A Study of Personal Information in Human-chosen Passwords and Its Security Implications

ABSTRACT

While it is not recommended, Internet users often include parts of personal information in their passwords for easy memorization. However, the use of personal information in passwords and its security implications have not yet been studied systematically in the past. In this paper, we first dissect user passwords from a leaked dataset to investigate how and to what extent user personal information resides in a password. In particular, we extract the most popular password structures expressed by personal information and show the usage of personal information. Then we introduce a new metric called Coverage to quantify the correlation between passwords and personal information. Afterwards, based on our analysis, we extend the Probabilistic Context-Free Grammars (PCFG) method to be semantics-rich and propose Personal-PCFG to crack passwords by generating personalized guesses. Through offline and online attack scenarios, we demonstrate that Personal-PCFG cracks passwords much faster than PCFG and makes online attacks much easier to succeed. To defend against such semantics-aware attacks, we propose to use simple distortion functions that are chosen by users to mitigate unwanted correlation between personal information and passwords. Our experimental results show that a simple distortion can effectively protect a password from personalized cracking without sacrificing usability.

1. INTRODUCTION

Text-based passwords still remain a dominating and irreplaceable authentication method in foreseeable future. Although people have proposed different authentication mechanisms, no alternative can bring all benefits of passwords without introducing any extra burden to users [3]. However, passwords have long been criticized to be one of the weakest link in authentication. Due to human-memorability requirement, user passwords are usually far from true random strings [2, 27, 31, 38, 41]. For example, "secret" is more likely a human-chosen password than "zjorqpe". In other words, human users are prone to choose weak passwords simply because they are easy to remember. As a result, most passwords are chosen within only a small portion of the entire large password space, being vulnerable to brute-force and dictionary attacks.

To increase password security, online authentication systems start to enforce more strict password policies. Meanwhile, currently most websites have password strength meters deployed to help users choose secure passwords. However, these meters are proved to be ad-hoc and inconsistent [13]. To better assess the strength of passwords, we need to have a deeper understanding on how users construct their passwords. If an attacker knows exactly how users create their passwords, guessing their passwords will become much easier. Meanwhile, if a user is aware of the potential vulnerability induced by a commonly used password creation method, the user can avoid using the same method for creating passwords.

Toward this end, researchers have done a lot of work to unveil the structures of passwords. Traditional dictionary attacks on passwords have shown that users tend to use simple dictionary words to construct their passwords [17, 30]. Languages are also vital since users tend to use their own languages when constructing passwords [2]. Besides, passwords are mostly phonetically memorable [31] even they are not simple dictionary words. It is also indicated that users may use keyboard strings such as "qwerty" and "qweasdzxc", trivial strings such as "password", "123456", and date strings such as "19951225" in their passwords [26, 36, 38]. However, most studies only discover superficial password patterns, and the semantic-rich composition of passwords is still mysterious to be fully uncovered. Fortunately, an enlightening work investigates on how users generate their passwords by learning the semantic patterns in passwords [37].

In this paper, we study password semantics from a different perspective—personal information. We utilize a leaked password dataset, which contains personal information, from a Chinese website for this study. We first measure the usage of personal information in password creation and present interesting observations. We are able to obtain the most popular password structures with personal information embedded. We also observe that male and female behave differently when using personal information in password creation. Next, we introduce a new metric called Coverage to accurately quantify the correlation between personal information and user password. Since it considers both the length and continuation of personal information in a password, Coverage is a useful metric to measure the strength of a password.

.

Our quantification results using the Coverage metric confirm with our direct measurement results on the dataset, showing the efficacy of Coverage. Moreover, Coverage is easy to apply and to be integrated with existing tools, such as password strength meters to uncover the unnoticed corner of personal information.

To demonstrate the security vulnerability induced by using personal information in passwords, we propose a semantics-rich Probabilistic Context-Free Grammars (PCFG) method called Personal-PCFG, which extends PCFG [40] by considering personal information symbols in password structures. Personal-PCFG is able to crack passwords much faster than PCFG. It also makes an online attack more feasible by drastically increasing guessing success rate. Finally, we present simple distortion functions to defend against these semantics-aware attacks such as Personal-PCFG or [37]. Our evaluation results demonstrate that distortion functions can effectively protect passwords by significantly reducing the unwanted correlation between personal information and passwords.

Our study is based on a dataset collected from a Chinese website. Although measurement results could be different in some aspects if other datasets are used, our observations still show a valid grand picture on how personal information is used in passwords. As long as memorability plays an important role in password creation, the correlation between personal information and user password remains, regardless of which language users speak. We believe that our work on personal information quantification, password cracking, and password protection could be applicable for any other text-based password datasets from different websites.

The remainder of this paper is organized as follows. Section 2 measures how personal information resides in user passwords and shows the gender difference in password creation, based on a leaked dataset from an online ticket reservation website. Section 3 introduces the new metric, Coverage, to accurately quantify the correlation between personal information and user password. Section 4 details Personal-PCFG and shows cracking results compared with the original PCFG. Section 5 presents our defense against semantics-aware attacks by applying simple distortion functions on password creation. Section 6 surveys related work, and finally Section 7 concludes this paper.

2. PERSONAL INFO IN PASSWORDS

Intuitively, people tend to create their passwords based on their personal information since human beings are limited by their memory – true random passwords are much harder to remember. We show that user personal information plays an important role in human-chosen password generation by dissecting passwords in a mid-sized leaked password dataset. Understanding the usage of personal information in passwords and its security implications can help us to further enhance password security. To start with, we introduce the dataset used throughout this study.

2.1 12306 Dataset

A number of password datasets have been exposed to the public in recent years, and they usually contain several thousands to millions of real passwords. As a result, there are several password measurement or password cracking works based on analyzing those datasets [2, 26]. In this paper, a dataset called 12306 is used to illustrate how personal information is involved in password creation.

Table 1: Most Frequent Passwords.

Rank	Password	Amount	Percentage
1	123456	389	0.296%
2	a123456	280	0.213%
3	123456a	165	0.125%
4	5201314	160	0.121%
5	111111	156	0.118%
6	woaini1314	134	0.101%
7	qq123456	98	0.074%
8	123123	97	0.073%
9	000000	96	0.073%
10	1qaz2wsx	92	0.070%

2.1.1 Introduction to Dataset

At the end of year 2014, a Chinese dataset is leaked to the public by anonymous attackers. It is reported that the dataset is collected by trying passwords from other leaked passwords. We call this dataset 12306 because all passwords are from a website www.12306.cn, which is the official website of online railway ticket reservation system for Chinese users. There is no data available showing the exact number of users in the 12306 website; however, we infer at least tens of millions users registered in the system since it is the only official website for the entire Chinese railway systems.

The 12306 dataset contains over 130,000 Chinese passwords. Having witnessed so many large datasets leaked out, the size of the 12306 dataset is just medium. What makes it special is that together with plaintext passwords, the dataset also includes several types of user personal information, such as a user's name and the government-issued unique ID number (similar to the Social Security Number in USA). As the website requires a real ID number to register and people must provide real personal information to book a ticket, we consider that information in this dataset pretty reliable.

