USING CRYPTOANALYSIS POLICIES AND TECHNIQUES TO CREATE STRONG PASSWORD

Abdulrahman H. Majeed* and Sajaa G. Mohammed
Department of Mathematics, College of Science, University of Baghdad , Baghdad Iraq .
*Email: ahmajeed6@yahoo.com

ABSTRACT : *Password is still the widespread way used for securing the computer systems. In fact, a great number of organizations and institutions trust in the use of passwords, and for that reason, they strictly enforce their users to have secure passwords. These organizations typically attempt to carry out security by commanding users to come after password creation policies. However, password creation policies make an effort to assist computers' users to make secure passwords, they are in general not very efficient and have a tendency to disappoint users. Rule based, for instance, are the most common policies. They have been made known for their perfect limitations. Although, these policies impose their users on following certain rules such as a least possible length, symbols or numbers, they are not compliable with each other. In the current paper, the researchers refer to unlike password creation policies as well as password checkers by using Crypto Analysis policies and methods to find well-protected password that seek to assist users to make secure passwords in order to address their issues. As a matter of fact, the work involves some coding modules; the first indicates to implement recursive byte array transformation (i.e., create the total probable keys), the appropriateness of this modules was examined experimentally. The second cryptanalysis technique step is the use of the password strength tester (checker), the performance of this system has also been experienced. Furthermore, the message digest hashing function MD5 algorithm is made full use to reform password and produce further security on password OS file system. The reformed passwords have been examined by using old-dated traditional MD5. It used with password brute-force attacking. Moreover, an instrument named EP has been advanced; it is to be discussed in details in the present paper to show the way of use the secure passwords through applying such metrics. Indeed, the done crypt analysis system has been designed and applied, as well as results were examined. Standard cracked password data sets were used as test materials to examine the performance of the recommended cryptanalysis pattern; the results show that the competence of offered pattern gives one support in comparison with the other cryptanalysis pattern.*

Keywords: Cryptanalysis, Ciphering, Password, Cracking, Cryptology, Cryptography, Enhanced Passwords, Analyzing And Enhancing Password,

1. INTRODUCTION

Multiple accounts, and not to write their passwords down. Yet users have many passwords and are expected to create a password for every new service. Often, users are required to change their passwords at regular intervals. Taken as a whole, these requirements are difficult, if not impossible, for users to meet. So they needed always generate secure password [1]. Secure password generation is complicated by the tradeoff between developing passwords which are both challenging to crack and usable. Truly random passwords are difficult for users to memorize, and user-chosen passwords may be highly predictable. This limitation has led to the use and advocacy of *password creation policies* that purport to help the user in ensuring that the user chosen password is not easily breakable Password policies attempt to mediate between two goals which are challenging to crack and usable these two goals accomplish by forcing users to incorporate additional complexity into a password, such as by mandating the user include an odd character or use passwords of some minimal length. However, these policy mechanisms are hampered by an ill-defined understanding of their actual effectiveness against real attack techniques, and by circumvention strategies employed by the users [2]. The most prevalent password creation policy is the *rule-based approach* wherein users are given rules such as minimum length of eight characters and must contain an upper case letter and a special symbol. It has been shown by several authors that this approach by itself is not very effective [2, 3]. A second type of password creation policy can be termed the *random approach* where an effectively random string is given by a system to the user. Clearly the random approach has the problem that the given string is in general non memorable so the purpose of having a password that can easily be remembered is defeated. This paper presents a third approach which is to have a system that *analyzes* and checker a user proposed password Using Cryptoanalysis Policies and Techniques in order to detect wither the password is a strong and good for using or not . We show that empirical analysis based on trying to crack passwords using probabilistic techniques [4] can be adapted to analyzing the strength of passwords. We show how the associated probabilistic context-free grammar can be used to build a realistic reject function that can distinguish between strong and weak passwords based on a threshold probability. An obvious component of an empirical analysis might be to have a dictionary of popular passwords and ensure that the modified password is not one of these. But a more important consideration is to show that the modified password is not likely to be cracked using any technique as we are able to do. The black-listing approach is automatically subsumed by our approach simply by the choice of dictionaries. Note that we are interested in protecting against offline attacks where an attacker has obtained a set of hashes (and likely user names) and desires to break as many passwords as possible in a reasonable amount of time .We illustrate the effectiveness of our prototype system, called enhanced Passwords (EP), through a series of experiments on three lists of disclosed passwords. EP was able to modify weak passwords and strengthen them with strong passwords that were within an edit distance of one from the user passwords.

2. BACKGROUND & PREVIOUS WORK

In fact, several studies exerted efforts to outline various aspects on how users select passwords. Riley [5] conducted a study on 315 participants. He found that about 75% of them said that they have a set of fixed passwords use repeatedly. While almost 60% said that they used to use complicated passwords depending on the nature of the website they use. Stone-Gross *et al.* [6] conducted another

