International Journal of Safety and Security Engineering
Vol. 10, No. 5, October, 2020, pp. 713-720
Journal homepage: http://iieta.org/journals/ijsse

An Efficient Security Model for Password Generation and Time Complexity Analysis for Cracking the Password

Bathula Prasanna Kumar*, Edara Srinivasa Reddy

Computer Science and Engineering, Acharya Nagarjuna University, Guntur 522510, AP, India

Corresponding Author Email: bathulap547@gmail.com

https://doi.org/10.18280/ijsse.100517

Received: 18 August 2020
Accepted: 6 October 2020

Keywords:
password generation, password cracking, data security, unauthorized users, authorized users, time complexity

ABSTRACT

Passwords tend to be one of the most popular approaches to protect operating systems and user's data also. Most businesses rely on password protection schemes, and secure passwords are incredibly necessary to them. The proposed model typically aims to impose protection by forcing users to obey protocols to build passwords. For user protection, password has become a prevailing method in terms of exposure to scarce tools. The main problem with password is its consistency or power, i.e. how simple (or how difficult) a third person can be "assumed" to enter the tool that you use while claiming to be you. In operating systems, text-based passwords remain the primary form of authentication, following major improvements in attackers' skills in breaking passwords. The proposed Random Character Utilization with Hashing (RCUH) is used for generation of new passwords by considering user parameters. The proposed model introduces a new framework to design a password by considering nearly 10 parameters from the user and also analyze the time for cracking the generated password to provide the system strength. The proposed model aims to generate an efficient security model for password generation by considering several secret parameters from the user. To break a set of consistency passwords, analysis is also performed on time for password cracking. The tests show a close positive correlation between guessing complexity and password consistency. The proposed model is compared with the traditional password generation and cracking models. The proposed model takes much time in cracking the password that improves the systems security.

1. INTRODUCTION

For programs, the continuous use of authentication protection codes, identification of codes and hence the development of passwords are becoming increasingly relevant for study. Many people do not think enough about preserving their password or having a good password. Authentication development regulations also aim to implement encryption by allowing users to apply complexities to passwords such as numbers or special characters. Nevertheless, such politics typically contribute to methods for managing, such as repeating a phrase to render passwords longer regardless of the requirements, which that the reliability of an ideal password cracker. Several analysis of policy development of passwords has been made, but none of them demonstrated the efficiency of such policies against specific attacks [1]. Up to date, no research explicitly focused on password security that the password development policies can have. A significant research issue not yet addressed is then what strategy to build for passwords to be more successful in combating actual attacks [2]. So how do we direct users in creating safe and accessible passwords? A majority of passwords today require 14 or more characters. Nevertheless, the ability of humans to remember long passwords is restricted and a longer password typically contributes to repetitive passwords [3]. There are many password policies which attempt to attach randomness to a user-selected password. Nevertheless, people know a sequence of things like names or common numbers, but not an arbitrary character series. The development of passwords will also be a balance between protection and usability [4]. We may not think such strategies are strong enough on their own. The solution we recommend is focused on tacit policies for the development of passwords in which the program excludes a password depending on its approximate capacity [5]. If the intruder takes a long time to break the password, the password is secure.

Since the Internet was a necessity of society, from business to everyday life for ordinary people, online Security has been a major concern. Protecting data from unauthorized access is a crucial feature of online protection [6]. The most growing approach is to use password in the cycle of online entry. Password is a hidden string which only the user knows and is stored on a server offering access to the data. Users request data access, along with other Identity details such as user name or email, must enter the password [7]. A message digest cypher (Hash), which matches the previously-saved hash code of your database, will be determined and the Hashed value is sent to your server [8]. If a match is found, it is assumed that the user claims to be who he is, and that the access is allowed [9].

The password for operating applications is one of the most commonly used security techniques. Advanced technologies make biometrics and tokens encrypted simple to use, but the password is also an authentication tool often used [10]. Passwords do not need additional hardware, can be implemented easily, and can be used easily. Database administrators utilize password composition policies (PCPs) to discourage users from choosing vulnerable and quickly conjecturable passwords [11]. Password composition policies are commonly understood to make passwords more difficult to devise, but this doesn't always happen [12]. This is since not just the actual password but also the consumer behavior is influenced by the password composition procedure. Users may also use a password composition template to compose a secret, modify unsecure security schemes or adversely impact secret modifications [13]. Values differ in terms of composition of their passwords. The structure of passwords differs from user to user, and might not be followed in certain situations. Websites may also restrict the type of special characters to use and limit the length of their websites [14].

