This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2737971, IEEE Transactions on Information Forensics and Security

DPPG: A Dynamic Password Policy Generation System

Shukun Yang, *Member, IEEE,* Shouling Ji, *Member, IEEE,* and Raheem Beyah, *Senior Member, IEEE*

***Abstract*—To keep password users from creating simple and common passwords, major websites and applications provide a password-strength measure, namely a password checker. While critical requirements for a password checker to be stringent have prevailed in the study of password security, we show that regardless of the stringency, such static checkers can leak information and actually help the adversary enhance the performance of their attacks. To address this weakness, we propose and devise the *Dynamic Password Policy Generator*, namely *DPPG*, to be an effective and usable alternative to the existing password strength checker. *DPPG* aims to enforce an evenly-distributed password space and generate dynamic policies for users to create passwords that are diverse and that contribute to the overall security of the password database. Since *DPPG* is modular and can function with different underlying metrics for policy generation, we further introduce a diversity-based password security metric that evaluates the security of a password database in terms of password space and distribution. The metric is useful as a countermeasure to well-crafted offline cracking algorithms and theoretically illustrates why *DPPG* works well.**

I. Introduction

TEXT-BASED passwords have been used widely in both online and offline applications for decades. Since passwords are personal and portable, they are not likely to be replaced in the foreseeable future [1]. However, the phenomenon that people choose simple passwords and reuse common passwords [2] has raised great security concerns as such passwords are vulnerable to offline cracking attacks. To make things worse, a number of password leak incidents [3]–[6] have happened recently and frequently. Large datasets of leaked passwords can greatly enhance attackers' capability in conducting training-based password attacks, thus posing significant threats on password security.

The most direct and pervasive protective mechanism used by major websites and applications is the password strength checker [7], which evaluates the strength of passwords proactively during user registration. While the goal is to guide users to create strong passwords, in previous work [8]–[10], the lack of accuracy and consistency in the strength feedback has been widely observed and examined. That is, existing checkers do not demonstrate effective or uniform characterization of strong passwords. Furthermore, the space for the rules and policies of the checkers to be stringent is very limited as researchers have shown that the complexity of a password is a trade-off with the usability [11]. Therefore, password strength checkers simply cannot demand users to create passwords that are too complex.

On the other hand, the password strength checker itself can be a vulnerability, which has not been studied in previous research. By defining a set of password creation policies and showing users password strength scores, password checkers can exert a strong bias on password characteristics, especially when the policies and scoring mechanisms remain static. The passwords registered to a database are largely similar to the specific password patterns enforced by the associated checker. Although password checkers vary among websites, they inevitably rely on similar rules that focus on specific password properties (e.g., length, number of digits and special characters). When rules are relatively relaxed, password users may create simple passwords following a common distribution. When rules are relatively demanding, the password distribution is closely correlated to the scoring metrics and can be inferred. Since the password checkers are publicly available, attackers can easily make use of the password checkers to learn the password characteristics distribution that is shaped by the password checkers.

Our main contributions in this paper are summarized as follows.

- In Section II, we evaluate the impact of misusing current commercial password strength checkers from the attacker's perspective and explore the possibility and potential to leverage the checkers in offline cracking attacks. Using an attack-based model, we show that the password checkers are effective for attackers to facilitate password cracking. With a certain amount of computational power, the attacker can compromise more passwords with a specific rating with the help of the strength checkers. This implies that the static policies and scoring mechanisms used by password strength checkers exert bias on the password characteristics distribution. Passwords with the same rating follow an obvious pattern which can be exploited by the attacker to refine the training data.
- In Section III, to propose a countermeasure to protect the information on password distribution and to reduce the efficacy of well-crafted training-based attacks, we devise the *Dynamic Password Policy Generator*, namely *DPPG*, which generates dynamic password policies for users. Each new user obtains a different password policy to follow, which is generated in real-time from the server based on but not reflecting the current password distribution. The policies thus are not static and a user does not

S. Yang and R. Beyah are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: syang87@gatech.edu, rbeyah@ece.gatech.edu

S. Ji is with the College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang 310027, and with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332. E-mail: sji@gatech.edu

TABLE I
DATASETS.

Name	Size	Language	Site	Type
Renren	4.7M	Chinese	renren.com/	social networks
LinkedIn	5.4M	English	linkedin.com/	professional networks
Tianya	31M	Chinese	tianya.cn/	Internet forum
Rockyou	32.6M	English	rockyou.com/	game
Gamigo	6.3M	German	en.gamigo.com/	game

TABLE II
PERCENTAGE OF "STRONG" PASSWORDS.

checker	Gamigo	Renren	LinkedIn	Rockyou	Tianya
Bloomberg-Train	0.05%	6.30%	0.31%	0.72%	0.44%
Bloomberg-Test	0.05%	6.27%	0.31%	0.72%	0.44%
QQ-Train	12.44%	22.20%	1.75%	2.56%	5.20%
QQ-Test	12.44%	22.12%	1.74%	2.56%	5.20%

know what policies others receive. *DPPG* works to even out the password distribution in the database and expand the password space. Since the policies users receive are dynamic, and unpredictable, an adversary cannot use them to infer the characteristics of the password database or select password training data.

- To further understand the password distribution and evaluate the threats posed by the exposure of the password distribution of a leaked dataset, we introduce the concept of *password diversity* and propose a diversity-based metric to measure the security of a password dataset in Section IV. The metric considers an aggregation of password properties to analyze the password characteristics distribution within a dataset. It assigns a higher security score to a password dataset with a more uniform password distribution. The metric serves as the underlying mechanism used in *DPPG* to generate dynamic policies and aims to minimize biased password distributions and to expand the usable password space. Since the training-based attacks become powerful due to strong similarities between the training and target passwords, it is meaningful to study such a metric.
- We summarize the related research work in Section V and conclude the paper in Section VI.

II. COMMERCIAL PASSWORD CHECKERS

Traditional password policies have become less popular as the more user-friendly password strength checkers become widely adopted by major websites and software. The main reason is that good password policies can easily be too stringent to use, while password strength checkers push users to create "strong" passwords subtly. However, most of the existing research only evaluates the effectiveness and helpfulness of the password strength checkers. The fact that the checkers are based on unchanged policies which indirectly bias the password characteristics distribution has not been studied. Furthermore, due to the exposure of the policies and scoring mechanisms [9], [10], [12], careful attackers can utilize the password checkers to mount more powerful attacks on passwords with high strength ratings.

A. Datasets, Checkers, and Crackers

Table I lists the 5 datasets that add up to around 81 million passwords. The datasets are leaked from several incidents [13], [14] where attackers acquire passwords by online attacking techniques. Although the password data were leaked illegally, it has been once made publicly available and used widely in password research for benevolent purposes. In our study, we use the passwords for research only without attempting to verify them.

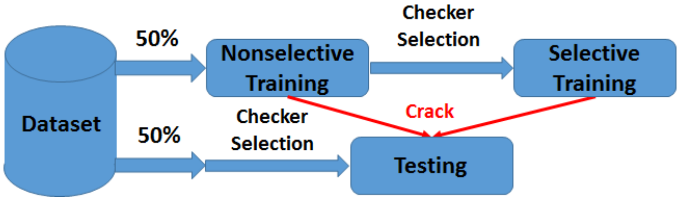


Fig. 1. Attack-based Evaluation Model

To obtain a collection of usable password strength checkers and cracking algorithms, we conduct our experiments with PARS [10]. Due to the space limitation, we only present two checkers listed in Table II. Other checkers showing consistent results are available on [15]. Bloomberg is a popular English business and news forum and QQ is a well-known Chinese portal providing numerous web services. According to evaluations in [10], [12], they provide relatively accurate and consistent feedback to users. There are 4 levels of password strength in both password checkers to make them comparable, and the highest rating is "strong" in common.