study by collecting around 298 thousand passwords from the Torpig botnet. The researchers found that almost 28% of users reused their passwords and succeeded to crack over 40% of the passwords in less than 75 minutes. This study shows that abstaining strong passwords for unessential websites, including social networking websites is probably being necessary for websites, such as online banking. On the other hand, most organizations and websites apply a rule-based method in commending or implementing password policies. In another study, Shay *et al*. [7] indicated that users were unhappy about varying the previous password creation procedure to a stricter one. This process took an average of 1.77 tries to select a new password recognized by the system based on a new password creation policy that recently founded. In addition study conducted by [8], it is shown that varying and inconsistent recommendations make unreliable advice for users. The U.S. NIST guideline [9], the basis for all rule-based strategies, offered a rule-based method that practices the notion of Shannon entropy for guessing password strength based on recommended values of the components of the passwords. On the other hand, Zhang *et al*. [10] showed that attackers can effortlessly get access to accounts by catching the account's' previous passwords. The researchers in their study recommend that at least 41% of passwords can be broken offline from previous passwords in seconds and merely five online passwords meet the requirements to break 17% of accounts. In a recent study made by [11], though at the present time users realize the significance of secure behavior, they still think that it is difficult to deal with password creation policies, and they hardly ever alter their passwords due to the frustration of forming a new password along with their struggle in remembering it. In their study, Charoen *et al*. [12] and Adams and Sasse [13] state that users are not even agree about the need of abstaining well-protected password. The reason behind choosing insecure passwords is because they are not familiar with the way of creating secure ones. Studies [14] show that even limiting password creation policies do not have an influence on the use of significant information in passwords. They do not reduce reuse of these passwords. The use of the previous passwords can subject users with different types of attacks such as phishing, key-logging and targeted attacks [15]. Furthermore, Shay *et al.* [16] in the different study show that the more limited and complicated policy, the less user-friendly is. Some others [17, 18] have explored the use of the random password generation method. The main problem in these studies focuses on the usability of the password for the user since such password has normally no context for the user and hardly to remember. Moreover, Forget *et al*. [19] studied the memorability of passwords by randomly inserting or exchanging the permanent number of characters in a selected password. Their study displayed that users who confirmed their changed passwords once could recall it effortlessly (passwords without change). However, they did not change a methodology for analyzing the strength of these passwords. Producing secure passwords is a tradeoff between generating passwords that are hard to crack and use. Although, some studies on creating suitable passwords [15, 18] attempt to provide an understanding on how several policy factors create passwords easier, memorable, and functional, none of them have been applied. However, the work of Verheul [18] is considered a perfect example on trying to understand the relationship between countless entropy measures to build secured passwords. Verheul showed the way to build sensible short secure passwords based on calculating the Shannon entropy with the norms of the min entropy and estimating entropy. Nonetheless, there was no attempt in the current paper to study the usability or memorability of the passwords or how to change user proposed password. The analyze-modify method also has certain related history. The analysis is typically a simple way to define whether a password is weak as checking against a dictionary or not. Note that, in reality, this is not certainly considered satisfied condition for a password to be strong. Current proactive password checkers commonly keep an eye on a black-listing approach. For example, see Yan [20] and Spafford [21]. However, simple black-listing approaches in general include some problems in any sophisticated dictionary based attack. Perhaps, there are relevant studies related to the approach applies in the current paper as he study conducted by Schechter *et al*. [22] that include the popularity of passwords. Schechter *et al*. [22] propose to build an oracle for current passwords that are obtainable to the Internet-scale authentication systems. They suggested that the popular passwords are rejected and the main thrust of their work is to devise a way to competently store countless numbers of popular passwords that would be prohibited. In their study, a question is posed on how to use the oracle without enlightening the actual password to attackers while asking online. The researchers of the present paper show their technique that gets around this problem. More recently, [23] search measuring the strength of passwords by using a Markov approach. Our approach show first how generate password then the second step is how cracking system can in fact be used for analyzing passwords. Once such an analysis is done. We next review some classical mathematical notions that have been proposed to measure strengths of passwords. After this, we review the probabilistic password cracking approach that we use for doing enhanced password (EP).

1. Different Policies and Advice on Password Creation

A secure password should include punctuation marks and/or numbers. You should mix capital and lowercase letters. Include substitutes such as "$" for "S" and "0" for "O". You should not use personal information like names and birth dates. Do not use dictionary words, keyword patterns and sequential numbers. Do not use repeating characters (aa11) [24].

Use at least 7 characters. The more characters your password contains, the harder it is for someone to guess it. A long but simple password can be safer than a short, complex one — and often easier to remember [25].

Strong passwords are important protections to help you have safer online transactions. The main keys to password strength are length and complexity the ideal password is long and has letters, punctuation, symbols, and numbers. The difficulty, of course, is remembering them especially when you need a password for every utility bill, bank account [26].

Table 1 illustrates a small sample of current password policies. As you can see there is no uniformity in these policies and the reason may be that there is no proof of effectiveness of any of these approaches for password security [27].

Table 1: Different password policies [27]

Organization	Length	Number	Special Char	Other
Chase.com	7-32	1+	Not allowed	At least one letter
Bank of America	8-20	1+	Certain ones allowed	At least one letter and certain special not allowed
Ets.org	8-16	1+	At least one	At least one uppercase and one lowercase letter
Banana Republic	5-20	ok	Not allowed	-

As indicated there are many different types of advice for creating a secure password. In fact, some of the recommendations, even contradict each other, which make them unreliable for users. One example of clearly contradictory advice is not including the website name (many sites suggest this) in the password vs. including it (as suggested by WikiHow). In the next section we discuss some existing password checking systems designed to help users develop better passwords.

With respect to password security, it is not only essential to have a secure system to store user's passwords, but it is also important how users create and use their passwords. The number of accounts for a single user is growing. The result of a survey of 2000 users has shown that a typical user has about 25 online accounts and one in four users uses a single password for most of their accounts [28]. Florencio *et al.* [29] showed that on average a user has 6.5 passwords and each password is typically being reused across 3.9 different websites. Enforcing complex password policies makes it harder for users to create memorable passwords. Because of this, many users reuse the same password for multiple accounts against experts' advice. This reduces the security tremendously since when an attacker obtains a password; it is often tried on many different websites. Thus, no matter how secure a service is; the security of it can be reduced because of its users' actions. As more and more websites replace usernames with email addresses, it becomes much easier for attackers to attack and access our accounts. Users are often forced to change their password on a given account because of a threat or simply due to expiration policies. In these situations users are more likely to apply only slight changes to their previous password instead of creating a new one. Furthermore, users also tend to use a password with slight modification across different websites. Having different password creation policies for different websites might prevent users from some reuse of the same password (an unintended consequence), but it does not prevent users from using passwords that are very similar. A study by Shay et al conducted on 470 University students, staff and faculty has shown that 60% used one password with slight changes for different accounts [30].

