A Typo-tolerant Password Authentication Scheme with Targeted Error Correction

Xin Chen, Xinyi Huang, Yi Mu
Fujian Provincial Key Laboratory of Network Security and Cryptology
College of Mathematics and Informatics
Fujian Normal University, Fuzhou, China
Email:xyhuang@fjnu.edu.cn

Ding Wang
School of EECS, Peking University,
Beijing 100871, China
Email: wangdingg@pku.edu.cn

***Abstract*—Password-based authentication is used in almost every computer system. However, users might not always type their passwords correctly. In order to improve the accuracy of password authentication without reducing security, Chatterjee et al. (IEEE S&P'16) and Guan et al. (SecureComm'17) proposed typo-tolerant schemes, respectively. However, these schemes do not consider the impact of personal information on password usages when generating candidate sets, while Wang et al. (NDSS'18) show that 36.95%∼51.43% of users employ their personal information to generate passwords. In this paper, we propose a typo-tolerant password authentication scheme with targeted error correction. Our scheme focuses on two aspects: fuzzy judgment and error correction. During the process of password authentication, we first use fuzzy judgment to determine whether the input password contains personal information, and then correct the password according to the result of the fuzzy judgment. The error correction is divided into two types: with personal information and without personal information. Our experimental results show that our solution, when correcting the password entered by the user, is generally able to achieve higher accuracy. When considering the four main errors occurring in the mobile dataset and the general dataset, the average correct rate is 96.29%. The analysis illustrates that the average success proportion of targeted error correction (68.56%) is two times higher than the average success proportion of non-targeted error correction (31.44%), indicating the feasibility of our scheme.**

1. Introduction

The methods of user authentication mainly include static password, dynamic password, smart card, USB key, digital certificate and biometric technology. Among these, password-based schemes are most widely used because of their low cost and ease of use. Due to the limitations of human memory, it is often difficult to remember long and random passwords. Users then tend to choose a short and simple password or reuse the password, which would be vulnerable to password guessing attacks. Password strength meters (PSM) [1] are proposed to thwart guessing attacks, by promptly feeding back the strength information of the password chosen by the user. Some PSMs also provide suggestions of how to modify the password towards a more complex password and a higher level of security.

It is difficult for users to remember complex passwords [2], [3], [4], [5]. A series of studies have shown that complex passwords chosen by the user are not only inconvenient for the user to remember, but also easy to mistype [5], [6], [7]. However, these studies did not explore the regular user input error types. In 2016, Chatterjee et al. [8] experimented on the Amazon Mechanical Turk platform, and their experimental results showed that about 10% of user login attempts failed because of some easily corrected errors (i.e., capitalization errors). In order to explore the concrete types of password typos, Chatterjee et al. [8] experimented on the Amazon Mechanical Turk platform, and they found that the main typos of users on the PC side and mobile devices include proximity errors, swc-all, swc-first, rm-first, rm-last and n2s-last. These typos can be modified with simple functions. After correcting these easily-correctable typos, the password acceptance rate can be increased by 3%, that is, an additional 3% of users will successfully log in.

Additionally, a few web services seem to intentionally allow a small set of typos [9], [10], [11], [12]. For instance, Facebook allows initial case errors when verifying passwords (assuming a password begins with a letter) [9], [12].

1.1. Related Work

Passwords have been widely used for common authentication. There is no any authentication method which can completely replace password-based authentication, as passwords can be easily remembered and have low cost in comparison with other authentication methods. In the simplest case, the traditional password authentication process generally includes two phases: registration phase and verification phase. During the registration phase, the user chooses his or her user name and suitable password and sends them to the server, which stores these information in plaintext or in a hashed form. In the verification phase, the user submits the input password to the authentication server. The server calculates the submitted password and matches the calculated result with the correct value stored in locate. If

the two values are equal, the verification passes; otherwise, the verification fails.

Mistype by users in the verification process has become a common problem. There are two kinds of solutions: server side solution and client side solution.

