Achieving Flatness: Honeywords Generation Method for Passwords based on User Behaviours

Omar Z. Akif[1]
Department of Computer
College of Education for Pure Science (Ibn al-Haitham)
University of Baghdad
Iraq, Baghdad

Ann F. Sabeeh[2]
Department of Computer
College of Education for Pure Science (Ibn al-Haitham)
University of Baghdad
Iraq, Baghdad

G. J. Rodgers[3]
Department of Mathematics
College of Engineering, Design and Physical Sciences
Brunel University London
Uxbridge, London, UK

H. S. Al-Raweshidy[4]
Electronic and Computer Engineering Department, WNCC
College of Engineering, Design and Physical Sciences
Brunel University London
Uxbridge, London, UK

***Abstract*—Honeywords (decoy passwords) have been proposed to detect attacks against hashed password databases. For each user account, the original password is stored with many honeywords in order to thwart any adversary. The honeywords are selected deliberately such that a cyber-attacker who steals a file of hashed passwords cannot be sure, if it is the real password or a honeyword for any account. Moreover, entering with a honeyword to login will trigger an alarm notifying the administrator about a password file breach. At the expense of increasing the storage requirement by 24 times, the authors introduce a simple and effective solution to the detection of password file disclosure events. In this study, we scrutinise the honeyword system and highlight possible weak points. Also, we suggest an alternative approach that selects the honeywords from existing user information, a generic password list, dictionary attack, and by shuffling the characters. Four sets of honeywords are added to the system that resembles the real passwords, thereby achieving an extremely flat honeywords generation method. To measure the human behaviours in relation to trying to crack the password, a testbed engaged with by 820 people was created to determine the appropriate words for the traditional and proposed methods. The results show that under the new method it is harder to obtain any indication of the real password (high flatness) when compared with traditional approaches and the probability of choosing the real password is 1/k, where k = number of honeywords plus the real password.**

Keywords—Honeywords; user behaviours; worst password list; dictionary attack

I. Introduction

When any user wants to access a network for security purposes, he or she is prompted to enter credentials [1]. A password is the popular authentication technique being used today despite many newer ones, such as biometric based techniques and dual factor authentication [2]. Users tend to use small simple passwords and for this reason as well as their somewhat universal use, they are vulnerable to being compromised. Hence, it has become important to make progress in combatting cracking techniques [3]. Since these are becoming increasingly sophisticated, has become a salient issue [4].

Intruders are increasingly eavesdropping on communication between legitimate users and servers as well as masquerading as authorised users or remote servers so as to be able to steal sensitive information [5]. A good password has to have two features: a user can remember it and it is difficult to guess [6]. Unfortunately, these two work against each other such that a password that is easy to remember is generally short and hence, easy to guess. Moreover, most people choose to use a single password for multiple accounts, because one is easy to remember. Invariably, people have a hierarchy of passwords, for example, they do not use the same password for email as they do for their bank account [7], in particular, because the bank requires more stringent security.

The idea behind honeywords is to create a relation between the real password and decoy hashed passwords, such that for every user the latter look like real passwords. The honeywords are these decoys. An attacker can recognise the presence of honeywords in a password file, as it is very unusual to have multiple passwords for a single user account. However, even if the attacker can crack multiple passwords associated with a user, he or she does not know which are honeywords, and which are the real ones [8]. What is the main focus of honeywords generation is the way in which they a produced. Currently, there are some problems regarding this, which are discussed later in this paper and a new generation method will be proposed to overcome these.

II. Related Work

Passwords are especially vulnerable to hash chain based and effective dictionary attacks. A sample of 19 million passwords, of different lengths, available online, have been studied according to the distribution of the symbols in the password strings. The results have shown that the native language of the user can affect the distribution of symbols in passwords [9].

Hashing the plaintext or password is a one-way function, which makes it hard to find the required password [10]. However, rainbow tables, which are massive tables filled with hash values and can be used to find a required password, whereby a hacker employs them to find the password by reversing the hashing function. Despite of a rainbow table taking up a lot of storage when holding it, attackers can usually crack the password in a shorter amount of time than when applying the brute force technique [11].

Most existing biometric template protection schemes (BTPS) do not offer as strong security as cryptographic tools. Moreover, they are unable to determine whether or not a probe template has been downloaded the database by an imposter or an authentic user. Consequently, the "honeywords" idea was proposed to detect the cracking of hashed password databases. In particular, an extra layer of protection is needed with biometric feature schemes, as these have been shown to be flawed. A honey template protection scheme relating to faces has been proposed and evaluated as representing an improvement on existing schemes [12].

Many researchers have pointed out that most password hashes are not safe against hackers and hence, the method of honeywords (decoy passwords) has been used to detect attacks against hashed password databases. Furthermore, cracking hashed passwords has become easier for an intruder, who wants to enter the account through an authenticated user. In addition, a user's password can be recovered by an intruder through using a brute-force attack on the hashed password. A user's real password can be distinguished among honeywords for each user by using a secure server called a "honeychecker", which triggers an alarm when a honeyword is used. [13].

III. Review of Honeywords

The concept of honeywords is to provide a technique to detect whether an intruder into password files has been originally invited in. Essentially, the scenario of honeywords is based on any user u_i being provided with a list of k called "*sweetwords"*, which are denoted as $W_i = \{w_1, w_2, w_3, \dots, w_k\}$. One of these sweetwords (for instance w_j) is the right user's password, while the rest of the list $(k-1)$ are fake and called *honeywords*. The main architecture feature is a new server, "the honeychecker", which contains a database, for each user u_i and the index $c(i) = j$, where $w_j \in W_i$ is the correct password of u_i. The right password is referred to as the "real" password in line with the person who introduces honeywords.

