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 we introduce a new metric called coverage to quantify the correlation between passwords and personal information. Then, exploiting the potential of cracking passwords based on our analysis, we develop a semantics-richer Probabilistic Context-Free Grammars method called Personal-PCFG to crack passwords. Through offlline and online attack scenarios, we demonstrate that Personal-PCFG cracks passwords much faster than the state-of-art techniques and makes online attacks much easier to succeed. To defend against such semantics-aware attacks, we propose to use 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 sacraficing usability.

Categories and Subject Descriptors

[**Security and privacy**]: *Human and societal aspects of security and privacy*; [**General and reference**]: *Metrics*

General Terms

Security

Keywords

passwords, password cracking, data processing, password protection

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WOODSTOCK '97 El Paso, Texas USA

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

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 random strings [2, 25, 29, 37, 40]. For example, "secret" is more likely a human-chosen password than "ziorqpe". In other words, human users often choose weak passwords simply because they are easy to remember. As a result, most passwords are within only a small portion of the large password space, vulnerable to brute-force and dictionary attacks.

To increase password security, a very common way is to restrict password creation policies by requiring non-alphabets in passwords. The bad news is that these policies do not keep passwords from vulnerable because users tend to simply mangle their passwords to fulfill the restriction [38], making the security improvement very limited. Nowadays most websites also have password strength meters enabled to provide direct feedbacks to users. However, these meters are proved to be ad-hoc and inconsistent [13]. To better assess the strength of passwords, we need to understand how users construct their passwords. If an attacker knows exactly how users construct their passwords, guessing their passwords will become much easier. On the other hand, if a user knows how other users construct their passwords, the user can easily improve his/her password strength by avoid using these password construction methods.

Toward this end, researchers have done much 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 [16][28]. Languages users speak are also vital since users tend to use their own languages when constructing passwords [2]. Besides, passwords are mostly phonetically memorable [29] even they are not simply dicionary words. It is also indicated that users use keyboard strings such as "qwerty" and "qweasdzxc", trivial strings such as "password", "123456", and date strings such as "19951225" in their passwords [24][34][37]. However, most studies are showing superficial password patterns. The actual composition of passwords are still mysterious to people. Fortunately, an enlightening work studied how users generate their passwords by learning the semantic patterns in passwords [36]. We would like to take another perspective – personal information, to study password semantics.

Based on our analysis and quantification of personal information, we conduct subsequent works on password cracking and protection.

Our contribution is 4-fold. First, we measure the importance of personal information in passwords and present interesting quantified results. We show most popular password structures with personal information notion embedded. We also found that male and female behave differently when putting personal information in their passwords. Second, we quantify the correlation between user password and personal information by developing a metric – Coverage. Coverage is useful in measuring password strength, which can potentially be used in password meters. Third, we introduce Personal-PCFG, an evolution of PCFG method by M. Weir[39] by adding personal information symbols in password structures. Personal-PCFG is able to crack password much faster than PCFG. It also makes online attack more feasible by drastically increasing guessing successful rate with a small amount of guesses. Forth, we discuss how to defend such semantics-aware attacks like Personal-PCFG or [36] by applying a simple distortion function on passwords.

This paper is organized as follows. Section 2 studies how personal information resides in user passwords and shows the gender difference in passwords. Section 3 introduces the Correlation descriptive metric – Coverage. In Section 4 we present Personal-PCFG and show cracking results compared to original PCFG method. In Section 5 we discussed defense of our attacks by applying distortion functions in passwords.

2. PERSONAL INFO IN PASSWORDS

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

2.1 12306 Dataset

Many password datasets have been exposed to the public in recent years, and there are password measurement or password cracking works based on these datasets [2, 24]. These leaked datasets usually contain several thousands to millions of real passwords. In this paper, a dataset called 12306 is used to illustrate how personal information is involved in password creation.

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 [35]. We call this dataset 12306 because all passwords are from a website www.12306.cn. The website is the official website of online railway ticket booking system for Chinese users. There is no data 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.

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%

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 plain text passwords, the dataset also includes several types of user personal information, such as a user's name and ID number. This is because the website requires a real ID number to register and people need to provide real personal information to book a ticket.

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 to register the web system and count 0.2% of the whole dataset. After cleansing our dataset, it contains 131,389 passwords for analysis Note that websites may have different password creation policy. With a strict password policy, users may apply mangling rules (For example, $abc->@bc$ or $abc1$) to their passwords to fulfill the policy requirement [38]. As the 12306 website has changed its password policy after the password leakage, we do not know exactly the password policy at the time the dataset was leaked. However, from the dataset, we infer that the password policy is quite simple — all passwords need to be no shorter than 6 symbols. There is no restriction on what type of symbols are used. Therefore, users are not forced to apply any mangling 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. It is evident that the dominating passwords are trivial passwords (123456, a123456, etc), keyboard passwords (1qaz2wsx and 1q2w3e4r), and "I love you" type passwords. Both "5201314" and "woaini1314" means "I love you forever" in Chinese. The most commonly used Chinese passwords are similar to previous studies [24]. 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 [24]. We believe that the passowrd 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 symbil accounts because real ID number is needed.

Then we show the basic structure of passwords. The most popular password structures are shown in Table 2. Like previous studied [24], 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, in which at most 2 or 3 letters

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.

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 security more when creating passwords for important systems.

