# Making the Memorable Unguessable: PII-Enhanced Password and Honeyword Generation for Usable Security

Hannah Ammah Serwaah Adjei and Miguel Vargas Martin

Ontario Tech University, Oshawa, Canada
hannah.adjei@ontariotechu.net, miguel.martin@ontariotechu.ca

**Abstract.** We treat personally identifiable information (PII) as a *controlled* security primitive for generating passwords and honeywords that remain memorable yet resist targeted guessing. Our generator composes three random 4-grams from a user's PII string (12 PII-derived characters) and inserts one random digit and one random special character at random positions, deliberately breaking word boundaries (non-semantic) to preserve usability while frustrating semantics-driven guessing. We calibrate an attacker using a compact LLM-based segmenter benchmarked against a hashtag segmentation system, and then mimic RankGuess under a TarGuess-I-style targeted threat model: a neural guesser enumerates candidates and a learned ranker scores them. Against a PCFG-PII (TarGuess-I) baseline, our joint model yields higher early-budget success. We assess strength and usability by computing `zxcvbn` scores for all sweetwords, using cosine similarity to verify honeyword closeness, and running a three-session recall study with a post-study System Usability Scale (SUS) survey ($n$=60), showing higher memorability for PII-based passwords. Overall, we present a practical, user-centric scheme where non-semantic, PII-derived passwords and honeywords withstand targeted guessing under the attacker capabilities quantified in our calibration while maintaining usability.

**Keywords:** PII · Password · Security · User Studies · Attacks · Honeywords.

## 1 Introduction

User authentication is a fundamental pillar of cybersecurity, acting as the initial barrier against unauthorized access to sensitive data and resources. Traditional authentication methods [26], such as passwords [35,4] and PINs [1], are vulnerable to brute-force [22], phishing [20], and password reuse. They also impose cognitive burdens on users, who must remember many secrets across services. As attacks grow more sophisticated [19] and password-related breaches remain common, the need to balance security and usability has become critical. Users often choose more complex passwords for high-risk accounts [31], yet traditional policies still struggle to scale against modern threats [9].

In response, some systems explicitly try to keep personally identifiable information (PII) out of passwords, using context-specific blocklists (e.g., service names, usernames) and pattern-aware strength meters that flag names and dates [2,27,32]. PII in passwords is viewed with suspicion because it can help targeted guessers and raises privacy concerns. Recent work on PII leakage from large language models (LLMs) [17] and automated PII extraction [16] further underlines the need to handle such data carefully.

Against this backdrop, we ask whether PII can be used as a *controlled* security primitive to improve memorability without giving up resistance to targeted guessing. We draw on prior evidence that personal information shapes human-chosen passwords [15] and on work showing that advanced AI systems can sometimes distinguish real passwords from honeywords [6], highlighting both risk and opportunity. Our approach uses PII to generate non-semantic passwords and honeywords [28,30,33], aiming to reduce cognitive burden while maintaining security. Rather than dismiss PII-derived passwords outright, we explore how carefully designed, non-semantic PII-based schemes can bridge the gap between usability [3] and robust defenses against targeted guessing.

### 1.1 Related Work

*Honeywords and sweetlists.* Juels and Rivest introduced *honeywords*: storing multiple candidates per account and using a separate *honeychecker* to detect decoy logins [13]. We adopt this standard honeyword architecture: the server stores multiple candidates per account and a separate honeychecker flags logins that hit decoys [13,37,29]. Follow-on work studies how to generate convincing decoys, how attackers adapt, and how detection interacts with system design [5,29,23]. Recent methods use pre-trained language models to craft *semantically close* decoys [37]. Our PII-derived, non-semantic passwords with honeywords drawn from the same process contribute to this line by examining how hard it is for attackers to distinguish the real secret from decoys under small online budgets.

*LLMs for password generation and guessing.* Large language models can synthesize password-like strings and act as interactive guessers or rankers. Prior work has used LLMs to generate candidate passwords conditioned on PII or seeds and to score or rank them [6,37,38], often exploiting semantic or format cues. Neural trawling guessers such as GNPassGAN [36] improve generic offline guessing but do not condition on per-user PII or interact with honeywords. Our work instead pairs a controlled, non-semantic, PII-aware generator with a RankGuess-style attacker and analyzes early-budget success against this fixed defender distribution.

*PII, targeted guessing, and defender goals.* Personal information is known to improve targeted guessing [28], which motivates designs that keep memorability benefits while limiting predictable structure. Prior work shows friction with conventional policies and supports lowering cognitive cost without weakening defenses [12]. Our scheme pursues this balance: PII-informed, non-semantic generation for memorability, paired with empirical tests of how hard it is for informed attackers, classical and LLM-based, to crack these secrets under targeted, small-budget conditions.

*Security–usability measurements.* Empirical studies highlight creation habits and the importance of feedback grounded in real risk [25]; memory aids can help with system-assigned secrets [8]. Strength meters that model user behavior, such as fuzzy Password Strength Meter (fuzzyPSM), aim to give more accurate signals during creation [26], and pattern-aware meters and context blocklists help avoid obvious PII leakage and service-specific cues [27,32,2]. These tools act as guardrails (strength, indistinguishability, usability); our work complements them by evaluating a PII-based, non-semantic generation scheme under a calibrated targeted attacker and standard usability metrics.

*How this study connects the threads.* Prior work advanced both decoy generation and targeted guessing; recent work showed that LLM-based attackers can leverage PII and semantics to improve small-budget guessing [6,37,38]. We connect these threads by asking: when the defender uses a non-semantic, PII-informed generator, how much advantage does a RankGuess-style attacker gain under realistic online budgets? This lets us study security and memorability jointly, while keeping both the generator and attacker model transparent and reproducible.

*Positioning w.r.t. RankGuess, LLM guessers, and honeywords.* RankGuess [34] is a general learn-to-rank framework spanning offline, online, targeted, and untargeted settings with flexible enumeration and rewards. Our scope is narrower: *online, targeted* guessing under small per-account budgets against a fixed, PII-aware but *non-semantic* defender distribution. We keep enumeration simple, report coverage explicitly, and tune a lightweight joint score (Language Model (LM) + ranker with a length penalty) for small-budget Hits@$B$/MRR i.e., within our setting, can a strong scorer reach high early-budget success under controlled generation and explicit coverage?

PassLLM [38] studies end-to-end LLM guessers that emit passwords directly from PII and achieve strong small-budget results. We take a different angle: we *control* the defender's generator (non-semantic 4-grams + insertions) and use a RankGuess-style attacker to probe its robustness. This enables clean ablations (e.g., 12→16 chars; 20→40 honeywords), keeps costs low, and is easy to reproduce; incorporating stronger end-to-end LLM attackers is future work.

For honeywords, the classic "flatness" goal [11] is that decoys look as likely as the real secret; HoneyGen [10] pursues this via representation learning and evaluates with $\epsilon$-flatness and related graphs. We share the objective but follow a non-semantic path: both real and decoys are drawn from the same chunk-and-insert process, which naturally raises $n$-gram overlap and edit-distance similarity while avoiding semantic leakage, yielding "already-flat" lists without relying on population leaks or external embeddings.