In [31] the authors examined leaked password sets and found that users often do simple tricks to slightly change their passwords and to work around different password policies.

2. Password Strength / Weakness

Passwords are a notoriously weak authentication mechanism. Users frequently choose poor passwords. An adversary who has stolen a file of hashed passwords can often use brute force search to find a password p whose hash value ***H (p)*** equals the hash value stored for a given user's password, thus allowing the adversary to impersonate the user [32]. There have been several attempts for measuring password security and developing techniques (such as hardening) for passwords to be both strong and usable for the user [33]. Yuan et al conducted an experiment involving 400 first-year students at Cambridge University. They found that users have difficulty memorizing random passwords and that mnemonic passwords could provide both good memorability and security [34]. Nowadays, many systems encourage users to create a mnemonic phrase-based passwords. For creating a mnemonic password, the user chooses a memorable phrase and uses a character (usually the first letter) to represent each word in the phrase. Organizations usually suggest mnemonic password as a stronger password because first, you cannot find the mnemonic password in dictionaries used for password cracking, and second, a user can incorporate different types of characters such as numbers and punctuations easily in their chosen password [33].

3. Cryptanalysis

Cryptanalysis is a study of how to compromise (defeat) cryptographic mechanism. There are two classes of key-based encryption algorithms: symmetric (or secret-key) and asymmetric (or public-key) algorithms. Symmetric algorithms use the same key for encryption and decryption, whereas asymmetric algorithms use different keys for encryption and decryption. Ideally, it is infeasible to compute the decryption key from the encryption key [35]. Cryptanalysis is the methods to attack cryptographic protection. There are several ways to achieve this goal. A cipher is breakable if it is possible to determine the plaintext or key from the ciphertext, or to determine the key from the plaintext-ciphertext pair [36].

Computationally secure is established with the two criteria meet at the same time first one is the cost of breaking the cipher exceeds the value of the encrypted information. And the second one is the time required to break the cipher exceeds the useful lifetime of the information [37]. In this paper, we use Cryptanalysis Policies and Techniques in order to create strong password.

4. Proposed Work (Cryptanalysis Schemas System)

The suggested schemes consist of some coding modules. The first coding modules are indicated to perform recursive byte array permutation (i.e., generate all possible keys), their suitability was investigated experimentally. The second cryptanalysis method step is the application of the password strength tester (checker), this system performance has been tested. The message digest hashing function MD5 algorithm is exploited to regenerate password, and used to give more security on password OS file system. The regenerated passwords have been created using traditional MD5. The third cryptanalysis method is password brute-force attacking.

The full cryptanalysis system has been designed and implemented and their results were analysed. Standard

cracked password data sets were used as test materials to investigate the performance of the suggested cryptanalysis scheme; the results indicate that the efficiency of the proposed scheme is encouraging when it is compared with state of the art other cryptanalysis scheme.

In the current research work several programming systems (6 programms) have been implemented and tested. These systems will demonstrate in details in the next sections.

4.1 Generate passwords

I. Generate password using Simple Recursive Permutation

This section is related to the password Generator Possibilities method. Takes in a string and splits out all possible permutations of the inputted characters using a simple recursive routine.

.

Table (2): A simple routine to generate all possible combinations of a given list of numbers

Algorithm (1): Generate All Possible Combinations of a Given List of Numbers
Goal: Takes in a string and spits out all possible permutations of the inputted characters using a simple recursive routine.
Input: Letters[] // String of characters
Output: Permut // List of all Possible Permutations of Letters characters

Algorithm Steps:
Step 1: If (Length (Letters) =1) Then
Print Built & Letters
Go to Step 3
Step 2: For all i Do {where 0 < i > Length (Letters) }
st ← Letters[i]
stmo ← Letters[i – 1]
stpo ← Letters[i + 1]
Letters ← stmo & stpo
Built ← Built & st
Go to Step 1
Step 3: Go to Step 2

II. Message Digest Algorithm

The MD5 algorithm is a widely used hash function, producing a 128-bit hash value. Although MD5 was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities. It can still be used as a checksum to verify data integrity, but only against unintentional corruption.

Like most hash functions, MD5 is neither encryption nor encoding. It can be cracked by brute force attack and suffers from extensive vulnerabilities as detailed in the security section below.

III. Length / Entropy

Effective evaluation of password strength requires a proper metric. One possible metric is information entropy, has been stated in a paper presented to the 2012 IEEE Symposium on Security and Privacy [38].

Eight characters is the minimum length for a password to be secure if it takes advantage of all the potential character types. With 26 possibilities from lower case, 26 from uppercase, 10 from numbers and 12 from the full set of symbols this means every keyspace has **95** possible entries.

Note: "`~!@#$%^&*()-_=+[{]};:',<.>/?\|" A number of these cannot be used with password systems. A total of 32 symbols are present.

The total number of possibilities (TNP) could be calculated using the following equation:

$$TNP = LS^{DM} \text{ ,.....(3.1)}$$

Where LS denoted to Length of string List and DM denoted to number of digit manipulation. The probability could be found using the following equation:

$$P = \frac{1}{TNP} \text{ ,.....(3.2)}$$

From this, it could determine that the possible combinations for a 8-letter password are 63^8. This means there are just less than 2.4×10^{14} combinations of passwords and probability is $\mathbf{4\times10^{-15}}$. Therefore the required time to crack if a computer could test 100 million (possibilities / second) is about 297 day (see figure (3).

4.2 Password Strength Checker Method

Now with common list of passwords database and customized analysis. With the fast growing of the Internet, the use of passwords has become very important for all of us. But not a lot of people uses strong password (which mean it cannot be easily cracked).

So, this method was programmed work has been made this utility to test user passwords to see if they are strong. If the user just has to enter a password and the program will tell you how is it strong and it will also tell how much time it would take to crack this password. Table (3) shows in details the password Strength Checker sub procedure.

