Universal Neural-Cracking-Machines:
Self-Configurable Password Models from Auxiliary Data*

Dario Pasquini
SPRING Lab, EPFL
Lausanne, Switzerland
dario.pasquini@epfl.ch

Giuseppe Ateniese
George Mason University
Fairfax, Virginia, USA
ateniese@gmu.edu

Carmela Troncoso
SPRING Lab, EPFL
Lausanne, Switzerland
carmela.troncoso@epfl.ch

Abstract—We introduce the concept of *"universal" password model*—a password model that, once pre-trained, can automatically adapt its guessing strategy based on the target system. To achieve this, the model does not need to access any plaintext passwords from the target credentials. Instead, it exploits users' auxiliary information, such as email addresses, as a proxy signal to predict the underlying password distribution.

Specifically, the model uses deep learning to capture the correlation between the auxiliary data of a group of users (e.g., users of a web application) and their passwords. It then exploits those patterns to create a tailored password model for the target system at inference time. No further training steps, targeted data collection, or prior knowledge of the community's password distribution is required.

Besides improving over current password strength estimation techniques and attacks, the model enables any end-user (e.g., system administrators) to autonomously generate tailored password models for their systems without the often unworkable requirements of collecting suitable training data and fitting the underlying machine learning model. Ultimately, our framework enables the democratization of well-calibrated password models to the community, addressing a major challenge in the deployment of password security solutions at scale.

I. INTRODUCTION

Passwords are still an integral part of our modern lives. Their true purpose is to provide a layer of protection for our personal data, which is always more susceptible to the risk of being *hacked* [13]. For decades there has been a heated debate over whether or not we should ditch passwords and rely on more advanced forms of authentication, such as biometrics and hardware-based authentication [12], [63]. However, that argument underestimates the potential for vulnerabilities in these techniques too. Furthermore, relying solely on these methods may shoulder a heavy burden for better outfitted and affluent populations but ignore the segments of society who are unable to equip themselves with these newer technologies due to financial or institutional constraints.

Password managers enable users to have unique, strong passwords for different accounts without having to remember them on their own. However, they have also been judged negligent in recent security breaches [54], [16]. If the company goes down or someone manages to hack into it, all the data encrypted with their systems may become exposed [44] or inaccessible. Users should still use password managers and ensure they have plenty of different, *hard-to-guess* passwords. Yet, it is important for all parties involved to understand that this system requires more trust from the user than traditional passwords [16].

Passwords are not dead, they are just evolving. We trust that they still provide an effective fallback method for security verification, even if biometrics or hardware authentication do not work. In fact, passwords are still the most popular form of authentication in use today. However, it is widely recognized that human-generated passwords have inherent flaws. To ensure that authentication systems provide a meaningful level of resilience, the security community has been working to create a range of tools and methodologies centered around passwords. Those include but are not limited to: reactive security analysis [35], [60], proactive approaches such as Password Strength Meters (PSMs) [59], [25], breach alert systems [49], honey-encryption-based password vaults [15], [65], and tailored hash functions [5]. While explicating very different purposes, almost all these techniques rely on the same fundamental and critical component—a so-called **password model**. This is a data-driven adversary model that allows to estimate password strength by simulating password-guessing attacks.

Unfortunately, password models are not universal tools. The composition habits of passwords vary from one community of users to another [20]. As a result, in order to be effective, **password models must be specifically designed to match the password distribution of the community where they will be used**. However, this is not an easy task.

Given a target credential database (e.g., the one associated to a web application), the expected workflow requires the end-user (e.g., the system administrator) to collect a suitable training set for the password model before deploying it. The system administrator must gather one or more public password leaks that have a password distribution *similar* to the target one, and use the collected data to train the password model; typically a machine learning model.

Previous research has often assumed an *intra-site* scenario, where the password model is trained on passwords from the same database where it will be used. However, this approach does not reflect any real-world scenario. When deploying a password model, system administrators can only access

---

the stored credential hashes and do not have a legitimate means to view the original plaintext passwords.[1] This prevents them from using the stored passwords as training data. More critically, it also precludes the use of any similarity metric to determine external password leaks with congruent distribution to the target one.

In the real world, the only viable option is to approximate the concept of "*similar distribution*" by relying on known macro-characteristics of the target community, such as the expected primary language(s) of users and/or the kind of service provided by the web-application, e.g., banking, e-shopping, etc. Pragmatically, if we want to train an accurate password model for a Swiss e-banking service, we would need to fetch a set of leaked passwords originating from a previously breached Swiss e-banking website.

Unfortunately, in practice, the system administrators need to learn where to find such suitable password leaks. Those are often available only in underground forums or hidden behind a paywall or credit system. Ultimately, data with the requested properties may not be available in enough quantity to fit a password model. However, even if the data is simply there and easy to access, state-of-the-art password models are still built on top of machine learning models that must be trained before use [43]. Unfortunately, training and calibrating machine learning models is never a straightforward process. This is especially true for system administrators of small web applications who might not have the expertise, experience and/or ability to afford the time (and cost) needed to train models that perform this job (which may take days of GPU time [43], [52]). Another thing to consider is that different training sets require different settings; while learning-based password models are more robust than simpler approaches such as dictionary attacks [39], they still rely on some degree of hyperparameter tuning that, whether successful or not, may end up with suboptimal password models regardless of the quality of the training set. **Ultimately, while password security depends on the existence of well-trained and calibrated password models, this requirement is inherently hard to meet in reality - especially by the average end-user.**

In this work, we move towards solving this fundamental issue. We introduce the first **universal password model**; a password model that, once trained, can automatically optimize its guessing strategy based on the context, without requiring users to run additional training steps or undertake targeted data collection. At inference time, the model does not need access to any plaintext passwords from the target set. Instead, it relies on auxiliary data, such as email addresses and usernames, to build a predictive model for the underlying target passwords. We call this family of models *Universal Neural-Cracking-Machines* or UNCMs for short.

The main intuition is that human-chosen passwords and personally identifiable information that a person typically shares at registration time (such as their name, address, and phone number) are naturally correlated with each other. A UNCM is a password model that exploits this natural correlation to adapt to the target password distribution. Going beyond just individual pairs of users and passwords [14], [64], [38], a UNCM extends this correlation to whole communities. In particular, **a UNCM uses deep learning, specifically an attention mechanism [61], to combine and interpolate together information about users in the same group to define a robust and accurate prior over their password distribution.** This prior is then utilized to dynamically optimize the model's configuration at inference time, enhancing its capability to guess the user passwords within the target system.

In the paper, we show that, although requiring no human intervention and having no explicit prior knowledge about the target, a UNCM can outperform equivalent password models that have been manually calibrated for the target distribution based on prior information (e.g., target users' language). We also show that user information used for the configuration process can be efficiently anonymized via Differential Privacy, enabling the publication of the model with formal privacy guarantees. The automatic configuration process is lightweight and has sub-second latency, making it possible for every end-user to generate accurate password models. Furthermore, the resulting password models can be efficiently compressed [43] and used as client-side password strength meters, enabling a wide range of applications.

Our contributions can be summarized as follows:

- We introduce the concept of universal password model and demonstrate that auxiliary information associated with users can be combined and used to learn an accurate prior on the password distribution of the underlying user's community.
- We develop a fully-automatic approach to exploit auxiliary information and instantiate context-aware password meters and guessing attacks without requiring any plaintext samples from the target password distribution.
- We introduce a differentially-private version of our framework, which is well-suited for building client-side password strength meters. This provides the **formal** guarantee that no meaningful amount of information on the individuals who provided the auxiliary data for the model configuration is leaked upon model publication. The proposed framework enables us to reach $\epsilon < 1$ (i.e., strong privacy guarantees [24]) with only a limited loss in utility.
- Incidentally, our work also defines the new state-of-the-art for offline password guessing attacks as well as password strength metering.

We provide access to our code and pre-trained models.[2]

II. PRELIMINARIES AND BACKGROUND

In this section we go over some necessary background. We start by covering password models and surveying relevant related works in password security. Section II-B introduces the concept of *attention* in machine learning. Finally, Section II-C

---

[1]The standard security principle is that plaintext passwords should not be stored on the server.

gives a brief introduction to differential privacy. Section II-D concludes by formalizing the notation used within the paper.

A. Passwords Models

Human-chosen passwords do not distribute uniformly in the key space $\mathcal{X}$ (the set of all possible passwords). Instead, users tend to choose easy-to-remember strings that aggregate in a relatively few dense clusters [11], [53]. Real-world passwords result in predictable distributions that can be easily modeled by an adversary, making authentication systems susceptible to **guessing attacks** [45], [47]. In a guessing attack, an adversary aims to recover plaintext credentials by attempting several candidate passwords (**guesses**) till success or budget exhaustion [9]; this happens by either searching for the pre-image of password hashes (offline attack) or attempting remote logins (online attack). In this process, the attacker relies on a so-called **password model** that defines which, and in which order, guesses should be tried to maximize the effectiveness of the attack i.e., the expected number of recovered credentials per guess [11].

In general, a password model can be understood as an estimate of the password distribution that allows for an informed search of the key space. However, the focus of this search can vary depending on the objective. It can be heavily conditioned on auxiliary information, as in the case of targeted online attacks against individual accounts [48], [64], [17], or based on a less informative prior, as in the more common trawling offline setting [60]. In this work, we focus on the latter case, and human-chosen passwords.

a) Implementations: Existing password models construct over a heterogeneous set of assumptions and implementations.

Probabilistic password models assume that real-world password distributions are sufficiently smooth to be accurately described by a suitable parametric probabilistic model f_Θ. Here, a password mass function is explicitly [40], [43], [47] or implicitly [52] derived from a set of observable data (i.e., previously leaked passwords), typically, via maximum likelihood estimation [43], [47]. This is then used to assign a probability to each element of the key space. During the guessing attack, guesses are produced by traversing the key space following the decreasing probability order imposed by f_Θ.

Other hybrid approaches such as Probabilistic Context-Free Grammars (PCFG) and dictionary attacks [45], [60], [39], [51], instead, rely on simpler and more intuitive constructions, which tend to be closer to human logic. Generally, those assume passwords as realizations of templates and generate novel guesses by abstracting and applying such patterns on token collections. Those tokens are either directly given as part of the model configuration (e.g., the dictionary and rules-set for dictionary attacks) or extracted from observed passwords in a training phase (e.g., terminals/grammar for PCFG). In contrast with parametric models, these can produce only a limited number of guesses, which is a function of the chosen configuration.

b) Applications: Password models are essential for password security. In addition to their use in simulated guessing attacks for reactive security analysis [35], [60] (e.g., penetration testing) and proactive approaches such as Password Strength Meters (PSMs), password models are widely used in a variety of password security applications, including breach alert systems [49], honey encryption/words [15], [65], and distribution-aware hash functions [5].

In this work, we primarily focus on the application of our methodology as PSMs (see Section II-A1). This is the most common application of password models in the real world [18] and therefore has the most impact. However, we emphasize that the results achieved by our methodology apply equally to all other applications of password models, affecting the password security community as a whole.

1) Password Strength Meters (PSMs): Within the PSM context, every password model can be understood as a function that maps each password $x \in \mathcal{X}$ to a **guess number** g—the rank of the password x in the guesses generation process of the attack. Under the assumption that the employed password model captures an accurate adversary model, the guess number is universally recognized as the best estimator of password security [35], [60]. Pragmatically, passwords with low guess numbers are attested to be weak, whereas passwords with high guess numbers are considered secure.

Once a password model has been fitted to model a suitable password distribution, it can be deployed as PSM; typically in web-applications [58], [43], [50], [67]. Here, the password model is embedded into the sign-up page of the target web app and used to meter users' passwords at composition time. Supported by tailored interfaces and feedback mechanisms [58], [67], [50], the meter dissuades users from picking passwords with low guess numbers. PSMs, when correctly developed [59], [25], have been shown to be one of the most effective tools to induce stronger passwords for the underlying user community.

2) Autoregressive password models: In the present work, we focus on a specific class of probabilistic password models: the autoregressive ones [43]. An autoregressive password model segments every password x in the key space as a list of atomic components $x = [c^1, c^2, \ldots, c^q]$; typically, at character-level.[3] Then, the model assigns to each element c^i a probability that is conditioned to all the elements that come before in the sequence $P(c^i) = P(c^i|c^1, \ldots, c^{i-1})$. The product of all those local probabilities results in the joint probability of the string x: $P(x) = f_\Theta(x) = f_\Theta(c^1) \cdot f_\Theta(c^2|c^1) \ldots f_\Theta(c^q|c^1, \ldots c^{q-1})$, where f_Θ is the password model—a parametric function (e.g., a neural network) in this context. As for the other models, the set of parameters Θ is derived/learned via maximum likelihood estimation [43], [47] from a set of observable data, which are typically set(s) of previously leaked (plaintext) passwords.

---

[3]Other kinds of segmentation processes such as *n-grams* or semantic tokens are possible [43]; however, in the paper, we focus on character-level models as this is the simplest and most natural implementation.
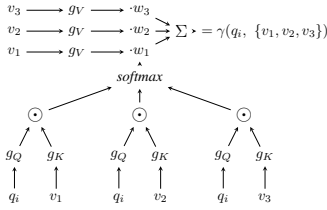
Fig. 1: Depiction of a partial execution of a simplified attention-mechanism for a single query vector q_i and the set of values: $\{v_1, v_2, v_3\}$.

Autoregressive models come with valuable practical properties. In contrast with other neural approaches such as GANs [29], [52] and similar non-autoregressive constructions [51]. Autoregressive models (1) offer enumeration algorithms that allow us to generate/traverse the key space in decreasing probability order and (2) can explicitly assign probabilities to the generated passwords. Such probabilities can then be efficiently mapped to the corresponding guess numbers via suitable estimation techniques [19]. We stress that these properties are pivotal to implementing efficient password strength meters, cast password guessing attacks, and support security studies on plaintext passwords [59], [11], [20].

B. Attention Mechanisms

Attention mechanisms (AMs) are the driving force of modern deep learning. Since their seminal application on Natural Language Processing (NLP) [4], AMs became critical components of state-of-the-art models, culminating in the current "*transformer revolution*" [61], [26], [22].

In contrast to classic architectures (e.g., RNN and CNN) that either imply spatial or temporal ordering on input data, AMs can be seen as a function defined over sets rather than sequences or images. Attention-based networks, unless intentionally amended for that propose (e.g., adding positional encoding [61]), operate on permutation-invariant inputs and naturally abstract operations on length-varying data.

