Description of a new scheme for generation and recovery of strong passwords (PassGen) Demonstrated through a mobile application for Android devices

Ravinder Singh[1], Nikhil Kumar Gupta[2], Shivam Sachdeva[3]
[1] Asstt Prof.Department of Computer Science Maharaja Surajmal Institute, c-4, Janakpuri, New Delhi , INDIA
[2 & 3] Department of Computer Science, Maharaja Surajmal Institute, New Delhi, India
Email - kajalravi9@gmail.com, techno848@gmail.com

Abstract: *PassGen, a new scheme for generation, management and recovery of passwords resilient against brute force and dictionary attacks, is proposed. It is an algorithm that contains multiple modules each dedicated to generate, encrypt and recover passwords. Basic information, that is, name and/or date of birth of the user is taken as input which is processed respective of the fields of data and then appended together to form a string of characters that can be generated again if the same steps are repeated on the* ***exact*** *same device. This string of characters can be used as a password wherever needed. The generated password is encrypted using a 4 digit user defined PIN, and presented as Quick Response Code* for the user to later scan and recover the same password, but again, only on the exact same device. It is also ensured that two devices cannot generate exactly the same password by mimicking each other's steps.*

Key Words*: password; brute force; encryption; Android.*

1. INTRODUCTION:

There exists a state of confusion for the modern day Internet user regarding the choice of passwords to protect access to their various accounts registered for various web applications and services. For the users who have data of high value at stake, the choice is clear: the protection of information is paramount, and the line of last defense against malicious entities are passwords.

Yet with the snowball effect of digitization for most of the users the importance and ways of creating string passwords are not clear. Therefore, they rely on password generation and management services created to capitalize on these needs. The most obvious drawback of this trend is the subscription fee for their services that is an expenditure towards something that virtually costs nothing to the providers except maintaining servers etc.

Most modern day password generators create completely random and obscure passwords that provide protection against both random and dictionary brute-force attacks in the incidence of any malevolent party obtaining their hashed form and the corresponding salt. The difficulty lies in the fact that such passwords are the most difficult to retain in human memory and any attempt at making them persist within a system renders the whole exercise of making strong passwords futile.

Passgen (short for Password Generator) is a completely free, off-line and robust application for regular end-users to ensure the strength of their passwords without putting in an effort to remember it or the risk of losing it permanently by accident. We have also created an in-house encryption mechanism to encrypt the user entered information with the password acting as a hash before converting it into a QR code. The QR code is subsequently read by our application upon requirement of recovery of the password, the user input data recovered, processed and the originally generated password is recovered.

2. CONCEPTS EMPLOYED:

WordGen

Word Gen is the name given to the small program stub created by us to obtain a legible string of characters from a numerical input of up to 3 digits in length. It is a piece of code written in Java that accepts an integer variable as input and assigns a string of characters respectively to the digit at the ones and tens (and hundreds if present) place of the number and then concatenates these strings and returns the concatenated string.

The string of characters themselves are more or less pronounceable as they are a modified version of how they are pronounced in the language Hindi. In short, it is a program used for converting numbers from Hindu-Arabic numeral representation to English alphabetical representation but using words taken from Devanagari lexicon.

For example: output of 52 will be a string "bawan".

Beads in a Sponge (BiaS)

Beads in a Sponge is a Symmetric key algorithm. The system is not universal so only the output generated from the Generation step of PassGen can be encrypted hence making it very strong.

It works in the following manner:

Step1: The Generated password is first hashed using Bcrypt Hashing Algorithm, the result is a string of length 60 characters which cannot be reversed. This is the **SPONGE** in the system, in which the beads will be put. After hashing some garbage value is appended at the trailing end of the generated hash.

Step 2: The user is asked to input a 4 digit pin which will be used as a key to encrypt the password. As this is a Symmetric key system, the same pin is used to **DECRYPT** the password as well.

Step 3: The input and choices made by the user while the creation of the password are obtained to be put inside the Hash to encrypt it. The beads can be stored input like date of birth, name or both as per choice made by the user, Imei of the mobile device, conversion choices made, format choice.