2.2 Personal Information

As we mentioned before, the 12306 dataset not only contains user passwords, but it 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 that the government issued ID number is an 18-digit unique number, which actually 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 because birthday is very important personal information in passowrd creation. Therefore, we finally have six types 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 measurement. The matched personal information is represented with corresponding tags, [Name] and [Birthday] in this case; for those segments that are not matched, we still use "D", "L", and "S" to describe the types of characters.

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. Using this representation, we apply the matching process to the whole 12306 dataset to see how these personal information tags appear in password structures.

2.2.2 Matching Method

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

Algorithm 1 Match personal information with password.

```
1: procedure MATCH(pwd,infolist)
2:     newform ← empty_string
3:     if l thenen(pwd) == 0
4:         return empty_string
5:     end if
6:     substring ← get_all_substring(pwd)
7:     reverse_length_sort(substring)
8:     for eachstring ∈ substring do
9:         if len(eachstring) ≥ 2 then
10:            if matchbd(eachstring,infolist) then
11:                tag ← "[BD]"
12:                leftover ← pwd.split(eachstring)
13:                break
14:            end if
15:            ...
16:            if matchID(eachstring,infolist) then
17:                tag ← "[ID]"
18:                leftover ← pwd.split(eachstring)
19:                break
20:            end if
21:        else
22:            break
23:        end if
24:    end for
25:    if leftover.size() ≥ 2 then
26:        for i ← 0 to leftover.size()-2 do
27:            newform ← MATCH(leftover[i],infolist) +
   tag
28:        end for
29:        newform ← MATCH(leftover[leftover.size()−
   1])+newform
30:    else
31:        newform ← seg(pwd)
32:    end if
33:    return newform
34: end procedure
```

Note that in Algorithm 1 we do not show the specific matching algorithm for each type of the personal information (line 10 and line 16) to keep it clean and simple. We describe the matching methods as follows.

Table 3: Most Frequent Password Structures.

Rank	Structure	Amount	Percentage
1	D7	7122	5.420%
2	[ACCT]	6820	5.190%
3	[NAME][BD]	5410	4.117%
4	D6	4886	3.718%
5	[BD]	4470	3.402%
6	D8	4245	3.230%
7	[EMAIL]	3807	2.897%
8	L1D7	3296	2.508%
9	[NAME]D7	2949	2.244%
10	[NAME]D3	2363	1.798%

First we make sure that the length of a password segment is at least 2 for matching. We try to match segments with length 2 or more to each kind of the personal information. For name, we convert Chinese names into Pinyin form, which is alphabetic representation of Chinese. Then we compare password segments to 10 possible permsutations 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. We list all the 10 permutations in the Appendices. For birthday, we list 17 possible permutations and compare a password segments with each of the permutation. If the segment is the same as any permutation, we consider a match is found. All the birthday permutations are also listed in the Appendices. For account name, cellphone number, and ID number, we further restrain the length of a segment to be at least 4 to avoid coincidence. We believe a match of at least length 4 is very likely to be an actual match. If the segment is a substring of any of the 3 personal information, we regard it a match to the corresponding personal information. Note that for some password segment, it may match to multiple types of personal information. In this case, each type that matched is counted once.

2.2.3 Matching Result

After applying Algorithm 1 to the 12306 dataset, we found that 70,892 out of 131,389 (54.0%) of the passwords contain at least one of the six types of personal information. Apparently, personal information is an essential part of user passwords and most users include certain personal information in their passwords. We believe that the ratio could be higher if we have more personal information at hand. However, this percentage has served its purpose properly. We present the top 10 password structures in Table 3 and most commonly used personal information in Table 4. As we have mentioned before, a password segment may match to multiple types of personal information and we count each matched type. Therefore, with 131,389 passwords we obtain 143373 password structures. Based on Tables 3 and 4, we have the following observations

1. Overall personal information is a vital part of passwords. 3 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, name, and account name are most popular personal information in user passwords with 24.63%, 22.56%, and 16.22% occurrence rate accordingly. A moderate percentage (8.4%) of users put email in their passwords. On the other hand, Only few people include their cellphone and ID number in their passwords.

3. Set aside personal information, 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. An interesting observation is that although account name has merely half percentage as birthday and name information, the structure [ACCT] ranks highest among all structures that contain personal information. The reason behind may be that users tend to directly use their account names as their passwords instead of using them as part of their passwords.

Table 4: Most Popular Personal Information.

Rank	Information Type	Amount	Percentage
1	[BD]	32373	24.63%
2	[NAME]	29646	22.56%
3	[ACCT]	21324	16.22%
4	[EMAIL]	11020	8.387%
5	[ID]	1210	0.920%
6	[CELL]	634	0.482%

2.2.4 Gender difference

Beside treating the dataset as a whole, it is also interesting to study the difference of password structures between males and females. Although the dataset does not have a gender column, user ID number actually contains gender information (The second last digit in ID number represents gender). However, we found that the dataset is biased in gender, with 9,856 females and 121,533 males in it. To balance the number, we randomly select 9,856 males from the male pool and compare them with females.

The average length of passwords for males and females are 8.41 and 8.51, which are quite similar. It shows that males and females do not differ much in the length of their passwords. We then apply the matching method to each of the 2 genders. We found that 54.9% of male passwords contain personal information while only 44.6% of female passwords contain personal information. We list the top 10 structures for each gender in Table 5 and personal information usage in Table 6. From the results we have following observations:

1. From the percentages of passwords containing personal information (54.9% for males and 44.6% for females) and the fact that each structure that contained personal information in the top 10 structures has higher percentage for male users, we can easily imply that male users are more likely to put personal information in their passwords.