### 1.2 Research Questions and Contributions

This work aims to address three research questions:

1. **RQ1:** How does our PII-derived password scheme compare to state-of-the-art targeted guessing?
2. **RQ2:** Do PII-derived passwords improve memorability/usability (recall, SUS) over random passwords *without* lowering measured strength (`zxcvbn`)?
3. **RQ3:** Are PII-derived honeywords indistinguishable from the real password to an informed attacker?

To answer the RQ1, we calibrated an attacker by matching Maddela et al.'s segmenter [18], then evaluated it in a RankGuess setting (LM guesser, learned ranker, and a post–reinforcement-learning (post-RL) joint scorer) against a PCFG-PII baseline (TarGuess-I). To answer RQ2, we ran a user study ($n$=60) comparing PII-derived passwords to purely random ones, and computed `zxcvbn` for all sweetwords (honeywords + passwords). We observed higher memorability/usability for PII passwords without lowering measured strength (`zxcvbn` remained high across sets). For RQ3 we measured cosine similarity between each real password and its 19 honeywords. Additionally, we allowed informed scorers (PCFG-PII and our LM/ranker) to attempt to identify the real password from its 19 honeywords. Under this setup, they were only marginally better than guessing at random and could not reliably identify the true password.

In summary, the main contributions and findings for this work include:

- **PII-driven password and honeyword generation.** We leverage personally identifiable information (PII) and generative language models to produce both passwords and honeywords. Our PII-based passwords score well on `zxcvbn`, and the resulting honeywords stay hard to single out, as checked by cosine similarity and discriminator-style tests.
- **Attacker calibration and targeted-guessing benchmark.** We calibrate an attacker by benchmarking a compact LLM segmenter against the system of Maddela et al. [18], then *mimic RankGuess* in a TarGuess-I setting with an LM guesser, learned ranker, and RL-tuned joint model. Against a PCFG-PII baseline, we see better early-budget resilience, while large-budget success is ultimately bounded by enumeration, supporting the robustness of our PII-based generator.
- **Usability and framework integration.** A user study (recall + SUS; $n$=60) shows improved memorability and usability over random passwords without reducing measured strength. We outline safeguards for responsible PII use and show how our PII-based passwords and honeywords fit into a practical authentication workflow.

## 2 Preliminaries and Threat Model

### 2.1 Notation and Data Model

Let $\mathcal{U}$ be the set of users and $\mathcal{C}$ the password space. For $u \in \mathcal{U}$, the unknown password is $p_u \in \mathcal{C}$. The attacker observes user-specific side information $q \in \mathcal{Q}$ consisting only of PII tags/values. A scoring function $f : \mathcal{Q} \times \mathcal{C} \to \mathbb{R}$ induces an order $L(q) = [p_{(1)}, p_{(2)}, \ldots]$ in descending likelihood. The *guess number* $\mathrm{gn}(p; q)$ is the rank position of $p$ in $L(q)$. We encode PII as typed tags

$$\mathcal{T} = \{\mathrm{YR}, \mathrm{CO}, \mathrm{CT}, \mathrm{RG}, \mathrm{LG}, \mathrm{NT}, \mathrm{HB}, \mathrm{MU}, \mathrm{FN}\},$$

for year_range, company, city, region, language, nationality, hobby, music, favorite_number. Each $t \in \mathcal{T}$ has an inventory $\mathcal{V}_t$. Strings (PII tokens and candidate passwords) are represented by overlapping 4-gram chunks $\chi_4(\cdot)$.

*Non-personal running example:* Define

$$q = (\mathrm{YR} = 2030,\ \mathrm{CO} = \texttt{AcmeCorp},\ \mathrm{CT} = \texttt{Calgary},\ \mathrm{FN} = 7).$$

A candidate such as `Calg2030Corp!` may be generated, where

$$\chi_4(\texttt{Calgary}) \mapsto \texttt{Calg}, \qquad \chi_4(\texttt{AcmeCorp}) \mapsto \texttt{Corp}.$$

### 2.2 Attacker Model (Targeted PII-Aware Online Guessing)

We adopt Kerckhoffs' principle (the mechanism is public; secrecy resides only in the secret).

*Capability (S1: PII side information):* The adversary may obtain *heterogeneous* PII $(t, v) \in \mathcal{T} \times \mathcal{V}_t$ (e.g., via OSINT/breaches) and use it to condition guessing.

*Adversary pre-processing assumption:* We assume the adversary can perform semantic segmentation of concatenated strings as a pre-processing step (see Sec. 3.10). Our generator remains non-semantic by design: three random 4-grams (12 chars), then one random digit and one random special character inserted at random positions.

*Objective under an online budget:* Let $B$ be the server-enforced per-account guess budget. The expected success rate (ESR) over a population $\mathcal{D}$ is

$$\mathrm{ESR}(B) = \frac{1}{N} \left|\{\, u \in \mathcal{U}_{\text{test}} \ :\ \mathrm{gn}(p_u; q_u) \le B \,\}\right|, \tag{1}$$

where $N = |\mathcal{U}_{\text{test}}|$ is the number of evaluated users and $q_u$ is the side information (PII) for user $u$.
Equivalently, the queries-to-hit at confidence $\alpha \in (0, 1)$ is

$$\mathrm{Q2H}_\alpha \ = \ \min\{\, k \in \mathbb{N} :\ \mathrm{ESR}(k) \ge \alpha \,\}. \tag{2}$$

### 2.3 Threat Scenario and Formalization (S1)

Given PII $q$, the attacker enumerates a candidate set $\mathrm{Cand}(u)$ by instantiating tag-aware structures and ranks candidates with $f(q, \cdot)$.

### 2.4 Attacking Methods Considered

**PCFG-PII Baseline (TarGuess-I style):** We use nonterminals

$$\mathcal{N} = \{L, D, S\} \cup \{\mathrm{YR}, \mathrm{CO}, \mathrm{CT}, \mathrm{RG}, \mathrm{LG}, \mathrm{NT}, \mathrm{HB}, \mathrm{MU}, \mathrm{FN}\}.$$

A structure $z \in \mathcal{Z}$ expands via productions with probabilities $\theta$; terminals come from tag inventories $\mathcal{V}_t$ or character $n$-grams (here 4-grams). The candidate likelihood is

$$P_\theta(p \mid q) \;=\; \sum_{z \in \mathcal{Z}} P_\theta(z \mid q) \; \prod_j P_\theta(x_j \mid z, q), \tag{3}$$

and the ranker is $f_{\mathrm{PCFG}}(q, p) = \log P_\theta(p \mid q)$.

**Rank-Style Guesser and Ranker (Our Model):** Let $E_{\mathrm{tag}}$, $E_{\mathrm{val}}$ be tag/value embeddings and $E_{4g}$ the 4-gram embedding. PII $q$ is encoded as $\{(t, v)\}$ and pooled into $h_0$; a Gated Recurrent Unit (GRU) encodes the candidate $p$ to $h_T$. We combine the LM (guesser) and ranker signals in a joint score:

$$\mathrm{JScore}(s \mid u) \;=\; \mu \log p_\theta(s \mid u) \;+\; (1 - \mu)\, R_\phi(s, u) \;-\; \gamma\, |s|, \tag{4}$$

*Definitions.* $R_\phi(s, u)$ is the learned *ranker score*: a scalar from a model with parameters $\phi$ that takes candidate $s$ and user/context $u$. The ranker is trained with the listwise objective in Eq. 7 on per-user candidate lists (see Sec. 3.8). Here, $\mu \in [0, 1]$ (we use $\mu$=0.6) and length penalty $\gamma \geq 0$.