1.1.1. Server side solution.

In 2004, Dodis et al. [13] propose two primitives: a fuzzy extractor and a secure sketch. In theory, they can correct password errors caused by typos in the server side, but the actual implementation of fuzzy extractor is not satisfying. In 2006, Mehler and Skiena [14] used a single password correction hash function to hash two different strings generated by a single data entry error to the same key value. Their method can only correct transposition and substitution errors. Both of these approaches would reduce the security against online guessing attacks.

In 2012, Jakobsson and Akavipat [15] suggested dictionary-checking-based error correction based on the work of Shay et al. [3], which they call fastwords. Fastwords are a sequence of dictionary words, that is, the words in each fastword are dictionary words. The scheme replaces the password with equivalence classes-the conceptual equivalence classes and the homophonic equivalence classes. The former classes may include different tenses of a given verb and synonyms of a targeted word. The latter classes are similar to mutual conversion between homonyms. In the registration phase, the server accepts the required strong security credential, performs equivalence class transformations, and then stores them in the backend. During the verification phase, after the server hashes the security credentials submitted by the user, the result was compared to all stored values in the server. Although the new method fastwords proposed in [15] can implement fuzzy authentication and improve the success rate of user authentication, the backend needs to store the salted hash value of all acceptable variants. Compared with traditional password verification, the server needs more storage space.

In contrast to the above works, Chatterjee et al. [8] provided the first way to handle typo-tolerant password authentication for arbitrary user-selected passwords. In their scheme, several common types of typos can be modified with simple functions. These simple functions are defined as error correction functions. During the verification phase of the password authentication process, the server side firstly use the password error correction functions to modify the input password and generate a substitute set. Then it verifies the hash value of each element in the password candidate set instead of just the hash value of the password currently entered by the user. Although the error correction functions consider five kinds of errors, the scheme does not solve the most common error called proximity errors (which means hitting an adjacent key regardless of the intended keyboard status).

1.1.2. Client side solution.

Based on Chatterjee et al.'s work [8], Guan et al. [16] propose a client-side typo-tolerant scheme, and their work differs from [8] in that they mainly study the error types of the mobile terminal and correct the newly defined proximity error, (i.e., hitting an adjacent key in the same keyboard status with the intended one). In addition, Guan et al. design a software similar to the password manager called VaultIME. During the authentication process, when the password entered by the user does not match the correct password recorded in VaultIME, VaultIME uses the error correction functions to modify the input password, and then matches the modified passwords with the correct password stored in the VaultIME. If the match is successful, the correct password is used to run the authentication with the server; Otherwise, the input password is used to run the authentication with the server. The solution allows the users to highly control and resolves the proximity error, but most users do not trust the password manager, placing the password manager on the client is not guaranteed to gain the user's trust.

Wang et al. [17] show that 36.95%∼51.43% of users employ their personal information to generate passwords, yet none of existing studies on password correction has considered this user behavior. This partially explains why the password candidate set generated by these previously mentioned schemes [8], [16] is not accurate enough. To handle this issue, we propose a typo-tolerant password authentication scheme with higher sucess rate. Our proposed solution is intended to generate a more reasonable candidate set and achieve a higher password authentication success rate. The contributions of our work are summarized as follows:

- We propose a typo-tolerant password authentication scheme based on mobile client side. Our experiment results show that when considering the four main errors occurring in the mobile dataset and the general dataset, the average successful error correction rate of our scheme can reach 96.29%. In addition, the average success rate of targeted error correction (68.56%) is much higher than the average success rate of non-targeted error correction (31.44%). This indicates that our solution is feasible and has certain advantages.
- In our scheme, in order to consider the impact of personal information on password settings, we have designed a fuzzy judgment to determine whether personal information is included in the input password. The result of fuzzy judgment provides the basis for the next error correction process.
- According to whether the password contains personal information, the error correction process of our scheme is divided into two types: with personal information (targeted error correction) and without personal information (non-targeted error correction). The design of our solution error correction process makes the generated candidate set more reasonable and achieves higher password authentication accuracy.

