International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101

A Multimodal Metric for Password Strength Estimation

Prof. Amruta Hingmire[1], Prof. Sinju Saliya[2]
[1,2]JSPM's Rajarshi Shahu College of Engineering, Pune

Abstract: A research on Password strength estimation always comes with the new approach for estimating a score indicating its strength, but there is no known single metrics which measures this strength for every possible password. In this paper we have analyzed several imperfect individual metrics to benefit from their strong points in order to overcome many of their weaknesses. We have uncovered a new way that combines these metrics in one unique multimodal metric for password strength detection. The final metric can be applied to password models like Attack-based, Heuristic-based & Probabilistic-based methods.

General Terms : Password security, strength metrics, multimodality, Markov chains, password policies, privacy
Keywords: Password strength metric, multimodal approach for password, password meter metric

I. INTRODUCTION

The answer to the question, how secure is passwords? is not clear: it depends on the password. Certain passwords are, stronger or more secure, than others. As such, password strength is commonly unstated as a way to measure how complicated it is to break passwords. To understand what makes users choose more (or less) secure passwords, and for constructing password, password strength metrics are used. Which takes as input a password and output a score related to the strength of that password. The purpose of password strength metrics is the identification of weak passwords in order to support password policies. A number of major websites like Dropbox, Gmail and eBay, rely on some kind of an indicator to indicate the strength of the password to the user during registration. The indicator serves as a good reminder for the user as to the level of difficulty to crack the password.

Common way to impose the strategy is the real-time checking of the password strength during the password selection process by the users. Very basic algorithm analyzes the string, gives bonuses for extra length, presence of numbers, symbols and upper case letters and penalties for letter or number only inputs. First we check the length of the input string. If it's greater than the minimum length, give it a base score of 50. Else make it 0. Next iterate through each character of the string and check if it is a symbol, number or upper case letter. If so, make a note of it. This good practice was already promoted by researchers in the field of password security in the early 1990's [3]: should the strength fall below an acceptable threshold, the user will be requested to provide a different password. The effectiveness of password strength metric in estimating the actual resistance of passwords will have a direct impact on the level of security for both the users and the provider [4].

A natural approach for password models is to use whole string Markov chains. Markov chains are used in the template based model for assigning probabilities to segments that consists of letters. Castelluccia et al. [5] proposed to use whole-string Markov models for evaluating password strengths, without comparing these models with other models.

Weir et al. [6] developed a template-based password model that uses Probabilistic Context-free Grammar. The guess number of a password according to a password model is defined to be the rank of the password in the order of decreasing probability. To compare two sets of passwords, one plots the number of guesses vs. the percentage of passwords cracked by the corresponding number of guesses in the testing dataset. Such guess-number graphs are currently the standard tool in password research. Plotting such graphs, however, requires the computation of guess numbers of passwords, which is computationally expensive.

Current password strength estimation methods may be classified into three main groups:

Attack-based methods. Such methods give a measure of the resistance of passwords depending on the time taken by a specific attack (or set of attacks) to break it [7][8]. The longer it takes for the attack to break the password, the stronger it is.

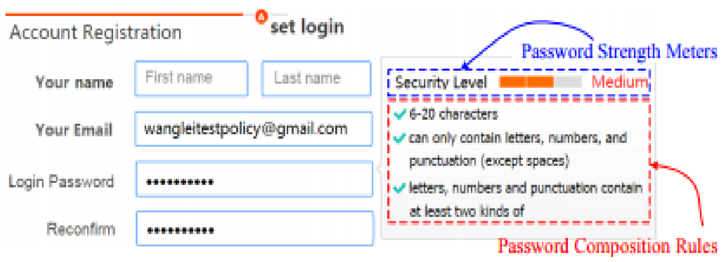
Heuristic-based methods. These methods focus on providing a measure of the password complexity based on heuristics [9]. The *de-facto* standard for this type of methods is the NIST 800-63 published in 2004 and updated in 2012. It proposes to measure password strength in entropy bits, following Shannon's Information Theory [10], on the basis of some simple convention such as the length of the password and the type of characters or symbols used (e.g., low-case, upper-case, or digits).