The real password of the authorised user will be generated and entered during the registration stage, while on the basis of such a password; the system generates and adds $(k-1)$ honeywords. Moreover, the honeywords generation algorithm is targeted at creating decoy passwords that are the same as the real one, so that an intruder will not be able to recognise them from the real password. Accordingly, the system chooses a random $1 \leq \mathrm{j} \leq \mathrm{k}$, gives the real password to $\mathrm{w_j}$ and populates the set $\mathrm{W_i}$ with the generated honeywords. Finally, the password along with the honeywords are "hashed" and saved in the password file in the form $\mathrm{H_i} = \{\mathrm{h_1} = \mathrm{hash(w_1)}, \dots, \mathrm{h_k} = \mathrm{hash(w_k)}\}$, while the index $\mathrm{c(i)} = \mathrm{j}$ is stored by the honeychecker.

TABLE I. Related Notation

Notations	Meaning
u_i	i^{th} user in system
P_i	password of i^{th} user
W_i	Tuple of passwords stored for u_i
k	Number of elements in W_i
$c(i)$	index of correct password in W_i
sweetword	each element of W_i

When $\mathrm{u_i}$ logs-in the system, he or she should enter the password and then, the system will check $\mathrm{hash(p)}$ against each hashed sweetword in $\mathrm{H_i}$. If the password that has been entered does not match with any elements of$\mathrm{H_i}$, the connection is rejected. In contrast, let j be such that $\mathrm{hash(p)} = \mathrm{h_j}$, then the pair $\mathrm{u_i, j}$ will be sent to the honeychecker. Hence, if $\mathrm{j} = \mathrm{c(i)}$, then the authentication succeeds, and the honeychecker will send back its "approval", whilst otherwise an alarm is triggered, as the password file has probably been attacked. Table I illustrates the related notation [14].

IV. Limitations of Honeywords

Despite of the fact that current honeyword based methodologies can provide security against brute force attacks, they do have some limitations, are described below.

1) Co-relational hazard: If a relationship exists between the username and password, then the real password of the user can easily be recognised from the list $\mathrm{W_i}$. In such cases honeywords cannot protect the original password, because of this association.

2) Distinguishable well-known password patterns: If a user chooses a password linked to some well-known object/fact, then an adversary can simply recognise the real password. For example, some of the passwords belonging to this category are bond007, james007, 007bond and 007007, which were found in a list of 10,000 most common passwords (these will be used to generate the honeywords in this paper).

3) Issue related to DoS resistivity: If an adversary knows the real password of the user, then he or she can recognise the honeywords and then, can intentionally submit honeywords to produce a false negative feedback signal by the "honeychecker". If the adversary obtains these honeywords from several accounts, then all the web server may become blocked. This is known as a Denial-of-Service (DoS) attack and the real password of user should be not giving any knowledge about system generated honeywords to avoid one.

4) The issue relating to multiple system vulnerability: If a user uses the same password in two (or more) different systems, and if two systems are employing the same honeyword generation algorithm, when an adversary gets access to both systems, then Multiple System Vulnerability can occur. In this case, the adversary may obtain two lists of

W_i for the user u_i. Let $W_i^{S_j}$refer to the list of sweetwords for user u_i in the system S_j. So, if honeywords that have been generated belong to $W_i^{S_p}$ and $W_i^{S_q}$ (where p ≠ q) are different (probability of which is close to 1), then by performing the connection operation $W_i^{S_p} \cap W_i^{S_q}$ the unauthorised user can obtain the real password. This is known as Multiple System Vulnerability (MSV) of the honeyword based authentication technique [15].

V. Password Attacks

Password attacks include different character combinations being tried until a match with the correct password is found. There are several types of password attacks, some of the most important of which are described next.

1) Brute force attacks: In this type of attack, all the possible combinations of the password are applied to break it. It can also be applied to crack encrypted passwords wherever they are saved in the form of encrypted text [16].

2) Dictionary attack: A dictionary attack is applied to verification data by trying every word in the dictionary. This kind of attack is targeted at sites with a high probability of success, such as those with weak passwords or with only a few key combination numbers. This attack is faster than an attack of brute force and is more successful when a weak, public or short password is used [17].

3) Phishing attack: This is where an attacker attempts to retrieve legitimate users' confidential and sensitive credentials fraudulently by mimicking electronic communications from a trustworthy or public organisation in an automated fashion. The aim of phishing is to steal sensitive information, such as online banking passwords and credit card information from Internet users [18]. These attacks use a combination of social engineering and technical spoofing techniques that persuade users into giving away sensitive information that the attacker then uses to make a financial profit [19].

4) Password guessing attack: In this attack, the adversary steals the file of the password from the main server, and also obtains plaintext passwords by reversing the hash values detected [20].

VI. Personal Information in Passwords and Human behaviours

A text-based password is the most common authentication method and is likely to remain so for the foreseeable future. Whilst users have been recommended different types of authentication mechanisms, passwords are still considered the best way to protect access to a system. That is, none of the alternative technique can provided all the benefits of passwords without introducing extra burden to the users. However, passwords have been criticised as being one of the weakest techniques in relation to authentication. One of the key reasons for this weakness can be put down to the limitations of the human memory. For, as a consequence, most passwords rather than being real random strings and hence, quite strong, are simple so they are easy to remember [21].

Basically, people prefer to create passwords according on their personal information, because of the limitation of their memory capacity and a random password can be difficult to remember [22]. From the other side, some people have exceptional skills when it comes to predicting human behaviour and they use these skills to launch attacks through password hacking this can cause serious problems, which have become the focus of much research [23].

VII. List of the Worst Passwords

Not only do most users create an easy password because they can easily remember it, for they often also use the same one in several systems. Whatever the case, frequently they are easy to guess by the intruder. A list of the 500 worst passwords has been created by researchers to help users avoid selecting them. Unfortunately, one in nine users employ one from this list and one in 50 use a password from the top 20 [24].