2. Preliminaries

Exact checker. In general, a password authentication process can be divided into two phases, i.e., registration and verification. In the registration phase, the user registers the username and password (w) with the server. The server in turn stores the salted hash value of w, which is denoted by the string s. In the verification phase, the user sends its password $\tilde{w}$ to the authentication server. The server in turn calculates the salted hash value of $\tilde{w}$ and then compares the calculated result with the correct value s stored on the server side. The checking is successful only if the input password $\tilde{w}$ and w are identical.

Relax checker. In contrast to an exact checker ExChk, a relaxed checker may return the true value for multiple strings other than w. When the user submits $\tilde{w}$, the authentication algorithm (rather than just checking w) checks a set of strings adjacent to w. This set is generated after $\tilde{w}$ is modified by some error correction functions. If any element in the set passes the exact checker ExChk, $\tilde{w}$ is accepted and the user successfully logs in.

Empirical Study of Typos on Mobile Devices. In 2016, Chatterjee et al. [8] carried out two experiments on the Amazon Mechanical Turk (MTurk) platform. One of them collected typos records of the user entering a password on the touch screen mobile device. When collecting the database, the user needs to complete the human-intelligence tasks (HITs) assigned by the web page. They need to input 10^{14} passwords in the HTML password input box within 300 seconds using the touch-screen mobile device. They analysed the collected data and ultimately derived the main typos types on the mobile device side. On the basis of [8], Guan et al. [16] further analysed the database mentioned above in [8], and finally obtained similar results as in [8]. Here, we find that the main typos on the mobile device side are prox-rs, rm-any, ins-any, swc-all, and swc-any in [16]. The four error correction functions are rep-prox-rs, rm-any, swc-all, and swc-any. We still use the four error correction functions defined in [16].

3. The Proposed Scheme

When the user enters the password, we do not know whether the password entered by the user is correct. In this case, we first roughly determine whether the input password contains the user's personal information, and then choose targeted error correction or non-targeted error correction according to the judgment result. When the input password contains personal information, targeted error correction is performed, and non-targeted error correction is also performed. When the input password does not contain personal information, only non-targeted error correction is performed. Finally, the password candidate set generated after error corrections is submitted to the authentication server, and the server verifies whether the user successfully logs in. The password candidate set includes the password entered by the user. Note that in the case of target error correction, the candidate passwords need to be input into our pre-trained model (see the Attack model) before being submitted to the server; in the case of non-targeted error correction, the generated candidate password set can be directly submitted to the authentication server. The non-targeted error correction mentioned here refers to the case where the input password does not contain the user's personal information. The detailed process of our scheme is shown in Figure 1. The overview flow of the system is shown in Algorithm 1.

