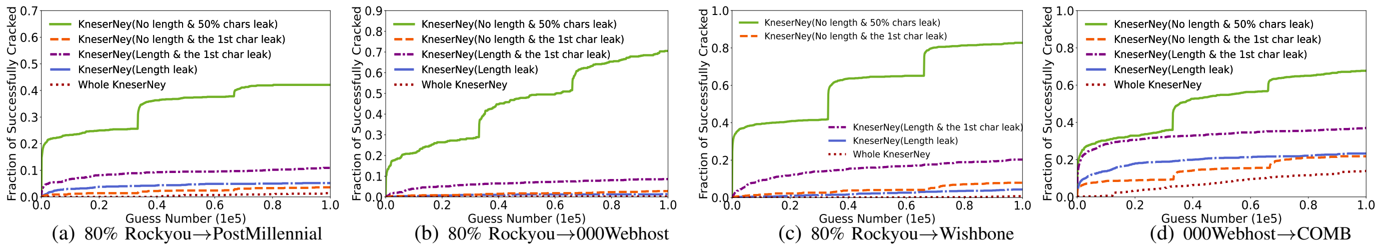
20%-59% of users [15], [61] directly reuse or simply modify their existing passwords). Thus, it is necessary and practical to evaluate the threat posed by a more powerful mask attacker knowing the target victim's PII or existing passwords. Tables V and VI summarize the basic information of our datasets.

PII-based attackers. For the PII-based mask attack scenario, we train our PassSeq on datasets containing PII (see Table V). More specifically, the training input for our PassSeq is the character sequence of the personal information sequence (e.g., `name|username|email|birthday`), and the output is the corresponding password sequence. When generating guesses, we input the target victim's PII sequence, using beam search to generate the corresponding guesses. Figs. 5(a) and 5(b) show that by merely knowing the victim's password length, the guessing success rates of the $\mathcal{A}_{\text{context}}$ attacker can increase by 63.48%-72.64% within 10 guesses, and this value *doubles* (+102.17%-129.32%) when one additional password character is leaked (see Table VIII for the confidence statistics).

Reuse-based attackers. For the reuse-based mask attack scenario, we train the PassSeq model by taking a user's existing password as input and their new password as output, following a training paradigm similar to Pass2Pass [47] and PointerGuess [66]. We model a realistic attacker $\mathcal{A}_{\text{context}}$ who knows both the existing password and the target password's length and/or one character. Here, our PassSeq serves as a representative model to evaluate the *additional* risks introduced by mask guessing in password reuse settings. Importantly, our goal is not to compare PassSeq with state-of-the-art reuse-based models (indeed, it achieves guessing success rates comparable to PointerGuess [66]). To validate our findings, we also apply the same evaluation to PointerGuess (see Fig. 13 in the full version), and observe consistent trends. Results (see Figs. 5(e)-5(h)) show that with only one guess, the guessing success rates of $\mathcal{A}_{\text{context}}$ increase by 62.30%-73.21% when both the password length and one character are disclosed, with the single-guess success rate reaching up to 69% (see Fig. 5(e)). All this reveals that mask guessing with users' PII or existing passwords is a damaging threat that needs more attention.

E. Uncertain-length mask guessing scenarios

To quantify the impact of *password length* information on mask guessing scenarios, we assume that the attacker only knows partial character information of the target password



(a) 80% Rockyou→PostMillennial (b) 80% Rockyou→000Webhost (c) 80% Rockyou→Wishbone (d) 000Webhost→COMB

Fig. 7. Length-uncertain mask guessing with/without one-character leakage (the $\mathcal{A}_{\text{length}}$ and $\mathcal{A}_{\text{stat}}$ attackers). We consider five attacker configurations with different prior knowledge: (1) no prior information (whole-password guessing), (2) unknown length with probabilistic character leakage (p=0.5), (3) unknown length with one leaked character, (4) known length without character leakage, and (5) known length with one leaked character. These scenarios collectively represent the upper (100% leakage) and lower bounds (0% leakage) of leaking length and/or one character. Our results show that at 10^5 guesses, leaking just one character (with unknown password length) at least doubles the attacker's success rate, achieving improvements from 1.57-14.50 times.

TABLE IX
PERFORMANCE COMPARISON ACROSS DIFFERENT MODELS[†].

Model	Training time	Model size	Password generation speed	
			Trawling (pw/s)	Mask (pw/s)
Kneser-Ney	00:05:45	3.0 GB	284,000	2,042,222
PassSeq	84:38:58	23.0 MB	610	151
RankGuess-MASK [69]	16:08:11	2.52 MB	/	30
PassBERT [68]	22:00:08	35.9 MB	/	320
CPG [50]	26:54:04	6.1 MB	/	677
Backoff[‡]	00:01:55	270.0 MB	297,600	2,195,121

[†] CPU: Xeon(R) Gold 6226R 2.9GHz; GPU: RTX 3090 (80% Rockyou).
[‡] We optimize the original Backoff implementation [40], achieving a several-fold speedup in password generation; see Appendix G of the full version.

but without knowing the exact password length. In this case, an intuitive strategy is to prioritize the selection of the top lengths counted in the training set as the priority choices for constructing the templates. More specifically, we choose the top-3 lengths from the training set to set three different lengths for each test template. Here, the top-3 lengths we select should satisfy the condition of being greater than or equal to the longest position of the leaked character in the target password that we know. For example, if we know that the character at the eighth position of the target password is `d`, then the selected top lengths should be $\geq$8.