Probabilistic-based methods. A probabilistic password model assigns a probability value to each string. Such models are useful for research into understanding what makes users choose more (or less) secure passwords, and for constructing password strength meters and password cracking tools. Guess number graphs generated from password models are a widely used method in password research. Searching to address the shortcomings of the previous techniques, new methods for password strength estimation have emerged based on the statistical evaluation of passwords [11].

Password strength is not a universal value. Rather, it is very much dependent on the circumstance or on the website in which it is used. That is, the exact same password can have a significantly different strength depending on the application where it is used. For example, a Hungarian word used to login to a computer in an Italian-based company can be a quite strong password. However, that same password, used in a Hungarian-based context would most likely be regarded as weak.

II. INTRODUCTION TO PASSWORD GUESSING ATTACKS

The main objective of strength metric is to estimate the guess of a password. This way, in many circumstances, users purposely select passwords easy to remember and, as a result, easy to guess [12]. One example is brute-force cracking, in which a system or a utility tries *every* possible key or password until it make it. More general methods of password cracking, such as dictionary attacks, pattern checking, word list substitution, etc. attempt to reduce the number of test required and will usually be attempted before brute force. Higher password bit strength exponentially increases the number of candidate passwords that must be checked, on average, to recover the password and reduces the likelihood that the password will be found in any cracking dictionary Password guessing attacks take advantage of this predictability to give preference and test first those passwords that are more likely to be selected by a person in order to speed up the guessing process.



Account Registration
set login

Your name	First name	Last name
Your Email	wangleitestpolicy@gmail.com	
Login Password	••••••••••	
Reconfirm	••••••••••	

Security Level Medium
✓ 6-20 characters
✓ can only contain letters, numbers, and punctuation (except spaces)
✓ letters, numbers and punctuation contain at least two kinds of

Figure 1: Typical password strength Estimator of registration form.

A rainbow table is a list of pre-computed hashes - the numerical value of an encrypted password, used by most systems today - and that's the hashes of all possible password combinations for any given hashing algorithm mind. The time it takes to crack a password using a rainbow table is reduced to the time it takes to look it up in the list. Phishing: There's an easy way to hack: ask the user for his or her password. A phishing email leads the unsuspecting reader to a faked online banking, payment or other site in order to login and put right some terrible problem with their security.

III. INTRODUCTION TO PASSWORD CRACKING TOOLS

Zxcvbn

zxcvbn, a library for measuring password strength offered by Dropbox. It is implemented in JavaScript. zxcvbn has a dictionary of over a hundred thousand words that are commonly used for passwords. It evaluates a password by matching it against all combinations of these words. Matching with L33T substitutes and close key strings scored high points for zxcvbn.

```
# n: password length
# c: password cardinality: the size of the symbol space
#  (26 for lowercase letters only, 62 for a mix of lower+upper+numbers)
entropy = n * lg(c) # base 2 log
```

zxcvbn considers qwER43@! weak because it's a short QWERTY pattern. It adds extra entropy for each turn and shifted character.

zxcvbn calculates the entropy of each constituent pattern. calc_entropy() is the entry point. It's a simple dispatch:

```
calc_entropy = (match) ->
  return match.entropy if match.entropy?
  entropy_func = switch match.pattern
    when 'repeat'     then repeat_entropy
    when 'sequence'   then sequence_entropy
    when 'digits'     then digits_entropy
    when 'year'       then year_entropy
    when 'date'       then date_entropy
    when 'spatial'    then spatial_entropy
    when 'dictionary' then dictionary_entropy
  match.entropy = entropy_func match
```

But when the value xyz@gmail.com is supplied to zxcvbn as a password then it considers the password a strong choice.

Estimating strength of password "xyz@gmail.com ":

Approx time to crack: centuries
(in seconds): 27218757696.39
Strength score (1-5): 5
Entropy estimate (bits): 48.952

Zxcvbn tool is based on heuristic password model only and cannot predict the further occurrences of sequence that can lead to weak password choice.