VIII. Honeywords Generation Methods and Discussion

In this section, some of the honeywords generation methods are discussed.

1) Chaffing-by-tweaking: This method involves tweaking the real password by selecting the character positions that will be tweaked to produce the honeywords, so the user password will be the seed of the generator algorithm. The same type of character will be selected: letters are replaced by letters, digits by digits, and special characters by special characters. For instance, when $t = 3$ and the last t characters have been selected for tweaking, the method for the generator algorithm is $Gen(k,t)$. While another approach called "*chaffing-by-tweaking-digits*" is carried out by tweaking the last t positions that contain digits. For instance, if the last algorithm has been used, then for the password *42hungry* and $t = 2$, the honeywords *12hungry* and *58hungry* may be generated.

Remark 1. Most people prefer to choose the numbers involved in passwords relating to a special date (birthday, anniversary or an important historical event). For this reason, it is highly probable that such a password includes a digit sequence like *19xx*, *20xx* or *xx*, where *xx* represents the last two digits of the date. Hence, for those passwords that involve applying the *chaffing-by-tweaking-digits* method, the date digits will be replaced with randomly selected digits. Basically, an adversary will recognise the true password easily among the honeywords. For example, assume the honeywords are generated with $t = 4$ and $k = 9$ for the password *alex1992*.. It can clearly be seen that the digits in the honeywords do not relate to a specific date and hence the correct password, *alex1992*, is logically deducible by an adversary [25].

alex6323 alex9058 alex1992
alex1270 alex0976 alex2785
alex5469 alex8147 alex9705

2) Chaffing-with-a-password model: In this technique, the generator algorithm takes the password from the user, and then a probabilistic model of the original passwords is relied upon to generate the honeywords. To give an example of applying this technique, known as *modelling syntax*, the model is divides the real password into character sets. For example, the password *mice3blind* is decomposed as four-letters + one-digit + five-letters (L4+D1+L5) and is replaced with the same structure, such as *gold5rings*.

Remark 2. There are some well-known patterns that have appeared when a password database has been leaked. For example all of the following passwords are included in the list of the 10,000 most common passwords and in the worst passwords list.

bond007 james007
007bond 007007

So, the adversary will easily identify the real password, if it is one of these generic passwords [25].

3) Hybrid method: This method involves combining of the strength of different honeyword generation methods, e.g. *chaffing-with a-password model* and *chaffing-by-tweaking-digits*. For instance, let the original password be *apple1903*, then the honeywords *angel2562* and *happy9137* might be produced as seeds to *chaffing-by-tweaking-digits*. For $t = 3$ and $k = 4$, for each seed, the honeywords will be as follows:

happy9679 apple1422 angel2656
happy9757 **apple1903** angel2036
happy9743 apple1172 angel2849
happy9392 apple1792 angel2562

Remark 3. This method will reduce the chance of an adversary recognising the real password. Nevertheless, the previous remarks are still valid for this case. For example, an intruder may make reasonable guesses regarding the real password [25].

IX. Analysis of the Security of Honeywords: Denial-of-Service Attack

A denial of service attack gives an adversary access to the network services, thus preventing the authorised users from doing so [26]. Once in the system, he/she will use intensive computation tasks against the victim thereby exploiting system vulnerability. Another method is flooding the system with a huge amount of useless packets and as a consequence, the victim can be forced out of service for from a few minutes to several days [27].

X. Proposed Honeywords Generation Algorithm

In the proposed honeywords generation algorithm, dictionary attack, personal questions-answers, the 500 worst passwords list and character shuffles have been used to generate the honeywords. The aim is to increase the flatness of the honeywords, thereby making an adversary confused when trying to identify the real password. The scenario for honeywords generation is the same as the traditional, whereby a list of k honeywords is provided for user u_i, denoted as $W_i = \{w_1, w_2, w_3, \dots, w_k\}$. One of these honeywords (for instance w_j) is used as the real password, while the remaining W_i $(k-1)$ are fake, with the aim being to as aforementioned to increase the flatness.

The proposed honeywords generation method with a password includes at least one letter and one digit. Illustration of the Whole Structure of the Proposed Honeywords Generation Method is shown in Fig. 1.

Step 1
Analysing the password:

a- How many digits?
b- How many letters (Upper case and lower case)?
c- How many special characters?

Step 2
Generating the Honeywords

1- Creating a database containing the public personal questions (50-60 questions), which are divided into two parts according to the type of answers. The first part is associated with the names, and will be generated as letters (for example, your nickname, city you like, your favourite team, pet's name and so on). Whilst the second part will be relating to digits (date of birth, anniversary, best year in your job and so on). Six questions will be chosen randomly from the database (three from each part). Then, five honeywords will be generated by combining the first part answers with the second. Any user can ignore any question, if he/she does not want to answer it and immediately, this question will be replaced by another. In addition, if there are just two digits in the original password, then the algorithm will select that number for the honeywords from the digits answers (This group is called G1).
For example:
a- Letters part Nickname: Mero
Child's name: Peter City: London
b- Digits part

Best year in your job?: 2005 In which year was your father born?: 1948
In which year did you have your last long journey? 2014
The honeyword results will be:
Mero2005 London1948 Peter2005
Mero1948 London2014

2- This type of honeywords is generated based on a dictionary attack, with four being created this type of group. The principle behind how to make suitable honeywords is about searching through the dictionary attack and using the real password with a difference up to three digits or letters (This group is called G2).
Note: Some passwords are not applicable with this type of group due to their being too difficult to find in the dictionary attack, in which case four honeywords will be generated from the other groups.

3- This group of honeywords is made according to the 500 worst passwords list; with five being chosen randomly from this list (This group is called G3).

