On the LPSE Password Meter's Discrepancies among Different Datasets

Agnieszka Rucka and Wojciech Wodo
Department of Fundamentals of Computer Science, Wroclaw University of Science and Technology, Wybrzeze Wyspianskiego 27, Wroclaw, Poland

Keywords: Password, Password Meter, Password Strength, Security, LPSE, PCFG, zxcvbn.

Abstract: The global dataset constantly grows, along with the number of online accounts. More and more data breaches occur, putting users' data at risk. At the same time, users still commonly choose weak passwords. It has been shown that password strength meters can contribute to better user choices. As the problem of password strength estimation is nontrivial, a number of solutions have been proposed. One of them is the *LPSE* (Guo and Zhang, 2018), which, according to its authors, shows very promising performance. However, we observed a significantly worse performance of *LPSE* in a different dataset. In this paper we present an extensive investigation of these discrepancies. We describe our recreation of the original experiment and confront the obtained results with the original. We analyze the data distribution in our dataset, and compare performance of the *LPSE* with the widely known lightweight password meter *zxcvbn*. Lastly, we discuss possible reasons for observed discrepancies (including methodological differences) and draw final conclusions.

1 INTRODUCTION

Over recent years the use of online services skyrocketed, bringing a significant rise in the number of accounts owned by a single person. Additionally, more often than before, these accounts contain personal or financial information, such as e-mail and postal addresses, phone numbers, or credit card credentials. Along with the growth of the global dataset, we observe a rising tendency in the yearly number of data breaches, with 2019's record in the number of leaked data records (Risk Based Security, 2020).

Analyzes of disclosed password sets show that, despite having contact with password-building advice on multiple occasions, users still tend to make very weak, predictable choices. The list of the 10 most popular passwords of 2019, published by National Cyber Security Centre (UK), is a vivid example of that: *123456, 123456789, qwerty, password, 1111111, 12345678, abc123, 1234567, password1, 12345*. The combination of sensitive data and weak protection poses serious risk to the users.

1.1 Motivation

As Stephen Furnell and Rawan Esmael's research suggests, users presented with a live feedback on the strength of their password during its creation, tend to choose stronger phrases (Furnell and Esmael, 2017). A reliable password meter can therefore be an important element of security framework. At the same time, we observe that practices regarding measuring password strength significantly differ among service providers. There are no common standards for password policies, even among the industry leaders. Simple sets of rules are not reliable, and often accept very weak choices. Password strength measuring is certainly a nontrival problem. Although multiple solutions have been proposed so far, this field still requires research and further development.

1.2 Contribution

The initial goal of our research was proposing a modification to a password strength measuring algorithm *LPSE* (Guo and Zhang, 2018). However, during our experiments, we observed severe discrepancies in the performance of the unmodified original. This paper presents a thorough analysis of this results' mismatch. Our contribution is providing a review for an existing password strength measuring algorithm.

2 ANALYSIS OF THE PROBLEM OF PASSWORD STRENGTH

This section is an introduction to the problem of passwords strength measuring. It aims to familiarize the

Rucka, A. and Wodo, W.
On the LPSE Password Meter's Discrepancies among Different Datasets.
DOI: 10.5220/0010766900003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 255-263
ISBN: 978-989-758-553-1; ISSN: 2184-4356

reader with challenges in this field, as well as common solutions. It provides an overview of existing practical approaches, along with exemplary algorithms, three of which (*LPSE*, *zxcvbn*, *PCFG*) are relevant to the later discussion.

2.1 Defining Password Strength

First, notice that the strength of a password is not an objective value. In order to be able to measure it, we need to establish some quantitative metric. We intuitively agree that a strong password should take long to break. Therefore, time could be our first candidate. However, running time is usually incomparable between machines, due to a number of technical issues. Having this in mind, we propose an algorithmics-like approach - to understand the time as the average number of operations performed. As we do not know the algorithms used by the attacker, let us express the strength of a password by the average number of guesses needed to break it.

2.2 Real-world Challenges

In an ideal setting, passwords are generated randomly. In this case, bruteforce attack is the best available method of guessing. The average number of guesses needed is proportional to the string's length and space size. However, this is a rather rare situation. More commonly, passwords are generated pseudorandomly (e.g. with password managers). Then, aside from length and character set, we need to consider the quality of the PRG used. This can be achieved by calculating the password's entropy, understood in this case as a ”measure of the source's unpredictability”. However, the most common setting in which password strength is measured are human-provided passwords. These are rarely random-looking, and are often based on existing words. As in human languages letters have different frequencies of occurrence and some sequences of letters are more common than others, estimation of human-created passwords strength is a complicated task. Basic entropy calculation is in this case insufficient. There is no universal remedy for this, and several approaches have been proposed.

