Adaptive Password-Strength Meters from Markov Models

Claude Castelluccia
INRIA
claude.castelluccia@inria.fr

Markus Dürmuth
Ruhr-University Bochum
markus.duermuth@rub.de

Daniele Perito
INRIA
daniele.perito@inria.fr

Abstract

Measuring the strength of passwords is crucial to ensure the security of password-based authentication. However, current methods to measure password strength have limited accuracy, first, because they use rules that are too simple to capture the complexity of passwords, and second, because password frequencies widely differ from one application to another. In this paper, we present the concept of adaptive password strength meters *that estimate password strength using Markov-models. We propose a* secure *implementation that greatly improves on the accuracy of current techniques.*

1 Introduction

Passwords are a traditional and widespread method of authentication, both on the Internet and off-line. Passwords are portable, easy to understand for laypersons, and easy to implement for the operator. Thus, password-based authentication is likely to stay for the foreseeable future. Most sites let users choose their password, as the usability of automatically generated passwords is low [32]. However, users tend to choose weak passwords, for instance, many users choose passwords from a rather small set of passwords, and hence, these passwords can easily be guessed.[1]

To ensure an acceptable level of security of user-chosen passwords, sites often use mechanisms to test the strength of a password (often called *pro-active password checkers*) and then reject weak passwords. Hopefully this ensures that passwords are reasonably strong on average and makes guessing passwords infeasible or at least too expensive for the adversary. Commonly used password checkers rely on rules such as requiring a number and a special character to be used. However, as we will show and also has been observed in previous work [28], the accuracy of such password checkers is low, which means that often insecure passwords are accepted and secure passwords are rejected. This adversely affects both security and usability. The commonly used rule-sets are too simple to capture the complexity of passwords, and users confronted with them often choose the same modifications to fulfil the rules (like the sadly famous `password1`). Furthermore, password distributions differ from site to site (for example due to language differences) and no password checker is equally suitable for all sites.

We propose to use password strength meters based on Markov-models, which estimate the true strength of a password more accurately than rule-based strength meters. Roughly speaking, the Markov-model estimates the strength of a password by estimating the probability of the n-grams that compose said password. Best results can be obtained when the Markov-models are trained on the actual password database. We show, in this paper, how to do so without sacrificing the security of the password database, even when the n-gram database is leaked. We call these *Adaptive* Password Strength Meters (APSMs), as they can react dynamically to changes in how users choose passwords. Previous work [6, 19] has already shown that Markov-model based password *crackers* outperform existing cracking techniques, and conjectured that they could be used to create better proactive password strength meters. However, before our work it was unclear how this could be implemented securely, and how accurate it would be be.

In this paper, we show how to build secure adaptive password strength meters, where security should hold even when the n-gram database leaks. This is similar to traditional password databases, where one tries to minimize the effects of a database breach by hashing and salting the stored passwords. This is not a trivial task. One potential problem is that, particularly strong passwords, can be leaked entirely by an n-gram database (without noise added). This is best illustrated with an example: If the password "`!*(%$.&`" is used, then from the 5-grams "`!*(%$`", "`*(%$.`", and "`(%$.&`" one can reconstruct the password with high probability by searching for the overlapping segments. By adding a carefully chosen amount of noise we can eliminate this kind of attack, and

[1] From leaked password lists we learn that 20% of passwords are covered by a list of only 5,000 common passwords [27].

in fact we can prove that the noisy n-gram database leaks very limited information about the actually stored password database. By varying the amount of noise added we can balance our system between security and privacy, and we show parameters that strike a reasonable balance between the two for large password databases.

The attacker model makes the following two assumptions: First we make the common assumption that the adversary does not attack one specific user, but is interested in guessing *any* password correctly in the system. This means that the attacker does not use information about a specific user such as his date of birth, children's names, and so on.

Second, we argue that a reasonable analysis of password strength needs to assume that the adversary knows the distribution of the passwords. This is a necessary assumption, as a number of password databases have leaked over the past years and are available to everybody. An attacker might be able to obtain a very accurate distribution for a given site by correlating user statistics. Consequently, assuming that the exact distribution is known to the attacker is the only way to conservatively estimate his knowledge.

1.1 Contributions

This paper makes the following contributions:

- We propose the first adaptive password strength meter (APSM). Our password meter, based on n-gram models, is both accurate in gauging password strength and secure against the strong attacker model described above.
- The need for adaptive password strength meters is motivated in Section 3 by showing that the way users choose passwords is greatly influenced by the type of service (among other factors), which reduces the accuracy of heuristic password checkers that are widely in use today.
- Since our construction needs to store additional (non hashed) information about passwords, we formally prove the security of our scheme, by proving an upper bound on the information that leaks per passwords if a data breach occurs and the actual n-gram database is leaked. We show a bound of 1.3 (per password) bits for reasonable parameters. This leakage only occurs if the password database is breached.
- We evaluate the accuracy of our scheme by performing extensive experiments. More specifically, we show that our scheme outperforms existing schemes, such as NIST, Microsoft, and Google schemes. We also show that the noise added on n-grams, necessary to provide security in case of leakage of the n-gram dataset, does *not* significantly affect performance. We discover that traditional password meters are inaccurate, and perform only slightly better than a scheme that would output random values. On the contrary, our construction can distinguish between strong and weak passwords with high accuracy.

1.2 Related Work

We review relevant previous work on password metrics as well as work in the closely related field of password guessing.

Password metrics: Estimating the strength of passwords as a measure to defend against guessing attacks has a long history. In [18], password checking was done by attempting to crack the hashed passwords. The ones successfully cracked were marked as weak and the users notified.

Later, one started to estimate the strength of a password before it is accepted by a system by what are called *pro-active password checkers* or *password strength meters (PSM)*, using certain rules-sets that aim at excluding weak passwords [24, 11, 2, 20]. An influential PSM was proposed by the NIST [4] (see Section 2.2 for a comparison of rule-sets that are used in practice). Recently, the NIST rule-set was shown [28] to be a rather weak measure for the actual password strength (see also [12] for more results on such a comparison). We reach a similar conclusion in Section 6, showing a low correlation of this measure and the actual password strength.

Schechter et al. [23] classify passwords as weak by counting the number of times a certain password is present in the password database. This password meter, however, cannot generalize on common variations of weak passwords, e.g., `password1`. These variations have to become *popular* before the system can mark them as weak. Our construction, instead, can easily meter those password by leveraging on the capabilities of Markov-models.

Several papers study user behaviour regarding (in-)secure passwords [22, 9, 26, 30]. Some more work on estimating password strength or related ways to increase the security of password-based authentication can be found in [21, 31, 13].

Password cracking: Password cracking is a problem in many ways similar to estimating a password's strength. [18, 5, 15]. To protect passwords they are usually stored in hashed form, and under the common assumption that an attacker cannot invert this hash function, his optimal strategy is to test passwords in decreasing likelihood, i.e., most frequent passwords first. This means the attacker needs a method to enumerate passwords with decreasing likelihood, in other words, with increasing strength.

While most previous attacks use large (external) dictionaries and ad-hoc mangling rules to modify these, Narayanan and Shmatikov [19] proposed Markov-models to overcome

some of the problems of dictionary-based attacks, by training Markov-models to general rules that passwords follow. This work also is the first to hint that Markov-models might yield useful password strength meters; however, they neither considered accuracy nor security of such a construction. While [19] uses ad-hoc "templates" on password structures, i.e., on the proportion and position of numbers and letters, subsequent work [29] learns these structures from sets of leaked passwords. An empirical study on the effectiveness of different attacks including those based on Markov-models can be found in [6]. A study also taking into account password re-use is [7].