*Policy-gradient update on the guesser.* The guesser policy $\pi_\theta$ is optimized to prefer strings valued by (4):

$$J(\theta) \;=\; \mathrm{E}_{s \sim \pi_\theta(\cdot \mid u)}\big[\mathrm{JScore}(s \mid u)\big], \tag{5}$$

$$\nabla_\theta J \;=\; \mathrm{E}\big[\nabla_\theta \log \pi_\theta(s \mid u)\, \big(\mathrm{JScore}(s \mid u) - b\big)\big] \;+\; \lambda\, \nabla_\theta H(\pi_\theta), \tag{6}$$

where $b$ is a baseline (variance reduction) and $H$ an entropy bonus with coefficient $\lambda \geq 0$.

*Listwise learning-to-rank for the ranker:* Given a list $\{s_1, \ldots, s_K\}$ and a target distribution $p_{\mathrm{adv}}(\cdot \mid u)$,

$$\min_\phi \; \mathrm{KL}\big(\, p_{\mathrm{adv}}(s \mid u) \;\big\|\; \mathrm{softmax}\big(R_\phi(s_1, u), \ldots, R_\phi(s_K, u)\big)\big)\,. \tag{7}$$

where KL denotes the Kullback–Leibler divergence.

### 2.5 Evaluation Metrics (Online Guessing)

Coverage is enumerator-limited:

$$\text{Coverage} = \frac{1}{N}\left|\{\, u : \; p_u \in \text{Cand}(u) \,\}\right|. \tag{8}$$

Hits@B (our plots/tables) equals success@$k$ at $k{=}B$:

$$\text{Hits@}B \;=\; \Pr\left[\text{gn}(p_u; q_u) \le B\right] \;=\; \text{ESR}(B). \tag{9}$$

**Mean Reciprocal Rank (MRR).** Define $\text{rank}_u = \text{gn}(p_u; q_u)$ if $p_u \in \text{Cand}(u)$ and $+\infty$ otherwise.

$$\text{MRR} = \frac{1}{N}\sum_{u=1}^{N}\frac{1}{\text{rank}_u}. \tag{10}$$

We also report AUAC (Area Under the attacker's Hits@B Curve), defined as the discrete average of Hits@$B$ up to the policy budget $B_{\max}$:

$$\text{AUAC}(B_{\max}) = \frac{1}{B_{\max}}\sum_{k=1}^{B_{\max}}\text{Hits@}k. \tag{11}$$

### 2.6 Defender Model and Scope

We target the practical lockout/throttling regime: a service enforces a small per-account guess budget $B$ with standard password checks. Multi-Factor Authentication (MFA) or WebAuthn are not assumed. Our scope is intentionally narrow: *online, targeted* guessing conditioned on user PII. The following are out of scope: sister-password/reuse, pivoted/partial-leak attacks, and offline hash compromise.

**Generator (fixed, non-semantic).** Defender passwords are built from fixed 4-gram (non-semantic) chunks with one random digit and one random special character inserted at random positions. The main setting uses three 4-grams (12 chars); a 4×4 (16-char) variant is evaluated as an ablation (Sec. 3.15). Per-user draws are secret; the mechanism is public (Kerckhoffs).

**Threat scope and evaluation choices.** We report *coverage* explicitly: an enumerator first determines whether the true password appears in a user's candidate set; Hits@$B$ and MRR then measure how well scorers rank those candidates. When needed, we run *forced-coverage* to isolate ranking quality from enumeration misses. Scoring uses a lightweight joint score mixing a PII-conditioned LM and a learned ranker (mixture $\mu$; simple length penalty), with optional policy-gradient tuning that does *not* change enumeration (coverage fixed). For honeywords, decoys are drawn from the same non-semantic process as the real secret; indistinguishability is tested with Top-1, Area Under the ROC Curve (AUC), and Mean Reciprocal Rank (MRR) on a 1+$H$ selection task and compared to chance. We also include a non-LLM control for decoy generation to show that flatness does not rely on the LLM.

# 3 Methodology

## 3.1 System Overview

Our goal is to generate passwords and honeywords from user-provided, consented PII and evaluate them for security and usability under a credible, targeted attacker.

*Quick terminology.* The *Enumerator* builds the candidate list for a user; whether it includes the true password determines *Coverage* (Eq. 8). The *Guesser* is a language model that scores strings $\log p_\theta(s \mid u)$, and the *Ranker* is a learned scorer $R_\phi(s, u)$. The *Joint* model combines both, with a small length penalty (Eq. 4). Our PCFG-PII baseline (TarGuess-I style) ranks by $\log P_\theta(p \mid q)$ using PII-conditioned grammar expansions. Online effectiveness is reported as *Hits@B* (success within $B$ guesses; Eq. 9) and *MRR* (Eq. 10); for large $B$, behavior is ultimately limited by coverage. For strength we report `zxcvbn` scores [32], and indistinguishability is assessed via Top-1, AUC, and MRR on the combined set of real passwords and honeywords.

## 3.2 Datasets

We use $N$=1000 users; each yields a 20-item sweetword list (1 real + 19 honeywords). We split *by user* 80/20: 800 for training, 200 for testing (no cross-user leakage). The targeted-guessing enumerator uses a per-slot cap of 30 and a global cap of 50,000; under this policy, test coverage is 70/200 = 35%. For calibration we use the public *HashtagMaster* benchmark; for both the RankGuess-style model and the TarGuess-I (PCFG-PII) baseline, we train/estimate on our *own* sweetword datasets (canonical PII strings and generated password data), not on external leaks.

*Study participants and evaluation data.* For the usability study, we recruited and consented $n_{\text{consented}} = 150$ participants; $n_{\text{completed}} = 60$ finished all recall stages and SUS. For targeted guessing and honeyword tests, we use 1,000 user records (20 sweetwords each): 150 records derived from consented participants' PII and 850 synthetic records constructed under the *same* non-semantic 4-gram rules to reach scale. All strings meet the same length/character-class constraints and strength screening. Participant data are de-identified and stored under approved ethics; no raw PII is released, and all security analyses are reported in aggregate.

## 3.3 Data Preparation and PII Password Generation

We begin with consented, survey-provided PII (e.g., countries, dates, places, organizations, languages, hobbies) and build a consolidated string per participant. That string is segmented into consistent, fixed-length chunks using an LLM–assisted procedure. From these chunks, our *PII password generation method* produces one account password and nineteen honeywords, yielding a 20-item *sweetword* list (1 real + 19 decoys). We assume an attacker who knows our methodology, *i.e.*, under Kerckhoffs' principle the adversary knows the generator design (3×4-grams + one digit + one special character), the enumerator and its caps (30 per slot, 50,000 global), the joint scoring rule (including $\mu$), and can perform semantic segmentation on the user's PII $q$; only the per-user random draws (which chunks, which digit/special character, and their positions) and the final password remain secret, and online attempts are bounded by the budget $B$.

For each account we generate a 20-item sweetword set using the same non-semantic process. *Before* user choice, all 20 strings are symmetric candidates none is designated "the real" by the system. The user selects one as their password; *after* selection, the chosen string becomes the verifier and the remaining 19 are stored as honeywords.