4- This type of honeywords is made by shuffle and then some letters or digits from the ID user mixed. Subsequently, the real password together with some digits and letters are generated to be inserted in the honeywords, with meaningless words then being generated. In this step, 10 honeywords are created (This group is called G4).

Special cases

5- If there is a special character(s) included in the password, then the honeywords will contain the same number of these generated randomly.

6- The number of upper case letters in the password will equal the number in the honeywords.

7- If there are two words the same in list W_i for the user u_i, then the algorithm explained in Fig. 2 will be applied. Basically, if the original password is one of these two words then the honeyword will be replaced

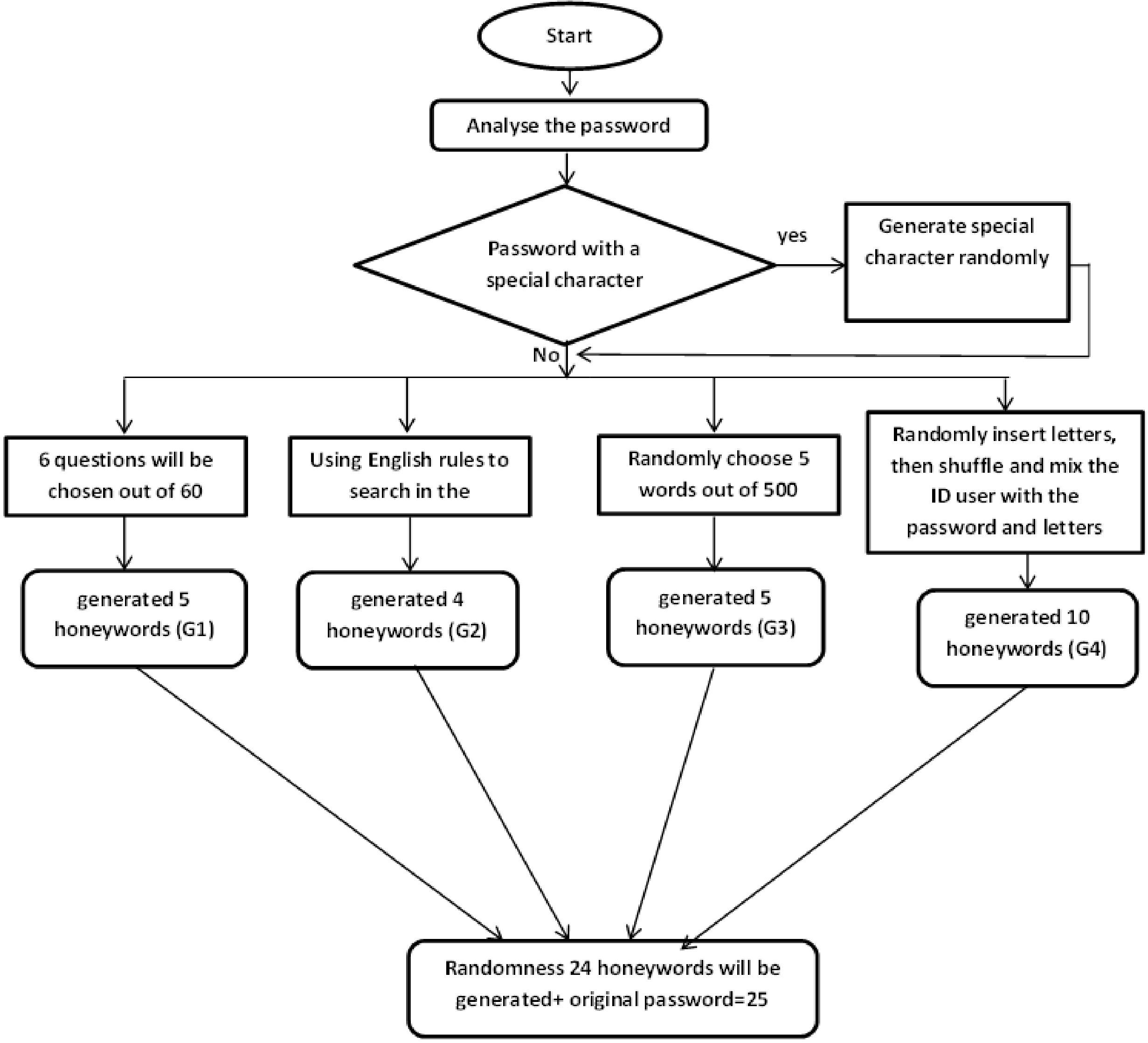
Start
Analyse the password
Password with a special character
yes
Generate special character randomly
No
6 questions will be chosen out of 60
Using English rules to search in the
Randomly choose 5 words out of 500
Randomly insert letters, then shuffle and mix the ID user with the password and letters
generated 5 honeywords (G1)
generated 4 honeywords (G2)
generated 5 honeywords (G3)
generated 10 honeywords (G4)
Randomness 24 honeywords will be generated+ original password=25

Fig. 1. Illustration of the Whole Structure of the Proposed Honeywords Generation Method.

XI. Analysis of the Security of the Proposed Generation Method: Denial of Service Attack

Using the proposed method will partially reduce the number of DoS attacks, but this is an improvement on current method, because it provides greater resistance, that is, increasing the flatness in the list of honeywords and the real password W_i makes the proposed method stronger than the traditional methods against these attacks. Because w_j in W_i can be either a honeyword or the real password, the flatness will make the attacker confused when trying to guess the real one. In the existing methods, an attacker has to know how the honeywords pertaining to a particular password have all been generated by random tweaking, which is possible and then he/she can run a DoS attack. With the proposed method the honeywords are generated according to several procedures, and not randomly.