IV. INTRODUCTION TO PASSWORD STRENGTH METRICS

Every single password can be broken. Given enough time, eventually, any password will be broken by means of an unlimited brute-force attack. A password which is broken in 5 minutes is weaker than one which is broken in 10 days, 10 months, 10 years or 10 centuries. However, time, as a metric for password strength, is very dependent on external variables, for instance, the time needed to crack a password using a certain attack can be drastically reduced if the computational resources are increased. The type of hashing or encryption algorithm also has a very important impact on the time needed to perform each comparison (e.g., attacking MD5 hashes is significantly faster than attacking SHA-3 ones.1)

Time is usually converted into "number of guesses required to break a password" (NoG), which is a strength unit independent of the actual contextual conditions. Then, depending on the "NoG per second" that can be performed in a given framework, it is straightforward to estimate the time that will take to break a certain password. By translating the "number of guesses" into "Number of bits" also used to measure the password strength in terms of fictional password entropy. For instance, a password that is cracked by an attack in 1024 guesses would mean that has 10 bit strength entropy.

V. MULTIMODAL STRENGTH METRIC

The main motivation behind the proposed approach for password strength estimation is that: by exploiting the capabilities of different individual techniques through their fusion, it will be possible to achieve one unique *multimodal* measure which overcomes many of their weaknesses [1].

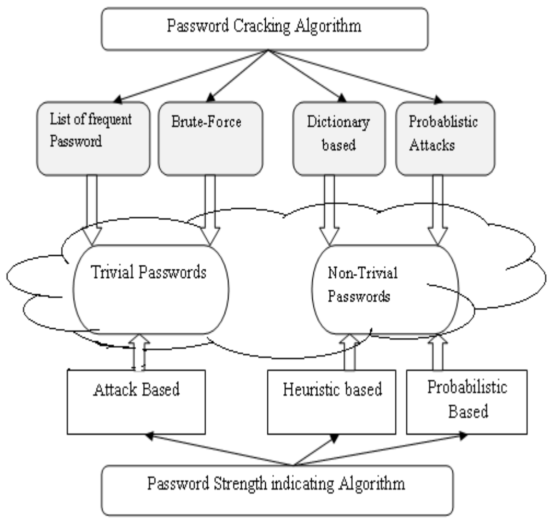


Figure 2: General diagram of password cracking algorithm and corresponding Strength estimation algorithm.

MODULE A: Trivial Password

Module A works for trivial password. Trivial passwords may be defined as those that can be broken, with all certainty, within a limited number of guesses. Therefore, these passwords should receive very low strength values. This module can contains number of sub module.

Trivial IA:

In order to detect extremely common passwords, the present strength module IA is introduced in the overall system. The module is based on blacklisting, which has now been shown in different works to be a highly valuable method against password cracking attacks.

The objective of this metric is to detect the very simple, but very effective, attack used by an adversary that seeks to gain access by just trying a list of common passwords. Such attack is especially relevant in a scenario where intruders try to get illegal access to an on-line application for which a *very limited* number of attempts (i.e., guesses) per time unit are allowed. In this very constrained situation in terms of access attempts, opponent will try to break the system by simply trying the known passwords. Accordingly, based on a password list *List pwd*, the module takes a test password *pwd* and returns a strength value S_D depending on whether the password complies with $L_D(List\ pwd\ ,\ pwd) \leq 1$. As such, only two strength values are possible: $S_D = 0$ for any of the most popular passwords or 1 character variations of them; and $S_D = 10$ for those that do not fall in the previous category.

Here is a list of the top 5 most commonly used LinkedIn passwords with their frequency:

Table1: List of top 5 common passwords of LinkedIn.

Rank	Password
1	123456
2	linkedin
3	password
4	123456789
5	12345678

The input parameter *List pwd*, a password list used in this module can be modified according to the website or application where it is used. At the time of registration the details supplied by the user like firstname, lastname, email id, date of birth can also be considered and as a blacklisted terms for that specific individual to make our metric more resistant to password cracking attacks .

Trivial IIA

This module takes as input a password and returns a strength value S_H based on its complexity [1]. It is specifically designed to detect those passwords that are brute-force able in a given situation. As such, only two strength values are possible. All passwords that are not complex enough and that, therefore, are prune to a brute-force attack, are given the similar low score $S_H = 0$. On the other hand, all passwords that are robust to brute-force are assigned the similar high score $S_H = 10$.