1.3 Organization

The rest of the paper is organized as follows: We review password strength meters in Section 2. We motivate and define adaptive password strength meters in Section 3 and give the construction of such a scheme in Section 4. Next, we prove its security in Section 5, demonstrate its accuracy in Section 6, and discuss some implementation details in Section 7 . Finally, Section 8 concludes the paper.

2 Password Strength Meters

A password strength meter is a function $f\colon \Sigma^* \to \mathbb{R}$, that takes as input a string (or password) x over an alphabet Σ and outputs a number s, a *score*, which is a measure of the strength of string x as a password. The output is, in general, a real number indicating the passwords strength. The strength s provides an estimation of the effort an attacker is required to invest to guess the password x and can be used either to advise the users on the strength of the passwords or to enforce mandatory password strength policies. Both approaches have their merits and problems.

The strength of a password is the amount of work an adversary needs to break the password. In the context of password guessing, the adversary has a way to check if a (guessed) password is the correct one, usually by a hash-value representing the hash of the correct password, by an online-login, or similar means. Consequently, the optimal strategy for an attacker is to guess passwords in increasing order of strength and decreasing order of probability, i.e., more likely passwords are tried before less likely ones. This also motivates the definition of guessing entropy, which gives the average number of passwords an attacker has to guess before finding the correct one. Let X be a random variable with finite domain D and $P(X = d_i) = p_i$, ordered with decreasing probabilities $p_i < p_j$ for $i < j$. The *guessing entropy* $G(X)$ is defined as

$$G(X) = \sum_{i=1}^{|D|} i \cdot Pr(X = i). \qquad (1)$$

2.1 An Ideal Password-Strength Meter

Definition 1 *Let us fix probabilities $P\colon \Sigma^* \to [0,1]$ on the space of passwords (i.e., strings over a certain alphabet). An* ideal password checker *$f(x)$ is given by the function $f(x) = -\log(P(x))$.*

We denote this password strength meter as "ideal", as the order which is given by this function is the same as the order with which passwords are guessed in an optimal guessing attack. Consequently, the following two functions $f'(x) = 1/P(x)$ and $f''(x) = R_P(x)$ also constitute ideal password checkers, where $R_P(x)$ is the rank of x according to the distribution P, i.e., if the probabilities $p_i = P(x_i)$ are ordered with $p_i < p_j$ for $i < j$, then $R_P(x_i) = i$.

Any password checker or password strength meter can be seen as an approximation to this function, often clustered in few buckets such as "insecure/secure", or "insecure/medium/high". We will survey some commonly used approximations in the next section.

2.2 Common Approximations

The commonly used (rule-based) password checkers can be seen as ad-hoc approximations to an ideal strength meter as described above, where passwords with strength over a certain threshold should be accepted, and otherwise rejected. There is a large number of password checkers employed today. Many web-services implement their own password checker using a combination of educated guessing and craft. However, as we will show, these approximations perform poorly in the task of gauging the strength of a password. As a consequence, both usability (by scoring strong passwords as weak) and security (by scoring weak passwords as strong) are affected.

We compare our work to three different password checkers that are widely used today: the NIST, Google and Microsoft password checkers. They were chosen because of their popularity and because they are representative for the techniques employed currently for password strength meters.

The *NIST password meter* tries to estimate the entropy of a password mainly based on their length. Special bonuses (in bits) are given if the password matches particular conditions, like containing special characters or a combination of upper case and numbers. The NIST also optionally suggests to give a bonus for a dictionary check. We did not implement this feature as it would have been bound to a specific dictionary, and it affects a small number of passwords only.

The *Microsoft password meter* [17] (employed, e.g., in Hotmail, Windows Live) outputs an integer in the range $[0, 4]$. The strongest score 4 is given to passwords that match all the "good" conditions: passing a dictionary test (included in the JavaScript), minimum length of 14, using at least three

types of characters (upper, lower, number, special). The tool is implemented as a JavaScript routine that measures the strength of passwords entered in a text box. For our tests we downloaded the JavaScript code and ran it inside SpiderMonkey [25], a Perl interface to the Mozilla JavaScript Engine. This allowed us to use the tool unchanged for our experiments.

The *Google password meter* also outputs an integer between 0 and 4. We do not know the inner workings of this meter, that is implemented on the server-side and not in JavaScript. This design choice is probably affected by security concerns, as it "hides", to possible attackers, how passwords are metered. Based on our observations, however, this password checker seems to be based on a set of fixed rules like the others. We used the server as an oracle to measure the password strength.

3 Adaptive Password Strength Meters

A password strength meter is a fixed function, and cannot take into account site-specific aspects of the password distribution. This is why we propose using adaptive password strength meters, which can additionally base their score on specifics of the site. To our knowledge this has not been defined before, even though the motivation is similar to [23].

Definition 2 *An* adaptive password strength meter (APSM) $f(x, L)$ *is a function* $f\colon \Sigma^* \times (\Sigma^*)^k \to \mathbb{R}$*, that takes a string (or password)* x *over an alphabet* Σ *and a password file* L *containing a number of passwords as input and outputs a* score S.

Intuitively, the password database L contains a number of passwords sampled from the same distribution, and the task of the password strength meter is to estimate the strength of the password x based on his estimation of P. Alternatively, as in our scheme, a *noisy* model of L is stored to preserve the secrecy of the password database. The amount of noise added to the model of L needs to strike a balance between accuracy and secrecy.

We note that the adaptive password strength meter f does not need to have a-priory knowledge of the distribution P, whereas a (non-adaptive) password strength meter does.

3.1 The Need for Adaptive Password Meters

The main motivation for adaptive password meters is the observation that password distributions are different for different sites. This is illustrated by Figure 1(b), which shows the ten most frequent passwords from three leaked password sets (more information about these datasets can be found in Section 6). At a glance it can be seen that the passwords in each service have distinctive characteristics. For example the password `rockyou` is only popular in the RockYou service. While this password and possible variations have a very low guessing entropy on RockYou, this is not the case on different web services. The MySpace passwords, at some point, have been influenced by a rule-based password checker that requires number to be included in the passwords.

The results of a more comprehensive analysis are shown in Figure 1(a). These graphs show the fraction of frequent passwords that are shared by two or more password lists, considering varying thresholds. While RockYou and PhpBB share 50% of the most common passwords, other services share substantially less passwords, with as few as 10% shared between RockYou and MySpace.[2]

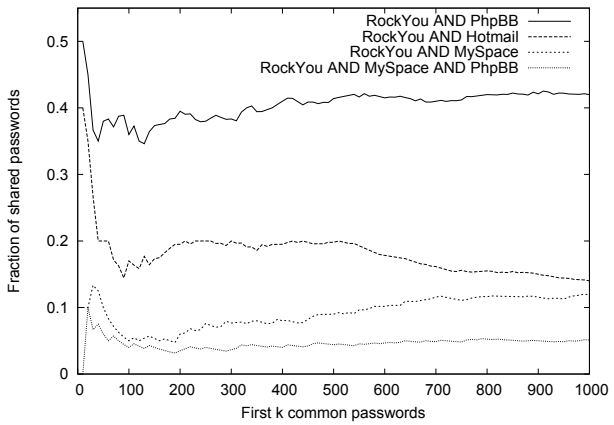
We believe these differences are caused by different languages spoken and cultural backgrounds of the users, different perceived importance of the password, different password checkers in place, and so on. This variety of different influences makes it very hard for a site to estimate the actual distribution in advance, without accessing the actual passwords. Furthermore, the password checkers influence the distribution of passwords, as exemplified by the MySpace passwords. This means that no static set of rules can capture the dynamic changes in password creation habits by the users.