2.1.2 Basic Analysis

We first conduct a simple analysis to reveal some general characteristics of the 12306 dataset. For data consistency, we remove users whose ID number is not 18-digit long. These users may have used other IDs (e.g., passport number) to register the web system and count 0.2% of the whole dataset. The dataset contains 131,389 passwords for analysis after being cleansed. Note that various websites may have different password creation policies. For instance, with a strict password policy, users may apply mangling rules (e.g., $abc -> @bc$ or $abc1$) to their passwords to fulfill the policy requirement [39]. Since the 12306 website has changed its password policy after the password leakage, we do not know the exact password policy when the dataset was leaked. However, from the leaked dataset, we infer that the password policy is quite simple – all passwords cannot be shorter than 6 symbols. There is no restriction on what type of symbols are used. Therefore, users are not required to apply any mangling rule to their passwords.

The average length of passwords in the 12306 dataset is 8.44. The most common passwords in the 12306 dataset are listed in Table 1. The dominating passwords are trivial passwords (e.g., 123456, a123456, etc.), keyboard passwords (e.g., 1qaz2wsx, 1q2w3e4r, etc.), and "I love you" type

Table 2: Most Frequent Password Structures.

Rank	Structure	Amount	Percentage
1	D_7	10893	8.290%
2	D_8	9442	7.186%
3	D_6	9084	6.913%
4	L_2D_7	5065	3.854%
5	L_3D_6	4820	3.668%
6	L_1D_7	4770	3.630%
7	L_2D_6	4261	3.243%
8	L_3D_7	3883	2.955%
9	D_9	3590	2.732%
10	L_2D_8	3362	2.558%

"D" represents digits and "L" represents English letters. The number indicates the segment length. For example, L_2D_7 means the password contains 2 letters following by 7 digits.

passwords. Both "5201314" and "woaini1314" means "I love you forever" in Chinese. The most commonly used Chinese passwords are similar to a previous study [26]; however, the 12306 dataset is much less congregated. The most popular password "123456" counts less than 0.3% of all passwords while the number is 2.17% in [26]. We believe that the password sparsity is due to the importance of the website so that users are less prone to use trivial passwords like "123456" and there is fewer sybil accounts because real ID number is needed for registration.

We also study the basic structures of the passwords in 12306. The most popular password structures are shown in Table 2. Similar to a previous study [26], our result again shows that Chinese users prefer to use digits in their passwords instead of letters as English-speaking users. The top five structures all have significant portion of digits, and at most 2 or 3 letters are appended in front. We reckon that the reason behind may be Chinese characters are logogram-based and digits seem to be the best alternative when creating a password.

Overall, the 12306 dataset is a Chinese password dataset that has general Chinese password characteristics. However, its passwords are more sparse than previously studied datasets because users concern password security more when creating passwords for critical service systems.

2.2 Personal Information

The 12306 dataset not only contains user passwords, but also include multiple types of personal information listed below.

```
1. Name: User's Chinese name.
2. Email address: User's registered email address.
3. Cellphone: User's registered cellphone number.
4. Account name: the username used in the system,
may contain digits and letters, like "myacct123".
5. ID number: Government issued ID number.
```

Note the government issued ID number is an 18-digit unique number, which includes personal information itself. The digits 1-6 represent the birth place of the owner, the digits 7-14 represent the birthday of the owner, and the digit 17 represents the gender of the owner – odd number means male and even number means female. We take out the 8-digit birthday information and treat it separately since birthday is very important personal information in password creation. Therefore, we finally have six types of personal information: name, birthday, email address, cellphone number, account name, and ID number (birthday excluded).

2.2.1 *New Password Representation*

To better illustrate how personal information correlates to user passwords, we develop a new representation of password by adding more semantic symbols besides the conventional "D", "L" and "S" symbols, which stand for digit, letter, and special symbol, respectively. We try to match parts of a password to the six types of user personal information, and express the password with these personal information. For example, a password "alice1987abc" can be represented as $[Name][Birthday]L_3$, instead of $L_3D_4L_3$ as in a traditional representation. The matched personal information is denoted by corresponding tags – [Name] and [Birthday] in this example; for segments that are not matched, we still use "D", "L", and "S" to describe the symbol types.

We believe that the representation like $[Name][Birthday]L_3$ is better than $L_5D_4L_3$ since it more accurately describes the composition of a user password with more detailed semantic information. Using this representation, we apply the following matching method to the entire 12306 dataset to see how these personal information tags appear in password structures.

2.2.2 *Matching Method*

We propose a matching method to locate personal information in a user password, which is shown in Algorithm 1. The high level idea is that we first generate all substrings of the password and sort them in descending length order. Then we match these substrings from the longest to the shortest to all types of personal information. If one match is found, the match function is recursively applied over the remaining password segments until no further match is found. The segments that are not matched to any personal information will be then labeled using the traditional "LDS" tags.

In Algorithm 1, we first make sure that the length of a password segment is at least 2 for matching. Then we try to match eligible segments to each kind of the personal information. Note that we do not present the specific matching methods for each type of the personal information (line 10 and line 16) to keep it clean and simple. Instead we describe the methods as follows. For the Chinese names, we convert them into Pinyin form, which is alphabetic representation of Chinese. Then we compare password segments to 10 possible permutations of a name, such as lastname+firstname and last_initial+firstname. If the segment is exactly same as anyone of the permutations, we consider a match is found. For birthday, we list 17 possible permutations and compare a password segment with these permutations. If the segment is the same as any permutation, we consider it as a match. For account name, Email address, cellphone number, and ID number, we further restrain the length of a segment to be at least 3 to avoid mismatching by coincidence. Besides, as people tend to memorize a sequence of numbers by dividing it into 3-digit groups, We believe that a match of at least length 3 is likely to be a real match.

Note that for a password segment, it may match to multiple types of personal information. In such cases, all matches are counted. The results of Algorithm 1 contain all possible matches.

Algorithm 1 Personal Information Matching.

```
procedure MATCH($pwd$,$infolist$)
    $newform \leftarrow$ empty_string
    if len($pwd$) == 0 then
        return empty_string
    end if
    $substring \leftarrow$ get_all_substring($pwd$)
    reverse_length_sort($substring$)
    for $eachstring \in substring$ do
        if len($eachstring$) $\geq 2$ then
            if matchbd($eachstring$,$infolist$) then
                $tag \leftarrow$ "[BD]"
                $leftover \leftarrow pwd$.split($eachstring$)
                break
            end if
            ...
            if matchID($eachstring$,$infolist$) then
                if tag != None then
                    $tag \leftarrow tag +$ "&[$ID$]"
                else
                    $tag \leftarrow$ "[ID]"
                end if
                $leftover \leftarrow pwd$.split($eachstring$)
                break
            end if
        else
            break
        end if
    end for
    if $leftover$.size() $\geq 2$ then
        for i $\leftarrow$ 0 to $leftover$.size()-2 do
            $newform \leftarrow$ MATCH($leftover[i]$,$infolist$) + $tag$
        end for
        $newform \leftarrow$ MATCH($leftover[leftover.size()-1]$)+$newform$
    else
        $newform \leftarrow$ seg($pwd$)
    end if
    $results \leftarrow$extract_ambiguous_structures($newform$)
    return $results$
end procedure
```

2.2.3 Matching Results

After applying Algorithm 1 to the 12306 dataset, we find that 78,975 out of 131,389 (60.1%) of the passwords contain at least one of the six types of personal information. Apparently, personal information is frequently used in password creation. We believe that the ratio could be even higher if we have more personal information at hand. We present the top 10 password structures in Table 3 and the usage of personal information in passwords in Table 4. As mentioned above, a password segment may match to multiple types of personal information and we count all these matches. Therefore, with 131,389 passwords we obtain 153,895 password structures. Based on Tables 3 and 4, we have the following observations.