For example, in the case of a password with only upper-case letters $N = 26$, while in the case of a password with lower-case and upper-case letters $N = 52$. This way, the complexity parameter to be determined on a case by case basis is the minimum length *Lmin* that passwords should have in order to withstand an exhaustive search given that specific *N*. This minimum length depends on two factors:

1:Maximum number of guesses per second allowed by the application being attacked, which depends, among others, on the type of encryption strategy and oracle being used Maximum time T_{max} allowed to run a brute-force attack.

2: Maximum time T_{max} allowed to run a brute-force attack. This is a design value to be determined by the service provider depending on the security level that he wants to offer. A higher T_{max} value will imply a higher security for the selected passwords.

Given the three parameters defined above, *N*, NoG_{max} and T_{max}, selected by the system designer, it holds that: $NoG_{max} \times T_{max} = NL_{min}$. Therefore, the minimum length L_{min} of non brute-forceable passwords can be estimated as:

$$L_{min} = \frac{\log(NoG_{Max} * T_{Max})}{\log(N)}$$

This is the minimum length required so that passwords generated from an alphabet of *N* symbols are not systematically broken by a brute-force attack running for T_{max} time with a *NoGmax* computational power.

In this module we have considered as brute-forceable passwords those that may be broken in roughly 1013 guesses. For a given alphabet with N symbols, there are N^L passwords of length L. This way, the minimum password length is defined by: 1013 = N^{Lmin}. Given this equation, the next values have been defined for the minimum length of passwords in module 1B:

- Passwords using all three character types (i.e., lower case, upper case, digits and special characters). The alphabet contains N = 94 characters: L_{min} = 7.
- Passwords formed by only lower case letters, only upper case letters, or only special characters. The alphabet contains (at least) N = 26 characters: L_{min} = 9.
- For only-digit passwords, the alphabet contains N = 10 characters: L_{min} = 12.
- For any pair-wise combination of the previous character classes the alphabet contains (at least) N = 36 characters: L_{min} = 8. This means that passwords longer than L_{min} for each of the possible alphabets are considered to be resistant to brute force guessing attacks.

MODULE B: Non-Trivial Password

Other increasingly complex guessing algorithms are applied for passwords that are robust to the *Brute-force attack* & attacks**.** These passwords are non-trivial in the sense that there is only a certain non-zero probability that they will be retrieved. This strength module is designed to handle with passwords that have been assigned the highest score value by the previous modules IA and IIA, i.e., non-trivial passwords that are strong to attacks based on lists of common passwords and to brute force attacks. As passwords are heavily influenced by language, we know that the transition probability to move from current state 'c' to next state 'a' is much higher if the sequence of precedent states was (M, o, n, i, c,), than if the sequence of precedent states was (A, d, v, a, n, c), where the highest transition probability would be from the state 'c' to the state 'e'. This argumentation shows the need to consider statistical models with memory for the representation of human chosen passwords, such as *AM*-th order Markov Chain, a.k.a.*AM*-gram model The Markov Chain with adaptive memory. *AM* Markov Chain based on hierarchical Chains methods are considered for understanding human password behavior. In their approach, both the previous sequence of characters and the ending character are taken into consideration. The memory is not bound by an upper limit but by the frequency of appearance of the considered subsequence (i.e., *AM*-gram in our model).

The model is formally defined by *S* states, where *S = (N* + 3*)AMmax* and by the conditional transition probabilities *ptαiαj*, where *(i, j)* ∈ {0, . . . , *S* − 1}2. Each state is a tuple of *AMmax* characters *c* taken from the set of *N* + 3 characters, that are the *N* characters from the alphabet plus three artificial characters: an *initialization character c*0, an *undefined character cu*, and an *end of password character ce*.