3.2 Challenges

Security: While adaptive password meters have the potential for a better and more robust estimation of password strength, computing the strength of a given password based on the actual password database (and outputting this value to the user) prompts a few security concerns. Our main contribution is to devise an adaptive password meter that is secure even when the n-gram database is leaked, while still providing high accuracy.

Accuracy: A password strength meter needs to correctly gauge the strength of a password as wrong estimations can affect both the usability and guessing entropy of the passwords: if strong passwords are marked as weak then users might get frustrated, reducing usability; if weak password are marked as strong, security is decreased. For example, users might "circumvent" rule-based password meters by modifying weak passwords to match the required minimum strength. For example, `password` is frequently "strengthened" by the addition of upper cases and numbers, leading to `Password1`.

[2] Further details on the datasets are given in Section 6.



(a) Fraction of passwords that are used in common by different sites

RockYou	Faithwriters	MySpace
123456	123456	password1
12345	writer	abc123
123456789	jesus1	fuckyou
password	christ	monkey1
iloveyou	blessed	iloveyou1
princess	john316	myspace1
1234567	jesuschrist	fuckyou1
rockyou	password	number1
12345678	heaven	football1
abc123	faithwriters	nicole1

(b) Ten most frequent passwords for different sites. Passwords underlined are shared by at least two services. The wide difference likely depend on background (e.g., Faithwriters) or password rules (e.g., MySpace).

Figure 1. Password Distributions for Different Sites

4 Constructing an Adaptive Password Meter

We use techniques from statistical language processing, namely Markov-Models [14], to implement an adaptive password strength meter that both *accurately estimates* the probabilities from a relatively small sample, and is *secure against leakage* of the locally stored data. Basically, for every password x which is added to the password list, we store the hashed (and possibly salted) value $Hash(x)$ as commonly done. In addition, we determine the n-gram counts for the password (i.e., frequencies of n consecutive characters in the password), merge this information with previously stored n-grams, to obtain the frequencies over the entire password database, and add some noise to this data. These n-gram frequencies can then be used to compute an estimate on the probability of a fresh password.

4.1 Markov Models

Over the last years, Markov models have proven very useful for computer security in general and for password security in particular. For example, Narayanan et al. [19] showed the effectiveness of Markov models to password cracking.

The idea is that adjacent letters in human-generated passwords are not independently chosen, but follow certain regularities (e.g., the 2-gram `th` is much more likely than `tq` and the letter `e` is very likely to follow `th`). In an n-gram Markov model, one models the probability of the next character in a string based on a prefix of length n. Hence for a given string $c_1, \ldots, c_m$ we can write

$$P(c_1, \ldots, c_m) = \prod_{i=1}^{m} P(c_i | c_{i-n+1}, \ldots, c_{i-1}).$$

Our construction only keeps track of the n-gram counts $count(x_1, \ldots, x_n)$, and the conditional probabilities can easily computed from these by the following formula:

$$P(c_i | c_{i-n+1}, \ldots, c_{i-1}) = \frac{count(c_{i-n+1}, \ldots, c_{i-1}, c_i)}{\sum_{x \in \Sigma} count(c_{i-n+1}, \ldots, c_{i-1}, x)}$$

Also note that the size of the password's alphabet (Σ) is quite important. We initially used an alphabet of size 96 to estimate the password strength. However, we observed that most of the characters are rarely used, leading to sparseness problems. In the final version of our scheme, we choose to use the following alphabet composed of 38 distinct characters $[a-z][0-9][U][S]$, where U and S are two symbols representing all upper-case letters and all special characters, respectively. This leads to reduce sparseness in the dataset and better probability estimations.

4.2 Our Construction

Our construction uses Markov Models [14] from the previous section to estimate the strength of passwords. What makes this construction interesting is that, by adding some fine-dosed amount of noise, we can actually prove that the construction is secure against leakage of the n-gram database. (For notational simplicity, we assume that all passwords have

length $m = l + n - 1$, thus the total number of n-grams is ln.)

4.2.1 N-gram Database Construction and Update (DCU algorithm)

The n-gram database is constructed and updated as follows:

1. The algorithm keeps as state the n-gram counts, $count(x_1, \ldots, x_n)$, for all $x_1, \ldots, x_n \in \Sigma$. All these counts are initialized to 0.

2. Whenever a password $c = c_1, \ldots, c_{l+n-1}$ is added, it is decomposed into n-grams. The database is then updated by incrementing the counts corresponding to each of the password's n-grams by 1, as follows:

   $$count(c_i, \ldots, c_{i+n-1}) = count(c_i, \ldots, c_{i+n-1}) + 1,$$

   for each $i = 1, \ldots, l$.

3. Additionally, noise is added to the database by increasing each individual n-gram's count, by one, with probability γ each (independently):

   $$\forall x_1, \ldots, x_n : \\ count(x_1, \ldots, x_n) = count(x_1, \ldots, x_n) + 1$$

   with probability γ.

4. Finally, the password is stored hashed with salt as usual, e.g., choose $salt \leftarrow^R \{0,1\}^{16}$ and store

   $$hash(x \parallel salt) \parallel salt.$$

4.2.2 Password Strength Estimation (PSE algorithm)

The strength of a password $c = c_1, \ldots, c_m$, where each character c_i is chosen from alphabet Σ, is estimated as follows:

1. For $i = 1, \ldots, m$, the following conditional probabilities are computed:

   $$P(c_i | c_{i-n+1}, \ldots, c_{i-1}) \\ = \frac{count(c_{i-n+1}, \ldots, c_i)}{count(c_{i-n+1}, \ldots, c_{i-1})} \\ = \frac{count(c_{i-n+1}, \ldots, c_i)}{\sum_{x \in \Sigma} count(c_{i-n+1}, \ldots, c_{i-1}, x)}.$$

   (If the numerator equals zero we use a small out-of-dictionary probability, to account for unseen n-grams. However, this will almost never happen, due to the added noise.)

2. Finally, the strength estimate $f(c)$ for the password c is:

   $$f(c) = -log_2(\prod_{i=0}^{m} P(c_i | c_{i-n+1}, \ldots, c_{i-1}))$$

Example: The probability of the string `password` (with $n = 5$) is computed as follows

$$P(password) = P(p)P(a|p)P(s|pa) \ldots P(d|swor)$$

Picking one of the elements as an example: $p(o|assw) = \frac{count(asswo)}{count(assw)} = \frac{98450}{101485} = 0.97$. This results in the overall estimation $P(password) = 0.0016$, where the actual frequency of this password in the RockYou database is 0.0018. A short list of passwords as scored by our markov model is included in Table 1. The example shows that the markov meter is able to correctly recognize weak passwords and, in fact, closely approximate the actual probability of the passwords. Furthermore, more complex and random passwords are correctly estimated as stronger.