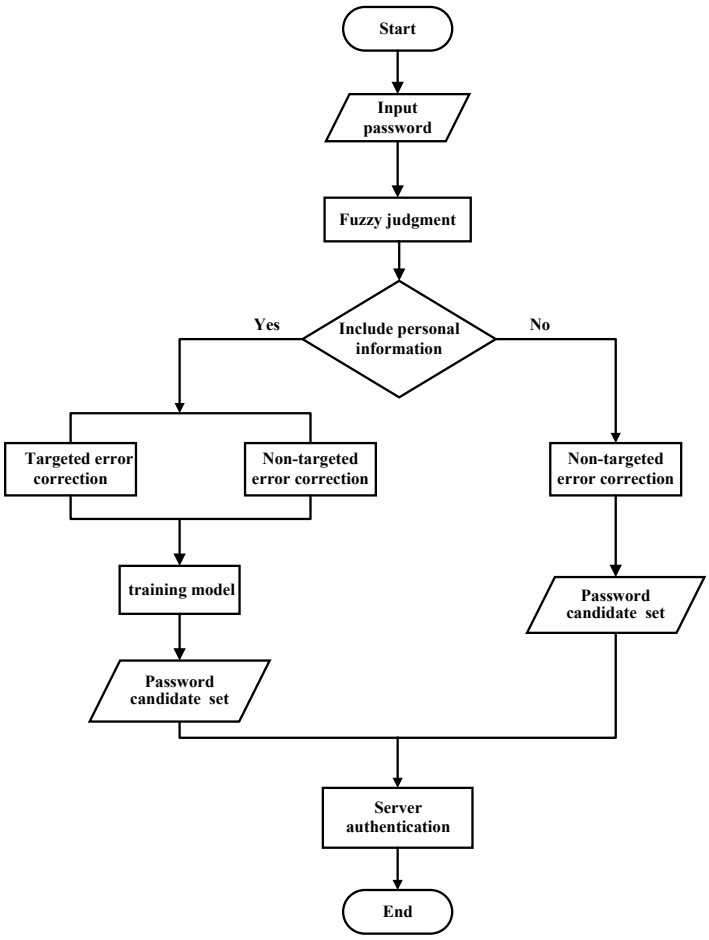

Figure 1. The flow chart of the scheme

The detail algorithms are presented in Algorithm 2, along with three sub-algorithms for modification (Algorithm 3 and 4) and training algorithm (Algorithm 5).

3.1. Fuzzy judgment

After the user enters his or her own password, we first determine whether the input password contains personal information, such as name, birthday, and phone number. This process is called fuzzy judgment. In the process, we first divide the input password into two sub-sections: the numeric part and the letter part, ignoring the special characters in the password. Then we use the relevant regular expressions to

Algorithm 1: The flow of the scheme

Input: password
Output: a candidate set

```
result = fuzzy_judgment(password);
if result contains personal information then
    result1=targeted_error_correction();
    result2=non-targeted_error_correction();
    correction_results=result1 + result2;
    probability_results=training_model(correction_results);
    canditate_set_B=sorted(probability_results);
else
    canditate_set_B=non-targeted_error_correction();
end
```

determine whether the numeric part of the password contains information about the ID number or phone number. In addition, we use the pre-generated Date Set to determine whether the numeric part of the password contains information related to the birthday. We also use the pre-generated Name Pinyin Sets to check whether the letter part of the password contains information about the name. During the process of matching, we follow the longest matching principle and save the final match result in a dictionary. According to the result of the fuzzy judgment, it is determined whether the current password is subjected to non-targeted error correction or targeted error correction. The Name Pinyin Sets and Date Set used in the fuzzy judgment process will be explained in Section 4. An example is shown in Figure 2.

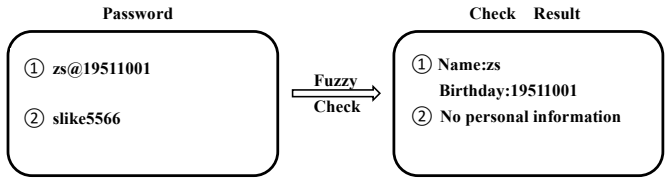

Figure 2. Fuzzy judgment process

3.2. Error-Correction

The error correction process mainly includes the following two parts:

Non-targeted error correction. When the result of the fuzzy judgment indicates that the input password does not contain personal information, we perform non-targeted error correction on the password. There are four error correction functions defined in [16] that can be used to correct the current password: rep-prox-rs converts any character in the password to its adjacent key in the same keyboard state, rm-any removes any character in the password, swc-all switches the case of all letters in the password, and swc-any switches the case of any letter in the password. We use the four error correction functions to correct the input password and

Algorithm 2: Concrete Algorithm

Input: password
Output: canditate_set_B

```
Step1. Fuzzy judgment;
split the password;
check if the letter part of the password in namesSet;
check if the numeric part of the password in birthdaySet;
check if the numeric part of the password in ID rule;
check if the numeric part of the password in phone rule;
return result;
Step2. Error correction;
if result contains element then
    Step3. modify the password using the edit distance according to result;
    calculate the editDistance between the subPassword and subPersonInfo;
    check if calculate result below the threshold;
    combine the modified results (modify1) of the subsections to get the final modified result;
    assign the result to result1;
    Step4. non-targeted_error_correction with password;
    modify (modify2) the input password with four error correction functions and obtain the result;
    insert the input password into the first of result;
    assign the result to result2;
    Step5. calculate the probability;
    enter the elements in result1 and result2 into the training model (reference [20]);
    get the matrix of trainData probability;
    use personInfo and trainData probability calculate the probability of each element in result1 and result2;
    obtain a result set with probability;
    Step6. obtain canditate_set_B;
    arrange the passwords in the result set in descending order of probability;
    assign the result to candidate_set_B;
else
    Step7. non-targeted_error_correction with password;
    modify (modify2) the input password with four error correction functions and obtain the result;
    insert the input password into the first of result;
    assign the result to candidate_set_B;
end
```