XII. Analysing the Flatness in the New Honeywords Generating Method

The flatness in the proposed method and how an adversary tries to analyse the honeywords in W_i for each $u_i$is discussed in this section. Obviously, the adversary does not have any predefined information about the password; however, he/she will try to analyse W_i to find any information about c_i. The honeywords created in this first group are associated with personal questions will most probably lead to personal answers. In this case, six answers, which are either in letter or digit form and the letters and digits are then randomly mixed to produce five honeywords. The high level of association of these honeywords with a user $u_i's$ real answers will make it difficult for the adversary to identify which one is the real password, i.e. this increases the flatness. In contrast, the traditional methods do not take into consideration whether there is a personal password, because all the honeywords are generated by tweaking some letters or digits in the real password.

It is clear that *chaffing-by-tweaking* has many problems; the first relates to when a digit is replaced by another. That is, generation of a honeyword does not relate to the human dates, starting either with 19xx or 20xx. The second problem is that not only is the digits tweaking easily recognised by the adversary, but also, he/she can be easily find the password when letters are replaced. For instance, when t=3, this means three letters will be replaced randomly by others, which results in the meaning of the original password not being present in the honeywords and hence they are completely distinct from the former. So, recognising the original password, which is the meaningful word among meaningless ones, will be very easy. For these reasons, the first group has been generated based on personal information, because most users continue to use this when they create their passwords.

Dictionary attacks are commonly used to break passwords, but in the proposed method they are used to generate the honeywords. Such an attack involves most of the passwords that have been created by users around the world, by using an algorithm based on English language rules to make the search in this dictionary to find honeywords very close to the original password. This algorithm tries to find words in the dictionary attack with the most same letters or digits with up to plus or minus three characters or digits. The first priority to find the different words will be regarding the digits if they are present, otherwise the four words with the closest letter sounds to the actual password are applied. For example, **ch** is mostly pronounced either as /**k**/, as in **ch**aracter, **ch**ord, or as /**tʃ**/, as in **ch**icken, **ch**est. Almost all words containing “chi” or “che” are pronounced with /tʃ/ (note exceptions like “chiropractor” /ˈkaɪərəʊpræktə/ **kaay**-roh-præk-tə and “chemistry” /ˈkemɪstri/ (**kem**-ist-ree), but there’s no reliable rule for “cha”, “cho”, and “chu”. The main benefit of using a dictionary attack is that all the honeywords that will be chosen are real passwords generated by users in the past, so the flatness will be very high in this group of honeywords as well.

As aforementioned, there are some passwords that are used commonly by users and researchers have collated them into one list, calling it the worst passwords in the world. Choosing some of them randomly and making a combination of them is a popular procedure for adversaries. Consequently, group three will be generated according to this list.

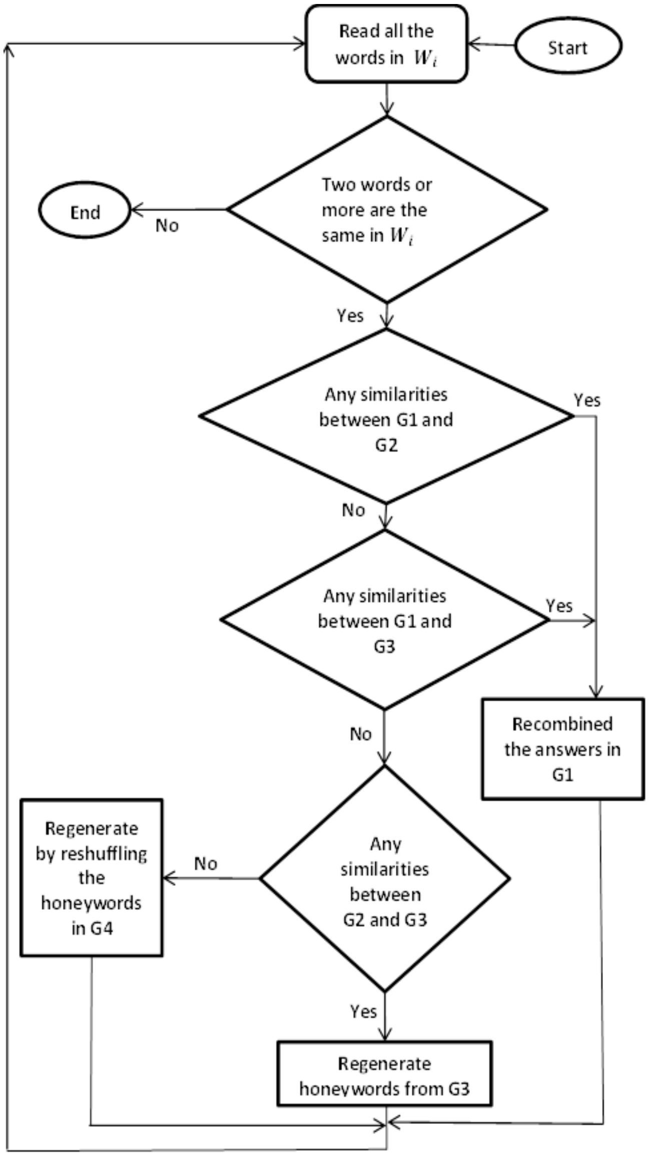
Read all the words in W_i
Start
Two words or more are the same in W_i
End
No
Yes
Any similarities between G1 and G2
Yes
No
Any similarities between G1 and G3
Yes
No
Recombined the answers in G1
Regenerate by reshuffling the honeywords in G4
No
Any similarities between G2 and G3
Yes
Regenerate honeywords from G3

Fig. 2. Illustration of the Algorithm when Two or more Honeywords are the Same in $\boldsymbol{W_i}$.

