International Journal of Information Security (2019) 18:647–663
https://doi.org/10.1007/s10207-019-00432-3

REGULAR CONTRIBUTION

Understanding user passwords through password prefix and postfix (P3) graph analysis and visualization

Xiaoying Yu[1] · Qi Liao[1]

Published online: 12 April 2019

Abstract
While other authentication methods exist, passwords are still the dominant way for user authentication and system security. Over the years, passwords have become long and complex thanks to security policy and awareness. However, the security of user passwords remains unclear. Therefore, understanding users passwords is vital to improve the strength of passwords and system security in general. In this paper, we investigate one specific pattern, i.e., the prefix and postfix of user passwords. To facilitate password prefix and postfix (P3) analysis, we propose both hierarchical segmentation / optimization algorithms and password prefix/postfix graphs (P3G) construction and P3G visualizations. Through case study over real-world user passwords, we demonstrate P3 analysis and visualization are effective in identifying unique patterns for different user categories. The results suggest strong correlations between prefix/postfix and their context in user passwords.

Keywords Computer security · Password analysis and visualization · Prefix and postfix graphs · Hierarchical segmentation · Dynamic programming

1 Introduction

We use passwords and keys in every aspect of daily life, e.g., logging into computer systems, purchasing items from online retailers, accessing bank accounts, encrypting local hard drives and messages, etc. The strength of passwords is critical to protect the security of modern systems, whether they are personal financial accounts, corporate servers, databases, or IoT devices. While there have been applications of other forms of authentications such as biometrics and graphical passwords [11], text-based passwords are still the dominant method of authentication, which will be unlikely to change in the foreseeable future [14]. Therefore, understanding user passwords has become increasingly important in order to assess the strength of passwords and security policies, and ultimately to build more secure systems [15].

With the benefits of understanding passwords, there has been previous research on analyzing password patterns and their structures. For example, keyboard patterns have been analyzed and visualized [9,18]. Dates are another pattern of interest for analysis [4,22]. Organizations have adopted increasingly complex password policies [13] with mandatory combination of characters. The security outcome of such policies, however, remains unknown. Research suggests longer passwords do not necessarily mean more security and there exist significant repetitive patterns of both directions in user passwords [28].

We have observed the trend for organizations to adopt changes that more closely reflect recommended best security practice as defined in NIST 800-63-3. In particular, paraphrases must have a minimum length, an expiration time, and patterns involving mixture of uppercase, lowercase, numbers, special characters. Recently, research has suggested using long, sentence-based passwords [26] (e.g., “my daughter is 12 year old.”) that are presumably easy to remember but still relatively secure while other research suggests the strength of long sentence-like or phrase-like passwords does not increase uniformly with length [17]. To that end, in this paper we try to understand user passwords by analyzing the *prefix* and *postfix* patterns and their structures. Some questions include “given a known plaintext, e.g., “love,” can we guess the words before and after “love” that users may choose in their passwords?”, “are there any unique patterns of prefix and postfix words associated with users from different backgrounds (culture, language, or religion)?” etc.

✉ Qi Liao
liao1q@cmich.edu

[1] Department of Computer Science, Central Michigan University, Mount Pleasant, MI 48859, USA

To analyze password prefix and postfix (P3) patterns, there exist several challenges. First, how to segment user passwords in a meaningful way for analysis. For example, a password "password123" may be divided as "pass,word,1,2,3" or "password,123". Which one makes more sense and also is more scalable for analysis? We developed new hierarchical segmentation algorithms ranging from greedy to dynamic programming algorithms. We studied methods of hierarchical segmentations and their impact on the P3 analysis in terms of granularity and scalability issues.

One major contribution of our paper is to propose a novel algorithm for the construction of password prefix and postfix graphs (P3G) and P3G interactive visualizations. Nodes in P3Gs are relative prefixes, postfixes and keywords, and paths along nodes indicate the relationships of words chosen to compose the final user passwords. Efficient algorithms have been designed and implemented to compute relevant subgraphs based on user interactions so that the unique patterns may be analyzed effectively. In particular, we developed a web-based visualization tool that combines multiple views such as P3G, P3 context (P3C), and P3 relationship (P3R) that together provide more insight toward understanding user passwords than from just one angle.

Through case studies over a real-world dataset (RockYou [10]) that contains 14 million user passwords, our findings suggest strong correlation between the prefix and postfix in user password construction. P3 analysis reveals numerous important patterns. For example, people's names are common components of passwords, and the prefix and postfix in such passwords often exhibit "name + name" and "emotion verb + name" patterns. We identify common substitutions of prefix/postfix patterns involving similar sounds, as well as not so obvious substitutions involving cultural and linguistic backgrounds. For example, Chinese speaking users tend to make one unique substitution of prefix in their passwords. We find users tend to use numbers as their password postfix. Consequently, we study the length distributions as well as unique patterns for these numerical postfixes, from the more general "word + digits" to the more specific "name + dates" patterns, in which dates are more uniformly distributed for people's names than location names. Other findings include patterns for users of diverse backgrounds (e.g., religious) and social roles (e.g., husband vs. wife), etc.

The rest of the paper is organized as follows. Section 2 discusses related work on password analysis. Section 3 introduces the hierarchical segmentation algorithms and optimizations using dynamic programming. Section 4 formally defines password prefix postfix graphs (P3Gs) and explains how to construct such graphs. Algorithms to compute subgraphs of P3Gs are also discussed. The visualization tool is presented in Sect. 5 to illustrate the design and implementation of P3G, P3C and P3R. Usage cases of P3 analysis are demonstrated in Sect. 6 over real user passwords. A number of unique patterns of prefix/postfix are identified in this section. Finally, we conclude our research paper in Sect. 7.

2 Related work

Passwords are the predominant way to authenticate users and to ensure security of information and systems. Understanding how passwords are constructed is useful to assess the strength of passwords [15]. Therefore, the analysis of patterns of user passwords has been the interest of security researchers. For example, approximately 70 million passwords of Yahoo! users [3] were analyzed and guessing difficulty of skewed distribution of passwords was evaluated.

Among the analyzed password patterns, dates have been significant components of passwords. Studies suggest that most of the dates that users choose to be part of their passwords are birthdays [22]. In particular, customers commonly choose 4-digit PINs based on sequence similar to MMDD [4]. In addition, dates in passwords have been visualized [22]. Another important pattern in user passwords is the keyboard pattern, which has been analyzed [9] and visualized [18]. While some passwords may appear random, they use certain keyboard combinations, e.g., "azsxdcfvg" for consecutive keys in "vvvv" shape.

With stricter policies and by enforcing password meters, users are forced to create longer passwords [13]. While entropies have been used to measure the security of password creation policies [25], such password strength has also been studied by an empirical analysis of real-world passwords [12]. An online experiment conducted by Carnegie Mellon University [20] evaluates the password policy for a security/usability trade-off. The study reveals that adding requirements to policies on longer passwords can reduce the number of easily guessed passwords and that certain combinations of requirements can increase both security and usability than the traditional complex policy. In addition, research suggests there have been significant repetitive patterns in user passwords from both directions, even though the passwords satisfy the length requirement [28].

Online password guessing algorithms such as TarGuess [24] use users' personally identifiable information (PII) and sister passwords leaked from users' other accounts. Personal-PCFG [16] analyzes the correlation between passwords and personal information (birthday, name, gender, etc.), and specifically investigates one Chinese railway ticket website (12306) dataset. The study shows that personal information can make password cracking much faster.

There has been research to discuss the semantics of passwords. Semantic patterns of passwords and the security impact on the model can be evaluated by segmentations using natural language processing (NPL) techniques [23] and N-gram frequencies. This research proposes segmentations with dynamic programming as well as a hierarchical segmentation approach, e.g., characters, dictionary words, merging, etc., for the password prefix/postfix analysis. Researchers found that most users create their passwords regarding themselves, and the primary elements of the passwords are composed of names and dates through surveys [6]. The research of grammatical structures [17] of passwords also suggests that long passwords and sentence-based passwords are not taken as strong passwords.

As for the hierarchical graph analysis, Google researchers have studied a distributed balanced partitioning problem [1], which is to partition graph vertices into k pieces by minimizing the total cut size. In addition, OnionGraph [21] has been proposed to examine large graphs by grouping and expanding nodes by topological and semantic categories. Similarities between user passwords can be analyzed in password graphs using edit distances [29]. In this paper, we design hierarchical segmentation algorithms based on dynamic programming. Most notably, we analyze a different pattern, i.e., the prefix and postfix of user passwords, using password prefix postfix graphs (P3G) combined with other visualization techniques.

3 Segmentation algorithms of passwords

Segmentation of user passwords is the key for password prefix and postfix analysis. Since different methods lead to different results, in this section, we discuss in detail the algorithms we develop to divide passwords into meaningful substrings in a hierarchical approach. In particular, we focus on how to achieve optimal segmentation using dynamic programming techniques.

3.1 Dynamic programming

Dividing passwords into meaningful words (prefix and postfix) is non-trivial. One way is to check whether any part of each password has known dictionary words. A greedy algorithm may work in most cases by reversing the iteration direction from end to front until matching a known word. However, such an algorithm may not be the optimal solution in some cases. For instance, a password “rockyou123” will be divided into “rocky,o,u,123” by using the greedy method. Apparently, the result makes little sense to humans. We make a crucial observation that further division is always based on the previous optimal result. Based on the observation, we develop a dynamic programming solution to get the optimal segmentation of each password. Before we discuss the details of the algorithm, we first introduce the rules of optimization.