Consider a toy example where the target password is `p@ssword`. A password template with masks is generated by masking each position with a 50% probability, and without loss of generality, assume the template `⋆@⋆sw⋆⋆d` is generated. Then, from the leaked dataset (i.e., the training set), $\mathcal{A}_{\text{length}}$ (who knows characters at positions 2, 4, 5 and 8, but *not* the total length) selects the top-3 common lengths ($\geq$8; e.g., 8, 9, and 10) and constructs three templates (i.e., `⋆@s⋆w⋆⋆d`, `⋆@s⋆w⋆⋆d⋆` and `⋆@s⋆w⋆⋆d⋆⋆`). $\mathcal{A}_{\text{length}}$ then generates an equal number of guesses for each template (1/3 of the total) to execute the attack. Similar to the PII/reuse-based guessing scenarios, we additionally consider three combined leakage cases in trawling scenarios: (i) length-only leakage, (ii) single-character leakage without length, and (iii) simultaneous leakage of both length and one character. As shown in Figs. 7(a)-7(d), at 10^5 guesses, leaking just one character (without password length) at least *doubles* the attacker's success rate, achieving improvements from 1.57-14.50 times.

F. Performance evaluation

Table IX shows that adapting existing statistical models such as Backoff [40] and Kneser-Ney (KN) to mask guessing scenarios significantly improves their generation speeds. In

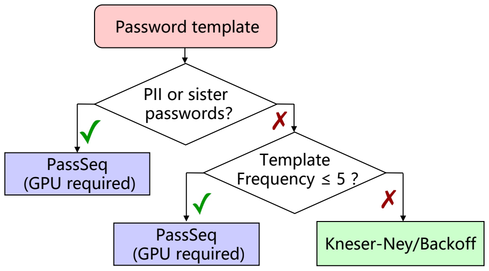
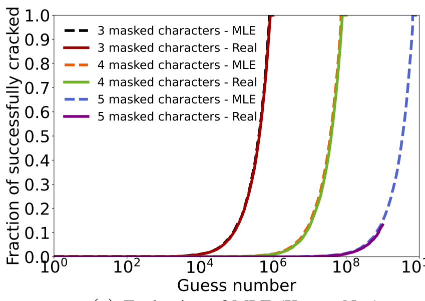


Fig. 8. Decision chart for mask password model selection. It is recommended to use our PassSeq model when the target victim's PII or sister passwords are available. For password templates with a frequency $\leq$5, PassSeq is preferred; otherwise, Kneser-Ney or Backoff is recommended.

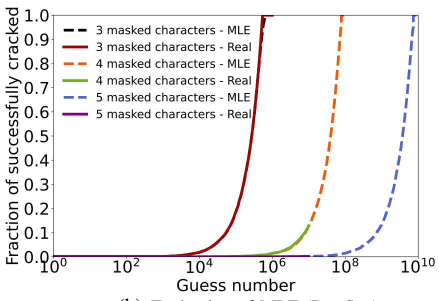
contrast, our PassSeq has slower generation speed in mask guessing scenarios (151 pw/s vs. 610 pw/s), primarily because mask generation relies on sequential token-by-token decoding with limited parallelism, compared to the more batch-friendly trawling mode. However, the slower generation speed of PassSeq is acceptable in practice. Mask guessing is typically employed in targeted online attacks, where the performance bottleneck lies in network latency and server-side throttling mechanisms [19], rather than in local computational cost. As such, PassSeq remains practical for real-world deployment.

It is worth noting that, besides achieving the highest overall guessing success rates in typical mask scenarios (see Table VII), our KN offers fast training and generation speeds without requiring GPU support (see Table IX), making it suitable for *on-site* training without the need to save/maintain model files. This property is quite desirable. Meanwhile, our Seq2Seq-based PassSeq framework supports a wider range of mask guessing scenarios (including PII- and reuse-based attacks) with strong generalization ability. Importantly, it performs best under *super-rare* template types (see Table VII), where statistical models generally struggle due to data sparsity.

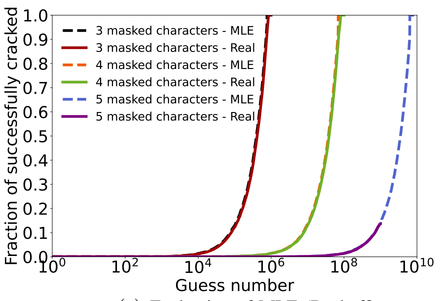
Recommended usage guidelines. Table IX shows that PassSeq outperforms KN and Backoff on templates with frequencies in the range of [1, 5], whereas its success rate is lower than KN when the template frequency increases to [10, 15]. To further examine this trend, we evaluate PassSeq, KN, and Backoff on templates with frequencies between [5, 10], where their guessing success rates are 93.93%, 98.02%, and 92.86%, respectively. Considering all these results alongside Fig. 5, we recommend using PassSeq for templates with frequencies $\leq$5 and for targeted scenarios involving PII or previously leaked



(a) Evaluation of MLE (Kneser-Ney)



(b) Evaluation of MLE (PassSeq)



(c) Evaluation of MLE (Backoff)

Fig. 9. Comparison between the actual password generation results (e.g., 3 masked characters-Real) and the MLE estimates (e.g., 3 masked characters-MLE) for Kneser-Ney, PassSeq, and Backoff [40] under three template types with different numbers of masks (n=3, 4, 5). Results show that the MLE estimates closely match the empirical generation outcomes, with the maximum deviation in guessing success rates $<1\%$.