2. From Table 6 we can see the percentage of each type of personal information in the passwords. Interestingly males and females are very different in the usage of name information. Males use their names as frequent as their birthday (23.43% passwords of males contain their names). Meanwhile only 13.03% passwords of females contain their names. We also notice that the name usage mostly contribute the 10% difference in personal information usage between males and females.

Table 5: Most Frequent Structures in Different Gender.

Rank	Male		Female	
	Structure	Percentage	Structure	Percentage
1	D_7	5.752%	D_6	4.890%
2	[ACCT]	5.418%	D_7	4.220%
3	[NAME][BD]	4.190%	[ACCT]	4.210%
4	[BD]	3.591%	D_8	3.256%
5	D_6	3.530%	[EMAIL]	2.678%
6	[EMAIL]	2.962%	[NAME][BD]	2.607%
7	D_8	2.861%	[BD]	2.221%
8	L_1D_7	2.851%	L_2D_6	2.079%
9	[NAME]D_7	2.353%	L_2D_7	1.704%
10	[NAME]D_6	1.978%	L_1D_7	1.684%

Table 6: Most Frequent Personal Information in Different Gender.

Rank	Male		Female	
	Information Type	Percentage	Information Type	Percentage
1	[BD]	25.02%	[BD]	21.14%
2	[NAME]	23.43%	[ACCT]	15.35%
3	[ACCT]	16.27%	[NAME]	13.02%
4	[EMAIL]	7.96%	[EMAIL]	8.81%
5	[ID]	1.02%	[ID]	0.45%
6	[CELL]	0.30%	[CELL]	0.41%

3. Set aside personal information, the top 10 structures count 33.62% of males passwords but only 28.31% of female passwords. Besides, the number of total password structures of females is 1,194, which is 8.7% more than the number of males. We can see passwords of males are more congregated, and therefore tend to be more predictable.

In conclusion, passwords of males are generally composed of more personal information, especially names of the users. In addition, the password sparsity for males is lower. Our analysis indicates that passwords of males maybe more vulnerable from the perspective of using personal information in passwords.

2.3 Service Information

Cao et al [8] proposed an interesting idea that uses service information to crack user passwords. However, the idea has not been carried out with an experiment. It draws our interest because we have shown the importance of personal information in user password and naturally we are also interested in how important the service information can be. By service information we mean the information of service provider. For example, the famous "Rockyou" dataset is leaked from a website *www.rockyou.com*, the service information could be "rockyou". In our case, the service information may be "12306". It is reasonable for a user to add service information to their passwords to keep his/her passwords different in each site. This approach balances password security and memorability. Therefore we wish to verify whether service information is as important as personal information. As 12306 is just a medium-sized Chinese dataset, the result may not be very representative. Therefore, we would like to verify more datasets, including Tianya and Rockyou datasets, etc. in each of the datasets, we try to search the service information in the passwords. The result is shown in Table 7.

[1]We found 45,574 passwords in Tianya dataset is "111222tiany". It does not make much sense for so many user using such same password so we doubt they are mostly sybil accounts. Therefore the duplications are removed in our analysis

Table 7: Service Information in Passwords.

Dataset	Password Amount	Service info Amount	Percentage
Rockyou	14,344,391	44,025	0.3%
Tianya	26,832,592	29,430 [1]	0.11%
PHPBB	184,389	2,209	1.2%
12306	131,389	490	0.4%
MySpace	37144	72	0.2%

As shown in Table 7, we can see indeed some users are using service information in their passwords. Our results indicate that all datasets examined contain 0.11% to 1.2% passwords that relate to service information. However, the portion is quite small. Such small percentage indicates that though adding service information in user passwords may maintain good memorability and security, currently few users are using such method to construct their passwords. Therefore, we concluded that only few users include service information in their passwords. the method proposed by Cao et al that uses service information to crack passwords may not bring much improvement over state-of-art technique.

2.4 Ethical Consideration

We do realize that studying leaked datasets involves much ethical concern. We claim that we only use the datasets for researching purpose. All data are carefully stored and used. We will not expose any user personal information or password or use these information in any other way except for research use.

3. CORRELATION QUANTIFICATION

In Section 2.2.3 we have shown password structure distributions and the percentages of passwords containing each type of personal information. These results helped us to draw interesting observations. However, until this point we do not have a clear idea on how much personal information correlates to passwords in a quantitative way. Although we have shown useful numbers between each type of personal information and passwords (such as birthday appears in 24.63% passwords), these numbers are not ideal to describe the passwords because they are too general to accurately describe the correlations. In [9] C. Castelluccia tried to use Longest Common Sequence and a modified JS to measure the correlations. However, their results end up having 2 metrics for each type of personal information. Besides, their metrics fail to differentiate fragmented and continued personal information. We seek a comprehensive method to quantify the correlation between passwords and all types of available personal information. Thereby we create a novel metric – Coverage.

3.1 Coverage

Coverage is a metric ranging from 0 to 1. Larger Coverage implies stronger correlations. It can be used on every individual user in the dataset. Average Coverage also reflects general correlation of the entire dataset. We will show the algorithm to compute coverage and give a detailed example in the following section to illustrate how Coverage works.

Besides, we elaborate the features of Coverage which make it a useful metric.