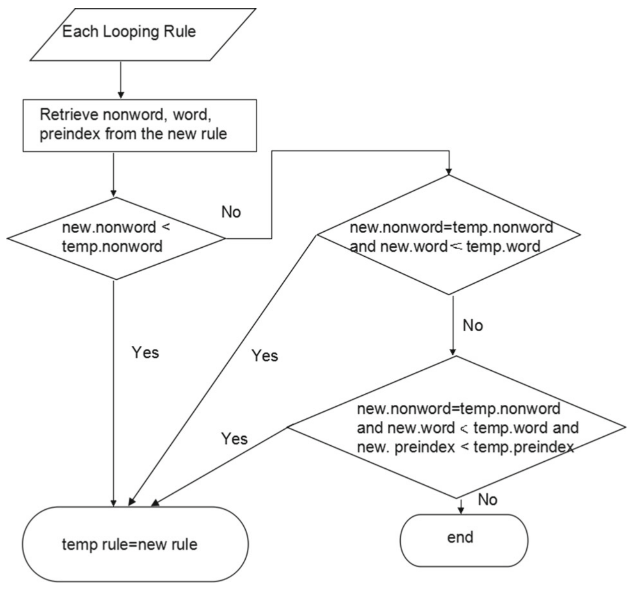

Each Looping Rule
Retrieve nonword, word, preindex from the new rule
new.nonword < temp.nonword
No
new.nonword=temp.nonword and new.word< temp.word
Yes
Yes
No
new.nonword=temp.nonword and new.word < temp.word and new. preindex < temp.preindex
Yes
No
temp rule=new rule
end

Fig. 1 Rules of selecting optimal segmentation results

3.2 Rules of optimization

We formulate a set of rules, which define how to choose the ideal segmentation. A result set R is defined as a three-tuple, i.e., *{nonword, word, preindex}*. The “*nonword*” is the quantity of substrings that are not taken as meaningful words (i.e., dictionary based or common words), while the “*word*” represents the number of meaningful words. The “*preindex*” records the index position, where a string or substring is segmented. During each iteration, the result set is recorded. The best solution $r_{opt} \in R$ is then selected. The process of identifying the optimal solution is illustrated in Fig. 1.

The primary rule of selecting the optimal solution is to choose a result tuple r that contains the minimum *nonwords* so that each password can be partitioned into as many *words* as possible. However, if the *nonwords* in both new and temporary (i.e., holding current best solution) result tuples are same, the result with fewer *words* is considered as a better solution. The reasoning behind this rule is that had we taken the result with more *words* as a better solution, words would have been further segmented into even more words. For example, a substring “heart” will be further divided into “he” and “art.” In this case, the prefix and suffix of keywords are losing their meanings. Furthermore, if the *nonword* and *word* in both results are same, the smaller *preindex*, which means the parsed substring is longer in the new result than that in the temporary result, is preferred.

Algorithm 1 Password Segmentation using Dynamic Programming

```
 1: procedure SEGMENT
 2:     read passwords from file
 3:     d ← Add dictionary words and names
 4:     for each password do
 5:         N ← the length of one password
 6:         Add initial value to result[0]
 7:         for $i$ ← 1 to $N + 1$ do
 8:             result append (0, i, i-1)
 9:             if isAWord(d, password[:i]) then
10:                 $result[i] \leftarrow (1, 0, 0)$
11:                 continue
12:             for $j \leftarrow i$ to 0 do
13:                 if isAWord(d, password[j:i]) then
14:                     $tmpresult \leftarrow result[j]$
15:                     $tmpresult.d4 += 1$
16:                     $tmpresult.d6 = j$
17:                 else
18:                     $tmpresult \leftarrow result[j]$
19:                     $tmpresult.d5 += 1$
20:                     $tmpresult.d6 = j$
21:                 if needChange($result[i]$, $tmpresult$) then
22:                     $result[i] = tmpresult$
23:         DIVIDE(result[N])
24: end procedure
```

Algorithm 2 Divide Password

```
 1: procedure DIVIDE(result[N])
 2:     $word = result[N].d1$
 3:     $nonword = result[N].d2$
 4:     $preindex = result[N].d3$
 5:     for $i \leftarrow N - 1$ to 0 do
 6:         if nonword = result[i].d2 & result[i].d1 = word-1 & preindex
            > result[i].preindex then
 7:             word = result[i].d1
 8:             position append result[i].d3
 9:             preindex = result[i].d3
10:     L ← length of position
11:     for $i \leftarrow L - 2$ to 0 do
12:         startpos = position[i+1]
13:         endpos = position[i]
14:         res append password[startpos:endpos]
15:     if res is not NULL then
16:         D1 append res
17: end procedure
```

3.3 Segmentation algorithms using dynamic programming

The entire process of password segmentation using dynamic programming is explained in Algorithm 1. Among the parameters, $password$ represents each password string; d is a map of dictionary words; $result$ is a list storing each position's optimal solution; $result(d1, d2, d3)$ contains $d1$ (number of words), $d2$ (number of nonwords), and $d3$ (preindex); *tmpresult*$(d4, d5, d6)$ is the list of temporary results that contain $d4$ (number of words), $d5$ (number of nonwords) and $d6$ (preindex).

Through iteration from index 1 to the length of each password, each outcome is recorded in $result$ tuple. The initial values of $result[i]$ are set, i.e., $word = 0$, $nonwords = i$, and $preindex = i - 1$. If the substring from position 0 to i is a word, $result[i]$ equals "{"word":1, "nonword":0, "preindex":0}". Otherwise, a nested loop is applied with index j ranging from i to 0. If $password[j : i]$ is a word, the $word$ value in $result[j]$ is increased by 1, and the $preindex$ records the j position. If it is not a word, the $nonword$ value in $result[j]$ is increased by 1, and the $preindex$ records the current position. After the computation, the optimal result of the current position is retrieved, and the value is saved in $result[i]$.

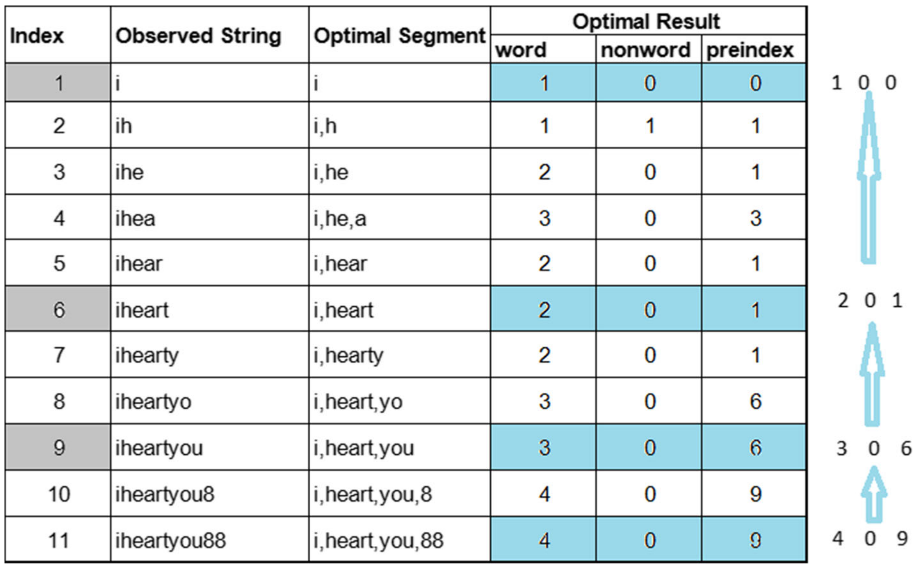
DIVIDE procedure in Algorithm 2 is used to divide a password after iterating all positions in a password. Among the parameters, res is a result list for storing each divided password; $position$ is a list for saving the positions of $preindice$ that are chosen from the previous dynamic programming results; $D1$ is the output list storing all the divided passwords. In this procedure, the last value represents the best value after the whole analysis.

The time complexity of Algorithms 1 and 2 takes $O(M * N^2)$ steps, where M is the total number of passwords for analysis and N is the average length of passwords. N is usually small for each password, i.e., < 26. The space complexity of this segmentation algorithm is $O(N)$, which is for storing the optimal result in every position in each password.

3.4 Segmentation example and comparison

Figure 2 demonstrates one example of how to get the optimal segmentation result using dynamic programming. In the dynamic programming table, there are 4 main columns. The first column, named *Index*, is for recording the exact position where the substring is observed. The indices are prepared for later revision as well. The *Observed String* column shows the substring that has been processed so far. The column of *Optimal Segment* explains how the string is divided in the best way in current stage. The *Optimal Result* column includes the three-tuple result set, which includes sub-columns: *word*, *nonword*, and *preindex*. These sub-columns store the temporary optimal solution in the current stage. The idea is that if one of the previous results has 0 *nonword*, 1 fewer *word*, and the smaller *preindex*, the position is where the password should be segmented. The segmentation is traced from back to front. After that, the substrings within these positions are stored into D1 array, which stores the final segmentation of the password.

The result of each step of "ihear" segmentation is shown in Table 1. The same analysis can be applied to other observed strings as well to get each segment result. According to Algo-


1 0 0
2 0 1
3 0 6
4 0 9

Index	Observed String	Optimal Segment	Optimal Result		
			word	nonword	preindex
1	i	i	1	0	0
2	ih	i,h	1	1	1
3	ihe	i,he	2	0	1
4	ihea	i,he,a	3	0	3
5	ihear	i,hear	2	0	1
6	iheart	i,heart	2	0	1
7	ihearty	i,hearty	2	0	1
8	iheartyo	i,heart,yo	3	0	6
9	iheartyou	i,heart,you	3	0	6
10	iheartyou8	i,heart,you,8	4	0	9
11	iheartyou88	i,heart,you,88	4	0	9

Fig. 2 Example of password segmentation algorithm using dynamic programming. Table 1 illustrates the segmentation steps of one observed string "ihear"

Table 1 Segmentation steps of one observed substring "ihear" (index 5 in Fig. 2)

Division	Based result	Current result
ihea,r	303	314
ihe,ar	201	213
ih,ear	111	212
i,hear	100	201
ihear	000	010

"i,hear" is the best segmentation in the current stage with the optimal result 3-tuple (201)

rithm 1, each substring is iterated from back to front. The first observed result is "ihea,r", and the segmentation position, recognized as comma position in the string, is 4. Since 'r' is not recognized as a word, the *word* part is still same, the *nonword* part increases by 1, and the segment position (*preindex*) is 4. The result tuple becomes 314. The optimal result set for the substring "ihea" is already computed as 303. From the "Current Result" column in Table 1, according to the rule defined earlier, we know that 201 is the optimal result of "ihear," where 2 means there are two *words*, 0 means there is no *nonword*, and 1 means the segmentation position is after the first character, which is "i." Therefore, the optimal segment in the current round now becomes "i,hear".