Table (3): Password Strength Tester script written in Visual Basic 0.6

```
Start Sub
Dim sAnalysis As String 'The variable to store the customized analysis
lLenPass = Len(tPass.Text) Getting the length of the password
sPass = tPass.Text Getting the password
Checking if the password is in the common list of passwords
If InStr(1, PasswordList, ";" & sPass & ";") <> 0 Then
   commonFlag = True 'Setting the commonpass flag to true so the program will consider it
End If
Seeking for uppercase letters
For i = 1 To lLenPass
   If UCase(Mid(sPass, i, 1)) = Mid(sPass, i, 1) And IsAlpha(Mid(sPass, i, 1)) = True Then upperFlag = True: Exit For
Next i
Seeking for lowercase letters
For i = 1 To lLenPass
   If LCase(Mid(sPass, i, 1)) = Mid(sPass, i, 1) And IsAlpha(Mid(sPass, i, 1)) = True Then lowerFlag = True: Exit For
Next i
'Seeking for numbers Chr 048-057
For i = 1 To lLenPass
   If Asc(Mid(sPass, i, 1)) <= 57 And Asc(Mid(sPass, i, 1)) >= 48 Then numberFlag = True: Exit For
Next i
Seeking for char other than those ranges 065-090 097-122 048-057
For i = 1 To lLenPass
   tmpchar = Asc(Mid(sPass, i, 1))
   If tmpchar < 65 Or tmpchar > 90 Then
      If tmpchar < 97 Or tmpchar > 122 Then
         If tmpchar < 48 Or tmpchar > 57 Then
            specialFlag = True
            Exit For
         End If
      End If
   End If
Next i
Now calculating an index considering all the Flags values
Calculating possibilities
If upperFlag = True Then
   range = range + 26: flagtot = flagtot + 1
Else
   sAnalysis = sAnalysis & "Weakness: There's no uppercase letters in your password" & vbCrLf
End If
If lowerFlag = True Then
   range = range + 26: flagtot = flagtot + 1
Else
   sAnalysis = sAnalysis & "Weakness: There's no lowercase letters in your password." & vbCrLf
End If
If numberFlag = True Then
   range = range + 10: flagtot = flagtot + 1
```

```
Else
   sAnalysis = sAnalysis & "Weakness: There's no numbers in your password." & vbCrLf
End If
If specialFlag = True Then
   This is an arbitrary value for number of special printable characters
   range = range + 30: flagtot = flagtot + 1
Else
   sAnalysis = sAnalysis & "Weakness: There's no special chars in your password." & vbCrLf
End If
If lLenPass < 8 Then
   sAnalysis = sAnalysis & "Weakness: Your password length is under 8."
End If
If commonFlag = True Then
   Number of possibilities is the number of passwords in the common list
   dPossib = PasswordNum
   sAnalysis = "MAJOR WEAKNESS: Your password is detected as one of the common passwords used by users. If a hacker wants to crack your password, he will first try this list." & vbCrLf & sAnalysis
Else
   dPossib = range ^ lLenPass
End If
Calculating the time it will take
dTime = (((dPossib / (NUMBERPERSECOND)) / (365 * 24)) / 3600) / 2 'The /2 is because it takes approximatly the half of the test to find the pass
Setting the Progress Bar: Note that you can customize the const for how much years to crack you consider to be weak, ok, strong
If dTime >= CVRYSTRONG Then
   Progress.Value = 100
ElseIf dTime >= CSTRONG Then
   Progress.Value = 75
ElseIf dTime >= COK Then
   Progress.Value = 47
ElseIf dTime >= CWEAK Then
   Progress.Value = 23
ElseIf dTime <= CVRYWEAK Then
   Progress.Value = 1
End If
Formatting the time it will take
lMes.Caption = "years."
If dTime < 1 Then
   dTime = dTime * 365
   lMes.Caption = "days."
   If dTime < 1 Then
      dTime = dTime * 24
      lMes.Caption = "hours."
      If dTime < 1 Then
         dTime = dTime * 60
         lMes.Caption = "minutes."
         If dTime < 1 Then
```

```
            dTime = dTime * 60
            lMes.Caption = "seconds."
          End If
        End If
      End If
    End If
tTime.Text = dTime display the formatted time
If sAnalysis = "" Then sAnalysis = "No weaknesses found on your password!"
tAnalysis.Text = sAnalysis 'display the analysis
End Sub
```

4.3 Methods of Password Cracking

A great deal of methods exists for gaining access to systems by bypassing the standard security settings. These methods may range from executing small portions of code using exploits on the vulnerable machine to gain complete control via backdoors or providing illicitly obtained but legitimate login information .

In this paper the work focuses on the options regarding password security and cracking methodology is Brute Force attacks.

I. Offline Attack Phase (Brute Force Attack)

Permutation with Repetitions Algorithm

This method permutated a byte array of a given size using a given byte set. Note "repetitions" means the same character can be repeated in the permutation not that there are repeats of the permutation. Compile before testing it will be slow in the IDE.

A brute force attack performs and exhaustive search on the hash or hashes by calculating the hash of each and every string combination for a chosen character set and string length. The calculated hashes compared with the hashes to be recovered until a match is found or the attack is finished.

When attempting a brute force attack on more than 10 characters the time needed to perform it becomes infeasible because of the huge key space and the exponential growth in possible strings with the addition of each extra character. This thesis refers to the time spent on calculating each unsuccessful match string as noise.

It is important to reduce "noise" in order to end up with an efficient attack method that will produce the most recovered hashes in the least time with the least computations. By this definition, a brute force attack is extremely inefficient because it attempts many strings that are unlikely to produce a match.

4.4 Analyzing Passwords

I. Analyses and Enhancing Password using AEP

In this section, our work will review on developed system will call "**Analyzing and Enhancing Password**" **AEP**. The key to a good password checker is the ability to help a user create a secure password while ensuring the password is easy for the particular user to memorize. Both of these aspects are important since it is very easy to develop a policy that results in strong passwords (using random password generators) that are particularly unusable. In current approach it will used Different Policies and Advice on Password Creation that shown previously to apply all the advice and benefits in order to create strong passwords.