Fig. 10. A further validation: mask guessing with keystroke acoustic information leaked from different types keyboards. (The experimental setup we used for keystroke collection is the same as [21], [70], [71]).

passwords, while KN/Backoff are better suited for higher-frequency templates or deployments that prioritize fast, GPU-free training and inference (see Fig. 8).

Overhead of MLE. Note that neural password models (e.g., our PassSeq) require a significant amount of time to generate a large number of guesses (e.g., $>10^7$) that *naturally conform to the given password templates*. For example, it takes PassSeq 18.5 hours to explicitly generate 10^7 guesses conforming *a single* given template with four or five masks. Fortunately, our MLE can reduce the time cost of guess number estimation (for any password under the given template) to minutes. Given a template containing three masks (e.g., `⋆w⋆rt⋆12`), there are a total of 94^3 unique passwords under this template. We employ MLE to estimate the guess number required to crack a target password (e.g., `qwErtY12`) under this template. The MLE parameters are set as follows: the number of samples N=10^5, and the number of repetitions t=4. The average time consumed by MLE to estimate the guess number for any password under this given template (i.e., `⋆w⋆rt⋆12`) is 107s (using PassSeq). Notably, the time consumption is mainly in the sampling and probability calculation process for a given template. Once this process is completed, the time for estimating the guess number of any 94^4 passwords under this password template is <0.1s.

Evaluation of MLE. To empirically evaluate the effectiveness of our proposed MLE estimator, we categorize password templates by the number of masks they contain, namely three, four, and five masks, which correspond to increasingly large guessing spaces (e.g., a four-mask template corresponds to a space of size 94^4). These three categories represent the most frequently occurring template types in typical mask guessing scenarios and are therefore selected as representative cases. For each category (a total of 30 templates are tested), we conduct random sampling with sample sizes of 10^4, 10^5, and 10^6, and repeat the sampling process four times, using 10,000 randomly selected passwords as the test set under the given template. In these settings, the MLE results exhibit stable convergence. We further applied PassSeq, Kneser-Ney, and Backoff [40] to perform both actual password generation and MLE-based estimation across the three template types. Considering that Kneser-Ney and Backoff generate guesses more efficiently than PassSeq, their actual generation sizes are larger than PassSeq (10^9 vs. 10^7). Fig. 9 shows that the empirical guessing success rates closely match the MLE estimates, with a maximum deviation of less than 1%, demonstrating the estimator's accuracy and robustness across different models and templates.

G. A further validation of PassSeq

To evaluate the practical effectiveness of PassSeq under realistic side-channel attack scenarios, we integrate it into keystroke acoustic inference attacks (see Fig. 10). Following prior work [21], [70], [71], we collect keystroke audio from 11 keyboard units across commonly used brands and models at realistic eavesdropping distances (17-30 cm), involving 16 diverse participants. Among them, nine distinct keyboards are used for same-device/same-user evaluations, each providing 100 keystroke audio samples recorded by individual participants entering 6-digit PINs sampled from the top-100 most frequent PINs in the Taobao dataset. To improve robustness and capture subtle acoustic variations, we additionally record 800 samples (top-800 PINs) on a 2020 MacBook Air entered by six different participants. Moreover, two identical keyboards of the same brand and model are typed by two users to enable realistic cross-user evaluations. Each same-device dataset is split into training and test sets with an 80:20 ratio, and each is used to train an independent acoustic inference model.

For acoustic inference, we start from the open-source CNN-based implementation of [21] and further enhance it following [2] by integrating CNNs, random forest classifiers, and MFCC-based feature extraction into a unified ensemble framework. This improves the single-key inference accuracy from about

TABLE X

GUESSING SUCCESS RATES OF THE ACOUSTIC, PASSSEQ, AND INTEGRATED MODELS ACROSS DIFFERENT KEYBOARDS[†].