On the other hand, the NIST, Microsoft and Google password checkers fail to correctly gauge password strength. For example, the Google meter gives a 0 score to a random password (`dkriouh`) only because it does not match the minimum length of 8 characters. The Microsoft checker gives a very high score to `P4ssw0rd` and `Password1`, while giving a lower score to the randomly generated password `dkriouh`. The NIST checker assigns a higher score to `Password1` than to `2GWapWis`. Mistakes of this type are mostly inevitable in heuristic password checkers and are reflected in the poor accuracy of these as shown in the next sections.

5 Security of our Construction

Under normal operation, the n-gram database will be secret and not accessible to an attacker. However, best practices mandate that even when an adversary gets access to the n-gram database (e.g., by breaking into the server), little information about the plain-text passwords should be leaked. In this section we will show that our system has this property, by proving a strong bound on the (Shannon) entropy that leaks when the n-gram database leaks. Consequently, the remaining guessing entropy of the passwords in the database remains high by the results from [16].

5.1 Security Definition

When defining the security of an adaptive password checker, it is necessary to consider the adversary's prior knowledge on the password distribution. An adversary can

Password	Ideal	Markov	NIST	MS	Google
password	9.09	9.25	21	1	1
password1	11.52	11.83	22.5	2	1
Password1	16.15	17.08	28.5	3	1
P4ssw0rd	22.37	21.67	27	3	1
naeemha	21.96	28.42	19.5	1	0
dkriouh	N/A	42.64	19.5	1	0
2GWapWis	N/A	63.67	21	3	4
Wp8E&NCc	N/A	67.15	27	3	4

Table 1. Scores (in bits) as computed by the markov model and the ideal password meter from Section 2.1. For the ideal meter the probability is the empirical frequency in the RockYou dataset. The last three passwords were generated at random using, respectively, the following rules `[a-z]{7}`**,** `[a-zA-Z0-9]{8}` **and** `[a-zA-Z0-9\special]{8}`**.**

have access to statistics from lists such as language dictionaries and leaked password lists, and knowledge about common mangling rules (i.e., rules to derive more passwords from such lists by appending numbers and special characters). However, there are many more sources of information an adversary has access to: their technical background, the password policies enforced by the site, theme of the site, etc.

It is hard, if not impossible, to come up with a comprehensive list of sources that the adversary is using. Therefore, in this paper, we consider an adversary who knows the distribution of the service's passwords. This automatically considers all the above sources of information, and protects us against any future improvements of password cracking software, such as John the Ripper.

We explicitly note that in normal operation our password strength meter hardly leaks information. (An exception is the unavoidable leakage from the password strength meter itself; as we cannot prevent the attacker from accessing the password strength meter f, we cannot prevent him from learning a limited number of data points of this distribution.) Only when the n-gram database is leaked, then the distribution, as well as some bits of additional distribution about the actual passwords in the database, leak.

One might argue that while the password distribution is generally known, it is not known for very unlikely passwords, and the n-gram database might leak these passwords. This is incorrect since we are adding noise to all n-grams, including rare ones. In addition, as explained previously, knowing the n-gram frequencies does not help the adversary to break unlikely, i.e., strong passwords.

The assumption that the password distribution is known can be seen as an extension of Kerckhoffs' principle [10]. We do not assume that the distribution is secret, but only the password chosen with this distribution are. It also underlies the definition of guessing entropy (see Equation (1)), which also considers the optimal guessing strategy.

Finally, we argue that this assumption does not weaken security since by enforcing a minimum password strength (similar to [23]) we can still guarantee a minimal guessing entropy of the passwords. In fact, if we assume that X is a distribution on passwords with $Pr(x) \leq t$ for all passwords x, then the Shannon entropy of X is bounded by

$$H(X) \geq -\log(t). \tag{2}$$

By using the results from [16] we can compute a lower bound on guessing entropy as

$$G(X) \quad \geq \quad \frac{1}{4}2^{H(X)} \geq \frac{1}{4t}. \tag{3}$$

Enforcing, e.g., a maximum probability of 2^{-20} yields a lower bound on the guessing entropy of 2^{18}. In other words, a strong password will remain strong, even if the password distribution is known. The adversary will be able to compute its guessing entropy from the distribution, but not more. On the other hand, weak passwords might be easier to break since the adversary will be able to optimize his guessing strategy. However, such passwords should be prohibited in most services.

5.2 Information and Entropy

The (noisy) n-gram database, when leaked to an adversary, constitutes a noisy channel that transports information about the stored passwords in the database. In this section we introduce the notion of entropy and mutual information, including some basic properties.

The information content of a random variable is measured in terms of entropy. For two discrete random variables X, Y with finite domain $D = \{d_1, \ldots, d_n\}$, *Conditional Shannon entropy* is defined as

$$H(Y|X=x) \quad \stackrel{\mathsf{def}}{=} \quad -\sum_{i=1}^{n} Pr(Y=d_i|X=x)\log(Pr(Y=d_i|X=x)) \tag{4}$$

and

$$H(Y|X) \stackrel{\mathsf{def}}{=} \sum_{x\in X} Pr(X=x)H(Y|X=x).$$

For independent random variables X, Y, a simple calculation shows

$$H(X+Y|X) = H(Y). \tag{5}$$

A central notion used to define the transport of information on a (noisy) channel is the notion of mutual information. The *mutual information* between the input X and the output

Y, where both X and Y are discrete random variables with finite domain, is given by

$$I(X,Y) \stackrel{\mathsf{def}}{=} H(Y) - H(Y|X). \tag{6}$$

When defining channel capacity, one takes the maximum of the mutual information over input distributions; in our application the input distribution is fixed by the construction, so the information flow is given directly by the mutual information $I(X,Y)$.

Computing a closed formula for the entropy of a given distribution can be hard. However, for some distributions such as the binomial distribution the literature gives closed formulas. Let $Bin(m,p)$ denote the binomial distribution with m trials that follow a Bernoulli distribution with parameter p each, and write $q = 1-p$. From [1] we obtain the following estimates on the entropy of binomial variables:

$$\begin{aligned} &H(Bin(m,p)) \quad \geq \\ &\frac{1}{2} + \frac{1}{2}\log(2\pi mpq) + \frac{C_1^{(p)}}{m} + \frac{C_2^{(p)}}{m^2} + \frac{C_3^{(p)}}{m^3} \end{aligned} \tag{7}$$

and

$$\begin{aligned} &H(Bin(m,p)) \leq \\ &\frac{1}{2} + \frac{1}{2}\log(2\pi mpq) + \frac{C_4^{(p)}}{m} \end{aligned} \tag{8}$$

for $C_1^{(p)} = \frac{13}{12} - \frac{3\log(pq)}{2} - \frac{5}{6pq}$, $C_2^{(p)} = -\frac{7}{3} + 4\log(pq) + \frac{8}{3pq} - \frac{1}{6(pq)^2}$, $C_3^{(p)} = -\frac{1}{360}$, and $C_4^{(p)} = \frac{1}{12} + \frac{\log(pq)}{2} + \frac{1}{6pq}$.

From [33] we learn that the entropy of the sum of two binomially distributed random variables with the same number of trials can be estimated by the the entropy of a binomial variable with double the number of trials and the average parameter:

$$H\Big(Bin(m,a) + Bin(m,b)\Big) \leq H\Big(Bin(2m, \frac{a+b}{2})\Big). \tag{9}$$

5.3 Leakage Estimation

To prove the security of the scheme we prove that the total amount of information leaked (in terms of Shannon entropy) is limited (A detailed version of this section can be found in Appendix A).