2.3 Overview of Existing Approaches

The existing approaches can be, according to Yimin Guo and Zhenfeng Zhang (Guo and Zhang, 2018), divided into three main groups:

2.3.1 Rule-based Approach

The input passwords are checked against a set of rules that a strong password should follow. The overall score is calculated awarding positive choices and penalizing negative. This approach was commonly found in the early, rather naive password meters, which employed some simple rules, such as use of one capital letter, one number, etc. However, when the rules are properly chosen, the meter might have satisfying performance. An example of a good rule-based meter is the *zxcvbn* (Wheeler, 2016).

Example - the *zxcvbn* algorithm
It detects weak patterns in the password, and divides it into substrings accordingly. The weak patterns identified are:

- repeat (e.g. 111, abcabc)
- sequence (e.g. 12345)
- reverse (e.g. drowssap)
- keyboard (e.g. qwerty, zxcvbn)
- date (e.g. 05122020)

The score of each of substrings is calculated separately. If a dictionary entry is found, it is scored based the word's rank in the dictionary. The remaining parts of the password are considered random strings and their score is defined by their entropy. The overall score of the password is a sum of all the substrings' scores.

2.3.2 Similarity-evaluation Approach

The core of this method is the concept of an ideal strong password. A model of such universally strong password is proposed and described. The strength meter evaluates similarity of input to the universally strong password. The closer the input password to the ideal one, the stronger it is. This approach is represented by the *LPSE* algorithm (Guo and Zhang, 2018).

Example - the *LPSE (lightweight password strength estimation)* algorithm
In this method the measured password is encoded as a vector of the string's characteristics. The universally strong password, also expressed as such vector, serves as a reference. For this pair of vectors, two metrics are calculated: *cosine-length similarity* and *password distance similarity*. The pair of values is the translated to the overall score of the input. The characteristics' vector calculation is tweaked by a small set of rules that detect highly predictable sequences.

2.3.3 Probability-based Approach

This class of methods refers directly to number of guesses needed to break a password, rather than estimating its entropy. Firstly, a model is prepared, basing on a list of plaintext passwords. Later, it is used to aid a guessing algorithm in determining the input password. The strength of the password is defined by the number of guesses needed to figure it out correctly.

As the class is broadly defined, there are several approaches that can be identified as its representatives. Here, the two most well-known will be presented. As they are significantly more complex than the previous two algorithms, their description will be more general. For a deeper insight, please refer to the original works.

Example - Markov chains and finite automata method (Narayanan and Shmatikov, 2005)
This method is based upon the following assumptions:

- humans choose passwords to be memorable, meaning that they either consist of existing words of some language or resemble them. Therefore, they tend to choose letters with probabilities close to their frequency of occurrence in the chosen language. These can be predicted with Markov chains-based methods.
- the use of numbers and special characters in passwords usually follows a few popular patterns, which are deployed as "mangling rules" (like in popular password cracker *John the Ripper*). These can be generalized by finite automata.

The authors of the solution present several algorithms for generation of a wordlist for a dictionary attack, basing on the Markov chains, the finite automata and, finally, a mixed approach. Their method is reported to outperform the Oechslin's "rainbow" attack (Oechslin, 2003), widely regarded as the fastest method for large spaces exhaustion.

Example - *PCFG* (probabilistic context-free grammar based meter) (Weir et al., 2009)
The method itself is not a password meter, but a password guesser. Yet, it can be used for estimating passwords strength as well. A list of passwords in plaintext is used to automatically extract common structure patterns in them and create a probabilistic context-free grammar. From the grammar, mangling rules for the guesser are extracted. Then, a dictionary attack is performed, using these rules to modify dictionary entries. This method of creating mangling rules (as opposed to setting them manually, or relying on a default set, as in standard password crackers) leads to significantly better performance - in the original paper, up to 129% more passwords were cracked than by *John the Ripper*.

3 THE LPSE ALGORITHM

This section presents a description of the *LPSE* algorithm. It summarizes Chapter 3 of the original work (Guo and Zhang, 2018). Its goal is to provide the Reader with understanding of the method, as it will be analysed further in this paper.

3.1 Design Objectives

The *LPSE* authors' goal was to provide a lightweight password meter, suitable for use on the client-side, that would show better performance than existing leading solutions, such as *zxcvbn*. The algorithm was designed to be computationally lightweight, need a limited amount of additional data (such as lengthy dictionaries), and be easy to implement.

3.2 Vectors of Characteristics

In this method, the passwords are represented as vectors $\alpha = (x_1, x_2, x_3, x_4, x_5)$, where x_i's represent, respectively: digits, lowercase letters, uppercase letters, special characters and password length. As uppercase letters and special characters tend to be included in strong passwords, their use is awarded with a higher score. On the contrary, as doubling letters or choosing consecutive characters is a weak choice, it is penalized - the overall score for the sequence is equal to score for a single building block. The general rules, as described by the authors of the solution, are shown in Figure 1. As the users tend to follow some weak patterns when creating passwords, these basic calculations are tuned up by detecting and penalizing them. The list of the weak patterns identified by the algorithm is presented in Figure 2.