As illustrated in Table 2, the solution searching direction is from the 11th step back to the first step. A highlighted row is the final position of each substring. In the 11th step, the optimal result is 409, which means that the best result of the segmentation of the password is to divide it into 4 *words*, 0 *nonwords*, and the last segment position is after the 9th character. The third word is defined at the 9th position, at which 306 is retrieved, and the last word is "88." Based on 306, we know that the previous word should be defined at the 6th index, and 201 is retrieved, and the third word is from the 7th character to the 9th character, i.e., "you." The same logic applies to get the second word "heart" and the first word "i." Therefore, the resulting optimal segmentation of the password "iheartyou88" is "i,heart,you,88".

Compared to an existing segmentation library WordSegment (WS) derived from Natural Language Corpus Data chapter in the book Beautiful Data [19], both WS and our program perform roughly the same on most cases. For example, "statefarmisthere" will be divided into "state, farm, is, there." WS performs better sometimes, e.g., pokemon and Samsung are considered one word in WS. We do not consider this is a fundamental limitation of our algorithm since our system uses a smaller dictionary that does not contain the game and company names. On the other hand, our program performs better in some cases. For example, password "iheartyou88" is treated as a whole word in WS, but is divided as "i, heart, you, 88" (see Fig. 2) in our program. The other limitation in WS is that it removes all special characters during segmentations. For example, "iloveyou123!567" is divided into "i, love, you, 123567" using WS, but "i, love, you, 123, !, 567" in our program. In our password prefix/postfix analysis, special characters are indeed important, especially for the passwords that contain feeling/emotional words and substitutions. Again, we do not view this is a fundamental limitation of WS implementations since the code can be easily modified to retain all special characters.

We note that sometimes the algorithm designed to take the fewest number of words may not always be the ideal case for human analysis. Even with the same number of words, e.g., "andad," sometimes it is difficult to decide which one is better, "an, dad" or "and, ad". In our implementations, we used Python Enchant library. Choices of dictionaries with different sizes may also affect the results of segmentations.

Table 2 Trade-off of hierarchical password prefix and postfix (P3) analysis

Layer	Pros	Cons
Layer 1	Best scalability	Dense mesh-like layout reducing insights
Layer 2	Comprehensive segmentation. Can be used to investigate hidden patterns	Less scalable. May decouple relational words
Layer 3	More scalable. Focused and humanized segmentation	Bias, incomplete segmentation. Lost hidden patterns
Layer 4	Comprehensive segmentation. More scalable.	Limited analysis of digits and symbols
Layer 5	Most scalable. Focused and humanized segmentation	Limited analysis of digits and symbols

Table 3 Comparison of the magnitude of substrings (graph nodes) with various segmentation options, i.e., layer 1 (character), layer 2 (dictionary), layer 3 (common words), layer 4 (layer 2 + aggregation) and layer 5 (layer 3 + aggregation)

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Non-word node	96	6,917,503	9,273,483	5,172,368	7,395,428
Word node	96	307,374	193,782	40,899	10,260

Rather than evaluating which segmentation algorithm or dictionary is superior, we focus on the study of different methods of hierarchical segmentations with dynamic programming approaches and their impact on the P3 analysis in terms of granularity and scalability issues, as discussed in the next section.

3.5 Hierarchy of segmentations

In this section, we compare and discuss the impact of a hierarchical design of password prefix and postfix ($P3$) analysis based on various segmentation algorithms. The visualization (discussed in the next section) that we develop is able to analyze the password prefix and postfix graphs (P3G) that result from the hierarchical segmentations.

Table 2 summarizes the trade-off among the layers of segmentation algorithms. Since the magnitude of passwords for a large-scale system is commonly huge, an important research question is how to scale down the data size while preserving meaningful granularity for the accuracy of analysis. Table 3 compares the quantity of segmented substrings under various schemes. The sample size of passwords is 14,344,391 in this comparison study. “Non-word node” means segmented passwords may include one or more non-words, while “word node” means segmented passwords contain only meaningful words. Since the substrings are used as prefix and postfix nodes in $P3$ graphs, the sizes of graphs are impacted accordingly.

Layer 1 (Character): In the simplest form, each password is divided by characters. There are 96 displayable characters (letters, digits and symbols) leading to the smallest graphs. Therefore, there are only 96 graph nodes in this layer. Although the graph size (in terms of number of nodes and edges) is the smallest, the edges tend to be uniformly distributed among all nodes, forming a mesh-like topological layout, thus reducing analytic insight significantly. This layer is suitable for the analysis of frequency of characters in user passwords for large graphs since the size does not scale in terms of the number of passwords.

Layer 2 (Dictionary): Layer 2 and above use our dynamic programming approach with options. In this layer, each password is divided into substrings according to the dictionary database. After segmentation with dynamic programming algorithms, there are 6,917,503 meaningful substrings. In addition, we match each substring of passwords against a dictionary database, and only dictionary words are chosen. The reason for taking this measure is that it is unlikely we can retrieve meaningful semantics from non-meaningful words (or non-words). By filtering the non-words, the result contains 307,374 words. This layer indicates the most comprehensive segmentation with meaningful substrings and can potentially be used for investigating hidden patterns. However, it also means that the segmentation of passwords may sometimes obfuscate the relationships among words. This layer is suitable for private information investigation, where prefix and suffix information includes birth dates, such as a string of digits, national culture.

Layer 3 (Common): In contrast to Layer 2, Layer 3 uses a smaller dictionary, i.e., the top 5,000 frequently used words. Within this layer, each password is divided into substrings by comparing to so called common words database. Since many combinations of digits and symbols are not among the common words, this option results in the most unique substrings, i.e., 9,273,483 substrings are produced by the segmentation.

By filtering passwords that contain at least one non-word, 193,782 words are staged in the final analysis. Unlike Layer 2 which is more general and broad, Layer 3 is more focused and scalable. The segmentation in this layer is closer to routine words in human society because the database is from the summarization of the most frequent words. However, this segmentation also brings some effects, such as bias in choosing common words, which leads to incomplete segmentation and lost hidden patterns. This layer is suitable for investigation of the relationship between digits' suffix and frequent words.

Layer 4 (Layer 2 with aggregation/merging) and Layer 5 (Layer 3 with aggregation/merging): Layers 2 and 3 produce excessive number of nodes of digits or symbols. For example, "1," "12," "21," "1234," "234," "!," "!!," etc., will be all considered as different nodes. For most analyses, it is not important to distinguish these digits or symbols. Based on this, we aggregate all digits, symbols and anything else that are less important as mega-nodes, denoted as "DIGIT," "SPECIAL," and "OTHERS," respectively. Layer 4 divides each password by comparing it to the dictionary words, while Layer 5 segments passwords by comparing them to common words, both encapsulating the digits, the symbols and other nodes at the same time. Layers 4 and 5 produce 5,172,368 and 7,395,428 substrings, respectively. After removing passwords that contain non-words, there are 40,899 and 10,260 nodes left. The advantage of Layers 4 and 5 is that they inherit the pros of Layers 2 and 3 while making them much more scalable to larger datasets. The resulting graphs' sizes for visualization are reduced significantly, making it easier to reveal the patterns in password prefix and postfix graphs. The disadvantage is that any pattern within digits and special characters is lost since these nodes are aggregated. Therefore, Layers 4 and 5 are suitable for the analysis of semantic patterns of a large number of passwords but not for finer granularity of special words (e.g., birthdays, encoded phrases using special characters).

As discussed above, different layers have their own merits and shortcomings. There exists trade-off between scalability of analysis and granularity of patterns. Some focus on general dictionary words, while others focus on common words and special words that contain digits and symbols. Different patterns of password prefix and postfix compositions may be derived by applying suitable hierarchical layers for analysis.

4 Password prefix postfix graphs (P3Gs)

In this section, we define our password prefix postfix graphs (P3Gs) based on the segmentation algorithms presented in previous sections and illustrate the algorithms for constructing such graphs. In particular, we discuss the hierarchical design of P3Gs by node aggregation resulting from various layers of segmentation algorithms. Additional implementation details are also discussed.

4.1 P3G

A password prefix postfix graph (P3G) can be represented by a directed, weighted node-link graph $G_{P3} = \langle V_{P3}, E_{P3} \rangle$, in which V_{P3} represents prefix and postfix words; E_{P3} represents the relationship of prefix and postfix words; directions (denoted by arrows) signify the relative order of prefix and postfix words that are used to compose the final passwords. A path in P3G is defined as a sequence of adjacent vertices $\{v_1, v_2, ..., v_i\}$ such that these word nodes form a valid user password. For each edge property $P_{E_i} \in P_E$, two node indices are defined for two attributes, i.e., order and magnitude of connecting prefix and postfix words in user passwords. For example, if a link object is "source:5, target:213, LinkTimes:100," it means the fifth node is the previous substring of the 213th node and that relationship appears 100 times in all the passwords.