5. System implementation

The system was implemented using visual basic language. The system was attached to a network in order to ensure accurate conditions as close to real-world as possible. Two computers were used based on the specifications provided in **Table 4**

Table 4 computer specifications

	Attacker PC	Server PC
CPU	Pentium D	Pentium D
RAM	2GB	1GB
HDD	160GB	80GB / 160GB
NIC speed	100mbps	100mbps
IP Address	192.168.1.83	192.168.1.85
GPU	Radeon X550	N/A
OS	Kali Linux	Debian (Virtual Machine)

The addition of a graphics card in Attacker PC may seem arbitrary but, as brute force attacks require parallel calculation, the GPU is far better than the CPU. Given the exponential increase in processing power of GPUs since the release of the Radeon X550, it is safe to assume any parallel calculations will be faster under today's systems he Server-PC would attempt to simulate an access server in a working environment. It would have minimal security features (given that this thesis discusses security of passwords, rather than preventative measures). A number of user accounts were created, each with incrementally secure passwords and higher permissions depending on their role inside the virtual environment.

6. SIMULATION RESULTS

The first step in the system is generating password. This operation is accomplished by using various techniques that explain in previous sections. Figures (1) show the interface of password generation operation.

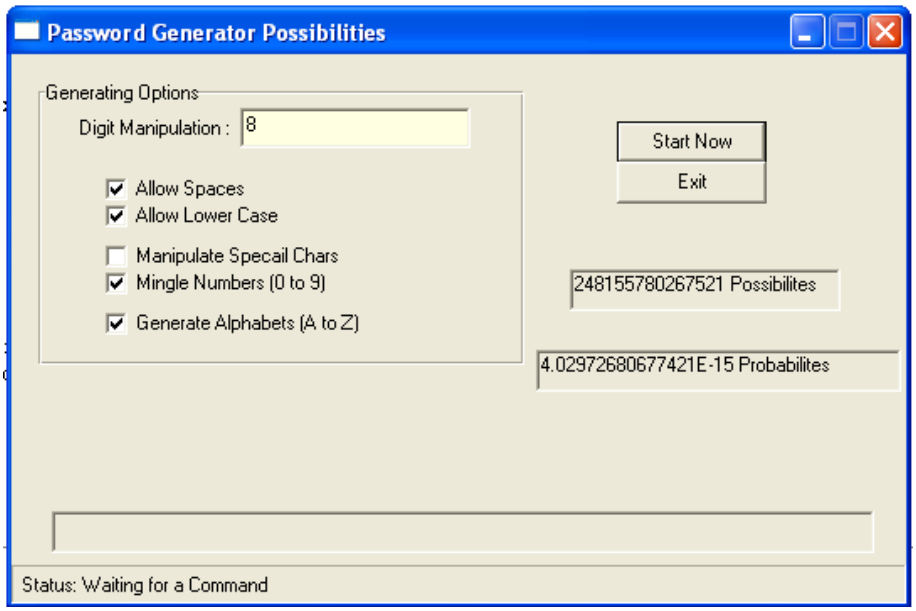

Fig (1): Snapshot of password generator possibilities

The Second step is checking password strength tester (**PST**) In Figure (2) a snapshot of password strength tester (***PST***). The user enters a password such as "***life45!***". ***PST*** calculates the probability of this password which is (**1.48×10^{-13}**) see. Since the probability of this password is greater than the threshold, ***PST*** suggests some options for user to create a new password using the ***PST*** system algorithm and choosing one of the operations randomly. Here for example the suggested password is "academia" which is exists in password data sets. By selecting the new passwords randomly, we are avoiding the possibility of suggesting the same password to different users with the same original password.

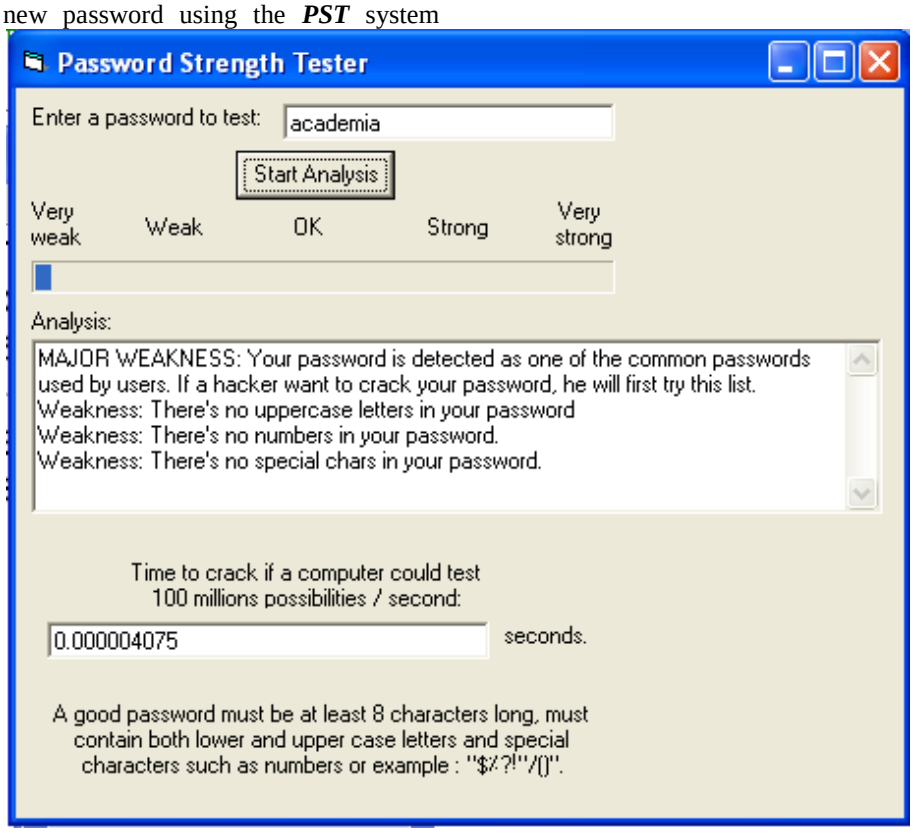

Fig (2): Snapshot of PST suggesting a good password option to the user.