Keyboard type / Cross-device Setting	Acoustic model [2], [21]			PassSeq model			Acoustic+PassSeq model v1[‡]			Acoustic+PassSeq model v2[‡]		
	Top-1	Top-10	Top-100	Top-1	Top-10	Top-100	Top-1	Top-10	Top-100	Top-1	Top-10	Top-100
HP EliteBook	30.00%	60.00%	80.00%	0.00%	0.00%	0.00%	45.00%	75.00%	85.00%	15.00%	55.00%	80.00%
Logitech K220	35.00%	50.00%	55.00%	0.00%	5.00%	20.00%	85.00%	90.00%	90.00%	35.00%	55.00%	85.00%
Apple Keyboard	40.00%	75.00%	95.00%	0.00%	0.00%	0.00%	40.00%	85.00%	90.00%	40.00%	75.00%	85.00%
Varmilo	75.00%	90.00%	95.00%	0.00%	0.00%	0.00%	55.00%	95.00%	95.00%	75.00%	95.00%	95.00%
Mech. KB	25.00%	30.00%	70.00%	0.00%	0.00%	0.00%	25.00%	55.00%	80.00%	30.00%	55.00%	85.00%
Dell KB216T	15.00%	25.00%	25.00%	10.00%	15.00%	25.00%	55.00%	65.00%	70.00%	15.00%	40.00%	60.00%
Dell KB212-B	15.00%	20.00%	25.00%	0.00%	10.00%	20.00%	25.00%	35.00%	45.00%	10.00%	30.00%	65.00%
Lenovo	15.00%	15.00%	60.00%	0.00%	0.00%	20.00%	30.00%	40.00%	45.00%	30.00%	55.00%	75.00%
MacBook Air 25	15.00%	25.00%	40.00%	0.00%	5.00%	25.00%	40.00%	55.00%	55.00%	20.00%	50.00%	65.00%
MacBook Air 20	18.75%	39.38%	53.12%	0.62%	2.50%	6.88%	28.75%	47.50%	60.00%	18.12%	28.75%	36.88%
Dell KB216T → Dell KB212-B	0.00%	0.00%	0.00%	1.00%	3.00%	24.00%	0.00%	0.00%	0.00%	1.00%	4.00%	24.00%
Dell KB212-B → Dell KB216T	0.00%	0.00%	0.00%	2.00%	9.00%	21.00%	0.00%	0.00%	0.00%	1.00%	2.00%	28.00%
MacBook Air 25 → MacBook Air 20	0.62%	0.62%	0.62%	0.25%	0.88%	3.14%	0.62%	0.62%	0.62%	0.25%	1.13%	3.39%
MacBook Air 20 → MacBook Air 25	0.00%	0.00%	0.00%	1.00%	4.00%	18.00%	0.00%	0.00%	0.00%	1.00%	4.04%	21.21%
Dell KB216T$_A$ → Dell KB216T$_B$	7.00%	10.00%	21.00%	0.00%	3.00%	17.00%	14.00%	21.00%	25.00%	7.00%	15.00%	17.00%
Dell KB216T$_B$ → Dell KB216T$_A$	7.00%	8.00%	15.00%	1.00%	3.00%	14.00%	9.00%	14.00%	17.00%	4.00%	6.00%	14.00%

[†] All percentages represent Top-k success rates under each model. Each within-device test sets $N = 20$ (except MacBook Air 20 with $N = 160$). Cells with dark gray and light gray backgrounds denote the highest and second-highest values (per keyboard and Top-k) across the four models, respectively.

[‡] v1 means using the acoustic model's predicted character probabilities to construct candidate password templates (e.g., `1*2**3`), corresponding to the oracle distribution. v2 directly adds the acoustic character probabilities to those produced by PassSeq, forming the no-oracle distribution.

40% to 94% under same-device in-domain splits. This ensemble model is adopted as our final acoustic backbone.

Besides same-device evaluations, we further conduct cross-device experiments to examine generalization. Specifically, we train and test on different keyboards from the same brand (and vice versa). We investigate two strategies for integrating PassSeq: (1) *Template-driven integration* (oracle distribution), where keystroke acoustic posteriors are used to construct candidate password templates (e.g., `1*2**3`) that guide PassSeq's generation; and (2) *Probability-fusion integration* (no-oracle distribution), where posterior probabilities from the acoustic and password models are directly combined using normalized weights. Based on empirical tuning, we assign higher weights (0.8-0.9) to the acoustic model under same-device settings, while under cross-device settings, a lower acoustic weight (0.1-0.2) achieves better results. Moreover, we have explored a complementary approach in cross-device settings, where acoustic posteriors are used to reorder PassSeq's beam-search outputs. Its effectiveness is comparable to that of character-level probability fusion, with neither approach consistently outperforming the other.

The overall template construction procedure is driven by per-keystroke acoustic posteriors. For each segmented keystroke, we extract three uncertainty-related statistics from the posterior distribution: (1) the top-1 probability, (2) the top-k candidates (k=3), and (3) the entropy to quantify prediction uncertainty. Based on these statistics, each position is either fixed to its most likely digit or masked, according to a set of entropy- and confidence-thresholding rules. Specifically, positions with high confidence (low entropy or high top-1 probability) are fixed, while uncertain positions are replaced by a mask token. To further enrich template diversity, we additionally generate single-position-masked variants from the top acoustic sequence prediction, as well as global Top-N masking templates that retain only the most reliable positions. All generated templates are then evaluated by a posterior-guided scoring function considering per-position confidence, consistency with the base acoustic prediction, and the proportion of fixed characters. The top-ranked templates (e.g., top-5 per acoustic candidate) are finally selected to guide the subsequent PassSeq-based password generation.

