Intelligent Password Assistant

Major Project

Submitted in partial fulfilment of the requirement

For the

Award of Bachelor of Technology

In

Computer Science & Engineering

To

Faculty of Engineering & Technology

Gurukula Kangri Vishwavidyalaya

By

Rakesh Singh Rawat
Enrolment No. 130319

Shubham Verma
Enrolment No. 130336

Pulkit Agarwal
Enrolment No. 130313

Himanshu Kumar
Enrolment No. 130293

Under the Guidance of

Dr.Mayank Aggarwal (Ph.D)

Department of Computer Science & Engineering
Faculty of Engineering & Technology
Gurukula Kangri Vishwavidyalaya
Haridwar(Uttarakhand)-249408
2017

GURUKUL KANGRI UNIVERSITY

BONAFIDE CERTIFICATE

Certified that this project report "**Intelligent Password Assistant**" is the bonafide work of "**RAKESH SINGH RAWAT, SHUBHAM VERMA**, **PULKIT AGARWAL**, **HIMANSHU KUMAR**" who carried out the project work under my supervision.

SIGNATURE

(Dr. Mayank Aggarwal)
HEAD OF THE DEPARTMENT
(Dept. of Computer Science & Engineering)

SIGNATURE

(Dr. Mayank Aggarwal)
SUPERVISOR
(Dept. of Computer Science & Engineering)

DISCLAIMER

This project, "**Intelligent Password Assistant**" report and completely functional project has been prepared by the author under the Major Project of the Faculty of Engineering & Technology, Gurukula Kangri Vishwavidyalaya, for academic purpose only. The views expressed in the report are personal to the students and do not necessarily reflect the view of the FET, GKV or any of its staff or personnel and do not bind the Faculty of Engineering & Technology ,Gurukula Kangri Vishwavidyalaya in any manner. This report and functional project is the intellectual property of the Faculty of Engineering & Technology, Gurukula Kangri Vishwavidyalaya and the same or any part thereof may not be used in any manner whatsoever, without express permission of the Faculty of Engineering & Technology, Gurukula Kangri Vishwavidyalaya, in writing.

Rakesh Singh Rawat

Shubham Verma

Pulkit Agarwal

Himanshu Kumar

ABSTRACT

The most common method of preventing unauthorized access to digital information is through the use of a password-based authentication system. Humans are often considered the weak link in security systems. A user who is uneducated on standard security practices may leave the most advanced security system vulnerable to a determined attacker. The strength of a password-based system relies on a human's ability to generate a password that is not easily guessed, but too complex passwords are not memorable. There are many websites which require passwords .Problem arises when human uses different password for every website which creates a difficulty to remember each and every password .If he uses same password on every website, it can become vulnerable to attack.

We have developed a novel password generation model that uses natural language processing to generate a password that is both memorable and difficult to crack. Passwords are human memorable, if they have phonetic similarity and are related to human, we used Markov Models and users' native information to capture phonetic similarity. Passwords become difficult to crack, if they are not easily guessable and consists of different combinations of letters, digits and special symbols. In our project, we incorporated both security and memorability of passwords as the major concern.

Table of Contents

List of Tables:

List of Figures:

Chapter 1

1.1 INTRODUCTION

The capacity of human brain to remember everything is generally limited, and in this nuclear age the digital market has developed manifold and there are a number of websites that provide services to their users by authenticating them using text-based password systems. However, generally humans are not so clever in generating the passwords that are difficult for a system to guess or detect and quite memorable. So, there arises a need to generate passwords that are memorable by humans and at the same time difficult to crack or track.

Moreover, the growth of natural language processing (NLP) in the recent years has been notable and the technique can be used in a variety of domains whether it is about to generate an infinite grammar of some kind based on production rules or a more complicated task to understand the context of the sentence. Thus, NLP can be used extensively in text based problems and seems to be a good option to incorporate in this project to generate memorable passwords by a certain grammar designed especially for this purpose viz. **LDS,**

where,

L=Letter (0, 1, …….. 9) = 10

D=Digit (a-z, A-Z) = 52

S=Special symbol (@, #, $, ……...) = 33

Thus, we can say that the domain of all passwords lie under the umbrella of different combinations of LDS, ex.

P->LDS | LDDS | DSL | LSSDS | DLSD

and hence the complexity to crack our password is:

(95)^L , where L is the total length of generated password

Or the number of characters in the generated password

Besides it, NLP also includes a large corpora that can be used effectively to catch the genre of the user based on the questions asked to them and can be of great help to capture the phonetic similarity of the user and consequently helps in making the password memorable and difficult to crack.

Besides, there are a number of cryptographic hashing algorithms that can be used for the purpose of maintaining a security to a greater extent and for aiding in the authentication, like SHA (Secure Hash Algorithm), MD (Message Digest), etc. that produces the summary of a message known as message digest or hash value of the message that is a fixed length alphanumeric string derived by the complicated hashing algorithm which is unique for every message.

The most common method of protecting information is through the use of a symmetric key cryptosystem. In a symmetric key cryptosystem, both the encryption and decryption processes are controlled by a cryptographic key. The most common cryptographic key is a password. Additionally, the availability of commercial and open source encryption tools has provided the average computer user the ability to protect their personal files. Secure encryption is necessary in shielding information from unauthorized users.

The strength of a password lies in its entropy, which is a measure of the passwords uncertainty. A password that consists of random letters, digits and symbols will have a higher entropy than a password that is just a simple English word. A weak link in most passwords is that they must be human-memorable; therefore most passwords have relatively low entropy or randomness. However, there are a number of attacks like birthday attack that exploits the number of bits used in hashing algorithms to produce the same message digest in half the total number of bits used for producing digest by gradually approaching to a collision. Thus, SHA-512 is being used in this project to maintain the higher order for securing from such attacks. In fact, Researchers from the University of Texas at Austin, Arvind Narayanan and Vital Shmatikov, contend that as long as humans are required to memorize passwords, the passwords are vulnerable to ”smart dictionary” attacks that arises the need of a secure and elegant password manager or more precisely assistant that not only generates a strong password but also maintain its memorability, but the current password managers that are being used have many shortcomings like they store the password information related to a specific URL in encrypted form and when the user wants to visit the website then a simple matching is done of URL to password that makes them vulnerable to several attacks like phishing, eavesdropping, intrusion, etc. However, the approach used in this project is quite unique as the passwords are generated at the runtime when the user inputs the URL to retrieve the corresponding password and besides we have also used the native information related to the user to capture the phonetic similarity that aids in making the generated password memorable.

Moreover, the use of Markov Models as a probabilistic approach to predict the future state of the system is quite boom nowadays. There are

basically four types of Markov models that are used in different situations according to the need arises viz. Markov Chain, Hidden Markov Model, Markov decision process and Partially observable Markov decision process. In this project we have incorporated a Markov Chain to predict the characters in the password based on the order of the model as a Markov assumption and utilized it to substitute some of the characters by the special symbols to increase its entropy and extensively utilized Markov assumptions to capture the phonetic similarity by the users' native information.