In this section we use the following variable names: n denotes the length of the n-grams, $N = |\Sigma|^n$ is the total number of n-grams, l the number of n-grams per password, k the number of passwords in the database, $m \stackrel{\mathsf{def}}{=} k \cdot l$ the total number of N-grams in the database, γ is the probability of adding noise used in the construction, and L is the information leaked from the database.

First, we consider the leakage of an individual n-gram with index j (where $1 \leq j \leq N$) with expected frequency t_j, and we consider n-grams at the i-th position of the passwords only (thus $1 \leq i \leq l$). Define the random variables $S_h^{i,j} := \chi_j(P_h^i)$, where $\chi_j(P_h^i)$ is the indicator function for the j-th n-gram on the i-th position in the h-th password. The random variables $S_h^{i,j}$ all have $Pr(S_h^{i,j} = 1) = t_j$ and $Pr(S_{i,j}^h = 0) = 1 - t_j$.

For $1 \leq i \leq l$ and $1 \leq j \leq N$, we define the empirical frequency obtained from a password database with k passwords as $T_k^{i,j} = \frac{1}{k}\sum_{h=1,\ldots,k} S_h^{i,j}$. $T_k^{i,j}$ follows a binomial distribution $Bin(k,t_j)$ with k trials with success probability t_j each.

Let $R^{i,j} \sim Bin(k,\gamma)$ be random variables describing the noise added to the i-th n-gram in our construction, where $1 \leq i \leq l$ ranges over the n-grams per password, and $1 \leq j \leq N$ ranges over all n-grams, and let $O_k^{i,j} \stackrel{\mathsf{def}}{=} T_k^{i,j} + R^{i,j}$ be the observed noisy value. The information that can leak by publishing a single noisy n-grams is the mutual information between $T_k^{i,j}$ and $O_k^{i,j}$, i.e., the quantity $I(T_k^{i,j}; O_k^{i,j})$:

$$\begin{aligned} &I(T_k^{i,j}; O_k^{i,j}) \\ &\quad = H(O_k^{i,j}) - H(O_k^{i,j}|T_k^{i,j}) \\ &\quad = H(O_k^{i,j}) - H(R^{i,j}), \end{aligned} \tag{10}$$

where the first equality is the definition of mutual information (Eq. (6)), and the second equality follows from Equation (5).

Using first Equation (9) and then Equations (7) and (8) we can evaluate this further as follows:

$$\begin{aligned} &H(O^{i,j}) - H(R^{i,j}) \\ &\quad = H(Bin(k,\gamma) + Bin(k,t_j)) - H(Bin(k,\gamma)) \\ &\quad \leq \gamma + \frac{t_j}{2\gamma} + \frac{C_4^{(\gamma+t_j)/2}}{2k} - \frac{C_1^\gamma}{k} - \frac{C_2^\gamma}{k^2} - \frac{C_3^\gamma}{k^3} \end{aligned} \tag{11}$$

The full n-gram database which is leaked is the concatenation of the sums of the individual n-gram counts, i.e., the actual information leakage is[3]:

$$\begin{aligned} L &= I((\sum_{i=1}^{l} T_k^{i,j})_{1\leq j\leq N}; (\sum_{i=1}^{l} O_k^{i,j})_{1\leq j\leq N})) \\ &\leq 4k\gamma + \frac{1}{\gamma}\left(6.1 + \frac{1}{5k\gamma}\right) \end{aligned} \tag{12}$$

for $l = 4$. For a reasonable choice of parameters $\gamma = \frac{1}{1.000.000}$ and $k = 5.000.000$ this means that overall, the n-gram database can leak at most 6.2 million bits, or on "average" about 1.3 bits per password. Values for different parameters are given in Table 2.

[3] Appendix A presents in details how the results presented in this section were derived.

	$k = 1 \cdot 10^6$	$k = 5 \cdot 10^6$	$k = 1 \cdot 10^7$
$\gamma = 5 \cdot 10^{-7}$	$13.1 \cdot 10^6$ (13.1)	$12.4 \cdot 10^6$ (2.5)	$12.4 \cdot 10^6$ (**1.24**)
$\gamma = 1 \cdot 10^{-6}$	$6.4 \cdot 10^6$ (6.4)	$6.2 \cdot 10^6$ (**1.24**)	$6.2 \cdot 10^6$ (**0.62**)
$\gamma = 5 \cdot 10^{-6}$	$1.3 \cdot 10^6$ (**1.3**)	$1.3 \cdot 10^6$ (**0.3**)	$1.3 \cdot 10^6$ (**0.13**)

Table 2. Total leakage (and average leakage per password) in bits for different parameters of k and γ, bold numbers indicate parameters that strike a good balance between accuracy and security.

6 Accuracy of our Construction

We now evaluate the accuracy of our scheme and compare it with other password strength meters.

Datasets: In order to evaluate the accuracy of our scheme, we use the RockYou password list. This list consists of over 32.6 million passwords that were released to the public after a breach occurred in December 2009 to the website `www.rockyou.com`.[4] The dataset is valuable for different reasons. First, it was the result of an SQL injection attack against the password database that was successful because passwords were stored in clear and not hashed. This means that all the passwords were collected, not only weak ones. A number of other, smaller password lists exist, e.g., the MySpace list which was obtained by a phishing attack and thus might include weaker passwords as well as fake input by users that realized the phishing attempt. Also, RockYou did not enforce any password rules, yielding a "plain" password set that provides a clear insight of how users choose passwords.

We also used the MySpace list of $37,000$ passwords leaked from MySpace by a phishing attack in 2006, $184,389$ leaked by PhpBB in 2009, and a list of 8347 passwords from the religious forum Faithwriters. Note that these passwords were not used to train our model for the experiments, but only to assess the need of adaptive password strength meters as shown in Figure 1(a).

6.1 Measuring Accuracy Using Rank Correlation

First, we analyse the accuracy of our Markov-based password meter (with and without adding noise) as well as the NIST, Google, and Microsoft password meters, by comparing their score to the ideal password strength meter (see Section 2.1). The ideal password meter is built upon the knowledge of the most common passwords in the RockYou dataset. This comparison only makes sense if there are enough data-points to keep the approximation error for the probability low. The estimation is reasonable for the 10000 most frequent passwords; see Appendix B for a derivation of this bound.

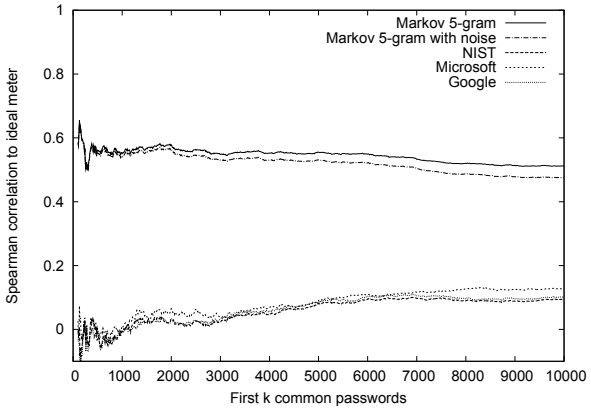
[4] One can remark the irony in the fact that a failure in password storing best-practices led to a deeper understanding of user chosen passwords.