For PassSeq (implemented using a lightweight Seq2Seq architecture with two unidirectional LSTM layers), we use 6-digit numeric PINs from the Dodonew dataset as the training set. Guided by the constructed templates, the trained PassSeq generates ranked password guesses under both the template-driven and probability-fusion integration strategies.

Overall, integrating the acoustic model with the password model significantly improves guessing success rates across most keyboards (see Table X). More specifically, the oracle distribution integration (v1) achieves substantial gains: within 10 guesses, it generally outperforms the acoustic model by 5.6%-166.7%, and the PassSeq model by 3-18 times. The no-oracle distribution integration (v2) also outperforms the acoustic-only model in 13/16 cases, though to a lesser extent. Notably, in cross-device scenarios, the acoustic-only and oracle models perform poorly due to strong inter-device variability. While the v2 integration still provides measurable improvements, most of its gains stem from the password model itself. In contrast, in cross-user scenarios (two users typing on two identical keyboards, i.e., Dell KB216T$_A$ and Dell KB216T$_B$), the oracle integration significantly outperforms the acoustic baseline by 12.5%-110% within 10 guesses.

In this study, we primarily focus on 6-digit numeric PINs as a representative setting for validating the practical effectiveness of PassSeq. Extending our analysis to alphanumeric and symbol-rich passwords is left for future work.

V. PRACTICAL DEFENSES AGAINST MASK GUESSING

Employ fully static masking and minimize UI feedback granularity. At CCS'24, Hu et al. [30] found that dynamic masking, widely adopted on mobile browsers, provides little usability benefit compared with fully static masking. Together with our findings that partial character exposure (particularly the first character; see Fig. 5) significantly boosts attackers' guessing success rates, we recommend that mobile platforms also adopt fully static masking (already widely deployed on desktop systems [30]) and avoid any deterministic per-keystroke reveals or position-dependent hints. Such uniform masking effectively mitigates visual side-channel leakage without significantly sacrificing usability.

Blind password length across UI, transport, and application layers. Prior work by Harsha et al. [22] demonstrates that >84% of Alexa Top-100 websites are vulnerable to leak exact password length through request sizes even under TLS. Consistent with their findings, our evaluation indicates that leaking only the password length can increase an attacker's guessing success rate by 1.57 times (see Fig. 7). We therefore recommend end-to-end length blinding: using fixed-width input fields and placeholder elements on the client, disabling length-threshold hints, and padding both requests and error responses so that their observable sizes remain uniform.

Apply multi-dimensional adaptive rate limiting. Current throttling implementations remain inconsistent, effectively allowing thousands of online guesses per month (e.g., Amazon, which enforces the strictest lockout policy among the Alexa Top-10 websites, allows 3,600 monthly guessing attempts; see Table 7 in [59]). This tolerance suggests that our proposed password models remain effective across both online and offline scenarios, since our experimental configurations span guess budgets from 10^0-10^6, fully covering the scale of typical online limits. To counter such structured, mask-aware attacks, we recommend behavior-driven adaptive throttling: dynamically adjusting thresholds based on account history, device fingerprint, and the presence of mask-like guessing attempts (e.g., fixed-length sequences, position-constrained character classes, or leakage-conditioned templates).

Mitigate acoustic leakage and password-level inference. Our results show that the integration of per-keystroke acoustic leakage and password models amplifies the attacker's capability in two ways: keystroke sounds expose character-level cues, while password models capture structural and semantic characteristics. We thus provide two complementary mitigation directions. On the signal side, device- or application-level masking, such as mixing background sounds (which can be generated on the input device and does not impede users from entering passwords or performing tasks effectively [38]), has been shown to substantially reduce key-specific recognition cues. On the user side, adjusting typing behavior before entering sensitive information, and creating long, non-semantic passwords limits the gain attackers obtain from both acoustic features and password model-driven inference.

Practical applications for protection. By employing the Monte Carlo method [16], PassSeq can serve as a password strength meter (PSM), providing a security evaluation of passwords from an attacker's perspective. We evaluate PassSeq-PSM using the unsafe error metric (i.e., the rate at which a PSM incorrectly labels weak or easily guessable passwords as strong, a widely adopted measure in prior work [43], [67]). We compare PassSeq-PSM with FLA-PSM [43] and Markov-PSM [13] on 100,000 randomly sampled passwords from the 000Webhost, Wishbone, Taobao, and COMB datasets. Results (see Fig. 11 of the full version) shows that PassSeq-PSM has significantly fewer unsafe errors than both baselines, demonstrating improved reliability in password strength evaluation.

VI. Conclusion

We have presented the first systematic study of how attackers can substantially amplify password guessing effectiveness by exploiting *partial password information* (e.g., some characters and/or password length) combined with side-channel priors, PII, and sister passwords. Extensive experiments with 15 real-world datasets demonstrate the effectiveness and wide applicability of our proposed PassSeq and Kneser-Ney. Our results highlight the damaging threat of mask guessing, especially when combined with keystroke inference attacks or PIIs. We believe that the new algorithms and insights into mask guessing threats can help re-evaluate existing password practices and inform future password security research.