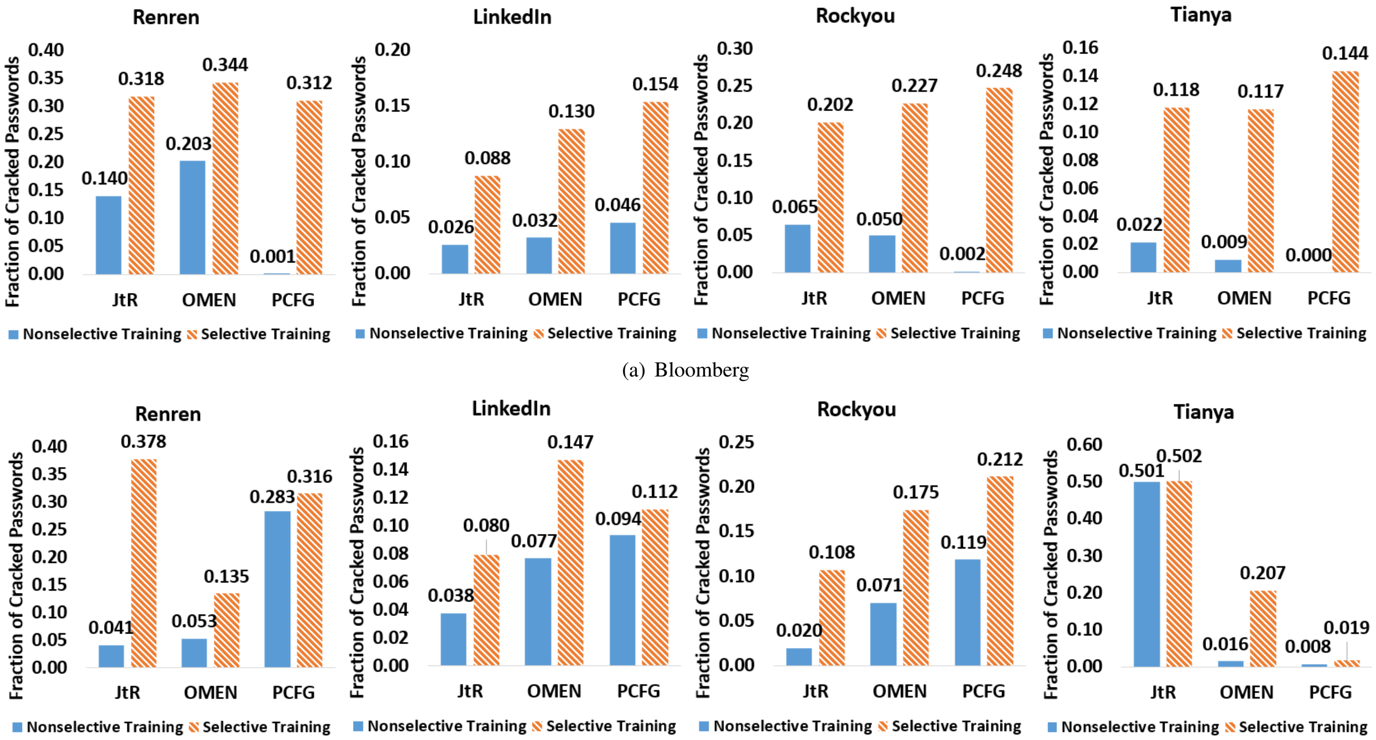
We use three state-of-the-art password cracking algorithms, JtR (John the Ripper-Markov) [16], OMEN (Ordered Markov ENumerator) [17], and PCFG (Probabilistic Context-free Grammar) [18], which have relatively optimal performance in password cracking as shown consistently in previous research literature.

B. Threat Model: Take Your Checker, Crack Your Passwords

From an attacker's perspective, we evaluate quantitatively how existing commercial password checkers can be used to enhance offline password attacks. We are particularly interested in the pool of "strong" passwords because intuitively users trust the strength feedback and create passwords that have better ratings.

In our threat model, we assume an attacker aims to crack a target set of password hashes leaked from a website which uses a password strength checker. This means that the hashed passwords can have different strength ratings[1]. We also assume the attacker has access to the checker and obtained another dataset of plain text passwords leaked from other websites as prior knowledge, which is used to train the password crackers. Since the attacker does not know the correlation between the plain text and the hashed passwords, a straightforward

[1] In general, strength checkers can accept passwords of any ratings from "weak" to "strong", but there are only several ratings available.



(a) Bloomberg

(b) QQ

Fig. 2. Intra-site Password Cracking (Bloomberg and QQ Password Checkers).

TABLE III
CROSS-SITE PASSWORD CRACKING (BLOOMBERG PASSWORD CHECKER).

Training	Renren						LinkedIn						Rockyou					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Renren	-	-	-	-	-	-	3.57%	7.17%	2.59%	7.17%	2.43%	10.97%	1.75%	15.32%	1.13%	19.22%	0.29%	11.34%
LinkedIn	0.23%	1.94%	0.05%	1.14%	0.01%	10.49%	-	-	-	-	-	-	0.66%	5.87%	0.41%	7.98%	0.03%	14.29%
Rockyou	1.09%	6.90%	0.26%	4.30%	0.08%	18.59%	10.00%	17.37%	6.22%	15.54%	6.91%	21.57%	-	-	-	-	-	-
Tianya	1.43%	4.81%	0.73%	4.78%	0.01%	9.77%	2.83%	5.46%	2.82%	6.70%	1.87%	11.89%	1.14%	5.41%	1.00%	6.93%	0.16%	11.28%
Gamigo	0.67%	4.37%	0.36%	3.46%	0.00%	20.41%	4.74%	12.76%	4.80%	15.13%	6.62%	24.30%	2.13%	11.48%	1.15%	15.37%	0.24%	25.15%

Training	Tianya						Gamigo					
Algorithms	JtR		OMEN		PCFG		JtR		OMEN		PCFG	
	NS	S	NS	S	NS	S	NS	S	NS	S	NS	S
Renren	1.69%	16.21%	0.80%	16.31%	0.07%	9.24%	0.15%	6.00%	0.01%	1.58%	0.12%	7.74%
LinkedIn	0.17%	3.24%	0.05%	0.85%	0.01%	9.04%	0.03%	6.48%	0.01%	1.17%	0.12%	11.47%
Rockyou	0.86%	8.84%	0.12%	1.85%	0.06%	10.79%	0.57%	15.53%	0.01%	2.70%	0.32%	19.96%
Tianya	-	-	-	-	-	-	0.07%	4.53%	0.02%	1.36%	0.07%	5.75%
Gamigo	0.55%	5.65%	0.06%	1.94%	0.00%	16.28%	0.18%	13.00%	0.00%	3.77%	0.06%	22.24%

strategy is to assume a common distribution in both datasets and use all the plain text passwords to train the cracking model. However, the target passwords might have been created mostly by users who trust the strength feedback from the checker and create passwords only if they are labelled as “strong”. Then, the target passwords are reasonably dissimilar from the training passwords which come from other sources. Therefore, to compromise such biased target passwords, the attacker will likely have better cracking results if the training passwords are also “strong”.

In our experiment, the objective is to see if more “strong” passwords in the target dataset can be compromised when the attacker uses the password strength checker to select training data. Figure 1 summarizes the evaluation process. First, we randomly select 50% of the passwords from a dataset in Table I to be the *Nonselective Training* dataset. Then, we apply a password strength checker in Table II to score each password in the *Nonselective Training* dataset, and select only those passwords labelled as “strong” to make up the the *Selective Training* dataset. From the other 50% of the passwords, we apply the same checker selection method to build the *Testing* dataset. Finally, we use *Nonselective Training* and *Selective Training* datasets separately, as input to JtR, OMEN, and PCFG, to crack the *Testing* dataset.

In Table II, we show the percentages of selected passwords from the datasets, e.g., *Bloomberg-Train* and *Bloomberg-Test* indicate the percentages of "strong" passwords marked by Bloomberg's checker in the datasets from which we sample training and testing data, respectively. Since we randomly divide an original dataset into halves, the distributions of "strong" passwords in both halves are approximately the same.

To conduct a comprehensive and comparable evaluation, we perform passwords cracking in both *Intra-site* and *Cross-site* scenarios. In *Intra-site* cracking, the training data and target data come from the same original dataset and in *Cross-site* cracking, the training data is from a different dataset. To make the comparison fair, we limit each cracking session to 10 billion passwords guesses uniformly. Due to the space limitation, we present *intra-site* cracking results of Renren, LinkedIn, Rockyou, and Tianya in Figure 2, and *cross-site* cracking results with Bloomberg's password checker in Table III. Other results in the appendix are consistent as well.

In Figure 2, the *intra-site* results show that *Selective Training* enable all the cracking algorithms to compromise much more "strong" passwords than *Nonselective Training*. Figure 2 (a) shows the cracking scenario where the passwords are selected by Bloomberg's checker. The performance gain of using *Selective Training* is significant. Specifically, regarding PCFG, with *Nonselective Training*, it can only crack 0.07%, 4.58%, 0.22%, and 0.01% of the passwords in the target data from Renren, LinkedIn, Rockyou, and Tianya, respectively, whereas with *Selective Training*, it can crack 31.15%, 15.40%, 24.78%, and 14.37%, respectively.

Figure 2 (b) shows the cracking scenario where QQ's checker is used. Although *Selective Training* can boost the cracking capability uniformly, the performance gain is smaller compared to that with Bloomberg's checker. For Tianya, we see that the cracking results of *Nonselective Training* and *Selective Training* are almost the same when JtR and PCFG are in use. The likely reason for this phenomenon is that QQ's checker is not as stringent as Bloomberg's, thus having less bias on the selected "strong" passwords. Another interesting observation is that PCFG, in Figure 2 (a) and (b) has very different performance. It shows much more performance gain when Bloomberg's checker is used. Due to PCFG's nature, this confirms that Bloomberg's checker is more stringent on password structure than QQ's checker.

In Table III, we show the results of *cross-site* cracking with Bloomberg's password checker. Surprisingly, we see that the cracking performance with *Selective Training* is uniformly and significantly better without exception. Gamigo, as a typical dataset with German linguistic patterns, is also subjective to a greater cracking enhancement when the adversary uses the checkers to select training data from a Chinese or English dataset e.g., a performance gain of up to 24% is observed when training from Rockyou and cracking with PCFG. This means that regardless of where attackers obtain passwords for training, they can always improve their cracking capability drastically by using the password checker associated with the target data to make a good selection of training data [2].