For comparing the different password meters we use Spearman's rank correlation. It describes how well the relationship between two vectors can be described using a monotonic function. The coefficient lies between $[-1, 1]$, with 0 indicating no correlation, $+1$ indicating perfect positive correlation and -1 indicating perfect negative correlation (i.e., given two sets of values X and Y, the Spearman correlation coefficient becomes 1 when X and Y are perfectly monotonically related, i.e., $\forall i, j\ x_i > x_j$ implies $y_i > y_j$). All the password meters under examination where measured by computing the rank correlation of their scores against the scores of the ideal meters. (The intuition is that if the ideal score of password p_1 is higher than the ideal score of password p_2, then an approximation of the ideal meter should rate these two passwords in the same order.) To obtain a more fine-grained comparison, we plotted the correlation at different intervals, using the first $10, 20, ..., 10000$ passwords in the set. The results are shown in Figure 2(a).

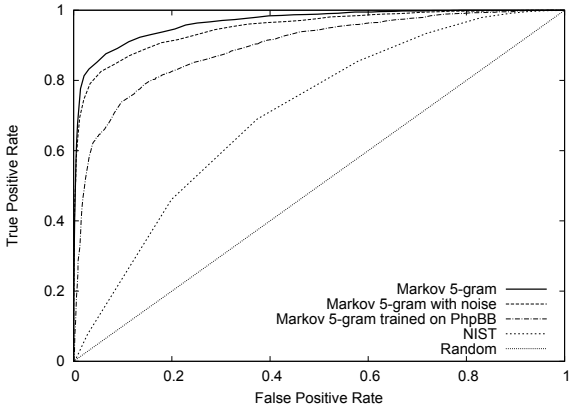
First, we can see that our Markov-based password meter outperforms the other checkers, even when noise is added. Second, the noise added to establish the security of the scheme only slightly decreases the accuracy of the meter. Third, the NIST, Microsoft and Google checker perform equally badly and are not much better than random guessing the password strength, as their correlation with the ideal meter is close to zero.

Training and Testing using Different Datasets: In order to quantify the importance of using the correct password database, in an additional experiment we included data from the PhpBB password list. We computed the optimal score for the 1000 most frequent passwords of the PhpBB list from their empirical frequency, let us call the resulting vector X. Then we scored the same 1000 passwords using the Markov 5-gram Model trained on the RockYou password dataset, we call the resulting vector Y. The correlation between the two vectors X and Y is 0.27 only, much lower than 0.55, the result we obtained by training and testing on the same password list.

This test shows that training a Markov Model on a web service (in this case RockYou) and using it to score passwords on a second service (in our case PhpBB) potentially significantly lowers accuracy, which further strengthens the previous findings in Section 3.1. This serves as justification for our APSM to be trained on the same web service where



(a) Spearman correlation coefficient against the ideal password meter.



(b) ROC Curve comparing the performance of our password meter (with and without noise) to the NIST password checker. $\gamma = 5 \cdot 10^{-6}$ was chosen for the noise.

Figure 2. Accuracy Analysis

it is used to score passwords.

6.2 Measuring Accuracy based on Binary Classification

Next, we compare the accuracy of the password meters in terms of binary classification accuracy, i.e., we establish the accuracy of distinguishing between "strong" and "weak" passwords. The security of a password is the inverse of its probability. In this experiment we set a probability threshold of 2^{-20} to distinguish between strong and weak passwords. For guessing attacks, this threshold guarantees that in order to correctly guess one *single* password the attacker has to guess 2^{18} different passwords. This number should be sufficiently high to prevent online guessing attack if the server uses simple rate limiting on the password guesses. We refer the interested reader to [8] for an analysis of rate limiting policies on passwords.

This threshold was chosen because, given our dataset of 32 million password, we can divide the password in "strong" and "weak" using their frequency and build a *ground truth* set. Roughly, all the passwords that appear with frequency higher than 2^{-20} are marked as weak, the others are marked as strong. However, special caution must be taken when dealing with this probability estimation. In fact, a small estimation error near the threshold can mean that a password is incorrectly labelled. Thus we excluded the passwords with frequency too close to the threshold using Wilson interval estimation for the binomial probability estimation [3], with z-value $z_{1-\alpha/2} = 2.58$.

Using this estimation we can find the passwords above and below the threshold probability with 99% confidence. We divided our testing dataset of 600000 passwords in 4163 surely weak password, called W, and 142205 strong passwords, called S. In this experiment too, we built an ideal ground-truth using the RockYou dataset and sound probability estimations.

We test the accuracy of our Markov-based password meter (with and without noise) in distinguishing between these two sets. We also test the NIST meter alongside as a comparison. The Microsoft and Google meter were not tested in this experiment, as we cannot map the threshold t to the scoring given by those meters. We computed the score of all the passwords in the sets W and S using both the Markov method and NIST. For different thresholds, we measured true positives ratio (passwords correctly marked as weak) and false positive ratio (strong password incorrectly marked as weak). The results can be seen in the ROC curve showed in Figure 2(b). The diagonal line gives the performance of a random password meter, which scores the passwords as weak or strong at random (this is a inherent property of ROC curves). The closer a curve is to the top-left corner (false positive rate of zero and true positive rate of one) the better the classification is. An ideal meter would be a perfect classifier of strong and weak passwords. Again, we can see that our password strength meter (with and without noise) clearly outperforms the NIST meter. Also, the introduction of noise affects performance only slightly. The Markov based meter performs close to optimal with a success rate of 93.4%.

7 Implementation Considerations

7.1 Bootstrapping

One remaining question is how to bootstrap our system. In fact, when only few passwords are stored in the system, those might be particularly vulnerable to leakage as the adversary can easily learn what passwords are *not* present, by

checking what n-grams are not present in the database. We notice, however, that both these problems can be easily addressed. The solution lies in inserting the noise in batches of size k rather than every time a password is inserted. In practice we execute step 3 of the DCU algorithm, (described in Section 4.2.1) k times in advance. The value k should be high enough to provide a balance between usability and security and a reasonable value of k could be $1.000.000$ (Section 5.3 details how k should be selected in order to provide security). Then, for the first k passwords entered in the system, only step 2 and step 4 of the DCU algorithm are executed (i.e., no noise is added as it was added in advance). After the first k passwords, the full DCU algorithm (step 2, step 3 and step 4) is then executed.

Note that our system cannot be used to estimate the strength of the first k passwords (the noise would introduce too much error in the estimations). For these passwords, we suggest to use our scheme with a n-gram database configured with n-grams extracted from a public source such as the RockYou database, and then switch back to our database when the number of passwords exceeds k.

One additional remark is that our password meter is better suited for large web services, that can leverage on password databases of several thousand passwords and larger. In our tests we observed that the accuracy of the Markov model increases rapidly with small training sets and then achieves only marginal better performance with larger training sets. For example, the Markov model trained with only 100.000 passwords achieved an accuracy of 90.3%, when trained with 50.000 passwords an accuracy of 92.9%, and when trained with 32 million passwords an accuracy of 93.4%. This is most likely due to the fact that the most common n-grams are quickly learned, while there is a very long tail of uncommon n-grams. Services that do not have access to tens of thousands of passwords (e.g., smaller web services or local services) must either rely on training the markov model on dictionaries [19] or leaked password databases like RockYou.

7.2 Usability

Our password meter returns real numbers between 0 and 1. However, these values might not be easy for users to interpret. We envision three ways in which the output of our password strength meter can be presented to the users. First, the user can be given the score computed by the Markov model, but converted to a discrete value, e.g., too weak, weak, medium, strong, instead of a real number. The probability could be translated into a sliding "strength bar" that reflects the score computed. This approach would look exactly like current password strength meters, albeit with the higher accuracy given by our construction.