Acknowledgement

The authors are 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 Natural Science Foundation of Tianjin, China under Grant No. 25JCJQJC00230. See the full version of this paper at https://wangdingg.weebly.com/publications.html.

Ethical considerations

Our work is inherently dual-use. While it advances the understanding of password security under partial-information leakage (e.g., side-channel priors) and helps defenders quantify real-world risk, the same techniques could be misused to facilitate more efficient password guessing if deployed irresponsibly. We explicitly acknowledge this risk and adopt concrete mitigation measures to ensure that the defensive and societal benefits outweigh the potential for misuse.

Specifically, our study uses publicly available password datasets that were leaked through past breaches and have been widely used in prior password research (e.g., [40], [43], [50], [61], [68], [69]). However, besides passwords, our study incorporates associated PIIs (e.g., usernames or emails) in the context of targeted guessing scenarios. We recognize that affected individuals did not provide informed consent for research use of their data and that incorporating PII raises greater ethical concerns than password-only studies. While these datasets are already publicly accessible, their continued use still warrants careful ethical consideration.

To minimize potential harm to data subjects, we adopt strict data-handling safeguards. We do not redistribute any raw breached datasets, PII, or unprocessed side-channel traces, and our study does not introduce any additional exposure beyond what already exists in the public domain. All datasets are stored and processed on offline machines not connected to the Internet. Individual accounts are treated as strictly confidential, and all reported results are aggregated and anonymized, ensuring that no personally identifiable information or individual account details can be inferred. Sensitive attributes, including PII, are deleted immediately after the completion of analysis. All acoustic data are collected with explicit participant consent under controlled conditions.

Open science

We release the source code of Kneser-Ney, Backoff, and PassSeq in mask mode, together with full training/generation configurations (see https://github.com/CSSLabNKU). We also provide the implementation of our MLE rank estimator. For the side-channel integration experiments, we release a pretrained acoustic model, a pretrained password model, and 20 test audio samples collected from one keyboard, enabling reproduction one of our integrated-attack results in Table X.

References

[1] *Global Visual Hacking Experimental Study: Analysis*, Aug. 2016, https://multimedia.3m.com/mws/media/1254232O/global-visual-hacking-experiment-study-summary.pdf.

[2] *Keystroke Acoustic Recognition System*, April 2025, https://github.com/s2cr3t/Keystroke_Acoustic_Recognition_System.

[3] Y. Abdelrahman, M. Khamis, S. Schneegass, and F. Alt, "Stay cool! understanding thermal attacks on mobile-based user authentication," in *Proc. ACM CHI 2017*, pp. 3751–3763.

[4] N. Alotaibi, J. Williamson, and M. Khamis, "Thermosecure: Investigating the effectiveness of AI-driven thermal attacks on commonly used computer keyboards," *ACM Trans. on Priv. and Secur.*, vol. 26, no. 12, pp. 1–24, 2023.

[5] A. J. Aviv, J. T. Davin, F. Wolf, and R. Kuber, "Towards baselines for shoulder surfing on mobile authentication," in *Proc. ACSAC 2017*, pp. 486–498.

[6] A. J. Aviv, K. L. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. USENIX WOOT 2010*, pp. 1–7.

[7] J. Bai, B. Liu, and L. Song, "I know your keyboard input: A robust keystroke eavesdropper based-on acoustic signals," in *Proc. ACM MM 2021*, pp. 1239–1247.

[8] K. S. Balagani, M. Cardaioli, S. Cecconello, M. Conti, and G. Tsudik, "We can hear your PIN drop: An acoustic side-channel attack on ATM PIN pads," in *Proc. ESORICS 2022*, pp. 633–652.

[9] F. Binbeshr, M. L. M. Kiah, L. Y. Por, and A. A. Zaidan, "A systematic review of pin-entry methods resistant to shoulder-surfing attacks," *Comput. Secur.*, vol. 101, no. 102116, 2021.

[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] 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.

[12] M. Cardaioli, M. Conti, K. S. Balagani, and P. Gasti, "Your PIN sounds good! augmentation of PIN guessing strategies via audio leakage," in *Proc. ESORICS 2020*, pp. 720–735.

[13] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from markov models," in *Proc. NDSS 2012*, pp. 1–14.

[14] S. Cha, S. Kwag, H. Kim, and J. H. Huh, "Boosting the guessing attack performance on android lock patterns with smudge attacks," in *AsiaCCS*. ACM, 2017, pp. 313–326.

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

[16] M. Dell'Amico and M. Filippone, "Monte carlo strength evaluation: Fast and reliable password checking," in *Proc. ACM CCS 2015*, pp. 158–169.

[17] Q. Dong, C. Jia, F. Duan, and D. Wang, "RLS-PSM: A robust and accurate password strength meter based on reuse, leet and separation," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4988–5002, 2021.

[18] M. Dürmuth, F. Angelstorf, C. Castelluccia, D. Perito, and C. Abdelberi, "OMEN: faster password guessing using an ordered markov enumerator," in *Proc. ESSoS 2015*, pp. 119–132.

[19] M. Dürmuth, D. Freeman, S. Jain, B. Biggio, and G. Giacinto, "Who are you? A statistical approach to measuring user authenticity," in *Proc. NDSS 2016*, pp. 1–15.