3.3 Universally Strong Password

As stated in the previous section, an ideal strong password should be randomly generated and sufficiently long. The authors of *LPSE* propose an 18-characters long sequence. Let us assume that the characters are chosen uniformly at random from a 94-piece set consisting of 10 digits, 26 lower- and 26 uppercase letters, and 32 special characters. Then, characters from all groups should appear proportionally, namely: 2

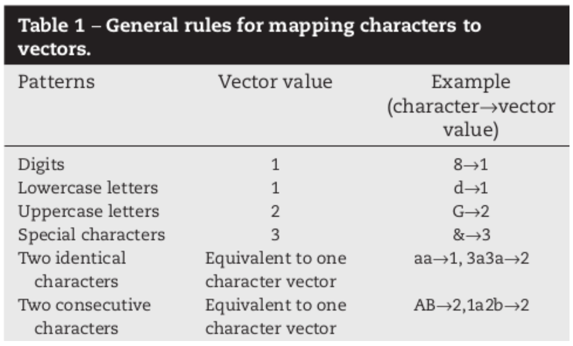

Table 1 – General rules for mapping characters to vectors.

Patterns	Vector value	Example (character→vector value)
Digits	1	8→1
Lowercase letters	1	d→1
Uppercase letters	2	G→2
Special characters	3	&→3
Two identical characters	Equivalent to one character vector	aa→1, 3a3a→2
Two consecutive characters	Equivalent to one character vector	AB→2,1a2b→2

Figure 1: Basic scoring rules, Table 1 from the original work (Guo and Zhang, 2018).

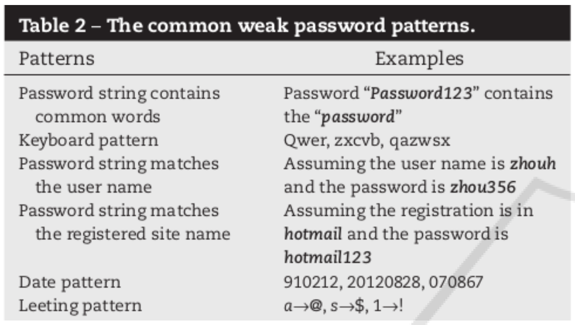

Table 2 – The common weak password patterns.

Patterns	Examples
Password string contains common words	Password "***Password123***" contains the "***password***"
Keyboard pattern	Qwer, zxcvb, qazwsx
Password string matches the user name	Assuming the user name is ***zhouh*** and the password is ***zhou356***
Password string matches the registered site name	Assuming the registration is in ***hotmail*** and the password is ***hotmail123***
Date pattern	910212, 20120828, 070867
Leeting pattern	*a*→@, s→$, 1→!

Figure 2: Common patterns detected by the algorithm, Table 2 from the original work (Guo and Zhang, 2018).

digits, 5 lower- and 5 uppercase letters, and 6 special characters.

When choosing 4-character strings fully at random, *abcd* and *5K%p* could be chosen with equal probability. However, an adversary optimized for use with human-generated passwords would easily break the first one. Therefore, we must additionally assume that the universally strong password does not "accidentally" follow any weak pattern. The authors of *LPSE* propose to assume that no two adjacent characters come from the same group. We will use this characteristic again later. Under this assumption, the vector for a universally strong 18-characters long password is $(2, 5, 10, 18, 18)$.

3.4 Scoring

To provide better accuracy, the algorithm aims to consider three aspects of password composition:

- structure (type and proportions of characters contained)
- length
- editorial distance (number of transformations needed to transform it into the universally strong password)

To capture all these characteristics, two measures are used: the *cosine-length similarity* for length and structure, and *password distance similarity* for the editorial distance. For their definitions consult the original *LPSE* paper. The pair of values is translated into a score expressed within a 3-level scale (*weak*/*medium*/*strong*). The algorithm's authors proposed classification thresholds, based of an analysis of a leaked passwords dataset. The two similarities were calculated for passwords previously identified as strong and weak, and intervals were determined accordingly. Figure 3 presents the thresholds established, s_c being *cosine-length similarity* and s_p - *password distance similarity*.

$$T_\alpha = (s_c(\alpha), s_p(\alpha)) = \begin{cases} \textit{strong} & \text{if } s_c(\alpha) \geq 0.4 \text{ or } s_p(\alpha) \geq 0.55 \\ \textit{strong} & \text{if } 0.3 \leq s_c(\alpha) < 0.4 \\ & \text{and } 0.4 \leq s_p(\alpha) < 0.55 \\ \textit{weak} & \text{if } s_c(\alpha) \leq 0.19 \text{ or } s_p(\alpha) < 0.35 \\ \textit{medium} & \text{otherwise} \end{cases}$$

Figure 3: Classification thresholds for password scores, unnumbered definition from Chapter 4.3 of the original paper (Guo and Zhang, 2018).