In the setup considered in this paper, an attention mechanism can be understood as a function $\gamma(\cdot, \cdot)$ defined over two sets of equal size vectors.[4] Given a set $Q = \{q_1, q_2, \ldots, q_n\}$ (called **queries**) and a set $V = \{v_1, v_2, \ldots, v_d\}$ (called **values**), for each element $q_i \in Q$, the AM computes a function of q_i and all the other vectors in V. Formally, for each pair $(q_i, v_j \in V)$, γ first maps q_i and v_j via two different dense layers (g_K and g_Q, respectively) and it computes the dot product on their outputs. The resulting scalar values are then scaled via the *softmax* function that returns a weight w_j for each pair (q_i, v_j). The output of the attention mechanism is then the sum of all the input tokens V (mapped through a third dense layer g_V) weighted by the weights produced by the *softmax*. Figure 1 depicts the *computational graph* for a simplified version of the Multi-Head Attention mechanism proposed in [61] on a single query vector and set of three values.

[4]In the most general construction, an attention mechanism is defined over three input sets. In our paper, we consider *keys* and *values* to be the same.

In the password security context, following the success of attention-based architectures in NLP, a growing body of work [7], [37] is proposing to use them as a replacement for classic architectures such as RNN [43] to implement password models, obtaining mixed results. We highlight that, in our work, we do not seek incremental results harnessing improved architectures based on AMs; instead, we rely on their unique technical properties to implement modules of our framework that would be impossible to efficiently implement otherwise (see Section IV-B2).

C. Differential Privacy

Given a privacy budget $\epsilon \in \mathcal{R}^+$, a randomized function $\mathcal{M}$ is ϵ-differentially-private if this verifies Eq. 1 for each possible input pair (D, D') of neighbor sets i.e., sets that differ for a single element.

$$P\left[\mathcal{M}(D) \in \mathbb{S}\right] \leq P\left[\mathcal{M}(D') \in \mathbb{S}\right] \cdot e^\epsilon, \tag{1}$$

for any subset of outputs $\mathbb{S} \subseteq \texttt{Range}(\mathcal{M})$. More pragmatically, being differential private implies that no adversary can infer the result of the predicate $d \in D$ from $\mathcal{M}(D)$ better than $\ln(\frac{TPR}{FPR}) \leq \epsilon$ [33], where TPR and FPR are the *true positive rate* and the *false positive rate* of the attacker in inferring whether $d \in D$. This holds regardless of the amount of information and influences the attacker has on $D/\{d\}$ and $\mathcal{M}$. Given a function $\mathcal{M}$, there are different techniques one can use to make it differentially private. In the paper, we use the **Gaussian mechanism**, which is the ubiquitous choice in ML [2]. The Gaussian mechanism results in (ϵ, δ)-differential-privacy; a relaxed form of DP that accounts for a failure event in which Eq. 1 does not hold for $\mathcal{M}$ with probability δ. Another necessary notion we use in the paper is the one of **sensitivity** of a function $\mathcal{M}$. This is defined as: $\max_{D,D'} \|\mathcal{M}(D) - \mathcal{M}(D')\|$, where D and D' are neighbors sets. The sensitivity quantifies the magnitude by which a single element can change the function $\mathcal{M}$ in the worst case [23].

D. Notation & Definitions

In the paper, we model a **credential database** as a pair $S = (A_{\text{inf}}, H(X))$ (e.g., the credential database of a web application), where X is the set of user-chosen passwords and A_{inf} is the set of **auxiliary information** associated to the passwords in X: users' email addresses, usernames, etc. The letter x is used to identify a single password in X. With H, we denote a secure cryptographic hash function, where $H(X) = \{H(x) | x \in X\}$. We abuse the term **"community"** to refer to the group of users who originated the data in S and use n to express the cardinality of a set of passwords $n = |A_{\text{inf}}| = |X|$. We use the letter u to identify a single user in S and U for a set of users. In the paper, we must often refer to sets of credential databases $\{S_1, \ldots, S_m\}$. For those, we use the term **"leak collection"** and the notation **L**.

Hereafter, without loss of generality, we use the notation f_Θ to refer to a password model, where Θ are the model's parameters. We focus on probabilistic password models; thus, f_Θ is
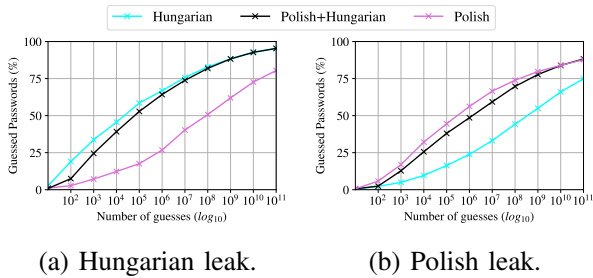
(a) Hungarian leak. (b) Polish leak.

Fig. 2: Results of two guessing attacks for three password models [43] trained to model different language-specific password distributions.

Fig. 3: Depiction of a UNCM and its internal working at deployment time.

intended as a mass function; a function that assigns a probability to each element of the key-space $\mathcal{X}$ and $\sum_{x\in\mathcal{X}} f_\Theta(x) = 1$. Additionally, we use the shorthand notation $f_\Theta(X)$ to refer to $\sum_{x\in X} f_\Theta(x)$. Given a password model, we call **"guessing strategy"** the list of guesses $[x_0, x_1, \dots]$ generated by the model when running a guessing attack. For probabilistic models, the guessing strategy is the key-space sorted by decreasing probability according to f_Θ.

III. On Universal Password Models

In this section, we introduce our main contribution—the concept of Universal Neural-Cracking-Machine (UNCM). To this end, we first review some notions and intuitions that are pivotal to motivating our work.

A. Password Strength is not a Universal Property

Password strength is not a universal concept. While passwords tend to have similar characteristics, different communities choose passwords arbitrarily, resulting in disparate password distributions [20]. In other words, while there are guidelines for what makes a strong or weak password, these are not universal and will vary depending on the context in which the password is created and used.

This has important implications for password models, which are fully data-driven and rely solely on the data used to train them. As a result, when a model is trained, it can only provide meaningful strength estimates for passwords that are distributed similarly to the ones used to train it [46]. More specifically, regardless of the size of the training set, if a password model is trained to be optimal for a given password distribution, it cannot also be optimal for a different distribution.

The toy example in Figure 2 provides an illustration of this phenomenon. In this experiment, three instances of a password model [43] are trained on three different training sets: a collection of Polish password leaks (1,434,802 passwords in total), a collection of Hungarian password leaks (1,215,767 passwords in total), and the combination of the two. When the Hungarian and Polish models are used to carry out a guessing attack on a Hungarian and a Polish credential database[5] (Figure 2a and Figure 2b, respectively), these models

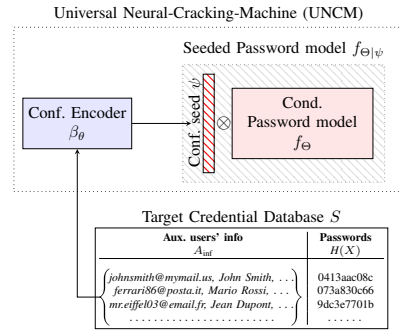[5]Which are disjointed from their respective training collection.

demonstrate optimal performance on their respective matching database, but their performance is suboptimal when applied to each other's database. This mutual exclusivity phenomenon is further demonstrated by the results obtained by third model, which is trained on the combined Hungarian and Polish password lists. While this model is trained on twice as much data than the previous two, it performs worse than the models trained specifically for the Hungarian and Polish distributions on both leaks. However, it does achieve better performance on average over the two password sets.

Ultimately, the mutual exclusivity of password models entails that no single model can achieve optimal performance across all distributions. Although one can attain an optimal performance **on average** by approximating the universal password distribution (i.e., training the model on the combination of all known password distributions), no model can meet the optimal thresholds for all password distributions simultaneously, at least with current implementation methods. In this work, however, we show that universal password models can exist and can be implemented through novel deep learning techniques. We call this family of models **Universal Neural-Cracking-Machines**, or UNCMs for short, and describe them in the following.

B. A Universal Neural-Cracking-Machine (UNCM)

It should be clear from the previous section that there is no single guessing strategy that is optimal for every guessing attack. Therefore, by definition, **a universal password model must be adaptive**; that is, it must conditionally change its guessing strategy based on the context. In a nutshell, a UNCM is a deep neural network built and trained to implement this functionality. Given a target credential database S, a UNCM can be understood as an "*adaptive password model*" that is able to automatically optimize its guessing strategy based on S, even when the passwords in S are unknown at inference time. Nonetheless, for this to work, the model must either have observed the optimal guessing strategy for S as part of the training data or be able to derive it by interpolating similar known distributions.

The general structure of a UNCM is reported in Figure 3; it is the combination of two main modules—two deep neural

networks.

a) The Conditional Password model: The first module is something that we can define as a **conditional password model**. Pragmatically speaking, this is just a probabilistic password model whose mass function can be altered by an additional input ψ that is disjointed from the set of parameters learned during the training phase. Within the paper, we call this additional input: **configuration** or **configuration seed**. The main idea here is that, while the parameters Θ of the password model f_Θ are trained to capture multiple password distributions simultaneously (i.e., all the password distributions), the input configuration seed can instruct f_Θ to focus on specific modalities of the modeled distribution without acting on the parameters Θ, i.e., without retraining the underlying model.

Given a target credential database $S=(A_{\inf},H(X))$, the role of a UNCM is straightforward. The objective is to find a suitable configuration seed such that the conditional password model performs optimally on the (unknown) passwords in X. Finding such a configuration is the role of the second module—the **configuration encoder**.

b) The Configuration Encoder - auxiliary data as a proxy for password distributions: The configuration encoder β_θ is another deep neural network (with parameters θ) whose task is to generate a suitable configuration ψ for the conditional password model f_Θ, conditionally to S. In essence, the configuration seed generated by the encoder should be understood as a succinct representation of the target password distribution that the conditional password model can leverage to maximize the allocated mass on the target passwords X. Ideally, the task of the configuration encoder is to find ψ such that:

$$\beta_\theta(X) = \arg\max_\psi f_\Theta(X \mid \psi). \tag{2}$$

In other words, the objective of β_θ is to define a prior probability distribution over the passwords X for the given credential database S. However, there is a fundamental problem: the passwords X and the specific password distribution from which X is sampled are entirely unknown e.g., the passwords are hashed or completely inaccessible.

As mentioned earlier, the configuration encoder circumvents the issue of obtaining direct knowledge of the password distribution X by using personal information of users as a proxy. Indeed, while obtaining any part of the plaintext passwords X may not be feasible, it is always possible to derive insights into their distribution using other information available in the target credential database. Password hashes are always accompanied by **auxiliary** data such as email addresses, usernames, and related tags. As demonstrated in previous studies on targeted guessing attacks [14], [64], [38], [48], [17], this auxiliary data and users' passwords are not independent. Users often incorporate personal information into their passwords, and an attacker with knowledge of a user's personal information can launch a targeted attack and guess the user's password more efficiently.

The configuration encoder takes this correlation one step further by aiming to **capture general properties of the pass-word distribution associated with the entire community of users in a target system**. Rather than trying to model the correlation between an individual user and their chosen password [14], [64], [38], [48], [17], the configuration encoder receives the auxiliary data of all users simultaneously and correlates these individual pieces of information to extract general patterns that can be used to model the password distribution for the whole community.

For example, returning to the toy example in Section III-A, if most of the email addresses in the target credential database have the domain "*.pl*" or/and usernames built on Polish words, the encoder should detect this general trend and instruct the password model to generate "*Polish passwords*" as output. When given Hungarian-related auxiliary data, the same model should focus on "*Hungarian passwords*" instead. In addition to simple and pre-defined characteristics like the language of the community, the configuration encoder is trained to identify and generalize these patterns (which may be latent) from the available data. At inference time, it applies them to the input auxiliary information $A_{\inf}$ to define an accurate *priori* over the unknown passwords in X, which the conditional password model then uses. Essentially, we can see a UNCM as a "*password model factory*" - an algorithm that takes the auxiliary information available for the target community as input and produces a password model tailored to the underlying password distribution in a fully autonomous way. In Section IV, we discuss how a UNCM is implemented and trained for this purpose.

1) Deploying a UNCM - seeded password models: Once the UNCM has been trained and made public[6], end-users (e.g., system administrators) can use it to generate accurate password models for the target distribution without any additional dataset collection or training step.

Formally, let *Bob* be the system administrator of a system $\mathcal{S}$ (e.g., a web application or a multi-user system) with a credential database $S=(A_{\inf},H(X))$. *Bob* wants to deploy an accurate password model for $\mathcal{S}$. To this end, *Bob* **(1)** downloads the pre-trained UNCM and **(2)** uses the auxiliary information $A_{\inf}$ associated with the users in $\mathcal{S}$ as input for the UNCM. This process results in a **seeded password model** $f_{\Theta|\psi}$, which is the combination of the pre-trained conditional password model f_Θ and the configuration seed ψ specifically generated for the system $\mathcal{S}$ by the configuration encoder. As demonstrated in Section V, the seeded model is an accurate representation of $\mathcal{S}$'s password distribution and may even outperform equivalent password models that have been manually configured for the target. Once the seeded password model for $\mathcal{S}$ has been generated, it can be used like a standard password model, such as for deploying a client-side password strength meter or conducting a guessing attack on $H(X)$.

In summary, starting from a pre-trained UNCM, *Bob* can automatically generate a tailored, state-of-the-art password model for his system. This process is fast (see Section IV-D),

---

[6]Following the "*foundation models*" paradigm in modern ML [10], the UNCM is pre-trained and made public by a third party e.g., our research group or an external company.

TABLE I: Number of leaks for the top-25 top-level domains appearing in *Cit0day.in* before and after pre-processing.

Raw # leaks	*.com*	*.ru*	*.net*	*.org*	*.de*	*.kr*	*.br*	*.pl*	*.uk*	*.it*	*.fr*	*.cz*	*.tw*	*.in*	*.hu*	*.jp*	*.info*	*.ca*	*.th*	*.eu*	*.nl*	*.ua*	*.au*	*.es*	*.ar*
	9539	1208	1167	982	874	798	609	551	533	467	438	400	330	322	305	226	223	193	191	175	172	163	142	139	134
Clean # leaks	*.com*	*.ru*	*.net*	*.org*	*.de*	*.kr*	*.pl*	*.br*	*.uk*	*.it*	*.cz*	*.fr*	*.hu*	*.tw*	*.in*	*.jp*	*.fj*	*.info*	*.ca*	*.eu*	*.ua*	*.ar*	*.au*	*.nl*	*.ro*
	4741	567	560	464	426	403	389	299	286	238	210	200	192	175	172	112	102	102	99	91	79	79	72	72	71

requiring only users' auxiliary information (no external password leaks collection), and can be implemented with a single API call. *Bob* does not need to share users' auxiliary data or password hashes with any external parties. Additionally, the configuration step only needs to be performed once, although it can be periodically updated to accommodate an evolving user community. In the differentially private setting (discussed in Section VI-A), updates to the configuration seed must be carefully considered, as they will increase privacy loss.