[20] M. Eiband, M. Khamis, E. von Zezschwitz, H. Hussmann, and F. Alt, "Understanding shoulder surfing in the wild: Stories from users and observers," in *Proc. CHI 2017*, pp. 4254–4265.

[21] J. Harrison, E. Toreini, and M. Mehrnezhad, "A practical deep learning-based acoustic side channel attack on keyboards," in *Proc. EuroS&P Workshops*, 2023, pp. 270–280.

[22] B. Harsha, R. Morton, J. Blocki, J. A. Springer, and M. Dark, "Bicycle attacks considered harmful: Quantifying the damage of widespread password length leakage," *Comput. Secur.*, vol. 100, p. 102068, 2021.

[23] Hashcat Project, "Hashcat," https://hashcat.net/hashcat/.

[24] K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn, "Scalable modified kneser-ney language model estimation," in *Proc. ACL 2013*, 2013, pp. 690–696.

[25] C. Herley and P. Van Oorschot, "A research agenda acknowledging the persistence of passwords," *IEEE Secur. Priv.*, vol. 10, pp. 28–36, 2012.

[26] B. Hitaj, P. Gasti, G. Ateniese, and F. Pérez-Cruz, "PassGAN: A deep learning approach for password guessing," in *Proc. ACNS 2019*.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] B. Honan, *Visual Data Security White Paper*, July 2012, https://multimedia.3m.com/mws/media/950026O/secure-white-paper.pdf.

[29] S. Houshmand, S. Aggarwal, and R. Flood, "Next gen PCFG password cracking," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 8, pp. 1776–1791, 2015.

[30] Y. Hu, S. Alroomi, S. Sahin, and F. Li, "Unmasking the security and usability of password masking," in *Proc. CCS 2024*, pp. 4241–4255.

[31] Z. Huang, D. Wang, and Y. Zou, "Prob-hashcat: Accelerating probabilistic password guessing with hashcat by hundreds of times," in *Proc. RAID 2024*, pp. 674–692.

[32] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *Proc. ACM CCS 2013*, pp. 145–160.

[33] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 35, no. 3, pp. 400–401, 1987.

[34] 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.

[35] R. Kneser and H. Ney, "Improved backing-off for M-gram language modeling," in *Proc. IEEE ICASSP 1995*, pp. 181–184.

[36] T. Kwon, S. Shin, and S. Na, "Covert attentional shoulder surfing: Human adversaries are more powerful than expected," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 44, no. 6, pp. 716–727, 2014.

[37] E. Liu, A. Nakanishi, M. Golla, D. Cash, and B. Ur, "Reasoning analytically about password-cracking software," in *Proc. IEEE S&P 2019*, pp. 380–397.

[38] A. Lobanova, J. He, N. Zhu, A. Nazir, and X. Ma, "Machine-learning-based adaptive keyboard sound masking against acoustic channel attacks," in *Proc. CCNS 2024*, pp. 271–277.

[39] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. ACL EMNLP 2015*, pp. 1412–1421.

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

[41] R. Manning, *2025 Global State of Authentication survey: A world of difference in cybersecurity habits*, Sept. 2025, https://www.yubico.com/blog/2025-global-state-of-authentication-survey-a-world-of-difference-in-cybersecurity-habits/.

[42] D. Marques, I. Muslukhov, T. J. Guerreiro, L. Carriço, and K. Beznosov, "Snooping on mobile phones: Prevalence and trends," in *Proc. SOUPS 2016*, pp. 159–174.

[43] W. Melicher, B. Ur, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean and accurate: Modeling password guessability using neural networks," in *Proc. USENIX SEC 2017*, pp. 175–191.