Algorithm 3: modify1

Input: result, personInfo
Output: result1

result = fuzzy_judgment(password);
if *result contains element* **then**
  if *result contains 'name'* **then**
    modify this subsection with name related information;
    calculate the editDistance between the subPassword and subPersonInfo;
    if calculated result below the threshold, replace the corresponding subPassword with the name-related subPersonInfo;
  else if *result contains 'birth* **then**
    modify this subsection with ID related information;
    calculate the editDistance between the subPassword and subPersonInfo;
    if calculated result below the threshold, replace the corresponding subPassword with the ID related subPersonInfo;
  else if *result contains 'mobile'* **then**
    modify this subsection with phone number related information;
    calculate the editDistance between the subPassword and subPersonInfo;
    if calculated result below the threshold, replace the corresponding subPassword with the phone number related subPersonInfo;
  else if *result contains 'ID'* **then**
    modify this subsection with ID related information;
    calculate the editDistance between the subPassword and subPersonInfo;
    if calculated result below the threshold, replace the corresponding subPassword with the ID related subPersonInfo;
  modify this subsection with email related information;
  calculate the editDistance between the subPassword and subPersonInfo;
  if calculated result below the threshold, replace the corresponding subPassword with the email related subPersonInfo;
  combine all the modified results of the subsections to get the final modified result;
  assign the result to result1;
else
end

Algorithm 4: modify2

Input: password
Output: a candidate set

modify any character of the password to its adjacent characters on the keyboard;
remove any character from the password;
switch the case of all letters in the password;
switch the case of any letter in the password;
store the modified results of the above steps in a list;
insert the input password into the first of the list;
obtain a candidate set;

Algorithm 5: Training model

Input: password, personInfo, trainData probability
Output: a dictionary

extract personal information (personInfo) and password;
match the password with name related information;
match the password with email related information;
match the password with phone number related information;
match the password with ID related information;
combine these matching results to get the best match result for the password;
determine the character type of the unmatched part of the password;
obtain the final parsing structure of the password;
calculate the probability of the password;
store the final result in a dictionary, the password as the key value and the probability of the password as the element;

save the passwords generated in the error correction process. These passwords, together with the input password, form the password candidate set.

Targeted error correction. After determining that the input password contains personal information, we perform targeted error correction on the current password. First, we obtain the personal information submitted by the user in advance, then analyze the password structure, and compare the parsed parts with the corresponding personal information. If they are the same, they will remain unchanged. Otherwise, the password will be modified using the editing distance. We set the edit distance to be less than or equal to 3. For example, the user password is 'xjd19830307', the user name is abbreviated as 'xjs', and for the 'xjd' part, we modify it to 'xjs' with an edit distance of 1. In particular, when we perform targeted error correction, we also perform non-targeted error correction. The passwords generated after error corrections are input into the pre-trained model (see the Attack model), and then the probabilities corresponding to the passwords are obtained. These passwords together with the input password form the final password candidate

set, and the elements in the candidate set are sorted in descending order of probability. An example is shown in Figure 3.

