

### A PROJECT REPORT ON

**“Strong Password Generation Based On User Inputs”**

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

# BACHELOR OF ENGINEERING

**(Computer Engineering)**

# BY

Santosh Doiphode Exam Seat No: S191004217

## Under The Guidance of

Dr.Sulochana Sonkamble Mam



# DEPARTMENT OF COMPUTER ENGINEERING

**JSPM Narhe Technical Campus Rajarshi Shahu School of Engineering & Research**

**Narhe, PUNE 411041**

**(2022-23)**

JSPM Narhe Technical Campus DEPARTMENT OF COMPUTER ENGINEERING



CERTIFICATE

This is to certify that the BE Project Report entitled

**“Strong Password Generation Based On User Inputs”**

Submitted by

Santosh Doiphode Exam Seat No: S191004217

is a bonafide work carried out by them under the supervision of **Dr.Sulochana Sonkamble** and it is submitted towards partial fulfillment of the requirement of Savitribai Phule Pune University, Pune, for the award of the degree of Bachelor of Engineering (Computer Engineering).

**Dr.Sulochana Sonkamble** Name:

Internal Guide Signature:

External Examiner

**Dr. N. A. Auti**

HOD, Computer Engineering

**Dr. S. A. Choudhari**

Director

Place: Pune Date: 18/05/22

# ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this report. Special thanks is due to our supervisor **Dr.Sulochana Sonkamble** whose help, stimulating suggestions and encouragement helped us in all time of fabrication process and in writing this report. We also sincerely thank you for the time spent proofreading and correcting our many mistakes.

We would also like to extend our gratitude to the Head of department **Dr. N.A. Auti** and also our respected Director **Dr. S.A. Choudhari** for providing us with all the facilities that were required.

Lastly we would like to thank our parents and friends who have helped us with their valuable suggestions and guidance, it has been helpful in various phases of the completion of the project.

Sincerely,

Santosh Doiphode

Exam Seat No: S191004217

# ABSTRACT

* Every person using different online services is concerned with the security and privacy for protecting individual information from the intruders.
* Many authentication systems are available for the protection of individuals’ data, and the password authentication system is one of them.
* Due to the increment of information sharing, internet popularization, electronic commerce transactions, and data transferring, both password security and authenticity have become an essential and necessary subject.
* But it is also mandatory to ensure the strength of the password.
* We also have examined that our generated passwords can defend against two password cracking attacks named the “Dictionary attack” and the “Brute Force attack”.
* We have implemented our system in Python programming language

# INDEX

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Content**  List of Figures | **Page No.**  7 |
|  | List of Tables | 8 |
| Chapter 1 | Synopsis | 9 |
| 1.1 | Project Title | 10 |
| 1.2 | Technical Keywords | 10 |
| 1.3 | Abstract | 10 |
| Chapter 2 | Technical Keywords | 12 |
| 2.1 | Area of Project | 12 |
| 2.2 | Technical Keywords | 12 |
| Chapter 3 | Introduction | 15 |
| 3.1 | Introduction | 14 |
| 3.2 | Motivation | 15 |
| 3.3 | Objectives | 15 |
| Chapter 4 | Literature Survey | 16 |
| Chapter 5 | Problem Definition and Scope | 19 |
| 5.1 | Problem Definition | 20 |
| 5.2 | Project Scope | 20 |
| Chapter 6 | Project Plan | 21 |
| Chapter 7 | Software Requirement Specification | 23 |
| Chapter 8 | Methodology & Algorithms | 25 |
| Chapter 9 | High Level Design Document | 29 |
| Chapter 10 | Experimentation & Results | 35 |
| Chapter 11 | Summary and Conclusion | 39 |
| Chapter 12 | Annexure |  |

Annexure A: Laboratory assignments on Project Analysis of Algorithmic design

Annexure B: Laboratory assignments (OOMD) on Project Quality and Reliability testing of Project Design Annexure C: Project planner

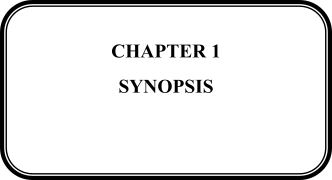
Annexure D: Published Paper as per author guidelines and format given by conference, Certificate of Publication Annexure E: References (In IEEE Format)

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name of Figure** | **Page No.** |
| 8.2 | Proposed system architecture | 37 |
| 9.1.1 | DFD 0 Level | 30 |
| 9.1.2 | DFD Multi Level | 30 |
| 9.2.1 | Class Diagram | 31 |
| 9.2.2 | Use Case diagram | 32 |
| 9.2.3 | Activity Diagram | 34 |
| 10.1 | User Interface and Result | 36 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Name of Table** | **Page No.** |
| 6.1 | System Implementation Plan | 25 |



**1.1 Project Title:**

Strong Password Generation Based On User Inputs

**1.2 Technical Keywords**

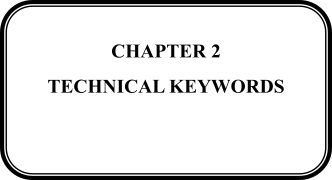
Auto-generated, Password generator, Strong password pattern, Password cracking, Case

sensitive words, Combination of numbers, Special characters, Password entropy,

Brute Force Attack, Dictionary Attack, Randomization technique, Strength checkers