4 DESCRIPTION OF THE EXPERIMENTS PERFORMED

This section describes the performance evaluation experiments on the *LPSE*. First, it briefly summarizes the original experiment. For full description consult Chapter 4 of (Guo and Zhang, 2018). Then, our experiment is presented. Resources used by our team can be found at: `https://github.com/arucka/mod_LPSE`. Descriptions of the experiments are relevant for understanding the observed discrepancies in results, discussed later.

4.1 The Original Experiment

The aim of the original experiment was to compare *LPSE*'s performance with other lightweight passwords meters. Believing that a good lightweight meter should yield similar results to a more technologically advanced one, the authors chose to use *PCFG* as the point of reference regarding the "real" strength of a password. The other two lightweight meters under comparison were *zxcvbn* and *PM*[1].

The dataset was composed of passwords coming from various leaks, mainly of Chinese and English origin. It was divided into three subsets, the first of

[1]http://www.passwordmeter.com/

which was used to train a *PCFG* instance, the second - to determine *LPSE* classification thresholds, and the third one - to compare the performance of the lightweight meters. Each password in the third set was scored with all four meters. The scores granted by each lightweight meter to passwords identified as *weak* and *strong* by the PCFG were summarized. As it can be observed in Figures 4 and 5, the *LPSE*'s scoring was the most consistent with *PCFG*'s.

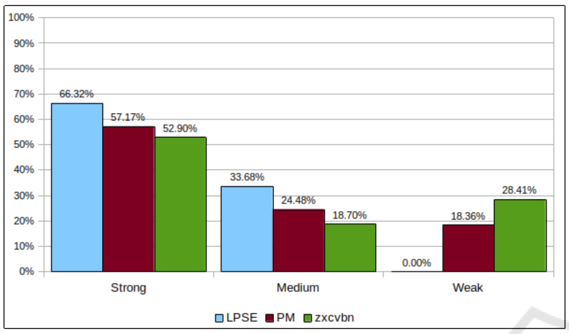

Figure 4: Classification of strong passwords by *LPSE*, *PM* and *zxcvbn*. For numerical values see Figure 3 from the original paper (Guo and Zhang, 2018).

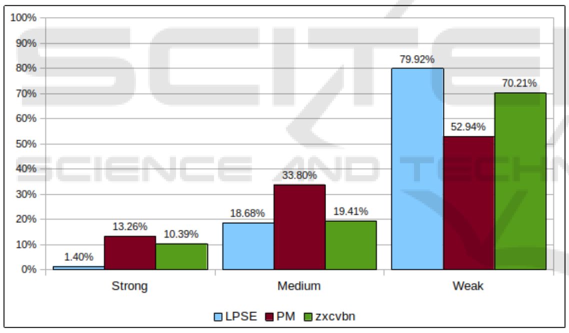

Figure 5: Classification of weak passwords by *LPSE*, *PM* and *zxcvbn*. For numerical values see Figure 4 from the original paper (Guo and Zhang, 2018).

4.2 Our Experiment

The general outline of our experiment was the same as the original. However, we introduced several modifications. As the original dataset was unavailable, as well as some of the password sets included, we prepared our own. We chose password leaks with a more international origin, and prepared two datasets: *small* (199 032 passwords) and *large* (527 163 passwords). All steps of the experiment were performed on both of them identically. Originally, our goal was to propose a modification for the *LPSE*, therefore we did not include other lightweight meters, just the original and modified *LPSE*. Courtesy of Dr. Guo, we received the original code, which we slightly corrected due to compilation errors. We also corrected one minor discrepancy in implementation (one of the bonuses included in the algorithm description was not included in the code). As we did not want to re-establish *LPSE* classification thresholds, we skipped this stage of the original experiment, and divided our datasets in only two subsets.

Originally, estimation of strength with the *PCFG* was performed using a Monte Carlo simulation method (Dell'Amico and Filippone, 2015). The authors of the *LPSE* did not provide any details on their application of this method. Therefore, we chose to use Dr. Weir's implementation of the *PCFG* and a Monte Carlo simulation script by GitHub user `cwwang15`, available in the same repository.[2]

The original and modified *LPSE*, showed almost identical performance in both our sets. However, it was significantly different from the results reported in the original paper. Figures 6 and 7 present a comparison of performance of the *LPSE* in the original research and our experiment.