Our attack-based evaluation is meaningful in the following ways. We do not make assumptions on what datasets the attacker possesses. We show that as long as the corresponding password checker of the target dataset exists, the attacker can successfully crack more passwords in the target dataset that are labelled as "strong". In our experiment, *Nonselective Training* represents the original dataset that the attacker has, without applying any selection. This makes sense as the attacker will not have prior knowledge of how to select the training data simply because the target dataset is hashed. When the password checker is available, it provides information for the attacker about the target dataset, thus enabling them to select training data accordingly. Therefore, it is meaningful to compare the cracking performance with and without the password checker.

The testing dataset represents the target dataset that attackers aim to compromise, which in our case is limited to only passwords rated "strong" by the password checkers. This can be applied to passwords of any ratings, e.g., "moderate", "weak". Although we do not have Bloomberg or QQ's password datasets, by using their checkers to sample data from the available datasets, we can regard the selected data as their fair representatives.

Remarks. In this section, we conduct a comprehensive evaluation to study the feasibility and effectiveness for attackers to use existing commercial password strength checkers to launch more powerful attacks. The results are surprising that commercial password checkers can actually significantly help attackers compromise more "strong" passwords rated by the password strength checkers. This means that if the users trust a password strength checker and always creates "strong" passwords, their accounts are not necessarily more secure and are vulnerable to training-based attacks when the adversary obtains the checker. Since the training-based crackers only become more powerful when the training data is more similar to the target data, in our evaluation model, the *Selective Training* data is more similar to the *Testing* data, which further implies that the checkers exert bias on the selected passwords. Due to the nature that password policies and scoring mechanisms are static, the password distribution is consistently biased. Such bias, while not necessarily enforcing good password strength, poses significant threats on the overall password dataset security. Therefore, it is meaningful to address this limitation of the password strength checkers, and investigate how to enhance overall password data security.

III. Dynamic Password Policy Generator

One could argue that a potential solution to the password checker limitations is to have better web technologies to hide the policies and detect malignant password strength querying. However, it can result in delay in strength feedback and high false-positive rates in detection. Further, it does not resolve the fundamental bias in password distribution. Therefore, we take another approach to the problem and explore the feasibility of providing dynamic password policies to users. Considering usability, rather than forcing all users to create extremely complex passwords, we focus on the overall strength of the

[2] Of course, as previous work has shown, choosing a training dataset that has similar characteristics as the target dataset is also important to optimize cracking (e.g., choose a Chinese dataset for training if the target dataset is likely Chinese).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2737971, IEEE Transactions on Information Forensics and Security

TABLE IV
PASSWORD POLICY REQUIREMENT TYPES.

Type	Description
Length	use a range of password length
Composition	use a number of different character types
Alternation	use a number of character type transitions
Good Chars	include specific characters
Bad Chars	exclude specific characters
Structure	use a specific structure

password dataset and ensure that the passwords created by the users have diversity (i.e., cover the vast majority of the entire password space uniformly). In this section, we propose the Dynamic Password Policy Generator, namely *DPPG*, as an alternative to traditional password strength checkers.

A. Overview

DPPG is a diversity-based and database-aware application that generates password creation policies dynamically for the users. Instead of purely focusing on the complexity of candidate passwords, *DPPG* enforces a baseline complexity on the passwords (e.g., more than 6 characters long) to protect them from simple attacks, e.g., dictionary, brute-forcing. However, more focus is put on protecting the password distribution within a database by preventing aggregation of similar passwords that form a characteristically biased distribution. As long as a candidate password meets the policy, it is accepted and no additional strength feedback is provided. The policies are generated to search for candidate passwords that balance the password characteristics distribution. The underlying diversity-based metric implemented in *DPPG* is further elaborated in Section IV.

In Figure 3, we show how *DPPG* works. Initially, system administrators can place complex or random passwords as seeds in the password database. The seeds can form a white list to inject certain desired password characteristics, e.g., structures, *n*-grams. Based on the seeds, *DPPG* can start to generate password policies to users. Since dynamically generating policies requires necessary computational time depending on the number of existing passwords, to avoid delay in responding to users' requests, a policy queue is used to store policies as a buffer each time when a batch of policies are created. When the size of the policy queue reduces below a threshold, e.g., 25%, *DPPG* is signalled to generate new policies.

B. Two Modes: Explore and Exploit

In order to intelligently generate password policies based on the current password distribution, *DPPG* maintains a global characteristics frequency map and a history of generated password policies[3] that can approximate the current password distribution. There are two modes for *DPPG* to expand the usable password space and balance the password distribution.

The `exploration` mode mainly aims to expand the password space by actively introducing new characteristics

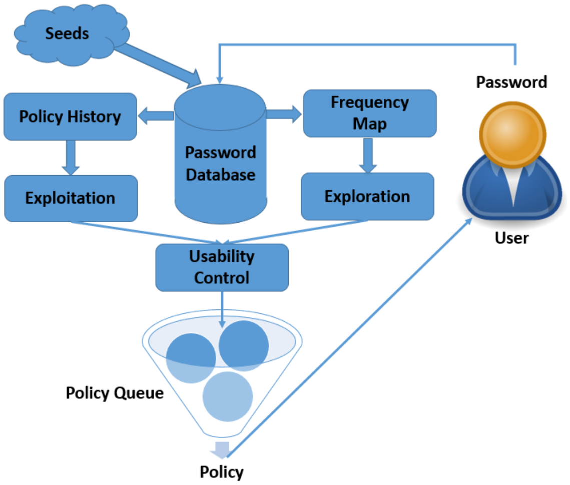


Fig. 3. Dynamic Password Policy Generator

based on the global characteristics frequency map. Before an incoming password is hashed, *DPPG* extracts its characteristics and stores the metadata in the frequency map, which keeps tracks of the overall distribution of password attributes e.g., frequency of *structures*, *characters*, and denotes the current password space. In `exploration` mode, *DPPG* creates policies that require users to be more "creative" in making a password e,g., using the character "(" which is not usual even in special characters. In this way, the passwords can cover a larger textual search space than the regular human linguistic patterns. Initially when there are not many passwords, a random mechanism is adjusted to launch the `exploration` mode more often to aggressively enlarge the password space. When the password characteristics distribution is relatively uniform as observed from the `exploitation` mode, the `exploration` mode is also evoked to introduce new characteristics.

Since purely expanding the password space is equivalent to making random passwords, *DPPG* also relies on another major component. The `exploitation` mode aims to enhance password diversity and balance the current password distribution, with the help of the password policy history. Since passwords are hashed in the database, *DPPG* stores previously generated password policies to approximate current password distributions and analyze the password diversity through the metric and algorithm discussed in Section IV. *DPPG* then identifies password characteristics that exist in the database with low appearance frequencies, and generate policies that require such characteristics. Therefore, *DPPG* creates policies that are usable and balance the password distribution by temporarily increasing the frequencies of less common password attributes.

Based on the two modes, *DPPG* determines the critical characteristics requirements, but only renders the final policies after passing them to the usability control module. In our implementation, there are 6 types of requirements that can form a policy as shown in Table IV, and the usability control module is evoked to ensure that the final policies contain only a reasonable number of requirements and are in different formats

[3] No plain text passwords are stored.

as shown in examples below.

Include the character(s): ‘v’, ‘Z’
Avoid the character(s): a, s, e
Use the structure: LLLLLUUS

Number of characters: 8 to 12 (inclusively)
Number of character types: 4
Number of alternations: 3 to 4 (inclusively)
Include the character(s): ‘?’, ‘U’, ‘)’

C. Usability Analysis

The usability of *DPPG* can translate into the ability for users to follow the policies and maintain the passwords they create. In this section, we test *DPPG* on real users and collect passwords for further analysis. We aim to show the comparison of usability between commercial password checkers and *DPPG*.

Recruitment. After our protocol was approved by the Institutional Review Board (IRB), we conducted a usability test of *DPPG* on Amazon Mechanical Turk [19]. We restricted the participants to a qualification type that requires at least 95% of approval rate, and 500 approved tasks. We excluded minors and only included English speakers.

Protocol. Our approach is to test if users who create their passwords by *DPPG* policies can successfully remember them for a reasonable time period. We also require the same participants to create passwords using QQ’s password strength checker as the control group. The participants are not informed of the purpose of our study or anything introduced in this paper. Each participant who accepts our human intelligence task (HIT) on Amazon Mechanical Turk is asked to access our web server with registration and login services. Participants are asked to complete 4 sessions of experiments which are separated by time intervals of 24 hours, 48 hours, and 72 hours to finish the entire study.

In the first session, the participant is directed to visit two artificial websites to register two accounts with usernames and passwords, following a *DPPG* policy and using QQ’s checker, respectively. Then the participant simply concludes the session by logging into the accounts with the credentials they just created. For the rest of the sessions, participants simply return to our web interface during the time specified at the end of each previous session and logged into the accounts with their credentials. All participants are informed in the beginning of the study that forgetting their passwords during the study was fine and would not penalize them. If they did forget their passwords, they were prompted to make new ones.