All sweetwords are screened with `zxcvbn` [32]. We first tune the generation policy until candidates consistently meet the highest strength tier, then freeze the policy for all subsequent creation. For context, we also prepare two baselines under the same pipeline: (i) *Random-PII (RaPII)*, formed by randomly recombining PII-like chunks from a publicly available dataset (4iQ dataset); and (ii) *Pure Random (RaPass)*, generated using a true-random source (random.org). Both baselines are evaluated with `zxcvbn` in the same way. In short, this yields per-user sweetword lists with uniform chunking and strength checks across PII, RaPII, and RaPass, and sets up the downstream analyses on targeted guessing, honeyword indistinguishability, and usability.

### 3.4 Design Rationale (4-grams, length, honeyword count)

*Non-semantic 4-grams.* We slice consented PII into fixed 4-grams and sample 20 candidates; the item the user selects becomes the secret and the rest are honeywords. Fixed windows break word boundaries (suppressing semantic cues), keep short user-anchored substrings that aid recall, and keep the generator auditable and independent of external corpora or leaks, yielding a deliberately *non-semantic* surface.

*Length: 14 main; 16 as policy-aligned robustness.* The main setting uses 12-character PII-derived core (3×4 PII-derived characters + digit + special) to balance memorability with small-budget online security. To align with guidance recommending $\geq$ 15 characters for single-factor passwords [24], we also test a 16-char variant (4×4). Under forced coverage (Cov=1.0), the Joint model's early-budget success at 16 chars is near zero (Fig. 1b), indicating that longer non-semantic strings harden small-budget attacks in our setting.

*Honeywords: $H$=20 default; $H$=40 variant.* We use $H$=20 as a practical default that balances detection, storage, and user support. We also evaluate $H$=40 on 16-char lists; strong scorers remain near chance (Table 3), indicating that indistinguishability persists as $H$ grows.

*Security scope, operations, and reproducibility.* We scope claims to online attacks under small, service-enforced per-account budgets. Let $L$ be the site's cap; we report Hits@$B$ only for $B \leq L$ and make no claims for unrestricted or offline guessing. Prompts and policies are fixed across experiments, and code will be released upon publication. Temporarily withholding non-essential templates during review does not affect the specified mechanism (non-semantic 4-grams + one digit + one special; insertion rules; honeyword counts) or the reported results; redacted prompts and model configurations appear in Appendices C and D, with anonymized examples in Appendix E.

*Risk of misremembering a honeyword (and mitigations).* Enrollment presents a 20-item sweetlist; the chosen item is the secret and the remaining 19 are honeywords. A residual risk is that, at login, a user may recall a similar decoy and trigger the honeychecker (a benign false alarm). We mitigate this by spacing decoys so they are similar yet distinct (same length and character classes but avoiding near-clones), offering a brief mnemonic and optional one-time rehearsal at enrollment, and treating honeyword hits as suspicious but recoverable events (e.g., temporary throttling or an extra verification step) rather than an irreversible lockout. The exact handling policy is deployment-specific and outside the scope of this work.

### 3.5 Why (and How) We Use an LLM for Honeywords

We use the LLM to bias decoys toward realistic, PII-flavoured variations (e.g., mixing chunks, digits, and symbols in more human-like ways), improving diversity and perceived plausibility without changing the underlying non-semantic process.

*Non-LLM control.* To show that indistinguishability does not rely on the LLM, we also generate all $H$ decoys directly from the same 4-gram process (same length; one digit and one special under identical position priors). On 16-char lists with $H$=40, discrimination is at or below chance for Ranker/LM/Joint (Top-1 $\leq$ 2%, AUC $\approx$ 0.43–0.47); see Table 3 in Sec. 3.15. In other words, *process-matched* decoys are already flat; the LLM is a convenience for shaping closeness and diversity, not a security assumption.

### 3.6 Architecture and Workflow

**Step 1: Data prep:** PII $\rightarrow$ consolidated string; fixed-length chunking.
**Step 2: Generation:** Non-semantic password + 19 honeywords (no templates disclosed); strength screen with `zxcvbn`.
**Step 3: Attacker calibration:** Assume semantic segmentation; setup in Sec. 3.10.
**Step 4: Targeted guessing (TarGuess-I style):** Per user: *enumerator* builds candidates (defines *Coverage*); *guesser* scores; *ranker* reorders; *joint* score (Eq. 4) ranks the list *(pipeline in Alg. A1, Appendix. A)*.
**Step 5: Training:** Ranker trained with listwise/pairwise loss; guesser optionally

fine-tuned with policy gradient (Eqs. 5, 6) *(training loop in Alg. A2; optional policy gradient in Alg. A3).*
**Step 6: Metrics under budget $B$:** Hits@$B$ (Eq. 9), MRR (Eq. 10), Coverage (Eq. 8); honeyword indistinguishability and usability studies as described.

### 3.7 Targeted-Guessing Evaluation (TarGuess-I Style)

We evaluate targeted guessing by mimicking RankGuess in a TarGuess-I setting. For each user, an *enumerator* first builds a candidate list; its *coverage* tells us whether the true password appears in that list at all. A language-model *guesser* then assigns scores to candidates, a learned *ranker* reorders them using features learned from data, and a *joint* model (after policy-gradient updates) aligns the guesser with the ranker's preferences. We compare our pipeline against a PCFG-PII (TarGuess-like) baseline under the same per-user guess budgets $B \in \{20, 50, 200, 500\}$. The main metrics are *Hits@B* (success within $B$ guesses) and *MRR* (Mean Reciprocal Rank). We report coverage explicitly, since at larger budgets performance is limited primarily by whether enumeration included the true password, not by how candidates are ordered. In short, small-$B$ tests highlight ranking quality; large-$B$ tests highlight enumeration. Full pipeline pseudocode is in Appendix A.

### 3.8 Joint Training (RankGuess-Style)

We first pretrain the components separately, then align them jointly. The guesser LM $\theta$ is pretrained with maximum likelihood on enumerated candidates conditioned on user context (PII), so it learns a smooth $\log p_\theta(s \mid u)$ over the candidate space. The ranker $\phi$ is trained with a listwise objective (Eq. 7) on per-user candidate lists, producing $R_\phi(s, u)$. We then combine both signals via the joint score in Eq. 4 with mixture $\mu$ and length penalty $\gamma$. Finally, we fine-tune the guesser with policy gradient while keeping the ranker fixed, so that the guesser's sampling policy drifts toward strings preferred by the joint score without changing enumeration (thus coverage remains identical pre/post). Full training-loop pseudocode is in Appendix A, Alg. A2.

### 3.9 Policy-Gradient Fine-Tuning

We optimize the guesser policy $\pi_\theta$ to maximize the expected joint score (Eq. 5). For each user $u$, we sample a candidate $s$ from $\pi_\theta(\cdot \mid u)$ restricted to $\mathrm{Cand}(u)$, compute the reward $\mathrm{JScore}(s \mid u)$ (Eq. 4), and update $\theta$ with the policy-gradient estimator in Eq. 6, using a moving-average baseline $b$ for variance reduction and an entropy bonus (coefficient $\lambda$) for exploration. The ranker $\phi$ is kept fixed; the enumerator is unchanged, so coverage is unaffected. We early-stop based on validation Hits@$B$ and MRR, ensuring the post-RL model improves early-budget success without overfitting. Step-by-step updates appear in Appendix A, Alg. A3.