.

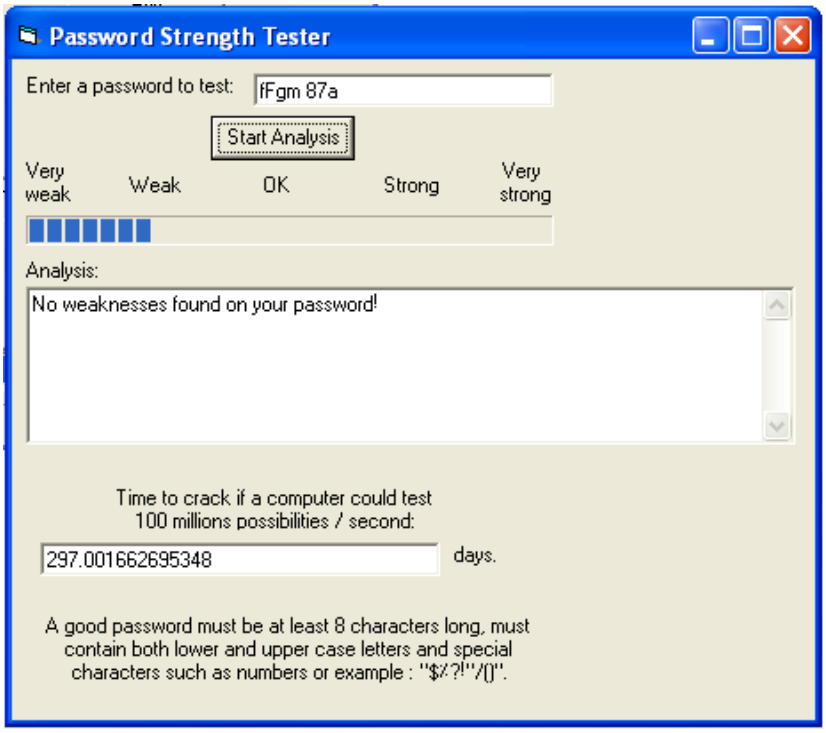

Fig (3): Snapshot of PST calculated the required time to crack "*fFgm 87a*" password

However, at the Passwords^12 Security Conference in December of 2012 Jeremi M. Gosney gave a presentation that stipulated password crackers need more power. He went on to introduce a system that contained 25 AMD Radeon GPUs [38].

Limiting oneself to only a subset of the available symbols can dramatically reduce the strength of the password. Using the alphanumeric subset of 62 characters (with a speed of 2 million attempts per second), the crack would take three years to complete. Using only lowercase and numbers the subset is only 32 characters and would complete within 6 days. Against a more powerful system, such passwords would hold up for mere seconds.

Length can serve to offset this issue somewhat. Each additional character space that needs to be calculated increases the entropy of a password. Even the addition of a single extra letter gives an additional exponent equal to the number of characters in the subset used. For example: Increasing the 32-character subset's password length to 10 instead of 8 increases the cracking time (at 2 million guesses per second) to over 17 years.

When we increase the length of a password using 74 possible characters to we see such an exponential increase in the cracking time that even Jeremi Gosney's system would need 4750 years to break the password when working against MD5 hashes.

After generate passwords step and checking passwords strength step the next step in the system is trying to attack password using brute force method see figure (4)

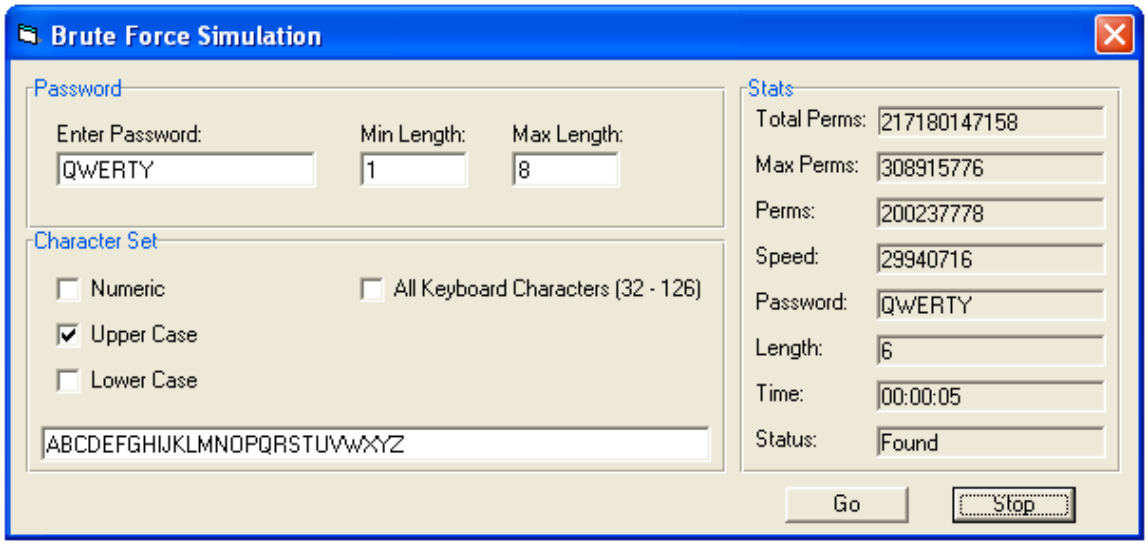

Fig (4): Snapshot of password cracking using Brute force method.