IV. ON THE IMPLEMENTATION OF A UNCM

In this section, we describe the relevant technical details about the implementation of our UNCM. We start by introducing the training set we use to develop and validate our approach. Then, we cover the architecture of the configuration encoder, and the one of the conditional password model we use to build the UNCM. Finally, we discuss how we train the UNCM and the baseline models we compare with.

A. The training set (Cit0Day.in)

To train and validate the proposed model, we rely on the credential leak collection originated by *Cit0day.in* [8], [31]. *Cit0day.in* was a service aimed at collecting leaked credential databases and provisioning them to malicious actors for a premium. In November 2020, *Cit0day.in* suffered a security incident that resulted in the leakage of more than $22,500$ (previously breached) credential databases stored by the service. The originated data dump was shortly made publicly available after the incident. Hereafter, we refer to this leak collection as Cit0day. The complete list of the web applications appearing in Cit0day is publicly available at [1].

Unlike similar leak collections harnessed in previous works (e.g., the one identified by *4iQ* in 2017 [32] and used in [48], [51]), Cit0day stores the compromised credentials partitioned by source web-application rather than merging them in a single set.

The collection covers a heterogeneous set of leaks with broad geographic coverage. Table I (raw) reports the most frequent top domains associated with websites appearing in the collection. Not all the password leaks reported in Cit0day are in plaintext.[7] A non-negligible fraction of credentials appears in hashed form or other noisy formats. After an extensive cleaning process (which is detailed in Appendix E), the number of suitable leaks was reduced to a total of $11,922$. We report the updated distribution of top domains in Table I (clean). The cleaned version of Cit0day contains

a total of $120,521,803$ compromised accounts with an average cardinality per leak (i.e., number of users) of $10,088$. We report the exact leaks' cardinality distribution in Figure 14 in the appendix.

Available auxiliary data: For every web application in Cit0day, credentials are reported in the format *email address* and *password*; no other personally identifiable information (PII) associated with users appears in the leaks. Ultimately, this fundamentally limits the amount of auxiliary information we can provide to the configuration encoder. Indeed, hereafter, we consider the email addresses as the only modality of auxiliary data available for the configuration seed computation. We stress that, in web applications, email addresses are always associated with users' passwords; thus, this setting represents the minimal and most general setup for the application of our model. Nonetheless, we forecast that UNCM's performance should be improved by providing additional auxiliary data when available.

Pre-processing: Given the cleaned leak collection, we partition it uniformly at random into two disjoint subsets (90-10%): a training set composed of 10054 leaks (for a total of 102 million accounts) and a test set composed of 1766 leaks (for a total of 11 million accounts). Additionally, we removed all the users who appeared in the training leaks from the test leaks. Hereafter, we refer to these two sub-collection of leaks as $\mathbf{L}_{\text{train}}$ and $\mathbf{L}_{\text{test}}$, respectively. In the paper, the training set $\mathbf{L}_{\text{train}}$ is used to train the proposed UNCM as well as the baselines we compare with. The test portion $\mathbf{L}_{\text{test}}$ is used to simulate the target credential databases during the evaluation phase.

B. The Configuration Encoder

We now start discussing the implementation of the configuration encoder β_θ used to build the UNCM. The general structure of β_θ is depicted in Figure 4. It comprises two main logical components: a **Sub-encoder** η and a **Mixing-encoder** γ.

1) The Sub-encoder: The role of the sub-encoder η is to encode all the pieces of auxiliary information associated with a single user and map them to a standard format that can be interpreted by the mixing-encoder (i.e., a dense vector of fixed size d). In the current setup, the sub-encoder is designed to process only users' email addresses. However, this can be extended to handle arbitrary auxiliary data modalities as described in Appendix VII-D.

Given an input email address (e.g., "*johnsmith@mymail.us*"), the sub-encoder partitions it in three segments: **(1)** the username ("*johnsmith*"), **(2)** the provider ("*@mymail*"), and **(3)** the domain ("*.us*"). These segments are then processed separately, relying on three sub-models.

---

[7]Plaintext passwords in Cit0day are likely to be the ones that have been recovered via offline guessing attacks or that were stored in plaintext on the source credential database.
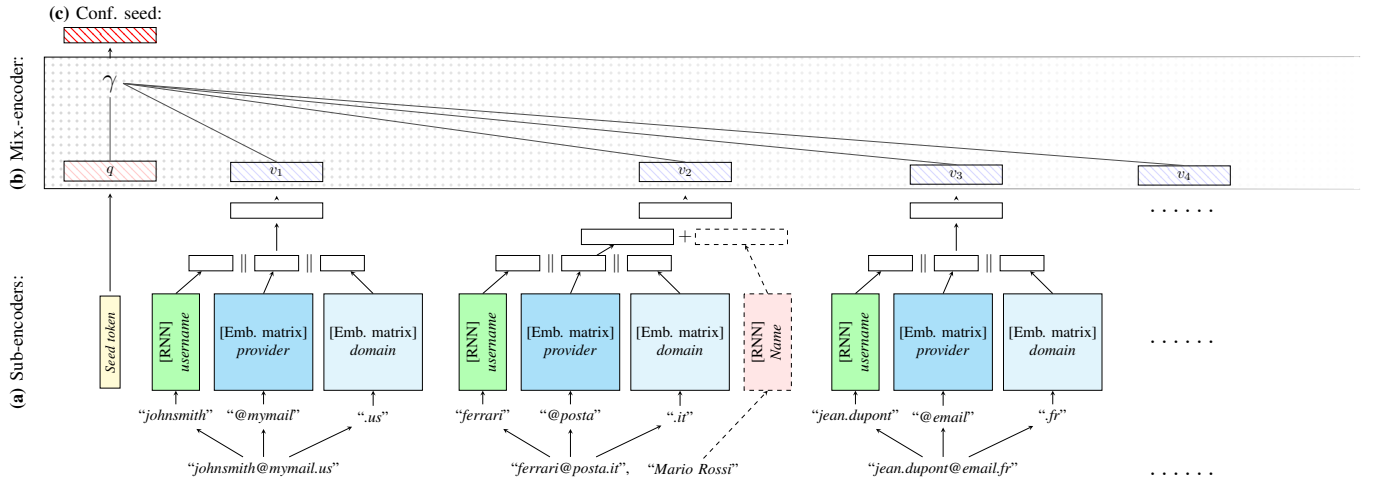
Fig. 4: Graphical representation of the **configuration encoder**. Panel **(a)**: Sub-encoder. The sub-encoder is replicated for each available input. Panel **(b)**: Oversimplified version of the mixing-encoder. Panel **(c)**: The resulting configuration seed.

A small RNN handles the username; a GRU in our implementation—the green rectangle at the bottom of Figure 4.[8] This sub-model iterates over the username string at the character level and returns the output obtained on the terminal character of the string. Instead, the provider and the domain strings are processed differently. Those are first discretized and then embedded via two separated embedding matrices (the two cyan squares in Figure 4). Each provider and domain string is associated with a row in the corresponding embedding matrix, which value is learned during the encoder's training process. To restrain the size of the matrices in our implementation, we exclude providers and domains that appear with low frequency. Specifically, we apply a frequency cutoff threshold of 300, as this results in appropriately sized embedding matrices (2974 and 268 for providers and domains, respectively). Provider and domain strings that fall outside this range are mapped to a special *out-of-the-vocabulary* embedding.

Finally, to create the embedding for the complete email address, the vectors produced by each sub-model (username, provider, and domain) are concatenated together, forming a single vector of size d as depicted in Figure 4.

2) The Mixing-encoder: The objective of the mixing-encoder is to *mix* the outputs produced by the sub-encoder (one for each user in the target credential database) and condense them in a single dense vector—the configuration seed. More pragmatically, the mixing encoder is another deep neural network trained to interpolate over the provided (projected) auxiliary data and distill out the information that the password model can exploit to predict the password distribution of the underlying community.

Given that the auxiliary input information does not follow any particular ordering and may drastically vary in size depending on the target credential database, a natural choice

to implement the mixing encoder is to rely on an attention mechanism (Section II-B). In our setup, we implement it with a single AM layer of the form described in Section II-B. Here, every output produced by a (replica) of a sub-encoder is considered a value vector v_i, whereas there is a single query vector q. The latter is an additional embedding vector learned during the training (the yellow rectangle in Figure 4). After the training, this vector is used as a constant input for the model.[9]

The main reason behind using a single attention mechanism is that this allows an efficient application of differential privacy, as we cover in Section VI-A.

3) Configuration seed computation: Provided the set of auxiliary information A_{inf} (i.e., email addresses in the current setup), the sub-encoder η and the mix. encoder γ are used to compute the configuration seed ψ. This process is formalized in Algorithm 1.

Credential databases can come with millions of users, and computing the seed using the whole population may result computationally impractical. Thus, rather than providing the entire set A_{inf} to the configuration encoder, we sample a uniform set of k users' auxiliary information $\tilde{\tilde{A}}_{\text{inf}}$ from A_{inf}

[9]Essentially, this is equivalent to the special `[CLS]` vector used in BERT-like models [21].

[8]In the deployment phase, we noticed that utilizing a more complex architecture such as a LSTM did not yield any added advantages. Consequently, we decided to employ a simpler configuration, such as GRU for efficiency.

Algorithm 1: Seeded password model from UNCM.

Data: UNCM, Auxiliary information: A_{inf}
Result: Seeded password model: $f_{\Theta|\psi}$
$(\eta,\ (\gamma,\ q)),\ f_\Theta = \text{UNCM}$;
$\tilde{\tilde{A}}_{\text{inf}} \sim_k A_{\text{inf}}$;   #Sample at most k users
$V = \{\}$;
for $a_{inf} \in \tilde{\tilde{A}}_{inf}$ **do**
   | $v_i = \eta(a_{\text{inf}})$;   #Apply Sub. enc.
   | $V = V \cup \{v_i\}$;
end
$\psi = \gamma(\{q\}, V)$;   #Apply Mix. enc.
$f_{\Theta|\psi} = f_\Theta \otimes \psi$;   #Init. p. model with ψ
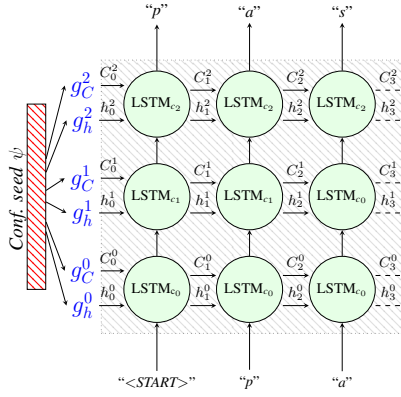
Fig. 5: Simplified representation of the **seeded password model** $f_{\Theta|\psi}$ running on the partial input "*password*". We do not depict the final dense prediction layers.

which is then used as input for the encoders instead. We tested different values for k; surprisingly, increasing it beyond 2048 does not result in meaningful utility gains for the average case. Thus, eventually, we opt for $k=8192$ in the standard setup (as this value still provides better performance in some edge cases) and $k=2048$ for the private UNCM so to better capitalize on the effect of *privacy amplification by subsampling* (more on Section VI-A).

As reported in Algorithm 1, once sampled the input subset of auxiliary information, the sub encoder η is applied on every entry in $\tilde{\tilde{A}}_{\text{inf}}$, resulting in a set V of at most k vectors. This set is then provided to the mix. encoder γ as the set of input values aside with the query vector q (see Section IV-B2). Regardless of the number of users k, the mix. encoder results in a single output vector of size d; the configuration seed ψ, with $d=756$ in our setup. We tested configuration seeds with dimensionality higher than 756 achieving no meaningful improvements. Regardless of its apparent complexity, the configuration encoder can produce a configuration seed with a sub-second latency. When $k=8192$, our unoptimized implementation, on average, requires 0.65 *sec* on a GPU and 0.97 *sec* on CPU on a NVIDIA DGX-2. It is worth noting that the configuration seed is generated only once before the seeded model is deployed, and a slower seed generation would not significantly affect the overall performance of the framework.

Finally, the produced configuration seed can be applied to the conditional password model. Details on this final step are provided in the next section.

C. The Conditional Password Model

In order to implement the conditional password model f_{Θ} for the UNCM, we extend the password model proposed by Melicher et al. [43]. This is a standard, non-conditional, autoregressive password model implemented using a plain 3-layers LSTM network. The model has been shown to outperform most existing password models, in addition to providing the practical advantages described in Section II-A2. More importantly, the model enables for heavy compression, and

it can be used as a client-side password strength meter with limited loss in utility [43].

In order to convert this model into a conditional password model (see Section III-B), we need to modify it to support the configuration seed ψ as an additional input. Inspired by the classic NLP encoder-decoder paradigm for recurrent networks [55], we achieve this by providing ψ as the initial state of the LSTM cells. As depicted in Figure 5, given the vector ψ, we apply two different transformations (one for the hidden state and one for the cell state) for each LSTM layer of the model. In our setup, those transformations are implemented using a pair of different dense layers $(g_h^i,\ g_C^i)$ for each layer i. The resulting $3\cdot2=6$ vectors are then used as initial states for the 3-layer LSTM. We refer to this initialization process with the symbol $f_{\Theta} \otimes \psi$ (last line of Algorithm 1). The model resulting from this operation (i.e., the password model with the custom initialized states as depicted in Figure 5) is what we call a seeded password model $f_{\Theta|\psi}$.

The initialization via the configuration seed is the only modification to the original model [43], and both its functionality and internal workings remain unchanged. In the original model, the LSTM's states would be initialized to all zeros. Therefore, **the two models are virtually identical** and the seeded password models produced by the UNCM can be compressed using the same techniques used for the original model [43]. Similarly, a seeded model has the same computational complexity as the original model [43]. Note that the dense layers g_h^i and g_C^i do not need to be included in the seeded password model at the deployment phase. Once the configuration seed is generated by β_{θ}, the outputs of those layers (i.e., $g_h^0(\psi),\ g_C^0(\psi),\ldots g_C^2(\psi)$) can be pre-computed and statically shipped with the model at the configuration phase: $f_{\Theta} \otimes \psi$. In total, without any form of compression, these layers account for only 24KB, so they do not hinder the use of the model as a client-side PSM.