Step 4: The pin obtained from the user is separated into 4 independent digits for eg: Pin = 4352

A = 4
B = 3
C = 5
D = 2

these digits are then put into simple mathematical formulas which are optimized to get a location from 0-120 as the encrypted password is length varies from 110-130 according to the choices made by the user.

Some of the used formulas are:-

_ _ ⊔

_ ⊔ _

_ _ ⊔ _ _ ⊔ _

_ _ ⊔ _ _ ⊔

These formulas are not hard-coded or static as all the possible combinations are evaluated for the same formula i.e in case of A*B other combinations B*C, C*D, D*B, A*D and A*C are also evaluated.

The values i.e. the beads which will be put Inside the Sponge cannot be at random places and certainly cannot be concentrated in a same location. Hence to make the Algorithm strong against crypto analysis, results are evaluated in such a way that locations chosen cannot be concentrated in a specific area of the encrypted text, hence different formulas will be used for different keys. The formulas are divided into 3 categories I.e. Start, middle and end hence the values generated by these formulas correspond to these parts of the string.

Step 5: Both the locations and beads are obtained. Beads are placed inside the Hashed password according to the locations provided. For eg:-

1) Bead Date = 13, first taking the digit at the ten's place of Date i.e. 13.
2) Location = 11
3) Hashed password with garbage value:-

$50882a$11000$6a3lAbdV9R0Z2yfMVV50xQsa5EAMXBg76.ERwDh6bdp1w7COY66QfBsjfbB6917qmgYXx005o66e92a$11f30a302$3f9Z3uf.Fg3Hk309

The character at location 11 i.e. "0" marked in blue is replaced with "1" acquired from the date.

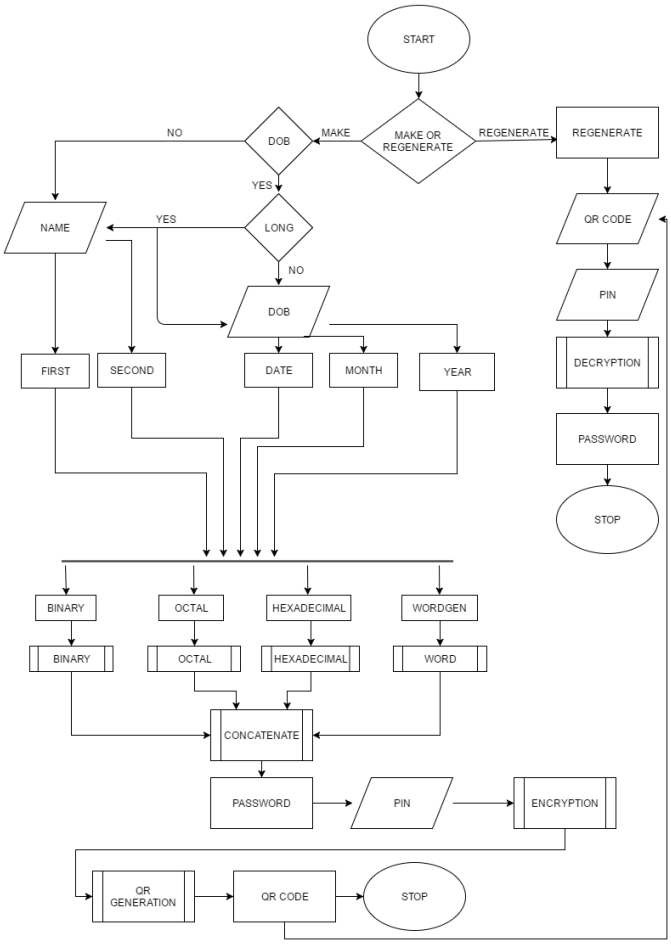
After replacing:-

$50882a$11100$6a3lAbdV9R0Z2yfMVV50xQsa5EAMXBg76.ERwDh6bdp1w7COY66QfBsjfbB6917qmgYXx005o66e92a$11f30a302$3f9Z3uf.Fg3Hk309

This process is continued until all the Beads are placed inside the sponge.

3. FUNCTIONING:

The following is a conceptual flowchart for the PassGen scheme.