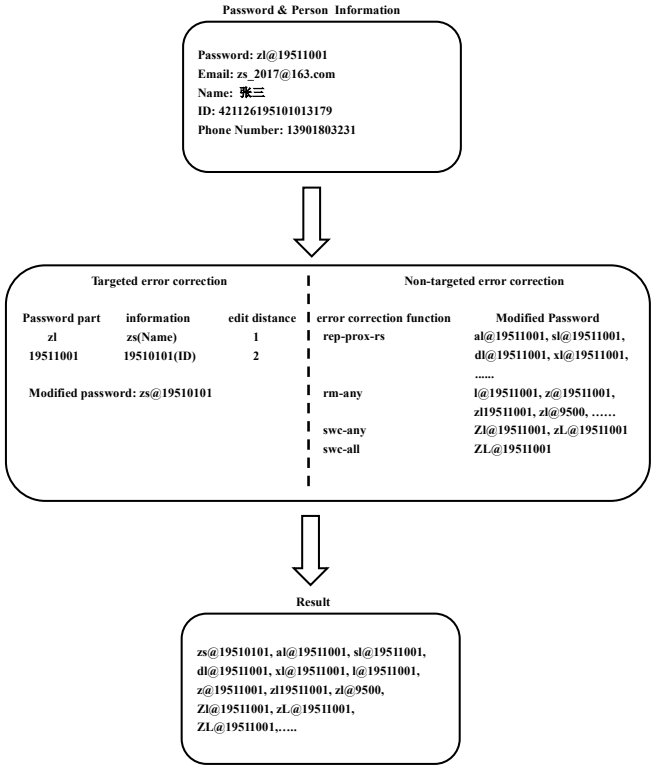

Figure 3. targeted error correction

Attack model. In 2016, Wang et al. [18] proposed a targeted online password guessing attack framework that includes seven theoretical attack models to characterize real-world attackers with seven different capabilities. The first attack model, TarGuess-I, was built with the goal of making online guesses about the user's password by using some of the user's personal information (PII) (such as name, phone number and birthday) that can be directly part of the password. It is based on the PCFG-based probability model proposed by Weir et al. [19]. In addition to the L, D, and S tags in the PCFG model [19] , Wang et al. also defined a series of PII tags based on the type of personal information in order to perceive the PII semantics in the password. In this way, a PCFG algorithm that is PII-aware would be generated during the training phase of the model. Using the TarGuess-I guess password, the TarGuess-I attack success rate was 37.11%∼73.33% higher than the Personal-PCFG [20] within $10\sim10^3$ guesses. In particular, TarGuess-I is highly adaptive. More details about this algorithm can be seen in the literature [18].

4. Experiment

In this section, we conduct experiments on our password correcting system. Using the experimental results to verify the feasibility of our solution and highlight the advantages of our program.

4.1. Data Setup

In recent years, a large number of website user databases have been leaked, which have accumulated about 276 websites and 4.97 billion accounts. Because our work mainly focuses on Chinese user password correction, we select the leaked 12306 dataset which contains relatively more personal informations to conduct our experiments. During our research, we strictly followed the ethical practice and never used the leaked accounts for reasons other than conducting the overall statistical observation and research of passwords.
Generation of Name Pinyin Sets. We generate the sets in advance, and use them to initially determine whether the user's password contains information related to the user's name in the fuzzy judgment stage. Since the name contained in the password does not emphasize the tone, we use a Chinese character corresponding to one of the pinyin according to the online Xinhua Dictionary to generate the name pinyin set. Then we combine the elements of the Chinese character set with the single surname of the hundred family names to generate Chinese character strings of length 2 and 3, respectively. Similarly, the elements of the Chinese character set and the multiple surnames of the hundred family names are combined into lengths of 3 and 4 respectively. Finally four Chinese character name sets are generated. We convert the elements in the four Chinese character name sets into Pinyin and transform them. For example, the first column of the txt file is the full name, the second column is the first letter of the first and last name, etc. Then, according to the length of these strings, the four Chinese character name sets are divided into multiple subsets, and the deduplication operation is performed in the process of dividing the subset, thereby reducing the search time in the process of fuzzy judgment. After the Chinese character is converted to pinyin, the pinyin does not correspond to a Chinese character. For example, since the tone is not emphasized, when we use Chinese characters to turn pinyin, "fei" does not only represent "霏", and also represents "菲", "妃", etc. Similarly, "wangfei" does not only represent "王菲", "lili" does not only represent "李莉" and so on. We also recombine the first and last names in Pinyin and add them to the document. For example, "cl" and "chenl" will be generated after "chenli" is operated. It should be noted that the letters in these collections are all lowercase. When making fuzzy judgments, uppercase needs to be considered.
Generation of Date Set. We pre-generated a document with a randomly generated legal date between 1900-2018. Then we recombine the year, month, and day of these dates to generate a new document, such as "19920312" to generate "199203", "1992" and so on. This new document is used to

