A Novel Methodology Proposed To Produce A Secure Password

Veera Babu Sreesailam[1] , Divya Gowri Pentakota[2] , Thanmai Pappala[3] , Shankar Kopanati[4] , Chalice Prajwal Siripurapu[5]

[1]Department of Computer Science Engineering Nadimpalli Satyanarayana Raju Institute of Technology Visakhapatnam, India

[2]Department of Computer Science Engineering Nadimpalli Satyanarayana Raju Institute of Technology Visakhapatnam, India

[3]Department of Computer Science Engineering Nadimpalli Satyanarayana Raju Institute of Technology Visakhapatnam,India

[4]Associate Professor Department of Computer Science Engineering Nadimpalli Satyanarayana Raju Institute of Technology Visakhapatnam, India

[5]Department of Computer Science Engineering Nadimpalli Satyanarayana Raju Institute of Technology Visakhapatnam,India

DOI: 10.47750/pnr.2022.13.S07.634

Abstract

A password is a string of alphabets special symbols which is used for the giving security of user accounts/files to protect the user. The more efficient the password is the strongest security it provides. Previously so many strategies were created for password generation like mnemonic shape, alphapwd, etc. Here a new strategy is proposed in this paper to create a secure password. In this strategy it performs combination of randomly generated 10-digit numbers (generated using a random function available in a python programming language) and takeskey numbers from user as input and encrypts those key numbers to random 10 digits and strings concept to produce a stronger password, this strategy is implemented in python environment. The generated password satisfies all conditions required for a strong password. The generated password consists of one digit, one capital letter, a small-cap combination, and one special character. In general, a password is between 8 to 10 characters only, but in this strategy, the generated password is up to 12 characters which provide more security. Even though any hacker tries to hack it's become very complex for hackers to generate a 12-digit character password because a normal password is 8 to 10 characters only. This is more advantageous and provides more security. We also checked the strength of the passwords that are generated using this algorithm in 3 popular online websites which determine the strength of the password.

Keywords—python programming language, random module, strings, Explicit type conversion, ASCII value, password, security

I. INTRODUCTION

In today's modern world we are using online services for various purposes like education, banking, shopping, accounting, social media, e-mail, food services (like Zomato, Swiggy), and so on ,for protecting our individual access to our online registered accounts we need more security. To provide security we had so many ways like Authentication, Authorization, firewalls, and so on. In this paper, we talk about strengthening the password-Authentication system by generating a strong and securable password. A typical password is used for authentication of the systems, it is defined as a process that involves a user inputs a unique ID and key or password that are then checked against stored credentials in the database. To strengthen the password authentication system we need a complex password which is not easy to hack,such type of complex password is generated by using our algorithm.

A. Prior Knowledge required to understand our algorithm:

As mentioned in the abstract we implemented our algorithm in python programming language. The learner need not have very good at python programming language the learner should be familiar or known with randint function available, in random module in python programming language as a first step then next step to know about explicit type conversion in python programming language. and as a last step need to know about mainly string concatenation and upper() function in python is enough.

1) python programming language:

Python is a programming language which is most popular and easy to use. Which has easy syntax and more built-in or predefined functions available in it. Anyone can choose python programming language because of its easy syntax.

2) randint function in python programming language:

Random is a function which is used to generate a number from a specified range. The random function available in random module in python programming language.

Syntax:randint(startingnumber_of_range,ending_number_of_range).

3) Explicit Type conversion:

It can be defined as conversion of one data type to another data type. Example: number is a integer data type, by using explicit type conversion in python we can convert integer data type to character data type by using the chr() function in python. That character value to that integer value is known as the ASCII value of that integer value. here we need to know about chr() function. it is a function used to convert integer values into corresponding character values.

Ex: chr(97) is 'a'.

4) String concatenation and string upper() function in python:

String concatenation means is a process to combine two strings. To concatenate, or combine, two strings we can use the '+' operator in python.

Ex:

St1="hi"
St2=" hello"
St1+St2 is "hi hello"

And upper() is built-in function or method in python which is used to convert string to upper case string.