The attack this time is much faster. Running at over 16000 attempts per second (as demonstrated in Figure 4), a much larger password file (in excess of 300 megabytes) was used and in a far shorter time. Additionally, the GPU can be used instead of the CPU to dramatically increase cracking speed. It is this result that lends credence to the idea that password-cracking requirements have stayed stagnant while cracking techniques have advanced.

Even on a standard processor, a password with 8 characters is surprisingly weak. Sixteen thousand attempts a second would take several years to break into an 8 character password. However, when running on GPU the dramatic increase in speeds renders an 8-letter password mostly useless. The actual code stays from this concept slightly as each element of the count array can have because only hold one value, it is merely incremented or reset according to the last value. Also the reel that stores the units (the reel on the right) is not used.

At the first tested this algorithm, it must admit didn't expect it to be so fast, having said that don't be disappointed if the input a long password with all the characters selected and the program just keeps churning away. Brute forcing a password could take years in some cases. At this point in the experiment, brute force options were considered. However, there were several drawbacks that prevented us from running them. Primary among them was the available hardware. While GPUs have drastically increase the speed of cracking passwords, the devices available in our testing environment would not have provided the required power. The cost of obtaining such devices was also unfeasible for testing purposes, as for a reliable result, high end graphics cards would be required.

A second drawback was in the speed of the devices. Running with substandard hardware, if we could attain 200,000 guesses a second, an 8 letter password containing only lowercase letters would still take 12 days to run. While this is not an unreasonable runtime in real world situations, the number of passwords we needed to test combined with the complexity made testing brute force methods impossible. For the purposes of this thesis, however, we can consider theoretical values. Brute force attacks can be considered more effective, but far slower than dictionary attacks. Both attacks will be subject to similar network traffic when used online and thus our online results only need consideration when it comes to dictionary attacks.

Brute force attacks excel when it comes to offline password cracking on dedicated hardware. As such, we can presume that any system designed for these attacks will provide far better results than our offline dictionary attack.

A loop counting in a given base is the mechanism for this permutation with repetition algorithm. It's easy to figure out if you think of a mile-o-meter (the small group of reels found on a speed-o-graph) that records a vehicles mileage.

As you know the reels are numbered from 0 – 9 (base 10) and when a reel rotates and reaches 0 again the next reel to the left is incremented by 1. Each element of the count array (m_bCountArr) is a virtual reel, but numbered in the base (integer representation) according to the chosen character set. So, if the chosen character set was "Lower Case" the base would be 26 and each reel would be numbered 0 - 25.

Using each current reel value as an index to the character set (m_bByteSet) it is then possible to permute the password array (m_bPwrdArr) in the correct sequence.The last step in system is Enhancing Password using (EP**) ,** the EP is worked after analyses the password in order to detecting wither the entered password is(Very Strong , Fairly Strong , Medium, Weak , Very weak)depending on the policies that shown in previous sections EP enhanced password if the password is a weak and needed to enhanced see table (5)

Table 5 User and password setup values

Name	User Name	Role	Password	Strength
Dr.Abdulrahman Hammed	Ahammed-6789	Head of Math. Dep.	No weaknesses found on your password!	Very Strong
Dr.Ahmed Molied	amolied12d45	IT Dep.	1-There's no uppercase letters in your password 2-There's no special chars in your password.	Fairly Strong
Ali Mhommed	Amhommed9	IT intern	There's no special chars in your password	Weak
Hassan Ahamed	Hahamed12	CEO	There's no special chars in your password.	Medium
Najlaa Azher	Nazher	Accounting	1-There's no numbers in your password. 2- There's no special chars in your password. 3- Your password length is under 8. Weakpa	Very weak

6. CONCLUSION AND FURTHER SUGGESTIONS

In this paper we developed a new approach to help users create strong passwords based on cryptanalysis techniques. this study could promote password creation policies and a more secure and usable approach to enforce users to have strong passwords. As mentioned previously in the current research work, with further studies on usability of the passwords. Brute force attacks are clearly the most reliable method of getting access to a password. However, network speed would clearly be a controlling factor in an online attack. With modern hardware, the computer would be able to guess passwords far faster than the login attempt can be made. Indeed, it could be capable of guessing more passwords than the network can handle. This once again leads to very obvious traffic that should raise alarms with any system administrator. We also built a system (EP) that enhanced password when the system detected the password is weak and needed to enhance and showed the effectiveness of our approach through a series of experiments. When looking at the theoretical results outlined it seems clear that our current password systems are unsatisfactory with regards to the guidelines offered to users. Six to ten letters do not provide enough entropy when faced with modern cracking techniques, even when

each possible letter set is used. Systems which mandate the use of at least two or three of the possible character subsets lead to distinctly more secure passwords than those which do not. Should people insist on using passwords easy to remember, such as a string of words, they should be as long as possible. A word string such as ***dogcatrabbitbirdpigsnake*** is far more difficult to crack than ***D21Fx0e3*** which is an example of a short, high entropy password. For Further Suggestions the password modification algorithm must be used and improved to generate passwords with distances more than one more efficiently. this paper focuses on the latter, though options about the former will be discussed to provide a base for comparison. We will touch on other areas, like Phishing and Keylogging and Man-in-the-Middle attacks ,also a number of options could be regarding password security and cracking methodology like Dictionary attacks and Online vs Offline attacks.

REFERENCES

[1] Stobert, E. and R. Biddle (2014). The password life cycle: user behaviour in managing passwords. Proc. SOUPS.

[2] Weir, M., Aggarwal, S., Collins, M., & Stern, H. (2010, October). 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). ACM.

[3] Verheul, E. R. (2007, February). Selecting secure passwords. In *Cryptographers' Track at the RSA Conference* (pp. 49-66). Springer Berlin Heidelberg.

[4] Weir, M., Aggarwal, S., De Medeiros, B., & Glodek, B. (2009, May). Password cracking using probabilistic context-free grammars. In *Security and Privacy, 2009 30th IEEE Symposium on* (pp. 391-405). IEEE

[5] Wash, R., Rader, E., Berman, R., & Wellmer, Z. (2016, June). Understanding password choices: How frequently entered passwords are re-used across websites. In *Symposium on Usable Privacy and Security (SOUPS)*.