D. Training a UNCM

All existing password models [47], [43], [29], [52], [51] are trained at the *password granularity*. Given a set of plaintext passwords X (i.e., the union of one or more password leaks), at each training round, a single or a batch of passwords is sampled from X and used to tune the parameters of the model; typically, via maximum likelihood estimation. A UNCM, instead, is the first model trained at ***password-leak granularity***; that is, the atomic training instance for the model is a whole password leak rather than a single password. To be trained, therefore, a UNCM requires a collection of credential databases. Hereafter, we use the training leak collection $\mathbf{L}_{\text{train}}$ from Cit0day presented in Section IV-A.

The password-leak-based training process is defined as follows. At each training step, **(1)** we sample a credential database $S=(A_{\text{inf}}, X)$ from the leak collection $\mathbf{L}_{\text{train}}$. **(2)** Following Algorithm 1, a subset of auxiliary information $\tilde{\tilde{A}}_{\text{inf}}$ is given as input to the configuration encoder β_{θ} that produces a configuration seed ψ as output. **(3)** The configuration seed ψ is then applied to the password model f_{Θ} that, in turn, is trained

to assign a high probability to **all** the passwords in X [43], completing a single training iteration. In the process, **the configuration encoder (i.e., the mixing-encoder and the sub-encoder) and the password model are jointly trained as a single network** to maximize the performance of the password model in the conditional password generation task. This objective is summarized by the following optimization problem:

$$\underset{\theta,\,\Theta}{\arg\max}\left\{\mathbb{E}_{(A_{\mathrm{inf}},X)\sim\mathbf{L}_{\mathrm{train}}}\sum_{x\in X}\frac{\mathrm{xen}(x,\,f_{\Theta|\beta_\theta(A_{\mathrm{inf}})}(x))}{|X|}\right\},\quad(3)$$

where xen is cross entropy defined at the character level, quantifying the discrepancy between the prediction of the seeded password model and the ground-truth password x.[10] Intuitively, we are training the configuration encoder to retrieve suitable information from the auxiliary data and shift the mass function of f_Θ towards the passwords in X. At the same time, we train the password model f_Θ to generate the passwords X following the instructions provided by the configuration seed. At the next training round, the models are challenged with another credential database, forcing them to generalize over to auxiliary data and password space, respectively.

A meta-learning interpretation: Another pragmatic interpretation is that a UNCM is just a meta-ML model [30] trained to produce as output a password model $f_{\Theta|\beta_\theta(A_{\mathrm{inf}})}$ such that the probability $f_{\Theta|\beta_\theta(A_{\mathrm{inf}})}(X)$ is maximized for a set of passwords X defined at inference time. The core difference with standard meta-learning approaches (few-shot learning, mainly [69]) is that, instead of directly providing a set of support examples to define the target task (this would be a subsample of passwords from X in our setting), the meta-model gets only a noisy proxy signal as input (the user auxiliary information).

It is important to understand that, in Eq 3, loss values computed on every $x \in X$ are accumulated in a single error signal that is then used to jointly train both the encoder and the password model within a single SGD iteration. In our implementation, we rely on minibatch-SGD. Thus, the loss is computed over multiple leaks at each training iteration. To reduce memory consumption, we rely on gradient accumulation; we compute the gradient signal sequentially for each leak in the mini-batch. Then, we average and apply them to the models' parameters. As discussed in Section IV-B3, we sample only a subsample of at most k users to compute the seed. This means that, at most, in a single iteration, we replicate the sub-encoder 8192 times, run the attention mechanism on a value set of 8192 vectors, and compute probabilities for 8192 passwords with the password model. Nonetheless, despite the size of the models and the large sample size, we managed to train the proposed UNCM using a single *NVIDIA V100 GPU* with 32GB of memory thanks to gradient-accumulation.

We train the model until the loss on the test set stops improving (i.e., early stopping). We use *Adam* as optimizer [36]

---

[10]This loss is computed via *teacher forcing* for the RNN [68].

with the default setting, and we implement the code in *TensorFlow2*.

E. Baseline: Approx. of the Universal Distribution

Within the paper, we use a baseline model to empirically demonstrate the advantages provided by the proposed UNCM's models over previous methodologies. This is a reimplementation of the state-of-the-art password model proposed by Melicher et al. [43]. We use the same architecture used for the password model in the UNCM (see Section II-A). However, we train it following the standard training process considered in [43] that can be summarized by the following optimization problem:

$$\underset{\Theta}{\arg\max}\left\{\mathbb{E}_{x\sim\mathbf{X}}\left[\mathrm{xen}(\,x,\,f_\Theta(x)\,)\right]\right\}.\qquad(4)$$

To train this baseline model, we use the same training and validation set used for the UNCM. In Eq. 4, the training set $\mathbf{X}$ is the union of all the leaks in $\mathbf{L}_{\mathrm{train}}$ i.e., $\bigcup_{\{A_{\mathrm{inf}},X\}\in\mathbf{L}_{\mathrm{train}}}X$. All the hyper-parameters are kept the same as for the UNCM.

Eventually, given the heterogeneity of Cit0day, this model can be understood as a suitable approximation of the **universal password distribution** (see Section III-A). That is, an unconditional model optimized to perform well on the average case. Hereafter, we refer to this model as $f_{\dot\Theta}$. Pragmatically speaking, fixed the training data, $f_{\dot\Theta}$ represents the best *off-the-shelf* model that system administrators can use for their systems when there is no possibility of producing tailored password models. Details on the training process and architectures are reported in Appendix C.

V. EVALUATION

Next, we empirically validate the capabilities of the password models produced by the UNCM. We first compare them to the baseline. Then, we show that the seeded models produced by the pre-trained UNCM can outperform password models that have been manually configured and trained for the target password distribution.

A. UNCMs vs. Universal approximation

We now compare the pre-trained UNCM with the baseline model $f_{\dot\Theta}$ i.e., the approximation of the universal password distribution (see Section IV-E).

a) Evaluation process: Given the pre-trained UNCM, for each leak in $S=(A_{\mathrm{inf}},X)\in\mathbf{L}_{\mathrm{test}}$ (where A_{inf} are just the users' email addresses), we run Algorithm 1 and generate a seeded password model $f_{\Theta|\psi}$ for S. Then, we use $f_{\Theta|\psi}$ and $f_{\dot\Theta}$ to guess the unknown passwords in X. In the process, we use the Monte Carlo method proposed by Dell'Amico et al. [19] to efficiently estimate the guess numbers for both models. Details on the hyper-parameters are given in Appendix C.

b) Results: Figure 6 reports the result of the guessing attack for 5 example leaks in $\mathbf{L}_{\mathrm{test}}$. The performance gain obtained by the seeded password model (green) over the baseline (black) is consistent but heterogeneous. As we discuss bellow, seeded models guess passwords faster than the baseline within the initial guesses. In the general case, this advantage
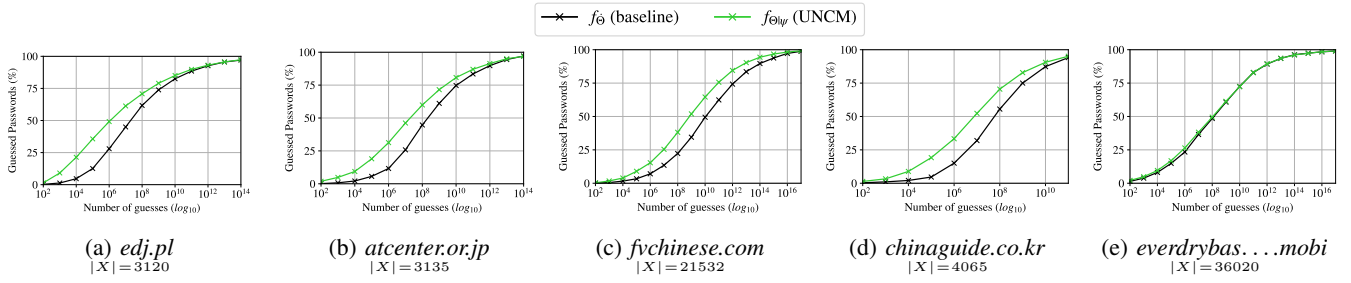
Fig. 6: Comparison between the seeded password model and baseline on guessing attacks for 5 example password leaks sampled from $\mathbf{L}_{\text{test}}$. In the caption of each plot, we report the size of the tested password leak aside from the leak identifier.
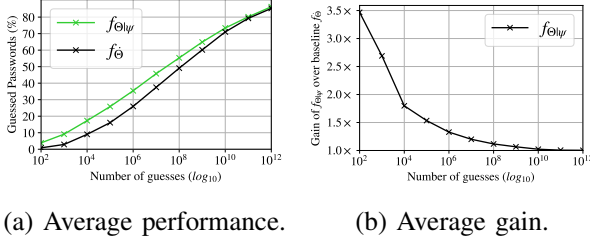

(a) Average performance.   (b) Average gain.

Fig. 7: Panel (a) reports the **average** performance of the seeded password models and the baseline on guessing attacks. Panel (b) reports the number of guessed passwords of $f_{\Theta|\psi}$ over the baseline during the attacks. For instance, the coordinates $(10^2, 3.4)$ indicate that, on average, the seeded password models guessed 3.4 times the number of passwords guessed by the baseline within 10^2 guesses.

TABLE II: Examples of weak passwords overestimated by the universal distribution approximation (guess numbers).

okmedicina.it	pippicalzelunghe	ascolipiceno	nonsaprei1	baiaimperiale	piazzacairoli
$f_{\dot{\Theta}}$	$2 \cdot 10^{18}$	$4 \cdot 10^{12}$	$2 \cdot 10^{12}$	$5 \cdot 10^{13}$	$3 \cdot 10^{13}$
$f_{\Theta|\psi}$	$4 \cdot 10^{9}$	$2 \cdot 10^{6}$	$3 \cdot 10^{6}$	$2 \cdot 10^{9}$	$9 \cdot 10^{9}$
weryfikatorium.pl	rzeczpospolita	dywizjon303	kurkawodna3	samarytanin1	kotwbutach28
$f_{\dot{\Theta}}$	$6 \cdot 10^{15}$	$1 \cdot 10^{15}$	$8 \cdot 10^{12}$	$2 \cdot 10^{13}$	$1 \cdot 10^{15}$
$f_{\Theta|\psi}$	$6 \cdot 10^{4}$	$1 \cdot 10^{8}$	$9 \cdot 10^{5}$	$3 \cdot 10^{6}$	$4 \cdot 10^{8}$
atcenter.jp	kannkoku415	78hokuhoku	yutarikari7	honbadakara	taikyokuken
$f_{\dot{\Theta}}$	$9 \cdot 10^{12}$	$2 \cdot 10^{12}$	$3 \cdot 10^{12}$	$9 \cdot 10^{12}$	$1 \cdot 10^{12}$
$f_{\Theta|\psi}$	$2 \cdot 10^{9}$	$7 \cdot 10^{8}$	$2 \cdot 10^{9}$	$6 \cdot 10^{9}$	$1 \cdot 10^{9}$
le_monde_en_enigmes.fr	mot de passe	ouarzazate	mvtmjsunp	10ruedesroches	Mousquetaires
$f_{\dot{\Theta}}$	$4 \cdot 10^{11}$	$1 \cdot 10^{11}$	$2 \cdot 10^{13}$	$2 \cdot 10^{14}$	$1 \cdot 10^{11}$
$f_{\Theta|\psi}$	$4 \cdot 10^{4}$	$2 \cdot 10^{4}$	$8 \cdot 10^{6}$	$1 \cdot 10^{8}$	$7 \cdot 10^{5}$
seminarky.cz	ptakopysk	faktnevim123	Vysokaskola1	vyjebanec1	milujulukase
$f_{\dot{\Theta}}$	$1 \cdot 10^{12}$	$4 \cdot 10^{14}$	$2 \cdot 10^{13}$	$9 \cdot 10^{12}$	$5 \cdot 10^{12}$
$f_{\Theta|\psi}$	$3 \cdot 10^{4}$	$1 \cdot 10^{8}$	$1 \cdot 10^{7}$	$5 \cdot 10^{7}$	$3 \cdot 10^{7}$
shop.santool.de	Maschinenbau	entwicklung	freyssinet99	Modellbau+1	baumwolle5
$f_{\dot{\Theta}}$	$1 \cdot 10^{12}$	$3 \cdot 10^{12}$	$1 \cdot 10^{14}$	$3 \cdot 10^{13}$	$3 \cdot 10^{12}$
$f_{\Theta|\psi}$	$4 \cdot 10^{4}$	$4 \cdot 10^{6}$	$1 \cdot 10^{9}$	$4 \cdot 10^{8}$	$1 \cdot 10^{8}$
diarioelsur.cl	jehovaesmipastor123	teamarexsiempre	serviciomovil	toyotires13	estudiojuridico
$f_{\dot{\Theta}}$	$3 \cdot 10^{15}$	$3 \cdot 10^{14}$	$1 \cdot 10^{13}$	$4 \cdot 10^{12}$	$6 \cdot 10^{12}$
$f_{\Theta|\psi}$	$1 \cdot 10^{9}$	$3 \cdot 10^{9}$	$1 \cdot 10^{8}$	$2 \cdot 10^{8}$	$3 \cdot 10^{8}$
leilaoonlinerural.br	fellipeleao1	harasoficial	MeuArquivo	damiaomatos	comunica?ao
$f_{\dot{\Theta}}$	$1 \cdot 10^{14}$	$2 \cdot 10^{14}$	$2 \cdot 10^{14}$	$4 \cdot 10^{12}$	$2 \cdot 10^{12}$
$f_{\Theta|\psi}$	$2 \cdot 10^{9}$	$6 \cdot 10^{9}$	$8 \cdot 10^{9}$	$2 \cdot 10^{8}$	$3 \cdot 10^{8}$

tends to decrease as the attack progresses. In other cases, this gain remains consistent even after 10^{12} guesses e.g., Figure 6c. A desirable property of the UNCM is that it will never perform worse than the baseline. Indeed, when no suitable information can be inferred from A_{inf}, it seems that the UNCM automatically falls back to model an approximation of the universal password distribution, performing on par with the baseline. Figure 6e shows an example of this behavior.