Ex:St1="hi"

upper(st1) is "HI"
Ex: randint(0,9)

II. LITERATURE REVIEW

Farhana Zaman Glory et al. proposed an algorithm that examined by them that their generated passwords can defend against two password cracking attacks named the "Dictionary attack" and the "Brute Force attack" [1].

Abejide Ade-Ibijola et al. have proposed a password generation methodology that is based on Context-free-Grammar and their generated passwords are easy to remember [2].

Fatma Al Maqbali and Chris J Mitchell have found a password generator scheme that designs site-specific passwords on demand [3].

Salisu Ibrahim Yusuf et al. in their paper proposed a dynamic password authentication scheme that is used to avoid OTP and third-party services in case of a forgotten password. They concluded the Dynamic password is more efficient and gives the best results [4].

Pieris Tsokkis and Eliana Stavrou have found that weak passwords can results in cyber-attacks from hackers so a stronger password can increase security in the authentication. They proposed a password Generator tool that will increase security and guide people on bad password construction methods [5].

Noor Afiza Mohd Ariffin et al. proposed scheme was proven to be able to withstand the attacks. They that their implementation of the attack recognition and key generator technique together with the use of multi-factor in their proposed scheme [6].

Jianhua Song et al. invented a new password generation strategy or algorithm which depends on the mnemonic shape and alphapwd. Alphapwd combines the order of writing strokes of letters with password generation to help users create safe and memorable passwords. In their paper the passwords generated using their strategy which was rememberable and safe included [7].

Muhammad Taufiq et al... In their paper they proposed, a prototype of a client-server-based room access control system using Raspberry Pi. Which implements a one-time password mutual authentication scheme on sharing renewed finite random sub-passwords. The result of implementing a one-time mutual authentication scheme on sharing renewed finite random sub-passwords on this system prototype is proven to be resistant to replay attacks. By concluding their results, they had proven the prevention of the occurrence of passive eavesdropper and replay attacks. The OTP value for each authentication session always changes, so the attacker cannot perform a replay attack [8].

III. PROPOSED STATERGY

We aim to develop an algorithm & code for password generation which allows users to give a key value of the random size they wish. Then every single digit in the randomly generated number is added to the key value given. After adding it iterates the result to decrement or increment until it reaches the suitable ASCII value. After reaching to a certain ASCII value, then it converts the number into a character related to the value. This process is done throughout for a 10-digit number for every single digit. After converting all the 10-digit code then it converts it to a single string of characters. The starting character of the result is converted to uppercase, and a special character and number are added at the last.

The proposed algorithm produces a password that takes more time to crack and does not contain any of the password databases. It may lack behind in remembrance, but it has high security as per the experiments or validations done. We can remember our passwords using a google password manager which is trusted and secured or any other that we trust.

IV. METHODOLOGY

The proposed strategy is based on a combination of random input and user input. At first, a 10-digit random number is generated using a random function which is available in a random module in python. In the next stage, the user needs to enter a number the number can be anything. Stage 3, this stage is very important because this stage is a logic stage as I contribute. In stage 3 each digit from a random number generated at stage 1 and the total number from stage 2 which the user is gives a number as a user input are taken and added to numbers and next same process is repeated with other numbers in this stage until all each digit in stage 1 gets added to user input. After completion, a new 10-digit number is generated. At stage 5 we need to convert the new random number to alpha ASCII but here there is a problem. Because the alphabet's ASCII lies between 65 to 90 and 97 to 122. If we observe the 10-digit newly generated number at stage 3 to be greater than 122. So, the newly generated 10-digit number can be purified by using a while loop in stage 4. In stage 4 the newly generated 10-digit number at stage 3 can be purified in the loop present in this stage 4 can make each number in stage 3 available between the range 97 to 122.As we come to stage 5 we convert the purified input obtained at stage 4 can be converted into the required alpha ASCII value by using explicit type conversion in python. As come to stage 6 the password needs to satisfy the conditions in which a strong password has elements the strong password should have a 1-capital letter, small-caps combination,1-digit, and 1-special character as all these can be done using string methods available in python. As we capitalize the first character in the string variable (our password is stored in the string variable at stage 5) using the upper() function and we append the last @ and 1 to the string variable

A. Algorithm

Stage 1: A 10-digit number is produced using the random() function in the random module.