Table 3: Most Frequent Password Structures.

Rank	Structure	Amount	Percentage
1	[ACCT]	6820	5.190%
2	D7	6224	4.737%
3	[NAME][BD]	5410	4.117%
4	[BD]	4470	3.402%
5	D6	4326	3.292%
6	[EMAIL]	3807	2.897%
7	D8	3745	2.850%
8	L1D7	2829	2.153%
9	[NAME]D7	2504	1.905%
10	[ACCT][BD]	2191	1.667%

Table 4: Personal Information Usage.

Rank	Information Type	Amount	Percentage [1]
1	Birthday	31674	24.10%
2	Account Name	31017	23.60%
3	Name	29377	22.35%
4	Email	16642	12.66%
5	ID Number	3937	2.996%
6	Cell Phone	3582	2.726%

1. Even for security sensitive websites like 12306, people tend to use personal information for password creation. 5 out of the top 10 structures are composed by pure personal information, and 6 out of the top 10 structures have personal information segments.

2. Birthday, account name, and name are the most popular personal information contained in user passwords, with 24.10%, 23.60%, and 22.35% occurrence rates, respectively. Meanwhile, about 12.66% of users include email in their passwords. However, only few percentage of people include their cellphone and ID number in their passwords (less than 3%).

3. Digits are still dominating user passwords. The dominating structures D_7, D_6, and D_8 in Table 2 still rank fairly high. Only one structure from the top 10 structures contains letter segment. The result confirms that Chinese users prefer to use digits in their passwords.

4. The structure [ACCT] ranks the highest among all structures. The reason may be that users tend to directly use their account names as their passwords, instead of using them as part of their passwords.

2.2.4 Gender Password Preference

As the user ID number in our dataset actually contains gender information (i.e., the second last digit in the ID number represents gender), we compare the password structures between males and females to see if there is any difference on password preference. Since the dataset is biased in gender with 9,856 females and 121,533 males, we randomly select 9,856 males from the male pool and compare them with females.

The average password lengths for males and females are 8.41 and 8.51, respectively, which shows that males and females do not differ much in the length of their passwords. We then apply the matching method to each gender. We observe that 61.0% of male passwords contain personal information while only 54.1% of female passwords contain personal information. We list the top 10 structures for each gen-

[1]Note that one password segment can be matched to multiple types of personal information. If so, all matched types are counted. Therefore the sum of the percentages is larger than 60.1%

Table 5: Most Frequent Structures in Different Gender.

Rank	Male		Female	
	Structure	Percentage	Structure	Percentage
1	[ACCT]	4.647%	D6	3.909%
2	D7	4.325%	[ACCT]	3.729%
3	[NAME][BD]	3.594%	D7	3.172%
4	[BD]	3.080%	D8	2.453%
5	D6	2.645%	[EMAIL]	2.372%
6	[EMAIL]	2.541%	[NAME][BD]	2.309%
7	D8	2.158%	[BD]	1.968%
8	L1D7	2.088%	L2D6	1.518%
9	[NAME]D7	1.749%	L1D7	1.267%
10	[ACCT][BD]	1.557%	L2D7	1.240%
NA	TOTAL	28.384%	TOTAL	23.937%

Table 6: Most Frequent Personal Information in Different Gender.

Rank	Male		Female	
	Information Type	Percentage	Information Type	Percentage
1	[BD]	24.56%	[ACCT]	22.59%
2	[ACCT]	23.70%	[BD]	20.56%
3	[NAME]	23.31%	[NAME]	12.94%
4	[EMAIL]	12.10%	[EMAIL]	13.62%
5	[ID]	2.698%	[CELL]	2.982%
6	[CELL]	2.506%	[ID]	2.739%

der in Table 5 and personal information usage in Table 6. These results demonstrate that male users are more likely to include personal information in their passwords than female users. In addition, we have the following two interesting observations:

1. As Table 5 shows, 28.38% of males' passwords fall into the category of the top 10 structures but there are only 23.94% of female passwords whose structures belong to the top 10. Moreover, the total number of different password structures of females is 1,756, which is 10.3% more than the total number of males. Thus, the passwords of males are more congregated and hence tend to be more predictable.

2. Males and females vary in the usage of name information, as shown in Table 6. Males use their names as frequent as their birthday (23.31% passwords of males contain their names). In contrast, only 12.94% of females' passwords contain their names. We also notice that the name usage is the dominating factor that causes the difference in personal information between males and females. In other words, except for name usage, male and female have no significant difference in using personal information for password creation.

In summary, passwords of males are generally composed of more personal information, especially names of the users. In addition, the password diversity for males is lower. Our analysis indicates that the passwords of males are more vulnerable than those of females. At least from the perspective of personal-information-related attacks, our observations are different from the conclusion drawn in [29] that males have slightly stronger passwords than females.

2.3 Domain Information

Cao et al. [8] proposed an interesting idea that uses domain information to crack user passwords. It draws our attention since we have shown the involvement of personal information in user password and naturally we are also interested in the involvement of the domain information as another aspect of semantic information in password creation. By domain information, we mean the information of an Internet domain, e.g., a web service. For example, the famous "Rockyou" dataset is leaked from a website *www.rockyou.com*, the domain information here is "rockyou". In our personal information study, the domain information is "12306". It is reasonable for users to include domain information in their passwords to keep their passwords different from site to site but still easy to remember. This approach is promising to balance password security and memorability; however, the idea has not been validated with a large scale experiments. Therefore, we attempt to verify whether domain information is involved in password creation as personal information. In addition to the medium-sized 12306 dataset, we study more password datasets including Tianya, Rockyou, PHPBB, and MySpace datasets. In each dataset, we search the domain information and its meaningful substrings in the passwords, and the results are shown in Table 7.

Table 7: Domain Information in Passwords.

Dataset	Password Amount	Domain info Amount	Percentage
Rockyou	14,344,391	44,025	0.3%
Tianya	26,832,592	29,430 [2]	0.11%
PHPBB	184,389	2,209	1.2%
12306	131,389	490	0.4%
MySpace	37144	72	0.2%

From Table 7, we can see that indeed some users include domain information in their passwords. Our results indicate that all the datasets examined contain 0.11% to 1.2% of passwords that relate to domain information. However, the small percentage indicates that though including domain information in a password helps users to have different passwords for different websites, currently not many users are using such a method to construct their passwords.

3. CORRELATION QUANTIFICATION

While the statistical numbers above show the correlation between each type of personal information and passwords, they cannot accurately measure the degree of personal information involvement in an individual password. In [9], Longest Common Sequence and a modified Jaccard similarity (JS) index are used to measure the correlations. However, their results end up having two metrics for each type of personal information. Besides, their metrics fail to differentiate fragmented and continued personal information. Thus, we introduce a new metric—Coverage—to quantify the involvement of personal information in the creation of an individual password in an accurate and systematic fashion.