**1.3 Abstract**

* Every person using different online services is concerned with the security and privacy for protecting individual information from the intruders.
* Many authentication systems are available for the protection of individuals’ data, and the password authentication system is one of them.
* Due to the increment of information sharing, internet popularization, electronic commerce transactions, and data transferring, both password security and authenticity have become an essential and necessary subject.
* But it is also mandatory to ensure the strength of the password.
* We also have examined that our generated passwords can defend against two password cracking attacks named the “Dictionary attack” and the “Brute Force attack”.
* We have implemented our system in Python programming language

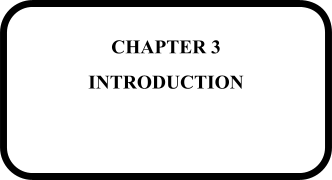


* 1. **Area of Project**
     + **PHP- I**n the case, the PHP microtime is used, which returns the current Unix timestamp with microseconds. This increases the number of possibilities, but someone with a good guess of when the password was generated, for example, the date an employee started work, still has a reasonably small search space. Also, some operating systems do not provide time to microsecond resolution, sharply reducing the number of choices. Finally, the *rand*[[5]](https://en.wikipedia.org/wiki/Random_password_generator#cite_note-5) function usually uses the underlying C *rand* function, and may have a small state space, depending on how it is implemented. An alternative random number generator, which is based on the Mersene Twister pseudorandom number generator, is available in PHP, but it also has a 32-bit state. There are proposals for adding strong random number generation to PHP.
     + **STRONGER METHODS** :-

A variety of methods exist for generating strong, cryptographically secure random passwords. On Unix platform /dev/random and /dev/urandom are commonly used, either programmatically or in conjunction with a program such as makepassword. Windows programmers can use the cryptographic application programming interface

 Function CryptGenRandom. The Java programming language includes a class called Secure Random. Another possibility is to derive randomness by measuring some external phenomenon, such as timing user keyboard input.

* 1. **Technical Keywords**
* Auto-generated.
* Password generator.
* Strong password pattern.
* Password cracking.
* Case sensitive word.
* Combinantion of numbers.

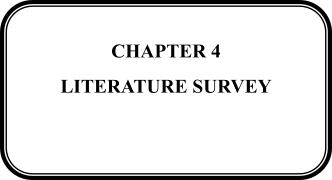


**3.1 Introduction**

Internet security is recently becoming a significant issue with the increasingly wide range of internet applications. Bank and commercial exchanges are now being carried out online in the form of internet banking and commercial electronic transactions. The level of information transmission is becoming more critical for occurring information leakage, and damages due to such leakages are more significant. User authentication is a necessary security element in the open network environment, and the use of simple authentication information also has some severe problems. One problem is that it is easy for the attackers to guess passwords whenever the users often choose personal information, such as their ID or telephone number as passwords. The users do this to remember them quickly. Sometimes they use the same password for many web sites. They do this so that they do not have to remember too many passwords. Another problem is that users rarely choose passwords that are both hard to guess and easy to remember. To help users in choosing good passwords, many experts proposed different kinds of guidelines for following many policies. Another problem is that many of the deficiencies of password authentication systems arise from human memory limitations. We know that a maximally secure password with maximum entropy consists of a string having numerous random special characters as long as the system permits. But human memory can not remember such long as well as complex passwords. When humans do remember a sequence of items, those items must be familiar pieces of information such as words or familiar symbols. However, most people are much better at recalling the information when we encode them in multiple ways. So, password authentication involves a trade-off. Some passwords may be easy to remember (for example, best friends name, pets name, kids name, etc.) but also easy to crack through dictionary searches. Other passwords may be secure against guessing but challenging to recall. Besides, some users tend to create different passwords for their various online accounts. When users create different passwords for different accounts, they need to remember several passwords which may be problematic or confusing for them during use. In this case, for remembering all the passwords, they sometimes keep insecure written records of them. Having a written document of passwords is a terrible idea because this act is not free of password cracking attacks. Eventually, an attacker can easily guess the password. So to address this issue, in this paper, we have proposed an automated system to generate user-friendly and robust passwords by combining some texts and numbers. These passwords will be made from the user given information. So we can assume that they will be easier to remember than randomly generated online passwords. The generated passwords are unconjecturable because of the features of our methodology and can be used in many and different applications and internet services like social networks, secured systems, distributed systems, and online services. Our proposed password generator can achieve diffusion, randomness, and confusions which are very necessary and required in case an intruder tries to crack the generated passwords.

**3.2 Motivation AND OBJECTIVES**

Every individual who works with different modem online services is concerned with his/her security and privacy for protecting personal information from the attackers. Password authentication is one of the popular authentication systems that has been used for many years for defending online accounts or services. At the same time, the users also need to create a strong password for protecting their services from a real-world attacker. Thus it is recommended to create a unique password with a healthy pattem so that they can protect it from the intruders. But usually, the users forget their passwords because it is not easy to remember a string with a robust and intricate pattem. To recall their passwords, they save them in their own devices, in notebooks or on sticky notes from where the passwords can get easily compromised by the intraders. So, it is always expected that the users should use strong passwords which they can easily remember without the help of writing down or saving them in their own devices or notebooks. Thus, the idea of auto-generating strong passwords based on their inputs will be convenient for them to remember easily. Let us clarify our goal by describing an example. Suppose a user has been provided with a password like “#XeV65a$PzH08!” from a random online password generator in the web for his usage. These kinds of passwords are definitely secure, but are almost impossible to remember for the average user. On the other hand, our system will ask the user to provide five input texts and any two numbers for generating passwords. Let us assume the user provides this input: “mango, cat, red, kathy, ice, 67, 81”. After that using them, our system will generate a password like “81KathY%m@ngO|”. We see that both passwords have healthy patterns, so they are safe to use. Now if the user is given a choice of choosing one password from these two passwords “#XeV65a$PzH08!” and “81KathY%m@ngO|”, the user should go for the second one. The reason is that the first password from the random generator is clueless to the user and very tough to remember. On the contrary, the second password from our system has two words and one number which they can not easily forget because they are based on the user’s suggestions. In summary, we aim to generate reliable and non-crackable passwords for the online services of the user based on the information provided by the individual which can be remembered easily.