Stage 2: take key input from user (preferred to be 3-digit input in integer form).

Stage 3: This key is added to every single digit of the 10-digit number from Stage 1.

Stage 4: The newly generated 10-digit number at stage 3can be purified for converting it into alpha ASCII value so each digit in stage 3 is in the range of 97 to 122.

Stage 5: In this stage, the purified input at stage 4 can be converted into the required ASCII value and the total value is stored as a string in a string variable.

Stage 6: Return strong password which is generated at stage 5.
This generated password satisfies conditions required to be a strong password as the password which are, it should contain a 1 capital letter, small-capital letters combination,1-digit, and 1 special letter.

B. Flowchart

A flowchart is a diagrammatic depiction that shows you the consecutive order of a work process or flow of the work. figure 1 is the flowchart that represents the flow of algorithm in Figure1.

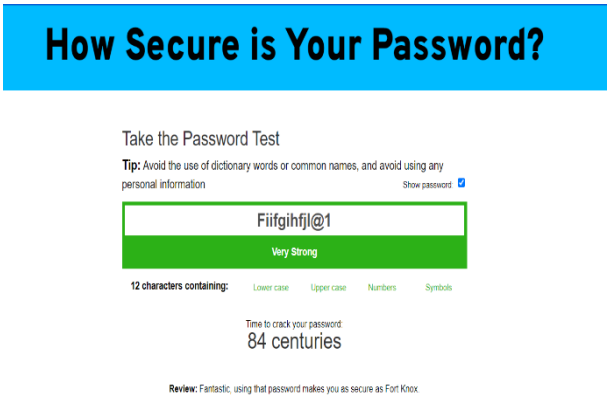

Fig. 2. Crackability time of the password from password monster.

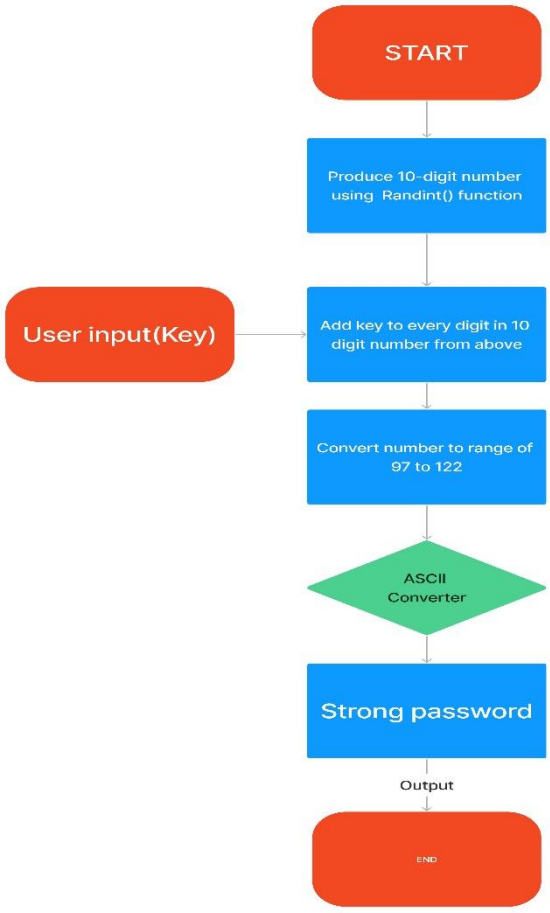

Figure1: Flowchart for password maker

V. EXPERIMENTAL RESULTS

We have generated some passwords using our proposed algorithm. To check the crackability and strength of the generated passwords. We used some online websites which check the strength and show the crackability time period of the password.

A. Password monster:

This password-strength calculator uses many techniques to check the strength of the password. It uses dictionaries like password dictionaries regular dictionaries, and name dictionaries which also include first-name dictionaries and last-name dictionaries. It performs substitution attacks on these passwords such that it checks for the replaced names, symbols with numbers, or special characters. This also checks for the proximity of the words used. figure 2 which shows the crackability time period of the given password from the password monster.