The users have always been curious to make a password that is related to them and in this regard they generally come out with the passwords that are like English words, available in the dictionary, that make them prone to dictionary attack in which all the words from the dictionary are tried to crack the password of the user in a quite short span of time. Thus, in this project we have incorporated such models that not only generate the passwords as words that are not available in the dictionary, saving from dictionary attack, but also at the same time make them memorable due to the information used that is related to the user.

1.2 PROJECT REQUIREMENT

The main objective of this project is to generate memorable password that cannot be easily cracked.The password is not stored in database in any form to reduce its vulnerability of being attacked by an attacker.

1.2.1 TOOLS AND LIBRARIES

Python 2.7.12
Python is general purpose programming language consist of a vast variety of libraries so uses in development field and very simple to learn and use. That's why we selected python for our project.

Natural Language Toolkit 3.0 (NLTK)
The Natural Language Toolkit, or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as **WordNet,** which we have used in our project for its **hierarchial** and **structured characteristics** that provide various functionalities to capture similarity among words. Besides, **Brown Corpus** has also been used for its **rich lexical diversity** that is used to test our model.

In our Project , we have used NLTK for its enormous corpuses , that helped in related word generation in module-II [Ref.4].

HashLib - The Python Cryptography Toolkit 2.1.0

HashLib is a collection of both secure hash functions (such as SHA256 and RIPEMD160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.).This library is available in python .

We have used this library for our **Hashed Value Generation** using **SHA-512[2.1.1.2]** and **PRNG [2.1.2.5].**As of PyCrypto 2.1.0, PyCrypto provides an easy-to-use random number generator.

PyQt

PyQt is a Python binding of the cross-platform GUI toolkit Qt, implemented as a Python plug-in. PyQt is free software developed by the British firm Riverbank Computing.

In our project , we have used PyQt library to create GUI , which will take user's input and generate output password.

1.3 LITERATURE REVIEW

Personal information , organizational data or any kind of confidential data which user do not want to be shared with anyone or someone should be protected from outsiders . A strong alphanumeric password with several kind of properties like must contain at least one special character , must contain more than two digits and must contain small and capital letter combination and must be at least of length 8 or 10 plays significant role in security of information/data . Collaborators can be said a enemy of the system who observe that users, when not told how to choose good passwords, make up rules for password generation, resulting in insecure passwords instead of users.

The two most common method to attack for a guess to password to access a particular security are brute-force and dictionary attack . Brute force guarantees the password guess but because of password guessing time constraint brute force most of the time is not applicable . Dictionary attack do not tries all possible combination that can be made but tries a fix set of combination by applying different word mangling rules on dictionary words with digits and other characters as specified in mangling rules .

So A sense-able password which do not have dictionary word included but keeps something sense-able or memorable words mangling can be a better choice for user .

All memorable password generator generates either a too predictable password which can be a easy task to find out by attacker or generates a too complex random password for user which shows difficulties to remember . Most of these types of password generators uses user language corpora , dictionary words or user's personal information to generate memorable password but this way password becomes too predictable and easy to guess for attackers . Other way to generate memorable password generates a random more complex but not sense-able password which becomes very hard to memorize from users perspective .

So if a password generator which generates a memorable password which is not sense-able and far from being predictable can be worthy for user to remember and worthy in perspective of cracking by attacker .

In the other side of project schema password manager comes in . A basic password manager stores user credential like user name and password basically (can store different types of fields too according to specification of password manager) for different websites on the behalf of user . Password manager controls access to this information via a complex and non-guessable master

password . Whenever user require to enter the credential in application or in website he/she retrieve the password from password manager's access point via use of master password and uses according to need . Due to the particularly sensitive nature of the data handled by password managers , password managers aim to minimize the amount of code and personnel with access to the credentials in the clear . One common technique is encrypting the credential data base on the user's computer , thus preventing a passive attacker from accessing the credentials in plain text . Encryption techniques varies from one password manager to another , some of them use AES (Advance Encryption Standard) with cryptography involved in first two or second and third level .

A password assistant with same variety of features and security and which generates password at run time can be more efficient in this way.

Modern password manager provides a number of security features and convenience with respect to old managers . Modern password manager has a special ability to share password with collaborators which works as a mediator between user and password requesting site/application . Whenever a password request comes according to user's use of particular service/facility , collaborator provides password to authority holding the service . This way both user and service provider needs to have account with collaborator . Even we can set read/ write permission on password but this still not handles permission of collaborator from changing the password from application side . Collaborator can still change the password from service provider's side .

Modern password manager also provides facility to autofill the credential on behalf of user whenever needed by user but most of them uses an distinct software for browser which is formally known as extension for browser . So only extension supported browser can implement this feature . Todays, most of the popularity is of small computing devices like android mobile phones instead of large scale system/desktops and because lack of support for extention on this device's supported browsers like mobile safari , internet explorer , an extension needing for autofill of passwords and other fields can be a big issue . A password assistent which auto works for safe autofill can be a better choice in this context . Password assistent can also implement several modern password manager's features and can have feature to secure remote access to password from anywhere . This way password manager will expand to its services and will provide a more convenient , secure manager with assistentship .

Chapter 2

2.1 METHODOLOGY:

As humans are considered the weak links between the generation of password and the level of security and memorability they have, thus there is a need to develop a password generation computational model that not only maintains the randomness or entropy of the password but also at the same time maintains the memorability of the password that makes user use different memorable passwords on different websites that require text based password authentication.

In this project we present a novel model to generate memorable passwords by capturing the phonetic similarity of the user by asking some related questions specific to the user and then incorporating techniques of NLP to produce certain grammar rules to aid in generating password, besides markov assumption has also been used to estimate future substitutions in the generated password to increase the domain and consequently leading to a strong and memorable password.

The project is constructed in three modules providing the necessary functionality for generation of password viz. Module-1, Module-2 and Module-3 that are explained below:

Flow chart in Fig. 2.2 has all the details regarding flow of password generation process ,but for the sake of simplicity , it is divided into modules .The memorable password generation process has three different stages.

Generated password possesses three characteristics :

(1) It will be a phonetically similar text of certain length , which will be highly related to the person whose data is kept in the form of answers of introductory questions .
(2) Password consists of combinations of letters (**L**), special characters (**S**) and digits (**D**) .
(3) Generated passwords are not stored in the database , if password is needed in future , it will be generated in runtime. Thus, reducing its vulnerability.

Each stages' output will be a input to the next stage.

2.1.1 MODULE - 1

HASHED VALUE GENERATION

In this module we ensure that password generation should be unique for every website . We impose a restriction on user's access to the password generation by using a single user made **master password** .

This master password will be a key for the users identity verification . The module also takes **URL** for unique **Hashed Value** generation [Fig. 2.1].

2.1.1.1 HASHED VALUE GENERATION PROCESS

(1) The algorithm will take two inputs :

- Master Password
- URL (Uniform Resource Location) of website

(2) Using Master Password as a key and URL as message , we generate a hashed value using **SHA-512** [2.1.1.2] which will contain 128 random alphanumeric characters.