3.1 Coverage

The value of Coverage ranges from 0 to 1. A larger Coverage implies a stronger correlation, and Coverage "0" means no personal information is included in a password and Coverage "1" means the entire password is perfectly matched

[2]We find that 45,574 passwords in the Tianya dataset is "111222tiany". It does not make much sense for so many users using the same password and highly likely they are sybil accounts. Thus, these duplications are removed from our analysis.

with one type of personal information. While Coverage is mainly used for measuring an individual password, the average Coverage also reflects the degree of correlation in a set of passwords. In the following, we describe the algorithm to compute Coverage, present a detailed example to illustrate how Coverage works, and elaborate the key features of Coverage.

3.1.1 Computation Method

To compute Coverage, we take password and personal information in terms of strings as input and use a sliding window approach to conducting the computation. We maintain a dynamic-sized window sliding from the beginning to the end of the password. The initial size of the window is 2. If the segment covered by the window matches to a certain type of personal information, we enlarge the window size by 1. Then we try again to match the segment in the grown window to personal information. If a match is found, we further enlarge the window size until a mismatch happens. At this point, we reset the window size to the initial value 2 and slide the window to the password symbol that causes the mismatch in the previous window. We maintain an array called *tag array* with the same length as the password to record the length of each matched password segment. For example, assume a password with length of 8, its tag array is $[4_1, 4_2, 4_3, 4_4, 0, 0, 2_1, 2_2]$. The first four elements in the array, i.e., $\{4_1, 4_2, 4_3, 4_4\}$, indicate that the first 4 password symbols match certain type of personal information. The following two elements in the array, i.e., {0,0}, indicate that the 5th and 6th symbols have no match. The last two elements in the array, i.e., $\{2_1, 2_2\}$, indicate that the 7th and 8th symbols again match certain type of personal information. The personal information types matched with different password segments may or may not be the same. After we eventually slide window through the entire password string, the tag array is used to compute the value of Coverage—the sum of squares of matched password segment length divided by the square of password length. Mathematically we have

$$CVG = \sum_{i=1}^{n} (\frac{l_i^2}{L^2}), \tag{1}$$

where n denotes the number of matched password segments, l_i denotes the length of corresponding matched password segment, and L is the length of the password. We show the algorithm of computing Coverage in Algorithm 2. A match is found if at least a 2-symbol long password segment matches to a substring of certain personal information.

To better illustrate how Coverage is computed, we show a simple example in Figure 1. Here we assume a user named Alice, who was born on August 16, 1988. We also assume that her password happens to be "alice816". If we apply the matching algorithm in Section 2.2.3, the structure of this password will be [NAME][BD]. Apparently her password is highly related to her personal information. To quantify this correlation, we follow the Algorithm 2 to compute Coverage for her. The computation steps are shown in Figure 1, and each step is detailed as follows. In step (a), the password is prepared for a window to slide through. Note the personal information includes "alice" as name and "19880816" as birthday. The tag array is initialized as [0, 0, 0, 0, 0, 0, 0, 0]. In step (b), the window size is initialized to 2 so that the first two symbols in the password are covered. As "al" is a substring of Alice's name, a match is found. Therefore, we extend the window size by 1 and the tag array is updated as $[2_1, 2_2, 0, 0, 0, 0, 0, 0]$. From step (c) to step (e), the window keeps growing since matches are continuously found. The tag array also keeps being updated. Until step (f), the window now covers "alice8", which is not a substring of "alice" or "19880816." Therefore, the window size is reset to 2 and the window slides to the position of the symbol of the previous window that causes the mismatch (i.e., the position of "8"). The tag array remains unchanged. In step (g), the window of size 2 now covers "81", which is a substring of birthday, again we extend the window by 1 and update the tag array to $[5_1, 5_2, 5_3, 5_4, 5_5, 2_1, 2_2, 0]$. After the window grows by 1 in step (h), "816" is again found as a match. The tag array is updated to $[5_1, 5_2, 5_3, 5_4, 5_5, 3_1, 3_2, 3_3]$. The window does not grow or slide any more since it has reached the end of the password. In the last step (i), the tag array is ready to be used in computing the Coverage value. Based on Equation 1, the coverage is computed as

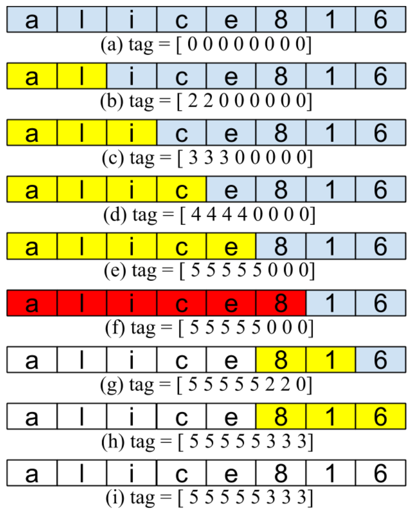
$$CVG = \sum_{i=1}^{2} \frac{l_i^2}{L^2} = \frac{5^2 + 3^2}{8^2} = 0.52.$$

Coverage is independent of password datasets. As long as we can build a complete string list of personal information, Coverage can accurately quantify the correlation between a user's password and its personal information. For personal information segments with the same length, Coverage stresses the continuation of matching. A continuous match is stronger than fragmented matches. That is to say, for a given password of length L, a matched segment of length l ($l \leq L$) has stronger correlation to personal in-

Algorithm 2 Compute Coverage.

```
1: procedure CVG($pwd$,$infolist$)
2:     $windowsize \leftarrow 2$
3:     $pwdlen \leftarrow$ len($pwd$)
4:     $matchtag \leftarrow [0]*pwdlen$
5:     $matchmark \leftarrow 0$
6:     $cvg \leftarrow 0$
7:     while $windowsize \leq$len($pwd$) do
8:         $passseg \leftarrow pwd[0 : windowsize]$
9:         if $passseg$ = substring of $anyinfo$ in $infolist$ then
10:            for $j \leftarrow matchmark$ to $matchmark$+$windowsize$ do
11:                $matchtag[j] \leftarrow windowsize$
12:            end for
13:            if $windowsize$ != len($pwd$) then
14:                $windowsize \leftarrow windowsize+1$
15:            end if
16:        else
17:            $matchmark \leftarrow matchmark+windowsize$
18:            $pwd \leftarrow pwd[windowsize :]$
19:            $windowsize \leftarrow 2$
20:        end if
21:    end while
22:    for $eachitem$ in $matchtag$ do
23:        $cvg \leftarrow cvg + eachitem$
24:    end for
25:    return $cvg/(pwdlen * pwdlen)$
26: end procedure
```

Figure 1: Coverage - An Example.


a l i c e 8 1 6
(a) tag = [0 0 0 0 0 0 0 0]
(b) tag = [2 2 0 0 0 0 0 0]
(c) tag = [3 3 3 0 0 0 0 0]
(d) tag = [4 4 4 4 0 0 0 0]
(e) tag = [5 5 5 5 5 0 0 0]
(f) tag = [5 5 5 5 5 0 0 0]
(g) tag = [5 5 5 5 5 2 2 0]
(h) tag = [5 5 5 5 5 3 3 3]
(i) tag = [5 5 5 5 5 3 3 3]

Grey boxes hold unvisited password symbols. Yellow and red boxes denote that the symbols inside are covered by the sliding window. White boxes denote the symbols inside have been settled (window stops extending).

formation than two matched segments of length l_1 and l_2 with $l = l_1 + l_2$. For example, a matched segment of length 6 is expected to have stronger correlation than 2 matched segments of length 3. This feature of Coverage is desirable because multiple shorter segments (i.e., originated from different types of personal information) are usually harder to guess and may involve a wrong match due to coincidence. Since it is difficult to differentiate a real match from a coincident match, we would like to minimize the effect of wrong matches by taking squares of the matched segments to compute Coverage, in favor of a continuous match.