3.1.1 Computation Method

To compute Coverage, we take password and personal information in terms of strings as input. And we use a sliding window approach as in data transmission protocols to conduct 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 matched 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 no match exists. At this point we reset the window size to be the initial size and slide the window to the last password symbol that is covered by the window. In the mean time of sliding, we maintain an array of same length as the password. This array is called tag array. Tag array is used to record the length of each matched password segment. For example, a tag array may be [4,4,4,4,0,0,2,2]. This array implies that the first 4 password symbols match certain type of personal information. The following 2 symbols have no match and the last 2 symbols again match certain type of personal information. The personal information types that first 4 symbols and last 2 symbols matched to 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 Coverage metric. Coverage is computed as the sum of squares of matched password segment length divided by the square of password length. Mathematically

$$CVG = \sum_{i=1}^{n} (\frac{l_n^2}{L^2}) \quad (1)$$

in which n denotes the number of matched password segment, l_n denotes the length of corresponding matched password segment, and L is the length of the password. We show the algorithm that is used to obtain Coverage in Algorithm 2. Note that differently from the matching algorithm introduced in Section 2.2.2, we aim to make the matching for Coverage simpler in order to keep it universally applicable. Namely we try to restrain ad-hoc processing on each type of personal information minimum. Toward this end, we do not include various permutations of birthday. Cases like DD+MM+YYYY are ignored and we keep the only case of YYYY+MM+DD, which conforms to Chinese conventions. However, we keep full name and name initials to match names because name initials are very common in passwords. We believe little ad-hoc processing is still acceptable in computation of Coverage. As a conclusion, only names have 2 permutations, other personal information keep their original format. A match is found if an at least 2-symbol long password segment is a substring of any of the personal information.

To better illustrate how Coverage is computed, we show a simple example in Figure 1. We take a user named Alice, who was born in Aug. 16, 1988 as an example. We assume her password happens to be "alice816". If we apply the algorithm in Section 2.2.3, the structure of this password will be [NAME][BD]. Apparently her password is quite related to her personal information. To quantify this relation, we follow Algorithm 2 to compute Coverage for her and each step is shown in Figure 1. (a) The password is prepared for a window to slide through. Note the personal information include "alice" as name and "19880816" as birthday. The tag array is initialized as [0 0 0 0 0 0 0 0]. (b) Window size is initialed to 2 so the first 2 symbols in the password are covered. "al" is a substring of Alice's name, so a match is found. Therefore we extend the window size by 1 and the tag array is updated as [2 2 0 0 0 0 0 0]. (c)-(e) Window keeps growing because matches are continuously found. The tag array also keeps updating. (f) The window now covers "alice8", which is not a substring of "alice" or "19880816". Therefore, window size is reset to 2 and the window slides to the position of last symbol of the window (position of "8"). Tag array remains unchanged. (g) The window of size 2 now covers "81", which is a substring of birthday, again we extend the window by 1 and update tag array to [5 5 5 5 5 2 2 0] (h) After window grows, "816" is also found a match. Tag array is updated to [5 5 5 5 5 3 3 3]. The window does not grow or slide any more because it has reached the end of the password. (i) All symbols have settled. Tag array can now be used to compute Coverage. Based on Equation 1, the coverage is computed as $CVG = \sum_{i=1}^{2} \frac{l_n^2}{L^2} = \frac{5^2+3^2}{8^2} = 0.52$.

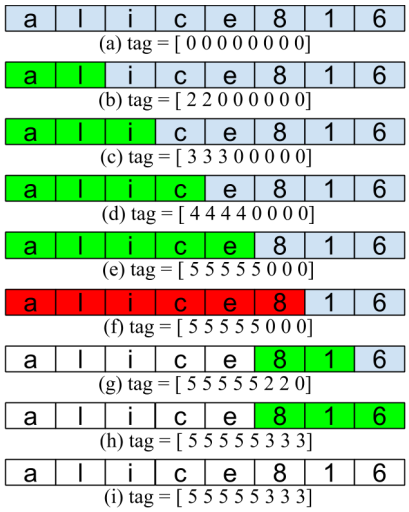
Algorithm 2 Compute Coverage.

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

In order to show that Coverage is a reasonable and useful metric, we present 4 features that Coverage carries and why we need these features.

1. By the nature of Equation 1, Coverage ranges from 0 to 1, in which 0 means no personal information is matched to any password segment and 1 means the entire password is matched to only one type of personal information. The easy-to-handle format (a float number) and range make it convenient to be used in various scenarios.

Figure 1: Coverage - An Example.



Grey boxes hold unvisited password symbols. green 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).

2. Coverage can be universally applicable for its simplicity. Any kind of personal information can be formatted as a string, check whether or not a password segment is a substring of the string is straightforward. Besides, a few variations of personal information strings may be applied to enhance the completeness and accuracy (like names in our application). Therefore Coverage can consistently serve its purpose given different sets of personal information.

3. Coverage can reflect the length of personal information. We need this feature because longer matched password segment naturally indicates stronger correlation. From Equation **??**, for a matched password segment, longer the segment implies larger numerator, and therefore higher Coverage when the length of the password is unchanged .It is obvious Coverage measures the correlation strength in this sense.

4. For personal information segments of same length, Coverage stresses the Continuation of matching. We argue that continuous match is stronger than fragmented match. That is to say, for a given password of length L, a matched segment of length l ($l < L$) has stronger correlation to personal information than 2 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. We desire this feature because shorter segments are usually harder to guess and may involve wrong match (coincidence). Since it is difficult to differentiate a real match and a coincidence match, we would like to minimize the effect of wrong matches by taking squares of the matched segments to compute the Coverage.