(3) This alphanumeric string will be converted to all digits using another ASCII based hash function [2.1.1.1].

(4) 128-digit hash value will act as a seed for all the modules [Fig. 2.1].

2.1.1.2 SHA-512

SHA-512 (Secure Hash Algorithm 512) is a cryptographic hash function designed by the United States National Security Agency (NSA). Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the output from execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity. For example, computing the hash of a downloaded file and comparing the result to a previously published hash result can show whether the download has been modified or tampered with. A key aspect of cryptographic hash functions is their collision resistance: nobody should be able to find two different input values that result in the same hash output.

2.1.1.3 ASCII BASED HASH FUNCTION

This is a simple function which takes the alphanumeric output of SHA-512 and convert them to all 128-digits.

It checks the hashed value string and convert every alphabet into numeral digit by taking lower offset of its ASCII value.

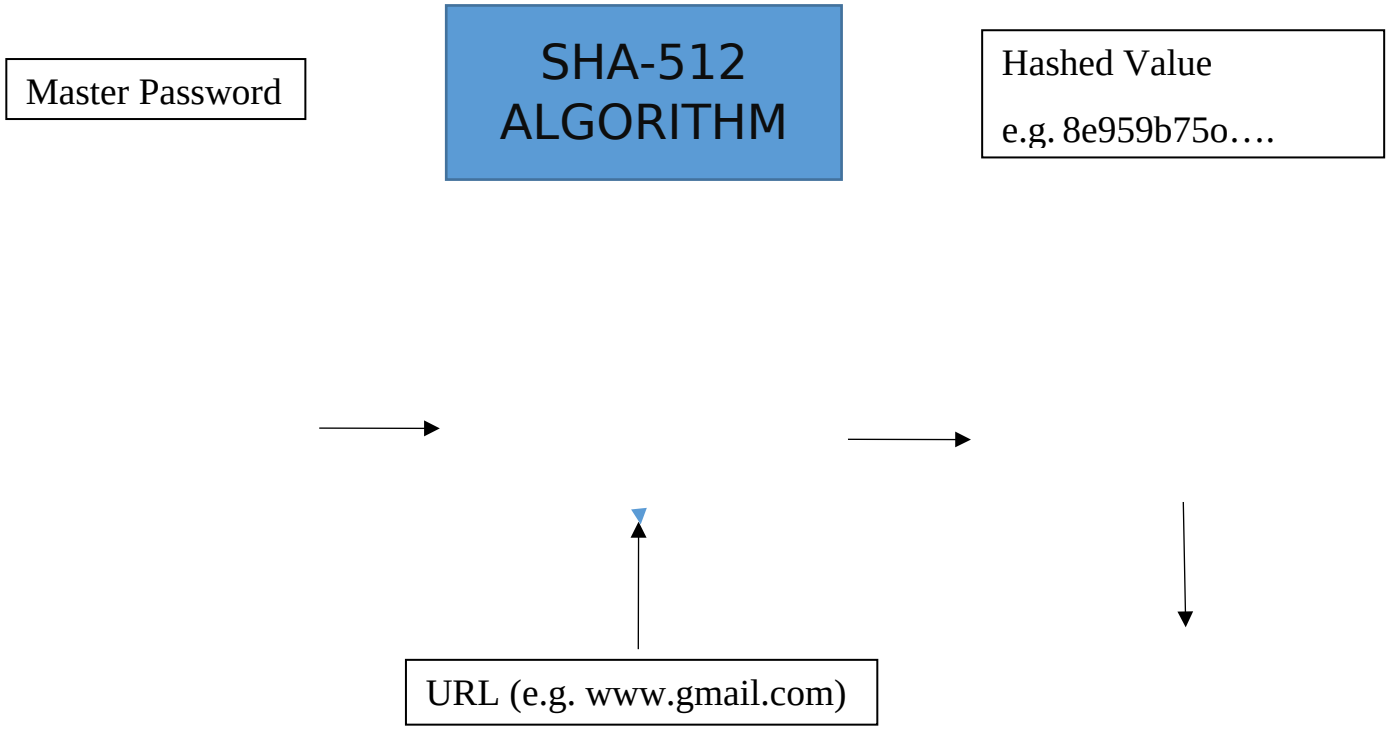


Fig. 2.1 Module – I UNIQUE HASHED VALUE GENERATION

2.1.2 MODULE - 2

RELATED PHONETIC GENERATION

This stage deals with establishing correlation with the user using his data .We have asked some questions from the user which will help in increasing memorability of a password . Output from generated hashed value from module – I is used as an input for selecting related word using **pseudo random number generation** (**PRNG**).

We have also used probabilistic phonetically similar word generation using **markov chains.**

Following components incorporates the phonetic password generation:

2.1.2.1 USER'S DATABASE

A collection of questions represents user's characteristics are taken from user before the start of the whole password generation process. This survey will be one time only.

For Example , when was the born , his mobile no , his favourite movie genre and his native language .

The answers' database help us to select appropriate combinations of phonetically similar text .

To maintain randomness in this process , we have used text selection criteria using **PRNG[**2.1.2.5**].**

2.1.2.2 TEXT SELECTION CRITERIA

Selection algorithm is based on following steps :

- For every question's answer , there is a list of associated corpora's which contains words of user's significance . For Example, if the user is Bengali , there exists a Bengali corpus in NLTK [Ref.4] .
- From these list of corpuses , our algorithm will take some highly similar words based on synsets and **Wu & Palmer** similarity score [2.1.2.6] .
- Now on each corpus , we have so many words , some of them will be useless such as stopwords in news corpora .
- We will select first '**K**' most similar words . The value of '**K**' will depend on the value taken from hashed digits.
- Now that , we have a mixed corpus , we made arrangement of these words using random arrangements from **PRNG**[2.1.2.5].
- These words will act as an input for the **Markov Model** [2.1.2.3].

2.1.2.3 MARKOVIAN MODELLING OF PASSWORDS

Markov modeling refers to the process of defining" a probability distribution over sequences of symbols". Markov modeling is commonplace in natural language processing, as Markov models easily apply data to well known language patterns.The simplest statistical information that can be utilized by Markov modeling is the underlying letter frequency distribution of a particular language. For example, in the English language the letter E occurs 12.7% of the time, while the letter Q only occurs 0.1% of the time. Therefore, when creating

words based on the English language, the letter E should appear more often than the letter Q.