3.2 Coverage Results on 12306

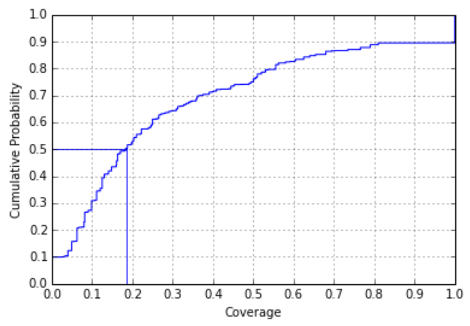
We compute the Coverage value for each user in the 12306 dataset, and the result is shown as a cumulative distribution function in Figure 2. To easily understand the value of Coverage, we discuss a few examples to illustrate the implication of a roughly 0.2 Coverage. Suppose we have a 10-symbol long password. One matched segment with length 5 will yield 0.25 Coverage. Two matched segments with length 3 (i.e., in total 6 symbols are matched to personal information) yield 0.18 Coverage. Moreover, 5 matched segments with length 2 (i.e., all symbols are matched but in a fragmented fashion) yield 0.2 Coverage. Apparently Coverage of 0.2 indicates a fairly high correlation between personal information and a password. Furthermore, we have the following observations from Figure 2.

1. The median value for a user's Coverage is 0.186, which implies that a significant portion of user passwords have relatively high correlation to personal information.

2. Around 9.9% of users have zero Coverage. Zero Coverage indicates that no personal information is included in a user password.

3. Around 10.5% of users have Coverage of 1, which means that 10.5% of passwords are perfectly matched to exactly one type of personal information. However, only 13.4% users fall in the range [0.5, 1)

Figure 2: Coverage distribution - 12306.



The average Coverage for the entire 12306 dataset is 0.309. We also compute the average Coverages for male and female groups, since we observe that male users are more likely to include personal information in their passwords in Section 2.2.4. The average Coverage for the male group is 0.314 and the average Coverage for the female group is 0.269. It complies with our previous observation and indicates that the correlation for male users is higher than that of female users. Conversely, it also shows that Coverage works very well to quantify the correlation between passwords and personal information.

3.3 Coverage Usage

Apparently Coverage could be very useful for constructing password strength meters, which have been reported as mostly ad-hoc [13]. Most meters give scores dumbly based on password structure and length or blacklist commonly used passwords (e.g., the notorious "password"). There are also meters that perform simple social profile analysis, such as a password cannot contain the user's names or the password cannot be the same as the account name. However, these simple analysis mechanisms can be easily fooled around by mangling a password a little, while the password remains weak. Using the metric of Coverage, password strength meters can be improved to more accurately measure the strength of a password. Moreover, it is straightforward to implement Coverage as a part of strength measurement (only a few lines of Javascript should do). More importantly, since users cannot easily defeat Coverage measurement through simple mangling methods, they are forced to select more secure passwords.

Coverage can also be integrated into existing tools to enhance their capabilities. There are several Markov Model based tools that predict the next symbol when a user creates a password [23, 39]. These tools rank the probability of the next symbol based on the Markov model learned from dictionaries or leaked datasets, and then show the most probable predictions. Since most users would be surprised to find that the next symbol in their mind matches exactly the tool's output, they may switch to choose a more unpredictable password symbol. Coverage helps to determine if personal information prediction ranks high enough in probability to remind a user of avoiding the use of personal information in password creation.

4. PERSONAL-PCFG

After investigating the correlation between personal information and user passwords through measurement and quantification, we further study their potential usage to crack passwords from an attacker's point of view. Based on PCFG approach [40], we develop Personal-PCFG as an individual-oriented password cracker that can generate personalized guesses towards a targeted user by exploiting the already known personal information.

4.1 Attack Scenarios

We assume that the attacker knows certain amount of personal information about the targets. The attacker can be an evil neighbor, a curious friend, a jealous husband, a blackmailer, or even a company that buys personal information from other companies. Under these conditions, targeted personal information is rather easy to obtain by knowing the victim personally or searching online, especially on social networks sites (SNS) [16, 24]. Personal-PCFG can be used in both offline and online attacks.

In traditional offline password attacks, attackers usually steal hashed passwords from victim systems, and then try to find out the unhashed values of these passwords. As a secure hash function cannot be simply reversed, the most popular attacking strategy is to guess and verify passwords by brute force. Each guess is verified by hashing a password (salt needs to be added) from a password dictionary and comparing the result to the hashed values in the leaked password database. High-probability password guesses can usually match many hashed values in the password database and thus are expected to be tried first. For offline attacks, Personal-PCFG is much faster in guessing out the correct password than conventional methods since it can generate high-probability personalized passwords and verify them first.

For online attack, since the attacker does not even have a hashed password database, he or she instead tries to directly log in the real systems by guessing the passwords. It is more difficult to succeed in online attacks than offline attacks because online service systems usually have restrictions on login attempts for a given period of time. If the attempt quota has been reached without inputting a correct password, the account may be locked for some time or even permanently unless certain actions are taken (e.g., call the service provider). Therefore, online attacks require very accurate guesses, which can be achieved by integrating personal information. Personal-PCFG is able to crack around 1 out of 20 passwords within only 5 guesses.

4.2 A Revisit of PCFG

Personal-PCFG is based on the basic idea of PCFG method [40] and provide an extension to further improve its efficiency. Before we introduce Personal-PCFG, we briefly revisit principles of PCFG. PCFG pre-processes passwords and generates base password structures such as "$L_5D_3S_1$" for each of the passwords. Starting from high-probability structures, PCFG method substitutes the "D" and "S" segments using segments of same length learned from the training set. These substitute segments are ranked by probability of occurring learned from the training set. Therefore high probability segments will be tried first. One base structure may have a number of substitutions, for example, "$L_5D_3S_1$" can have "$L_5$123!" and "$L_5$691!" as its substitutions. These new representations are called pre-terminal structures. No "L" segment is currently substituted since the space of alpha strings is too large to learn from the training set. Next, these pre-terminals are ranked from high probability to low probability. Finally "L" segments are substituted using a dictionary to generate actual guesses. Besides the basic idea, PCFG method also carries an efficient algorithm to enumerate passwords from high probability to low probability on the fly. These guesses are hashed to compare with the values in password databases. Since PCFG can generate statistically high probability password first, it can significantly reduce the guessing number of traditional dictionary attacks.