3.2 Results

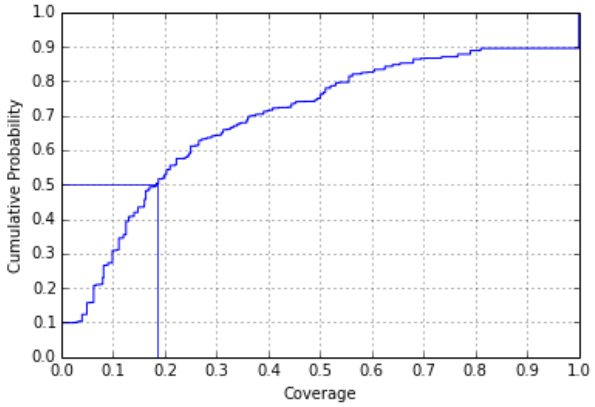
We apply Algorithm 2 on every user of 12306 dataset. The result is shown as a cumulative distribution graph in Figure 2. Before looking at the table, we give several examples to show implication of a roughly 0.2 Coverage. If we have a 10-symbol long passwords. One matched segment of length 5 will yield 0.25 Coverage. Two matched segments of length 3 (which means 6 symbols are matched to personal information) yield 0.18 Coverage. And 5 matched segments of length 2 (which means all symbols are matched, though fragmentarily) yield 0.2 Coverage. Apparently Coverage of 0.2 indicates a fairly high correlation between personal information and passwords. With such knowledge, we have following observations from Figure 2

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

2. 9.9% users have 0 Coverage. 0 Coverage indicates there is no possible correlation between the users' passwords and their personal information.

3. 10.5% users have Coverage of 1, which means that 11.9% passwords are perfectly matched to exactly one type of personal information. However, Only 13.4% users have Coverage higher than 0.5 and lower than 1.0.

Figure 2: Coverage distribution - 12306.



The average Coverage for 12306 dataset is 0.309. We also compute the average Coverages for male and female groups that are used in Section 2.2.4, in which we concluded that male users are more likely to include personal information in their passwords. It turns out that the average Coverage for male group is 0.314 and the average Coverage for female group is 0.269. Our result indicates that such correlation for male users is higher. It complies with our previous conclusion. Conversely, it also shows that Coverage works well to determine the correlation between passwords and personal information.

3.3 Coverage Usage

Coverage can be used in many scenarios. Firstly, it can be used as a general metric to describe a dataset. Coverage directly reflects the correlation between password and available personal information. Investigating how users from different systems behave regarding to constructing their passwords using personal information would be an interesting future work.

Apparently, Coverage would be very useful in password strength meters. Strength meters are reported as mostly ad-hoc [13]. Most meters dumbly give score based on password structures and length, or blacklist commonly used passwords (such as notorious "password"). There are also meters that do simple social profile analysis such as passwords cannot contain the users' names or passwords cannot be the same as account names. However, these simple analysis mechanisms can be easily fooled around by mangling the password a little. In cases of password strength meters, Coverage is able to cover the unnoticed corner of personal information. Consider the routine for creating an account. Users usually fill in some profile information such as name, birthday, email address, etc and create a password. At this moment password meters will offer feedback of password strength normally in bar or score forms. Since personal information is available, Coverage matches the scenario perfectly. In addition, it is straightforward to set Coverage as part of strength measurement (only a few lines of Javascript will do). Apparently, higher Coverage indicates vulnerable passwords. Systems can even ban passwords when they surpass a preset Coverage threshold. Furthermore, users cannot easily escape Coverage measurement by simple mangling, thus they are forced to select more secure passwords.

Coverage can also be added to existing tools to enhance their capabilities. There are several Markov Model based tools that predict the next character when the user creates a password [22][38]. Users would be surprised to find that the next character in their mind matches exactly the tool output. They then switch to a more unpredictable passwords. Coverage is able to work in these tools as an enhancement – When Coverage is rather high, it is easy to predict what the next character of user input will be.

4. PERSONAL-PCFG, AN INDIVIDUAL ORIENTED PASSWORD CRACKER

We have presented abundant findings regarding user personal information and their passwords, as well as a metric to quantify their correlation. Naturally we wonder how these findings are useful. An obvious answer is that given there is significant correlation between user password and personal information, much potential can be seen to crack passwords faster. To this end, we create Personal-PCFG, which relies on the idea of PCFG approach created by M. Weir [39]. However, unlike original PCFG method, Personal-PCFG is an individual-oriented password cracker, which generates personalized guesses towards each user. Note that we do not claim to be the first trying to leverage personal information to crack passwords. OMEN+ [9], which improves Markov Models [29] to crack passwords, has done simple experiment to prove usefulness of personal information. However, we do not consider this work well done because 1) The experiment is done on only 3,140 users with personal information both insufficient and unreliable. 2) Personal information is not treated "personally" – they are simply divided into 3-grams and the probabilities of these 3-grams are raised up a little in the attack dictionary. 3) Dividing personal information into 3-grams internally break the continuation of personal information and causes many coincidences.

We believe the effect of OMEN+ [9] is roughly equivalent to adding common names, locations, etc to training data. It tries to use personal information of individual user to generate guesses for all users, which is by its nature improper.