Many technologies were developed to improve the level of Passwords for the security of the user typically provide: Identity management systems or simply using powerful and not easily guys passwords make authentication easier. authentication. In this article we focus on the third of these categories in accordance with previous remarks on the continuing omnipresence of passwords [15]. The fact that many different strong passwords are expected to be saved by users simply for a daily business on the Internet clearly forces users to compromise their own security [16].

The security and privacy of individual information from intruders is a concern to all those who use different online services [17]. Many authentication systems are available for the protection of individuals' data, and the password authentication system is one of them [18]. Increasing sharing of information, popularisation of the Internet, electronically traded transactions and the transmission of data have made password security and authenticity an essential and essential subject [19]. Because of usage of computer in many areas there is a strong need for better authentication model for securing user's information [20].

2. LITERATURE SURVEY

The most popular method to secure keys to data, devices, and network servers is through passwords. Since the first introduction of the UNIX password scheme [21], the password protection analysis has been carried out. Passwords are seen as easy and cost-effective but at the same time as an incredibly poor means of security, they are remembered. System administrators have been proposed to evaluate the risk of attacks by trying to break the codes of their own customers. Therefore, password cracking programs should be accessible and can check consumer passwords' effectiveness. Code crackers were also very helpful in law enforcement during their operations to cope with encrypted files and hard drives. There are two different forms of web and offline password cracking assaults. The intruder attempts to get entry to an active account in an online password cracking attack. In this scenario, authentication mechanisms such as just a few unsuccessful logins are still available with each account. On the other side, the intruder already got a password or encrypted file in an offline password break, and then attempts to decode or locate the password by guessing and attempting different passwords. In this situation, it is not necessary to limit the number of attempts that the intruder can create in order to find the answer, even when he is ready to spend. The typical method for breaking a password is frequently to seek password guessing because the attacker has the password hash. The intruder guesses first, by using the same hash algorithm, the answer for the first password for "football123." When you compare the two hashes, you split the secret, if this is not replicated until you find a match.

2.1 Tips to break password

Every password p hashed to create its hatch code h as defined in q by using an encryption function f. [1]. The task of breaking the password is to use a method c such that c(h) = p because h (instead of p) is only saved in the resource server database. Therefore, the hacker will work out which tool c is to be used to help locate p. Used to break codes, brute force attack, dictionary assault and certain variants of time-space compensation requirements are among the most popular techniques.

2.1.1 Brute force attack

The alphabet l is a character collection alphabet and N = length l, let p = a1, a2 ... al, ai length l passphrase. Since of hash code h = f(p), the assault by a brute force is to seek any string s= x1, x2 .. xl, xi the meaning of oscillating before f(s) = h. In the case of Nl the size of a string set, the possible passwords are identified by the alphabet - T è t (Nl) in the period of time to break a password is equal to the number of possible strings. This is, since brute-force attacks are dynamic, N polynomial and l exponential. The letter è t is typically one of the basic sets or variations.

2.1.2 Strike in dictionary

Since the majority of the people are using human-memorable passwords that are usually words in dictionaries or certain combinations, a hacker can seek could word in the dictionary and not the brute force random string. Any term wi (f) to D in the dictionary D is tested with this approach to see if f(wi) = h is the specified hash code. Thus the assumption that the secret is a word in the dictionary is very easy to crack. In addition, all f(wi) hash codes are pre-computed and processed in a database, instead of computed on a working time basis.

2.1.3 Panel lookup

To render crack time fast, a table search attack saves precomputed hazards in a database of possible passwords and removes a specific password danger by scanning the database. The attack is far simpler than the initial dictionary attack, only because the hash does not have to be determined at runtime for increasing conjecture. The way to store "all" imaginable passwords and hazelnuts, however needs a significant amount of space. The rainbow table solution is a timer to save even less hash codes, but with a vast amount of passwords, rather than pre computing the hash codes of a great deal of possible passwords and storing in a bin. The fundamental concept is to construct a K-length password-hash chain containing k possible passwords and hash codes.