Due to our task requirements, participants are involved for 6-7 days to attend all sessions. The total in-session time is around 12 minutes. We paid $1.5 to each participant who completed all sessions in time. In order to collect more passwords for further analysis, we made the tasks on mechanical turk viewable to all qualified users who can attempt using *DPPG* before deciding to join the study. We also hosted standalone sessions purely to collect passwords from users. Although passwords are stored in plain text for future analysis, they are not visible to *DPPG* which only approximates password distribution by the history of policies. Due to the space limitation, we present the details of the user-study documents e.g., consent form, session screen shots on [15].

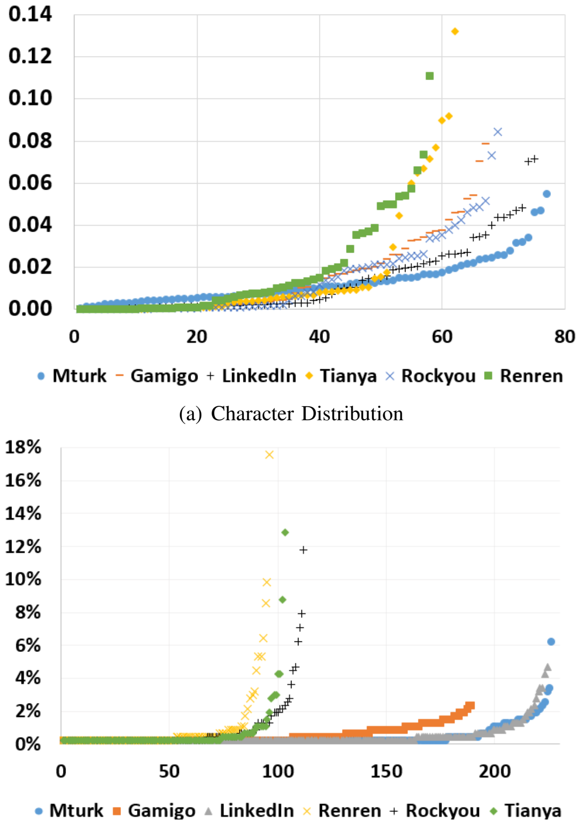
Results. After we conducted our study on Amazon Mechanical Turk for 1.5 months, there are 115 users who accepted our study and 81 of them finished 4 sessions completely. Since we do not keep track of participants’ email addresses for privacy reasons, we do not explicitly survey those who dropped out of the study. We show the results based on the records of these 81 participants in Table V, where *Times* denotes the number of sessions where participants failed to log in with the correct passwords after some trials, and *Session* denotes the indices of the sessions where users re-created passwords. The column with 0 in *times* and *session* indicates participants who logged in all sessions successfully. From the results, we see that 74 participants out of 81 consecutively succeeded in logging into our sessions with the right passwords they created according to the policies thus demonstrating the ability to remember the passwords up to a week. Of the 7 participants who forgot their passwords in at least one session, 4, 2, and 1 of them had to re-create their passwords in exactly 1, 2, and 3 sessions, respectively. Most of the participants who had to re-create passwords in one session did it in session 2 and if the participant successfully logs in in session 2, it is almost certain for them to pass the rest of the sessions. QQ’s checker as the control group shows very similar statistics but is slightly better. This demonstrates that policies generated dynamically from *DPPG* are not less usable, at least compared to existing password checkers, in terms of the ability of users to maintain the passwords.

Furthermore, the passwords from *DPPG* and QQ’s checker share 54 out of 90 passwords in common. Since password policies from *DPPG* are dynamic, unpredictable and in various templates, a more likely explanation for this phenomenon is that about half of the users reused passwords created with *DPPG* for the QQ’ checker. Although reusing passwords is a bad practice, this implies that users either reuse the passwords to get strong strength feedback in QQ’s checker, or to better maintain the passwords.

In a final survey, we further obtain subjective feedback on the usability of *DPPG* and QQ’s checker. When asked about the ease in following the policies or strength feedback, 63.75% and 73.75% of participants thought *DPPG* and QQ’s checker, respectively were above average. In addition, 65.43% and 61.73% thought *DPPG* and QQ’s checker, respectively

TABLE V
MECHANICAL TURK USER STUDY.

DPPG										
Times	0	1			2			3		
Session	0	2	3	4	2	3	4	2	3	4
#	74	3	0	1	1	1	2	1	1	1
%	91.36%	4.94%			2.47%			1.23%		
QQ Checker										
Times	0	1			2			3		
Session	0	2	3	4	2	3	4	2	3	4
#	75	2	0	1	2	1	3	0	0	0
%	92.59%	3.70%			3.70%			0.00%		



(a) Character Distribution

(b) Structure Distribution

Fig. 4. Mturk Password Analysis.

enable them to make more secure passwords. Finally, 77.78% of participants indicate the passwords they created with *DPPG* are drastically different from their other passwords and the rest indicated the passwords are somewhat similar but not the same. This shows that *DPPG* helps reduce the frequency of reusing passwords because users are unlikely to leverage their other passwords to satisfy the dynamic policies.

D. Passwords Evaluation

Using a total of 467 passwords collected from our usability study and other standalone sessions, which we denote as the Mturk dataset, we provide a statistical analysis of the password characteristics. In order to have a comparison with passwords created without *DPPG*, we randomly sampled the same number of passwords from datasets shown in Table I to form other testing datasets.

To compare the password space in the testing datasets, we conduct a character distribution analysis shown in Figure 4 (a). We assign each unique character existing in a testing dataset with a number ordered by its appearance frequency. From the plot, we see that the Mturk dataset contains more unique characters in its passwords than any of the other testing datasets. The character distribution of the Mturk dataset is also more uniform than the other datasets, suggesting that *DPPG* can expand the password space while also enforcing a balanced password distribution.

In Figure 4 (b), we conduct a similar analysis on the password structure. Again, the Mturk dataset demonstrates the largest variety of password structures among all the testing datasets and a fairly balanced distribution. Such diversity in password structure is meaningful and implies that a structure-based cracking algorithm like PCFG will be less efficient in cracking the datasets, because all structures are almost equally likely.

To further compare the security of the Mturk dataset and other testing datasets, we employ an attack-based analysis using the datasets in Table I excluding the data in the testing datasets for training to crack the Mturk and other testing datasets with 10 billion guesses. To eliminate the bias due to a small sample size of the testing data, we re-sample testing datasets and crack them in 10 repeated sessions to obtain the average cracking rates. Table VI shows the partial cracking results of Mturk, LinkedIn, and Renren datasets, and the full results are in the appendix. As we see, compared to other testing datasets, the Mturk dataset is much less vulnerable to the attacks. This is consistent with the results in Figure 4. It also shows that passwords created with *DPPG* policies are more diverse and dissimilar to other passwords, thus being more secure from training-based cracking.

One interesting observation is that although the LinkedIn dataset has a very close structure distribution with Mturk dataset in Figure 4 (b), and also a sub-optimal character distribution in Figure 4 (a), it is still noticeably more vulnerable to cracking than the Mturk dataset. Since the performance of the training-based cracking algorithms mainly depends on the training data as shown in previous work, it means that our training datasets are much more similar to the LinkedIn target sample than to the Mturk dataset. Due to the space limitation and results consistency, we place cracking results of other testing datasets in the appendix.

TABLE VI
CRACKING EVALUATION ON MTURK PASSWORDS

JtR	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	4.93%	7.92%	7.49%	7.49%	7.28%
LinkedIn Sample	22.27%	26.34%	19.49%	23.98%	11.35%
Renren Sample	59.31%	63.60%	71.52%	67.02%	68.95%
OMEN	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	5.35%	7.49%	7.49%	7.71%	6.00%
LinkedIn Sample	11.35%	24.84%	11.13%	21.20%	3.64%
Renren Sample	41.97%	56.75%	62.96%	60.17%	53.10%
PCFG	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	2.57%	3.43%	3.00%	3.43%	4.71%
LinkedIn Sample	17.77%	19.91%	17.99%	19.06%	17.13%
Renren Sample	28.91%	30.62%	50.54%	44.11%	52.46%

E. Prevention and Detection of Misuse

While *DPPG* aims to protect users' passwords, it is important to understand how it can be misused and leveraged by the adversary. It would be no surprise if the adversary can eventually hack the server of *DPPG* and obtain crucial information on the password distribution. In this subsection, we discuss the security of *DPPG* and the impact of possible misuse.

First we assume the adversary seeks to obtain crucial information on the password distribution by monitoring the dynamic password policies. Since the policies are designed to request diverse passwords that are dissimilar to the existing passwords, they only reflect password characteristics that are rare or absent, which is not useful information for cracking. In fact, the general password space is approximately infinite, and the usable passwords created by people only take a small part of it. Therefore, the adversary cannot infer the current password distribution from the policies. The adversary may try to record the password policies they see by not submitting a password during registration so as to learn what policies other users might follow. For unfulfilled password policies, *DPPG* is designed to expire such policies within a short period of time, by artificially creating dummy passwords according to the policies. Such dummy passwords can be selected from leaked password data.