In addition, a minority of users have a strong password, whereby they select some letters randomly and create a meaningless one. However, most users in this group still select letters or digits from their names and/or personal dates. To avoid these types of passwords, with the proposed method and the goal of flatness, the fourth group is created. Some characters are chosen from the original password and ID and then some digits and letters are generated randomly to be inserted in the honeywords.

XIII. Discussion on Attacks on the New Method

Table II illustrates how an attacker could compromise the password by nominating some words from the honeywords table and then, analysing the results.

TABLE II. DISCUSSION ON THE TYPES OF ATTACKS AGAINST PASSWORDS

	Good Password	**Personal Password**	**Generic Password**
Dictionary Attack	Not Working	Not Working	Not Working
Brute-Force Attack	Not Working	Not Working	Not Working
Password Guessing Attack	If the attacker chooses all good honeywords with the password, then he/she will obtain 11 words, one of which is the real password (1/11). In addition, If the attacker is going to choose just the meaningful words, he will obtain 14 words and fortunately the real password is not one of them (0/14). This type of password is not popular, because most users prefer to use passwords are easy to remember.	If the attacker chooses just the meaningful words. he/she will obtain 15, among which the real password will be included. However, the flatness is very high because the honeywords are coming from real passwords lists; some of them relating to the user, some of them chosen from the dictionary attack and the final set is chosen from the public list of passwords (1/15). As a result, the guessing method will be chosen by the attacker..	If the attacker chooses just the meaningful words, he will obtain 15, among which the real password will be included. However, the flatness is very high because the honeywords are coming from real passwords lists, some of which are related to the user, some are chosen from the dictionary attack, and the final set is selected from the public list of passwords (1/15). As a result, the guessing method will be chosen by the attacker.
Clever Attacker	If the attacker chooses all good honeywords with the password, then he/she will obtain 11 words, one of which is the real password (1/11). In addition, If the attacker is going to choose just the meaningful words he will obtain 14 words, but fortunately the real password is not one of them (0/14). This type of password is not popular to be used due to most users are prefer to use passwords are easy remembering.	If the attacker chooses just the meaningful words, he/she will obtain 15, among which the real password will be included. However, the flatness is very high, because the honeywords are coming from real passwords lists, some of which are related to the user, some are chosen from the dictionary attack and the final set is selected from the public list of passwords (1/15). Now the attacker has just one choice, which is to try to analyse the words and nominate some of them as real password.	If the attacker chooses just the meaningful words, he will obtain 15, among which the real password will be included. However, the flatness is very high, because the honeywords are coming from real passwords lists, some of which are related to the user, some of them are chosen from the dictionary attack, and the final set was chosen from the public list of passwords (1/15). Now the attacker has just one choice, which is to try to analyse the words and nominate some of them as real password.

XIV. TESTBED AND RESULTS

It is a difficult to measure how people are thinking when they are creating a password, because it depends on unpredictable user behaviour. To address this, a testbed engaged with by 820 people was developed to determine whether users can recognise the real password among honeywords. The scenario involved dividing the passwords into three groups: good, personal, and generic. Then, the participants were provided with the W_i, and ask to nominate words that could be passwords, this column being titled “nomination”. The idea behind this step was to ascertain how many people would nominate the real password among the honeywords, and how many words they would choose amongst which they believed the password would be found. Having chosen their words, they were asked to identify the single one that they thought was the real password and if they got it wrong then Intrusion Detection System IDS would trigger attempted intrusion, but if successful access was granted. The first type, namely the good password, was strong, being created with random letters, digits, and special characters. The results showed that this type of password is very strong, as most people who participated in the testbed experiment did not choose it among the honeywords, i.e. no one was able to guess the real password when it was good and random.

The second type of password is the personal password, which was created based on information relating to the users. The testbed revealed that the new method is better than the traditional methods. Finally, with the same scenario, the third type of password, i.e. the generic password, was applied.

Fig. 3 illustrates the results of the strong password for the new method, with the total nominated representing how many words the participants chose, while the frequency is how many people selected a particular amount of passwords. For example, the number of people who nominated 14 words was 224 (27.317%), whereas 11 words were nominated by 78 (9.512%). No one guessed the real password, even if they had nominated it, which shows it was very strong and flat.

Fig. 4 shows the results of the proposed method when the real password is the personal information type and clearly, the number of people who nominated the password amongst their choices increased, being 502 out of 820 (61.219%). Moreover, there were two people who guessed the correct password (0.244%).

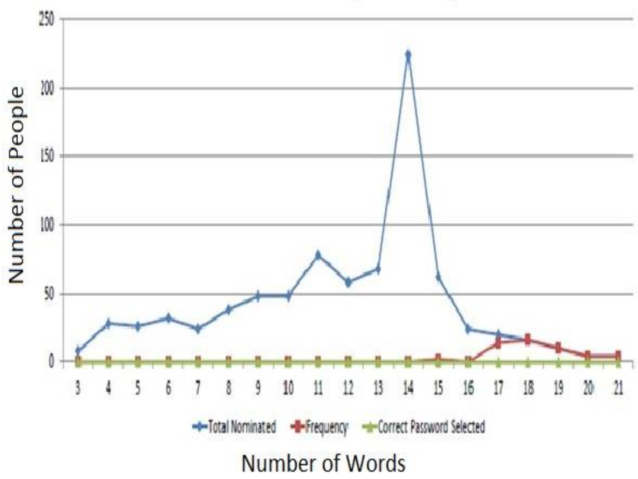

Fig. 3. The Results of the Proposed Method when a Strong Password was Applied.