Narayanan and Shmatikov utilized statistical information of the English language with two Markov model approaches: zero-order and first-order. The zero-order Markov will independently consider each letter when generating passwords. The primary motivation for utilizing the zero- order Markov model is to mimic a ”commonly used password generation [strategy],” where the password is an acronym that is obtained by taking the first letter of each word in a sentence”.

$$P(w_1 w_2 \ldots w_n) \approx \pi P(w_i \mid w_{i-k} \ldots w_{i-1}) \quad \text{-(i)}$$

2.1.2.3.1 ZERO ORDER MARKOV CHAINS

A Markov model defines a probability distribution over sequences of symbols. Said differently, a Markov model defines a probabilistic mechanism for randomly generating sequences over some alphabet of symbols. In a *zero-th order Markov model*, the symbols in the sequence are generated with a fixed probability that does not in any way depend on other symbols in the sequence. For instance, the alphabet might consist of just the letters a and b, and the zero-th order Markov model might specify that each symbol in the sequence is a with probability 2/3 and b with probability 1/3. Thus, such a zero-th order Markov model might generate a sequence like the following:

a b a b a a a b b b a a a a b b b a a b a a a a a b a a a a b a a a b b

$$P(w_1 w_2 \ldots w_n) \approx \pi P(w_i) \quad \text{-(ii)}$$

However , using zero order markov chains is similar to using random password generation .Using zero may not help in password generation , but users have multiple options to set password complexity 0-4 ,where 0 refers to simple word selection . In our project we haven’t incorporated zero –order markov chains.Instead , we generated related word using rather simple **SELECTION CRITERIA** [2.1.2.2] .

2.1.2.3.2 FIRST ORDER MARKOV CHAINS

In a *first order Markov model*, the probability of the current symbol that is being generated can depend on preceding symbol. For instance, consider a sequence in which every occurrence of b is always followed by a, while an a is followed with equal probability by either a or b. For instance, such a model might generate a sequence like the following.for e.g Existrol , mael .

$$P(w_1 w_2 \ldots w_n) \approx \pi P(w_i | w_{i-1}) \quad \text{-(iii)}$$

2.1.2.4 MARKOVIAN INTERMEDIATE PASSWORD GENERATION ALGORITHM

- First , we make a matrix which computes the probability of occurrence of a symbol behind a particular symbol .
- Then , we start with a random symbol whose selection is done through **PRNG[**2.1.2.5**].**
- Next we select its follower. The selection of follower symbol is based on PRNG.
- We repeat from step 3 till the length is reached .

Word generation through first order markov process has more chance of being phonetically similar.

Generated word has very less chance of being available in a dictionary , so it is less vulnerable to dictionary attack .For e.g : existrol doesn't exist in dictionary , but it appears to be phonetically correct .

From these markov models , it will be easier to generate memorable passwords that will not be easily crackable through dictionary attacks . We have used other two versions of markovian assumptions such as second order and third order markov chains to increase the phonetic correctness of intermediate password .

2.1.2.5 PSEUDO RANDOM NUMBER GENERATOR (PRNG)

A **pseudorandom number generator** (**PRNG**), also known as a **deterministic random bit generator** (**DRBG**),[1] is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers. The PRNG-generated sequence is not truly random, because it is completely determined by a relatively small set of initial values, called the PRNG's *seed* (which may include truly random values). Although sequences that are closer to truly random can be generated using hardware random number generators, *pseudorandom* number generators are important in practice for their speed in number generation and their reproducibility.

PRNGs are central in applications such as simulations (e.g. for the Monte Carlo method), electronic games (e.g. for procedural generation), and cryptography. Cryptographic applications require the output not to be predictable from earlier outputs, and more elaborate algorithms, which do not inherit the linearity of simpler PRNGs, are needed.

2.1.2.6 WU & PALMER SIMILARITY ALGORITHM

The Wu & Palmer calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the LCS (Least Common Subsumer).

The formula is :

score = 2 * depth (lcs) / (depth (s1) + depth (s2))

This means that 0 < score <= 1. The score can never be zero because the depth of the LCS is never zero (the depth of the root of a taxonomy is one). The score is one if the two input concepts are the same.

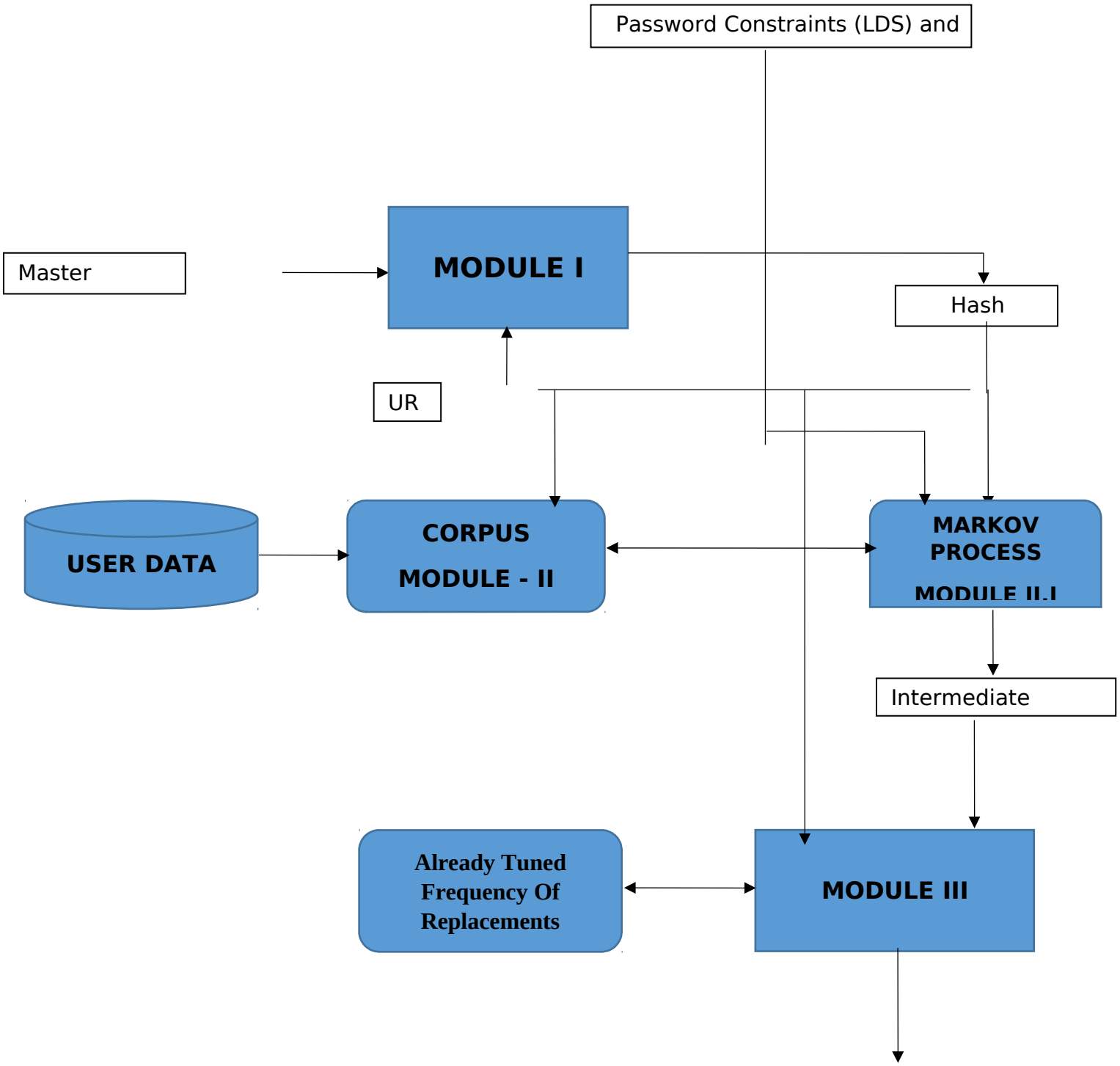

Password Constraints (LDS) and
Master
MODULE I
Hash
UR
USER DATA
CORPUS
MODULE - II
MARKOV
PROCESS
MODULE II.I
Intermediate
Already Tuned
Frequency Of
Replacements
MODULE III