Regardless of these limit cases, the UNCM performs better than the baseline with a substantial margin on average. This is shown in Figure 7a, where we report the guessing performance of the seeded and baseline models averaged over 100 different leaks uniformly sampled from $\mathbf{L}_{\text{test}}$. Such a performance gain can be better understood in Figure 7b, where we plot the average ratio of guessed passwords for the two models during the attacks (i.e., $\frac{\#\text{passwords guessed by} f_{\Theta|\psi}}{\#\text{passwords guessed by} f_{\dot{\Theta}}}$). Within the initial 10^2 guesses, the UNCM guesses about 3.4 times (340%) more passwords than the universal approximation. Then, this gain gently decreases with the number of guesses. Additional results are provided in Appendices A and D-A, where we compare with other standard password models and dynamic dictionary attacks [51], respectively.

Remark: It is important to emphasize that the passwords models $f_{\Theta|\psi}$ and $f_{\dot{\Theta}}$ have an equal number of parameters

[11] and they have been trained on the same training data. The only difference is that the seeded models have access to the configuration seed computed over a subset of users' auxiliary information (only email addresses in the current setup). Additionally, recall that we removed all the users appearing in the training set from the test leaks in $\mathbf{L}_{\text{test}}$; thus, we can exclude that the UNCM could memorize users' email addresses and simulate credential stuffing/tweaking attacks [48].

1) Evidence of adaptation to the target distribution: Recall that a password strength meter's main objective is to detect and prevent users from picking easily guessable passwords. Thus, correctly identifying passwords with low-guess numbers is the most valuable property of a meter [28]. The results above showcase the capability of the UNCM to detect weak passwords that are missed by the universal approximation model. In this section, we provide further intuition on the nature of these mislabeled weak passwords.

Table II reports examples of weak passwords mislabeled by the baseline $f_{\dot{\Theta}}$ for 8 leaks with different top domains in $\mathbf{L}_{\text{test}}$. For each leak, we report the first 5 passwords such that the difference between the guess number attributed by $f_{\dot{\Theta}}$ and the one attributed by $f_{\Theta|\psi}$ is maximal; formally,

---

[11] For the UNCM, the configuration encoder is not used during the guessing attack; only the seeded password model is employed.
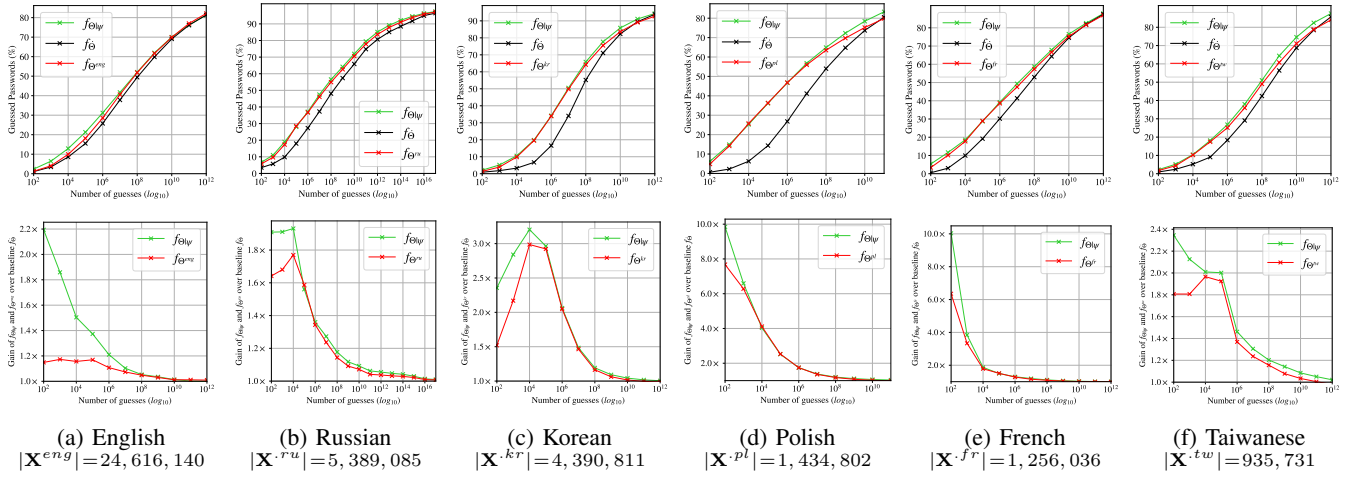
(a) English	(b) Russian	(c) Korean	(d) Polish	(e) French	(f) Taiwanese
$|\mathbf{X}^{eng}| = 24,616,140$	$|\mathbf{X}^{\cdot ru}| = 5,389,085$	$|\mathbf{X}^{\cdot kr}| = 4,390,811$	$|\mathbf{X}^{\cdot pl}| = 1,434,802$	$|\mathbf{X}^{\cdot fr}| = 1,256,036$	$|\mathbf{X}^{\cdot tw}| = 935,731$

Fig. 8: For 6 top domains `lan`, each plot in the first row compares the average guessing performance of three password models (baseline, tailored model for `lan`, and seeded password model generated by UNCM) computed over all the leaks with top domain `lan` in $\mathbf{L}_{\text{test}}$. The second row reports the relative ratio of the number of guessed passwords by $f_{\Theta^{\text{lan}}}$ and $f_{\Theta|\psi}$ over the baseline. In the caption of each column, we report the cardinality of the leak collection used to train the tailored model $f_{\Theta^{\text{lan}}}$.

$top5(\frac{f_{\dot\Theta}(x)}{f_{\Theta|\psi}(x)})$. All the passwords reported in the table are $x\in X$
considered relatively secure by the baseline (i.e., the model assigns high guess numbers to them). However, as can be seen, most of those passwords are combinations of common nouns for the underlying community's language that, arguably, would be easily guessed by any realistic attack e.g., a dictionary attack based on the source language's vocabulary.

As discussed in Section III-B, the UNCM is able to automatically adapt to the target password distribution by relying on users' auxiliary information. This enables the seeded model to correctly estimate the guessability of those peculiar weak passwords even if they lie in underrepresented modalities of the training set. To provide an intuitive example, the password "*pippicalzelunghe*"[12] (first row) requires 10^{18} guesses for the baseline (i.e., computationally unguessable) while the UNCM demonstrates that this password is weak and can be guessed within the first 10^9 guesses even if *Italian* passwords only account for 1.2% of the whole Cit0day. The majority of the passwords listed in the table illustrate the same phenomenon but for different communities.

B. UNCM vs. Manual configuration

While we have shown that a pre-trained UNCM performs better than a password model trained to approximate the universal password distribution, it is necessary to ask if a UNCM can reach comparable performance to models that have been manually configured for the target system. As already discussed, given a target credential database S, one can seek an (*a priori*) optimal configuration for the password model by collecting and training on password leaks that share common characteristics with the target password distribution. In the

---

[12]Italian name for "*Pippi Longstocking*".

following experiment, we use the target community's language as the main characteristic to define password similarity. This is universally recognized as the most impactful feature in shaping password distributions [20].

a) Evaluation process: In the experiments, we consider 6 different languages: "*English*", "*Russian*", "*Korean*", "*Polish*", "*French*", and "*Taiwanese*" (listed in decreasing password count). To automate the collection of tailored training sets from Cit0day, we use the top domain of the leaked website to infer the language of the underlying community; for instance, if a website has top-domain ".*pl*", we assume that the community has *Polish* as the main language. For each top domain `lan`, evaluation follows as:

1) We collect all the leaks in $\mathbf{L}_{\text{train}}$ (training collection) and $\mathbf{L}_{\text{test}}$ (test collection) with top domain `lan`, creating two leak sub-collections $\mathbf{L}_{\text{train}}^{\text{lan}}$ and $\mathbf{L}_{\text{test}}^{\text{lan}}$, respectively. For the "*English*" passwords, we use a different filtering process which is described in Appendix C-B.

2) We train the baseline model [43] as described in Section IV-E but with the training set $\mathbf{X}^{\text{lan}} = \bigcup_{\mathbf{L}_{\text{train}}^{\text{lan}}} X$, obtaining a model $f_{\Theta^{\text{lan}}}$ i.e., a tailored password model for the passwords with language `lan`.

3) Then, we evaluate $f_{\Theta^{\text{lan}}}$, $f_{\dot\Theta}$ and $f_{\Theta|\psi}$ on the collection of leaks $\mathbf{L}_{\text{test}}^{\text{lan}}$ (password leaks from communities with language `lan`). The evaluation follows the general procedure used in the previous setting.

b) Results: Results are summarized in Figure 8, where the number of guesses is averaged over all the leaks in $\mathbf{L}_{\text{test}}^{\text{lan}}$ for any `lan`, respectively. As expected, both the seeded password models $f_{\Theta|\psi}$ and the manually tailored ones $f_{\Theta^{\text{lan}}}$ outperform the baseline. However, a valuable result is that not only do the seeded models match the performance of the manually configured ones, but they also manage to perform

slightly better. In particular, the seeded models always guess more passwords correctly on the initial part of the attack and occasionally achieve higher performance also on high-guess-number passwords, as shown by the Korean, Polish, and Taiwanese cases. This suggests that the prior learned by the UNCM from the users' email addresses can be more informative than the language prior exploited by the manually configured models $f_{\Theta^{\text{lan}}}$. Another decisive factor is that, while the UNCM adapts to the target distribution, it can still exploit the knowledge offered by the whole training set $\mathbf{L}_{\text{train}}$. This additional knowledge can help the model to guess strong passwords that may be independent of the target distribution, while also improving its ability to generalize during the training process. We empirically confirmed this intuition by training a UNCM exclusively on $\mathbf{X}^{eng}$ and compare it against f_{Θ}^{eng}. In this data constrained setting, the seeded model did not provide any significant advantage over the tailored model. Nonetheless, we stress that the additional knowledge offered by non-target distributions cannot be capitalized by standard password models as demonstrated by the low performance of the baseline model which has been trained on all available data.

The advantage of the seeded models seems to be inversely proportional to the size of the training set used to train $f_{\Theta^{\text{lan}}}$. Intuitively, the less data is available to train a tailored model, the larger is the potential performance gain achievable by the UNCM. In the results, the only exception is on the weakest passwords of the English setting (i.e., up to 10^6 guesses), where seeded models outperform the tailored one regardless of the size of the curated training set. We attribute this phenomenon to the inherent internal diversity of this distribution. Indeed, in contrast to the other settings in the figure, this is obtained as the aggregation of multiple sub-modalities (e.g., English, American, Australian, etc.) which the UNCM can still potentially leverage for adaptation. However, it appears that, in this particular setting, the UNCM does not provide any advantage beyond the 10^8 mark.

In Appendix B, we extend our analysis to other semantic factors such as the service provided by the web application (e.g., Shopping, Education, Entertainment, etc.), achieving congruent results with what observed on the language-specific setting.

In conclusion, a pre-trained UNCM is capable of achieving equivalent if no better performance than equivalent state-of-the-art password models, even if those are specifically configured for the target password distribution. To achieve this result, end-users do not need to manually retrieve tailored training sets or undertake any additional training phases; starting from the pre-trained model, a UNCM only requires access to users' email address to automatically generate a password model configured for the target system.

VI. PRIVATE UNCMs

As discussed in Section III-B1, *Bob* can download the pre-trained UNCM and configure it for the target distribution by executing Algorithm 1. The obtained seeded password model $f_{\Theta|\psi}$ can be then deployed as client-side PSM, for instance, by embedding it in the sign-up web page of $\mathcal{S}$. However, by releasing the model $f_{\Theta|\psi}$, *Bob* may introduce privacy risk for the users whose information has been used to forge the seed ψ. Indeed, having access to the seeded password model, and, so, to ψ, a *privacy-attacker* would be potentially able to infer information on $\widetilde{\widetilde{A}}_{\text{inf}}$. For instance, the adversary may be able to run membership inference attacks on the seed and infer whether a particular email address has been used to compute it, and, so, whether the user has an account in $\mathcal{S}$. While, for the current setup, the introduced privacy risk is marginal[13], the concern becomes relevant in the case other users' PII is available and used to forge the configuration seed. Ultimately, even if the configuration seed is a highly compressed representation of the auxiliary input data, it is not possible to exclude that it would leak arbitrary information on the web application's users. Next, we show that it is possible to **efficiently achieve formal user-level privacy guarantees** by designing a configuration encoder with differentially-private output. To this end, we introduce a differentially private attention mechanism that may have further applications in other contexts.

Remark: It's important to note that the use of a private UNCM is only necessary when the seeded model is made public, such as in the case of a PSM. In other standard settings, like reactive guessing attacks [35], *Bob* can use the original construction described in Section III-B without further modifications, even when sensitive data is used for the configuration step.

A. Differentially-private configuration seeds

Threat model: Formally, we model an adversary $\mathcal{A}$ who aims at inferring information on the set U of users used to forge the configuration seed ψ i.e., the users associated to $\widetilde{\widetilde{A}}_{\text{inf}}$. Following the differential privacy model, $\mathcal{A}$ is a strong adversary who has arbitrary information on the distribution of users and unbounded computational capabilities.

Given $\mathcal{A}$, we are challenged to produce a perturbed configuration seed $\widetilde{\psi}$ such that the adversary has a bounded advantage in inferring information about the users. We achieve this by making the seed differentially private (see Section VI-A). We stress that the training set originally used to fit the UNCM is public (e.g., *Cit0day* used in the paper) and it does not require to be protected from the adversary $\mathcal{A}$.[14] However, if the training set is assumed to be private, standard DP-SGD [2] can be used during the training process of the UNCM as for any other password model.

1) A Differentially-Private Attention-Mechanism: In order to achieve differentially-private configuration seeds, we need to act on the configuration encoder. As depicted in Figure 4, the information $\widetilde{\widetilde{A}}_{\text{inf}}$ we need to protect is individually projected through the sub-encoder, **becoming the set of value**

---

[13]Attackers can typically learn if an email address is in $\mathcal{S}$ in more efficient and reliable ways [56].

[14]Under the easily enforceable assumption that $\mathcal{S}$ and the training set are disjointed.
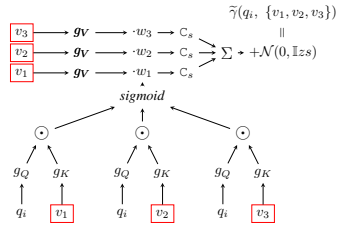
Fig. 9: Depiction of the introduced differentially-private attention-mechanism (DP-AM). In the computational graph, private inputs are highlighted in red.