### 3.10 Attacker Calibration (Segmentation Benchmark)

To avoid underestimating attacker capabilities, we calibrate semantic capability by benchmarking a compact LLM segmenter against [18] on their public *HashtagMaster* corpus (12,594 annotated hashtags). We report standard metrics Boundary-F1, Segment-F1, Exact Match, and Over/Under-seg. The compact LLM is competitive with [18], so all subsequent security experiments assume best-in-class segmentation.

*Why this calibration?* We calibrate attacker pre-processing against [18], which provides public code and a widely used 12,594-item benchmark. Subsequent datasets and studies report results against this model as a reference point [14], and zero-shot multilingual work cites and positions results with respect to it [21]; broader surveys also reference this approach [7]. We reproduce its evaluation protocol on the same corpus and attain comparable Boundary/Segment-F1, reflecting an attacker equipped with an established segmentation capability.

### 3.11 Honeyword Indistinguishability

*Generating honeywords:* For each real password produced by our method, we use a fixed, LLM–assisted procedure (GPT-4) to synthesize nineteen honeywords from the same chunk source. We tune the prompt/policy once and then freeze it, targeting a high `zxcvbn` strength tier for all sweetwords.

*Closeness check (cosine similarity):* Before testing attackers, we verify that honeywords are close to the real password. We embed strings as vectors and compute cosine similarity,

$$\cos\,\mathrm{sim}(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\,\|\mathbf{B}\|},$$

where values near 1 indicate high similarity. We use this check to screen for decoys that are plausibly close to the real password while avoiding duplicates.

*Attacker discrimination test.* We then ask informed scorers to pick the real password from its 19 honeywords (20 candidates total). To make the task well-defined, we restrict to users whose real password appears in the enumerated set (coverage). We evaluate two scorers our post-RL *Joint* model and the *Ranker-only* variant and report Top-1, AUC, and MRR on the covered users. For a uniform random picker over 20 items (1 real + 19 honeywords), the analytic chance baseline is: Top-1 $= 1/20 = 0.05$, AUC $= 0.50$, and MRR $\approx 0.18$.

*Goal:* This two-step procedure (i) enforces closeness and (ii) tests discrimination first checks that honeywords are close to the real password, then asks whether a strong attacker can spot the real one; if they cannot do better than guessing, the honeywords are indistinguishable.

### 3.12 Strength and Usability

**Strength:** We score every sweetword with `zxcvbn` [32]. We first tune the generation policy (prompt/settings) until candidates reliably hit the highest strength tier, then freeze it for all subsequent creation. We evaluate three groups under the same screening: (i) *PII* passwords created by our PII password generation method; (ii) *Random-PII (RaPII)* built by randomly recombining PII-like chunks from the 4iQ corpus; and (iii) *Pure Random (RaPass)* produced with a true-random source (random.org).

**Usability and memorability:** We ran a three-stage user study to assess memorability and perceived usability across three conditions: PII passwords, Random-PII (RaPII), and Pure Random (RaPass) ($n$=60). Participants (age $\geq$ 18) were recruited by email at our university[1] and were assigned a single, condition-specific password to test recall. Stage 1 (day 0): memorize and immediate recall. Stage 2 (day 2): short-term recall. Stage 3 (day 6): long-term recall. We quantify recall with success rates and Levenshtein edit distance between the intended and recalled strings. After the long-term test, participants completed the System Usability Scale (SUS); the PII group also answered brief items on comfort with PII-derived passwords. To quantify memorability (recall burden) in this paper, we analyze two SUS-derived, negatively worded items (Q1–Q2), reverse-coded and scaled to 0–100; we do not compute the SUS-10 composite in this analysis to avoid conflating constructs beyond recall.

### 3.13 Segmentation Calibration Results

We benchmark our compact LLM-based segmenter against a hashtag segmenter on the public test set. As shown in Table 1, our model slightly improves Boundary-F1 and Segment-F1 while reducing over/under-segmentation, so we treat segmentation as competitive in the attacker calibration.

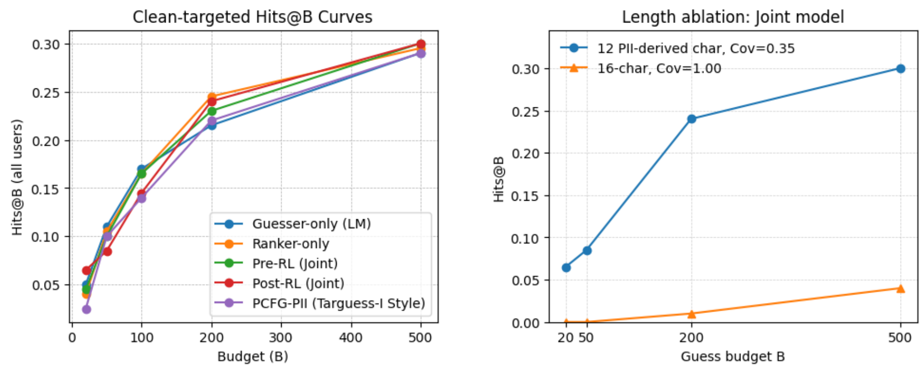
### 3.14 Targeted Guessing (TarGuess-I Style)

**Findings:** Coverage is identical (0.35) across methods enumeration sets the ceiling. Post-RL improves early budgets (H@20 0.045$\rightarrow$0.065; MRR 0.039$\rightarrow$0.053). Curves converge by $B\geq$200 (all $\approx$0.29–0.30), showing enumeration dominates at large budgets. The LM is stronger at very small $B$, the Ranker at larger $B$, and the Joint balances both; against PCFG-PII, our post-RL joint model is best where online attacks matter most (small $B$).
*Robustness.* We next test (i) lengthening to 16 non-semantic characters and (ii) increasing decoys to $H$=40; results appear in Sec. 3.15.

[1] All participants provided informed consent under an approved Research Ethics Board protocol.

Table 1: Semantic segmentation on the public hashtag test set.

| Metric | Baseline | LLM (ours) | $\Delta$ |
|---|---|---|---|
| Boundary-F1 | 0.8555 | **0.8593** | +0.0038 |
| Segment-F1 | 0.6924 | **0.7014** | +0.0090 |
| Exact-match | **0.8178** | 0.6883 | −0.1296 |
| Over/Under | 0.1252 | **0.0584** | −0.0668 |




(a) Hits@B under per-user budgets $B \in \{20, 50, 100, 200, 500\}$.

(b) Length ablation: 12 vs. 16 characters under forced coverage.

Fig. 1: Main guessing performance and length ablation results.