**4.1 Literature Survey**

Maréchal et al.[1] surveyed various techniques that had been used in public or private tools in order to enhance the password cracking processes.

Weir et al. [2] developed a probabilistic context-free grammar-based training set of previously disclosed passwords. Using this grammar they generated word-mangling rules, and from them, password guesses were used in password cracking. Their work showed that their approach provided a more effective way to crack passwords comparing to traditional methods by testing their tools and techniques on real password sets. In one series of experiments, training on a set of disclosed passwords, their approach was able to crack 28% to 129% more passwords than John the Ripper, a publicly available standard password cracking program.

After that Amico et al. [3] developed a system for estimating password strength to be used as a basis for creating more effective proactive password checkers for the users and security auditing tools for the administrators.

Houshmand et al.[4] worked on defining metrics to help analyze and improve attack dictionaries. Using their approach to improve the dictionary, they achieved an additional improvement of 33% on their previous work by increasing the coverage of a standard attack dictionary.

Aggarwal et al.[5] approached some of the cutting edge approaches of password cracking that might become more prevalent in the near future.

Hitaj et al.[6] introduced PassGAN, a novel approach that replaced human-generated password rules with theory-grounded machine learning algorithms. Instead of relying on the manual password analysis, PassGAN used a Generative Adversarial Network (GAN) to autonomously learn the distribution of real passwords from actual password leaks and to generate high-quality password guesses.

Agrawal et al. [7] proposed the strong password generation technique by considering multiple input parameters of the cloud paradigm referred to as a multidimensional password.

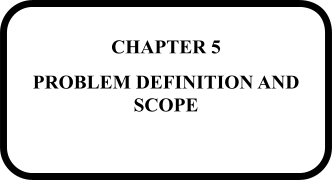
Masui [8] proposed a password generation system EpisoDAS. It took a seed string by a secret pattern drawn by the user’s episodic memories and created a password.

Mohammed [9] proposed a password generator using the genetic algorithm which achieved the requirements of a standard password.

In [10], Huh et al. proposed a system-initiated password scheme and conducted a large-scale usability test. These works showed that usability was an important factor in designing password policies.

Abadi et al. [11] and Manber [12] proposed an approach in which password was concatenated with an arbitrary value before hashing was applied. This random value was called password supplement. The mnemonic strategy had been recommended to help users generate secure and memorable passwords.

Yang et al. [13] evaluated the security of 6 mnemonic strategy variants in a series of online studies involving 5,484 participants. In addition to applying the standard method of using guess numbers or similar metrics to compare the generated passwords, they also measured the frequencies of the most commonly chosen 0418 sentences as well as the resulting passwords. Differences in the exact instructions had a tremendous impact on the security level of the resulting passwords. They examined the mental workload and memorability of 2 mnemonic strategy variants in another online study with 752 participants. Although perceived workloads for the mnemonic strategy variants were higher than that for the control group where no strategy was required, no significant reduction in password recall after 1 week was obtained.

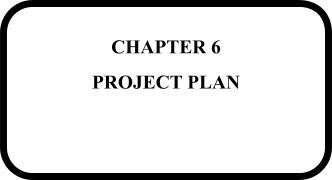


## 5.1. Problem Definition

Strong Password Generation Based On User Inputs

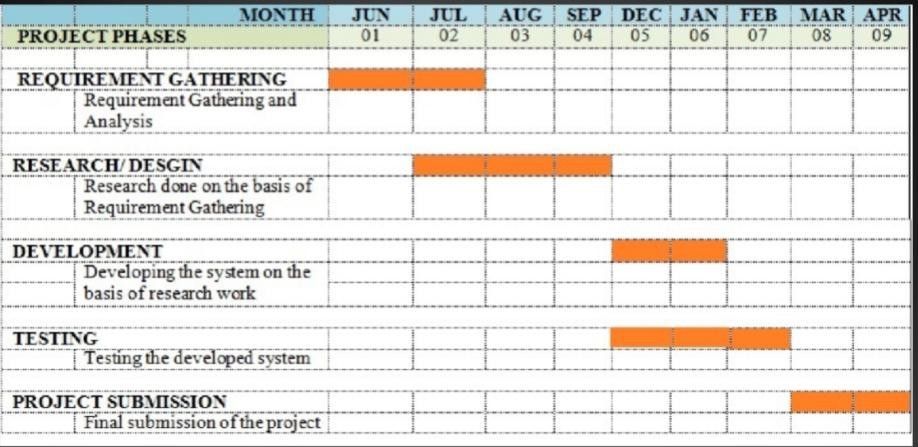
## 5.2 Project Scope

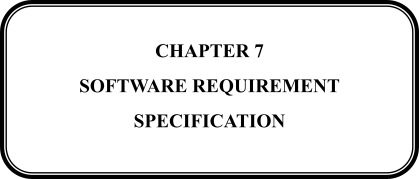
* There is another password cracking attack named “the social attack” which has not been covered in our work.
* In the future, we will extend our work and devise a method so that our generated passwords can defend against the social attack.
* Through another user study, we can also get feedback from the users to evaluate whether they can remember the passwords or not.
* We also have plans to add more features and steps to our algorithm to strengthen our system.
* We will also make it open source for the user’s convenience.