Next, we consider the scenario where the adversary can hack into the hosting server and obtain subsets of password hashes and the metadata used in *DPPG*. The most sensitive information from the generator is the global frequency map, which stores the appearance frequencies of password characteristics. While the dynamic policies presented to users reveal password characteristics not in the database, the metadata that is leaked may expose the characteristics in the database. However, the global frequency map will ideally present a nearly uniform distribution because *DPPG* tries to constantly balance the current distribution. The adversary is likely to see password structures with similar possibilities as shown in Figure 4, and has to enumerate all possible cases. Furthermore, no semantic or positional information is stored in the map. Therefore, the adversary can only leverage this information minimally and will still need to put a significant amount of effort in cracking the password hashes. This scenario is further discussed in Section IV-C when we evaluate the cracking performance on diverse passwords.

Remarks. To the best of our knowledge, *DPPG* is the first password policy generator that can generate password policies dynamically according to the current password distribution. Since no password strength feedback is returned and the policies generated by *DPPG* are dynamic and unpredictable, the attacker will find it extremely difficult, if not impossible, to learn the system or the inner password distribution. The policies themselves are in different formats and only contain information that is ideally contrary to the distribution in the database, because *DPPG* always tries to balance the current distribution and expanding password space. Through the characteristics analysis and the attack-based evaluation, we further verify that password datasets created with *DPPG* are diverse and relatively robust to training-based cracking attacks. Furthermore, the usability of *DPPG* is not sacrificed for dynamic policies according to our user study, which makes *DPPG* practical to use. Therefore, *DPPG* can be a more secure alternative to current password strength checkers in terms of protecting password distribution information and preventing crafted training-based offline attack.

IV. Password Diversity

In this section, we propose to measure the strength of a password dataset in terms of password distribution, by evaluating the password diversity in the dataset.

TABLE VII
Password Attributes.

Attribute	Type	Weight	Function
`Length`	absolute	w_1	$f^1_{p_{ij}}$
`Comp`	absolute	w_2	$f^2_{p_{ij}}$
`Alt`	absolute	w_3	$f^3_{p_{ij}}$
`CompFreq`	absolute	w_4,w_5,w_6,w_7	$f^4_{p_{ij}}$,$f^5_{p_{ij}}$,$f^6_{p_{ij}}$,$f^7_{p_{ij}}$
`LCS`	relative	w_8	$f^8_{p_{ij}}$
`LDist`	relative	w_9	$f^9_{p_{ij}}$
`Alt-Str`	absolute	w_{10}	$f^{10}_{p_{ij}}$
`LCS-Str`	relative	w_{11}	$f^{11}_{p_{ij}}$
`LDist-Str`	relative	w_{12}	$f^{12}_{p_{ij}}$

We define password diversity as within a password dataset, how dissimilar passwords are with each other regarding a specific set of characteristics. For example, "*forgetme886*" and "*iloveyou775*" are very similar even though they don't share many common characters. They are similar because they both have 11 characters; they contain only lower-case English alphabets and numerical digits; and they are composed by 8 letters followed by 3 digits. If password length, types of characters and structure are the characteristics of individual passwords used to determine similarity, we can claim these two passwords are very similar. However, it is also interesting to point out that, if we want to consider more sophisticated characteristic such as semantics, the actual meaning of words in the passwords can conversely make them less similar. Therefore, the similarity should be a conglomerate measure of all password properties of interest, rather than a measure of a single or typical attribute.

In a password dataset, the distribution of such characteristics can then be used to describe the diversity of the passwords. If the distribution is closer to a uniform distribution, the passwords are less similar to each other and the password dataset is more diverse. In this section, we will quantify password similarity and provide a systematic way to measure the dataset-wise diversity.

A. Password Similarity Measure

To quantify password similarity, we first clarify the characteristics that are considered in our measure in Table VII. The *type* of attributes is *absolute* if the attribute is independent and contribute to restraining the password space, or *relative* if it is dependent of both passwords that are in comparison and does not affect the password space. The *weight* of each attribute is its weight in the password similarity quantification. The *function* associated with each attribute, is a normalized measure of the difference between such attributes in two passwords, $\mathbf{p_i}$ and $\mathbf{p_j}$, when quantifying their similarity. The choices of attributes are elaborated as follows.

`Length` is the number of characters in a password. Almost all password policies and strength checkers enforce a minimum length limit due to brute-force attack.

$$f^1_{p_{ij}} = 1 - \left| \frac{length\, of\, p_i - length\, of\, p_j}{\max\{length\, of\, p_i, length\, of\, p_j\}} \right| \tag{1}$$

`Comp` is the number of different character types used in the password. *L*, *U*, *D*, and *S* represent lower-case characters, upper-case characters, numerical digits, and special characters, respectively. In password policies and checkers, `Comp` is also a popular measure. In previous analysis [11], it is shown that requiring more character types reduces usability of the passwords.

$$f^2_{p_{ij}} = 1 - \left| \frac{comp\, of\, p_i - comp\, of\, p_j}{4} \right| \qquad (2)$$

`Alt` is short for alternation, which means the number of character switches in a password normalized by the password length. For example, “pssS55” has 3 alternations at “p-s”, “s-S”, and “S-5” and 2 structural alternations at “s-S”, and “S-5”. It is meaningful to consider alternation in that it relates to both semantic and structural information about the password. Furthermore, alternation is another strong factor in limiting the usability of a password. *DPPG* limits the alternations in the policies it generates to make them more usable.

$$f^3_{p_{ij}} = 1 - \left| \frac{alt\, of\, p_i}{length\, of\, p_i - 1} - \frac{alt\, of\, p_j}{length\, of\, p_j - 1} \right| \qquad (3)$$

`CompFreq` is the character type appearance frequency.

$$f^{4-7}_{p_{ij}} = 1 - |CompFreq\, in\, p_i - CompFreq\, in\, p_j| \qquad (4)$$

`LCS` stands for longest common substring, which is a relative attribute. For a pair of passwords in comparison, we regard the length of the longest common substring as a shared attribute.

$$f^8_{p_{ij}} = 1 - \frac{LCS(p_i, p_j)}{\min\{length\, of\, p_i, length\, of\, p_j\}} \qquad (5)$$

`LDist` is Levenshtein Distance, which calculates the minimum number of character changes, through insertion, modification, and deletion, that are needed to transform one password to another.

$$f^9_{p_{ij}} = 1 - \frac{LevenshteinDistance(p_i, p_j)}{\max\{length\, of\, p_i, length\, of\, p_j\}} \qquad (6)$$

We use $\mathbf{S(p_i)}$ to indicate the structure of $\mathbf{p_i}$, and `Alt-Str`, `LCS-Str`, and `LDist-Str` in Table VII to account for structural information when quantifying the similarity of two passwords.

$$f^{10}_{p_{ij}} = 1 - \left| \frac{alt\, of\, S(p_i)}{length\, of\, p_i - 1} - \frac{alt\, of\, S(p_j)}{length\, of\, p_j - 1} \right| \qquad (7)$$

$$f^{11}_{p_{ij}} = 1 - \left| \frac{LCS(S(p_i), S(p_j))}{\min\{length\, of\, p_i, length\, of\, p_j\}} \right| \qquad (8)$$

$$f^{12}_{p_{ij}} = 1 - \frac{LevenshteinDistance(S(p_i), S(p_j))}{\max\{length\, of\, p_i, length\, of\, p_j\}} \qquad (9)$$

We further define the *similarity score* as

$$\mathbf{D}_{p_{ij}} = \sqrt{\sum_{k=1}^{12} (f^k_{p_{ij}})^2 \times w_k} \quad , \quad \sum_{k=1}^{12} w_k = 1.$$

To the best of our knowledge, our quantification of the password similarity is the first attempt to provide a comparable measure on how similar two passwords are with regards to various primitive attributes of the passwords. Different from [8] and [20] where only structure and *n*-gram, respectively is considered, our quantification takes into account a vector of password attributes and has the flexibility to allow weight adjustment for better performance. By assigning different weights to the password characteristics, researchers can put more focus on the evaluation of specific attributes. This is also potentially helpful when new attack models/algorithms emerge based on a composite of the password characteristics. We use $\frac{1}{12}$ for all weights by default to consider all attributes equally and discuss more on weights selection in the appendix. More sophisticated password attributes, e.g., semantics, positions of characters can be added to the quantification of password similarity thus making the approach extensible. Based on this quantification, we further propose a metric and a systematic way to measure the diversity of a password dataset.

B. Diversity-based Metric: Graph Model and Communities

To evaluate the diversity of a sizeable password dataset, we propose to group passwords into communities based on our similarity quantification. A password community contains passwords that have higher similarity with each other, than with passwords in other communities. When password datasets are large, the password diversity can then be represented by the number of communities detected, and the sizes of the communities.

Conceptually and computationally, we connect passwords in a graph model which enables us to analyze the similarities among the passwords. Each password dataset can be built into a graph, with nodes being the passwords, and edges being their relations weighted by the pairwise *similarity score* quantified in the previous subsection. For a password dataset, we can compute a *similarity score* for each pair of passwords and obtain a weighted complete graph.