2.1.4 Markov model attack

The point was that people wanted easy to recall passwords. Many users are aware of dictionary attack and recognizable human passwords are usually unique (unique passwords created are difficult to remember). Some of the methods to combat famous passwords is an assault by "digital dictionaries," which users may create through dictionaries. A rapid dictionary attack method based on the likelihood of the

password sequence of characters was introduced by Colnago et al. [1]. The approach uses the Markov standard model to build a far smaller smart dictionary than conventional dictionary assaults. The main observation is, "the distribution of letters in passwords which can be easily remembered is likely to be similar to the distribution of letters in their native language." The Markovian dictionary can therefore be developed in a sequence, depending on the probability of the signs. The passwords are generated as
1. By founding the cypher text "Å" the decimal
Value ASCII.
2. 197 is 11000101 binary value.
3. After 00101 (the last 5 digits) have been multiplied by
The outcome of 1000 (Key) is 101000.
4. After you have added 110 (first three digits of the code)
The outcome is 101110,000.
5. As 101010 isn't an 8-bit we don't have to.
Dear friend
6. It will be 01110100 after the number was reversed.
7. 01110100's decimal value is 116.
In the upper case sequence is the decimal value of "116"
List 50 should therefore be withdrawn
8. ASCII importance detection 8
= 116-50 (decimal-constant value-50) = 66 (ASCII value)
9. Convert the given ASCII to "B" alphabet.

2.2 Password authentication models

Knieriem et al. [2] suggested to incorporate biometric details into the password, a strategy to improve the reliability of password-based programs. Once each user typed his or her password, they registered keystroke features (keystroke length and distance between keystrokes). There were 20 participants, 481 logins and 1 password in this trial. The hardening of device passwords is close to the hardening of passwords. A salt is a randomly generated bit number that is used throughout the encryption process to permute any bits. This technique can also be used to improve salting using users' typing characteristics by determining some or all salt bits. This may also be effective against an intruder who learns the secret and logs in as a customer. Their approach increases the time it takes on an attacker to look for the hard password thoroughly.

The use of Shannon entropy, as described in NIST is not an efficient calculation of password power that was demonstrated by Furnell et al. [3] in regular password cracking attacks against several real-life password sets. We used Rock You user set codes, initially collected through an assault on the domain of rockyou.com. The number contained 32 million passwords, although smaller keys were used to perform the study. They tested the minimum password duration rule in the development of passwords and since they did not have access to the actual passwords, they formed this approach by separating the test set depending on the minimum length.

The entropy value is determined independently by Grassi et al. [4] from distributions of password types, code placings, amount of code kinds, and quality of each character. The entropy value is then summarized as overall entropy. In a broad report, they proceeded to examine the power of their passwords. You carried out an online analysis of five policies for the development of passwords.

3. PROPOSED MODEL

Password guess strengths traditionally is measured by running a password cracking tool and recording when each password is crashed. This works fine when the scan is limited to a relatively small amount of divinations. Nevertheless, it is necessary to remember how many passwords may be solved with even further attempts as the computational ability of potential adversaries grows.

In most deterministic password devaluation algorithms, we take advantage of the fact that a calculator function may be generated which maps a password to the number of guesses required to devise the password. This performance value is called the password guessing number. With any cracking algorithm under review, a new guess-number calculator must be introduced. A new calculator tuning is created for each new training set to be checked for algorithms which use a training set of known passwords to prioritize derivations. Because we gather passwords with plaintext, we can use the calculator method of an algorithm for searching for the corresponding password guess number without actually running the algorithm.

The proposed approach is used to measure in a number of ways the guess ability of the passwords. They calculate the percentage of passwords that a given algorithm will split, which is important since heuristics are used by most successful cracking tools and not every password. They also measure the number of assumptions that will be broken. We also use computers to compare the performance of various splitting algorithms and various training sets in each algorithm. If we integrate the effects of guess numbers in a variety of algorithms and configurations, we may build an analysis of a number of passwords' cumulative power. The proposed Random Character Utilization with Hashing (RCUH) model is explained in the algorithm.

Algorithm Random_Character_Utilization_with_Hashing (RCUH)
```
{
Input the user parameters
UN← getInput(user_name);
PH← getInput(phnone_number);
EI← getInput(email_id);
DOB← getInput(date_of_birth);
EID← getInput(employee_id);
FN← getInput(father_name);
MN← getInput(mother_name);
SN← getInput(school_name);
UGN← getInput(ug_college_name);
SEN← getInput(secret_number);
String  arr[11]={UN,PH,EI,DOB,EID,FN,MN,SN,UGN,SEN}
Hash function is applied on the arr[]
H(x) ← y where $x \in Z$ and $y \in Z_n$
H(x) =←{UN ⊕ SEN}
H(x)← H(x) && {PH || UGN}
H(x)←H(x) &&{EI&&SN}
H(x)←H(x) +{DOB⊕EID}
H(x)←H(x) ⊕{FN||MN}
H(x) =H(x)+$T_h$ (Threshold value considered by system owner)
P(U)←H(x)
Display Password of User→ P(U)
}
```