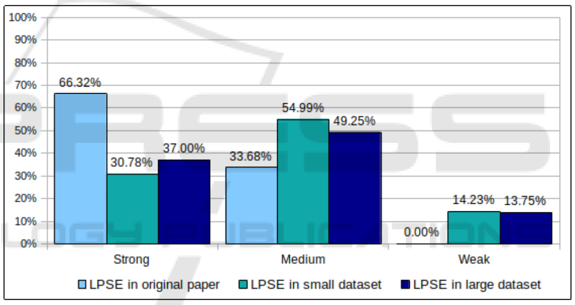

Figure 6: Classification of strong passwords by the *LPSE* in the three datasets.

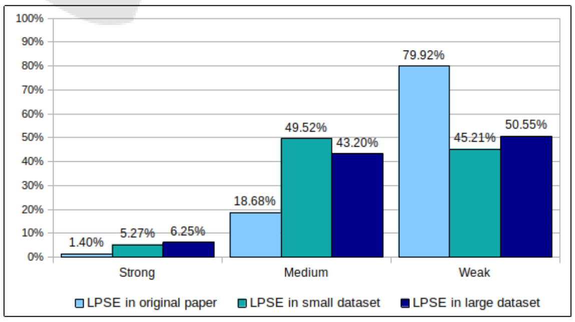

Figure 7: Classification of weak passwords by the *LPSE* in the three datasets.

Observe that while *LPSE*'s performance in both our datasets (*small* and *large*) is similar, it is significantly worse than the one declared in the original pa-

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

per. This turned our research towards an investigation of these discrepancies. It is described in the next section. From that moment on, our modification of the *LPSE* is not considered anymore, and any mention of the *LPSE* refers to the original version.

5 INVESTIGATION ON LPSE'S RESULTS

This section investigates the discrepancies in the claimed and observed performance of the *LPSE*. First, the cause of the differences is examined. Distribution of our dataset is analyzed and compared to the original dataset's. Then, the possible reasons for differences among them, including variances in methodology, are discussed. This analysis leads to final conclusions, provided in the next section.

5.1 Analysis of the Discrepancies

The starting point for our analysis is the high rate of misclassification of both strong and weak passwords as *medium*, which can be observed in Figures 6 and 7. As the *LPSE*'s classification thresholds were established based on a specific dataset, we propose a hypothesis that they are overfit to it. To verify this, let us compare the original thresholds (Figure 8) with the distribution of passwords from our *large* dataset, regarding their strength as assigned by the PCFG. Figure 9 presents this distribution, color-coded depending on their PCFG-determined strength. The original *medium* interval is outlined in blue for comparison.

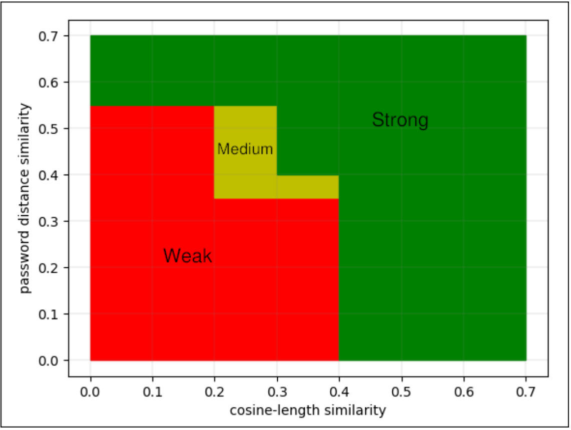

Figure 8: Division of the plot of relationship between *cosine-length similarity* and *password distance similarity* by *LPSE*'s original thresholds.

Observe that the *medium* interval is densely occupied by strong passwords, and that the plot points

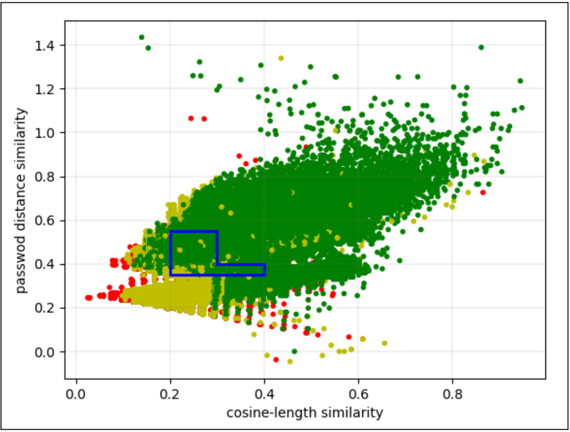

Figure 9: Distribution of passwords from the *large test set* in terms of their *cosine-length similarity* and *password distance similarity*. Colour indicates strength of the password as measured by *PCFG*: green-strong, yellow-medium, red-weak.

seem to cover one another. Therefore, to obtain better visibility, let us plot the strong and weak passwords separately - see Figures 10 and 11.