Fig. 2.2 (Project Workflow)

2.1.3 MODULE - 3

SUBSTITUTION AND CAPITALIZATION

Intermediate password obtained from chapter 2 [2.1.2.4] is in lower case characters . We made certain substitutions and capitalizations in the intermediate password , to facilitate the increase in domain of password . The difference between intermediate password and substituted password is shown below :

With intermediate password

We have a-z lower case characters , So a password of length '**K'** will have $\mathbf{26^K}$ combinations which will be easy for a brute force attack.

With Subsitution

In modern keyboard , we have following combinations of key characters :

- 10 digit characters {0,1,2,3,4,5,6,7,8,9}
- 26 lower case characters {[a-z]}
- 26 upper case characters {[A-Z]}
- 33 special case characters {`~!@#$%^&*()_+[]{}\|;':"/<>?,.}

For a password of length '**K**' , we have $\mathbf{95^K}$ combinations, which is pretty large for a password of decent length. From password length analysis.

2.1.3.1 TYPES OF SUBSTITUTIONS

Basic contents of a password string consists of these three types of characters (L , D and S) . Analysis of **MySpace** password database shows some facts about the pattern of passwords by general public.

Simplified Password Structure	Number of Passwords	Percentage
LD	45120	67.3011
L	5151	7.68324
DL	4499	6.71072
LS	3311	4.9387
LDL	2863	4.27046
LSL	998	1.48862
D	817	1.21864
LSD	566	0.844247
DLD	526	0.784583
LDLD	492	0.733868
LDS	396	0.590675
LDLDL	275	0.410191
LSLD	185	0.275946
SL	158	0.235673
SLS	153	0.228215

Table 2.1

A common piece of password advice is to substitute characters, such as numbers or special characters, for letters.

For example, *password* becomes *p@$$w0rd*. These are sometimes called “leetspeak” passwords, because “elite” hackers originally used such character substitutions. Thus, there seems to be a good option to substitute some of the characters from the generated password with special symbols or digits based on the type of substitution as discussed below:

As explained earlier [Ref], that the basic structure for generating the password consists of LDS structure i.e. Letter, Digit and Special Symbol, thus we can make the substitutions in each of these three sets as:

Letter Substitution (L) –

Domain of letters lie in two basic categories whether they are Uppercase or Lowercase and thus the following substitutions seems to be at mark:

Lowercase	*Uppercase counter substitution*
o	D
c	C
p	P
S	S
……, etc	……, etc

Table 2.2

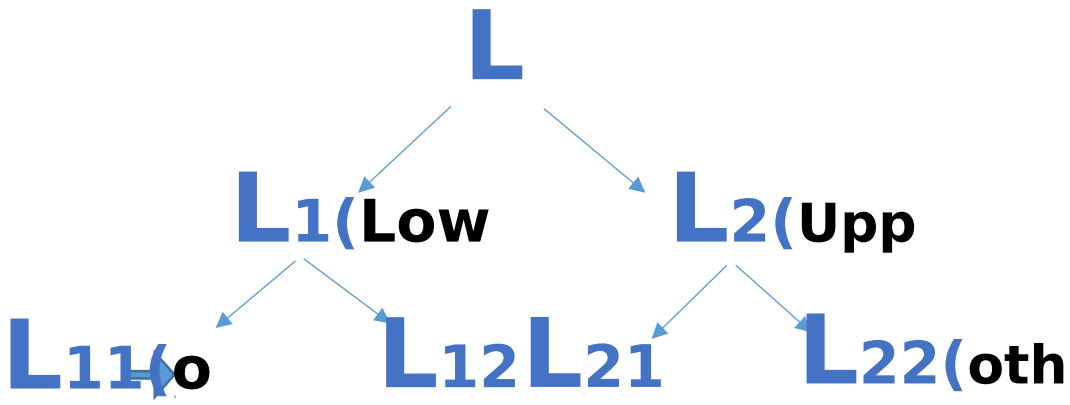


Fig. 2.3

Digit Substitution (D)-

We can also substitute the digits used in the generated password by their characters lookalike so as to increase the domain of the possible combinations of the password and also aids in the memorability of the password, as follows:

Digits	*Character Lookalike*
0(zero)	O
0(zero)	o
1	I
4	A
5	S
6	b

Table 2.3

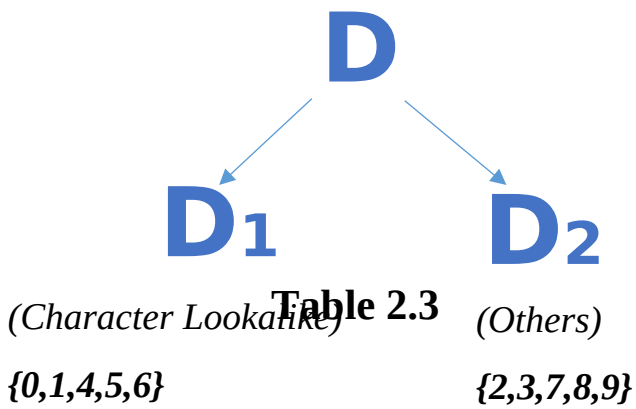


Fig. 2.3

Special Symbol Substitution (S)-

Out of the 95 characters on the keyboard, 33 are the special symbols and thus they help in making a sufficiently large domain for the passwords and provide an elegant way of using substitution of various characters in the password by the available special symbols as follows:

Special Symbol	*Character Lookalike*
@	*a*
*	(any vowel)
!	i
$	S
….., etc.	….., etc.

Table 2.4

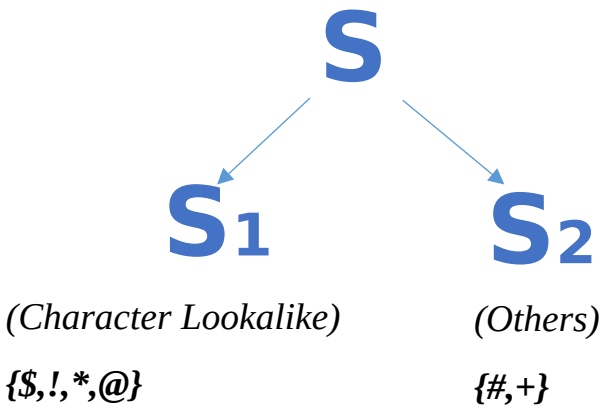


Fig. 2.5

2.1.3.2 Working of Module 3:

This module takes the intermediate generated password from the Module 2 as the input and take the reference from a replacement file that contains the information of which characters are to be substituted with the characters in the already present password.