## Project Plan

* + - Phase- 1 Requirement Gathering
    - Phase- 2 Analysis
    - Phase- 3 Research Design
    - Phase- 4 Requirement Gathering
    - Phase- 5 Development of System
    - Phase- 6 GUI and Front End Development
    - Phase- 7 Part Testing
    - Phase- 8 Testing of hypothesis and performance analysis of system.
    - Phase- 9 Submission of Project



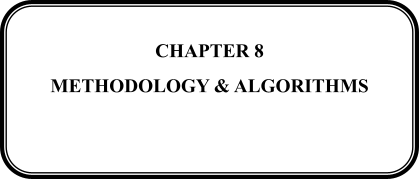


* 1. **Software Requirements (Platform Choice)**

Technologies and tools used in Policy system project are as follows Technology used:

**Front End**

* + - **Internet Explorer 6.0/above**- Microsoft Internet Explorer 6 (IE6) is a graphica lweb browser developed by Microsoft for Windows operating systems. Released on August 24, 2001, it is the sixth, and by now discontinued, version of Internet Explorer and the successor to Internet Explorer 5. It was the default browser in Windows XP 5 (later default was Internet Explorer 8) and Windows Server 2003 and can replace previous versions of Internet Explorer on Windows NT 4.0 SP6a, Windows 98, Windows 2000 and Windows ME but unlike version 5, this version does not support Windows 95 or an earlier version. IE6 SP2+ and IE7 were only included (IE6 SP2+) in or available (IE7) for Windows XP SP2+.
    - **Tool : PyCharm** on words- **PyCharm** is an integrated development environment (IDE) used in computer programming, specifically for the Python programming language. It is developed by the Czech company JetBrains (formerly known as IntelliJ).[[5]](https://en.wikipedia.org/wiki/PyCharm#cite_note-5) It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.[6]
    - **Programming Language: Python 2.7 onwards**
  1. **Hardware Requirements**
     + Processor:- Intel Pentium 4 or above
     + Memory:- 2 GB or above
     + Other peripheral:- Printer
     + Hard Disk:- 500gb



**8.1 Methodology**

In our methodology, at first, we take input information from the user who is asking for a password. The input information consists of five texts and any two numbers. In our methodology, the number of input numbers “2” is not exact; it is kind of arbitrary. On the other hand, the number of input texts has some base-analogy. If we take only two texts as input, our system will use these two texts for password generation. Then if somehow the adversary can know or detect the input texts he can surely think that both these two texts are used in the password. So from the security perspective, we have increased the number of texts to five where it can ensure some randomness for password generation. Now, think about the case if the number of input texts increases from five to ten for password generation. We can say that this input will be very secured as it will have more randomness than using five input texts. Even if the adversary knows all the ten texts prompted to the system, he can never know the chosen ones from them by our generator for generating passwords. But from the user’s perspective, inputting this large number of texts can make the process slow and complicated and also can annoy the users. So we have picked the number “5” for textsas standard for balancing between security and users’ comfort.

## 8.2 Proposed system design

## 

* Inputs will vary from individual to individual, and for this reason, passwords cannot be compromised easily.
* Input data provided by the user will consist of different interesting pieces of information such as favourite novel’s name, the number of grand mother’s children, secret dates etc.
* For ensuring security from the intruders, these data should ideally never have been exposed to any kind of social media.
* We propose an efficient algorithm to generate a strong password based on the provided information so that the adversary can not identify the password using different cracking algorithms.

**8.2 Algorithm Design**

Algorithm 1 Password Generating Algorithm

Require: Five texts, two numbers as input and the number of demanded passwords to be generated

l: Begin Procedure

2: data 4— five texts and two numbers

3: N 4— number of demanded passwords (from 1 to 10)

4: specialchars 4— { @ , $, !, # , % , & , (, ), 0, 3, 8, < , |}

5: punctuation 4 - {,\*,+ ,-,:,” ,/,\,~ ,? ,[,],{ ,} ,$ ,!,# ,% ,& ,(,),\_ ,< ,|}

6: while N > 0 do

7: stringl 4— randomly selected from five texts

8: string2 4— randomly selected from four texts

9: Number 4— randomly selected from two numbers

10: stringl 4— capitalize (stringl) some letters randomly

11: string2 4— capitalize (string2) some letters randomly

12: Final\_string 4— merge stringl and string2 randomly

13: Alphabets 4— {a, s, i, r, x, q, c, j, o, e, b, k, 1}

14: Final\_string 4— Final\_string. replace(Alphabets, specialchars)

15: Final\_string 4— randomly insert Number in the final string

16: either in the middle, or in the beginning or in the end of it

17: Final\_string 4— randomly append one special character from

18: punctuation at the end of the Final string

19: C = len (Finaljstring)