4.3 Personal-PCFG

Personal-PCFG leverages the basic idea of PCFG. Besides "L", "D", and "S" symbols in PCFG, we add more semantic symbols including "B" for birthday, "N" for name, "E" for email address, "A" for account name, "C" for cellphone number, and "I" for ID number. Richer semantics makes Personal-PCFG more accurate in guessing passwords. To make Personal-PCFG work, an additional personal information matching phase and an adaptive-substitution phase are added to the original PCFG method. Therefore, Personal-PCFG has 4 phases and the output of each phase will be fed to the next phase as input. The output of last phase is the actual guesses for trying. We now describe each phase in detail along with simple examples.

4.3.1 *Personal Information Matching*

Given a password string, we first match the entire password or a substring of the password to its personal information. The detailed algorithm is similar to Algorithm 1. However, this time we also record the length of the matching segment. We replace the matched segments in password with corresponding symbols and mark the symbols with length. Unmatched segments remain unchanged. For instance, we assume Alice was born in August 16, 1988 and her password is "helloalice816!". The matching phase will replace "alice" with "N_5" and "816" with "B_3". The leftover "hello" is kept unchanged. Therefore the outcome of this phase is "$helloN_5B_3$!".

4.3.2 *Password Pre-processing*

This phase is similar to the pre-processing routine of the original PCFG; however, based on the output of personal information matching phase, the segments already matched to personal information will not be processed. For instance, the sample structure "$helloN_5B_3$!" will be updated to "$L_5N_5B_3S_1$" in this phase. Now the password is fully described by semantic symbols of Personal-PCFG, and the output in this phase provides base structures for Personal-PCFG.

4.3.3 *Guess Generation*

Similar to the original PCFG, we replace "D" and "S" symbols with actual strings learned from the training set in descending probability order. "L" symbols are replaced with words from a dictionary. Similar to PCFG [40], we output the results on the fly so we do not need to wait all possibilities for guesses are calculated and sorted. The results are kept outputting for next step. Note that we have not replaced any symbols for personal information so the guesses are still not actual guesses. We do not handle personal information in this step, since personal information for each user

Figure 3: Compare PCFG and Personal-PCFG.

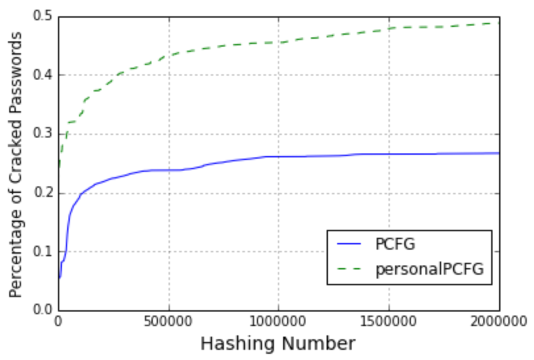
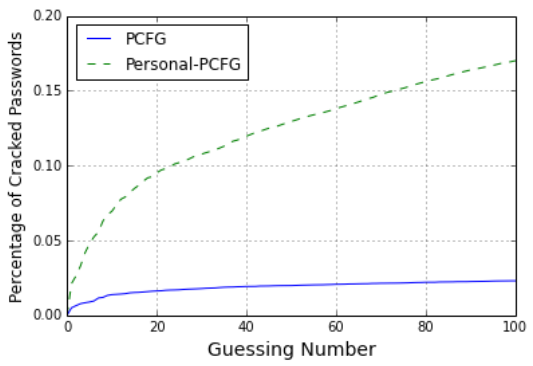


Figure 4: PCFG and Personal-PCFG – Online attacks.



is likely to be different and personal information symbols can only be substituted until the target is specific. Therefore, in this phase our base structures only generate pre-terminals, which are partial guesses that contain part of actual guesses and part of Personal-PCFG semantic symbols. For instance, the example "$L_5N_5B_3S_1$" is instantiated to "$helloN_5B_3$!" if "hello" is the first 5-symbol long string in the input dictionary and "!" has highest probability of occurring among 1 symbol special character in the training set. Note that for "L" segments, each word of the same length has the same probability. The probability of "hello" is simply $\frac{1}{N}$, in which N is the total number of words of length 5 in the input dictionary.

4.3.4 Adaptive Substitution

In the original PCFG, the output of guess generation can be applied to any target user. However, in Personal-PCFG, the guesses will be further instantiated with personal information, which are specific to only one target user. Each personal information symbol is replaced by corresponding personal information of the same length. If there are multiple candidates of the same length, all of them will be included for trial. In our example $helloN_5B_3$!, N_5 will be directly replaced by "alice". However, since B_3 has many candidate segments and any length 3 substring of "19880816" may be a candidate, the guesses include all substrings such as "helloalice198!", "helloalice988!", ..., "helloalice816!". We then try these candidate guesses one by one until we find out that the last candidate matches exactly the password of Alice. Note that on the opposite of having multiple candidate, not all personal information segments can be replaced since the same length segments may not always be available. For instance, a pre-terminal structure $helloN_6B_3$! is not suitable for Alice since her name is at most 5 symbols long. In this case no guesses from this structure should be generated for Alice.

4.4 Cracking Results

We compare the performance of Personal-PCFG and the original PCFG using the 12306 dataset, which has 131,389 users. We use half of the dataset as the training set, and the other half as the testing set. For the "L" segments, both methods need to use a dictionary, which is a critical part for password cracking. To eliminate the effect of unfair dictionary selection, we use "perfect" dictionaries in both methods. Perfect dictionaries are dictionaries we collected directly from the testing set, so that any string in the dictionary is useful and any letter segments in the passwords must appear in the dictionary. Thus, a perfect dictionary is a guarantee to find correct alpha strings efficiently. In our study, both PCFG perfect dictionary and Personal-PCFG perfect dictionary contain 15,000 to 17,000 entries.

Most previous works on password cracking use *guessing number* as a metric to compare their performance; however, we propose to use *hashing number* to evaluate Personal-PCFG. Because hashing operations are the bottleneck of password cracking attacks, it is more reasonable to compare hashing number instead of guessing number. Researchers use guessing number in their works because the guessing number is independent of password dataset size and one hashed guess can be compared to all the passwords in the dataset. However, because almost all modern password datasets are protected by the salt mechanism, each non-personalized guess still needs to be padded by the individual salt value for each account before hashing. Therefore, a dataset of N passwords requires N hashes to verify one guess. In Personal-PCFG, one guess is usually personalized to test on one specific password so that it requires only 1 hash to verify a guess.

Figure 3 shows the comparison of the original PCFG and Personal-PCFG using average hashing number. We compute the probability of a password to be cracked given various hashing numbers. Figure 3 implies that generally Personal-PCFG works substantially better than original PCFG. With a moderate size of 500,000 hashes Personal-PCFG achieves a similar success rate that can be achieved with over 200 million hashes by original PCFG. That is to say, Personal-PCFG can crack password much faster than PCFG does. Moreover, Personal-PCFG is able to cover a larger password space than PCFG. This is because personal information provides rich personalized strings that may not appear in the dictionaries or training set. It is not surprising that both PCFG and Personal-PCFG can crack passwords quickly when they just start, since both methods always try high probability guesses first. However, the lines become relatively flat when the guessing number is growing large.