B. Kaspersky :

This password checker shows you, whether the password is hack resistant or not, and whether is it found in any of the databases of leaked passwords or not. This checks whether the password can be brute forced or not and if yes in what period can it be done, using an average personal computer. figure 3 shows the time in which the password produced using our algorithm can be brute forced.

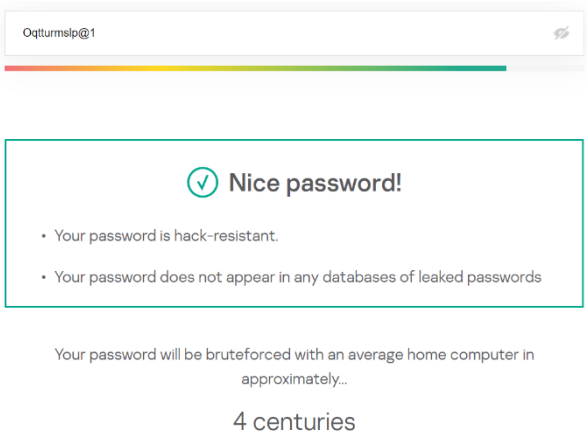

Fig. 3. Result from the Kaspersky password checker

C. Password meter:

This password meter is designed to show the user the strength percentage of the password through 100%. And also shows how strong it is and if the password is weak gives you suggestions on the basis of the requirements needed. It shows the additions and deductions for our password using colors. Blue color indicates exceptional which means that it exceeds the minimum standard, green indicates sufficient which means it has met minimum requirements, yellow indicates warning that it is against some bad practices, and red indicates failure which says it doesn't meet the minimum requirements of a strong password. Figure 4 shows the strength of the password using the password meter, and table 1 shows the additions of the password generated by our password.

We ran multiple generated passwords into password meter and represented those strengths in pictorial in Figure 5.

Test Your Password	
Password:	Oqtturmslp@1
Score:	**80%**
Complexity:	Very Strong

Fig. 4. Strength % of the password from password meter

Additions	Type	Rate	Count	Bonus

Exceptional	Number of Characters	Flat	+(n*4)	12	**48**
Sufficient	Uppercase Letters	Cond/Incr	=+((len-n)*2)	1	**22**
Exceptional	Lowercase Letters	Cond/Incr	=+((len-n)*2)	9	**6**
Sufficient	Numbers	Cond	=+(n*4)	1	**4**
Sufficient	Symbols	Flat	=+(n*6)	1	**6**
Sufficient	Middle Numbers or Symbols	Flat	=+(n*2)	1	**2**
Exceptional	Requirements	Flat	=+(n*2)	5	**10**

Table 1. shows the additions of passwords from the password meter

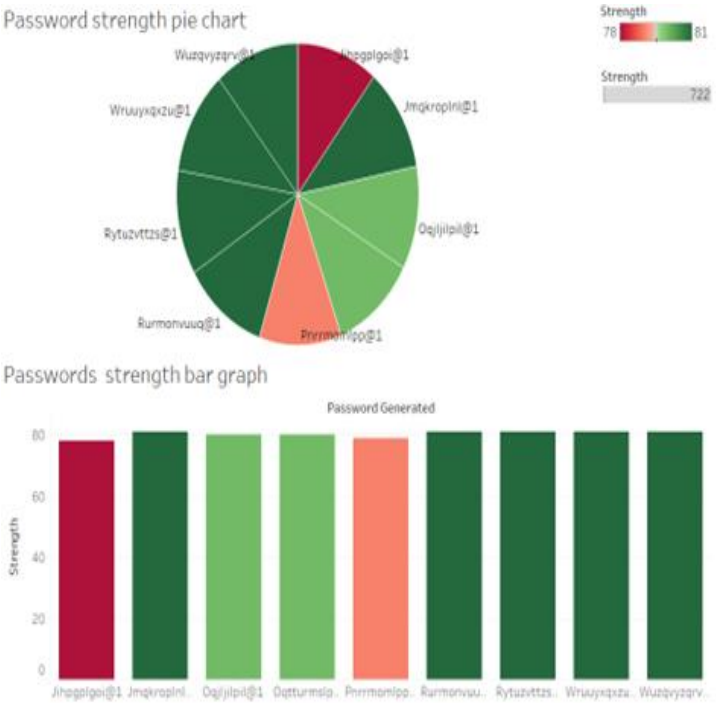

Fig. 5. Representing the password strength by password meter in form of bar graph and pie chart