TABLE 1. EXPERIMENTAL RESULTS

Error type	Total data set	Successful corrections	Rate1	Targeted corrections	Rate2	Non-targeted corrections	Rate3
prox-rs	57588	56783	98.60%	36049	63.49%	20734	36.51%
ins-any	29115	25697	88.26%	15903	61.89%	9794	38.11%
swc-all	21515	21343	99.20%	15181	71.13%	6162	28.87%
swc-any	20613	20373	98.84%	14612	71.72%	5761	28.28%
random	128834	36768	28.54%	27412	74.55%	9356	25.45%

determine whether the password contains information about the birthday in the fuzzy judgment process.

4.2. Experimental results

At the time of the experiment, our primary work is to generate the wrong password sets before running the program. In [16], Guan et al. pointed out that in both mobile and general datasets, the four typos (prox-rs, ins-any, swc-all, swc-any) can be modified. The proportion of these four types of errors occurred in mobile dataset and general dataset are 21.4%, 10.8%, 8.0%, 7.6%, respectively. According to the proportional situation, we randomly divide our 12306 data set into four experimental data sets, and the data in the four data sets are different. The correct passwords in the four sets are modified by one of the four error functions mentioned above, and we randomly select one from the modified result of each correct password to add to the wrong password set corresponding to the error function. The error function used by each collection is different. For example, modifying the password 'slike5566' with the prox-rs error function will generate a series of passwords, and we randomly select one of these passwords to add to the error password set corresponding to the prox-rs error function.

For each time we input an error password set and the personal information set corresponding to the password set into the program algorithm, execute the program, and the final result is shown in Table 1. From Table 1, we can see that the average probability of successful correction of the wrong password set generated by the four error functions is 96.29%, which shows that our scheme can significantly correct these four types of errors. In particular, our scheme proposes targeted error correction. From the data in the last four columns of the table, we can find that the probability of targeted error correction is generally higher than that of non-targeted error correction, which indicates that the targeted error correction proposed by our scheme is feasible and effective, and obviously improves the error correction success rate. This also indicates that most of the 12306 users' passwords contain personal information.

In addition to using the four error functions to generate the wrong password set, we additionally consider a situation where we randomly modify one bit for each password in the 12306 data set, and these modified passwords form the wrong password set. The wrong password set and its corresponding personal information set are entered into the program. The result in this case is shown in the last row of Table 1. We can find that the success error correction rate of our scheme is significantly reduced under the random condition. But even in this case, the targeted error correction still maintains its advantage.

5. Conclusion

In this paper, we proposed a client-based typo-tolerant password authentication scheme to improve the user's authentication success rate. In our scheme, we first made a fuzzy judgment on the input password. According to the result of the fuzzy judgment, the scheme then determines whether to perform targeted error correction or non-targeted error correction on the input password. We conducted experiments to evaluate the feasibility of our solution. The experimental results show that our scheme can reach a higher correct rate of error correction, and in the five cases we consider, the targeted error correction plays an important role. We find that the targeted error correction success rate is generally higher than non-targeted error correction success rate.

At present, we mainly use the leaked 12306 data set. Although the data set contains relatively sufficient personal information, the amount of data is not large enough. Our follow-up work will consider larger data sets, and on this basis, we further optimize the fuzzy judgment process and the training model.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61872087, 61822202, 61872089, 61802006).