20: if C < 8 then

21: Final\_string 4— append (8 - C) special characters randomly

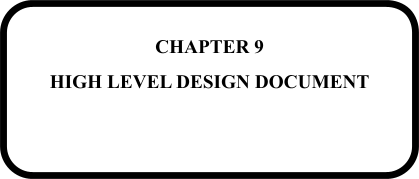
22: print Final\_string

23: end if

24: N 4— N — 1

25: end while

26:End



* 1. **Data Flow Diagrams**

### DFD ZERO

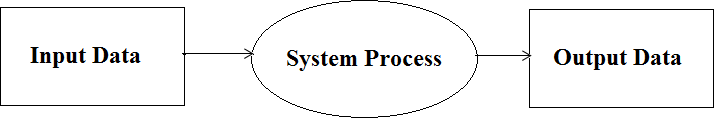


Figure 9.1: DFD 0 Level

### DFD MULTI LEVEL

### 

Figure 9.2: DFD Multi Level

* 1. **UML Diagrams**

### CLASS DIAGRAM

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

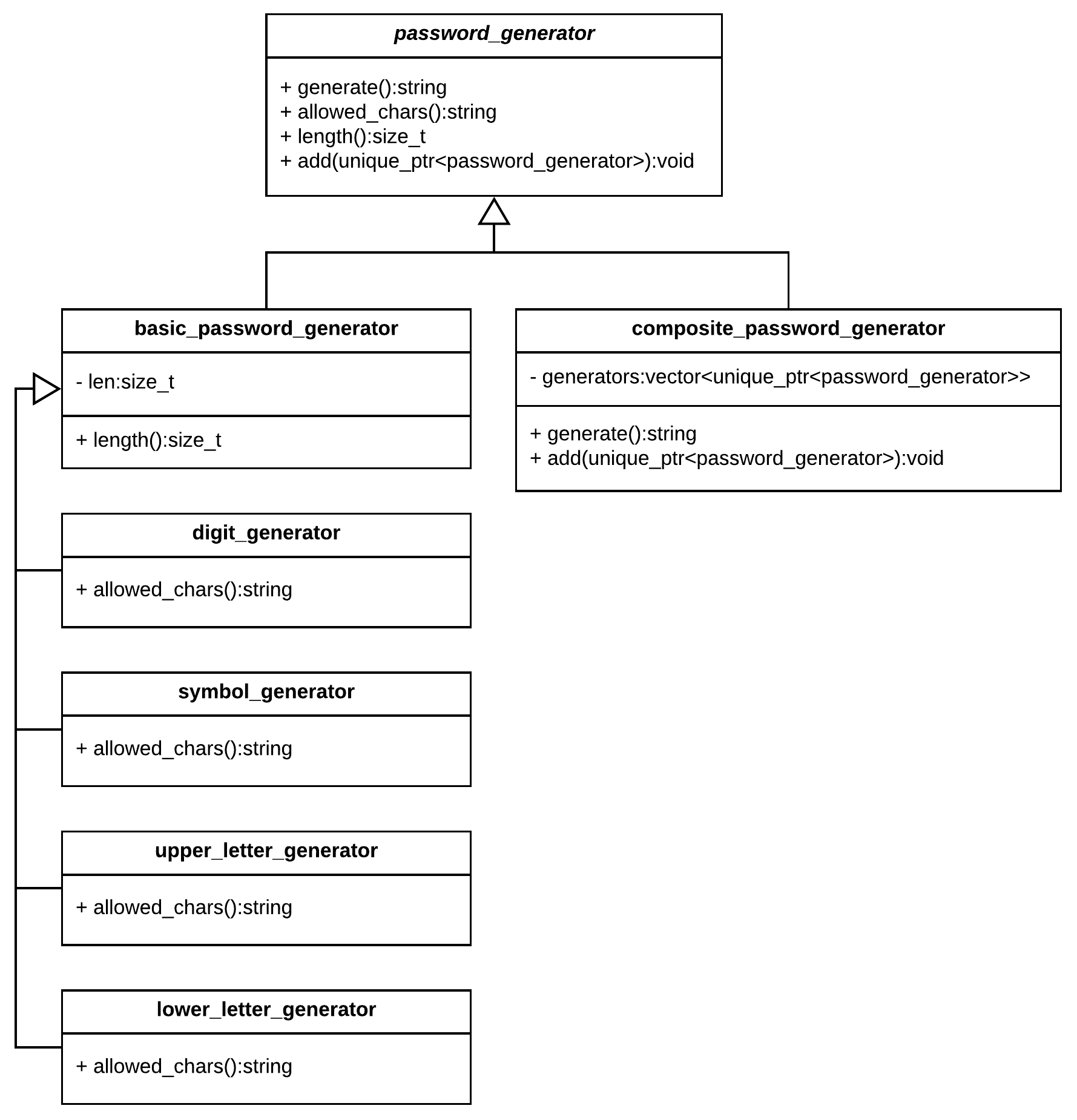


Figure 9.3 : Class Diagram

### USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

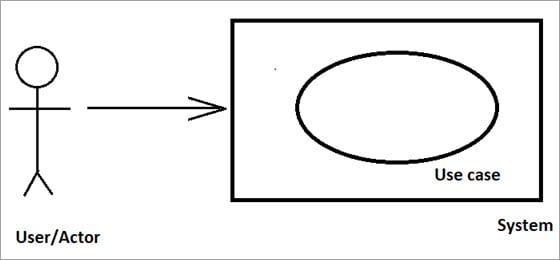


Figure 9.5 : Use Case diagram

### ACTIVITY DIAGRAM

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

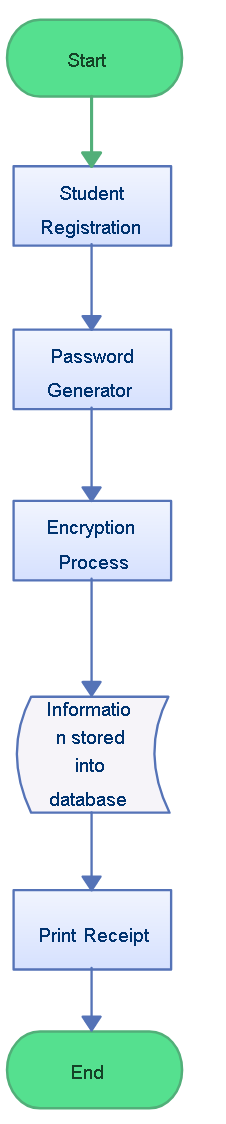
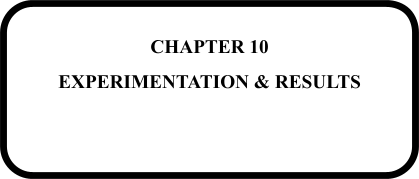
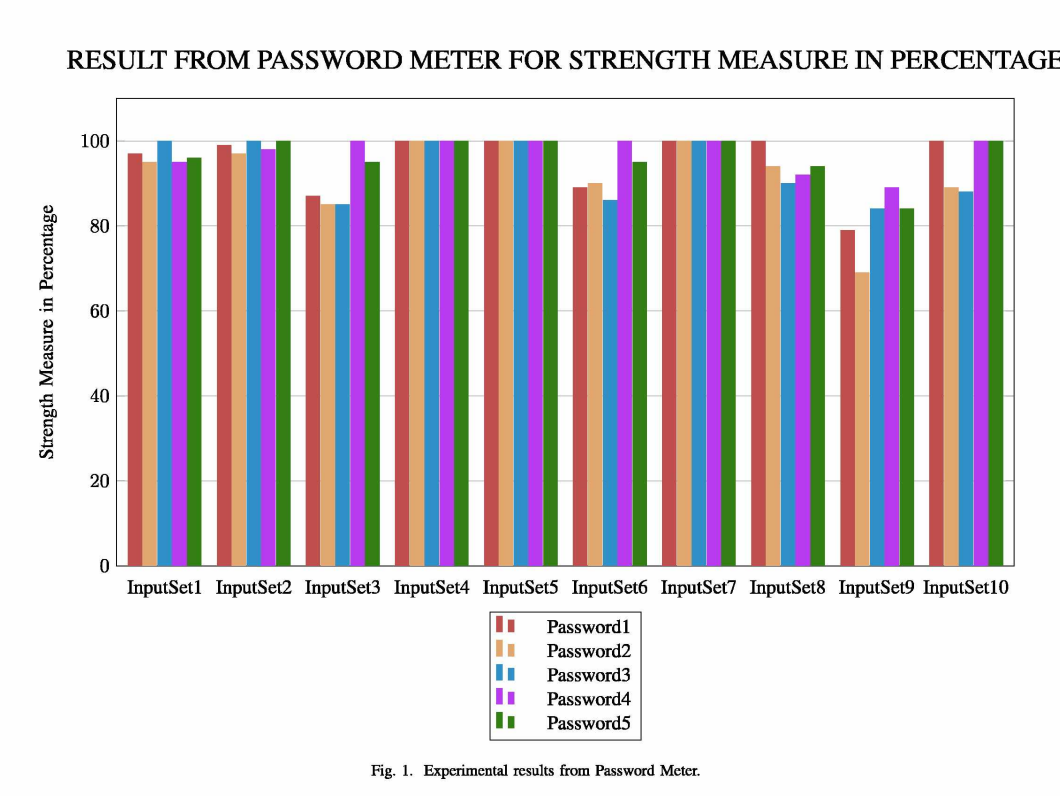