A state is denoted *αi = (c*1 *i, c*2 *i. . . ,cAMmaxi),* for *i* ∈ {0, . . . , *S*− 1}, and by convention *α*0 *= (c*0, . . . , *c*0*).*

The transition probabilities from one state to another have the two following properties:

1) The sum of the transition probabilities from a state *αi* , for any *i* ∈ {0, . . . , *S* − 1} to any other state in the chain (including himself) must be equal to 1, that is

$$\sum_{j=0}^{S-1} pt_{\alpha_i\alpha_j} = 1;$$

2) The transition probability from one state $\alpha_i = (c^1_i, c^2_i, \dots, c^{AM}_i)$ to another state $\alpha_j = (c^1_j, c^2_j, \dots, c^{AM}_j)$can be greater than 0 only if $(c^2_i, \dots, c^{AM}_i) = (c^1_j, c^2_j, \dots, c^{AM-1}_j)$This represents the transition probability from a sequence of *AM* characters to a single character.

To evaluate the score S_{AM} of a password in the adaptive memory Markov chain model, all the transition probabilities of the sequence of states composing the password are evaluated, starting from the initial state α_0 to the last one containing the end of password character c_e. When a transition from one state to another is equal to zero, the first character of the considered state is replaced by the undefined character *cu*, virtually reducing the size of the memory. The score of the password is multiplied by a correction coefficient *CrCoef*, with *CrCoef* < 1, corresponding to the bonus attributed to the password strength value previously mentioned. *CrCoef* is defined as the ratio between: the minimum non-zero probability of moving from the given sequence of characters of size *AM* to any individual character; and the maximum nonzero probability of moving from the sequence of characters of size $AM - 1$ (after the memory reduction) to any individual character. This process is repeated until a transition is found or eventually the length of the sequence is reduced to a single character, which is equivalent to the Simple Markov Chain.

The final score S_{AM} attributed to a password is equal to

$$S_{AM} = -10 \log_{10} (po_{pwd})$$

MODULE C:

Module C is used to check probabilistic attacks. Probabilistic approaches apply statistical models, in most cases based on some variation of Markov Models that try to capture the way in which humans produce passwords. Such Models are sampled to generate human-like guesses.

Context-free grammars have long been used in the study of natural languages [12], where they are used to generate strings with particular structures. We show in the following that the similar approach is useful in the automatic generation of password guesses which are kind of human-created passwords.

Hierarchical Markov Chain is a tree-like model that first estimates the probability of occurrence of the higher-order structures within the password and then, in a subsequent step, estimates the probability of occurrence of characters within each structure.[1][2]

In particular, the higher-order elements considered in the model are the next *SS* subsequence divided in three classes:

• Letter-class: formed by letter subsequence of sizes 1 to *lcmax.*
• Digit-class: formed by digit subsequence of sizes 1 to *dcmax.*
• Special characters-class: formed by special character subsequence of size 1 to *scmax.*