(a) Privacy budget. (b) Average performance.

Fig. 10: Panel (a) reports the value ϵ (y axes) for every leak in $\mathbf{L}_{\text{test}}$ plotted against the leak size (x axes). Panel (b) is the average guessing performance of the seeded models, private seeded models and baseline on 100 leaks from $\mathbf{L}_{\text{test}}$.

vectors for the mixing encoder—an attention mechanism (AM). Thus, if we make the AM differentially private w.r.t. the value vectors, we obtain a differentially private configuration seed. We highlight that, unlike the typical ML setting, **we want our model's outputs to be differentially private and not its parameters.**

Luckily, the AM described in Section II-B offers natural support to the application of DP. This becomes evident once we rename the weighted projection of the i'th value vector as $v_i^w = w_i \cdot g_V(v_i)$. Then, the output of the attention mechanism can be rewritten as: $\gamma = \sum_{i=1}^n v_i^w$, which is a standard setting for the application of DP, e.g., [2], [42].

However, there is still a problem; the output of the *softmax* used to compute the attention weights w_i is a function of all the value vectors as it normalizes every dot product $d_i = g_Q(q) \cdot g_K(v_i)^T$ over their sum: $w_i = \frac{e^{d_i}}{\sum_j e^{d_j}}$. This indirectly induces a correlation among the private vectors before the application of the noise, preventing us from achieving instance-level-DP (where "*instance*" is the single value vector). To sidestep this issue, we replace the *softmax* with an element-wise *sigmoid* function. Similarly to the *softmax*, the *sigmoid* function maps the output of every dot product in the interval $[0, 1]$ independently, making every attention weight w_i a function of the sole corresponding value vector v_i.[15]

Once we have independent attention weights, making the final sum (ϵ, δ)-DP reduces to bound the sensitivity of γ and then adds calibrated Gaussian noise to its output. In particular, we bound the sensitivity of the weighted values by clipping the L_2 norm of each vector v_i^w to a chosen value s via the clipping function: $\mathtt{C}_s(v_i^w) = \frac{v_i^w}{max(1, \frac{\|v_i^w\|}{s})}$ [2].

Then, Gaussian noise proportional to s is applied to the sum of the clipped values. Putting all together, the output of the differentially private attention mechanism is defined as:

$$\widetilde{\gamma} = \sum_{i=1}^n [\mathtt{C}_s(v_i^w)] + \mathcal{N}(0, \mathbb{I}zs), \tag{5}$$

where z is the noise multiplier—a second hyper-parameter that controls the amount of applied noise, and n is the number of value vectors. The resulting DP-AM is depicted in Figure 9.

---

[15]However, those cannot be properly called "*attention weights*" as their sum can be different from 1.
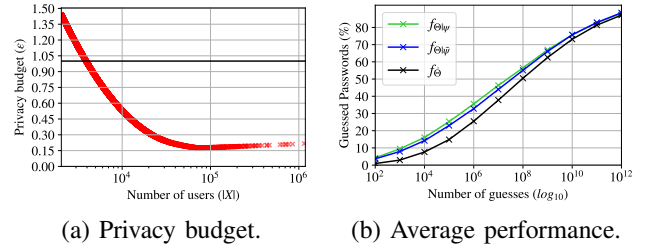
The attention-mechanism $\widetilde{\gamma}$ provides instance-level-DP for the value vectors. **As every value vector represents auxiliary information for a single user, instance-level-DP implies user-level privacy guarantees.** Here, the only requirement is that no user should be associated with more than a value vector; that is, no user should have multiple accounts on the same service.

Returning to *Bob*, he can now download the private version of the UNCM and obtain a private seeded model by just replacing γ with $\widetilde{\gamma}$ in Algorithm 1. Hereafter, we refer to a differentially-private configuration seed as $\widetilde{\psi}$ and to a DP-seeded model as $f_{\Theta|\widetilde{\psi}}$.

B. Privacy budget and Utility loss

Here, we quantify the privacy level as well as the utility loss induced by the differential private configuration encoder. In our setup, we seek to attain strong privacy guarantees for the users' auxiliary information i.e., $\epsilon \leq 1$ [24]. In this regard, we set the noise multiplier to $z = 3$, while computing δ individually for each tested credential database X as $\delta = \frac{10^{-2}}{|X|}$. We also evaluate stricter privacy guarantees by considering smaller δ values: $\frac{10^{-3}}{|X|}$ and $\frac{10^{-4}}{|X|}$.

a) Privacy budget: Figure 10a reports the value of ϵ associated with the differential private configuration seed computed for every leak in $\mathbf{L}_{\text{test}}$. Recall that the configuration seed $\widetilde{\psi}$ is computed only on a uniformly sampled subset of A_{inf} of size at most k. Thus, the ϵ associated with the configuration seed of a credential database S is a function of the credential database size (i.e., the number of users) due to *privacy amplification by sub-sampling* [34], [6]. Privacy amplification by subsampling tells us that we can scale our privacy budget ϵ proportionally to the ratio between the size of the used input subset and the whole population A_{inf}: $\frac{k}{|A_{\text{inf}}|}$. As shown in Figure 10a, on average, we achieve $\epsilon = 0.76$. In the worst-case scenario (i.e., for the smallest leaks), we achieve $\epsilon = 1.44$. While 1.44 still provides meaningful privacy guarantees (see Appendix F), in these cases, the noise multiplier z can be increased at inference time to push ϵ under 1 (at the cost of additional utility loss). Results for smaller values of δ are reported in Figure 12.

b) Utility: Surprisingly, while achieving a remarkably small ϵ (on average), the utility loss induced by the DP-seeds

and the sigmoid-based attention mechanism is limited. This is shown in Figure 10b where the average guessing performance of the non-private seeded password models (green) and the ones with differential private configuration seeds (blue) is reported. While demonstrating a slight performance degradation compared to the non-private model, the DP-seeded models still maintain a consistent advantage over the baseline observed for the non-private setup. We attribute the success of the private model (low ϵ value at low utility loss) to two main factors: (1) The privacy gain obtained through amplification via sub-sampling. (2) Informally, the configuration seed is only required to model a general description of the target password distribution; therefore, intuitively, the impact of the single user on the final seed is limited. That is, we can still model the core distributional properties of the target community even if information about individuals is destroyed.

In order to substantiate the validity of our differentially private attention mechanism and its implementation, we conducted an empirical evaluation utilizing Membership Inference Attacks (MIA), as recommended practice for the introduction of novel differential-privacy-based mechanisms [57]. Results and details on the setup are given in Appendix F.

In conclusion, we demonstrated that a seeded password model can be efficiently made private with a limited loss in utility. This property paves the way for future deployments of UNCMs that exploit a broader spectrum of auxiliary input information without threatening users' privacy.

VII. FUTURE WORK AND UNCMs EXTENSIONS

In this section, we provide a brief overview of some potential extensions of the UNCM framework, which could serve as a foundation for future investigations.

A. Dynamic UNCMs

As described in Appendix D-A, dynamic password guessing techniques such as those presented in [52], [51] may not be directly applicable to password strength metering. Nonetheless, these techniques offer a useful means of constructing more precise adversary models. Fortunately, integrating dynamic attacks into the UNCM framework can be achieved through a relatively straightforward process. The subsequent section provides a high-level overview of the implementation steps involved in creating a Dynamic UNCM.

In essence, during an attack with a seeded password model, the intermediate set of guessed passwords $X_{\leq t}$ can serve as supplementary auxiliary information to generate an updated configuration seed at each time step t. For each password $x \in X_{\leq t}$, it is sufficient to concatenate a projected version of x with the encoded email address of the associated user, similar to the approach illustrated for the user's "*name*" in Figure 4. As the attack progresses, the configuration seed can be gradually refined by recomputing it at regular intervals using the accumulated guessed passwords up to that point. In a similar fashion, the seed can also be bootstrapped using prior knowledge of users' passwords, such as leveraging *sister passwords* obtained from previously breached services.
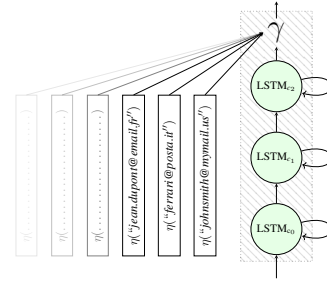

Fig. 11: Simplified depiction of a *Unbounded* UNCM. Rectangles on the left represent the outputs of the sub encoder η for all the provided auxiliary information.

Nonetheless, there are certain technical challenges that arise in deployment of dynamic autoregressive password models, such as adapting the enumeration (e.g., beam search) to operate on dynamic probability distributions in a efficient way. We defer this research avenue to future work.

B. Unbounded UNCMs (UNCMs with unlimited attention span)

In the form presented in the paper, a UNCM condenses the available auxiliary information on the target set in a single, fixed-size, vector via the mixing encoder. This choice offers some fundamental advantages. Mainly, it allows the deployment of compact seeded password models and efficiently ensures differential privacy. However, this also limits the amount of information that can be extracted from the auxiliary data, and, ultimately, exploited, by the UNCM during guessing attack. Nonetheless, this is not a fundamental requirement of a UNCM and, at the cost of sacrificing performance and applicability, one can implement a UNCM with no auxiliary information bottleneck. Next, we briefly sketch this model which we dub "*Unbounded*" UNCMs.

The idea is that, instead of training and using the mix. encoder to compress auxiliary information in a single vector, we allow the password model to access individual pieces of auxiliary information during the inference process. Figure 11 sketches this architecture. In particular, as in the standard setting (see Section IV), we use the sub encoder η to parse all the auxiliary information of users and project them in a set of equal-sized vectors. Then, a "*Bahdanau-like*" attention mechanism [3] is used to dynamically bias the password model's behavior at inference time. More concretely, all the encoded auxiliary information are value vectors, while the query is dynamically generated as the output of the last layer of the LSTM. Upon every character generation step, the password model can attend to all the encoded pieces of auxiliary information and dynamically focus on / retrieve the required knowledge from them.

We anticipate that enabling the conditional password model to attend to all auxiliary information and dynamically directing its attention based on the generated password will enhance the model's effectiveness in guessing attacks. However, this improvement would be paid in terms of efficiency, as inference

would now have overhead linear in the number of auxiliary pieces of information. Moreover, the model would be too cumbersome to run in a browser, thereby limiting its applicability. Nonetheless, the model could prove useful for obtaining more accurate strength estimates in reactive studies.

C. Different Password Models

In our study, we focused on using an LSTM network to implement the conditional password model of the UNCM. This choice was based on the model's proven effectiveness in estimating password strength and its familiarity within the password security community. However, it is worth noting that the password model architecture could potentially be replaced by any other neural network. For instance, one can rely on transformer-based or similar novel constructions [51]. All that is required is to attach the new password model to the configuration encoder at training time and define a suitable way to provide the configuration seed as additional input to it.

D. Additional auxiliary data modalities

Although the current implementation only utilizes email addresses, the sub-encoder can be extended to process various types of additional input data. In these cases, it is enough to introduce additional sub-models suited to handle the new auxiliary data modalities. In Figure 4, we offer the example of the name of the user (the second replica of the sub-encoder). Similarly to the username of the email address, this string can be parsed by introducing an additional RNN in the sub-encoder.

The only requirement here is that all the information associated with a user should be mapped to a single vector. That is, the output of any new sub-model should not be used as direct input for the mixing-encoder, but, as shown in the figure, it should be combined/merged (e.g., via element-wise addition) with the output of the email-encoder for the associated email address. This requirement offers two main advantages: (1) it allows the mixing-encoder to model the (possible) semantic relation between the different sources of input data provided and (2) it drastically simplifies achieving differential privacy for the configuration seed as we discuss in Section VI-A.

VIII. ETHICAL CONSIDERATIONS

In our study, we utilized a collection of previously leaked credentials to develop and validate the effectiveness of our password model. The use of this data raises ethical concerns. However, the information we used is already publicly available online, and the compromised accounts have already been added to breach alert systems [31].

For the model training and evaluation, we only considered passwords that appeared in plaintext in the original leak and did not attempt to recover additional credentials stored as hashes. In the same direction, we did not augment the data in Cit0day by combining it with other public sources of information or preprocessing it in a way that would provide additional knowledge to an adversary. Therefore, our analysis

does not exacerbate or extend the harm already caused to users by the leakage.

Although adversaries could use our techniques to improve attacks performance, the introduced models enable security practitioners to automatically generate accurate password models for their systems, regardless of their lack of expertise and/or resources. By doing so, our approach will contribute to stronger password security practices and ultimately help protect users from cyber threats. Given that using leaks is the only way of carrying out this research, we believe that this benefit outweighs the possible risk introduced by our study.

IX. CONCLUSION

We present the first self-configurable password model that uses auxiliary data to adapt to the target password distribution at inference time. This addresses a major problem in the application of password security techniques in the real world. We make our pre-trained models public, making accurate password models more widely accessible.

Our framework is general and can be extended to benefit from future improvements in password models and deep learning techniques, as well as data availability, without compromising user privacy (see Section VI). Our work also demonstrates the potential of using auxiliary data to improve the performance of password guessing models in the classic trawling offline setting, paving the way for further research in this field.

REFERENCES

[1] Cit0day breach, list of sites. https://gist.github.com/gvolluz/dd0df2ba2400c4891f95d05de3dde1da, 2020. Accessed: 2022-08-09.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[5] Wenjie Bai and Jeremiah Blocki. Dahash: Distribution aware tuning of password hashing costs. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security*, pages 382–405, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg.

[6] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. *Machine Learning*, 94(3):401–437, 2014.

[7] David Biesner, Kostadin Cvejoski, and Rafet Sifa. Combining variational autoencoders and transformer language models for improved password generation. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ARES '22, New York, NY, USA, 2022. Association for Computing Machinery.

[8] Alina Bizga. Data breach saga: What you need to know about the cit0day data leak. https://www.bitdefender.com/blog/hotforsecurity/data-breach-saga-what-you-need-to-know-about-the-cit0day-data-leak, 2020. Accessed: 2022-08-09.

[9] Jeremiah Blocki, Benjamin Harsha, and Samson Zhou. On the economics of offline password cracking. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 853–871, 2018.

[10] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[11] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552, 2012.

[12] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy*, pages 553–567, 2012.