VI. Algorithm Future Prototype In Real Life Situations:

We want to implement an app which uses the algorithm mentioned in this paper. The algorithm runs internally in the app as a background code for password generation. We assumed that the interface has to be like when we open the app then home page displayed at first and the homepage includes a menu bar or navigation bar is placed at left side of the app. when we click on menu bar or navigational bar option the values or options in it is displayed as the sub-menu bar. And the sub-menu bar is displayed to the user. The options available in sub-menu bar be like the profile of the user, online status of the user, and number of devices the users logged in. In going to the profile option, the values in the profile option be like profile name, online Active or not is displayed to the user. In coming back to home page, it also shows the list of devices and device lock and device unlock two buttons are present. For unlocked no passwords is rotated or not allowing passwords to enter or generate but lock will allow to generate passwords in a time quantum period. The quantum time period will be 15 to 30 seconds. The app can also be implemented in smart watches also we except to run the app in mobile and smart watches also. In figures mentioned we can see the interface architecture be like. The figures we can observe that the app design and interface.

Blow Figure 6 shows user goal problem with their actions to complete their login into a device.

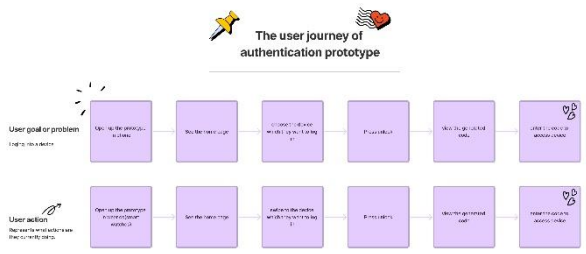

Figure 6: User journey Map to login

A. Frame Design of Proposed algorithm running in an app

1) Home page:

It consists of all the devices user use in list and user can off and on the lock and can even view the access password for limited time and can't see if its active which prevent visibility to online threats.

It shows all the list with their signal strengths, active/inactive locked or unlocked.

Represented in Figure 7.

2) Menu page:

It consists of menu bar with features present in it

a) It has list of multiple accounts

b) Additional features

c) Bluetooth unlocable devices

Represented in Figure 8.