After comparing the passwords we have collected to lists of 1, 5, and 1 lack guesses produced through various cracking tools and tunings. we selected these as the most promising brute-force and heuristic alternatives. The proposed model involves a training set: a repository of recognized passwords used to create a list of devices and to decide whether to test them. We are investigating a variety of training programs made up of the numerous variations of lists of public word lists and subsets of passwords that have gathered. It enables us to determine how the efficiency of cracking algorithms is enhanced by complementing publicly accessible data with passwords obtained from the device under attack. They do make curriculum improvements that are precisely adapted to our strategies, comprehensive and fundamental. The password calculation parameters are indicated in Table 1.

We use a multi level cross-validation method to measure calculation numbers in each experiment just for the passwords. Passwords are divided into n partitions or folds for a given experiment. We create a publicly accessible training set plus (n−1) folds and check it against the remaining folds.

Each n folds are used precisely once as test data for n learning and training iterations. We will merge tests with all our passwords with data from the n folds. Because training typically requires significant computational resources, we restrict our testing to limited iterations. It appears to be appropriate based on the consistency of the effects we obtained of iterations. Such testing sets or methods are not known to be the best method of guessing the passwords we obtained. They concentrate more on measuring the resistance to guessing through password structure policies. The analysis of algorithms' output for various tunings often gives insight into the kind of data set an intruder may like to efficiently conceive of passwords generated in compliance with the particular password composition policy.

When applications begin to use authentication protection codes, policies to build codes become particularly necessary to examine. When mentioned earlier, neither of the policy-making experiments particularly concentrated on the effectiveness of the authentication. A significant research issue is not addressed yet is then how to establish the most successful model against attacks in the development of passwords. In our view, our solution to this issue is focused on education and studying context-free grammar, and then on using it to create functional passwords for the user. We also developed and created a method to determine user-selected password intensity dependent on the probability that an intruder would break the password. After generating the password by applying hash functions, the password should be checked for capability. The password cracking is analyzed using the model depicted below.

Algorithm RCUH_Cracking

```
{
Input parameters from the USER.
String data[11];
For i=0
do
Data[i] ← getInput();
done
while (data > 0)
{
    Crapass ← digits[data % size];
    val ← val/size;
    $T_i$ ← $I_{iter\_count}$ + len(val);
   Hash (k) ← $\lambda Ti + \sum_{i=0}^{L}$ val $+ k(i) + \beta + M$
  }
$T_i$ ←  $I_{iter\_count} + \lambda Ti + \beta$
Where $\lambda$ represents password length and $\beta$ represents iteration level.
for (int j = $T_i$; j >= 0; --j)
  {
    if (crapass [j] == 0)
    {
      --T;
      crapass[j] ←val[0];
      break;
    }
    string::size_t_found ← alphabet.find(crapass[j]);
    ++ $I_{iter\_count}$;
    if (size_t_found < crapass.length)
    {
      crapass[j] ← val[size_t_found];
      break;
    }

crapass[j] ←  val[0];
    if (j == 0)
    {
      return crapass("");
    }
    continue;
  }
Displat $T_i$, crapass.
}
```

Table 1. Password calculation parameters

Parameters Considered	No.of users	Password Length in bits	Passowrd Strength	Password Calculation time in Miliseconds
5	100	56	Average	3
7	200	128	Good	5
9	300	128	Good	7
10	400	256	Strong	9

Table 2. Password cracking parameters

Parameters Considered	No.of users	No.of Iterations	Passowrd Strength	Password cracking time in Miliseconds	Cracked Status
5	500		Average	7	Success
7	500		Good	10	Failed
9	500		Good	12	Failed
10	500		Strong	15	Failed

Explicit protocols include guidelines for how to create a password. A majority of passwords today require 14 or more characters. Nevertheless, the ability of humans to remember long passwords is restricted and a longer password typically contributes to repetitive passwords. There are now a number of password generation policies, requiring users to use minimum-length passwords, or, for example, to include at least two or non-alphanumeric numbers. The practical efficacy of the tactics of specific assault was not, however, researched. For example, a protocol that allows a user to use at least three digits in a user's password would typically automatically attach "123" to a user's password at the end of an unreliable one. Regulations, including 14 characters, can not only hinder the consumer but can also be helpful for an attacker as people tend to follow common patterns and may end up choosing popular variations of keys or simply repeating the same password two or three times. The parameters considered for password cracking are listed in Table 2.