Algorithm 3 Password Prefix Postfix Graph (P3G) and Properties Construction

```
1: procedure GraphConstruct
2:     $wordSet \leftarrow set$
3:     $subHash \leftarrow HashMap$
4:     $connectHash \leftarrow HashMap$
5:     for $password \leftarrow passwords$ do
6:         wordsArray = password.split()
7:         length = len(wordsArray)
8:         for $i \leftarrow range(0, length)$ do
9:             add wordArray[i] to wordSet
10:            $source \leftarrow Index(wordArray[i])$
11:            $target \leftarrow Index(wordArray[i+1])$
12:            if subHash[source,target] then
13:                subHash[source,target]+=1
14:            else
15:                subHash[source,target]=1
16:        if length > 1 then
17:            for $i \leftarrow range(0, length)$ do
18:                for $j \leftarrow range(0, length)$ do
19:                    if wordArray[i] ! = wordArray[j] then
20:                        if connectHash does not contain key wordArray[i] then
21:                            connectHash[wordArray[i]] = wordArray[j]
22:    store node and link information to a file
23: end procedure
```

4.2 P3G construction algorithms

After having segmented passwords, we construct P3Gs for visualization and analysis by building and connecting the prefix and postfix words and their underneath properties. The details of P3Gs and their property construction are explained in Algorithm 3. Initially, it is necessary to make each word

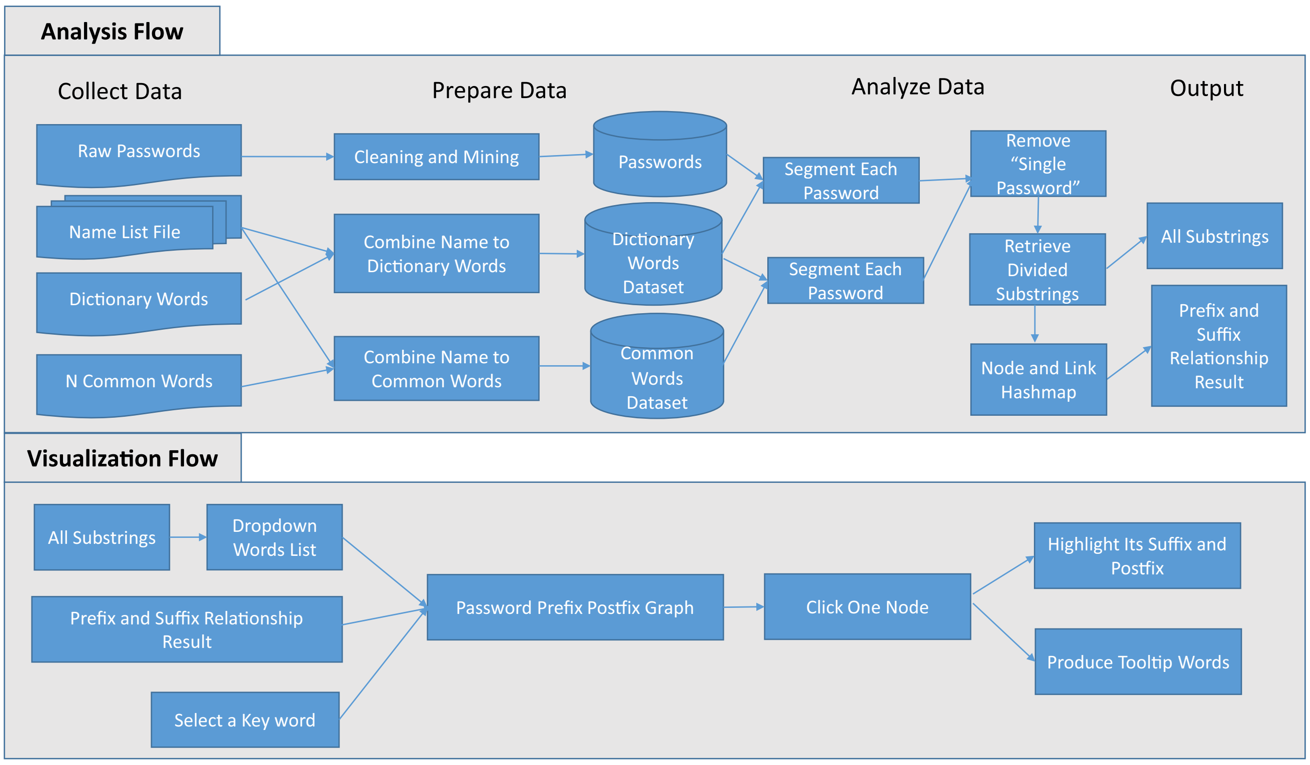


Fig. 3 Password prefix postfix (P3) analytic and visualization process

unique through the usage of a word set. For each keyword, the words that appear in the same password with the keyword are put into two groups, i.e., prefix and postfix. The source–target connection of adjacent word nodes in every password needs to be classified, and the connection magnitude of the same source–target pair is recorded. Then, for each word, it uses source and target variables to record the relationship information. Two hashmaps are used to record the link frequencies and store all other words that appear in the same password as the keys. The graph is then outputted to a file for visualization. The time complexity of P3G and its properties construction algorithm is $O(M * N^2)$, where M is the total number of passwords and N is the average length of passwords.

4.3 P3G interaction algorithms

Interactive visualization is an important design factor in our P3G analytic tool, whose interaction process of visual analytic model is illustrated in Fig. 3. One important problem with any interactive visualization design is how to improve the efficiency of analysis and decrease the time lapse between user interactions, especially when analyzing large datasets. The P3G is designed as an interactive graph, in which users can select any one or more specific nodes to visualize all other nodes that directly or indirectly connect to the selected node. In other words, by selecting multiple k words in the interactive P3G, the problem is to automatically generate a password context subgraph $\hat{G}_{P3} \in G_{P3}$ that contains all prefix and postfix words of the selected k words that together form the final passwords. Such interaction is the key feature of the P3G visualization tool.

The first step of interactive P3G visualization is to prepare the node dictionary, link dictionary, and node index dictionary for faster lookup later. The node information is an array of multiple objects, each of which has node name, connected nodes, and group properties. For each node object, the node name is used as the key of the node dictionary, and the value is an object that contains connection and group information. Within the loop, each node is also matched to one index. The link information is an array of objects including source word, target word, and link frequency properties. The link dictionary is applied to record the target node index and the link times to each source node. The above node and link hashmaps significantly reduce the response time of user interactions.

When a user selects one or more word nodes from the drop down list, results will be generated automatically, which includes the nodes that directly or indirectly connect to the selected node, as well as the links that record the source node index, the target node index, and link times. Utilizing the dictionaries built earlier, the nodes in the connect list are retrieved and stored in data structures, from which new indices are assigned to each node. Then, the link information is separated into target and link times, and each link object is

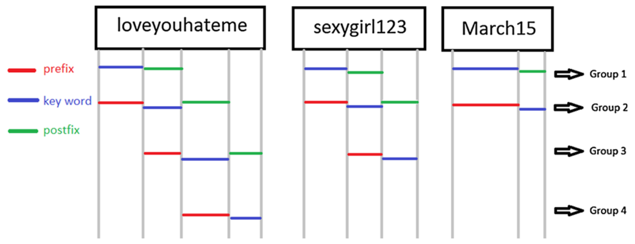


Fig. 4 Relative prefix and postfix construction through different grouping methods

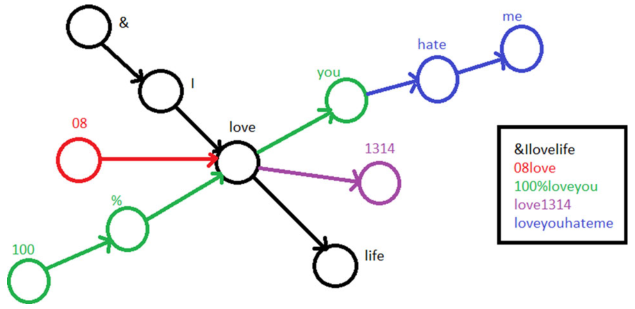


Fig. 5 An example of a P3G context subgraph, which represents five user passwords

produced. Finally, the resulting subgraph $\hat{G}_{P3}$ is generated and stored in a local file ready for the input of front-end P3G visualization.

4.4 P3G construction examples

Figure 4 illustrates an example of relative prefix and postfix in P3G construction. The red, blue and green lines represent prefix, keyword and postfix, respectively. In each password, the word before the keyword is taken as the prefix if it exists, and the word after the keyword becomes the postfix. The recursive process continues on the relative prefix and postfix. For example, a password "loveyouhateme" is segmented into four groups. In the first group, "love" is a keyword and "you" is its postfix. In the next group, "you" is a keyword and "love" is a prefix, while "hate" is a postfix. In the third group, "hate" is a keyword, "you" is a prefix, and "me" is a postfix. In the last group, "me" is a keyword and "hate" is a prefix. It is the same for password "sexygirl123," in group one, when "sexy" is a keyword, "girl" is a postfix. In group two, "girl" is a keyword, "sexy" is its prefix and "123" is its postfix. Finally, in group three, "123" is a keyword and "girl" is its prefix. If a password is composed of two words, these two words are alternatively prefix and suffix. For instance, password "March15" can be treated as two groups. In group one, "March" is a keyword and "15" is its postfix. In group two, "15" is a keyword and "March" is its prefix.

In addition, Fig. 5 illustrates an example of an actual P3G context subgraph. With the "love" node selected as a relative keyword, a subgraph is constructed to represent all relative prefix and postfix that are involved to make up the final user passwords. There are five real passwords which are listed to the right. A smaller, on-demand subgraph like this allows a much more scalable and focused view for an investigator to analyze the complex relationships of prefix and postfix in user passwords.

5 Password prefix and postfix visualization

To facilitate the password prefix and postfix (P3) analysis process, we designed and implemented an interactive P3 visualization tool. In this section, we introduce the components of P3 visualization consisting of the P3 graph (P3G) view, P3 context (P3C) view, and P3 relationship (P3R) view.