Table 2: Targeted guessing: coverage, MRR, and Hits@B.

| Model | Coverage | MRR | H@20 | H@50 | H@100 | H@200 | H@500 |
|---|---|---|---|---|---|---|---|
| Guesser-only (LM) | 70/200 (0.350) | 0.045 | 0.050 | **0.110** | **0.170** | 0.215 | 0.290 |
| Ranker-only | 70/200 (0.350) | 0.026 | 0.040 | 0.105 | 0.165 | **0.245** | 0.295 |
| Pre-RL (Joint) | 70/200 (0.350) | 0.039 | 0.045 | 0.100 | 0.165 | 0.230 | **0.300** |
| Post-RL (Joint) | 70/200 (0.350) | **0.053** | **0.065** | 0.085 | 0.145 | 0.240 | **0.300** |
| PCFG-PII (TarGuess-I style) | 70/200 (0.350) | 0.016 | 0.025 | 0.100 | 0.140 | 0.220 | 0.290 |

### 3.15 Ablations and Robustness

**Length: 12 vs. 16 non-semantic characters.** We compare the 12-char main setting (3×4 + digit + special; coverage ≈ 0.35) to a 16-char variant (4×4 + digit + special) and evaluate the latter under *forced coverage* (Cov=1.0) to isolate ranking quality.

For the Joint model, early-budget success decreases at 16 chars (H@20/H@50 ≈ 0; H@200 ≈ 0.01; H@500 ≈ 0.05; MRR ≈ 0)(Fig. 1b), indicating that longer *non-semantic* strings harden small-budget online attacks in our setting.

Takeaway. Increasing length with the same generator shifts difficulty from enumeration to ranking and reduces early Hits@$B$, consistent with policies favoring ≥15 characters. **Honeyword size:** $H$=20 **vs.** $H$=40 **(non-LLM control)** We generate all $H$ decoys from the *same* 4-gram process as the real (same length; identical digit/special insertion rules) and evaluate $H$=40 on 16-char lists. *Result (Table 3).* Discrimination remains at/below chance across strong scorers, suggesting robust indistinguishability under process-matched decoys.

Table 3: Honeyword discrimination at $H$=40 (16-char, $n$=100 users). Values closest to the chance row indicate better indistinguishability.

| Model | Top-1 (%) | AUC | MRR |
|---|---|---|---|
| Ranker-only | 0.00 | **0.4740** | 0.0783 |
| Guesser-only (LM) | **2.00** | 0.4325 | **0.0903** |
| Joint ($\mu$=0.6) | 1.00 | 0.4293 | 0.0774 |
| Chance ($H$=40) | 2.44 | 0.5000 | 0.1049 |

**Honeyword Indistinguishability.** We evaluate only covered users (real $\in$ enum): **70/200** (35%). For each user, the scorer must pick the real password from a set of 20 items (1 real + 19 honeywords). We evaluate two informed scorers our post-RL *Joint* model and the *Ranker-only* variant and report Top-1, AUC, and MRR on the covered users. For a uniform random picker over 20 items, the analytic chance baselines are: Top-1 = 1/20 = 0.05, AUC = 0.50, and MRR $\approx$ 0.18.
RQ3 (Are honeywords indistinguishable?). Coverage for this test is 70/200 users (35%). Against 20 honeywords, strong scorers are only slightly above chance (Joint: Top-1 7.1%, AUC 0.557; Ranker-only: Top-1 12.9%, AUC 0.566). In short, the real secret looks almost like the decoys exactly what we want. Ranker-only has a small edge, but it's still close to random guessing in this setup. **Strength (`zxcvbn`).** Across groups, `zxcvbn` scores are near-maximal. **PII** and **RaPass** achieve **100%** at score 4 with no instances in scores 0–3. **RaPII** scores **99.9%** at score 4 with a small **0.1%** tail at score 3 and none in 0–2. This rare low-score tail likely stems from reduced character variety and does not affect our downstream security conclusions.

**Honeyword Closeness.** Across $n$=11 participants, cosine similarity between each real password and its honeywords averages $\approx$ **0.89**, with minimum **0.824**, maximum **1.000**, and standard deviation $\approx$ **0.03**. Decoys therefore lie close to their real counterparts in embedding space while preserving diversity precisely the balance needed for indistinguishability without duplication.

**Usability and Memorability.** We assess memorability using Levenshtein distance between the intended password and the user's recall at Day 1, Day 3, and Day 7. Distances increase with delay, and Table 4 shows that *PII* and *RaPII* consistently achieve lower (better) distances than *RaPass*. These trajectories indicate that PII-derived passwords (and their RaPII counterpart) retain better recall fidelity over time than purely random strings. A two-way ANOVA on Levenshtein distance with factors *Password Type* (PII, RaPII, RaPass) and *Recall Day* (1, 3, 7) shows significant main effects of both Type and Day ($p < 0.001$) and no interaction, confirming that PII and RaPII have lower distances than RaPass and that distances increase with delay in a similar pattern across groups.

Table 4: Levenshtein distance between intended password and recalled password over time (mean and standard deviation). Lower values indicate better recall.

| Group | Day 1 mean | Day 1 SD | Day 3 mean | Day 3 SD | Day 7 mean | Day 7 SD |
|---|---|---|---|---|---|---|
| PII | 0.75 | 2.66 | 3.85 | 3.98 | 4.47 | 3.92 |
| RaPII | 0.25 | 0.53 | 2.64 | 3.43 | 3.32 | 4.74 |
| RaPass | 1.35 | 2.98 | 5.71 | 5.00 | 7.15 | 5.00 |

## 4 Discussion and Limitations

In targeted guessing, the RankGuess-style Joint scorer improves early-budget Hits@$B$/MRR, while performance at $B$=500 clusters near $\approx$ 0.30, consistent with the shared coverage cap $\approx$ 0.35, so small-$B$ outcomes reflect ranking quality and large-$B$ are enumeration-limited. Our attacker calibration assumes strong semantic pre-processing: the compact LLM segmenter matches a state-of-the-art hashtag benchmark (Table 1), and honeyword discrimination remains close to chance when decoys share the same non-semantic process (Table 5).

*Limitations.* Coverage $\approx$ 0.35, a finite cohort and PII mix, offline evaluation of an online targeted attacker (excluding offline hash-cracking, reuse, and pivoted leaks), uniform decoy sampling, and the lack of experiments on large real PII–password leaks or deep comparisons to end-to-end LLM guessers such as PassLLM all bound the scope and are left for future work.

## 5 Conclusion

We evaluated a non-semantic, PII-derived password and honeyword scheme under a targeted online attacker with strong semantic pre-processing and PII-conditioned scoring. The RankGuess-style Joint scorer improves early-budget success, but coverage caps performance at larger budgets, and honeyword discrimination remains near chance. Overall, non-semantic 4-gram construction with digit/symbol insertion and honeywords resists targeted online guessing under our calibrated attacker model.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

**Acknowledgements** The authors acknowledge the support of a Discovery Development Grant from the Natural Sciences and Engineering Research Council of Canada, number 110-2024-2025-Q1-4039.