The proposed model considers a threshold value 'T' that is a probability value t, which means that passwords are less than likely to be good enough. The effectiveness of a password depends on how much time an attacker takes to crack the password. As previously explained, there are growing types of online threats, such as hostile powers and dictionary attacks. We assume that probabilistic password cracking is the most effective solution. And, any time we think of an intrusion we believe the intruder is clever enough to find the right method to break the password. In any applications the intruders involve to degrade the system performance. To avoid intrusions in any application, Intrusion Detection System (IDS) model need to be integrated to the security application for attaining better results. If IDS model is integrated, then if any intrusion is identified, then the system will trigger a signal for identification and removal of malicious nodes in the system for enhancing the security levels.

The simple premise is that the intruder should seek passwords with the greatest likelihood. So we will calculate how many g(t) presumed an intruder will be achieved before guessing a password with a likelihood equivalent to the T threshold value if user begins to presume from the maximum probability level. By dividing the total sum of g(t) measurements by the value of c hour for each device, we know precisely how long it is appropriate for the intruder to arrive at this point, based on the form of hash, the machine speed etc.

$$Expected\ Time\ (Ti) = \frac{Total\_number\_of\_user\_guesses}{Calculations\_per\_minute}$$

They recommend two methods of describing the threshold: In the first solution, we would have a record that compares increasing possibility to the total number of attempts the intruder will create before hitting the probability by running the proposed password cracker once beforehand. While this is a simpler and more precise method, the optimal amount of estimates cannot always be met owing to time and money. For example, if the chance of a password is less than the threshold, we know that it takes at least 1 week and 3 days for the password to be broken using the optimal password cracking technique. The next method only helps users to reduce the number of assumptions g(t) before a specified value t is reached, but it just involves the use of context-free grammar, and does not necessarily need to produce any guesses. It is therefore conservative to ensure that the password proposed is secure. The algorithm starts with the identification of a threshold t and the estimation of the number of elements greater than this amount within each basis structure.

The challenge that can exist in this process is that the probability of accurate guessing in a very short time is not guaranteed. The concerns are what if there is no context-free grammar comprising the foundation structure or any other components of the user-chosen password.

• If the user-selection password's basic structure isn't found in context-free language, either we can presume and agree that the password is solid enough or we can consider the lower probability for the basic structures and use it as an estimate of the certainty of that basic structure. Since only the above method is being taken, and we did not test the system with these two solutions, there is no outcome showing which one is stronger.

• If the secret digit components or the special character components were originally not included in the training data, we still have a chance to find these values as we have a grammar that contains these baseless values in the training set.

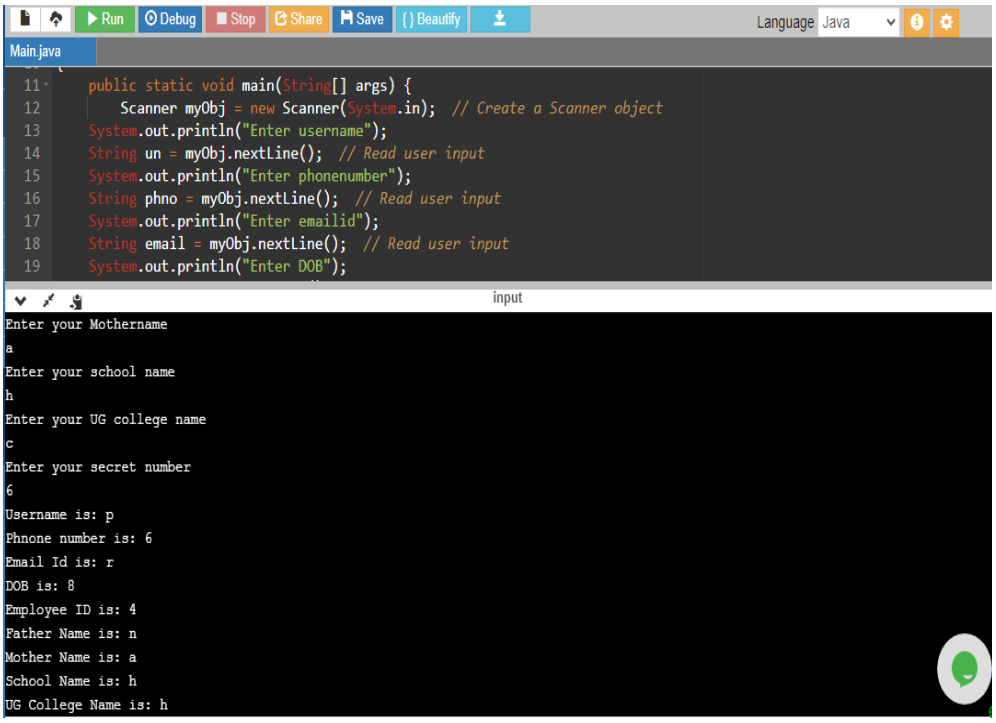
• If the password's alphabet component is omitted from the dictionary, we use the same length term possibility in the dictionary, as all terms with the same length are of equal probability. We always presume that the frequency of both words is the same and they are all contained in the dictionary. In future, we will be able to try various approaches such as giving the words not found in the dictionary much less chance. The proposed model generates passwords which takes more time to crack is observed in the proposed model.