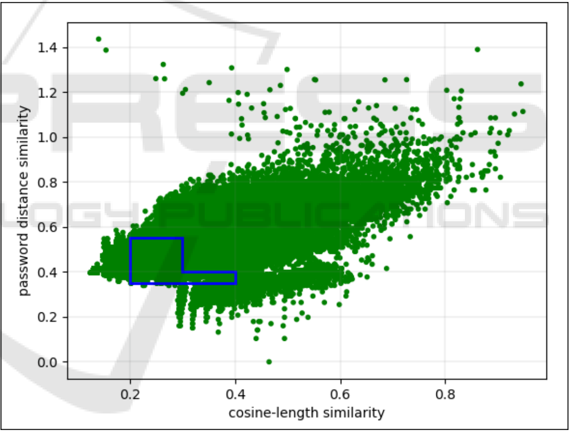

Figure 10: Distribution of strong passwords from the *large test set* in terms of their *cosine-length similarity* and *password distance similarity*.

Notice that the classification thresholds indeed do not fit the characteristics of our dataset. However, this is not the matter of simple overfitting, which could be easily solved by correcting the thresholds' values. The area common for both strong and weak passwords is extremely large, and the passwords are widely spread. This is very different from the distribution presented in the original paper, where strong and weak passwords occupied much smaller areas of the plot, which were almost disjoint from one another. In the original dataset, the distinction between strong and weak passwords was clear. In our case, it seems impossible to choose new classification thresholds so

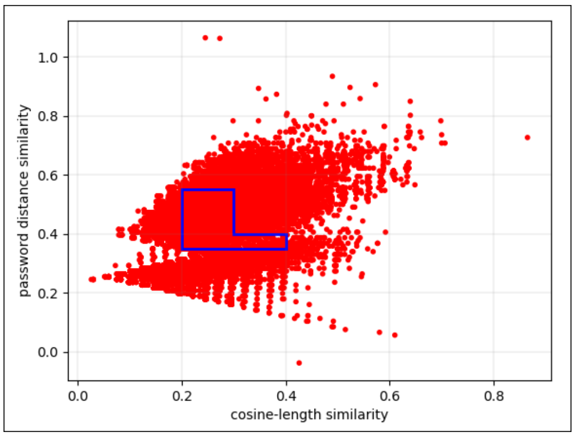

Figure 11: Distribution of weak passwords from the *large test set* in terms of their *cosine-length similarity* and *password distance similarity*.

that the performance will be even close to the original one. However, as a scatter plot does not provide clear data on the density of the passwords, let us verify this suspicion by re-plotting the same data as density heatmaps. This will let us observe, whether the high-density areas of strong and weak passwords align. The heatmaps are presented in Figures 12 and 13. The plot area most densely populated by strong passwords is outlined in both Figures for comparison.

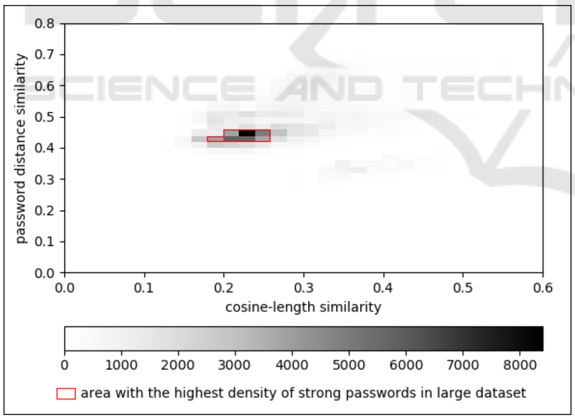

Figure 12: Distribution of strong passwords from *large set*. The red outline marks the area with the highest density of strong passwords.

Observe that the most dense groups of strong and weak passwords indeed overlap, confirming our suspicion. It is impossible to propose new thresholds, that will offer effective classification. As this might be a sign of a major design flaw in the *LPSE*, we performed an investigation on the possible reasons for such results, to ensure no methodological errors were made during our research. The next section covers this analysis.

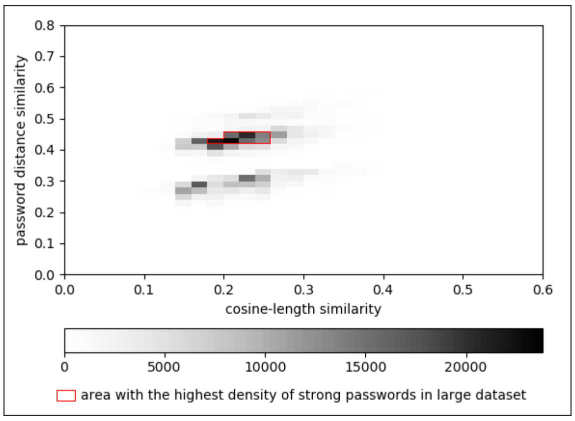

Figure 13: Distribution of weak passwords from *large set*. The red outline marks the area with the highest density of strong passwords for comparison.