The password graph preserves the similarities among the passwords and is convenient for further analysis of password diversity. Since community detection on a complete graph results in overwhelming time and space complexity, we further make the graph sparser by cutting edges that have weights less than a threshold, which is by default the mean value of all weights. Then we use a simple, light-weight, and efficient algorithm, the Louvain Method [21], to detect communities in the passwords graph. Finally, we calculate the diversity-based password dataset score $\mathbf{DivScore}$ by dividing the mean value of the sizes of detected communities by the standard deviation of the sizes. We show that $\mathbf{DivScore}$ can serve as an indicator of the overall security of a password dataset in the next subsection. The diversity-based metric also serves as a critical component in the *exploitation mode* of *DPPG*. By analyzing the password policy history and using passwords from leaked datasets, *DPPG* simulates the stored hashed passwords and evaluate the current password diversity to determine the new policy requirements balancing the password distribution.

C. Evaluation of the Diversity Measure

To evaluate the robustness of our metric, we look at both the effectiveness in its protecting passwords from cracking

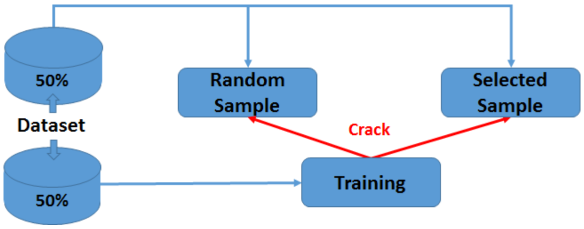


Fig. 5. Attack-based Evaluation

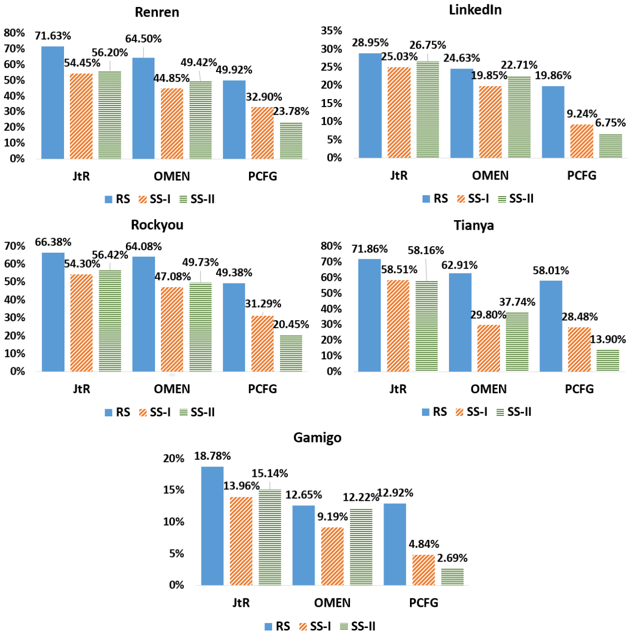


Fig. 6. Diversity-based Cracking.

TABLE VIII
DIVERSITY-BASED PASSWORD SECURITY METRIC.

Dataset	Std/Mean	DivScore	Dataset	Std/Mean	DivScore
Renren	1.63	0.61	Sample	1.97	0.51
Tianya	1.40	0.71	Sample	1.87	0.53
Rockyou	1.27	0.79	Sample	2.03	0.49
LinkedIn	1.11	0.90	Sample	2.23	0.45
Gamigo	0.44	2.27	Sample	2.28	0.44
–	-	-	Mturk	0.38	2.61

models, and the possibility of the metric leaking important password distribution information like existing commercial strength checkers discussed in Section II.

1) Attacking without Metric Details: In Figure 5, we describe the attack model to test if our proposed diversity-based measure can protect password datasets. We randomly select 50% of passwords from datasets shown in Table I and use them to train the cracking algorithms. From the other half of the passwords, we construct a password graph and run the Louvain Method to detect communities. Based on the number of communities and sizes of communities, we randomly select a fixed number of passwords from each of the communities as the selected samples. In our experiment setup, in each dataset we detect 5 communities. To make the selected sample size non-trivial, we randomly select 20000 passwords from each of the communities and thus obtaining 100000 passwords in each selected sample.

TABLE IX
CRACKING RESULTS OF THE MTURK DATASET.

Training:	Gamigo	LinkedIn	Renren	Rockyou	Tianya
JtR	6.42%	9.85%	7.71%	8.14%	6.85%
OMEN	7.49%	8.57%	7.49%	7.28%	6.00%
PCFG	1.93%	1.71%	1.71%	2.14%	1.93%

Finally, from the same password data we use to build the graph model, we randomly select 100000 passwords to form the random samples. Therefore, we obtain a selected sample that is based on our diversity-based metric, and a random sample that has the password distribution that is statistically the same with that of the original dataset. We crack these two samples separately, with 10 billion guesses.

In the left part of Table VIII, we show the diversity scores computed with the diversity metric of the 5 datasets. Ranked by the scores in ascending order, we see that Renren has the lowest diversity score while Gamigo has the highest. This suggests that Gamigo has a relatively more uniform distribution than other datasets and Renren has the most unbalanced distribution.

In Figure 6, we show the cracking results of the attack model. RS denotes the random samples and SS-I denotes the selected sample from the password communities. For all datasets, the random samples have more cracked passwords than the selected sample. The selected sample is formed with regards to the diversity metric which aims to eliminate bias on the dataset and make the password distribution more uniform. Therefore, the selected sample is less vulnerable to algorithms trained with biased password distribution.

Furthermore, we see in Figure 6 that the general cracking rates for the datasets tend to follow an order of the ranking in Table VIII. Renren, Tianya, and Rockyou have the highest cracking rates by JtR at 71.63%, 71.86%, and 66.38%, respectively, which are drastically higher than that in LinkedIn and Gamigo. This is consistent to statistics shown in Table VIII where Renren, Tianya, and Rockyou have similar diversity scores that are the lowest. Gamigo, having the highest diversity score, does have the lowest cracking rates with regards to all the cracking algorithms. Therefore, Figure 6 and Table VIII show consistent results, which means our diversity-based metric can provide effective and accurate indication on the security of passwords dataset.

2) Attacking with Metric Details: To examine if our metric has the same limitations as password strength checkers in Section II, we conduct the the same evaluation as for the password checkers illustrated in Figure 1. We use the training data in Section IV-C1 as *nonselective training* and the selected sample (SS-I in Figure 6) as the testing data. Assuming the attacker can obtain complete details of our metric including the weight values, and apply it to select training data by communities detection, we draw random samples from each of the communities to build the *Selective Training* dataset in the same way as we build the selected sample in Section IV-C1. In this set up, we ensure the cracking evaluation results are comparable and we place them in the same figure denoted as SS-II.

From Figure 6, we see that PCFG has better cracking performance consistently when trained with *nonselective training*, which implies the *selective training* does not introduce a similar structural distribution to the cracker. However, it is interesting that OMEN shows the exact opposite case with better performance when trained with *nonselective training*. JtR on the other hand, does not break the tie consistently either. The cracking rates of each scenario are generally very close with the *Nonselective* and *Selective* training, which is different from the obvious contrast observed in password strength checkers in Figure 2. The phenomenon shows that *selective training* in this case, may or may not enhance training-based cracking attack, and does not reveal useful password distribution to the adversary. This is reasonable because our diversity metric aims to balance the password distribution and eliminate possible bias. Therefore, it is not of typical interest for the adversary to leverage our diversity metric to enhance cracking.

3) Attack on Passwords from User-study: Since the password diversity metric is used as an underlying implementation of the *exploitation* mode in *DPPG*, it is interesting to test the metric and the passwords created with *DPPG* in the same experiment. In Table IX, we show the results of using the *selective training* in Section IV-C2 to crack the Mturk dataset in Section III, which is comparable to the partial results in Table VI. We observe the same inconsistencies again in the results of both tables. Further, in the right part of Table VIII, we show that the Mturk dataset has a higher *DivScore* than other sample datasets. When trained with random samples from original password datasets, PCFG can crack more passwords consistently of Mturk dataset and OMEN shows the opposite. The performance gain of OMEN in all scenarios are noticeable but insignificant. Therefore, passwords created with *DPPG* do not share common distribution with other passwords created using the similar diversity-based algorithm and thus are relatively secure even if the diversity metric is obtained by the adversary.

Remarks. In this section, to explore the security of password characteristics distribution, we define password diversity. To the best of our knowledge, this is the first attempt to quantify the diversity of passwords using various password attributes. Based on password diversity, we propose a useful password security metric to evaluate the password dataset security. Our metric is different from traditional max-likelihood or min-entropy metrics which depends on specific rules that relate to individual password strength. Instead, the metric focuses more on the security of the password distribution contributed by each individual password. Through several attack-based evaluations, we show that the diversity metric while improving the security of password dataset, does not leak crucial information that significantly helps the attacker. The diversity-based metric also serves as a key component in *DPPG* in Section III. Although *DPPG* uses the policy history to approximate password distribution, it still maintains accurate metadata with the global frequency map and cracking evaluation on the Mturk dataset is consistent with our assumptions.

V. Related Work

The trade off between the usability and stringency of password requirements has been explored extensively. In [22], Shay et al. found that users struggle with new and complex password requirements, and in [23], Mazurek et al. found that users who complain about complex password policies create vulnerable passwords. In [24], Huh et al. proposed a system-initiated password scheme and conducted a large-scale usability test. These works show that usability is an important factor in designing password policies.