Personal-PCFG not only improve the cracking efficiency in offline attacks, but also increase the guessing success opportunity in online attacks. Online attacks are only able to try a small number of guesses in a certain time period due to the system constrains on the login attempts. Therefore we limit the guesses to be at most 100 times. We present the results in Figure 4, from which it can seen that Personal-PCFG is able to crack 309% to 634% more passwords than

Figure 5: Representative Points – Online attacks.

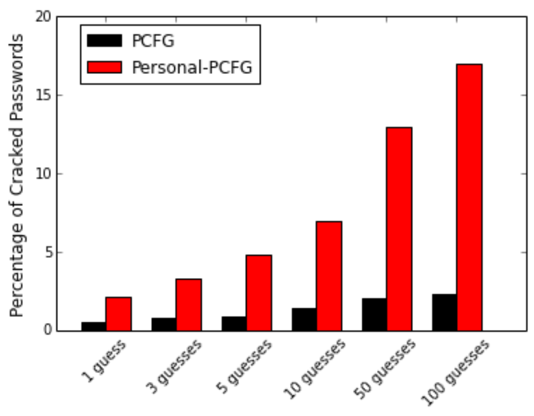
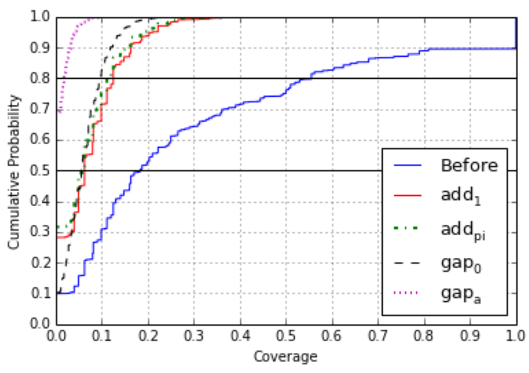


Figure 6: Coverage distribution.



original PCFG. We then show several representative guessing numbers in Figure 5. For a typical system that allows 5 attempts to input the correct passwords, Personal-PCFG is able to crack 4.8% passwords within only 5 guesses. Meanwhile the percentage is just 0.9% for original PCFG, and it takes around 2000 more guesses for PCFG to reach a success rate of 4.8%. Thus, Personal-PCFG is more efficient to crack the correct passwords within only a small number of guesses.

We conclude that Personal-PCFG substantially outperforms PCFG in both online and offline attacks due to the integration of the personal information into password guessing. On the other hand, the extra requirement of Personal-PCFG on personal information can be satisfied by knowing the victim personally or searching on social networks sites (SNS).

5. PASSWORD PROTECTION

In almost all systems, users are able to choose and update their passwords. However, they may sacrifice password security for usability since a long and random secure password is less memorable. As user passwords are easier to be compromised when their personal information is available to the attacker, we investigate how users can protect their passwords against such attacks.

To increase the password security while retaining good memorability, we introduce the *Distortion Function*, which performs a transformation on user passwords. A distortion function converts user passwords to more secure strings by either breaking password semantics or increasing password entropy. Therefore, the user only needs to remember the original password and apply a simple function on it to create a stronger password. This distortion function can be chosen by users so it could be either linear or non-linear.

We conducted a proof-of-concept study to show the effectiveness of distortion function on password security. In our study, we propose two types of distortion functions. The first type of distortion function maps each password character to another character. For instance, add_1 function simply replaces each letter with the one 1 letter later in the alphabet and replaces single digital number i with $(i+1)mod10$. It is similar to the Caesar Cipher [21]. In another example, add_pi is a non-linear function that replaces characters by the position specified by π, which is $314159\ldots$. It replaces the first letter with the one 3 letters later and the second letter with the one 1 letter later, etc. The second type of distortion function adds an extra fixed character between any pair of characters in passwords. The length of passwords become twice of the original password minus 1 when the length is larger than 1. We call this distortion function gap_x, in which "x" represents the extra symbol. For example, when $x =$ "a", Alice's password "alice816" will be extended to "aalaiacaea8a1a6" after the distortion function.

The distortion function must be simple enough for users to remember and generate the passwords. We apply a number of distortion functions on each of the password in 12306 dataset individually and calculate the Coverage for the converted passwords. As Figure 6 shows, the distortion functions are effective on increasing password security by greatly reducing the correlation between user passwords and personal information. Moreover, we notice that the impacts of various distortion functions are also different. For example, $add_!$ performs the best (Coverage is 0 for all users so it is not shown in Figure 6) since users rarely have special characters in their personal information. Surprisingly, the non-linear add_pi function does not produce a better result than other linear functions such as add_1 because digits preferred by Chinese users are more likely to have coincidence wrong matches due to its low entropy.

We conclude that distortion functions can mitigate the problem of including personal information in user passwords without sacrificing password usability. Moreover, distortion function is also a cure for semantics-aware password cracking methods [37], which leverages semantic patterns in passwords to crack other passwords. After applying a distortion function, the semantic pattern is no longer available. Besides, distortion function is effective against PCFG [40], since it generates unrecognizable letter segments, which are not likely to be covered in commonly-used password dictionaries.

The main weakness of distortion function is that it has to be kept secret. If an attacker knows the distortion function used by a user, cracking the user's password is as easy as cracking a plain password. However, as users are able to choose their own distortion functions, due to the large pool of potential distortion functions, it is hard for attackers to guess which one is used or test all popular distortion functions. Similar to the functionality of adding salt to password, distortion functions make it harder to perform off-line brute-force attack on a large password dataset. This is because distortion functions make the passwords in the dataset sparser and more unpredictable.

6. RELATED WORK

Researchers have done brilliant works on measuring real life passwords. In one of the earliest work [30], Morris and Thompson found that passwords are quite simple and thus are vulnerable to dictionary attacks. Malone et al. [27] studied the distribution of passwords on several large leaked datasets and found that user passwords fit Zipf distribution well. Gaw and Felton [15] showed how users manage their passwords. Mazurek et al. [29] measured 25,000 passwords from a university and revealed correlation between demographic or other factors, such as gender and field of study. Bonneau [2] studied language effect on user passwords from over 70 million passwords. Through measuring the guessability of 4-digit PINs on over 1,100 banking customers [4], Bonneau et al. found that birthday appears extensively in 4-digit PINs. Li et al. [26] conducted a large-scale measurement study on Chinese passwords, in which over 100 million real life passwords are studied and differences between Chinese and other languages passwords are presented.

There are several works investigating specific aspects of passwords. Yan et al. [41] and Kuo et al. [25] investigated the mnemonic based passwords. Veras et al. [38] showed the importance of date in passwords. Das et al. [11] studied how users mangle one password for different sites. Schweitzer et al. [36] studied the keyboard pattern in passwords. Beside password itself, researches have been done on human habit and psychology towards password security [14, 18].

It has been shown that Shannon entropy has lots of troubles accurately describing the security of passwords [7, 22, 34, 39]. Researchers developed a number of metrics to measure passwords. Massey [28] proposed guessing entropy, which shows the expected number of guesses to make a correct guess. Several other most commonly used metrics include marginal guesswork μ_α [34], which measures the number of expected guess to succeed with probability α, and marginal success rate λ_β [5], which is the probability of succeed in β guesses.