References

[1] D. Wang, D. He, H. Cheng, and P. Wang, "fuzzyPSM: A New Password Strength Meter Using Fuzzy Probabilistic Context-Free Grammars," in *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016, Toulouse, France, June 28 - July 1, 2016*, 2016, pp. 595–606. [Online]. Available: https://doi.org/10.1109/DSN.2016.60

[2] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "How Does Your Password Measure Up? The Effect of Strength Meters on Password Creation," in *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, 2012, pp. 65–80. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/ur

[3] R. Shay, P. G. Kelley, S. Komanduri, M. L. Mazurek, B. Ur, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "Correct horse battery staple: exploring the usability of system-assigned passphrases," in *Symposium On Usable Privacy and Security, SOUPS '12, Washington, DC, USA - July 11 - 13, 2012*, 2012, p. 7. [Online]. Available: https://doi.org/10.1145/2335356.2335366

[4] J. Bonneau and S. E. Schechter, "Towards Reliable Storage of 56-bit Secrets in Human Memory," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, 2014, pp. 607–623. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/bonneau

[5] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, "Can long passwords be secure and usable?" in *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*, 2014, pp. 2927–2936. [Online]. Available: https://doi.org/10.1145/2556288.2557377

[6] M. Keith, B. Shao, and P. J. Steinbart, "A Behavioral Analysis of Passphrase Design and Effectiveness," *J. AIS*, vol. 10, no. 2, p. 2, 2009. [Online]. Available: http://aisel.aisnet.org/jais/vol10/iss2/2

[7] ——, "The usability of passphrases for authentication: An empirical field study," *International Journal of Man-Machine Studies*, vol. 65, no. 1, pp. 17–28, 2007. [Online]. Available: https://doi.org/10.1016/j.ijhcs.2006.08.005

[8] R. Chatterjee, A. Athayle, D. Akhawe, A. Juels, and T. Ristenpart, "pASSWORD tYPOS and How to Correct Them Securely," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 799–818. [Online]. Available: https://doi.org/10.1109/SP.2016.53

[9] "Facebook passwords are not case sensitive," http://www.zdnet.com/article/facebook-passwords-are-not-case-sensitive-update/, 2011-11-12.

[10] "Is Vanguard making it too easy for cybercriminals to access your account?" http://www.thestreet.com/story/13213265/4/is-vanguard-making-it-too-easy-for-cybercriminals-to-access-your-account.html?startIndex=0, 2015-11-06.

[11] S. Antilla, "Vanguard group fires whistleblower who told TheStreet about flaws in customer security," *TheStreet.*, 2015-09-18.

[12] A. Muffet, "Facebook: Password hashing and authentication." Real World Crypto, 2015.

[13] Y. Dodis, L. Reyzin, and A. D. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, 2004, pp. 523–540. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_31

[14] A. Mehler and S. Skiena, "Improving Usability Through Password-Corrective Hashing," in *String Processing and Information Retrieval, 13th International Conference, SPIRE 2006, Glasgow, UK, October 11-13, 2006, Proceedings*, 2006, pp. 193–204. [Online]. Available: https://doi.org/10.1007/11880561_16

[15] Jakobsson, Markus, and R. Akavipat, "Rethinking passwords to adapt to constrained keyboards," *Proc. IEEE MoST*, pp. 1–11, 2012.

[16] L. Guan, S. Farhang, Y. Pu, P. Guo, J. Grossklags, and P. Liu, "VaultIME: Regaining User Control for Password Managers through Auto-correction," *ICST Trans. Security Safety*, vol. 4, no. 14, p. e4, 2018. [Online]. Available: https://doi.org/10.4108/eai.15-5-2018.154772

[17] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "A Security Analysis of Honeywords," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*, 2018. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_02B-2_Wang_paper.pdf

[18] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted Online Password Guessing: An Underestimated Threat," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 1242–1254. [Online]. Available: https://doi.org/10.1145/2976749.2978339

[19] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password Cracking Using Probabilistic Context-Free Grammars," in *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*, 2009, pp. 391–405. [Online]. Available: https://doi.org/10.1109/SP.2009.8

[20] Y. Li, H. Wang, and K. Sun, "A study of personal information in human-chosen passwords and its security implications," in *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, 2016, pp. 1–9. [Online]. Available: https://doi.org/10.1109/INFOCOM.2016.7524583