The explanation of each segment in full is presented below

Password Generation

The first choice a fresh end user has in this application is what kind of information does she want to base her password on. PassGen gives the user to base their password on

Date of Birth: It consists of three fields

Date: **A two digit integer ranging from 1 to 30.**

Month: **A two digit integer between 1 and 12**

Year: **A four digit integer number ranging from 0000 to 9999.**

Name: It consists two string inputs each ideally the user's first and second names.

The user can opt for a "Long" option which combines the DOB and Name modules, in that order

Once the data has been entered, it is processed. All the operations performed on the user entered data are mathematical, which is why the data entered as string of characters is converted to numbers by adding the ASCII values of each of the characters and then, if the obtained sum is more than 3 digits long, adding each of the individual digits in the sum until a number of length 3 digit or less is obtained.

The user has an option to choose between four methods in which each of the data entered in any of the five fields is converted into a partial password. These four methods are:

Binary: The numerical data is directly converted from decimal number system to binary number system and the result stored as partial password.

Octal: The numerical data is directly converted from decimal number system to octal number system and the result stored as partial password.

Hexadecimal: The numerical data is directly converted from decimal number system to hexadecimal number system and the result stored as partial password.

WordGen: The numerical data is passed on to the eponymous code stub and a string of characters is obtained and stored as partial password.

For simplicity, this step for each individual data field that the user has chosen to fill will be referred to as "encoding" in the subsequent sections.

Special Case: If the user opts to generate password by encoding their first and/or the last name using WordGen, the following steps are taken:

Step1: A person's name can be very unique but they all have one very common thing i.e. use of vowels to make them pronounceable, especially Indian names. Taking this same thing into consideration the word received is analyzed thoroughly to find a combination of CONSONANT+VOWEL+CONSONANT eg:
if the Name received by WordGen is Rohan, it will be divided into Sub strings according to the words length, here for eg in 3 sub strings {"ROH","OHA","HAN"}.

The algorithm chooses the First true case, in this case for eg Substring "ROH" will be chosen.

This Sub string will be stored

- **Step2:**The 3 digit number goes through a modified version of DOB WordGen, where it is converted to a number and the extra digit is converted into a vowel if the last character in the generated word is CONSONANT or if the last character is vowel then it is converted to a CONSONANT excluding the alphabets 'y' &' w' due to their properties being similar to a vowel. The generated word is stored.
- The word generated is analyzed for the same combination of CONSONANT+ VOWEL + CONSONANT substring following the same algorithm as in step 2.the first sub string is picked up and is replaced with the stored sub string in Step 2.
- The final word with a part of the name of user is stored as the partial password.

Concatenation

To introduce increased entropy in the generated passwords the user is offered the choice to arrange the partial passwords from each of the data field filled by them. For example if the user chose Name as a basis for generating their password, they have the choice to arrange two data generated after encoding the first and the last name as "First-NameLast-Name" or "Last-NameFirstName". Here Last-Name and First-Name are encoded forms of the last name and the first name entered by the user respectively. This choice though is not offered in the Long password generation mode as the number of possible combinations of 5 data fields is much higher than the user can make an informed decision amongst.

The output of this step is the final combination of partially encoded data provided by the user in the previous steps, which in the form of a string of a characters can be used as password wherever required by the user. The subsequent steps involve making the password persist in a stable state and recovering it.

Encryption

This is where the aforementioned "BiaS" encryption mechanism comes into play. The user is prompted to enter a four digit PIN each time they generate a password using this system. The PIN is then used as a key to encrypt the user entered information, along with a unique identification code associated with the generating device (in its current implementation as an Android Application, the IMEI number of the device is also used) while the generated password is used to generate a hash using the password hashing function bcrypt. The result is a string of seemingly random characters which has in it embedded the different individual units of information entered by the user, just like tiny beads in a sponge. Since a random salt is used every time the generated hash is different every time.

This string varies in length between a 100 and 130 characters, and is now directly converted to a Quick Response Code which is displayed on the user's device but not automatically stored anywhere. We have used the QRCodeWriter library to convert the encrypted password to a QR Code. The QR code can be made to persist by capturing its digital image and subsequently having a hard copy printed of it.