Study of password cracking method has been discussed for over three decades. Attackers usually tried to recover passwords from a hashed password database. While reverse hashing function is infeasible, early works found that passwords are vulnerable to dictionary attacks [30]. Time-memory Trade-off in passwords [17] made dictionary attacks much more efficient. Rainbow table [32] reduces table number in [17] using multiple reduction functions. However, recent years as the password policy becomes strict, simple dictionary passwords are less common. Narayanan and Shmatikov [31] used Markov model to generate guesses based on that passwords need to be phonetically similar to users' native languages. In 2009, Weir et al. [40] leveraged Probabilistic Context-Free Grammars (PCFG) to crack passwords. Veras et al. [37] tried to use semantic patterns in passwords. Besides, while attacking a hashed password database remains main attacking scenarios, there are other attacks on different scenarios, such as video eavesdropping on passwords [1].

OMEN+ [9] improves Markov Models [31] to crack passwords. It includes some simple experiments to prove usefulness of personal information in password cracking. However, since their experiments are performed on only 3,140 users, their results are insufficient and unreliable. Moreover, since the personal information is simply divided into 3-grams and the probabilities of these 3-grams are increased a little in the attack dictionary, OMEN+ does not treat personal information as "personally" and break the internal continuation of personal information. Therefore, we believe the effect of OMEN+ [9] is roughly equivalent to adding common names, locations, etc. to training data. Besides, it is naturally improper to use personal information of individual user to generate guesses for all users.

There have been works on protecting passwords by enforcing users to select more secure passwords, among which password strength meters seem to be one effective method. Castelluccia et al [10] proposed to use Markov Model as in [31] to measure the security of user passwords. Meanwhile commercial password meters adopted by popular websites are proved inconsistent [13]. There are works focusing on providing Feedback to users using trained leaked passwords or dictionaries [39, 23]. Several works discussed adding security questions on top of passwords [33, 35, 6]. Though some graphics-based [12, 20] or biometrics-based [19] authentication methods have been proposed, text-based passwords are expected to remain dominating [3].

7. CONCLUSION

In this work, we conduct a comprehensive quantitative study on how user personal information resides in human-chosen passwords. To the best of our knowledge, we are the first to systematically analyze personal information in passwords. We have some interesting and quantitative discovery such as 3.42% of the users in the 12306 dataset use their birthday as passwords, and male users are more likely to include their name in passwords than female users. We then introduce a new metric, Coverage, to accurately quantify the correlation between personal information and a password. Our coverage-based quantification results further confirm our disclosure on the serious involvement of personal information in password creation, which makes a user password more vulnerable to a targeted password cracking. We develop Personal-PCFG based on PCFG but consider more semantic symbols for cracking a password. Personal-PCFG generates personalized password guesses by integrating user personal information in the guesses. Our experimental results demonstrate that Personal-PCFG is significantly faster than PCFG in password cracking and eases the feasibility of mounting online attacks. Finally, we propose to use distortion functions to protect weak passwords that include personal information. Through a proof-of-concept study, we validate that distortion functions are effective to defend against personal-information-related and semantics-aware attacks.

8. REFERENCES

[1] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from video. In *IEEE Security & Privacy*, 2008.

[2] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Security & Privacy*, 2012.

[3] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *IEEE Security & Privacy*, 2012.

[4] J. Bonneau, S. Preibusch, and R. Anderson. A birthday present every eleven wallets? the security of customer-chosen banking pins. In *Financial Cryptography and Data Security*. Springer, 2012.

[5] S. Boztas. Entropies, guessing, and cryptography. *Department of Mathematics, Royal Melbourne Institute of Technology, Tech. Rep*, 1999.

[6] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung. Fourth-factor authentication: somebody you know. In *ACM CCS*, 2006.

[7] C. Cachin. *Entropy measures and unconditional security in cryptography*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1997.

[8] P. Cao, H. Li, K. Nahrstedt, Z. Kalbarczyk, R. Iyer, and A. J. Slagell. Personalized password guessing: a new security threat. In *ACM Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*, 2014.

[9] C. Castelluccia, A. Chaabane, M. Dürmuth, and D. Perito. When privacy meets security: Leveraging personal information for password cracking. *arXiv preprint arXiv:1304.6584*, 2013.

[10] C. Castelluccia, M. Dürmuth, and D. Perito. Adaptive password-strength meters from markov models. In *NDSS*, 2012.

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

[12] D. Davis, F. Monrose, and M. K. Reiter. On user choice in graphical password schemes. In *USENIX Security*, 2004.

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

[14] D. Florencio and C. Herley. A large-scale study of web password habits. In *ACM WWW*, 2007.

[15] S. Gaw and E. W. Felten. Password management strategies for online accounts. In *ACM SOUPS*, 2006.

[16] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *ACM WPES*, 2005.

[17] M. E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 1980.

[18] A. E. Howe, I. Ray, M. Roberts, M. Urbanska, and Z. Byrne. The psychology of security for the home computer user. In *IEEE Security & Privacy*, 2012.

[19] A. K. Jain, A. Ross, and S. Pankanti. Biometrics: a tool for information security. *IEEE Transactions on Information Forensics and Security*, 2006.

[20] I. Jermyn, A. J. Mayer, F. Monrose, M. K. Reiter, A. D. Rubin, et al. The design and analysis of graphical passwords. In *Usenix Security*, 1999.

[21] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice-Hall, Inc., 1995.

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

[23] S. Komanduri, R. Shay, L. F. Cranor, C. Herley, and S. Schechter. Telepathwords: Preventing weak passwords by reading users' minds. In *USENIX Security*, 2014.

[24] B. Krishnamurthy and C. E. Wills. On the leakage of personally identifiable information via online social networks. In *ACM COSN*, 2009.

[25] C. Kuo, S. Romanosky, and L. F. Cranor. Human selection of mnemonic phrase-based passwords. In *ACM SOUPS*, 2006.

[26] Z. Li, W. Han, and W. Xu. A large-scale empirical analysis of chinese web passwords. In *Proc. USENIX Security*, 2014.

[27] D. Malone and K. Maher. Investigating the distribution of password choices. In *ACM WWW*, 2012.

[28] J. L. Massey. Guessing and entropy. In *IEEE International Symposium on Information Theory*, 1994.

[29] 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. In *ACM CCS*, 2013.

[30] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 1979.

[31] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *ACM CCS*, 2005.

[32] P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology-CRYPTO 2003*. 2003.

[33] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *ACM CCS*, 2002.

[34] J. O. Pliam. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Progress in Cryptology-INDOCRYPT*. 2000.

[35] S. Schechter, A. B. Brush, and S. Egelman. It's no secret. measuring the security and reliability of authentication via "secret" questions. In *IEEE Security & Privacy*, 2009.

[36] D. Schweitzer, J. Boleng, C. Hughes, and L. Murphy. Visualizing keyboard pattern passwords. In *IEEE VizSec*, 2009.

[37] R. Veras, C. Collins, and J. Thorpe. On the semantic patterns of passwords and their security impact. In *NDSS*, 2014.

[38] R. Veras, J. Thorpe, and C. Collins. Visualizing semantics in passwords: The role of dates. In *IEEE VizSec*, 2012.

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

[40] M. Weir, S. Aggarwal, B. De Medeiros, and B. Glodek. Password cracking using probabilistic context-free grammars. In *IEEE Security & Privacy*, 2009.

[41] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security & Privacy Magazine*, 2004.