However, we notice that the fine-grained score given by our meter enables us to do more. We could translate the score of the password strength meter into a more concrete metric such as the effort required by an adversary to break the password in terms of time. This could be done, for example, by using the guessing entropy bound derived from the Shannon entropy. In this context, the user could be given a message of the type, "your password can be guessed – on average – in x attempts, that can be tried automatically in about y hours".

Third, the score could be used to compute an estimation of the strength of the password relative to their peers, which might provide an incentive for users to choose better passwords. Users could be prompted with a message that says, for example, "your password is amongst the 5% weakest passwords on our web site", which might encourage them to choose a better password. Conversely, having a password in the top 5% might be an incentive for users to choose stronger passwords.

8 Conclusion

In this work, we proposed a novel way to measure the strength of user-selected passwords. The construction is based on Markov-models and achieves much higher accuracy than commonly used password meters. The fine-grained measurement of the password strength provided by our strength meter allows for a very precise feedback to users.

We have formally proven that this construction is secure even when the local data storage is compromised by an attacker, thus, meeting best practices in securing password storage. We evaluated the accuracy of our scheme by performing extensive experiments and showed that it outperforms existing schemes.

References

[1] J. Adell, A. Lekuona, and Y. Yu. Sharp bounds on the entropy of the poisson law and related quantities. *Information Theory, IEEE Transactions on*, 56(5):2299–2306, 2010.

[2] M. Bishop and D. V. Klein. Improving system security via proactive password checking. *Computers & Security*, 14(3):233–249, 1995.

[3] L. D. Brown, T. T. Cai, and A. DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–133, 2001.

[4] W. E. Burr, D. F. Dodson, and W. T. Polk. Electronic authentication guideline: NIST special publication 800-63, 2006.

[5] J. A. Cazier and D. B. Medlin. Password security: An empirical investigation into e-commerce passwords and their crack times. *Information Security Journal: A Global Perspective*, 15(6):45–55, 2006.

[6] M. Dell'Amico, M. Pietro, and Y. Roudier. Password strength: An empirical analysis. In *INFOCOM '10: Proceedings of 29th Conference on Computer Communications*. IEEE, 2010.

[7] D. Florencio and C. Herley. A large-scale study of web password habits. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

[8] D. Florêncio, C. Herley, and B. Coskun. Do strong web passwords accomplish anything? In *Proceedings of the 2nd USENIX workshop on Hot topics in security*, Berkeley, CA, USA, 2007. USENIX Association.

[9] C. Herley. So long and no thanks for the externalities: The rational rejection of security advice by users. In *Proceedings of the 2009 Workshop on New security paradigms*, pages 133–144. ACM, 2009.

[10] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, January 1883.

[11] D. V. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proc. USENIX UNIX Security Workshop*, 1990.

[12] S. Komanduri, R. Shay, P. G. Kelley, M. L. Mazurek, L. Bauer, N. Christin, L. F. Cranor, and S. Egelman. Of passwords and people: Measuring the effect of password-composition policies. In *CHI 2011: Conference on Human Factors in Computing Systems*, 2011.

[13] C. Kuo, S. Romanosky, and L. F. Cranor. Human selection of mnemonic phrase-based passwords. In *Proc. Symposium on Usable Privacy and Security (SOUPS)*, pages 67–78, 2006.

[14] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[15] S. Marechal. Advances in password cracking. *Journal in Computer Virology*, 4(1):73–81, 2008.

[16] J. Massey. Guessing and entropy. In *IEEE International Symposium on Information Theory*, page 204, 1994.

[17] Microsoft password strength meter. Online at `https://www.microsoft.com/security/pc-security/password-checker.aspx`.

[18] R. Morris and K. Thompson. Password security: a case history. *Communications. ACM*, 22(11):594 – 597, 1979.

[19] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 364–372, New York, NY, USA, 2005. ACM.

[20] The password meter. Online at `http://www.passwordmeter.com/`.

[21] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *Proc. CCS '02*, pages 161–170, 2002.

[22] S. Riley. Password security: What users know and what they actually do. *Usability News*, 8(1), 2006.

[23] S. Schechter, C. Herley, and M. Mitzenmacher. Popularity is everything: a new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the 5th USENIX conference on Hot topics in security*, pages 1–8. USENIX Association, 2010.

[24] E. H. Spafford. Observing reusable password choices. In *Proceedings of the 3rd Security Symposium*, pages 299–312. USENIX, 1992.

[25] SpiderMonkey. Online at `http://search.cpan.org/~mschilli/JavaScript-SpiderMonkey`.

[26] J. M. Stanton, K. R. Stam, P. Mastrangelo, and J. Jolton. Analysis of end user security behaviors. *Comp. & Security*, 24(2):124–133, 2005.

[27] A. Vance. If your password is 123456, just make it hackme. New York Times, oneline at `http://www.nytimes.com/2010/01/21/technology/21password.html`, retrieved May 2011, January 2010.

[28] M. Weir, S. Aggarwal, M. Collins, and H. Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM conference on Computer and communications security (CCS 2010)*, pages 162–175. ACM, 2010.

[29] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek. Password cracking using probabilistic context-free grammars. In *IEEE Symposium on Security and Privacy*, pages 391–405. IEEE Computer Society, 2009.

[30] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy Magazine*, 2(5):25–31, 2004.

[31] J. J. Yan. A note on proactive password checking. In *Proc. NSPW '01*, pages 127–135. ACM, 2001.

[32] J. J. Yan, A. F. Blackwell, and R. J. Anderson. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2:25–31, 2004.

[33] Y. Yu. On the entropy of compound distributions on non-negative integers. *IEEE Trans. Inf. Theor.*, 55:3645–3650, 2009.

A Details of the Leakage Estimation

To prove the security of the scheme we prove that the total amount of information (in terms of Shannon entropy) is limited. In this section we use the following variable names: n denotes the length of the n-grams, $N = |\Sigma|^n$ is the total number of n-grams, l the number of n-grams per password, k the number of passwords in the database, $m \stackrel{\mathsf{def}}{=} k \cdot l$ the total number of N-grams in the database, γ is the probability of adding noise used in the construction, and L is the information leaked from the database.

First, we consider the leakage of an individual n-gram with index j (where $1 \leq j \leq N$) with expected frequency t_j, and we consider n-grams at the i-th position of the passwords only (thus $1 \leq i \leq l$). Define the random variables $S_h^{i,j} := \chi_j(P_h^i)$, where $\chi_j(P_h^i)$ is the indicator function for the j-th $n$gram on the i-th position in the h-th password. The random variables $S_h^{i,j}$ all have $Pr(S_h^{i,j} = 1) = t_j$ and $Pr(S_{i,j}^h = 0) = 1 - t_j$.

For $1 \leq i \leq l$ and $1 \leq j \leq N$, we define the empirical frequency obtained from a password database with k passwords as $T_k^{i,j} = \frac{1}{k} \sum_{h=1,\ldots,k} S_h^{i,j}$. $T_k^{i,j}$ follows a binomial distribution $Bin(k, t_j)$ with k trials with success probability t_j each.