5.1 Interactive P3G views

Figure 6 shows an overview of the P3 analysis and visualization tool. There are five views for the P3G visualization by implementing the hierarchical P3 segmentation design as described in Sect. 3.5. View 1 corresponds to Layer 1 segmentation, i.e., passwords are divided by characters. View 2 corresponds to Layer 2 segmentation, i.e., using dictionary words with dynamic programming algorithms. View 3 corresponds to Layer 3 with common words to divide passwords. Views 4 and 5 are graphs based on Layers 2 and 3 but after aggregating digits and special characters. We implemented P3G using D3 libraries and applied classic force layout to build graphs that have node and link properties.

For iterative on-demand analysis as well as better scalability for large graphs, a multi-section list and a slider are also implemented in the tool. The list allows users to select any combination of word nodes. As a result, a new P3G graph will be automatically generated and rendered for analysis. The generated subgraph of the original P3G graph will include password paths consisting of selected prefix, postfix, and keyword nodes. The algorithms are described in Sect. 4.3. As illustrated by the visualization analysis module in Fig. 3, the interaction begins when a user selects a word from the list, as shown in Fig. 6. Nodes, links, index hashmaps, and subgraphs are constructed to produce data that include information concerning the selected word node and all prefix and postfix nodes and links that are related to the selected node. If a user clicks one specific node in P3G, the node and its link will be highlighted. The server will call a PHP script to search the words that contain the selected node and the clicked node. The node information is then sent to P3G for tooltip display.

In addition, a filtering slider bar is implemented to further reduce graph size by the link frequencies. Prefix and postfix words that appear less frequently will be filtered out of

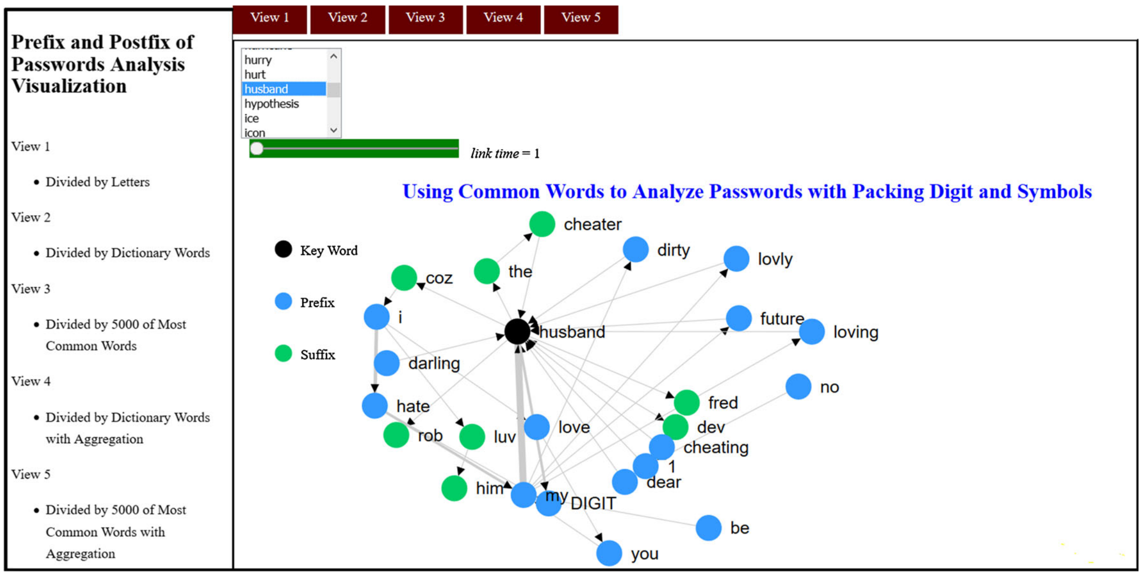

Prefix and Postfix of Passwords Analysis Visualization
View 1
• Divided by Letters
View 2
• Divided by Dictionary Words
View 3
• Divided by 5000 of Most Common Words
View 4
• Divided by Dictionary Words with Aggregation
View 5
• Divided by 5000 of Most Common Words with Aggregation
View 1 View 2 View 3 View 4 View 5
hurry
hurt
husband
hypothesis
ice
icon
link time = 1
Using Common Words to Analyze Passwords with Packing Digit and Symbols
Key Word
Prefix
Suffix
cheater
dirty
lovly
coz
the
i
husband
future
loving
darling
no
hate
fred
rob
luv
love
dev
cheating
1
dear
him
my
DIGIT
be
you

Fig. 6 Overview of password prefix postfix (P3) analysis and visualization using P3 graphs (P3Gs)

the final graph. For example, if the link frequency threshold is set at 1000, then the graph only shows the subgraph in which each pair of nodes connects more than or equal to 1000 times. A threshold of one will show the original graph. Different coloring schemes are used in P3G visualization for better understanding. For example, the black ones denote the relative keywords, the blue ones are prefixes, and the green ones are postfixes.

5.2 P3 context (P3C) views

While P3G offers an intuitive approach to visualize the complex relationships of prefix and postfix used in user passwords, the node-link nature of P3G limits the analysis to the exploration of the connecting information. It is nevertheless cumbersome to visualize the magnitude of each prefix and postfix being used in passwords. To that end, password P3C is implemented using D3 word cloud to compare top prefix and postfix words of particular keywords. The words with higher frequencies will be shown in bigger sizes. An example of P3C is shown in Fig. 10.

It is important to note that the words in P3C are *not* passwords but the particular prefix and postfix words used to make up the final passwords. P3C views are particularly useful to analyze the patterns of prefix and postfix related to a certain category of keywords under user-specific contexts (e.g., location, culture, religion, etc.). For example, are there any patterns (e.g., words used frequently) for users that live in a particular country?

We utilized the color dimension to encode the prefix/postfix information by adopting D3's "PuOr" diverging color scheme. Given a number p in the range [0,1], the function returns the corresponding color as an RGB string. Since each keyword may be either prefix or postfix in different passwords, we calculate probabilities and use dark orange to indicate high probabilities to be prefix and dark purple to indicate high probabilities to be postfix. Light colors indicate somewhere in between.

5.3 P3 relationship (P3R) views

P3R view is implemented using D3 heatmaps to visualize the patterns of pairwise prefix/postfix and keywords. There are various ways to organize the combination of x/y-coordinates for analyzing different patterns. Each axis can be either keyword, prefix, or postfix. The cells denote the ties or relationships among the keywords and prefix/postfix. Color schemes are applied to indicate the magnitude of such relationships, e.g., lighter colors mean less frequent while darker colors mean more frequent of such combination. The color legend at the bottom of P3R visually illustrates the value range. Each color in the legend represents an interval/range of values. The color spectrum starts at light yellow and ends at black.

Figure 11 shows one P3R view. In this particular example, the y-axis represents dictionary keywords of different lengths (from 1 character to 16 characters) and the x-axis represents the keywords' numeric postfixes (also from 1 to 16 characters). The P3R views offer intuitive visual distributions of relationships among the prefix, postfix, and keywords. Diversity is a feature of P3R views as they can be customized to analyze patterns of almost any combination of properties of P3G nodes and edges. We will show more examples of the above views in the case study section.

6 Case study

In this section, we demonstrate usage cases of P3 visualization tool and summarize findings through evaluation over a real-world user password dataset. In particular, the P3G, P3C, and P3R views, when combined, provide insights into the relational patterns of prefix and postfix compositions when users from different backgrounds compose their passwords.

6.1 Data source and preparation

For the case study, we utilize real user passwords from leaked sources. After RockYou.com was hacked, a list of passwords became publicly available [10]. The RockYou dataset contains around 14 million unique passwords from users around the world. For data preparation, we remove noise in the raw data file since the original data include things such as uniform resource identifiers that have more than 26 characters in a line or html code. Considering the purpose of this research study is to analyze the prefix and the postfix of passwords, passwords that contain only the same character in one password (e.g., *aaaaaaaa*) are not included in the analysis.

In addition, we use data that contain popular names [2] and common names [27]. The names are merged into a name set that has 90,453 unique names. As for the dictionary words, the American English Dictionary is imported into Python Enchant, which checks the input string with the words in the dictionary. In addition, we use the common words [8] that are the top 5,000 most frequent words in the human language. Furthermore, the name set and the dictionary words are combined and saved to a dictionary database; while the name set and 5,000 common words compose a common word database.

6.2 Prefix and postfix of people's names

Often when making up passwords, users incorporate their personal information. Names are commonly chosen by users to compose their passwords since they are easy to remember. We discover there are thousands of names that are taken as keywords in the password data. Our finding suggests that prefixes and postfixes of people's names often involve users' private information, such as other people's names, birthdays, feelings.

The prefixes of people's names are often other names. There are several variations, i.e., it can be the first name followed by the last name, or it can also be user's name followed by his/her important person's name (e.g., spouse, child, best friend, pet, etc.). In either case, we refer to it as "name + name" mode. Some users prefer to express their private feeling to another person by setting their passwords with "emotional verb + name," e.g., "hate + name" or "love + name," which we refer to as "feeling + name" mode. In

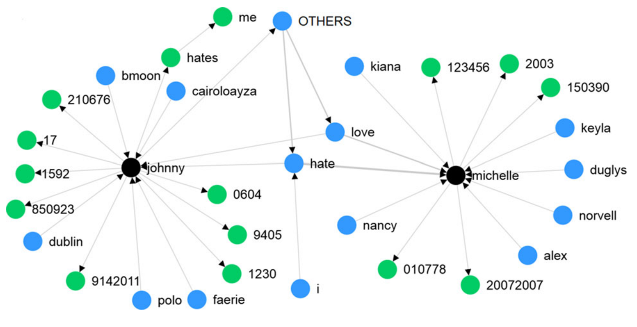


Fig. 7 A P3G showing the prefix and postfix relationship of people's names in the "name + name," "feeling + name," and "name + date" modes. Black nodes are keywords, blue nodes are prefixes, and green nodes are postfixes (color figure online)

contrast to the *prefix* of people's names, we observe that most of the *postfix* of names are numbers in date format, which we refer to as "name + date" mode.