5.2 Analysis of the Observed Data Distributions

Considering the two experiments on *LPSE* (original and ours), we propose the following three hypotheses on the possible reason for the observed differences in password distributions:

1. There were no significant differences between the two experiments, and the obtained results are correct - the dataset used in the original paper was peculiar.
2. The implementations of the *LPSE* are different in the two experiments, which leads to yielding different values of the two similarities for the same password.
3. The use of the *PCFG* in the two experiments was in some way different, leading to it assigning different strength scores to the same password.

The hypotheses will now be discussed in more detail.

5.2.1 Hypothesis 1: The Original Dataset Was Peculiar

As mentioned earlier, we encountered some obstacles preventing us from repeating the original experiment verbatim. A definitive verification whether the experiments were synonymous, would require repeating our experiment on the original dataset, or the original experiment on our datasets. As this is currently impossible, we cannot confirm nor refute this hypothesis.

5.2.2 Hypothesis 2: The LPSE Implementations Are Different

The *LPSE* implementation we received from Dr. Guo needed several code adjustments to compile, and a small correction. This leads to a suspicion that it could be a non-final version, and the implementation of the *LPSE* used in the original research was somehow different from ours. We cannot definitively verify that, however, we performed a code review for a better insight. We believe the code correctly implements the *LPSE* algorithm described in the paper. Therefore, there is no reason for the implementations to be significantly different. To rule out our small correction as the source of the mismatch, we also tried to use a non-corrected *LPSE* (meaning with no improvements other than needed to compile), yet its performance was worse compared to the corrected one. This does not definitively disprove the hypothesis, however suggests that it is rather unlikely.

5.2.3 Comparison to zxcvbn

As the first and second hypotheses cannot be definitively confirmed nor refuted, we decided to verify *zxcvbn*'s performance in all three datasets for comparison. Having no access to the original dataset, we had to rely on the results provided by the authors of the *LPSE*. For our datasets, we chose an officially approved Python port of *zxcvbn* (originally written in JavaScript), available on GitHub[3] and in official repository for the `pip` command. The method of evaluating performance was analogous to *LPSE*'s. The performance of *zxcvbn* in the three sets is presented in Figures 14 - 15 .

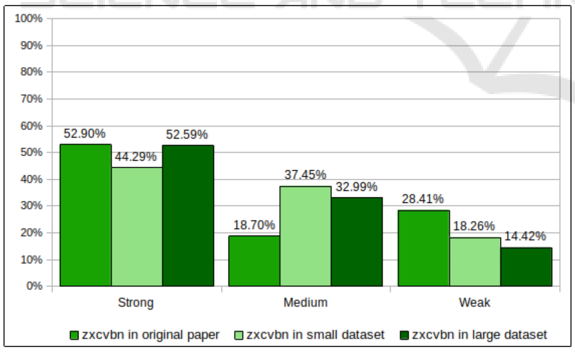

Figure 14: Classification of strong passwords by the *zxcvbn* in the three datasets.

First, observe that *zxcvbn*'s performance is more consistent than *LPSE*'s. This suggests that the latter might either be more sensitive to specific characteristics of the evaluated sets (supporting hypothesis #1) or that its implementations used in the original and our experiment indeed differed (supporting hypothesis #2). Although this does not speak in favour of any of the hypotheses, it is worth noting that de-

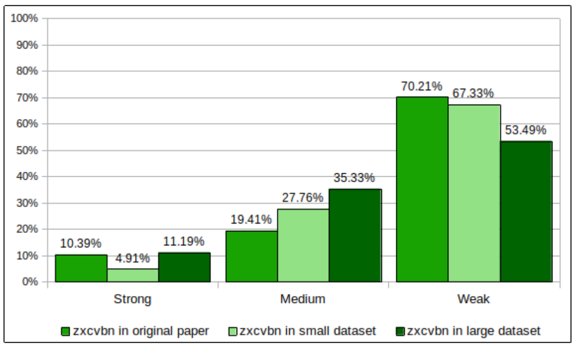

Figure 15: Classification of weak passwords by the *zxcvbn* in the three datasets.

spite the less distinctive distribution of our datasets, *zxcvbn*'s results are satisfying. It outperforms *LPSE* in correctly classifying strong and weak passwords, which contributes to enhanced security and building correct intuitions in users. On the other hand, it shows slightly higher misclassification rate of strong passwords as *weak* than *LPSE*, which might cause unnecessary rejection of good passwords, degrading usability. However, the claim that *LPSE* outperforms *zxcvbn*, which was made by the original authors, should be considered undermined and further verified in other datasets.