Regeneration

This is the process of obtaining the original generated password from a QR Code which was generated through PassGen after the successful generation of the aforementioned password at some point in the past. It consists of two steps:

Scanning in the code: The QR Code is scanned and the encoded encrypted string is obtained on any digital device with a camera connected to it. In the currrent implementation, the MaterialBarcodeScanner library is used to scan in a QR Code and obtain the ecrypted string using an Android Smartphone.

Decryption: Decryption is the process of receiving the original message for the intended user to be revealed from an encrypted text. In our system it is done through a 4 digit PIN.

Detecting the method used to encrypt the password: Once the QR code is scanned by the device, the algorithm chooses which decryption method to use based on the range that the length of the encrypted text falls into.

The Pin: Once the method to decrypt has been selected, the user is asked for the pin she used to encrypt the password with. Once the user inputs the PIN, the algorithm does the same "location generation" process done during encryption.

Identification of the device: **This is a security feature implemented in the encrypted text.**
Let's say this is the encrypted text is:
$50882a$11000$6a3lAbdV9R0Z2yfMVV50xQsa5EAMXBg76.ERwDh6bdp1w7COY66QfBsjfbB6917qmgYXx005o66e92a$11f30a302$3f9Z3uf.Fg3Hk309
The First location generated is 15. Character chosen is '6', once we store the digit the digit is then removed from the string hence all the succeeding characters fall back 1 step. This process is continued till the imei number is taken out of the encrypted text. This IMEI number is checked with IMEI of the phone it is scanned in and if it matches then only the process is continued else the process is stopped. Therefore Regeneration only works in the phone that was originally used to Generate the password.

If the IMEI number matches then all the generated locations are used to take out values from the encrypted text and stored in variables. For example
DATE: 06
MONTH: 05
YEAR: 1934
CHOICE FORMAT: 6
COLOR CHOSEN: #462ff2 etc.

Recovery of the original password: **Once all the required values have been taken out then the algorithm sets itself as an internal bot, goes through all the steps of Password Generation with the values obtained from the encrypted text used as input for each step, and presents the originally generated password to the user.**

4. CONCLUSION:

Like all developments in the field of digital security, PassGen suffers from the possibility of failure when deployed in a larger environment such as the World Wide Web. Any security scheme which aims to provide a complete protection against all forms of digital threats is bound to fail if employed on a global basis unless it has been tested in all deployable environments. To avoid running into a situation where full scale deployment leads to a compromise in integrity of an information system, PassGen has been developed as an auxiliary to all existing security systems currently employed in networks and computer systems.

All security schemes revolve around the unique identification of the user(s) attempting to access the data held digitally. The most common way to uniquely identify and verify a user is through a user defined password. This password is a string of characters that

- the user is trusted with keeping safe.
- Is expected to be long and complex enough to not be "cracked" by malevolent parties
- the user ought to retain for as long as they wish to access the data

PassGen aims at improving the quality of passwords on these three fronts. We have developed our own in-house encoding and encryption algorithms so that any compromise in the viability of most common password generation or encryption algorithms that we may use does not affect the viability of PassGen. The algorithm BiaS is designed to be able to accept only specific inputs in specific lengths and encrypt them to be decrypted later.

PassGen is therefore not a complete security solution, but rather a tool that improves the chances of user passwords surviving brute-force and dictionary attacks, which in turn improves the quality of security available to the modern user.

ACKNOWLEDGMENT:

The authors would like to acknowledge the constant guidance and encouragement of Mr Ravindra Kajal, Associate Professor, Maharaja Surajmal Institute. His dynamism, vision, sincerity and motivation have deeply inspired us. We would also like to thank the Stack Overflow community as a whole for helping us out in times we needed help with the development of the demonstrative application.

Finally and foremost we would like to thank God, and his images on earth, our parents.

REFERENCES:

1. Wikipedia contributors. Pigeonhole principle [Internet]. Wikipedia, The Free Encyclopedia; 2016 Nov 5, 21:30 UTC [cited 2016 Nov 5]. Available from: https://en.wikipedia.org/w/index.php?title=Pigeonhole_principle&oldid=748019760.
2. B. Schneier, Security in the Real World: How to Evaluate Security Technology, Computer Security Journal, v 15, n 4, 1999, pp. 1-14.