## References


1. AlQahtani, A.A.S., Alamleh, H., Gourd, J.: 0eisua: Zero effort indoor secure user authentication. IEEE Access **8**, 79069–79078 (2020)
2. Alroomi, S., Li, F.: Measuring website password creation policies at scale. In: Proceedings of the 2023 ACM SIGSAC conference on computer and communications security. pp. 3108–3122 (2023)
3. Baig, A.F., Eskeland, S.: Security, privacy, and usability in continuous authentication: A survey. Sensors **21**(17), 5967 (2021)
4. Bhana, B., Flowerday, S.V.: Usability of the login authentication process: Passphrases and passwords. Information & Computer Security **30**(2), 280–305 (2022)
5. Chakraborty, N., Li, J., Leung, V.C., Mondal, S., Pan, Y., Luo, C., Mukherjee, M.: Honeyword-based authentication techniques for protecting passwords: A survey. ACM Computing Surveys **55**(8), 1–37 (2022)
6. Chakraborty, N., Yamout, Y., Zulkernine, M.: The tables have turned: GPT-3 distinguishing passwords from honeywords. In: 2023 IEEE Conference on Communications and Network Security (CNS). pp. 1–5. IEEE (2023)
7. Chen, S., Zhang, Y., Yang, Q.: Multi-task learning in natural language processing: An overview. ACM Computing Surveys **56**(12), 1–32 (2024)
8. Clark, M., Ruoti, S., Mendoza, M., Seamons, K.: A comparison of three approaches to assist users in memorizing system-assigned passwords. In: Symposium on Usable Security and Privacy (USEC) 2024. Internet Society (2024)
9. Constantinides, A., Belk, M., Fidas, C., Beumers, R., Vidal, D., Huang, W., Bowles, J., Webber, T., Silvina, A., Pitsillides, A.: Security and usability of a personalized user authentication paradigm: Insights from a longitudinal study with three healthcare organizations. ACM Transactions on Computing for Healthcare **4**(1), 1–40 (2023)
10. Dionysiou, A., Vassiliades, V., Athanasopoulos, E.: Honeygen: Generating honeywords using representation learning. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. pp. 265–279 (2021)
11. Erguler, I.: Achieving flatness: Selecting the honeywords from existing user passwords. IEEE Transactions on Dependable and Secure Computing **13**(2), 284–295 (2015)
12. Ezugwu, A., Ukwandu, E., Ugwu, C., Ezema, M., Olebara, C., Ndunagu, J., Ofusori, L., Ome, U.: Password-based authentication and the experiences of end users. Scientific African **21**, e01743 (2023)
13. Juels, A., Rivest, R.L.: Honeywords: Making password-cracking detectable. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 145–160 (2013)
14. Kodali, P., Bhatnagar, A., Ahuja, N., Shrivastava, M., Kumaraguru, P.: HashSet - a dataset for hashtag segmentation. In: Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Odijk, J., Piperidis, S. (eds.) Proceedings of the Thirteenth Language Resources and Evaluation Conference. pp. 7215–7219. European Language Resources Association, Marseille, France (Jun 2022), https://aclanthology.org/2022.lrec-1.782/
15. Li, Y., Wang, H., Sun, K.: A study of personal information in human-chosen passwords and its security implications. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. pp. 1–9. IEEE (2016)

16. Liu, Y., Lin, F.Y., Ebrahimi, M., Li, W., Chen, H.: Automated PII extraction from social media for raising privacy awareness: A deep transfer learning approach. In: 2021 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 1–6. IEEE (2021)
17. Lukas, N., Salem, A., Sim, R., Tople, S., Wutschitz, L., Zanella-Béguelin, S.: Analyzing leakage of personally identifiable information in language models. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 346–363. IEEE (2023)
18. Maddela, M., Xu, W., Preoţiuc-Pietro, D.: Multi-task pairwise neural ranking for hashtag segmentation. In: Korhonen, A., Traum, D., Màrquez, L. (eds.) Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 2538–2549. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1242, https://aclanthology.org/P19-1242/
19. Nam, S., Jeon, S., Kim, H., Moon, J.: Recurrent GANs password cracker for IoT password security enhancement. Sensors **20**(11), 3106 (2020)
20. Oest, A., Zhang, P., Wardman, B., Nunes, E., Burgis, J., Zand, A., Thomas, K., Doupé, A., Ahn, G.J.: Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In: 29th USENIX Security Symposium (USENIX Security 20) (2020)
21. Rodrigues, R.C., Inuzuka, M.A., Gomes, J.R.S., Rocha, A.S., Calixto, I., Nascimento, H.A.D.d.: Zero-shot hashtag segmentation for multilingual sentiment analysis. arXiv preprint arXiv:2112.03213 (2021)
22. Roy, A., Memon, N., Ross, A.: Masterprint: Exploring the vulnerability of partial fingerprint-based authentication systems. IEEE Transactions on Information Forensics and Security **12**(9), 2013–2025 (2017)
23. Senthilnayaki, B., Mahalakshmi, G., Dharanyadevi, P., Narashiman, D.: Detection of attacker using honeywords (2022)
24. Temoshok, D., Choong, Y.Y., Regenscheid, A., Galluzzo, R., Fenton, J.L., Richer, J., Lefkovitz, N.: NIST SP 800-63B-4: Digital identity guidelines-authentication and authenticator management (2025)
25. Ur, B., Noma, F., Bees, J., Segreti, S.M., Shay, R., Bauer, L., Christin, N., Cranor, L.F.: "I added '!' at the end to make it secure": Observing password creation in the lab. In: Eleventh symposium on usable privacy and security (SOUPS 2015). pp. 123–140 (2015)
26. Wang, D., He, D., Cheng, H., Wang, P.: FuzzyPSM: A new password strength meter using fuzzy probabilistic context-free grammars. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 595–606. IEEE (2016)
27. Wang, D., Shan, X., Dong, Q., Shen, Y., Jia, C.: No single silver bullet: Measuring the accuracy of password strength meters. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 947–964 (2023)
28. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: An underestimated threat. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 1242–1254 (2016)
29. Wang, D., Zou, Y., Dong, Q., Song, Y., Huang, X.: How to attack and generate honeywords. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 966–983. IEEE (2022)
30. Wang, D., Zou, Y., Zhang, Z., Xiu, K.: Password guessing using random forest. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 965–982 (2023)
31. Wash, R., Rader, E.: Prioritizing security over usability: Strategies for how people choose passwords. Journal of Cybersecurity **7**(1) (2021)

32. Wheeler, D.L.: zxcvbn: Low-budget password strength estimation. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 157–173 (2016)
33. Xu, M., Yu, J., Zhang, X., Wang, C., Zhang, S., Wu, H., Han, W.: Improving real-world password guessing attacks via bi-directional transformers. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 1001–1018 (2023)
34. Yang, T., Wang, D.: RankGuess: Password guessing using adversarial ranking. In: 2025 IEEE Symposium on Security and Privacy (SP). pp. 682–700. IEEE (2025)
35. Yang, W., Chen, Z., Huang, J., Kang, W.: A novel system and experimental study for 3d finger multibiometrics. IEEE Transactions on Biometrics, Behavior, and Identity Science **4**(4), 471–485 (2022)
36. Yu, F., Vargas Martin, M.: GNPassGAN: Improved generative adversarial networks for trawling offline password guessing. In: 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). pp. 10–18. IEEE (2022)
37. Yu, F., Vargas Martin, M.: Honey, I chunked the passwords: Generating semantic honeywords resistant to targeted attacks using pre-trained language models. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 89–108. Springer (2023)
38. Zou, Y., An, M., Wang, D.: Password guessing using large language models. In: 34th USENIX Security Symposium (USENIX Security 25). pp. 7799–7818 (2025)

## Appendix A: Supplementary Algorithms and Tables

**Algorithm A1** Enumerate → Score → Rank (per user)

**Require:** user $u$, context $q$, enumerator $\mathcal{E}$, guesser $\theta$, ranker $\phi$, $\mu$, $\gamma$, budget $B$