4. RESULTS

The proposed RCUH model for password generation and cracking analysis model is implemented in JAVA. The proposed model is compared with the traditional models by considering the parameters password generation time, password cracking time, hash key calculation time, Security Level. The proposed model generates the passwords more strongly as it takes more time for the attacker to crack the key. The proposed model takes input from the user as a single character for different parameters and then applies hashing on it to result a strong password. The model is depicted in Figure 1.

The Password Generation Time of the proposed and traditional Weir model is depicted in Figure 2. The proposed model takes less time in generation of password. The codes have various user attributes and their password will be created with each generation model and then test its accuracy. The test results indicate that the matching attribute password devaluation in the required test range has the highest accuracy. The proposed RCUH model is strong enough to design the passwords to secure the system.

After taking the specified number of user characters for every parameter, the hash key is generated by applying the modern mathematic operations so as to generate the password that should be strong enough to secure the system and the cracking possibility should be complex. The key idea of a model generated by passwords is to predict the next character with the current password generated. Before the model was entered, each training password was interpreted as matrices. The duration of the input sequence in the model during the training process is provided by the user. The hash key calculation time is depicted in Figure 3.



```java
public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);  // Create a Scanner object
System.out.println("Enter username");
String un = myObj.nextLine();  // Read user input
System.out.println("Enter phonenumber");
String phno = myObj.nextLine();  // Read user input
System.out.println("Enter emailid");
String email = myObj.nextLine();  // Read user input
System.out.println("Enter DOB");
```

```
Enter your Mothername
a
Enter your school name
h
Enter your UG college name
c
Enter your secret number
6
Username is: p
Phnone number is: 6
Email Id is: r
DOB is: 8
Employee ID is: 4
Father Name is: n
Mother Name is: a
School Name is: h
UG College Name is: h
```

Figure 1. User character input for parameters

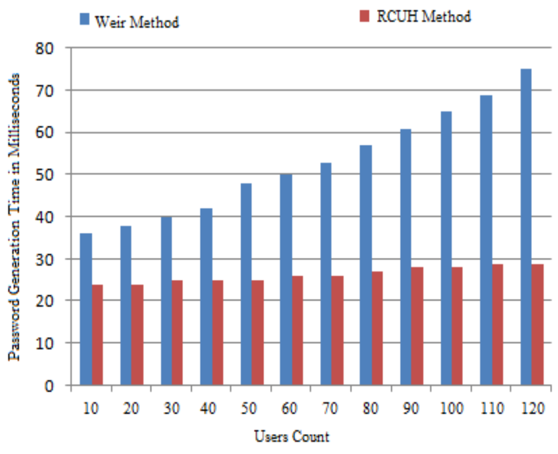


Figure 2. Password generation time

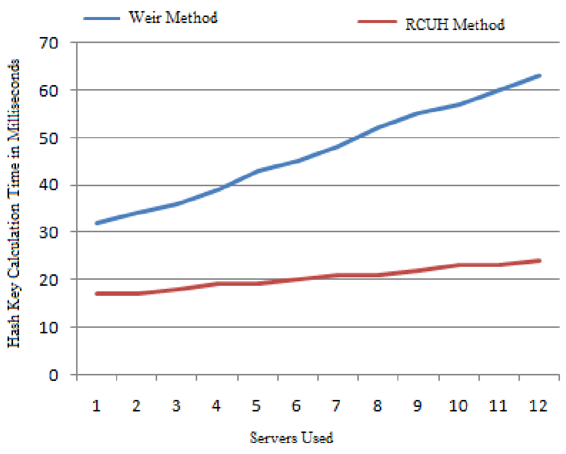


Figure 3. Hash key calculation time

The password that is generated should undergo, password cracking module to check its capability and also to analyze the time taken for cracking it. As the time is more, the capability of the password is high. The Password cracking time of the proposed and the traditional models are depicted in Figure 4.