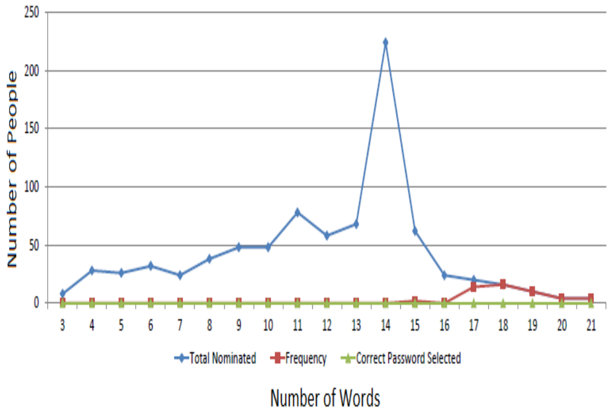

Fig. 4. The Testbed Results when the Real Password Contained Personal Information.

Fig. 5 illustrates the new method when a generic password was the real password and the results show that this type provides the worst outcomes of the three, but the new method still gives better results than with the traditional one. Specifically, the total number who chose this password was 630 out of 820 (76.829%), and it was guessed correctly 21 times (2.561%). This implies that most attackers focus on generic words.

In Fig. 6, showing the outcomes when Chaffing-by-Tweaking was applied in the testbed, it is clear that the number of participants guessing the real password was very high, standing at 794 times out of 820 (96.829%), whilst the number who nominated was 812 (99.024%). Moreover, most people nominated just one or two words out of 25 (3.048%) in W_i and no one nominated more than six, which suggests that many were confident they from the beginning which was the correct password.

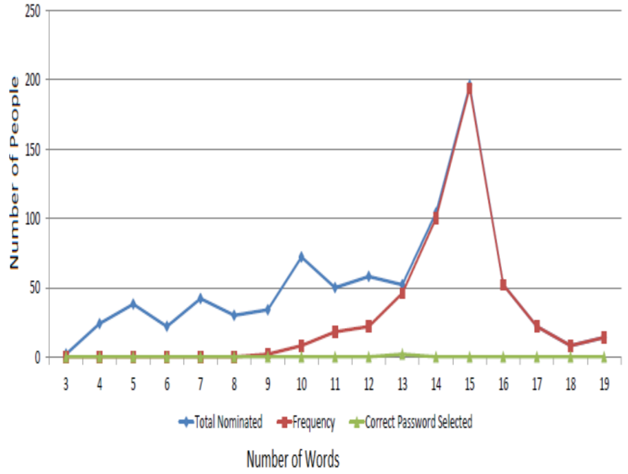

Fig. 5. The Testbed Results for the Proposed Method when the Real Password is Generic.

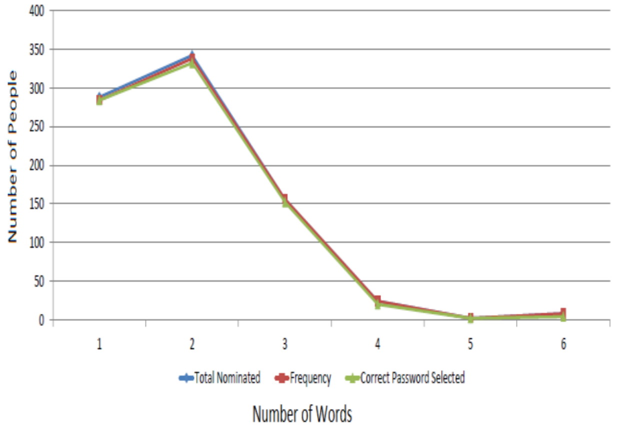

Fig. 6. The Testbed Results for the Traditional Method of "Chaffing-by-Tweaking".

In Fig. 7, the results for the traditional method of Chaffing-by-Tweaking-Digits are shown. This method provides slightly better results than Chaffing-by-tweaking in that the password was guessed correctly 756 times out of a possible 820 (92.195%). To give an example of how the proposed method generates the honeywords, in Table III the password is “Ujemgzae91#e”. Clearly, the first row contains honeywords generated based on personal information, while the second row has those created based on the worst passwords list. The rest of the table was generated by shuffle the letters and digits. A dictionary attack was not used in this table, because no word is similar this password.

Table IV illustrates an example when the testbed was applied with the generic password, “password222”, being drawn from the list of worst passwords. The honeywords in the second row were generated based on a dictionary attack.

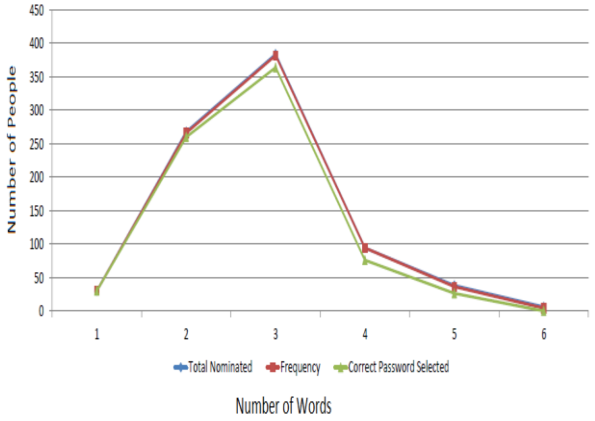

Fig. 7. The Testbed Results for the Traditional Method of "Chaffing-by-Tweaking-Digits".