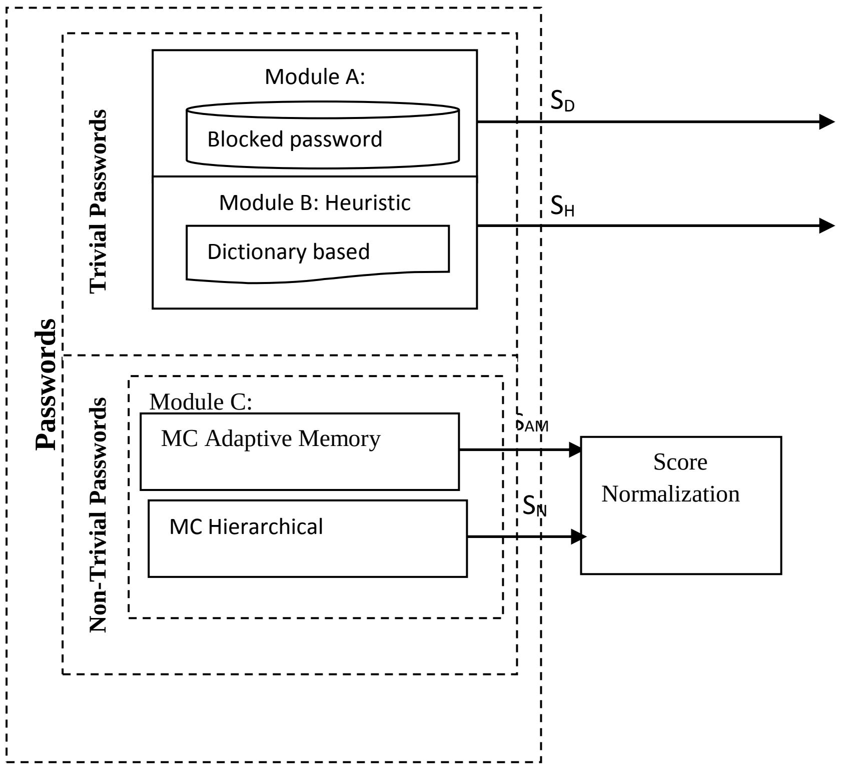


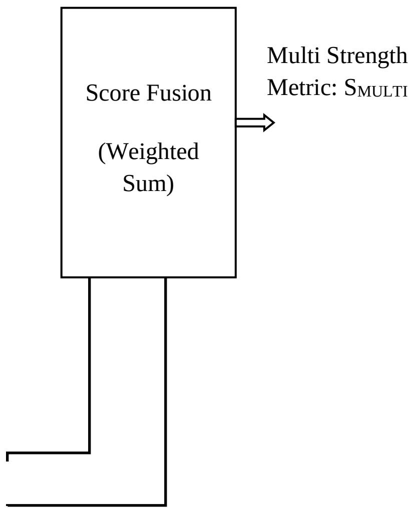
Figure 3: General diagram of multimodal strength estimation

In this model a generic password is formed by a sequence *P* higher-order elements taken from the *SS* subsequence belonging to the three classes defined above, that is $pwd = (ss^1, ss^2, \ldots ss^p, \ldots ss^P)$. In turn, each of those higher order elements is formed by *Rp* characters, that is $ss^p = (c^{p,1}, c^{p,2}, \ldots c^{p,r}, \ldots c^{p,Rp})$, where $c^{p,r}$ represents the *r* –th character of the *p*-th subsequence.

The probability of occurrence of higher-order structures is defined as $po_{ss}{}^p$, while the probability of occurrence of characters within each higher-order structure is defined a $po_c{}^{p,r}$. This way, the probability of occurrence of a generic password $pwd = (ss^1, ss^2, \ldots ss^p, \ldots ss^P)$can be computed as:

$$po_{pwd} = \prod_{p=1}^{P} [po_{ss}{}^{p} (\prod_{r=1}^{R} Po_c{}^{p,r})]$$

The final strength metric *SN* assigned to the password is finally computed as
$S_N(pwd) = -10 \log_{10}(po_{pwd})$
As a simple example, according to the Hierarchical Markov Chain, password “Pract43!” is formed by *SS* = 3 higher order elements: *ss*1 = Pract, *ss*2 = 43 and *ss*3 =!
The occurrence probability of this password would be:
• The probability that a password formed by 5 higher-order elements starts with a subsequence of 5 letters, multiplied by the probability that the first letter of a 5-letter subsequence is “P”, multiplied by the probability that in a 5-letter subsequence “P” is followed by “r”, multiplied by the probability that in a 5-letter subsequence “r” is followed by “a”, multiplied by the probability that in a 5-letter subsequence “a” is followed by “c” , multiplied by the probability that in a 5-letter subsequence “c” is followed by “t”, multiplied by the probability that
• in a password with 3 higher-order elements a 5-letter subsequence is followed by a 2-digit subsequence in the



second position, multiplied by the probability that the first digit of a 2-digit subsequence is “4”, multiplied by the probability that in a 2-digit subsequence “4” is followed by “3”, multiplied by the probability that

• In a password with 3 higher-order elements a 2-digit subsequence is followed by a 1-special character subsequence in the third position, multiplied by the probability that the first special character of a 1-special character subsequence is “!”.

Hierarchical Markov Chain is an evolution of the probabilistic PCFG model. The Hierarchical Markov Chain is more flexible and not so training-data driven, as it does not represent specific subsequence-combinations, but transition probabilities between subsequence.