The algorithm behind the substitution is described below:

Hash digits (from **P** **(i+1**

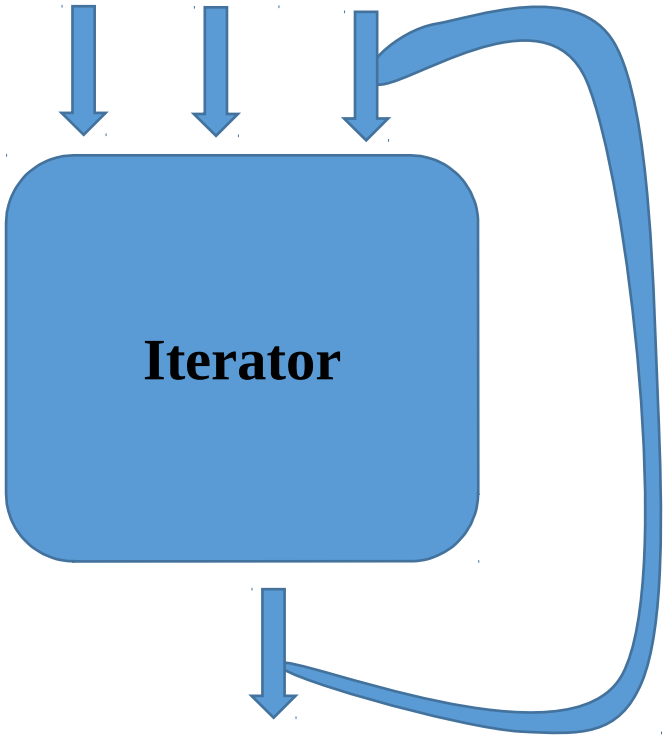


Fig. 2.6

The above module works by taking hash digits(r) and p value (learned gradually) and substitutes the character in the intermediate password with the symbol from the replacement file.

Function to change the probability of the next replacement -

F= Numerator/Denominator,

where Numerator or Denominator can take a value calculated from the following equation viz. , (1+p*r)

where, p= probability of substituting the next character in near future

r= hash digits used from module-1

besides, if there is a need for the substitution of character in the near future then the current value of F is increased by adding (1+p*r) in the numerator and when there is no need for the substitution of the characters then the value of F can be decreased by adding (1+p*r) in the denominator.

2.1.3.3 Tuning of p value:

The p value determines the probability of substitution of various characters in the intermediate password with the symbols from the replacement file. Thus, for this purpose the necessary information parameters are needed for every symbol that can be done by manually categorizing them into two different labels as MEMORABLE or UNMEMORABLE by observing somewhat 500-600 passwords and from them the p value can be determined and set in the replacement file for the purpose of substitution.

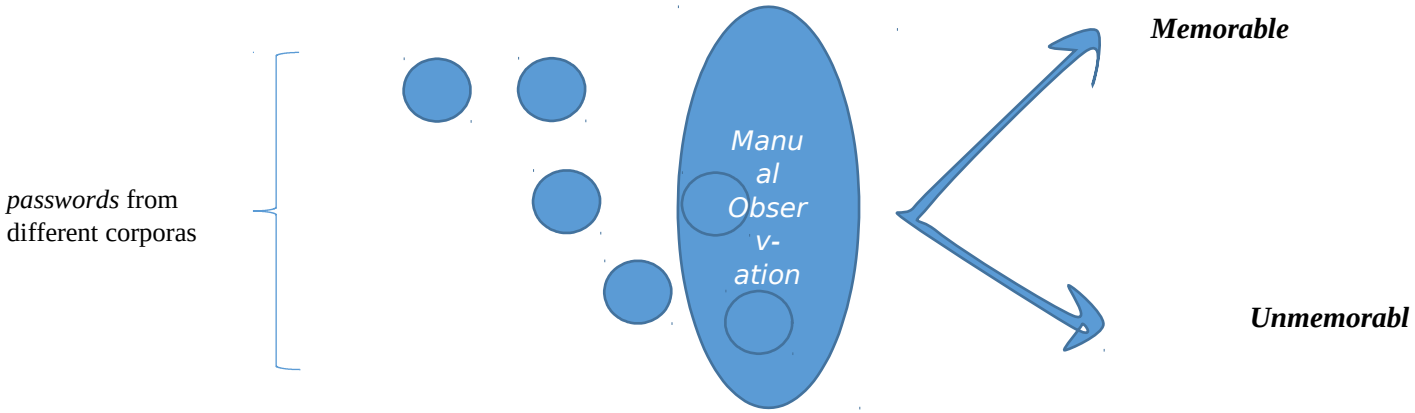


Fig. 2.7

Moreover, the p value can also be effectively learned by using the neural networks as they can be implemented in the more generalized scenarios and for that purpose we only need to have two output neurons determining memorable or unmemorable password and consequently the p value can be learned.

2.2 Graphical User Interface (GUI):

In this project, we have incorporated GUI using a python binding of the cross-platform GUI toolkit Qt implemented as a python plug-in.