[13] Chuck Brooks. Alarming cyber statistics for mid-year 2022 that you need to know. https://www.forbes.com/sites/chuckbrooks/2022/06/03/alarming-cyber-statistics-for-mid-year-2022-that-you-need-to-know/?sh=225d79f7864a, 2022. Accessed: 2023-01-01.

[14] Claude Castelluccia, Abdelberi Chaabane, Markus Dürmuth, and Daniele Perito. When privacy meets security: Leveraging personal information for password cracking. *arXiv preprint arXiv:1304.6584*, 2013.

[15] Rahul Chatterjee, Joseph Bonneau, Ari Juels, and Thomas Ristenpart. Cracking-resistant password vaults using natural language encoders. In *2015 IEEE Symposium on Security and Privacy*, pages 481–498, 2015.

[16] Mitchell Clark. Hackers stole encrypted lastpass password vaults, and we're just now hearing about it. https://www.theverge.com/2022/12/22/23523322/lastpass-data-breach-cloud-encrypted-password-vault-hackers, 2022. Accessed: 2022-12-23.

[17] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and Xiaofeng Wang. The tangled web of password reuse. 01 2014.

[18] Xavier de Carné de Carnavalet and Mohammad Mannan. From very weak to very strong: Analyzing password-strength meters. In *Network and Distributed System Security Symposium (NDSS 2014)*. Internet Society, 2014.

[19] Matteo Dell'Amico and Maurizio Filippone. Monte carlo strength evaluation: Fast and reliable password checking. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 158–169, New York, NY, USA, 2015. Association for Computing Machinery.

[20] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. Password strength: An empirical analysis. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, 2010.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[23] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[24] Cynthia Dwork and Adam Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2), Apr. 2010.

[25] Serge Egelman, Andreas Sotirakopoulos, Ildar Muslukhov, Konstantin Beznosov, and Cormac Herley. Does my password go up to eleven? the impact of password meters on password selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 2379–2388, New York, NY, USA, 2013. Association for Computing Machinery.

[26] Brown et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[27] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 657–666, New York, NY, USA, 2007. Association for Computing Machinery.

[28] Maximilian Golla and Markus Dürmuth. On the accuracy of password strength meters. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1567–1582, New York, NY, USA, 2018. Association for Computing Machinery.

[29] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. Passgan: A deep learning approach for password guessing. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 217–237, Cham, 2019. Springer International Publishing.

[30] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

[31] Troy Hunt. Inside the cit0day breach collection. https://www.troyhunt.com/inside-the-cit0day-breach-collection/, 2020. Accessed: 2022-08-09.

[32] Luke Irwin. 1.4 billion user credentials found on the dark web. https://www.itgovernanceusa.com/blog/1-4-billion-user-credentials-list-found-in-the-dark-web, 2017. Accessed: 2022-08-09.

[33] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[34] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[35] Patrick Gage Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Julio Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *2012 IEEE Symposium on Security and Privacy*, pages 523–537, 2012.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[37] Hang Li, Mengqi Chen, Shengbo Yan, Chunfu Jia, and Zhaohui Li. Password guessing via neural language modeling. In *Machine Learning for Cyber Security: Second International Conference, ML4CS 2019, Xi'an, China, September 19-21, 2019, Proceedings*, page 78–93, Berlin, Heidelberg, 2019. Springer-Verlag.

[38] Yue Li, Haining Wang, and Kun Sun. Personal information in passwords and its security implications. *IEEE Transactions on Information Forensics and Security*, 12(10):2320–2333, 2017.

[39] Enze Liu, Amanda Nakanishi, Maximilian Golla, David Cash, and Blase Ur. Reasoning analytically about password-cracking software. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 380–397, 2019.

[40] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. A study of probabilistic password models. In *2014 IEEE Symposium on Security and Privacy*, pages 689–704, 2014.

[41] Jerry Ma, Weining Yang, Min Luo, and Ninghui Li. A study of probabilistic password models. In *2014 IEEE Symposium on Security and Privacy*, pages 689–704, 2014.

[42] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[43] William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Fast, lean, and accurate: Modeling password guessability using neural networks. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 175–191, Austin, TX, August 2016. USENIX Association.

[44] modzero. Better make sure your password manager is secure or someone else will. https://www.modzero.com/modlog/archives/2022/12/19/better_make_sure_your_password_manager_is_secure/index.html, 2022. Accessed: 2022-12-22.

[45] R. Morris and K. Thompson. Password security: A case history. *Commun. ACM*, 22(11):594–597, November 1979.

[46] Hazel Murray and David Malone. Exploring the impact of password dataset distribution on guessing. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–8, 2018.

[47] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, page 364–372, New York, NY, USA, 2005. Association for Computing Machinery.

[48] Bijeeta Pal, Tal Daniel, Rahul Chatterjee, and Thomas Ristenpart. Beyond credential stuffing: Password similarity models using neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 417–434, 2019.

[49] Bijeeta Pal, Mazharul Islam, Marina Sanusi Bohuk, Nick Sullivan, Luke Valenta, Tara Whalen, Christopher Wood, Thomas Ristenpart, and Rahul Chatterjee. Might i get pwned: A second generation compromised credential checking service. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1831–1848, Boston, MA, August 2022. USENIX Association.

[50] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Interpretable probabilistic password strength meters via deep learning. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve Schneider, editors, *Computer Security – ESORICS 2020*, pages 502–522, Cham, 2020. Springer International Publishing.

[51] Dario Pasquini, Marco Cianfriglia, Giuseppe Ateniese, and Massimo Bernaschi. Reducing bias in modeling real-world password strength via deep learning and dynamic dictionaries. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 821–838. USENIX Association, August 2021.

[52] Dario Pasquini, Ankit Gangwal, Giuseppe Ateniese, Massimo Bernaschi, and Mauro Conti. Improving password guessing via representation learning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1382–1399. IEEE, 2021.

[53] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. Let's go in for a closer look: Observing passwords in their natural habitat. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 295–310, New York, NY, USA, 2017. Association for Computing Machinery.

[54] Emma Roth. Lastpass' latest data breach exposed some customer information. https://www.theverge.com/2022/11/30/23486902/lastpass-hackers-customer-information-breach, 2022. Accessed: 2022-12-01.

[55] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[56] The OWASP Foundation. Testing for account enumeration and guessable user account. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/04-Testing_for_Account_Enumeration_and_Guessable_User_Account, 2022. Accessed: 2022-08-09.

[57] Florian Tramer, Andreas Terzis, Thomas Steinke, Shuang Song, Matthew Jagielski, and Nicholas Carlini. Debugging differential privacy: A case study for privacy auditing. *arXiv preprint arXiv:2202.12219*, 2022.

[58] Blase Ur, Felicia Alfieri, Maung Aung, Lujo Bauer, Nicolas Christin, Jessica Colnago, Lorrie Faith Cranor, Henry Dixon, Pardis Emami Naeini, Hana Habib, Noah Johnson, and William Melicher. Design and evaluation of a data-driven password meter. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 3775–3786, New York, NY, USA, 2017. Association for Computing Machinery.

[59] Blase Ur, Patrick Gage Kelley, Saranga Komanduri, Joel Lee, Michael Maass, Michelle L. Mazurek, Timothy Passaro, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. How does your password measure up? the effect of strength meters on password creation. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 65–80, Bellevue, WA, August 2012. USENIX Association.

[60] Blase Ur, Sean M. Segreti, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Saranga Komanduri, Darya Kurilova, Michelle L. Mazurek, William Melicher, and Richard Shay. Measuring Real-World accuracies and biases in modeling password guessability. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 463–481, Washington, D.C., August 2015. USENIX Association.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[62] Rafael Veras, Christopher Collins, and Julie Thorpe. On semantic patterns of passwords and their security impact. In *NDSS*, 2014.

[63] Jaikumar Vijayan. The state of authentication: Is a passwords replacement imminent? https://techbeacon.com/security/state-authentication-passwords-replacement-imminent, 2022. Accessed: 2023-01-01.

[64] Ding Wang, Zijian Zhang, Ping Wang, Jeff Yan, and Xinyi Huang. Targeted online password guessing: An underestimated threat. In
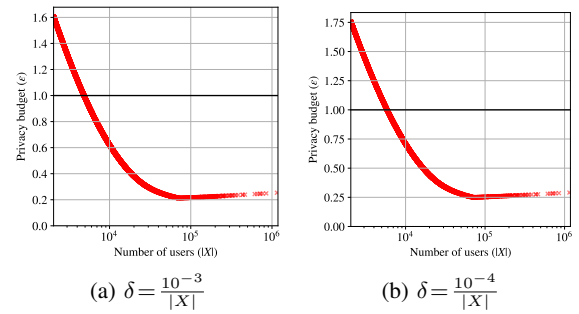
method [19] to estimate guess numbers. In particular, we compare with:

- **PCFG [66]:** We use the model with the default setting available in the implementation.[16]
- **Markov Chains [47], [41]:** We evaluate a total of 4 model variations. We train and test: a Markov chain of order 3 (MC_3), a Markov chain of order 4 (MC_4), a Markov chain of order 5 (MC_5), and a Markov chain with backoff [41] (MC_{boff}). All the hyper-parameters are set to their default values as provided with the implementation [19].[17]
- **semantic-PCFG [62]:** We use the model with the default setting available in the implementation.[18] Hereafter, we refer to this password model as semPCFG.
- **min-auto** We additionally report results for the ideal *min-auto* metric [60]. As defined by Us et al. [60], for every password x, *min-auto* is the minimum guess number across the pool of password models listed above:

$$\texttt{min}_{\texttt{auto}}(x) = \min[\texttt{PCFG}(x), \texttt{MC}_3(x), \texttt{MC}_4(x),$$
$$\texttt{MC}_5(x), \texttt{MC}_{bf}(x), \texttt{semPCFG}(x)], \quad (6)$$

where the model evocation returns the estimated guess number for a password x. Intuitively, this is the ideal/optimal combination of the tested models which provide a conservative estimation of password strength [60].

a) Results: Figure 13 reports the average guessing performance of the models over 100 leaks sampled from $\mathbf{L}_{\text{test}}$. We additionally report a close-up of their performance on 5 individual leaks in Figure 16. Consistently with the findings outlined in Section V-A, the seeded password models generated by the UNCM perform better than the other approaches. Most interestingly, the seeded models also achieves better performance than the ideal min-auto configuration (i.e., the combination of all the other models) on the initial guesses, while performing overall on par after the 10^8 mark. As discussed in Section V-B, the better performance on the initial guesses should be attributed to the capability of the UNCM of guessing weak passwords that are peculiar for the target distribution (e.g., language-specific passwords).

APPENDIX B
UNCM VS. MANUAL CONFIGURATION: EVALUATION ON OTHER SEMANTIC CATEGORIES

Here, we extend the analysis of Section V-B by including results on categorization based on users' account types. Indeed, it is well-established that the type of service for which a password is created can impact users' password choices [27]. In our analysis, we use the website category as an identifier, which refers to the type of service provided by the web application, such as shopping, education, entertainment, etc. Hereafter, we refer to it as "*website category*".

[16]https://github.com/lakiw/pcfg_cracker

[17]https://github.com/matteodellamico/montecarlopwd

[18]https://github.com/vialab/semantic-guesser

a) Setup: The password leaks in the Cit0day collection have been already labeled according website category by the original data curator. Although the exact method used to derive these labels is unknown[19], we have verified their accuracy through manual inspection of a batch of randomly chosen examples.

In the experiments, we consider the top 6 categories with the most passwords in Cit0day (across all the languages); those are: "*Shopping*" (Sh), "*IT*" (IT), "*Entertainment*" (En), "*Education*" (Ed), "*Sports*" (Sp), and "*Travel*" (Tr).[20] Then, we follow the procedure described in Section V-B: we train a separate password model [43] for each training set that contains leaked passwords from a specific category (e.g., only passwords from shopping websites), and test these models on a validation set with passwords from the same category.

b) Results: Results are reported in the first row of Figure 15. The seeded models exhibit superior performance compared to the manually tailored models, with a margin that surpasses that seen in language-specific distributions (Figure 8). This can be attributed to the fact that, as expected, the community language has a greater influence on shaping the password distribution than the website category.

To eliminate the influence of language on testing the UNCM's adaptability to service categories, we conduct the experiments again by fixing the language. Specifically, we filter the leaks by categories only in the English pool.[21] Also in this case, we considered the six categories with the highest number of passwords. In the English subset, these categories are the same as in the previous case, except for "*Travel*", which was replaced with "*Games*" (Gm). The results of these experiments are presented in the second row of Figure 15.

As per our hypothesis, the community language played a major role in the previous results, and, in this language-fixed setting, the difference in performance between the manually tailored models and the UNCM reduces. Nonetheless, the UNCM's generated models still outperform the tailored ones, supporting the argument made in Section V-B. However, it is important to note that the limited availability of training data for tailored models in such cases could be a significant factor that restricts their performance.

APPENDIX C
HYPER-PARAMETERS AND EVALUATION SETTING

Follow the list of hyper-parameters used for training and evaluation of the tested models:

- **Number LSTM cells** (f): 1000
- **Char. embedding size** (f): 512
- **(Virtual) batch size** (UNCM): 16
- **AM's projection width** (γ): 756
- **Emb. size provider** (η): 256
- **Emb. size domain** (η): 256

[19]For instance, it could have been obtained through an automatic tool or manually collected by the data curator.

[20]We have excluded the category "*Business*" from the plot as it is overly broad. Nonetheless, we evaluated this setting and the results are consistent with the other configurations.

[21]We used English leaks as those account for the highest number of passwords in total.
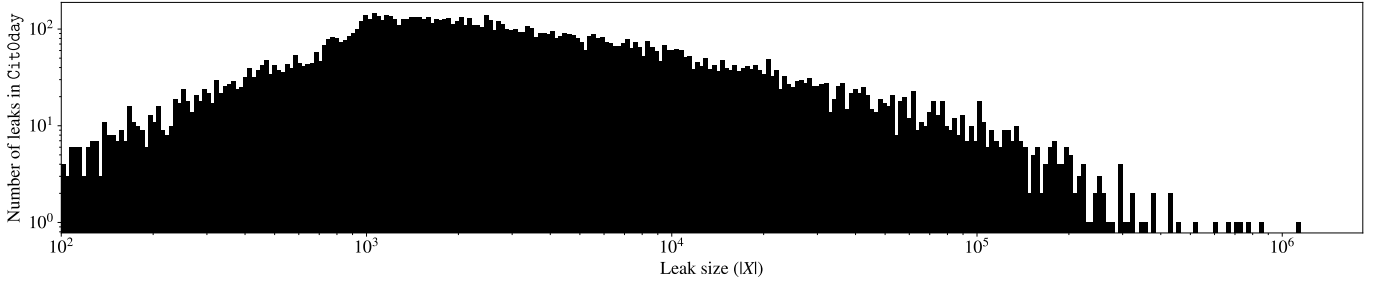
Fig. 14: Distribution of the cardinality (number of users) of the leaks in Cit0day.