$\mathrm{Cand}(u) \leftarrow \mathcal{E}(u, q)$
$\mathrm{Coverage} \leftarrow \mathbb{I}[\, p_u \in \mathrm{Cand}(u)\, ]$ ▷ Eq. 8
**for** $s \in \mathrm{Cand}(u)$ **do**
  $\ell(s) \leftarrow \log p_\theta(s \mid u); \quad r(s) \leftarrow R_\phi(s, u)$
  $\mathrm{JScore}(s \mid u) \leftarrow \mu\, \ell(s) + (1-\mu)\, r(s) - \gamma\, |s|$ ▷ Eq. 4
**end for**
$L(u) \leftarrow$ sort by JScore; $\mathrm{rank}_u \leftarrow$ rank of $p_u$
$\mathrm{Hits@}B \leftarrow \mathbb{I}[\mathrm{rank}_u \leq B]; \quad \mathrm{MRR}_u \leftarrow \mathrm{Coverage}/\mathrm{rank}_u$

**Algorithm A2** Training Joint RankGuess-Style Guesser/Ranker

**Require:** dataset $\mathcal{D}$, enumerator $\mathcal{E}$, budgets $B$, mixture $\mu$, penalty $\gamma$

Initialize $\phi$ (ranker), $\theta$ (guesser); pretrain $\theta$ (MLE), train $\phi$ (Eq. 7)
**for** epoch $= 1..E$ **do**
  **for** $u \in \mathcal{D}$ **do**
    $\mathrm{Cand}(u) \leftarrow \mathcal{E}(u)$; sample $s \sim \pi_\theta(\cdot \mid u)$
    reward $\leftarrow \mathrm{JScore}(s \mid u)$ (Eq. 4); update $\theta$ via Eq. 6
  **end for**
**end for**

**Algorithm A3** Policy-Gradient Fine-Tuning of the Guesser

**for** $e = 1..E$ **do**
  **for** $u \in \mathcal{D}$ **do**
    build $\mathrm{Cand}(u)$; sample $s \sim \pi_\theta(\cdot \mid u)$; $r \leftarrow \mathrm{JScore}(s \mid u)$
    $\theta \leftarrow \theta + \eta\, [\nabla_\theta \log \pi_\theta(s \mid u)(r - b) + \lambda \nabla_\theta H(\pi_\theta)]$
  **end for**
**end for**

Table 5: Honeyword discrimination (H=20 negatives; covered users only). Chance Top-1 = $1/(H+1) = 0.048$, AUC = 0.50, MRR $\approx 0.173$.

| Model (scorer) | Users eval. | Top-1 | MRR | AUC |
|---|---|---|---|---|
| Chance baseline | 70 | 0.048 | 0.173 | 0.500 |
| Joint ($\mu$=0.6) | 70 | 0.071 | 0.199 | 0.557 |
| Ranker-only | 70 | 0.129 | 0.254 | 0.566 |

## Appendix B: Human Participant Experiment

*Participants and Allocation:* Enrolled $n$=150 (50 per group); complete-case across all three sessions $n$=60 (20 per group). Between-subject factor: *Password Type* $\in$ {PII, RaPII, RaPass}. Inclusion: age $\geq$18; consented. Random assignment at enrollment; no cross-over.

*Procedure and Timeline:* Three sessions: Day 0 (enrollment, memorization, immediate recall), Day 2 (short-term recall), Day 6 (long-term recall + SUS). All sessions remote; instructions standardized.

*Materials:* Each participant received exactly one password according to group: PII (from our generator), RaPII (random recombination of PII-like chunks), or RaPass (true-random). No real accounts involved.

*Outcomes:* Recall success and SUS.

*Ethics:* Approved by the University Research Ethics Board (REB); consent obtained; All personal data was anonymized and securely stored to safeguard participant privacy, accessible only to the research team.

*Scope and Rationale:* This appendix intentionally reports *only* two SUS-derived memorability items: (Q1) "recalling was unnecessarily difficult" and (Q2) "passwords were very cumbersome to memorize." We did not compute a SUS-10 composite because our analysis here focuses narrowly on *recall burden*, not overall system usability. Item responses were reverse-coded and scaled to 0–100 so that higher values indicate easier recall. These results are *item-level* descriptors for comparing the study groups and are not intended for SUS-10 benchmarking.

*Summary:* Across the two items, PIIPass scored highest (easiest to recall), RaPII was intermediate, and RaPass scored lowest (hardest to recall). Confidence intervals indicate a clear gap between PIIPass and RaPass.

Table 6: Two memorability items (Q1–Q2), reverse-coded to 0–100 where higher indicates easier recall/usability.

| **Group** | $n_{\mathrm{Q1}}$ | **Mean Q1** | $n_{\mathrm{Q2}}$ | **Mean Q2** | $n_{\mathrm{all}}$ | **Overall [95% CI]** | **Interpretation** |
|---|---|---|---|---|---|---|---|
| PIIPass | 25 | 66.0 | 24 | 61.5 | 49 | 63.8 [56.9, 70.6] | Easiest to recall |
| RaPII | 24 | 58.3 | 23 | 51.1 | 47 | 54.8 [46.1, 63.4] | Moderate |
| RaPass | 30 | 40.0 | 30 | 35.8 | 60 | 37.9 [29.9, 45.9] | Hardest to recall |

## Appendix C: Generation Prompts

**Non-semantic chunking (fixed 4-grams).**

> Given a raw PII string $S$, output overlapping 4-gram windows preserving original casing and characters. Return as a JSON array of strings. Do not add or infer content.
> *(Redaction: prompt includes only procedural constraints; no live PII or site-specific examples.)*

**Seed construction (12 chars).**

> Sample three (or four) 4-grams uniformly at random from the window list and concatenate to 12 chars. Insert exactly one digit and one special from the allowed sets at random positions; avoid adjacent duplicates.
> *(Redaction: symbol set and sampling priors summarized, not enumerated verbatim.)*

**Honeyword generation (19 decoys).**

> From the same 4-gram pool, generate 19 decoys that match length and character-class constraints and avoid near-clones. Output a JSON list.
> *(Redaction: phrasing variants omitted)*

## Appendix D: Attacker/Ranker Model Configs (RankGuess-style mimic)

Table 7: Concise attacker/ranker settings for verification.

| **Component** | **Setting** |
|---|---|
| Joint score | $\mu = 0.6$, $\gamma = 0.0$ (len penalty off) unless stated; see Eq. 4 |
| Enumerator caps | per-slot = 30; global = 50,000 (fixed across runs) |
| Budgets | $B \in \{20, 50, 200, 500\}$; forced-coverage used only in ablation |

## Appendix E: Qualitative, Anonymized Examples

### PII-derived secret and honeywords (12-char case)

**Secret (user-chosen from the 20-item list):**

`nguaxxxxregi!5` (*structure:* 3×4-grams + special + digit)

**Honeywords (sample 5 of 19; similar yet distinct):**

`nguaxxxre!76gi`, `ngua@45xxregi!`, `92ngua#xxregii`, `nguaxx8%8xregi`, `ngu1axxxr!4egi`

*Notes:* Synthetic placeholders illustrate the *look* (4-gram surface + one digit + one special); no real PII used. Decoys share n-gram patterns and character classes but avoid near-duplicates.