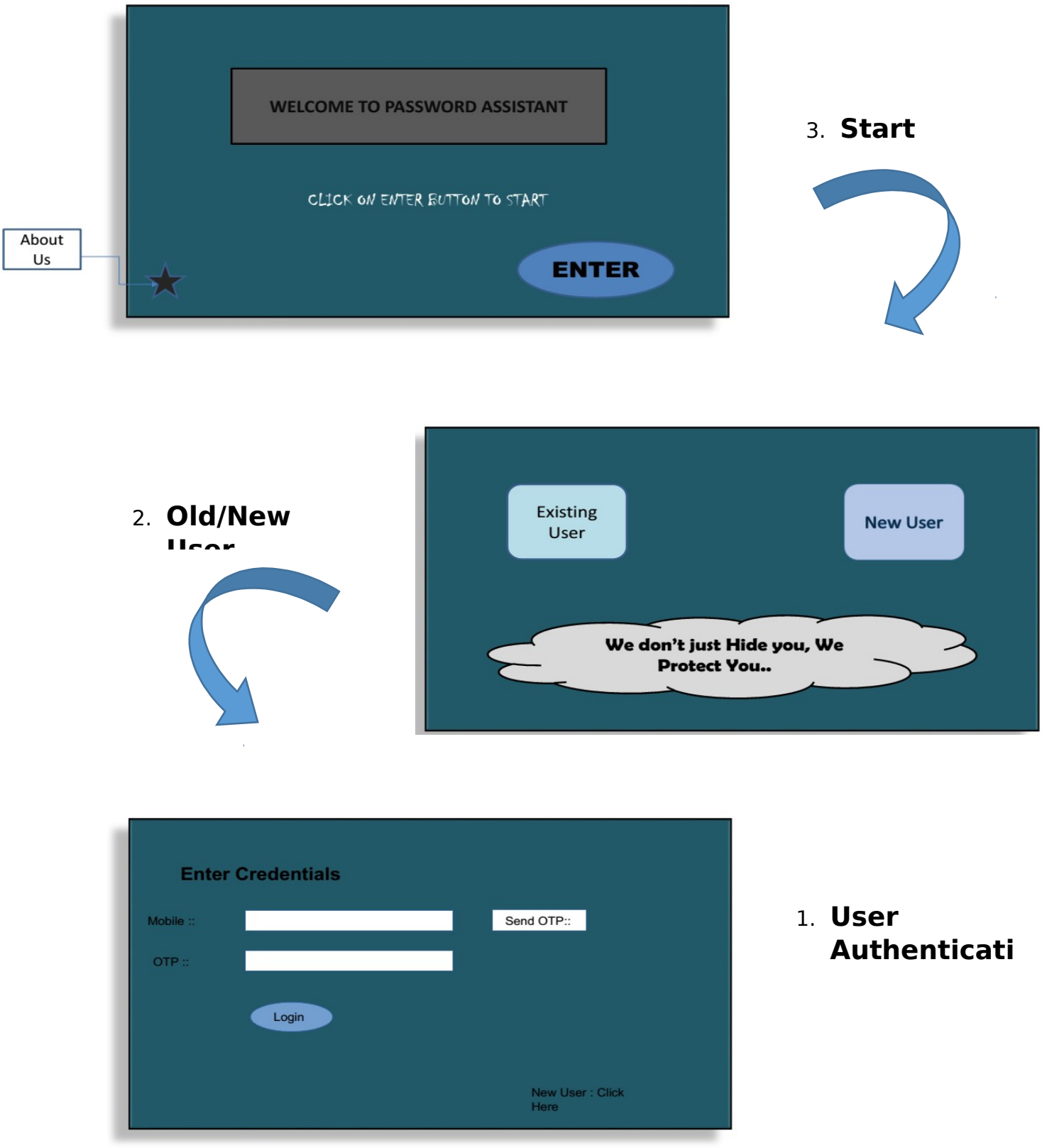
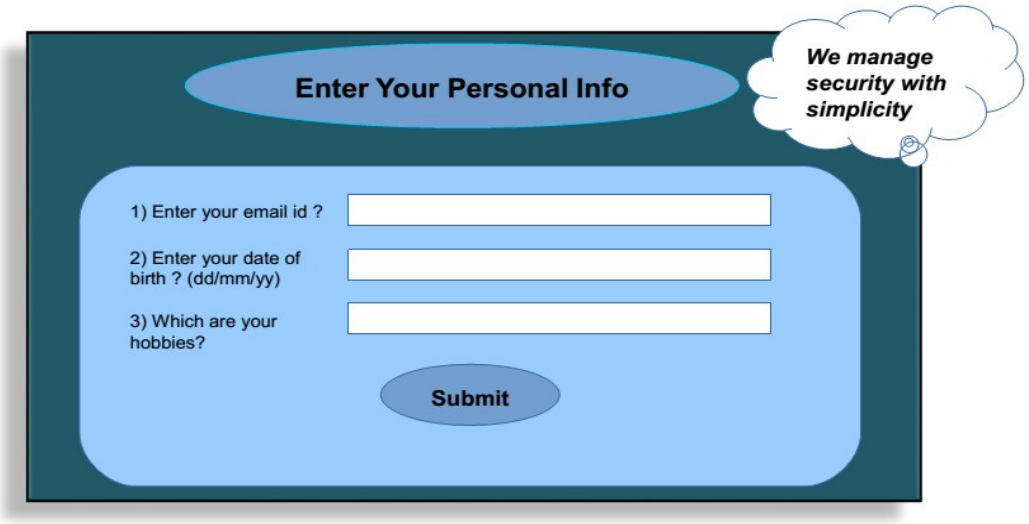


Fig. 2.8



4. **Collecting Related Informat**

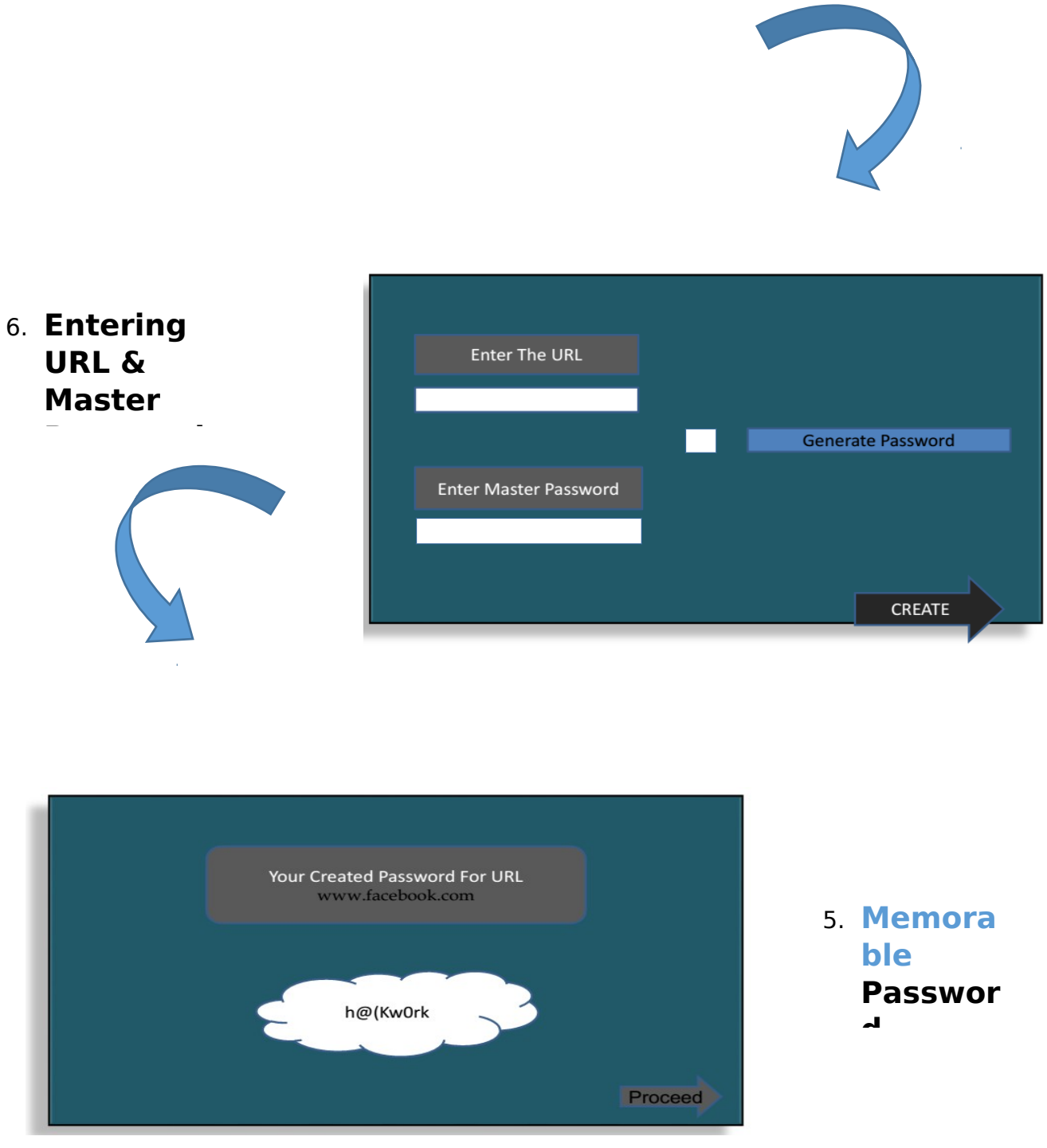

6. Entering URL & Master
Enter The URL
Generate Password
Enter Master Password
CREATE
Your Created Password For URL
www.facebook.com
h@(Kw0rk
Proceed
5. Memorable Password