4.1 Attack Scenarios

Personal-PCFG can be used in both offline and online attacks. We assume that the attacker knows certain amount of personal information about the targets. The attackers can be an evil neighbor, a curious friend, a jealous husband, a blackmailer, or even a company that buys personal information from other companies. We argue that under these conditions, target personal information is rather easy to obtain – The attackers might either know the victim personally or can easily get personal information by searching online, especially on social networks sites (SNS).

In traditional offline password attacks, attackers usually steal hashed passwords from victim systems, and try to find out the unhashed values of these passwords. As hash process cannot be simply reversed, the most usual solution is by brute force, namely guessing and verifying passwords. Every guess is verified by hashing the guess (salt is also likely to be added) and comparing the result to the hashed values. high-probability password guesses can usually match many hashed values and thus are expected to be tried first. In offline attacks, Personal-PCFG is much faster in guessing out the correct password than conventional methods by generating high-probability personalized passwords.

As for online attack, the attacker does not even have a hashed database. Instead the attacker tries to directly log in the real systems by guessing the passwords. It is considered more difficult than offline attacks because service systems usually have restrictions on log in attempts for a given period of time. If the attempt quota has been reached without inputting the correct password, the target account may be locked either for some time or permanently unless certain actions are taken (For example, call the service provider). As a matter of fact, online attacks need to be very fast – within at most tens of guesses the password should be cracked out. Personal-PCFG is able to crack 1 out of 20 passwords within only 5 guesses.

4.2 A revisit of PCFG

Personal-PCFG is based on the idea of PCFG method. Before we introduce Personal-PCFG. We revisit the basic principle of PCFG method. PCFG method pre-processes passwords by their structures. From the pre-process, base structures such as "$L_5D_3S_1$" are generated 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 in training set. Therefore higher probability segments will be tried first. As a result, one base structure may have many substitutions, for example, "$L_5D_3S_1$" can have "$L_5$123!" and "$L_5$691!", etc as its substitutions. These new representations are called pre-terminal structures. No "L" segment is currently substituted because the space of alpha strings is too large to learn from the training set. These pre-terminals are ranked from high probability to low probability. And finally "L" segments are substituted using a dictionary. Besides the basic idea, PCFG method also carries an efficient algorithm to enumerate passwords from high probability to low probability on the fly. Therefore PCFG method generates password guesses in descending probability order. These guesses are hashed to compare with the

values in password databases. Since PCFG method output statistically high probability password first. It reduces the guessing number of traditional dictionary attacks significantly.

4.3 Personal-PCFG

Personal-PCFG leverages the basic idea of PCFG method. Beside "L", "D", and "S" symbols, it adds more semantic symbols to PCFG method. These additional symbols include "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 predicting passwords. To make Personal-PCFG work, additional matching phase and adapt-substitution phases are added to the original PCFG method. Personal-PCFG works as following.

4.3.1 Matching

Given a password string, we first try to match it or a substring of it to personal information. The algorithm used is the same as that introduced in Section 2.2.2. 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 will remain unchanged. Again, We take Alice, who was born in Aug. 16 1988, as an example. This time we assume her password is "helloalice816!". Matching phase will replace "alice" with "N_5" and "816" with "B_3". The leftover is kept unchanged. Thereby the outcome of this step is "$helloN_5B_3$!".

4.3.2 Processing

In this step we simply run the processing routine of original PCFG method on the output of last step. The segments that are matched to personal information are not processed in this step. The output in this step is a base structure for Personal-PCFG. Now the password is fully described by semantic symbols of Personal-PCFG method. Our example password "$helloN_5B_3$!" will update to "$L_5N_5B_3S_1$".

4.3.3 Guess Generation

Like original PCFG method, we replace "D" and "S" symbols with actual strings learned from the training set in descending probability order. "L" symbols are also replaced with words from a dictionary. We use the "NEXT" function [39] in original PCFG method to output the results on the fly so we do not need to wait all possibilities for guesses are calculated and sorted. The results will be kept outputting for next step. Note that we do not replace any symbols for personal information so the guesses are still not actual guesses. We do not handle personal information in this step because personal information for each user is likely to be different. Personal information symbols can only be substituted until the target is specific. In this step our base structures generate pre-terminals, which are partial guesses that contain part of actual guesses and part of Personal-PCFG semantic symbols. The example "$L_5N_5B_3S_1$" becomes "$helloN_5B_3$!" in this step if "hello" is the first 5-symbol long string in the input dictionary and "!" has highest probability of occurring for 1 symbol long special character strings in the training set. Note that for "L" segments, every words of same length has 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 Adapt-substitution

The output of last step can be applied to any target user. However, this will no longer be true in this step because the guesses will be further substituted with personal information, which are specific to only one target user. Personal information symbols are replaced by corresponding personal information of same length. If there are multiple candidates of same length, every combination is output for trial. In our example guess $helloN_5B_3$!, N_5 will be directly replaced by "alice". However, B_3 has many candidate segments. Any length 3 substring of "19880816" can be a candidate. Therefore every substring will be output. The guesses are "helloalice198!", "helloalice988!", ..., "helloalice816!". We try these candidates one by one until we found 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 because same length segments may not always be available. For example, a $helloN_6B_3$! is not suitable for Alice since her name is at most 5 symbols long. In this case no guess from this pre-terminal structures will be generated for Alice.