(a) Shopping
$|\mathbf{X}^*_{Sh}| = 8,305,366$

(b) Entertain.
$|\mathbf{X}^*_{En}| = 6,915,205$

(c) IT.
$|\mathbf{X}^*_{IT}| = 6,564,410$

(d) Education
$|\mathbf{X}^*_{Ed}| = 6,168,965$

(e) Sports
$|\mathbf{X}^*_{Sp}| = 3,505,000$

(f) Travel
$|\mathbf{X}^*_{Tr}| = 2,542,684$

(g) Eng. Edu.
$|\mathbf{X}^{eng}_{Ed}| = 2,068,472$

(h) Eng. Shop.
$|\mathbf{X}^{eng}_{Sh}| = 1,961,553$

(i) Eng. IT
$|\mathbf{X}^{eng}_{IT}| = 1,855,932$

(j) Eng. Ent.
$|\mathbf{X}^{eng}_{En}| = 1,478,437$

(k) Eng. Games
$|\mathbf{X}^{eng}_{Gm}| = 1,721,813$

(l) Eng. Sports
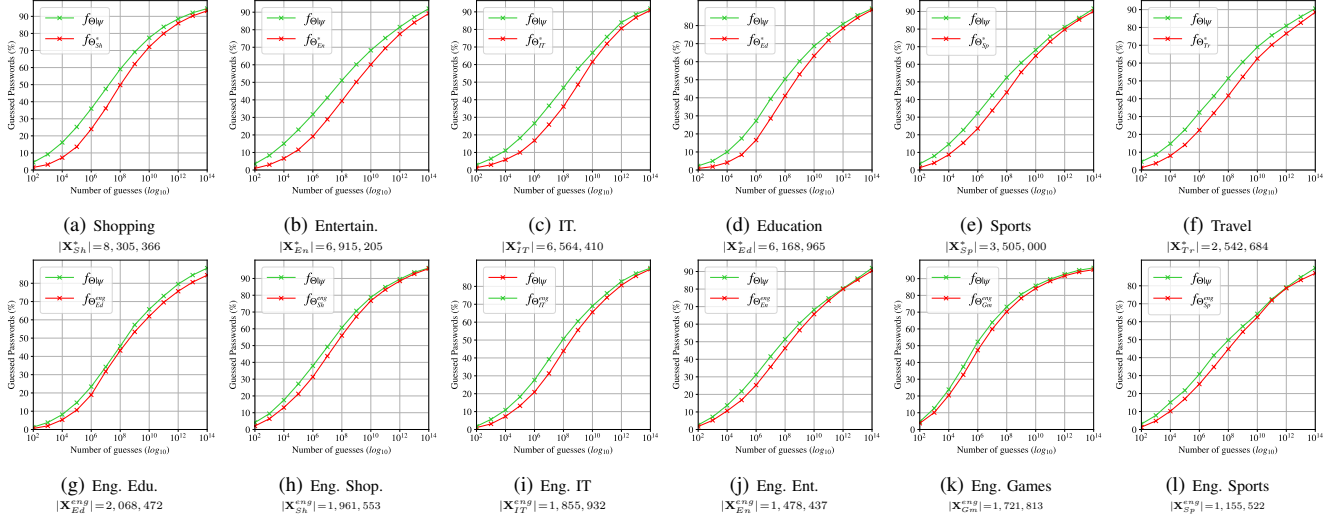$|\mathbf{X}^{eng}_{Sp}| = 1,155,522$

Fig. 15: Average guessing performance of seeded password models and manually tailored password models evaluated on 6 different semantic subsets of leaks (grouped by website categories). The caption of each plot reports the size of the training set used to fit the manually tailored models ($f_{\Theta|\psi}$ is the seeded model and $f_{\Theta^{\mathrm{lan}}_{cat}}$ indicates the tailored model).

- **Output size username GRU (η)**: 256
- **Char. embedding size username GRU (η)**: 16
- **Batch size $f_{\hat{\Theta}}$**: 64
- **Patience early stopping**: 5
- **Size Θ Monte Carlo guess number [19]**: 100,000

Values of ϵ reported in Section VI-B are computed using the implementation of [2] shipped with *TensorFlow Privacy*.

A. Setting toy-example Figure 2

To produce Figure 2, we use the same general procedure described in Section V-B. That is, we train the model of Melicher et al.[43] on the leaks with top-domain ".*pl*", ".*hu*", and ".*pl*||.*hu*" from $\mathbf{L}_{\mathrm{train}}$. The attack sets are respectively "*985ad8d0.pl*" and "*e1e222a8.hu*" from $\mathbf{L}_{\mathrm{test}}$.

B. Filtering english passwords

To identify password leaks of English speaking communities, we used a simple, yet, effective, heuristic. We started by considering all the leaks in Cit0day with top domains: {"*us*", "*uk*", "*au*", "*net*", "*org*", "*com*"}. Then, we removed all the leaks that had more than 2% of users' email addresses with a top domain different from {"*us*", "*uk*", "*au*", "*net*", "*org*", "*com*"}. The threshold "2%" is a parameter that we derived from a manually annotated subset of 50 examples. To validate the quality of the filtering process, we then, manually checked a random subsample of 50 leaks. All the inspected web applications associated to the leaks but one were targeted to an English-speaking user community.

We applied this process on both the training ($\mathbf{L}_{\mathrm{train}}$) and validation ($\mathbf{L}_{\mathrm{test}}$) leak collection, obtaining sub-collections of 1776 and 359 leaks, respectively.

APPENDIX D
ON DYNAMIC UNCMs

A. Comparison with Dynamic password models

Dynamic password models [52], [51] can be seen as another attempt to solve the universal password model problem. Those models, starting from a prior independent from the target password distribution, use the passwords that are guessed during the running attack as means to dynamic adapt the initial distribution to the target one. By design, this strategy requires to access plaintext passwords from the target set, making it applicable only in the guessing attacker setting. Indeed, in contrast with the UNCM, dynamic passwords models cannot be used as PSMs as no plaintext passwords are available besides the one tested by the meter.
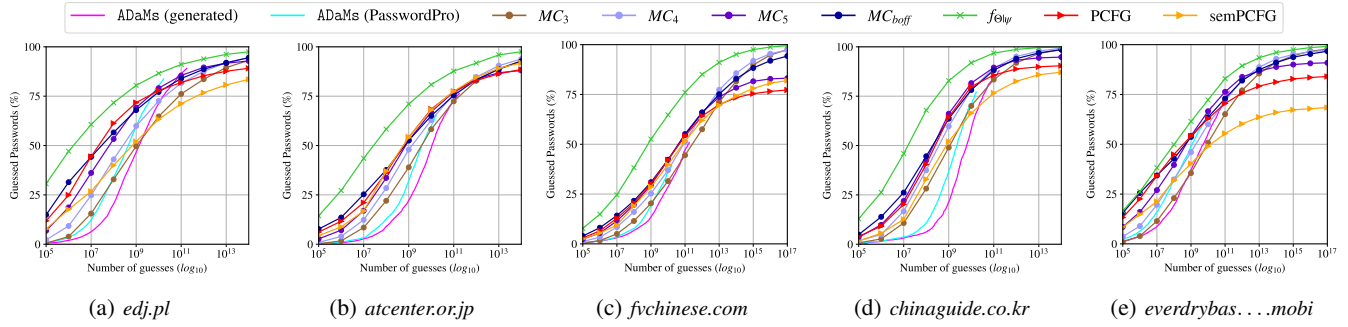
Fig. 16: Comparison between the seeded password models and *ADaMs* attack [51] (in addition to the models considered in Appendix A). The notation $f_{\Theta|\psi}$ refers to the seeded model generated by the UNCM.

For completeness, we report a comparison with the dynamic password model based on dictionary attack proposed in [51]. For this model, we use the two default models shipped with the official implementation: the adaptive models for the mangling rule-sets *PasswordPro* and *generated*. As input dictionary for the attacks, we use the union of all passwords in the leak collection $\mathbf{L}_{\text{train}}$. From the latter, we remove all the duplicate entries and sort the passwords by decreasing frequency order. We run the guessing attack on the example leaks considered in Section V-A. Results are reported in Figure 16. We also report results for the other models considered in Appendix A. The seeded password models outperform the dynamic dictionary attacks. Nonetheless, it is worth mentioning that the *ADaMs* attacks results in a consistently more efficient implementation that support real-world guessing attacks, while the model of Melicher [43] is known to have limited application in this setting [51]. Thus, we recognize these two approaches to be complementary.

APPENDIX E
DATA CLEANING

The cleaning procedure for the leak collection Cit0day encompasses different steps.

1) We built regular expressions for the most common hashes and removed all the accounts with passwords matching the regex(s).
2) We removed all the accounts that appear more than 150 times in the different leaks (to filter out bots).
3) After the previous steps, we removed all the leaks with less than 100 entries.
4) We train a password model [43] on a manually curated subset of (clean) leaks and used it as anomaly detector. We applied the model on every leak in Cit0day and removed the leaks with abnormal low-probability.
5) Some of the leaks in the collection were duplicated. We computed the intersection between each set of passwords and removed the leaks with intersection larger than 90%

Finally, a manual inspection on the test collection $\mathbf{L}_{\text{test}}$ has be carried out to remove abnormal leaks. The code for the cleaning pipeline as well as the list of leaks in *lct* and *lcv* is available in our repository.
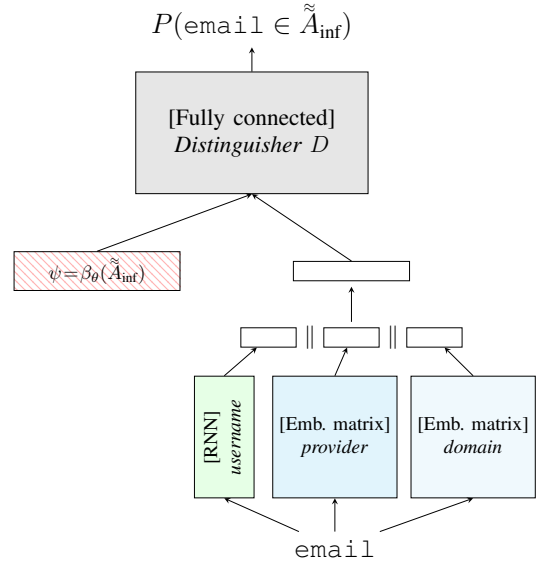

Fig. 17: Depiction of the attacker's model $\mathbf{A}$ trained to infer the membership of email addresses from a configuration seed ψ.

APPENDIX F
MEMBERSHIP INFERENCE ATTACKS ON ψ

In this section, we introduce a membership inference attack (MIA) targeting the email addresses used to generate the configuration seed. Then, we use the attack to empirically validate the soundness of the proposed *private* UNCM.

a) Setup: The setup for the attack involves a machine learning-based approach. Given a configuration seed $\psi = \beta_\theta(\tilde{\tilde{A}}_{\text{inf}})$ and an email address email, we train a neural network $\mathbf{A}$ to infer whether $\text{email} \in \tilde{\tilde{A}}_{\text{inf}}$ i.e., if the email address email has been used as input to compute the seed. The network $\mathbf{A}$ is depicted in Figure 17. Here, we recycle the general architecture of the sub-encoder η (see Section IV-B1) to build a feature extractor for the input email address. The output of the sub-network, together with the configuration seed ψ, are then provided as input to a model D. The

distinguisher D is a fully-connected network with architecture:

1) $\psi \parallel \eta$(`email`) $\rightarrow$ `dense(512)`
2) `dense(320)` $\rightarrow$ `batchnorm` $\rightarrow$ `ReLU`
3) `dense(160)` $\rightarrow$ `batchnorm` $\rightarrow$ `ReLU`
4) `dense(80)` $\rightarrow$ `batchnorm` $\rightarrow$ `ReLU`
5) `dense(40)` $\rightarrow$ `batchnorm` $\rightarrow$ `ReLU`
6) `dense(1)` $\rightarrow$ `sigmoid`

The task of the distinguisher is to infer the probability that `email` $\in \tilde{\tilde{A}}_{\text{inf}}$ given the two inputs.

To train the model **A**, we use $\mathbf{L}_{\text{train}}$ (which is public). For every leak $S = (A_{\text{inf}}, X) \in \mathbf{L}_{\text{train}}$, we sample a subset of k email address $\tilde{\tilde{A}}_{\text{inf}}$ from A_{inf} and use it to compute a configuration seed ψ via the pre-trained configuration encoder β_θ (which is public as well). Then, we create a training set for the attacker's model **A** by generating triplets of the form:

$$(\psi = \beta_\theta(\tilde{\tilde{A}}_{\text{inf}}), \text{email}, \text{email} \in \tilde{\tilde{A}}_{\text{inf}}),$$

where the last element is a binary label $\{0, 1\}$. During the training, the model is optimized to predict the correct label (i.e., membership) given the input pair (ψ, email). When generating the training sets, we ensure that the two labels have the same proportion (i.e., 50% "0" and 50% "1"). We do the same for $\mathbf{L}_{\text{test}}$ and use the output as the validation set for the attack. We generate the databases for both the private `UNCM` and the vanilla one (that we use as a baseline). In our setup, we study the attacks in the toy-setting with $k = 10$ as MIAs fail on the non-private configuration seed when larger values for k are used e.g., 2048. For the private setting, this results in $(\epsilon = 1.44, \delta = 5 \cdot 10^{-5})$-DP configuration seeds which is our worst-case setting.

For both the private and non-private settings, we train **A** using the *Adam* optimizer [36] in the default setting, a batch size of 256, and early stopping. After the training, we tested the two models on the validation sets derived from $\mathbf{L}_{\text{test}}$.

*b) **Results:*** The accuracy of the attack on the non-private seed is $62.13\% \pm 0.60$, where random guessing is 50%. For the $(\epsilon = 1.44, \delta = 5 \cdot 10^{-5})$-DP seed, the accuracy of the MIAs is $50.26\% \pm 0.64$ which is marginally bellow the bound imposed by $(\epsilon = 1.44, \delta = 5 \cdot 10^{-5})$-DP.