To achieve the goal of multimodality and adaptability, two new probabilistic methods based on the popular Markov Chains have been proposed to accurately estimate the strength of non-trivial passwords: 1) the Markov Chain with adaptive memory and 2) the Hierarchical Markov Chain. The two models are complementary as they focus on capturing different aspects of human selected passwords: the Adaptive Memory Markov Chain can be considered as a *local model* that searchers for specific word-related patterns, while the Hierarchical Markov Chain may be understood as a *global model* that accurately represents the general structure of passwords.

Finally smoothing technique is applied for all transition metrics involved in the model.The final objective of these two sub-modules is to combine the scores (S_D, S_H, S_{AM} and S_N) provided by the four individual strength estimation algorithms presented above, into the final unique multimodal score $S_{MULT\ I.}$ One of the challenges of fusion systems is that the scores from the individual sources can be very heterogeneous [1][2].

For instance, they may not necessarily be on the same numerical scale or may follow different statistical distributions. Therefore, prior to combining them into a single multimodal score, they need to be transformed into one common domain. This is accomplished through a process known as *score normalization*, which plays a critical part in the design of a combination scheme for score level fusion.

VI. CONCLUSIONS

Traditional password strength metrics are becoming ineffective against the new generation of most highly developed password guessing attacks that are being used against existent applications. In this context, new and more reliable approaches for the estimation of password robustness are required to protect users against potential external threats as shown by the drastic change of direction with respect to previous versions of NIST's latest draft of recommendation. This method is neither perfect nor foolproof, and should only be utilized as a better way in determining methods for improving the password creation process. Different works have shown the inefficiency of current methodologies used by service providers to convince users to select stronger passwords. This includes traditional password composition policies , oriented only to avoid the selection of trivial passwords vulnerable to brute-force attacks, The main rationale behind the proposed approach is to exploit the capabilities of different individual techniques and, through their fusion, achieve one unique multimodal measure which overcomes many of the weaknesses of the single model methods.

REFERENCES

I. Javier Galbally, Iwen Coisel, and Ignacio Sanchez, "A New Multimodal Approach for Password Strength Estimation—Part I: Theory and Algorithms" Vol. 12, No. 12, December 2017

II. Javier Galbally, Iwen Coisel, and Ignacio Sanchez "A New Multimodal Approach for Password Strength Estimation—Part II: Experimental Evaluation"

III. M. Bishop and D. V. Klein, "Improving system security via proactive password checking," J. Comput. Secur., vol. 14, no. 3, pp. 233–249, 1995.

IV. S. Furnell, "An assessment of website password practices," Comput. Secur., vol. 26, nos. 7–8, pp. 445–451, 2007.

V. C. Castelluccia, M. Durmuth, and D. Perito, "Adaptive passwordstrength meters from Markov models," in Proc. Netw. Distrib. Syst. Secur. Symp., 2012, pp. 1–14.

VI. M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in Proc. IEEE Symp. Secur. Privacy (SP), May 2009, pp. 391–405.

VII. P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, and T. Vidas, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in Proc. IEEE Symp. Secur. Privacy (SSS), May 2012, pp. 523–537.

VIII. D. Wheeler. (2012). Zxcvbn: Realistic Password Strength Estimation. Available: https://blogs.dropbox.com/tech/2012/04/zxcvbnrealistic- password-strength-estimation

IX. W. E. Burr et al., "NIST special publication 800-63-2: Electronic authentication guideline," NIST, Gaithersburg, MD, USA, Tech. Rep. 800-63-2, 2012.

X. C. E. Shannon, "A mathematical theory of communication," Bell Syst. Tech. J., vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.

XI. J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in Proc. IEEE Symp. Secur. Privacy (SP), May 2014, pp. 689–704.

XII. 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, 2009

XIII. A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In Proceedings of ACM CCS, pages 364–372, 2005.

XIV. R. Veras, C. Collins, and J. Thorpe, "On the semantic patterns of passwords and their security impact," in Proc. Netw. Distrib. Syst. Secur. Symp., 2014, doi: http://dx.doi.org/ndss.2014.23103

XV. G. J. Lidstone, "Note on the general case of the Bayes–Laplace formula for inductive or a posteriori probabilities," Trans. Faculty Actuaries, vol. 8, pp. 182–192, Apr. 1920.