TABLE III. TESTBED WITH THE NEW METHOD AND A GOOD PASSWORD

Prestol#70	Jordy$86	Steves@75	Mechaill$81	Anna^1945
Liverpool@2005	Football&1234	Password*1111	Music@6666	bond@007
Booboo&75	Love&2014	Mustang@16	Zme1qo@55req	Epalm#1999ks
Pufna*37xy	Msac^hs31	Neadjg_69	Vlpheo$10r	Kp#12zxme
Ltcbas!00j	Tg36$ewba	**Ujemgzae91#e**	Rpnq#fxg	Lsczyr&12

TABLE IV. TESTBED WITH THE NEW METHOD AND A GENERIC PASSWORD

StationRoad1960	Church2016	Morgan2010	Stevs1958	Andy2000
Alunaliceza	Andralice2004	Anasialice1977	Anaalice85	Hello131313
Nicholas123	Andrew 1212	**Password222**	Welcome777	Alice1974
ElArzd204	O9lefcm7ss	Oxsr15dox	Z7erpmc0	Enm12q
Movxg20w	Qica12r00	Hvagjr4193	Nlpqroo1870	Zaqu2w88

XV. CONCLUSION

In this paper, a new honeywords generation method has been proposed. This method was developed to overcome the problems that exist with the traditional methods. The proposed method is based on personal information, dictionary attacks, the worst password list (generic passwords) and shuffling the characters. User behaviour is the underpinning principle the new method, because creation of the passwords differs from one user to another. Some limitations regarding the extant honeywords methods were mentioned in this have been discussed and these have been overcome by the proposed method have been explained. A testbed has been applied to obtain the results using 820 participants and these have shown that the new method is better than the traditional ones.

REFERENCES

[1] S. M. Gurav et al., "Graphical password authentication: Cloud securing scheme," in 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies, 2014, pp. 479-483.

[2] D. Mishra et al., "A secure password-based authentication and key agreement scheme using smart cards," Journal of Information Security and Applications, vol. 23, pp. 28-43, 8, 2015.

[3] S. Houshmand, S. Aggarwal and R. Flood, "Next Gen PCFG Password Cracking," IEEE Transactions on Information Forensics and Security, vol. 10, pp. 1776-1791, 2015.

[4] B. Gurung et al., "Enhanced virtual password authentication scheme resistant to shoulder surfing," in 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI), 2015, pp. 134-139.

[5] Siqiong Fan, Zhenfu Cao and Xiaolei Dong, "Cryptanalysis and improvement of a smart card-based identity authentication scheme," in ICINS 2014 - 2014 International Conference on Information and Network Security, 2014, pp. 152-157.

[6] J. Ma et al., "A study of probabilistic password models," in 2014 IEEE Symposium on Security and Privacy, 2014, pp. 689-704.

[7] S. M. Taiabul Haque, M. Wright and S. Scielzo, "Hierarchy of users′ web passwords: Perceptions, practices and susceptibilities," International Journal of Human-Computer Studies, vol. 72, pp. 860-874, 12, 2014.

[8] Dr. Ari Juels RSA, Professor Ronald L. Rivest MIT. Dr. Ari Juels RSA, Professor Ronald L. Rivest MIT., " For Stronger Password Security, Try a Spoonful of Honeywords ," 2013.

[9] D. Vishwakarma and C. E. V. Madhavan, "Efficient dictionary for salted password analysis," in 2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2014, pp. 1-6.

[10] H. Kumar et al., "Rainbow table to crack password using MD5 hashing algorithm," in 2013 IEEE Conference on Information & Communication Technologies, 2013, pp. 433-439.

[11] H. M. Ying and N. Kunihiro, "Decryption of frequent password hashes in rainbow tables," in 2016 Fourth International Symposium on Computing and Networking (CANDAR), 2016, pp. 655-661.

[12] E. Martiri, B. Yang and C. Busch, "Protected honey face templates," in 2015 International Conference of the Biometrics Special Interest Group (BIOSIG), 2015, pp. 1-7.

[13] M. J. Bhole, "Honeywords: A New Approach for Enhancing Security," International Research Journal of Engineering and Technology (IRJET), vol. 02, pp. 1563, 2015.

[14] L. Catuogno, A. Castiglione and F. Palmieri, "A honeypot system with honeyword-driven fake interactive sessions," in 2015 International Conference on High Performance Computing & Simulation (HPCS), 2015, pp. 187-194.

[15] N. Chakraborty and S. Mondal, "A New Storage Optimized Honeyword Generation Approach for Enhancing Security and Usability," arXiv Preprint arXiv:1509.06094, 2015.

[16] A. Jesudoss and N. Subramaniam, "A SURVEY ON AUTHENTICATION ATTACKS AND COUNTERMEASURES IN A DISTRIBUTED ENVIRONMENT,".

[17] E. I. Tatli, "Cracking More Password Hashes With Patterns," IEEE Transactions on Information Forensics and Security, vol. 10, pp. 1656-1665, 2015.

[18] L. Wu, X. Du and J. Wu, "Effective Defense Schemes for Phishing Attacks on Mobile Computing Platforms," IEEE Transactions on Vehicular Technology, vol. 65, pp. 6678-6691, 2016.

[19] I. Uusitalo, J. M. Catot and R. Loureiro, "Phishing and countermeasures in spanish online banking," in Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference On, 2009, pp. 167-172.

[20] S. Kharod, N. Sharma and A. Sharma, "An improved hashing based password security scheme using salting and differential masking," in 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2015, pp. 1-5.

[21] Y. Li, H. Wang and K. Sun, "A study of personal information in human-chosen passwords and its security implications," in IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1-9.

[22] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in 2012 IEEE Symposium on Security and Privacy, 2012, pp. 538-552.

[23] M. Burnett, Perfect Password: Selection, Protection, Authentication. Syngress, 2006.

[24] Available: http://www.whatsmypass.com/the-top-500-worst-passwords-of-all-time.

[25] I. Erguler, "Achieving Flatness: Selecting the Honeywords from Existing User Passwords," IEEE Transactions on Dependable and Secure Computing, vol. 13, pp. 284-295, 2016.

[26] Ibrahim Salim M. and T. A. Razak, "A study on IDS for preventing denial of service attack using outliers techniques," in 2016 IEEE International Conference on Engineering and Technology (ICETECH), 2016, pp. 768-775.

[27] Z. Tan et al., "A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis," IEEE Transactions on Parallel and Distributed Systems, vol. 25, pp. 447-456, 2014.