Traditional password strength metrics have been found ineffective through previous work. In [8], Weir et al. evaluated NIST entropy and other traditional metrics and found them ineffective and proposed PCFG cracking-based password creation policies. Another work with a similar approach is [25], where Kelley et al. concluded that entropy is an ineffective measure of password security. Although interesting proposals were made to replace traditional metrics, they are still in terms of individual passwords without considering the overall password distribution of a password database. The similarities between passwords and their impact on password security are not studied. In [9], Carnavalet and Mannan analyzed the feedback from 11 commercial checkers on passwords in various datasets. They found significant inconsistencies among different checkers, which may confuse users. Ji et al. in [10] further conducted attack-based analysis on commercial checkers to find that many of them provide inaccurate and misleading feedback.

To suggest a different approach than commercial checkers, Castelluccia et al. presented the *Adaptive Password Strength Meter* that estimates password strength using Markov models [20]. In [26], Houshmand and Aggarwal proposed a tool, named *Analyzer and Modifier for Passwords* (AMP), to help users choose stronger passwords based on the PCFG cracking model. Komanduri et al. implemented *Telepathwords* to help users create strong passwords by making real-time predictions [27]. In [28], Forget et al. also developed a tool, namely *Persuasive Text Passwords* (PTP), which leverages the persuasive technology principle to influence users in creating more secure passwords without sacrificing usability. Schmidt and Jaeger evaluated the security of automated strengthening of passwords [29]. They found that passwords that were strengthened are still susceptible to modern cracks, provided that the adversary knows the strengthening algorithm. Camenisch et al. in [30] proposed a cryptographic protocol to protect passwords against server compromise by distributed verification. The work most related to this paper is [31], where Schechter et al. proposed to prevent users from creating popular passwords using a bloom filter. However, the filter only recognizes popular passwords rather than having the capability to identify popular password patterns.

In [2], Florêncio and Herley conducted a large scale study of web password habits. Several interesting facts are found such as on average a user has 6.5 passwords, and each of them is shared across 3.9 different sites. Similar to [2], Gaw and Felten studied the password reuse phenomenon [32]. Based on a study of 49 undergraduate students, they concluded that the majority of users have three or fewer passwords and their passwords are reused twice. Stobert and Biddle also studied user behavior in managing multiple passwords [33] to find that many users reuse and write down passwords.

VI. Conclusion

In this paper, we study the password space and distribution to understand password dataset security better. Due to the limitation of existing strength measuring mechanisms, we propose a new and usable alternative based on an effective diversity metric to better protect passwords from offline cracking attacks.

We start by identifying issues with the existing commercial password strength checkers and evaluate them from the adversarial perspective. While previous work has analyzed the consistency and accuracy of the checkers, much effort has not been spent on their limitations of biasing and leaking password distributions to the adversary. Through our evaluation, we find that password strength checkers are effective in helping attackers mount more powerful attacks. The reason is that password strength checkers rely on static scoring policies that exert bias on the password distribution. The checkers can be leveraged by the attackers easily to select training data that are similar to the target passwords.

To propose an effective alternative that addresses the limitations of password strength checkers, we implement *DPPG* to generate dynamic policies for users, which is based on a password diversity metric and the current password distribution. To the best of our knowledge, *DPPG* is the first dynamic password policy generator that provides unpredictable dynamic policies and enforces usability control. Through *exploration* and *exploitation* modes, *DPPG* can expand the password space and balance password characteristics distribution which increase the overall security of the password dataset. Through a usability study, we test *DPPG* in practice and collect passwords for further analysis. Experiments are also conducted to show that the collected passwords are more diverse in their attributes and have good security.

To study the password distribution and its security impact, we define the concept of password diversity. To the best of our knowledge, this is the first attempt to define and quantify password diversity considering a vector of password attributes. The quantification is extensible and can be adjusted with different weight values to shift the focus of measurement. To provide a way to analyze the password diversity of a dataset, we propose the diversity-based password security metric which is a key component for *DPPG* to generate effective policies. We also evaluate the metric from an adversarial perspective using it to sample data for an attack-based evaluation. Through cracking experiments in different setups, we conclude that the metric is effective in evaluating the security of password datasets and thus can serve as an effective start to evaluate password dataset security with regards to password diversity.

Acknowledgment

This work was partly supported by the Provincial Key Research and Development Program of Zhejiang, China under No. 2016C01G2010916, the Fundamental Research Funds for the Central Universities, the Alibaba-Zhejiang University Joint Research Institute for Frontier Technologies (A.Z.F.T.) under Program No. XT622017000118, and the CCF-Tencent Open Research Fund under No. AGR20160109.

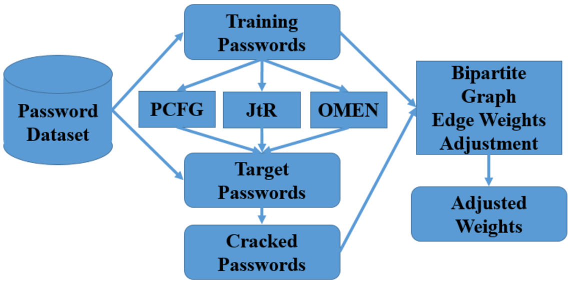


Fig. 7. Weights Model

Appendix A
Weights

The weights of different password attributes are important factors in the quantification and provide flexibility for system administrators or password researchers. The careful selection of the proper weight values can be studied using sophisticated machine learning techniques and would be itself an interesting and meaningful research topic. For the purpose of this paper, we do not delve into very complicated models to try to obtain an absolutely optimal set of weights. Instead, we propose a simple and intuitive way that solves the problem to learn a reasonable selection of the weights shown in Figure 7.

From a leaked password dataset, we randomly select a portion of the passwords as training data denoted as set $\mathbb{T}$, and use the remaining passwords as target data. Then we crack the target data with PCFG, OMEN, and JtR using the same training data and limiting the guess number of each algorithm to 10 billion. We aggregate the cracked passwords into a set denoted as $\mathbb{P}$, and then all training passwords in $\mathbb{T}$ are deemed similar to all cracked passwords in $\mathbb{P}$. Finally we form a bipartite graph between the two sets and each pair of passwords maintain an edge with a computed similarity vector.

In the final step, we traverse each edge in the bipartite graph and evaluate each similarity vector. For each similarity vector, we sort the attribute values and increase the weights of the top 4 attributes by $\frac{1}{2\times|\mathbb{P}|}$, and decrease those of the bottom 4 attributes by $\frac{1}{2\times|\mathbb{P}|}$. Therefore, we obtain an adjusted weight vector with reasonable values.

Appendix B
Mturk Dataset Cracking Evaluation

In Table XI, we show more results on cracking analysis on the Mturk dataset and the samples from datasets shown in Table I. We see that cracking performance on the Mturk dataset is much worse than the other datasets, which is consistent with results shown in Figure 4 and conclusions in Section III-D.

TABLE X
CROSS-SITE DIVERSITY-BASED CRACKING

Training	Gamigo		LinkedIn		Renren		Rockyou		Tianya	
JtR	RS	SS	RS	SS	RS	SS	RS	SS	RS	SS
Gamigo	18.78%	13.96%	20.68%	0.00%	17.13%	15.74%	20.60%	0.00%	12.98%	0.00%
LinkedIn	25.29%	21.50%	28.95%	25.03%	21.64%	22.61%	26.86%	25.74%	15.84%	19.70%
Renren	59.11%	44.48%	64.78%	48.66%	71.63%	54.45%	67.40%	52.40%	68.61%	49.14%
Rockyou	59.72%	43.49%	63.31%	47.15%	57.80%	45.76%	66.38%	54.30%	47.48%	38.23%
Tianya	56.68%	28.16%	61.58%	30.13%	68.83%	34.59%	63.92%	31.15%	71.86%	58.51%
Training	Gamigo		LinkedIn		Renren		Rockyou		Tianya	
OMEN	RS	SS	RS	SS	RS	SS	RS	SS	RS	SS
Gamigo	12.65%	9.19%	17.81%	12.79%	11.18%	11.91%	18.08%	14.16%	5.16%	9.75%
LinkedIn	12.83%	12.19%	24.63%	19.85%	12.86%	16.91%	22.61%	20.86%	5.12%	14.00%
Renren	42.37%	26.14%	55.82%	35.34%	64.50%	44.85%	61.65%	40.63%	53.75%	33.52%
Rockyou	42.42%	28.23%	55.44%	36.46%	43.66%	32.93%	64.08%	47.08%	20.11%	23.61%
Tianya	44.53%	19.55%	57.43%	25.85%	65.76%	31.17%	60.89%	27.69%	62.91%	29.80%
Training	Gamigo		LinkedIn		Renren		Rockyou		Tianya	
PCFG	RS	SS	RS	SS	RS	SS	RS	SS	RS	SS
Gamigo	12.92%	4.84%	13.42%	5.66%	13.79%	6.77%	15.05%	8.81%	13.44%	8.77%
LinkedIn	17.93%	8.50%	19.86%	9.24%	19.30%	11.07%	20.86%	13.65%	18.34%	14.37%
Renren	29.94%	20.53%	30.75%	20.95%	49.92%	32.90%	44.58%	29.59%	52.45%	34.13%
Rockyou	42.89%	23.04%	41.01%	22.01%	44.76%	25.17%	49.38%	31.29%	43.28%	28.11%
Tianya	25.01%	13.79%	28.08%	15.37%	42.80%	22.15%	41.28%	20.95%	58.01%	28.48%

TABLE XI
MTURK DATASET EVALUATION.