Figure 7 shows a P3G of such prefix and postfix relationships to names. In this example, "johnny" and "michelle" are taken as keywords, whose color is black. The blue nodes are prefixes, while green ones are postfixes of keywords. "OTHERS" node denotes other nodes that also appear in the same passwords as in "name + name" mode, "feeling + name" mode, or "name + date" mode. It is obvious to observe these modes. For instance, prefixes (blue nodes) are either names or feeling expressions, such as "nancy → michelle" for the "name + name" mode, "love → michelle" or "hate → johnny" for "feeling + name" mode. On the other hand, most postfixes (green nodes) of names are digits, most of which may be divided into date format. For instance, "michelle → 150390" can be interpreted as "March 15th, 1990", and "010778" may be interpreted as "January 7th, 1978", which could be the birthday. By cracking passwords using these three modes, an attacker may significantly reduce the cracking space for most passwords.

6.3 Substitution of password prefixes and postfixes

One interesting finding using P3G analysis is that people use special characters and digits to substitute words (usually similar in sounds) that may be culturally specific. Some of these phenomena are similar in people's online chatting or short messaging. For example, "4" substitutes "for"; "u" is used for substitution of "you," etc. As a result, we find it very common that "ever" is the postfix of "4" in user passwords.

Another common substitution is for dates. For instance, after a month, dates can be represented in different forms. Taking postfix "15" of "March" as example, there may be multiple forms: "march15," "march15th," "marchfifteen," and "marchl5." "marchl5" is composed of "march," "l" (use letter "l" to substitute digit "1"), and "5." Digit "0" is also a popular choice for letter "o." The date substitutions hap-

Table 4 Substitutions of dates are common in user passwords

Month	0 = o/O	1 = l/L	2 = Z/e	3 = E	4 = h/A	5 = S	6 = b/G	7 = T/j	8 = X	9 = g/J
January	X	X	X			X	X	X	X	
February					X			X		
March	X	X	X	X	X	X	X	X	X	X
April	X	X	X	X	X		X	X	X	X
May	X	X	X	X	X				X	X
June	X	X	X	X	X	X		X	X	X
July	X	X	X	X	X			X	X	X
August	X		X	X	X		X	X		
September		X	X	X				X		X
October			X	X						
November								X	X	
December			X	X				X		

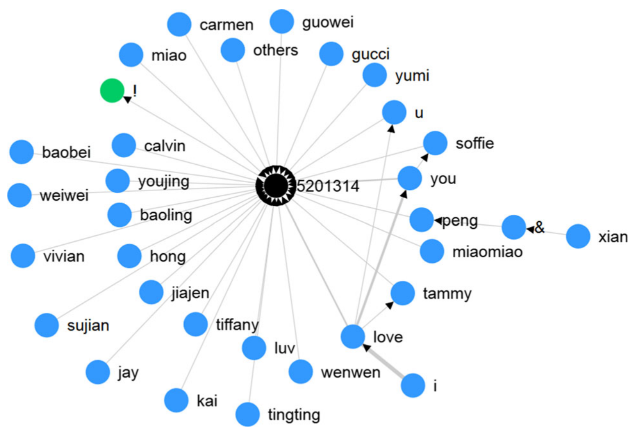


Fig. 8 One example of culture-based password substitution, where "5201314" stands for "I love you forever," and the prefixes are usually names for Chinese speaking users

pen in every month (see Table 4), e.g., januaryb13, julyo4, augustE1, novemberx, etc.

However, some substitutions are not clear to be recognized. Interestingly, we find some of these substitutions have deep roots in the users' cultures and languages. For example, "88" may be substituted for "byebye" in Chinese language because of the sound similarity. When segmenting digits from the passwords, most of them are long numeric strings. Some of the passwords we analyze are "e2345," "e234567890," etc. As far as we know, "e" is neither at the beginning of some keyboard rows, nor the initial letter in 26 letters. Actually, "e" has similar sound to "1" in Chinese, which probably is the reason why "e" is usually followed by the numeric series.

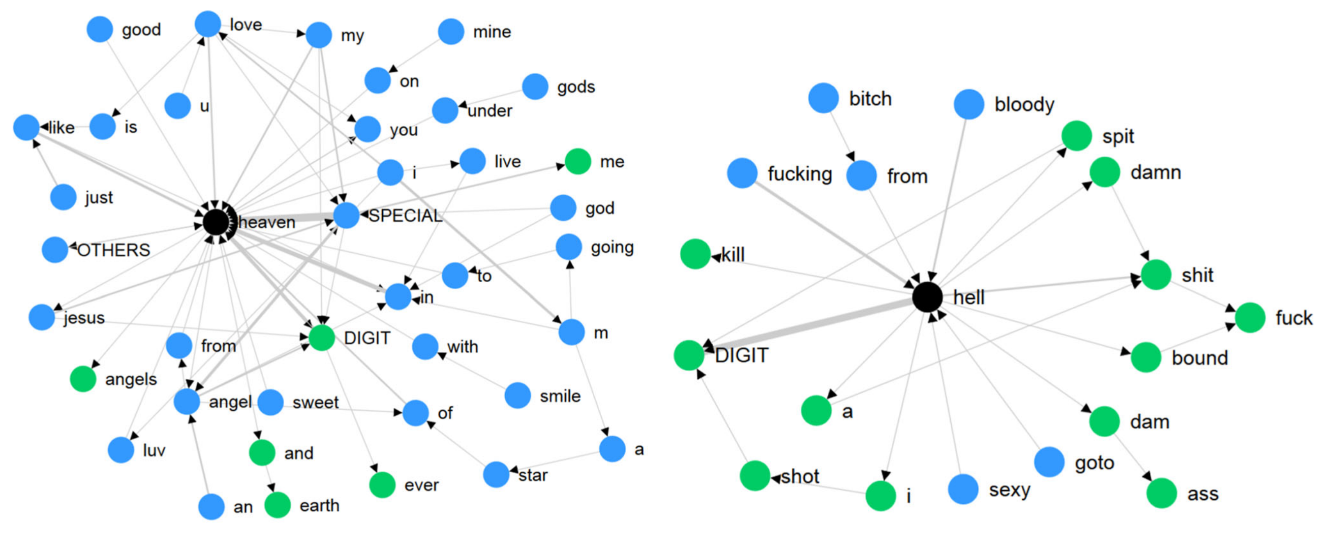
Figure 8 shows another example of substitution from real user passwords using P3G analysis. At first, it is hard to find what special meaning a substring "5201314" contains. However, from the neighboring nodes, almost all prefix nodes connecting to "5201314" are names, particularly "baobei" (which means "baby" in Chinese), "weiwei," "miaomiao," "vivian," "tiffany," etc. Based on the assumption that users most likely speak Chinese, we deduce "5201314" must mean "I love you forever" since they have similar sounds. Not surprisingly, most of the name prefixes of the substituted keyword are Chinese names as well.

The myth about choosing password using substitution (e.g., "0" for 'o') making it safer yet easy to remember may or may not be valid. By knowing the context of users' culture, geography location, language preference, etc., one may launch focused substitution attacks based on these context. On the other hand, the *diversity* of users' background may be a good thing in terms of balancing security and usability.

6.4 Contextual prefixes and postfixes

A few other questions we like to know are "are there correlations between positive and negative connotations in user passwords?", "will religious users tend to use words related to god than non-religious users?", etc. In general, do prefix and postfix exhibit different patterns for users in different context? Figure 9 shows a comparison of P3Gs using two contrasting keywords, i.e., "heaven" vs. "hell." The visualization (Fig. 10a) clearly shows that the commendatory prefix (blue nodes), e.g., "good," "love," "sweet," "like," "smile," etc., is constantly associated with passphrase "heaven." In contrast, Fig. 9b shows people constantly use bad words, such as "bitch," "fuck," "bloody," "damn," "shit," and "kill," as the prefixes and postfixes if their passwords contain "hell."

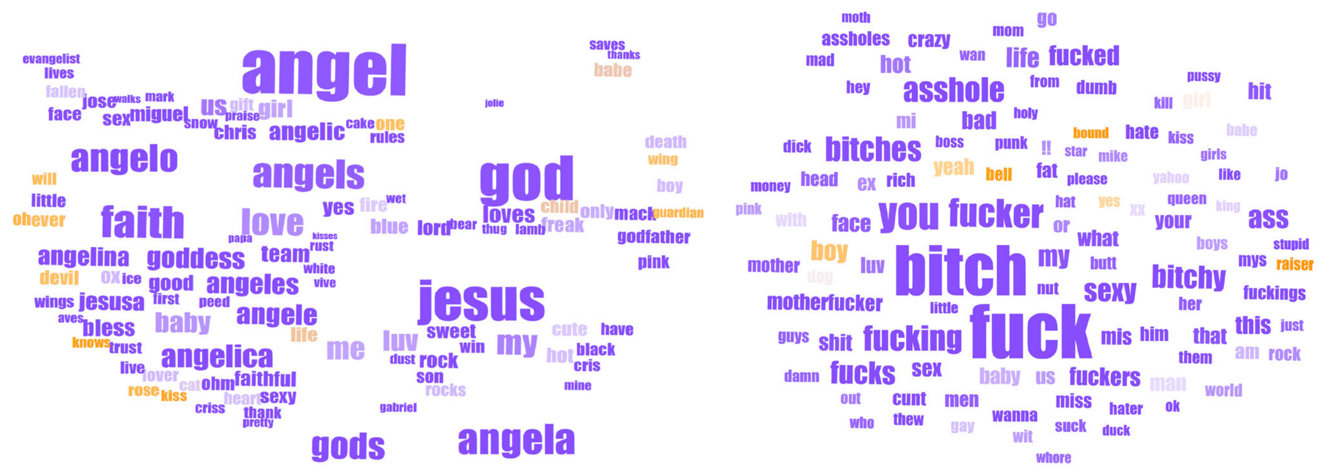
In the P3C views demonstrated in Fig. 10, we choose a few positive connotative words found in user passwords that are related to faith and religion such as "god," "faith," "jesus." In addition, we choose a few negative connotative words. Then, we generate P3C views to compare if the prefixes and postfixes of positive and negative connotative keywords reveal