Let $R^{i,j} \sim Bin(k, \gamma)$ be random variables describing the noise added to the i-th n-gram in our construction, where

$1 \le i \le l$ ranges over the n-grams per password, and $1 \le j \le N$ ranges over all n-grams, and let $O_k^{i,j} \stackrel{\mathsf{def}}{=} T_k^{i,j} + R^{i,j}$ be the observed noisy value. The information that can leak by publishing a single noisy n-grams is the mutual information between $T_k^{i,j}$ and $O_k^{i,j}$, i.e., the quantity $I(T_k^{i,j}; O_k^{i,j})$:

$$\begin{aligned} I(T_k^{i,j}; O_k^{i,j}) &= H(O_k^{i,j}) - H(O_k^{i,j}|T_k^{i,j}) \quad (13) \\ &= H(O_k^{i,j}) - H(R^{i,j}), \end{aligned}$$

where the first equality is the definition of mutual information (Eq. (6)), and the second equality follows from Equation (5).

Using first Equation (9) and then Equations (7) and (8) we can evaluate this further as follows:

$$\begin{aligned} & H(O^{i,j}) - H(R^{i,j}) \quad (14) \\ &= H(Bin(k,\gamma) + Bin(k,t_j)) - H(Bin(k,\gamma)) \\ &\le H(Bin(2k, \frac{\gamma + t_j}{2})) - H(Bin(k,\gamma)) \\ &\le \frac{1}{2}\ln\left(\frac{1}{1-\gamma}\right) + \frac{1}{2}\ln\left(1 + \frac{t_j}{\gamma}\right) \\ &\quad + \frac{C_4^{(\gamma+t)/2}}{2k} - \frac{C_1^\gamma}{k} - \frac{C_2^\gamma}{k^2} - \frac{C_3^\gamma}{k^3} \\ &\le \gamma + \frac{t_j}{2\gamma} + \frac{C_4^{(\gamma+t_j)/2}}{2k} - \frac{C_1^\gamma}{k} - \frac{C_2^\gamma}{k^2} - \frac{C_3^\gamma}{k^3} \end{aligned}$$

where the last inequality uses $\gamma \le \frac{1}{2}$ In addition to this estimation, for at most $k \cdot l$ n-grams can this difference be different from 0: if an n-gram never appeared, then the observed value $O_k^{i,j}$ is identical to the randomness $R^{i,j}$, and thus the entropy of both is the same. (*)

The full n-gram database which is leaked is the concatenation of the sums of the individual n-gram counts, i.e., the actual information leakage is

$$\begin{aligned} L &= I((\sum_{i=1}^{l} T_k^{i,j})_{1\le j\le N}; (\sum_{i=1}^{l} O_k^{i,j})_{1\le j\le N}) \\ &= H((\sum_{i=1}^{l} O_k^{i,j})_{1\le j\le N}) - H((\sum_{i=1}^{l} R_k^{i,j})_{1\le j\le N}) \\ &\le \sum_{j=1}^{N}\sum_{i=1}^{l}\Big(H(O_k^{i,j}) - H(R_k^{i,j})\Big) \quad (15) \end{aligned}$$

Now we estimate the constants from Equation (15) using that $\gamma, t \le 0.01$ and get:

$$\frac{l \cdot C_4^{(\gamma+t)/2}}{2k} \le \frac{1}{5.8\gamma \cdot k}, \qquad \frac{l \cdot C_1^\gamma}{k} \ge -\frac{5}{5.95\gamma \cdot k},$$

$$\frac{l \cdot C_2^\gamma}{k^2} \ge -\frac{1}{5lk^2\gamma^2}, \qquad \frac{l \cdot C_3^\gamma}{k^3} = -\frac{1}{360k^3l^2}.$$

Finally, from these estimations, from remark (*), and Equations (15) and (15) we get

$$\begin{aligned} L &\le \sum_{j=1}^{N}\sum_{i=1}^{l}\left(\gamma + \frac{t_j}{2\gamma} + \frac{C_4^{(\gamma+t_j)/2}}{2k} - \frac{C_1^\gamma}{k} - \frac{C_2^\gamma}{k^2} - \frac{C_3^\gamma}{k^3}\right) \\ &\overset{(*)}{\le} \frac{l}{2\gamma} + lk \cdot \left(\gamma + \frac{C_4^{(\gamma+t_j)/2}}{2k} - \frac{C_1^\gamma}{k} - \frac{C_2^\gamma}{k^2} - \frac{C_3^\gamma}{k^3}\right) \\ &\le 4k\gamma + \frac{1}{\gamma}\left(6.1 + \frac{1}{5k\gamma}\right) \end{aligned}$$

for $l = 4$. For a reasonable choice of parameters $\gamma = \frac{1}{1'000'000}$ and $k = 5'000'000$ this means that overall, the n-gram database can leak at most 6.2 million bits, or on "average" about 1.3 bits per password.

B Interval Estimation for Password Frequencies

As explained in Section 2.1, given the *true* probability of each password, we can easily build an ideal password meter by applying any monotonically increasing function to these probabilities. Furthermore, even if we only have an estimation of these probabilities the we could build an ideal password checker, as long as the relative ranking of the passwords remains unchanged.

Unfortunately, we do not in general the *true* probability of a password thus rendering this approach impractical. However, there exist certain classes of passwords for which it is possible to give a good probability estimation that can be used as a ground truth for measuring the accuracy of our construction in measuring password strength.

One such class is the most common passwords in the RockYou dataset. Intuitively, with a large dataset of 32.6 million passwords, we should be able to obtain a good probability estimation of, at least, the most common passwords. This probability estimation could then be used to be an ideal password meter for these common passwords. For example, the most common password in the dataset (`123456`) has been chosen 290729 times, while the second most common (`12345`) has been chosen 79076.

If we estimate the probability of `123456` using the empirical frequency we might introduce some small error (the mathematical treatment is explained later), however, we can be confident that `123456` will remain the most common password. This is not true for passwords with much lower counts though. For example, the password `zxcvbnmp` has been chosen 3 times in the dataset and it is the $288078th$ most common password together with almost $400'000$ other passwords, due to ties. However, if `zxcvbnmp` had been chosen only 2 times, it would be the $1136277th$ most common password. In this case a small difference in the probability estimation can make a big difference in the guessing entropy of a password.

In order to measure the accuracy of a password meters we would like to find a subset of common passwords for which we can estimate, with high confidence, the probability and therefore the ranking. By assuming that each passwords x binomially distributed, with *unknown* probability $p(x)$. With this interpretation, each password x in our dataset is binomially distributed with N number of trials and unknown probability $p(x)$. In our case, N is the number of times a password in our RockYou dataset was chosen, i.e., 32.6 million. Given the counts of each password x in our dataset, we can estimate its probability as $\hat{p}_x = \frac{count(x)}{N}$. However, this is only an estimation of the *true* probability mentioned above. The error of this estimation depends on $\hat{p}_x$ and N and, for example, larger N lead to better estimations. The sampling error of $\hat{p}_x$ can be estimated using Wilson score interval:

$$W(\hat{p}_x, N) = \frac{\hat{p}_x + \frac{z^2_{1-\alpha/2}}{2n} \pm z_{1-\alpha/2}\sqrt{\frac{\hat{p}_x(1-\hat{p}_x)}{N} + \frac{z^2_{1-\alpha/2}}{4N^2}}}{1 + \frac{1}{n}z^2_{1-\alpha/2}}$$

The relative ranking of two passwords x and x', with estimated probability $\hat{p}_x > \hat{p}'_x$, remains unchanged as long as $W(\hat{p}_x, N) > W(\hat{p}'_x, N)$. The confidence level depends on the z-score $z_{1-\alpha/2}$, for example if $z_{1-\alpha/2} = 2.58$ than the ordering is correct with 99 % probability.