5.2.4 Hypothesis 3: The PCFG Implementations Differ

The use of the *PCFG* in the original paper is described very vaguely. Apart from citing an article describing the Monte Carlo simulation algorithm (Dell'Amico and Filippone, 2015), it does not cover any details. On our second attempt to contact the authors of the *LPSE* paper we were not provided with any details as well. However, in the further stage of our research, we managed to retrieve the original script for performing the Monte Carlo estimation by Dell'Amico and Filippone. While we cannot confirm that it was used in the original evaluation of the *LPSE*, it can certainly be considered correct to use it. Therefore, we performed a reevaluation of the *LPSE* for our *large* dataset. The results, presented in Figures 16 and 17, support the initial results of our experiment, obtained with the script by GitHub user *cwwang15*. Therefore, we believe that the implementation of PCFG and Monte Carlo estimation used in our experiment was not the cause for discrepancies in the results, and this hypothesis should be rejected.

[3]https://github.com/dwolfhub/zxcvbn-python

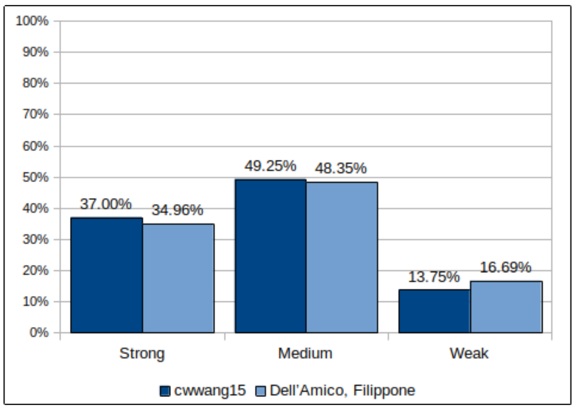

Figure 16: Classification of strong passwords from the *large* set, with `cwwang15`'s and Dell'Amico and Filippone Monte Carlo scripts used.

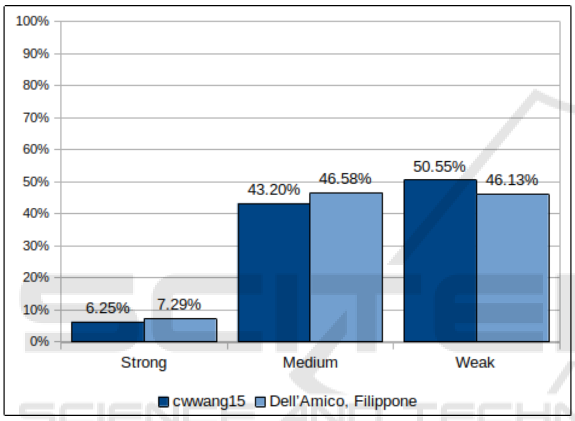

Figure 17: Classification of weak passwords from the *large* set, with `cwwang15`'s and Dell'Amico and Filippone Monte Carlo scripts used.

6 CONCLUSIONS

In this paper, we presented an overview of the problem of password strength measuring and discussed performance issues of the *LPSE* algorithm. Our experiment was explained, and observed discrepancies in performance were investigated. The findings suggest, that for our datasets, passwords' characteristics *LPSE* was based upon do not hold. The mismatch might be a result of either methodological differences between the experiments, or signify major design flaws in the *LPSE*. We rejected discrepancies in using the *PCFG* as the source of the problem, as we successfully repeated our results when using another implementation. We could not confirm, whether the original implementation of the *LPSE* was analogous to ours. Lastly, we also could not verify the original results, as the original dataset is not available anymore. We believe that our findings put *LPSE*'s quality in question, and that it should be further examined before it could be recommended for practical use. Currently, we suggest to use *zxcvbn* as the better choice for a lightweight password meter, as it is better known and shows more consistent results among datasets. It also provides better protection from using too weak passwords. We highly encourage the Readers to verify our experiment's results. The resources helpful to recreate it are provided in the GitHub repository of this research (`https://github.com/arucka/mod_LPSE`).

REFERENCES

Dell'Amico, M. and Filippone, M. (2015). 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. Association for Computing Machinery.

Furnell, S. and Esmael, R. (2017). Evaluating the effect of guidance and feedback upon password compliance. *Computer Fraud And Security*, 2017(1):5 – 10.

Guo, Y. and Zhang, Z. (2018). Lpse: Lightweight password-strength estimation for password meters. *Computers And Security*, 73:507 – 518.

Narayanan, A. and Shmatikov, V. (2005). 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. Association for Computing Machinery.

Oechslin, P. (2003). Making a faster cryptanalytic time-memory trade-off. In Boneh, D., editor, *Advances in Cryptology - CRYPTO 2003*, pages 617–630, Berlin, Heidelberg. Springer Berlin Heidelberg.

Risk Based Security (2020). 2019 year end report. data breach quickview.

Weir, M., Aggarwal, S., d. Medeiros, B., and Glodek, B. (2009). Password cracking using probabilistic context-free grammars. In *2009 30th IEEE Symposium on Security and Privacy*, pages 391–405.

Wheeler, D. L. (2016). zxcvbn: Low-budget password strength estimation. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 157–173, Austin, TX. USENIX Association.