Fig. 2.9

Generat

s+Rewn	aUtomAte	rE+orMatory	d|sgrAced
c4Vity	MOre	Ki<a	sY^^pAthize
v3Rstrichen	Do0r	se8mENtal	meOfIEld
s|Ourns	tr@dESmen	whiPPle	in*fFEctively
pROpylthiouracil	l'iMPeriale	v!sUAlly	h@cKWork
r4Vines	subSTances	m@NAgements	r*sTRains
s3manTically	c0MPany	eOGEd	u^SPeakable
wE1l-Planned	c0RRected	a$Signs	s+ATistically
restORing	j3Alousies	oXYgen	w4Ked
br@vESt-feathered	w#atSaPPle	s|Ckness	f|Rebreaks
greEN's	sUBic	t@pPAn	r3Semble
r*tAIns	n0Nviolent	s+eAMed	tWIsting
p0OLing	d0Om	r!m-Fire	w0Onsocket
y0U'll	w|lloWs	c0NCeivably	n3Ro
l3Adership	eXposIng	m0NOchromes	m|Ngus
k[S	irraDIation	b3Ing	c0Unter-balanced
eNDogamy	pl*aSIng	d|Sconcert	r3pliEd
600-degrees	e@rED	c0Nservative	sT@tiSticians
lUXurious	r@dIOpasteurization	d|Apiace	en1aRGe
gO0gL*	g0DS	iNTerrupt	dismISsed
kNOws	p@TRiarch	s+Eamer	fraNCes
eXAsperating	a|R-drifts	fUNgicides	sucCEed
r3volUtion	$800,000	l3Esona's	p!P's
fErroMagnetic	c#Ief's	o\/*eRconfident	f0OTman
re+rACed	kNOcked	BOO	d|Smisses
bestES	gROpe	seriOUs-minded	t#Ird-dimensionality
d!sABilities	s[Ramble	ex!lINg	s|<orich
g*oLOgy	b0yLSton	cI\/iLizations	s|Orinkle

Table 2.5

FUTURE WORK

- The time complexity of this project is quite large , In future we aim to use **multi-core** and **GPU** with parallel computation of program and encryption process
  .
- The project has option of generating only one password per person so we wish to introduce multiple password options for the users , so that they can choose whichever password they like using picture based methods.

- Encryption of user's answers can be done to make passwords more secure.

- Learning of replacement frequencies can be made more by **Neural Networks.**

- We are planning to add more corpora to this system, so that it becomes more generalized .

- We aim to improve its **GUI** so that it can be made effective for users responsive usage.

- For the future , we can introduce **improved password grammar rules** using probability.

- We can take this project on the popular platforms for users' easy access such as **android** , browser **addon**.

- We will validate the passwords generated through this system using state of the art password cracker, **John The Ripper.**

CHALLENGES

- We wanted to generate same password on every login by user , but probabilistic models were highly randomized . Randomness was necessary for the strength of passwords but at the same time it should be deterministic to generate .We removed the limitation of randomness through the use of **PRNG.**

- We didn’t used option of storage for password as it would be a matter if vulnerability , but generating passwords at runtime was a difficult task.

- We wanted to maximize the relation of password with user’s details but very less knowledge about this was available .

- The substitution need to be managed as a lot of substitution of characters can make the password lead to an unmemorable one thus the care has to be taken in the substitution at the appropriate positions

REFERENCES

1. William J. Glodek (2008) ,'Using a Specialized Grammar to Generate Probable Passwords'. Florida State University , Electronic Theses, Treatises and Dissertations.

2. Matt Weir, Sudhir Aggarwal, Breno de Medeiros, Bill Glodek , 'Password Cracking Using Probabilistic Context-Free Grammars' , 2009 30th IEEE Symposium on Security and Privacy.

3. Steven Bird , Evan Klein & Edward Loper.`Natural Language Processing with Python`.

4. Natural Language ToolKit (NLTK) , http://www.nltk.org .

5. J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password Memorability and Security: Empirical Results. IEEE Security and Privacy Magazine, Volume 2, Number 5, pages 25-31, 2004.

6. Python language https://docs.python.org/.

7. Python GUI Library , https://riverbankcomputing.com/software/pyqt/

8. Albert J. Greenfield, Robert; Marcella, editor. Cyber Forensics: A Field Manual for Collecting, Examining, and Preserving Evidence of Computer Crimes. Auerbach Publications, Boca 1.Raton, FL, 2002.

9. Wenbo Mao. Modern Cryptography: Theory and Practice. Hewlett-Packard Company, Upper Saddle River, NJ., 2004.

10. Openwall Project. John the Ripper password cracker, 2008 www.openwall.com/john/.

11. Philippe Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In The 23rd Annual International Cryptology Conference, CRYPTO '03, volume 2729 of Lecture Notes in Computer Science, pages 617–630, 2003.

12. P Oechslin. Password cracking: Rainbow tables explained. The International InformationSystems Security Certification Consortium, Inc., 2005.

13. Simon Singh. Letter frequencies, 2008. http://www.simonsingh.net/.

14. Florida State University. Electronic Crime Investigative Technologies Laboratory, 2008.http://ecit.fsu.edu/.

15. U. Manber. A simple scheme to make passwords based on one-way functions much harder to crack. Computers & Security Journal, Volume 15, Issue 2, 1996, Pages 171-176. Elsevier.

16. A. Narayanan and V. Shmatikov, Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff, CCS'05, November 7–11, 2005, Alexandria, Virginia.

17. J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison Wesley, 1979.

18. N. Chomsky. `Three models for the description of language. Information Theory`, IEEE Transactions on, 2(3):113–124, Sep 1956.

19. Xk password generator , https://xkpasswd.net/s/.

20. Thomas Anderson and Christopher DeWolfe. MySpace, 2008 , http://www.myspace.com.

21. Stallings William , `Cryptography and Network Security`.

22. Sudhir Aggarwal , Shiva Houshmand ,' Building better passwords using probabilistic techniques. Proceeding , ACSAC12 Proceedings of the 28th Annual Computer Security Applications.

23. https://www.quora.com.

24. https://crypto.stackexchange.com

25. **https://stackoverflow.com**