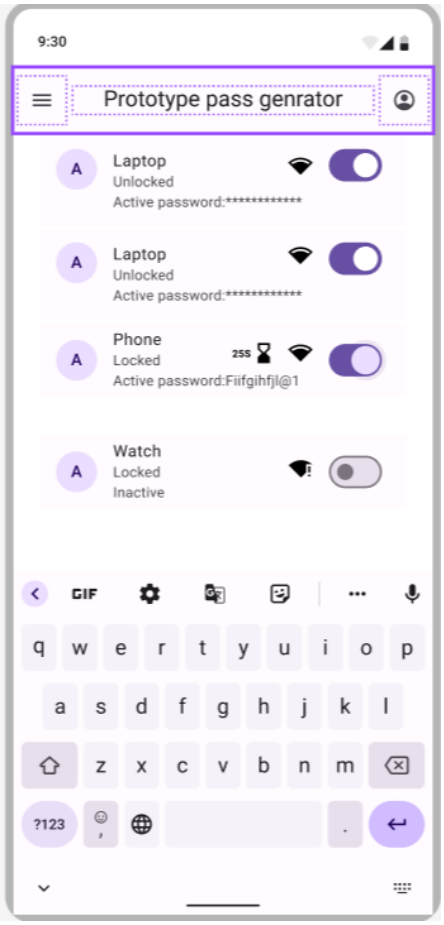

Figure 7:Home page

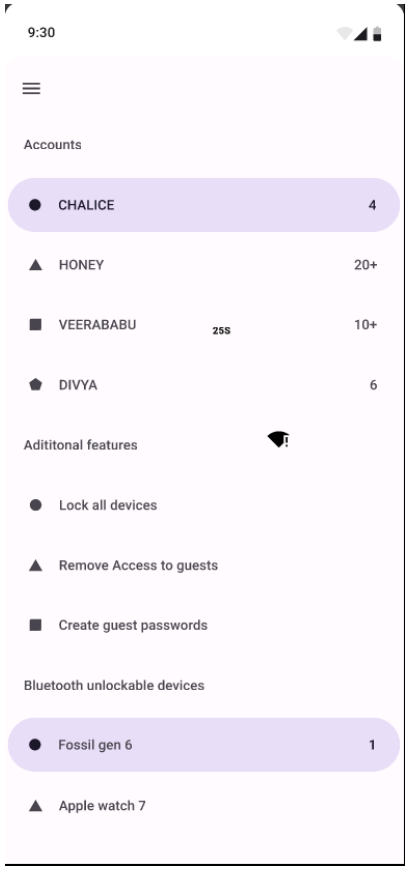

Figure 8: Menu bar

VII. CONCLUSION

In this paper, the proposed strategy to produce a password is based on a combination of random functions, strings, user input, which produce data encryption that will produce a satisfactory and secure password.
Our future work is to produce passwords that are easy to remember and tough for the hacker to generate/predict and integrate it with a cloud application which will generate a strong password every 30 seconds through which the user can access their devices/applications/profiles/files.

This integration of our algorithm to produce strong password every 30 seconds will work against competitors which are already present in market like google authenticator, Microsoft authenticator which has inability to hide active token, we want to overcome those drawbacks given by those services in our future and provide application which feel the same as its competitors to provide usability by users. We want to implement this future prototype in real situation in the development stage.

REFERENCES

[1] F. Z. Glory, A. Ul Aftab, O. Tremblay-Savard and N. Mohammed, "Strong Password Generation Based On User Inputs," 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2019, pp. 0416-0423, doi: 10.1109/IEMCON.2019.8936178.

[2] A. Ade-Ibijola and B. Ogbuokiri, "Syntactic Generation of Memorable Passwords," 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 2019, pp. 1-8, doi: 10.1109/IMITEC45504.2019.9015906.

[3] F. Al Maqbali and C. J. Mitchell, "AutoPass: An automatic password generator," 2017 International Carnahan Conference on Security Technology (ICCST), 2017, pp. 1-6, doi: 10.1109/CCST.2017.8167791.

[4] S. I. Yusuf, M. M. Boukar, A. Mukhtar and A. D. Yusuf, "User Define Time Based Change Pattern Dynamic Password Authentication Scheme," 2018 14th International Conference on Electronics Computer and Computation (ICECCO), 2018, pp. 206-212, doi: 10.1109/ICECCO.2018.8634675.

[5] P. Tsokkis and E. Stavrou, "A password generator tool to increase users' awareness on bad password construction strategies," 2018 International Symposium on Networks, Computers and Communications (ISNCC), 2018, pp. 1-5, doi: 10.1109/ISNCC.2018.8531061.

[6] N. A. Mohd Ariffin and N. F. Mohd Sani, "A Multi-factor Biometric Authentication Scheme Using Attack Recognition and Key Generator Technique for Security Vulnerabilities to Withstand Attacks," 2018 IEEE Conference on Application, Information and Network Security (AINS), 2018, pp. 43-48, doi: 10.1109/AINS.2018.8631509.

[7] J. Song, D. Wang, Z. Yun and X. Han, "Alphapwd: A Password Generation Strategy Based on Mnemonic Shape," in IEEE Access, vol. 7, pp. 119052-119059, 2019, doi: 10.1109/ACCESS.2019.2937030.

[8] M. Taufiq and D. Ogi, "Implementing One-Time Password Mutual Authentication Scheme on Sharing Renewed Finite Random Sub-Passwords Using Raspberry Pi as a Room Access Control to Prevent Replay Attack," 2018 International Conference on Electrical Engineering and Informatics (ICELTICs), 2018, pp. 13-18, doi: 10.1109/ICELTICS.2018.8548886.

[9] https://www.passwordmonster.com

[10] https://password.kaspersky.com

[11] http://www.passwordmeter.com