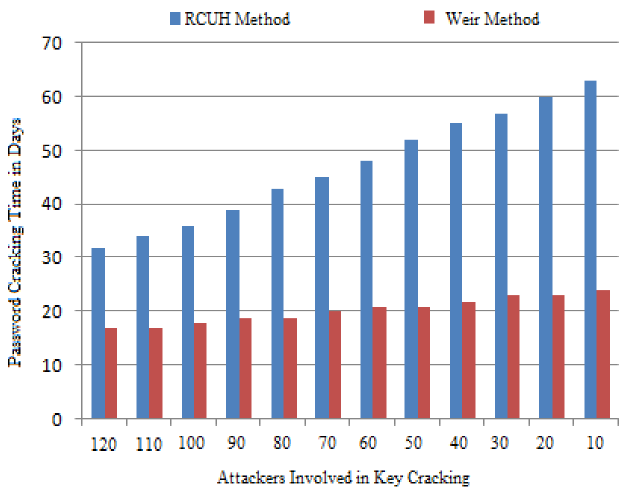


Figure 4. Password cracking time

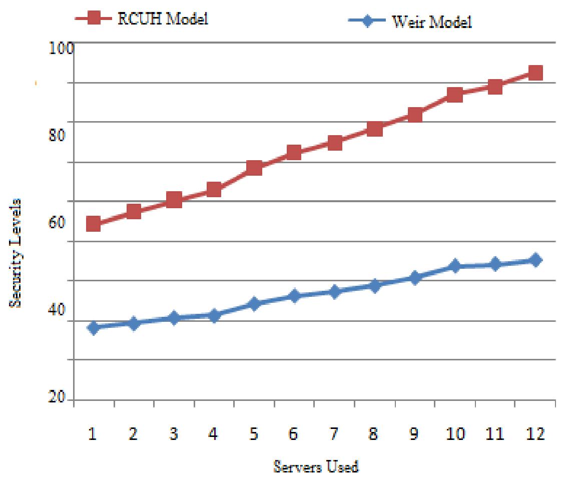


Figure 5. Security level

The proposed RCUH model exhibits better performance in terms of security and cracking time. The cracking model is used after the training to test the model for password power, and even passwords to expand the dictionary of passwords that are used for devaluation. Until devaluation, a network exploit method will be used to extract the password file. The Security levels of the proposed model and the traditional methods are depicted in Figure 5. The password if takes more time to crack, it represents that the password is strong enough to provider security to the system.

5. CONCLUSION

To certain individuals in today's country, managing passwords is a big issue. This is best to use codes that attackers consider challenging to formulate. Although network administrators are increasingly improved, the amount and difficulty of the specifications for password creation, the true added benefit of the specifications remains poorly known. Its research represents a major step in not only recognizing these concerns, but also the assessment process. Our studies often provide valuable details regarding the study of guessing resistance. The ample, closely related knowledge on training is needed for successful attack of passwords generated under a complicated or unusual in-practice composition of policies. Therefore, the collection of a subset of compatible passwords from a wider quantity will only accurately define this form of password set; it is possible that this subset does not represent passwords established under that program. Eventually, we stated that the proposed model generates a statistics approach for password power, and can accurately estimate any quantitative variations in deviations between password sets. The proposed model cracking time is very high when compared to traditional models.

REFERENCES

[1] Colnago, J., Devlin, S., Oates, M., Swoopes, C., Bauer, L., Cranor, L., Christin, N. (2018). “It’s not actually that horrible”: Exploring adoption of two-factor authentication at a university. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI ’18, ACM, New York, NY, USA, pp. 1-11. https://doi.org/10.1145/3173574.3174030

[2] Knieriem, B., Zhang, X., Levine, P., Breitinger, F., Baggili, I. (2018). An overview of the usage of default passwords. In: Matoušek, P., Schmiedecker, M. (eds.) Digital Forensics and Cyber Crime. ICDF2C 2017, Springer, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 216: 195-203. https://doi.org/10.1007/978-3-319-73697-6_15

[3] Furnell, S., Esmael, R., Yang, W., Li, N. (2018). Enhancing security behaviour by supporting the user. Computers & Security, 75: 1-9. https://doi.org/10.1016/j.cose.2018.01.016

[4] Grassi, P.A., Garcia, M.E., Fenton, J.L. (2017). Digital Identity Guidelines. Special Publication (NIST SP), pp. 800-63-3. https://doi.org/10.6028/NIST.SP.800-63b