JtR	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	4.93%	7.92%	7.49%	7.49%	7.28%
Gamigo Sample	16.06%	17.34%	14.56%	17.34%	9.85%
LinkedIn Sample	22.27%	26.34%	19.49%	23.98%	11.35%
Renren Sample	59.31%	63.60%	71.52%	67.02%	68.95%
Rockyou Sample	57.82%	62.10%	54.39%	64.03%	46.68%
Tianya Sample	57.17%	62.53%	71.73%	65.31%	73.88%
OMEN	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	5.35%	7.49%	7.49%	7.71%	6.00%
Gamigo Sample	13.06%	17.13%	9.42%	16.92%	4.28%
LinkedIn Sample	11.35%	24.84%	11.13%	21.20%	3.64%
Renren Sample	41.97%	56.75%	62.96%	60.17%	53.10%
Rockyou Sample	42.18%	53.53%	41.97%	62.74%	20.34%
Tianya Sample	45.40%	57.39%	65.10%	62.53%	62.31%
PCFG	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Mturk	2.57%	3.43%	3.00%	3.43%	4.71%
Gamigo Sample	10.49%	10.92%	10.71%	12.85%	11.13%
LinkedIn Sample	17.77%	19.91%	17.99%	19.06%	17.13%
Renren Sample	28.91%	30.62%	50.54%	44.11%	52.46%
Rockyou Sample	40.90%	38.54%	43.25%	47.97%	41.11%
Tianya Sample	25.48%	26.98%	43.04%	42.40%	59.96%

APPENDIX C
SELECTION ATTACK

In Table X and Table XII, we show the full results of cracking analysis conducted in Section IV-C1 and Section IV-C2, respectively.

TABLE XII
SELECTION ATTACK

JtR	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Gamigo	15.14%	15.44%	15.98%	16.98%	14.41%
LinkedIn	23.53%	26.75%	25.39%	27.34%	21.71%
Renren	46.54%	49.77%	56.20%	52.90%	50.48%
Rockyou	44.17%	50.75%	50.65%	56.42%	42.83%
Tianya	30.12%	29.51%	57.50%	31.21%	58.16%
OMEN	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Gamigo	12.22%	13.14%	11.95%	13.70%	9.17%
LinkedIn	15.95%	22.71%	18.19%	20.99%	13.51%
Renren	31.76%	38.24%	49.42%	41.84%	36.08%
Rockyou	28.79%	41.67%	39.67%	49.73%	24.95%
Tianya	24.80%	26.31%	31.48%	26.94%	37.74%
PCFG	Gamigo	LinkedIn	Renren	Rockyou	Tianya
Gamigo	2.69%	3.66%	3.55%	3.99%	3.06%
LinkedIn	4.53%	6.75%	5.97%	6.88%	5.11%
Renren	14.09%	16.83%	23.78%	19.43%	18.95%
Rockyou	15.36%	17.37%	19.01%	20.45%	16.64%
Tianya	9.21%	11.27%	13.59%	12.19%	13.90%

REFERENCES

[1] C. Herley, P. C. Oorschot, and A. S. Patrick, "Passwords: If we're so smart, why are we still using them?" *FC*, 2009.

[2] D. Florêncio and C. Herley, "A large-scale study of web password habits," *WWW*, 2007.

[3] "http://www.adeptus-mechanicus.com/codex/jtrhcmkv/jtrhcmkv.php."

[4] "http://www.zdnet.com/blog/security/chinese-hacker-arrested-for-leaking-6-million-logins/11064."

[5] "Yahoo! password leakege," *http://www.cnet.com/news/yahoos-password-leak-what-you-need-to-know-faq/*.

[6] "Gmail password leakage," *http://lifehacker.com/5-million-gmail-passwords-leaked-check-yours-now-1632983265*.

[7] D. Florêncio and C. Herley, "Where do security policies come from," *SOUPS*, 2010.

[8] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," *CCS*, 2010.

[9] X. C. Carnavalet and M. Mannan, "From very weak to very strong: Analyzing password-strength meters," *NDSS*, 2014.

[10] S. Ji, S. Yang, and R. Beyah, "Pars: A uniform and open-source password analysis and research system," *ACSAC*, 2015.

[11] J. Yan, A. Blackwell, and R. Anderson, "Password memorability and security: Empirical results," *S&P*, 2004.

[12] S. Ji., S. Yang, X. Hu, W. Han, Z. Li, and R. Beyah, "Zero-sum password cracking game: A large-scale empirical study on the crackability, correlation, and security of passwords," *Dependable and Secure Computing, IEEE Transactions on*, 2015.

[13] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," *NDSS*, 2014.

[14] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probilistic password models," *S&P*, 2014.

[15] "Additional supporting materials." [Online]. Available: https://www.dropbox.com/sh/e2qsvlca7cep7vw/AACEtntleyXE8OoitoIAUNhka?dl=0

[16] "John the ripper-bleeding-jumbo," *https://github.com/magnumripper/JohnTheRipper*.

[17] M. Dürmuth, A. Chaabane, D. Perito, and C. Castelluccia, "When privacy meets security: Leveraging personal information for password cracking," *CoRR abs/1304.6584*, 2013.

[18] M. Weir, S. Aggarwal, B. Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," *S&P*, 2009.

[19] "Amazon mechanical turk," *https://www.mturk.com/*.

[20] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive passwords-strength meters from markov models," *NDSS*, 2012.

[21] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Statistical Mechanics: Theory and Experiment*, 2008.

[22] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Crano, "Encountering stronger password requirements: User attitudes and behaviors," *SOUPS*, 2010.

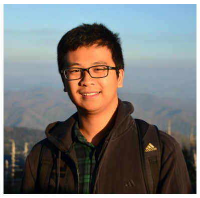
[23] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," *CCS*, 2013.

[24] J. H. Huh, S. Oh, H. Kim, K. Beznosov, A. Mohan, and S. R. Rajagopalan, "Surpass: System-initiated user-replaceable passwords," *CCS*, Dissertation.

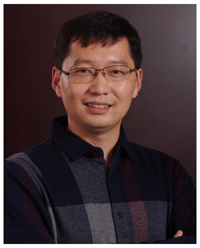
[25] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. López, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," *S&P*, 2012.

[26] S. Houshmand and S. Aggarwal, "Building better passwords using probabilistic techniques," *ACSAC*, 2012.

[27] S. Komanduri, R.Shay, L. F. Cranor, C. Herley, and S. Schechte, "Telepathwords: Preventing weak passwords by reading users' minds," *USENIX*, 2014.
[28] A. Forget, S. Chiasson, P. C. V. Oorschot, and R. Biddle, "Improving text passwords through persuasion," *SOUPS*, 2008.
[29] D. Schmidt and T. Jaeger, "Pitfalls in the automated strenghting of passwords," *ACSAC*, 2013.
[30] J. Camenisch, A. Lehmann, and G. Neven, "Optimal distributed password verification," *CCS*, 2015.
[31] S. Schechter, C. Herley, and M. Mitzenmacher, "Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks," *USENIX HotSec'10*, 2010.
[32] S. Gaw and E. W. Felten, "Password management strategies for online accounts," *SOUPS*, 2006.
[33] E. Stobert and R. Biddle, "The password life cycle: User behaviour in managing passwords," *SOUPS*, 2014.



Shukun Yang is a Master student in the School of Electrical and Computer Engineering at Georgia Institute of Technology. He received his Bachelor of Science in Electrical and Computer Engineering with highest honor from Georgia Institute of Technology. His research interests include passwords security, social network privacy, and machine learning. He is a student member of ACM and IEEE.



Shouling Ji is a ZJU 100-Young Professor in the College of Computer Science and Technology at Zhejiang University and a Research Faculty in the School of Electrical and Computer Engineering at Georgia Institute of Technology. He received a Ph.D. in Electrical and Computer Engineering from Georgia Institute of Technology, and a Ph.D. in Computer Science from Georgia State University. His research interests include Big Data Driven Security, Privacy, Adversarial Learning, Graph Theory and Algorithms, and Wireless Networks. He is a member of ACM and IEEE.

Raheem Beyah is the Motorola Foundation Professor in the School of Electrical and Computer Engineering at Georgia Tech, where he leads the Communications Assurance and Performance Group (CAP) and is a member of the Communications Systems Center (CSC). He received his Masters and Ph.D. in Electrical and Computer Engineering from Georgia Tech in 1999 and 2003, respectively. His research interests include network security, wireless networks, network traffic characterization and performance, and critical infrastructure security. He received the National Science Foundation CAREER award in 2009 and was selected for DARPAs Computer Science Study Panel in 2010. He is a member of AAAS and ASEE, is a lifetime member of NSBE, and is a senior member of ACM and IEEE.