(a) "heaven" as the keyword

(b) "hell" as the keyword

Fig. 9 P3Gs comparing the prefixes (blue) and postfixes (green) of contrasting keywords (black). Commendatory phrases such as "good," "love," "sweet," and "smile" are associated with the keyword "heaven." Bad words such as "bloody," "damn" and "kill" are associated with the keyword "hell" (color figure online)



(a) Prefixes/postfixes under positive connotative (religious) context

(b) Prefixes/postfixes under negative connotative context

Fig. 10 P3C views comparing the prefixes (orange) / postfixes (purple) in user passwords under contrasting context (color figure online)

distinct patterns. Dark orange indicates high probability of being prefix, while dark purple indicates high probabilities of being postfix. The result confirms our hypothesis and is in line with the results from Fig. 9. The top prefixes and postfixes for positive connotative words are highly likely to be positive as well (Fig. 10a), while the top prefixes and postfixes for negative connotative words tend to be negative too (Fig. 10b). There is, however, one more subtle pattern that we discover. Although both cases have "i" and "you" connected as prefix and postfix, the size of "i" in positive connotative case is larger than "you," while it is the opposite for negative connotative case. In accordance with Fig. 10a, the subjective words, such as "i," "my," "me," more frequently connect to the words that relate to the religious words. A plausible explanation could be that in Bible, Jesus Christ keeps conveying atonement and salvation in the subjective side to Christians. As Emil Brunner [7] said, "It is only in this subjective experience, in faith, that the Atonement becomes real."

The results suggest that the prefix and postfix are highly correlated with the context of keywords in user passwords. Hence, if an attacker knows the context of a possible word used in user passwords (as from certain groups, e.g., religion of users), he/she can minimize the guess range of prefix and postfix forming a password. The finding may be useful to customize stronger policies through specific cases based on the users' types.

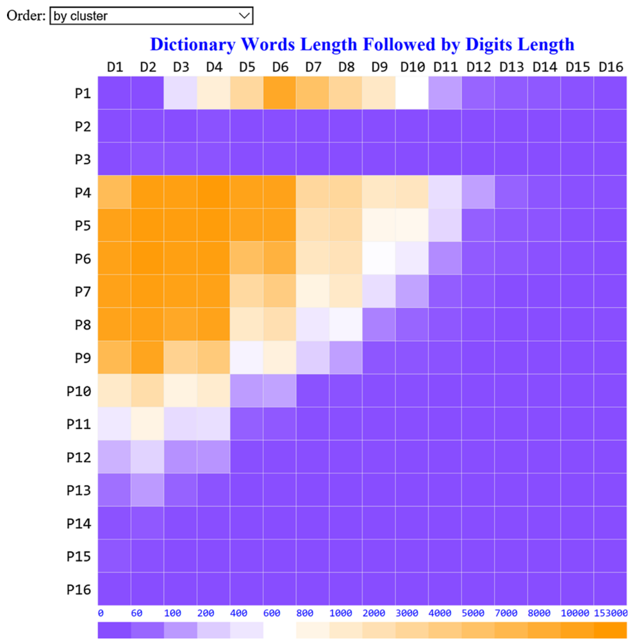


Fig. 11 The length distribution of keywords (y-axis) vs. their numeric postfix (x-axis). Majority is from P5 to P8 and from D2 to D6 with long tail distribution

6.5 Distribution of numerical postfixes

One of the dominant patterns in user passwords is that users prefer to set numbers at the end of passwords. There are 1,733,838 out of 11,484,552 unique passwords, or 15.1%, that have the format of "keyword + number." Among them, 313,477 passwords, or about 18.1%, have the form "name + date." The numbers are limited to the size of name dictionary we used and we only look for one specific combination of date format. We find there are several common phenomena. First, many users compose their passwords by using first name or last name plus date. The dates are usually their birthdays [5,22]. This is probably the most common pattern when users set their passwords. Second, some users just add digits at the end of names or some other meaningful words, for instance, "password1," "red123," or "animal123456789." Some words may represent the event that people experienced at particular time, e.g., "adventure99," "affairs07," "airassault69."

Figure 11 shows the P3R view illustrating the length distribution of passwords that involve numeric postfixes. In the P3R view, the y-axis (P1 to P16) represents the length of keywords, and the x-axis (D1 to D16) denotes the number of digits that follow the keywords. Blocks of dark brown areas, which edges are from P5 to P8 and from D2 to D6, represent higher frequency of usage. Intuitively, the longer the word, the shorter of digit postfix, and vice versa. The visualization suggests the total lengths of the majority of passwords in this "word + digits" mode are from 6 to 12. There are diminishing long tails with increasing digit lengths. Another observation is that for "word + digits" passwords, the minimum length of words starts at 4. One exception is P1, in which it is also common to have just one letter at the beginning, then followed by a series of digits (peaked at the length of 6). A popular date format `mmddyy` may be contributed to the digit length of 6.

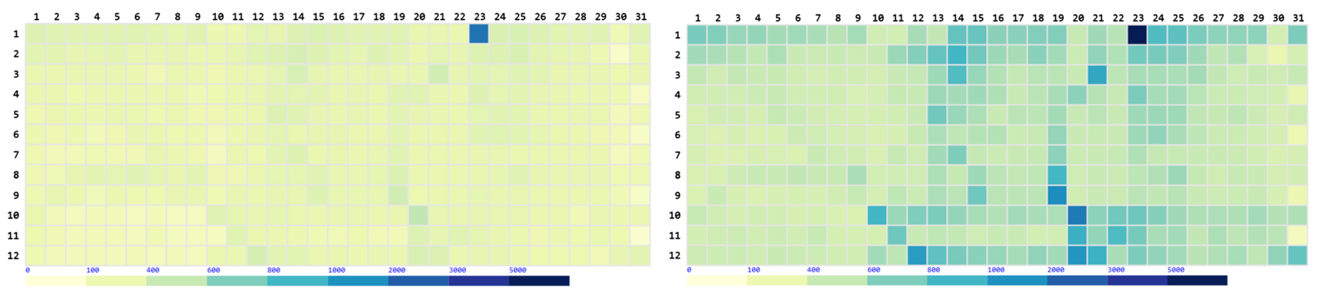
Although passwords with digits are common and easy for users to remember while satisfying the length requirement of passwords, the revealed patterns suggest such practice is not secure as the time complexity of brute-force attacks can be significantly reduced by following the demonstrated "word + digits" distribution in Fig. 11.

6.6 Patterns of date postfixes of names

Figure 12 uses P3R views to analyze one specific type of postfix, i.e., dates. As discussed in the previous section, here we examine one special case of "word + digits" distribution, i.e., "name + dates," which is also a very common pattern in user passwords. In particular, Fig. 12a examines the distribution patterns of date postfix after people's names and Fig. 12b examines the date distribution after location/place names. There exist distinctive patterns in both cases. First, for people's names, the date postfixes are uniformly distributed. This implies that people tend to use birthdays after their names since different people have different birthdays. However, for locations such as city names, we do not observe equal distribution as in the people's name case. We acknowledge that we do not look at all combinations of date formats but only a small subset of possible dates. It is also possible that some digits could be keyboard patterns or even zip codes. With further investigation on the concentration of areas, we find that some of the clusters are around special days such as holidays. For example, there is one cluster at the end of December, one at the beginning of January, and one near Thanksgiving day. In addition, dates of places are more often in the middle of months than at the beginning or end of months. Compared to people's names, dates associated with locations may have special meanings for the users such as traveling events like vacations, many of which happen during holidays. The results suggest the prefixes/postfixes such as dates exhibit distinguishable patterns depending on the types of names.

6.7 Prefix and postfix of locations and social roles

We further study the prefix and postfix patterns for different categories of keywords in user passwords. The first category we look at is the location/place names. Figure 13 shows a P3G of city names (Detroit and London). It can be observed that directional words (e.g., "east," "west," "south," and "southwest") and emotional words (e.g., "love," "luv," "hate," and



(a) Date postfixes following people's names. **(b)** Date postfixes following place's (city) names.

Fig. 12 P3R views of distinctive distribution patterns of date postfixes in user passwords. Dates are represented by months (y-axis) and days (x-axis)

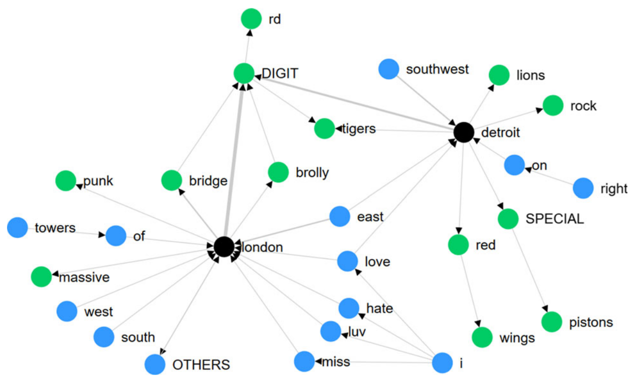
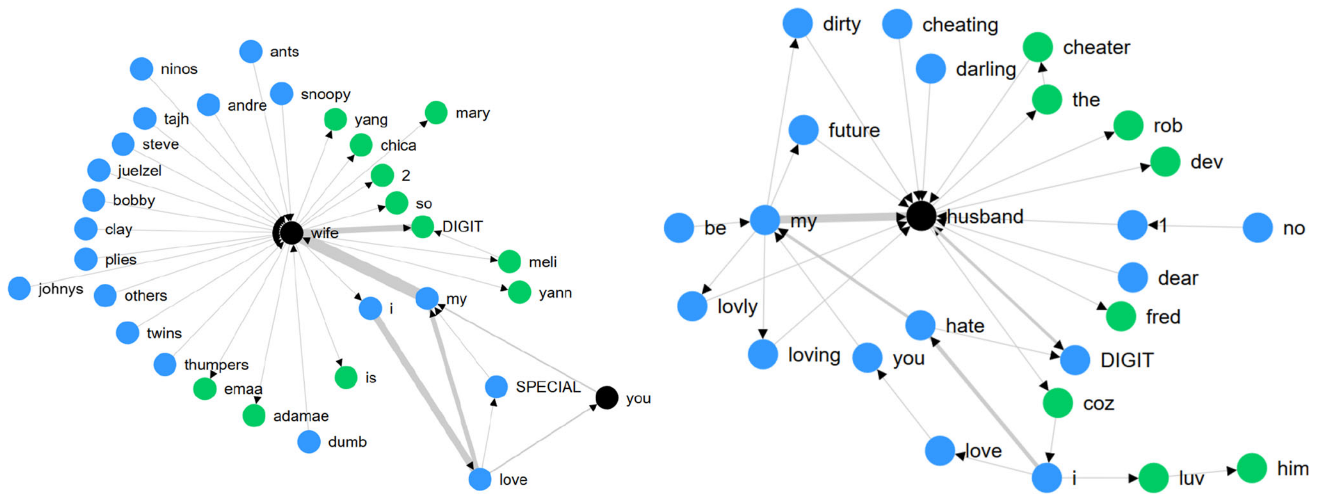