[6] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., ... & Vigna, G. (2009, November). Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security* (pp. 635-647). ACM.

[7] Aggarwal, S., Yazdi, S. H., & Weir, C. M. (2016). *U.S. Patent No. 9,524,393*. Washington, DC: U.S. Patent and Trademark Office.

[8] Aggarwal, S., Yazdi, S. H., & Weir, C. M. (2016). *U.S. Patent No. 9,524,393*. Washington, DC: U.S. Patent and Trademark Office.

[9] Burr, W. E., Dodson, D. F., & Polk, W. T. (2004). *Electronic authentication guideline*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology.

[10] Zhang, Y., Monrose, F., & Reiter, M. K. (2010, October). The security of modern password expiration: An algorithmic framework and empirical analysis. In *Proceedings of the 17th ACM conference on Computer and communications security* (pp. 176-186). ACM.

[11] Inglesant, P. G., & Sasse, M. A. (2010, April). The true cost of unusable password policies: password use in the wild. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 383-392). ACM.

[12] Charoen, D., Raman, M., & Olfman, L. (2008). Improving end user behaviour in password utilization: An action research initiative. *Systemic Practice and Action Research*, *21*(1), 55-72.

[13] Ruoti, S., Monson, T., Wu, J., Zappala, D., & Seamons, K. (2017, July). Weighing Context and Trade-offs: How Suburban Adults Selected Their Online Security Posture. In *Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)* (pp. 211-228). USENIX} Association}..

[14] Schroeder, J. (2017). *Advanced Persistent Training: Take Your Security Awareness Program to the Next Level*. Apress.

[15] Florencio, D., & Herley, C. (2007, May). A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*(pp. 657-666). ACM.

[16] Komanduri, S., Shay, R., Kelley, P. G., Mazurek, M. L., Bauer, L., Christin, N., ... & Egelman, S. (2011, May). Of passwords and people: measuring the effect of password-composition policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2595-2604). ACM.

[17] Bard, G. V. (2007, January). Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric. In *Proceedings of the fifth Australasian symposium on ACSW frontiers-Volume 68*(pp. 117-124). Australian Computer Society, Inc..

[18] Yan, J., Blackwell, A., Anderson, R., & Grant, A. (2000). *The memorability and security of passwords–some empirical results* (No. UCAM-CL-TR-500). University of Cambridge, Computer Laboratory.

[19] Chiasson, S., Forget, A., Stobert, E., van Oorschot, P. C., & Biddle, R. (2009, November). 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). ACM.

[20] Yan, J. J. (2001, September). A note on proactive password checking. In *Proceedings of the 2001 workshop on New security paradigms* (pp. 127-135). ACM.

[21] Spafford, E. H. (1992). Opus: Preventing weak password choices. Computers & Security, 11(3), 273-278.

[22] Schechter, S., Herley, C., & Mitzenmacher, M. (2010, August). Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the 5th USENIX conference on Hot topics in security*(pp. 1-8). USENIX Association.

[23] Castelluccia, C., Dürmuth, M., & Perito, D. (2012, February). Adaptive Password-Strength Meters from Markov Models. In *NDSS*.

[24] Yazdi, S. H. (2011). Analyzing Password Strength & Efficient Password Cracking.

[25] Yahoo Safety Center(2017) "Password Tips - Yahoo Safety",: Security : Password Tips.

[26] Burnett, M. (2006). Perfect password: Selection, protection, authentication. Syngress.

[27] Houshmand, S. (2011). Analyzing Password Strength and Efficient Password Cracking. *Electronic Thesis, Treatises and Dissertations. Paper*, *3737*.

[28] Waugh, R. (2012). No wonder hackers have it easy: Most of us now have 26 different online accounts-but only five passwords. *Daily Mail*.

[29] Florencio, D., & Herley, C. (2007, May). A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*(pp. 657-666). ACM.

[30] Houshmand, S., & Aggarwal, S. (2012, December). Building better passwords using probabilistic techniques. In *Proceedings of the 28th Annual Computer Security Applications Conference* (pp. 109-118). ACM.

[31] Das, A., Bonneau, J., Caesar, M., Borisov, N., & Wang, X. (2014, February). The Tangled Web of Password Reuse. In *NDSS* (Vol. 14, pp. 23-26).

[32] Juels, A., & Rivest, R. L. (2013, November). Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 145-160). ACM.

,

[33] Houshmand, S. (2011). Analyzing Password Strength and Efficient Password Cracking. *Electronic Thesis, Treatises and Dissertations. Paper*, *3737*.

[34] Yan, J., Blackwell, A., Anderson, R., & Grant, A. (2000). *The memorability and security of passwords–some empirical results* (No. UCAM-CL-TR-500). University of Cambridge, Computer Laboratory.

[35] Shoeb, M., & Gupta, V. K. (2013). A crypt analysis of the tiny encryption algorithm in key generation. *International Journal of Communication and Computer Technologies*, *1*(38).

[36] Uhl, A., & Pommer, A. (2005). Application scenarios for the encryption of still visual data. *Image and video encryption from Digital Rights Management to secured personal communication, Advances in Information Security*, *15*, 31-43.

[37] Robling Denning, D. E. (1982). *Cryptography and data security*. Addison-Wesley Longman Publishing Co., Inc..

[38] Velki, T., Šolić, K., & Nenadić, K. (2015). Development and Validation of Users' Information Security Awareness Questionnaire (UISAQ). *Psychological Topics*, *24*(3), 401-424.

[39] Gosney J.M. **,2012** "***Passwords^12 Security Conference",***Password CrackingHPC[Online]Location:http://passwords12.at.ifi.uio.no/Jeremi_Gosney_Password_Cracking_HPC/