4.4 Cracking Result

We use 12306 dataset in both Personal-PCFG and original PCFG method. 12306 dataset has 131,389 users. We use half of them as training set, and the other half as testing set. Besides, both of the methods involve selection of dictionaries for the "L" segments. Dictionaries are a vital part for password cracking. To eliminate the effect of unfair dictionary selection, we use "perfect" dictionaries in both of the methods. "Perfect" dictionaries are dictionaries we collected directly from the testing set, in which any string in the dictionary is useful and any string in the passwords appear in the dictionary. So a perfect dictionary is guaranteed to find correct alpha strings efficiently.

We realized that most previous works on password cracking use guessing number to compare different methods. However, we use hashing number to evaluate Personal-PCFG. We argue that as hashing is the bottleneck of such attacks, it is more reasonable to compare hashing number instead of guessing number. Researchers use guessing number in their works because guessing number is independent of dataset size while having linear relation to hashing number. However, we are facing a different condition. For non-personalized guesses, each of them needs to be padded by the salt before hashing. Therefore, a dataset of N passwords requires N hashes to verify each guess. However, in Personal-PCFG, one guess is usually personalized to test on only one password so it requires only 1 hash to verify a guess. In this sense, insist on using guessing number to compare 2 methods is inappropriate.

To adapt to tradition and make more understandable presentation, we show the result in Figure 3 in terms of average hashing number, which is essentially equivalent to guessing number as conventional password cracking mechanisms. We compute the probability of a password that can be cracked given the hashing number. Figure 3 implies that Personal-PCFG works substantially better than original PCFG method. Some interesting observations are listed in following.

1. Both PCFG and Personal-PCFG can crack passwords

Figure 3: Compare PCFG and Personal-PCFG.

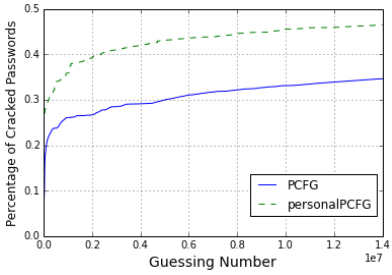


Figure 4: PCFG and Personal-PCFG – Online attacks.

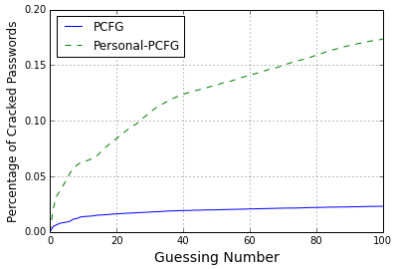
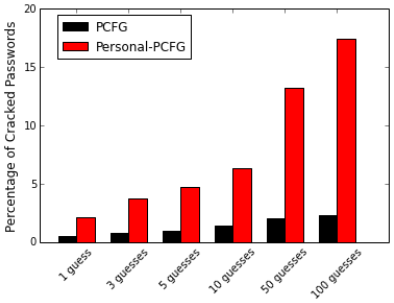


Figure 5: Representative Points – Online attacks.



quickly when they just start. However, the line become flat when the guessing number is growing large. This is not surprising because both methods try high probability guesses first.

2. Personal-PCFG works much better than PCFG. With 10 million hashes Personal-PCFG achieves the probability that is achieved with over 300 million guesses by original PCFG method. That is to say, Personal-PCFG can crack password much much faster than PCFG method.

3. Personal-PCFG is able to cover larger password space than PCFG method. This is because personal information provides rich personalized strings which may not appear in the training set.

After illustrating the benefit of Personal-PCFG in offline attacks, we also show the effectiveness of Personal-PCFG in online attacks. Online attacks allow much less guessing attempts because a real system usually restrain login attempts for a specific user. Therefore we limit the guesses to be at most 100 times. In fact, most systems will not allow login attempts to exceed 10 time.s. We present the results in Figure 4.

As shown in Figure 4, in online attacks, generally Personal-PCFG is able to crack 323% to 649% more passwords than original PCFG. We then take a look at several representative guessing numbers. These interesting points are shown in Figure 5. A most typical system allows 5 attempts to input the correct passwords. Personal-PCFG is able to crack 4.7% passwords within only 5 guesses. Meanwhile the percentage is only 0.9% for original PCFG method. In addition, it would take over 800 guesses for PCFG to reach a success rate of 4.7%. Clearly Personal-PCFG is more powerful to crack out the correct passwords within only a few guesses.

To conclude, we show that Personal-PCFG beats PCFG in both online and offline attacks substantially. The only restriction of Personal-PCFG is that it requires personal information. However, we have showed personal information is not hard to obtain under our attack scenarios.

5. PASSWORD PROTECTION

For human-chosen passwords that users have the right to choose and update their passwords, since a long and random secure password is less memorable, users may sacrifice password security for usability. Since 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 with good memorability, we introduce *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 calculate a simple function to create a stronger password. This distortion function can be chosen by users and 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, *add1* function simply replaces each letter with the one 1 letter later in the alphabet and replaces single digital number i with $(i+1) mod 10$. It is similar to the Caesar Cipher [20]. In another example, *addpi* 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 *gapx*, 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.