Figure 9.6 : Activity Diagram

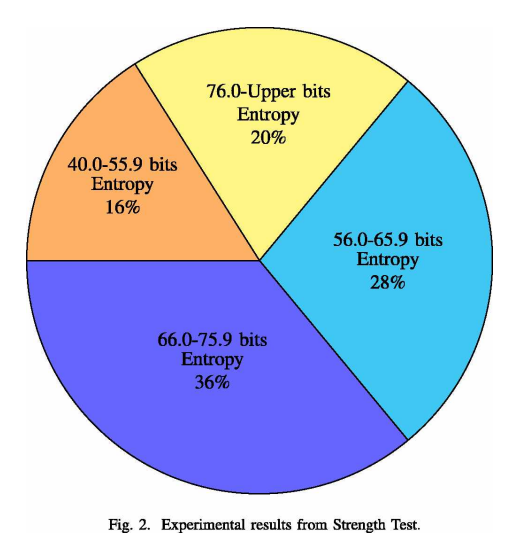


## 10.1 Results



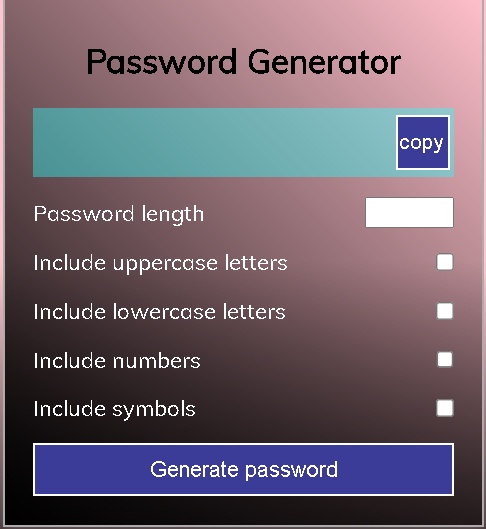
**10.1.1 Experimental Result**

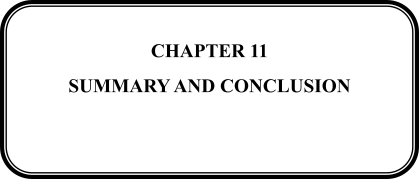
We have generated fifty passwords (five passwords for each input) from ten sample input data. We have used synthetic data as our input samples. A sample of generated passwords is shown in table 2 in this section. In our experiment, we at first have generated passwords for each of the respective input sample which consisted of five words/texts and two numbers each. Then to measure the strength and crackability we have tested all of them with four popular password strength checker online services named “The Password Meter” [27], Kaspersky Lab [28], “Password Checker Online” [29] and “Strength Test” [30]. “The Password Meter” checks the strength of the password pattern and gives a score of its strength measure from 0 to 100 including a ranking of four criteria: “Exceptional”, “sufficient”, “warning” and “failure”. If passwords’ strengths are good enough, then they will be in “exceptional” and “sufficient” criteria. If the score is not satisfactory, it means that the adversary can compromise the password and it will fall under the “warning” criterion. If the passwords don’t meet the minimum standards, then they will be underthe “failure” criterion. Experimenting with “The Password Meter”, for our 50 generated passwords we have an average score of 94.88% with a minimum of 69% and a maximum of 100% which denotes that our system-generated passwords are fairly strong. We can see the experimental results from “The Password Meter” in figure 1. “Kaspersky Lab” checks the crackabihty of passwords using the Brute Force attack to be accomplished on a home computer. Using this online service for the 50 passwords from sample input data generated by our system, we have obtained a minimum cracking span of 90 days and maximum 1217000 days by the Brute Force attack which again proves that our system is not easily penetrable to the Brute Force attack. “Password Checker Online” checks the crackability of passwords using both the Brute Force attack (on various machines: standard desktop PC, Fast Desktop PC, GPU, Fast GPU, Parallel GPUs, Medium Size Botnet) and the Dictionary attack. This checker gives an overall comment on the password depending on its strength. It also provides the cracking time estimation by the Brute Force attack and checks whether it is crackable under the Dictionary attack. By using this checker, our observation is that, for every password that we generated, the result for the Dictionary attack is “safe”. Another inspection is that the Brute Force attack’s cracking period for every password is similar to the results found from “Kaspersky Lab”. And the overall comment for all passwords ranged from medium (with 69% strength) to excellent (with 100% strength). We have also checked the results of the Brute force attack and the Dictionary attack, especially for the passwords of medium strength, and the observation is that those are safe from both attacks. For a password whose strength was 69% by using Brute Force in Medium Size Botnet, it has been shown crackable in 14 days. So, this evaluation also proves that our system-generated passwords are robust, maintaining all the password policies and requirement criteria.



“Strength Test” checks the entropy measure of a password. Password entropy is a measurement of the unpredictability of a password. It is the opposite of an ordered pattern. The bigger the entropy is, the harder a password is to crack. In cryptography, password entropy is usually expressed in terms of bits. When a password is already known, has zero bit of entropy. A password that has one bit of entropy can be guessed on the first attempt. Whenever the entropy of a password is between 28-35 bits, it means that the password is very weak. If it is between 36-59, then it is reasonable (meaning somewhat secured for network and company usage). Whenever the password has 60+ entropy, it means it is a robust password. In this checker, our system-generated passwords have entropy ranging from 47 (minimum) to 88.2 (maximum). So, with this check, we can also prove that our generated passwords are hard to crack. We can see theexperimental result of the strength test in figure 2.

**10.1.2 User Interface & Result**

****



* 1. **Conclusion**

In our proposed system, we are generating passwords by prompting the users to provide us with some texts and numbers (which they can easily remember) as inputs. Our generated passwords have ensured the minimum criteria of the password strength. Through various experiments, we have proved that our generated passwords are strong enough. Our generated passwords are different from the passwords of other online random password generators as they are not produced entirely at random. Instead, they are created from the inputs given by the user. They can be easily remembered as they contain only the texts and numbers which the users want to use in their passwords and can not forget easily. We also have experimented that our generated passwords can defend against the dictionary and the Brute Force attacks. However, there is another password cracking attack named “the social attack” which has not been covered in our work. Social attacks take an individual’s system and personal information into consideration to try and help speed up the process of dictionary attacks. Say the adversary knows the victim’s dog’s name as Pickle, mom’s name Jane and the birth date 01/07/1980. Then the adversary obviously will give this known data higher priority over other options. So the more the users will use secretive information (which have not been unveiled to any type of social media) as inputs, the more their passwords generated by our system can be safe from the social attack though we have planned nothing yet to defend this attack. In the future, we will extend our work and devise a method so that our generated passwords can defendagainst the social attack. For now, we are using now random data sets as input data. By conducting a user survey, we can collect a real data set which we can use in our generator to generate passwords. Through the use of a real data set, we can also check how much people can remember our generated passwords. Through another user study, we can also get feedback from the users to evaluate whether they can remember the passwords or not. We also have plans to add more features and steps to our algorithm to strengthen our system. We have a goal to add some exceptions in our methodology like (a) if the user inputs compound words like (“living room”; “ice cream”; “flower vase” etc.), our system will not receive these type of data and will prompt the user to provide different contexts as inputs to the system, (b) our framework will not accept the very common words like (“password”, “keyword”, “pass”, “admin”, “abc”, etc.) as these words are the first choice of the adversary to try for cracking passwords. Our other plan for this work is to experiment with how the length of the input words and numbers affect the strength of the generated passwords. We will make an online application for our system so that the users do not have to wait for installing our system into their machines for use. We will also make it open source for the user’s convenience.

* 1. **Future Work**
* In the near future, we have a plan to extend our work by developing an online free to use user interface.
* The passwords generated by our system are not only user-friendly but also have achieved most of the qualities of being strong as well as noncrackable passwords.