[5] Murray, H., Malone, D. (2017). Evaluating password advice. In: 28th Irish Signals and Systems Conference (ISSC), pp. 1-6. https://doi.org/10.1109/issc.2017.7983609

[6] Habib, H., Colnago, J., Melicher, W., Ur, B., Segreti, S.M., Bauer, L., Christin, N., Cranor, L.F. (2017). Password creation in the presence of blacklists. In: Workshop on Usable Security, USEC ’17, Internet Society. https://doi.org/10.14722/usec.2017.23043

[7] Ur, B., Noma, F., Bees, J., Segreti, S.M., Shay, R., Bauer, L., Christin, N., Cranor, L.F. (2015). “I added’!’ at the end to make it secure”: Observing password creation in the lab. In: Eleventh Symposium on Usable Privacy and Security (SOUPS 2015), USENIX Association, Ottawa, pp. 123-140. https://doi.org/10.14722/usec.2017.23043

[8] Florencio, D., Herley, C., van Oorschot, P.C. (2014). Password portfolios and the finite-effort user: sustainably managing large numbers of accounts. In: Proceedings USENIX Security, pp. 575-590.

[9] Shay, R., Komanduri, S., Kelley, P.G., Leon, P.G., Mazurek, M.L., Bauer, L., Cranor, L.F. (2010). Encountering stronger password requirements: user attitudes and behaviors. In Proceedings of the Sixth Symposium on Usable Privacy and Security, pp. 1-20. https://doi.org/10.1145/1837110.1837113

[10] Clair, L.S., Johansen, L., Enck, W., Pirretti, M., Traynor, P., McDaniel, P., Jaeger, T. (2006). Password exhaustion: Predicting the end of password usefulness. In International Conference on Information Systems Security, pp. 37-55. https://doi.org/10.1007/11961635_3

[11] Burr, W., Dodson, D., Polk, W. (2004). Electronic authentication guideline (No. NIST Special Publication (SP) 800-63 Ver. 1.0 (Withdrawn)). National Institute of Standards and Technology.

[12] Proctor, R.W., Lien, M.C., Vu, K.P.L., Schultz, E.E., Salvendy, G. (2002). Improving computer security for authentication of users: Influence of proactive password restrictions. Behavior Research Methods, Instruments, & Computers, 34(2): 163-169. https://doi.org/10.3758/bf03195438

[13] Weir, M., Aggarwal, S., Collins, M., Stern, H. (2010). Testing metrics for password creation policies by attacking large sets of revealed passwords. In Proceedings of the 17th ACM conference on Computer and communications security, pp. 162-175. https://doi.org/10.1145/1866307.1866327

[14] Ross, J., Irani, L., Silberman, M.S., Zaldivar, A., Tomlinson, B. (2010). Who are the crowdworkers? shifting demographics in Mechanical Turk. In CHI'10 Extended Abstracts on Human Factors in Computing Systems, pp. 2863-2872. https://doi.org/10.1145/1753846.1753873

[15] Ipeirotis, P. (2010). Demographics of mechanical Turk. New York University. Tech. Rep.

[16] Buhrmester, M., Kwang, T., Gosling, S.D. (2011). Amazon’s Mechanical Turk: A new source of inexpensive, yet high-quality, data? Perspectives on Psychological Science, 6(1): 3-5. https://doi.org/10.1037/e527772014-223

[17] Downs, J.S., Holbrook, M.B., Sheng, S., Cranor, L.F. (2010). Are your participants gaming the system? Screening Mechanical Turk workers. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2399-2402. https://doi.org/10.1145/1753326.1753688

[18] Kittur, A., Chi, E.H., Suh, B. (2008). Crowdsourcing user studies with Mechanical Turk. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 453-456. https://doi.org/10.1145/1357054.1357127

[19] Toomim, M., Kriplean, T., Pörtner, C., Landay, J. (2011). Utility of human-computer interactions: Toward a science of preference measurement. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2275-2284. https://doi.org/10.1145/1978942.1979277

[20] Chiasson, S., Forget, A., Stobert, E., Van Oorschot, P.C., Biddle, R. (2009). Multiple password interference in text passwords and click-based graphical passwords. In Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 500-511. https://doi.org/10.1145/1653662.1653722

[21] Kuo, C., Romanosky, S., Cranor, L.F. (2006). Human selection of mnemonic phrase-based passwords. In Proceedings of the Second Symposium on Usable Privacy and Security, pp. 67-78. https://doi.org/10.1145/1143120.1143129