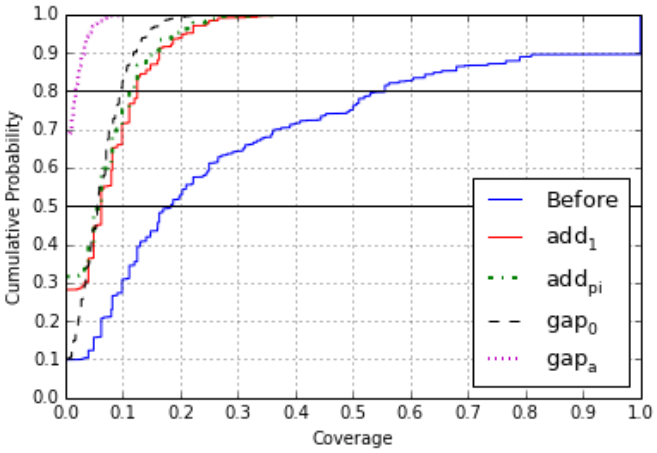
The distortion function must be simple enough for users to remember and generate the passwords. We apply a number of distortion functions on each of the password in 12306 dataset individually and calculate the Coverage for the converted passwords. As Figure 6 shows, the distortion functions are effective on increasing password security. In general, distortion functions can greatly reduce 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) since users rarely have special characters in their personal information. Surprisingly, the non-linear *addpi* function does not produce a better result than other linear functions such as *add1*, since digits prefered by Chinese users are likely to have coincidence 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 crack-

ing methods [36], which leverages semantic patterns in passwords to crack other passwords. After applying a distortion function, the semantic pattern is no longer available. Distortion function is also effective against context-free password crackers using PCFG method [39], since it generates unrecognizable dictionary words, which are not 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.

Figure 6: Coverage distribution.



6. RELATED WORK

Researchers have done brilliant works on measuring real life passwords. In one of the earliest work [28], Morris and Thompson found that passwords are quite simple and thus are vulnerable to dictionary attacks. Malone et al. [25] 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. [27] measured 25,000 passwords from a university and revealed correlation between demographic or other factors, such as field of study, and passwords. 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. [24] 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. [40] and Kuo et al. [23] investigated the mnemonic based passwords. Veras et al. [37] showed the importance of date in passwords. Das et al. [11] studied how users mangle one password for different sites. Schweitzer et al. [34] studied the keyboard pattern in passwords. Beside password itself, researches have been done on human habit and psychology towards password security [14, 17].

It has been shown that Shannon entropy has lots of troubles accurately describing the security of passwords [7, 21, 32, 38]. Researchers developed a number of metrics to measure passwords. Massey [26] 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 [32], 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 [28]. Time-memory Trade-off in passwords [16] made dictionary attacks much more efficient. Rainbow table [30] reduces table number in [16] using multiple reduction functions. However, recent years as the password policy becomes strict, simple dictionary passwords are less common. Narayanan and Shmatikov [29] used Markov model to generate guesses based on that passwords need to be phonetically similar to users' native languages. Castelluccia et al. [9] improved Markov attack by making guess in approximately descending probability order. In 2009, Weir et al. [39] leveraged Probabilistic Context-Free Grammars (PCFG) to crack passwords. Veras et al. [36] 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].

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 [29] to measure the security of user passwords. Meanwhile commercial password meters adopted by popular websites are proved inconsistent [13]. Feedback can be provided to users using trained leaked passwords or dictionaries [38, 22]. Several works discussed adding security questions on top of passwords [31, 33, 6]. Though some graphics-based [12, 19] or biometrics-based [18] 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 12306 dataset use their birthday as passwords, and male users are much more likely to include their name in passwords than female users. We then develop a novel metric, Coverage, that describes the correlation between a set of personal information and passwords. Besides the natural belief that longer personal information

should yield higher Coverage, Coverage also stresses that a continued personal information match is much stronger than fragmented matches. We develop Personal-PCFG method that which leverages the basic idea of PCFG but adds more semantic symbols to crack passwords. Personal-PCFG generates personalized password guesses by integrating user personal information in the guesses, so it is significantly faster than PCFG and makes online attacks feasible. Finally, we propose to use distortion functions to protect passwords. Through a proof-of-concept study, we verify that distortion functions are effective to defend personal information related and semantic-aware attacks.

8. REFERENCES

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

[2] J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Security & privacy*. IEEE, 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*. IEEE, 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 *CCS*. ACM, 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 *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*. ACM, 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 *WWW'07*. ACM, 2007.

[15] S. Gaw and E. W. Felten. Password management strategies for online accounts. In *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006.

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

[17] 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*. IEEE, 2012.

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

[19] 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.

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

[21] 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*. IEEE, 2012.

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

[23] C. Kuo, S. Romanosky, and L. F. Cranor. Human selection of mnemonic phrase-based passwords. In *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006.

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

[25] D. Malone and K. Maher. Investigating the distribution of password choices. In *WWW'12*. ACM, 2012.

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

[27] 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 *CCS'13*. ACM, 2013.

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

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

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

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

[32] J. O. Pliam. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Progress in CryptologyâĂŤINDOCRYPT 2000*. Springer, 2000.

[33] S. Schechter, A. B. Brush, and S. Egelman. It's no

secret. measuring the security and reliability of authentication via âĂIJsecretâĂİ questions. In *IEEE Security & Privacy*. IEEE, 2009.

[34] D. Schweitzer, J. Boleng, C. Hughes, and L. Murphy. Visualizing keyboard pattern passwords. In *VizSec'09*. IEEE, 2009.

[35] T. Tech. 12306 user info leackage, December 2014. [Online; posted 25-December-2012].

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

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

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

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

[40] J. Yan et al. Password memorability and security: Empirical results. *IEEE Security & privacy*, 2004.

APPENDIX

A. FULL LIST OF BIRTHDAY AND NAME IN MATCHING ALGORITHM

B. TODO

Male and Female passwords cracked