[44] R. Morris and K. Thompson, "Password security: A case history," *Commun. ACM*, vol. 22, no. 11, pp. 594–597, 1979.
[45] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. ACM CCS 2005*, pp. 364–372.
[46] Openwall Project, "John the ripper password cracker," http://www.openwall.com/john/.
[47] 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.
[48] D. Pasquini, G. Ateniese, and M. Bernaschi, "Interpretable probabilistic password strength meters via deep learning," in *Proc. ESORICS 2020.*
[49] 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. 36–54.
[50] 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.
[51] J. Rando, F. Pérez-Cruz, and B. Hitaj, "Passgpt: Password modeling and (guided) generation with large language models," in *Proc. ESORICS 2023*, pp. 164–183.
[52] E. Shareghi, M. Petri, G. Haffari, and T. Cohn, "Fast, small and exact: Infinite-order language modelling with compressed suffix trees," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 477–490, 2016.
[53] X. Su, X. Zhu, Y. Li, Y. Li, C. Chen, and P. J. E. Veríssimo, "Pagpassgpt: Pattern guided password guessing via generative pretrained transformer," in *Proc. DSN 2024*, pp. 429–442.
[54] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS 2014*, pp. 3104–3112.
[55] C. Team, *New Study Underscores Slow Adoption of Multifactor Authentication By Global SMBs*, Nov. 2024, https://cyberreadinessinstitute.org/news-and-events/new-study-underscores-slow-adoption-of-multifactor-authenification/.
[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS 2017*, pp. 5998–6008.
[57] R. Veras, C. Collins, and J. Thorpe, "On semantic patterns of passwords and their security impact," in *Proc. NDSS 2014*.
[58] C. Wang, J. Zhang, M. Xu, H. Zhang, and W. Han, "#segments: A dominant factor of password security to resist against data-driven guessing," *Comput. Secur.*, vol. 121, no. 102848, 2022.
[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, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: Understanding passwords of Chinese web users," in *Proc. USENIX SEC 2019*, pp. 1537–1555.
[61] 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.
[62] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *Proc. IEEE S&P 2022*, pp. 489–506.
[63] D. Wang, Y. Zou, Z. Zhang, and K. Xiu, "Password guessing using random forest," in *Proc. USENIX SEC 2023*, pp. 965–982.
[64] 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.
[65] D. Wheeler, "zxcvbn: Low-budget password strength estimation," in *Proc. USENIX SEC 2016*, pp. 157–173.
[66] K. Xiu and D. Wang, "Pointerguess: Targeted password guessing model using pointer mechanism," in *Proc. USENIX SEC 2024*, pp. 5555–5572.
[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] T. Yang and D. Wang, "Rankguess: Password guessing using adversarial ranking," in *Proc. S&P 2025*, pp. 682–700.
[70] J. Yu, L. Lu, Y. Chen, Y. Zhu, and L. Kong, "An indirect eavesdropping attack of keystrokes on touch screen through acoustic sensing," *IEEE Trans. Mob. Comput.*, vol. 20, no. 2, pp. 337–351, 2021.
[71] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *Proc. ACM CCS 2014*.
[72] V. Zimmermann, "From the quest to replace passwords towards supporting secure and usable password creation," Ph.D. dissertation, Technical University of Darmstadt, Germany, 2021.
[73] Y. Zou, M. An, and D. Wang, "Password guessing using large language models," in *Proc. USENIX SEC 2025*, pp. 7799–7818.

Appendix

A. Details of our maximum likelihood estimation

Let pw* be a password that conforms to the template PT*. Denote n as the number of unique passwords in the set Γ_{PT^*} that conform to PT*, and m as the number of passwords in Γ_{PT^*} that also satisfy $p(\text{pw}){>}p(\text{pw}^*)$. We want to estimate m. Let $p{=}m/n$, which is the probability that a uniformly sampled password from Γ_{PT^*} satisfies the condition $p(\text{pw}){>}p(\text{pw}^*)$.

Define a sampling experiment $\mathcal{S}$: Uniformly sample one password pw from Γ_{PT^*}. If $p(\text{pw}){>}p(\text{pw}^*)$, record a success. Repeat experiment $\mathcal{S}$ for N times. Let the number of successes be x, which follows a binomial distribution:

$$x \sim \text{Binomial}(N,p), \quad f(x|N,p) = \binom{N}{x} p^x (1-p)^{N-x}.$$

Define experiment $\mathcal{T}$ as repeating $\mathcal{S}$ N times. Repeat $\mathcal{T}$ for t times, and denote the sequence of success counts as $\{x_1, x_2, ..., x_t\}$. The likelihood function is:

$$\begin{aligned} L(p) &= \log\left(\prod_{i=1}^{t} \binom{N}{x_i} p^{x_i} (1-p)^{N-x_i}\right) \\ &= \sum_{i=1}^{t} \log\binom{N}{x_i} + \left(\sum_{i=1}^{t} x_i\right)\log(p) + \left(tN - \sum_{i=1}^{t} x_i\right)\log(1-p). \end{aligned} \tag{11}$$

Taking the derivative of $L(p)$ with respect to p gives:

$$\frac{\partial L(p)}{\partial p} = \frac{\sum_{i=1}^{t} x_i}{p} - \frac{tN - \sum_{i=1}^{t} x_i}{1-p}.$$

Setting $\frac{\partial L(p)}{\partial p} = 0$ and solving yields the MLE:

$$\tilde{p} = \frac{1}{Nt}\sum_{i=1}^{t} x_i.$$

Unbiasedness: The expectation of $\tilde{p}$ is

$$E[\tilde{p}] = \frac{1}{Nt}\sum_{i=1}^{t} E[x_i] = \frac{tpN}{Nt} = p.$$

Variance: The second derivative of the log-likelihood is:

$$\frac{\partial^2 L(p)}{\partial p^2} = -\frac{\sum_{i=1}^{t} x_i}{p^2} - \frac{tN - \sum_{i=1}^{t} x_i}{(1-p)^2}.$$

The Fisher Information is the negative expectation:

$$\mathcal{I}(p) = -E\left[\frac{\partial^2 L(p)}{\partial p^2}\right] = \frac{tN}{p(1-p)}.$$

The Cramér-Rao bound (inverse Fisher Information) gives:

$$\text{Var}(\tilde{p}) = \frac{p(1-p)}{tN} \le \frac{1}{4tN}.$$

Conclusion: The estimator $\tilde{p}$ is unbiased and converges to p in probability. The estimate of m is: $\tilde{m} = \tilde{p} \cdot n$.