Fig. 13 A P3G graph showing the prefixes and postfixes of one type of location words (city names). Directional words (e.g., "southwest") and emotional words (e.g., "love") are the common prefixes while symbols such as sport teams are usually the postfixes

"miss") are the common prefixes of city names. The city symbols such as landmarks, local organizations, sport teams, and digits are usually the postfixes of city names, for example, "bridge" as the postfix of "london," "lions" and "pistons" as the postfix of "detroit." We do notice there are many more diversities in prefix/postfix if a location word is a country name.

The second category we examine is people's social roles, e.g., family and occupation. In family roles, husband, wife, son, and daughter are observed. Figure 14 compares P3Gs of husbands and wives. Most of the prefixes and postfixes of the social roles are feelings and emotions. Specifically, in Fig. 14b, "love" (and variations of "love" such as "luv," "lovely" and "loving"), "dear," "hate" are frequent prefixes connecting to "husband." Interestingly, we observe there are unique prefix and postfix for "husband" such as "dirty," "cheating" and "cheater." In comparison, the prefixes and postfixes for "wife" (Fig. 14a) share common characteristics such as "love." However, the predominant prefixes for "wife" include men's names. The results suggest that the users' psychological associations about their spouses are possibly conveyed when creating passwords.



(a) Wife **(b)** Husband

Fig. 14 P3Gs showing prefix and postfix of social roles (husband vs. wife)

7 Conclusion

With the increasing length and complexity of passwords, understanding user passwords is important for designing secure systems. In this paper, we investigated the prefix and postfix patterns of user passwords. Since dividing passwords is not a trivial task, we designed novel hierarchical segmentation algorithms using both dynamic programming and optimization techniques. We analyzed the impact of each layer on the trade-off between scalability and granularity of password prefix and postfix (P3) analysis. To facilitate P3 analysis, we proposed novel password prefix postfix graph (P3G) construction algorithms. P3Gs intuitively encode the relationships between prefix and postfix of user passwords. In addition, we developed a P3 visualization tool that integrates views of P3G, P3C, and P3R. Through case study over real-world user passwords, we were able to identify a wide array of distinctive patterns of prefix and postfix of various categories of words such as people, locations, roles, digits, dates, religions, culture-specific substitutions. The results suggest strong correlations between prefix/postfix and the context in user passwords. The findings provide useful insights for security practitioners to better understand user passwords and ultimately design a stronger system by considering human factors. Our future work will apply further graph theoretical analysis to the P3G graphs.

Compliance with ethical standards

Conflict of Interest Authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Aydin, K., Bateni, M., Mirrokni, V.: Distributed balanced partitioning via linear embedding. In: Ninth ACM International Conference on Web Search and Data Mining, pp. 387–396. San Francisco CA (2016)
2. Bentley, R.A., Hahn, M.W., Shennan, S.J.: Random drift and culture change. Proc. R. Soc. London B: Biol. Sci. **271**(1547), 1443–1450 (2004)
3. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: IEEE Symposium on Security and Privacy (SP), pp. 538–552. San Francisco CA (2012)
4. Bonneau, J., Preibusch, S., Anderson, R.: A birthday present every eleven wallets? The security of customer-chosen banking pins. Financ. Cryptogr. Data Secur. **7397**, 25–40 (2012a)
5. Bonneau, J., Preibusch, S., Anderson, R.: A birthday present every eleven wallets? the security of customer-chosen banking pins. In: Proceedings of the 16th International Conference on Financial Cryptography and Data Security, Bonaire, pp 25–40 (2012b)
6. Brown, A.S., Bracken, E., Zoccoli, S., Douglas, K.: Generating and remembering passwords. Appl. Cognit. Psychol. **18**(6), 641–651 (2004)
7. Brunner, E., Wyon, O.: The Mediator: A Study of the Central Doctrine of the Christian Faith, vol. 3. James Clarke & Co, Plainview (1934)
8. i Cancho, R.F., Solé, R.V.: The small world of human language. Proc. R. Soc. London B: Biol. Sci. **268**(1482), 2261–2265 (2001)
9. Hc, Chou, Hc, Lee, Cw, Hsueh, Fp, Lai: Password cracking based on special keyboard patterns. Int. J. Innov. Comput. Inf. Control **8**(1A), 387–402 (2012)
10. Cubrilovic, N.: Rockyou hack: From bad to worse. https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/ (2009)
11. Davis, D., Monrose, F., Reiter, M.K.: On user choice in graphical password schemes. In: Proceedings of the 13th conference on USENIX Security Symposium (SSYM'04), pp 151–164. San Diego, CA (2004)
12. Dell'Amico, M., Michiardi, P., Roudier, Y.: Password strength: An empirical analysis. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), pp 1–9. San Diego, CA (2010)
13. Egelman, S., Sotirakopoulos, A., Muslukhov, I., Beznosov, K., Herley, C.: Does my password go up to eleven?: the impact of password meters on password selection. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI), pp 2379–2388. Paris, France (2013)
14. Herley, C., van Oorschot, P.: A research agenda acknowledging the persistence of passwords. IEEE Sec. Priv. **10**(1), 28–36 (2012)
15. Jakobsson, M., Dhiman, M.: Proceedings of the 7th usenix conference on hot topics in security (hotsec'12). In: The benefits of understanding passwords, p 10. Bellevue, WA (2012)
16. Li, Y., Wang, H., Sun, K.: A study of personal information in human-chosen passwords and its security implications. In: The 35th Annual IEEE International Conference on Computer Communications (INFOCOM). San Francisco, CA (2016)
17. Rao, A., Jha, B., Kini, G.: Effect of grammar on security of long passwords. In: Proceedings of the third ACM conference on Data and application security and privacy, pp 317–324. San Antonio, Texas (2013)
18. Schweitzer, D., Boleng, J., Hughes, C., Murphy, L.: Visualizing keyboard pattern passwords. In: 6th International Workshop on Visualization for Cyber Security (VizSec'09), pp. 69–73. Atlantic City, NJ (2009)
19. Segaran, T., Hammerbacher, J.: Beautiful Data: The Stories Behind Elegant Data Solutions, O'Reilly Media, p 386. ISBN 9780596157111 (2009)
20. Shay, R., Komanduri, S., Durity, A.L., Huh, P.S., Mazurek, M.L., Segreti, S.M., Ur, B., Bauer, L., Christin, N., Cranor, L.F.: Can long passwords be secure and usable? In: Proceedings of the 32nd annual ACM conference on Human factors in computing systems, pp 2927–2936. Toronto, Canada (2014)
21. Shi, L., Liao, Q., Tong, H., Hu, Y., Zhao, Y., Lin, C.: Hierarchical focus+context heterogeneous network visualization. In: Proceedings of the IEEE Pacific Visualization Symposium (PacificVis), pp 89–96. Yokohama, Japan (2014)
22. Veras, R., Thorpe, J., Collins, C.: Visualizing semantics in passwords: The role of dates. In: Proceedings of the Ninth International Symposium on Visualization for Cyber Security (VizSec'12)), pp 88–95. Seattle, WA (2012)
23. Veras, R., Collins, C., Thorpe, J.: On semantic patterns of passwords and their security impact. In: Network and Distributed System Security (NDSS) Symposium. San Diego, CA (2014)
24. Wang, D., Zhang, Z., Wang, P., Yan, J., Huang, X.: Targeted online password guessing: An underestimated threat. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), pp 1242–1254. Vienna, Austria (2016)
25. Weir, M., Aggawal, S., Collins, M., Stern, H.: Testing metrics for password creation policies by attacking large sets of revealed pass-

words. In: Proceedings of the 17th ACM conference on Computer and communications security (CCS '10), pp 162–175. Chicago, IL (2010)
26. Yang, W., Li, N., Chowdhury, O., Xiong, A., Proctor, R.W.: An empirical study of mnemonic sentence-based password generation strategies. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), pp 1216–1229. Vienna, Austria (2016)
27. Yeganova, L., Smith, L., Wilbur, W.J.: Identification of related gene/protein names based on an hmm of name variations. Comput. Biol. Chem. **28**(2), 97–107 (2004)
28. Yu, X., Liao, Q.: User password repetitive patterns analysis and visualization. Inf. Comput. Secur. **24**(1), 93–115 (2016)
29. Zheng, Z., Cheng, H., Zhang, Z., Zhao, Y., Wang, P.: An alternative method for understanding user-chosen passwords. Secur. Commun. Netw. **2018**, 6160125 (2018)