## References

[1] Richard Shay, Saranga Komanduri, Patrick Gage Kelley, Pedro Giovanni Leon, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, and Larrie Faith Cranor. Encountering stronger password requirements: user attitudes and behaviors. In Proceedings o f the Sixth Symposium on Usable Privacy and Security, page 2. ACM, 2010.

[2] Michelle L Mazurek, Saranga Komanduri, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Patrick Gage Kelley, Richard Shay, and Blase Ur. Measuring password guessability for an entire university. In Proceedings o f the 2013 ACM SIGSAC conference on Computer & communications security, pages 173—186. ACM, 2013.

[3] Matt Weir, Sudhir Aggarwal, Michael Collins, and Henry Stem. Testing metrics for password creation policies by attacking large sets of revealed passwords. In Proceedings o f the 17th ACM conference on Computer and communications security, pages 162—175. ACM, 2010.

[4] Hristo Bojinov, Elie Bursztein, Xavier Boyen, and Dan Boneh. Kamouflage: Loss-resistant password management. In European symposium on research in computer security, pages 286-302. Springer, 2010.

[5] Patrick Gage Kelley, Saranga Komanduri, Michelle L Mazurek, Richard Shay, Timothy Vidas, Lujo Bauer, Nicolas Christin, Lome Faith Cranor, and Julio Lopez. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In 2012 IEEE symposium on security and privacy, pages 523—537. IEEE, 2012.

[6] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L Mazurek, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, and Serge Egelman. Of passwords and people: measuring the effect of passwordcomposition policies. In Proceedings o f the SIGCHI Conference on Human Factors in Computing Systems, pages 2595-2604. ACM, 2011.

[7] Richard Shay, Saranga Komanduri, Adam L Durity, Phillip Seyoung Huh, Michelle L Mazurek, Sean M Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Designing password policies for strength and usability. ACM Transactions on Information and System Security (TISSEC), 18(4): 13, 2016.

[8] Cormac Herley and Paul Van Oorschot. A research agenda acknowledging the persistence of passwords. IEEE Security & Privacy, 10(l):28-36, 2011.

[9] Philip G Inglesant and Martina Angela Sasse. The true cost of unusable password policies: password use in the wild. ACM, 2010.

[10] Edward E Kelley, Franco Motika, and James В Webb. Universal password generation method, January 30 2007. US Patent 7,171,564.

[11] John Kelsey, Brace Schneier, Chris Hall, and David Wagner. Secure applications of low-entropy keys. In International Workshop on Information Security, pages 121-134. Springer, 1997.

[12] Kirsi Helkala and Einar Snekkenes. Password generation and search space reduction. 2009.

[13] Simon Maréchal. Advances in password cracking. Journal in computer virology, 4(1):73-81, 2008.

[14] Matt Weir, Sudhir Aggarwal, Breno De Medeiros, and Bill Glodek. Password cracking using probabilistic context-free grammars. In 2009 30th IEEE Symposium on Security and Privacy, pages 391-405. IEEE, 2009.

[15] Matteo Dell’Amico, Pietro Michiardi, and Yves Roudier. Password strength: An empirical analysis. In 2010 Proceedings IEEE 1NFOCOM, pages 1-9. IEEE, 2010.

[16] Shiva Houshmand, Sudhir Aggarwal, and Randy Flood. Next gen pcfg password cracking. IEEE Transactions on Information Forensics and Security, 10(8): 1776-1791, 2015.

[17] Sudhir Aggarwal, Shiva Houshmand, and Matt Weir. New technologies in password cracking techniques. In Cyber Security: Power and Technology, pages 179-198. Springer, 2018.

[18] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando PerezCruz. Passgan: A deep learning approach for password guessing. In International Conference on Applied Cryptography and Network Security, pages 217-237. Springer, 2019.

[19] HA Dinesha and VK Agrawal. Multi-dimensional password generation technique for accessing cloud services. Special Issue on: ’’ Cloud Computing and Web Services”, International Journal on Cloud Computing: Services and Architecture (IJCCSA), 2(3):31—39, 2012.

[20] Toshiyuki Masui. Episodas: Das-based password generation using episodic memories. In Proceedings o f the 2018 International Conference on Advanced Visual Interfaces, page 73. ACM, 2018.

[21] Sura Jasim Mohammed. A new algorithm of automatic complex password generator employing genetic algorithm. Journal o f University o f Babylon, 26(2):295-302, 2018.

[22] Jun Ho Huh, Seongyeol Oh, Hyoungshick Kim, Konstantin Beznosov, Apurva Mohan, and S Raj Rajagopalan. Surpass: System-initiated userreplaceable passwords. In Proceedings o f the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 170-181. ACM, 2015.

[23] Martin Abadi, T Mark Lomas, and Roger Needham. Strenghtening Passwords. Digital Equipment Corporation Systems Research Center [SRC], 1997.

[24] Udi Manber. A simple scheme to make passwords based on one-way functions much harder to crack. Computers & Security, 15(2): 171-176, 1996.

[25] Weining Yang, Ninghui Li, Omar Chowdhury, Aiping Xiong, and Robert W Proctor. An empirical study of mnemonic sentence-based password generation strategies. In Proceedings o f the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 1216-1229. ACM, 2016.

[26] <https://en.wikipedia.org/wiki/Password_policy>.

[27] <http://www.passwordmeter.com/>.

[28] <http://password-checker.online-domain-tools.com/>.

[29] http://rumkin.com/tools/password/passchk.php.