iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** V **Month of publication:** May 2020

DOI: http://doi.org/10.22214/ijraset.2020.5355

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

EnPass – The Password Guard

Farhin Mansur[1], Kewal Mahyawanshi[2], Anand Mariappan[3], Nikhil Srivastava[4], Snehal Patel[5]
[1]*Assistant Professor, Laxmi Institute of Technology, Sarigam*
[2, 3, 4, 5]*Laxmi Institute of Technology, Sarigam*

Abstract: ***Cyber security is becoming one of the largest growing fields in Computer Science and Technology industry; passwords have become the essential key to access any web page or to login any application. An average user accesses numerous sites that it becomes very hard for the user to remember all those passwords. This leads to problem like password reuse, in which the user uses same password for multiple sites and applications, which creates a vulnerability by which the hacker can access the user's content easily. There are also cases where users save their important credentials like password of net-banking without proper security in places like phone book, notes or their personal diaries which can be accessed by any other attacker. In order to subdue these issues, a system is proposed named EnPass, which is an Android application for password manager.***
Keywords: Password manager, password security, credential management, Cybersecurity, Vulnerability.

I. INTRODUCTION

Nowadays passwords have become the essential key to access any web page or to login any application. The criteria for a strong password contain at least 8-digit password which avoid brute forces and various other attacks in the password cracking mechanism. EnPass will ease the things for the users by storing their password on its own database and providing them whenever they are in need of it. EnPass is an Android application which can be used to store important credentials, where n-numbers of passwords can be saved and retrieved provided that the user enters a valid account name and the Master Password. The master password will remain same for all logs which are saved by the user.

The password stored by the user in each log is saved in an encrypted form in the database. AES 256-bit encryption technique is implemented in this password manager application for securing users' credentials.

The main purpose of this project is to make an application for the users for storing numerous passwords effortlessly without having to remember them and still be ensured about the security.

II. LITERATURE SURVEY

Maliheh Shirvanian, Stanislaw Jarecki, Hugo Krawczyk and Nitesh Saxena says that, we introduce a novel methodology with password management, called SPHINX, which stays secure in any event, when the password manager itself has been compromised. In SPHINX, the data stored on the gadget is information theoretically independent of the user's master password — an attacker breaking into the gadget learns no information about the master password or the user's site-specific passwords. Moreover, an attacker with full control of the gadget, even at the time the client communicates with it, adapts nothing about the master password — the password is not entered into the gadget in plaintext structure or in any other way that may leak information on it. Unlike existing managers, SPHINX creates carefully high-entropy passwords and makes it compulsory for the clients to enrol these randomized passwords with the web services, hence fully defeating offline dictionary attack upon service compromise. The structure and security of SPHINX depends on the gadget upgraded PAKE model of Jarecki et al. that provides the theoretical basis for this construction and is backed by rigorous cryptographic proofs of security. We present the plan, execution and execution assessment of SPHINX, offering model program modules, cell phone applications and transparent device-client communication [1].

Florian Zinggeler says that, Strong passwords comprise of a long sequence of irregular letters and symbols that have never been utilized. However, since recalling these passwords would be practically impossible, many use a password manager to help with that task. Popular password managers include LastPass [4], 1Password [5], Dashlane [6], KeePass [7]. Most password manager's work by putting away passwords in a database, encoded with a client picked master password. The security that a password manager offers is therefore proportional to the strength of the chosen master password. Clients at that point need to enter that master password each time they need to recover their saved passwords or in the event that they want to save password. In this work we present a distributed password manger called NoKey that doesn't utilize a master password. Rather, passwords are put away on different gadgets of the client so that they must be perused if enough gadgets meet up. With this application, using a saved password only requires that the user gives permission on another device. For instance, to access the stored passwords on the user's laptop, the user has to allow that access on her phone [2].

Moritz Horsch, Johannes Braun, and Johannes Buchmann says that, for decades, users are not able to realize secure passwords for their user accounts at Internet services. User's passwords need to fulfil general security requirements and the password requirements of services. Furthermore, users need to cope with different password implementations at services. Finally, clients need to play out a huge number of tasks to appropriately deal with their huge password portfolios. This is practically impossible. We present the vision of password assistance. It supports users in all duties and tasks with regard to their passwords, from the creation of secure passwords to the recovery of them in case of loss. Moreover, it provides an extensive automatization of all password tasks that reduces the user's efforts and activities to deal with passwords to a minimum. A password assistant enables high security for passwords as well as improves their ease of use [3].

III. HOW IT WORKS

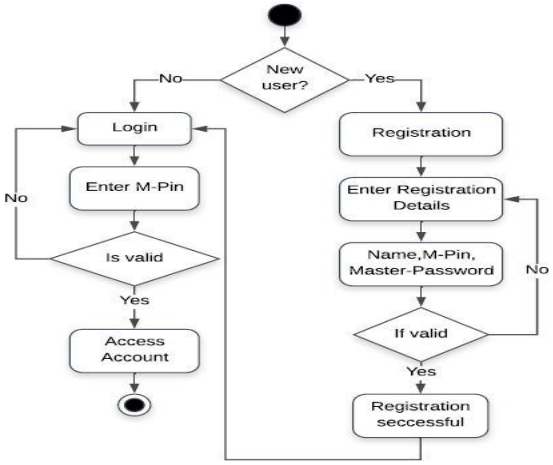


Figure 1.1: Flow of the Login/Registration process

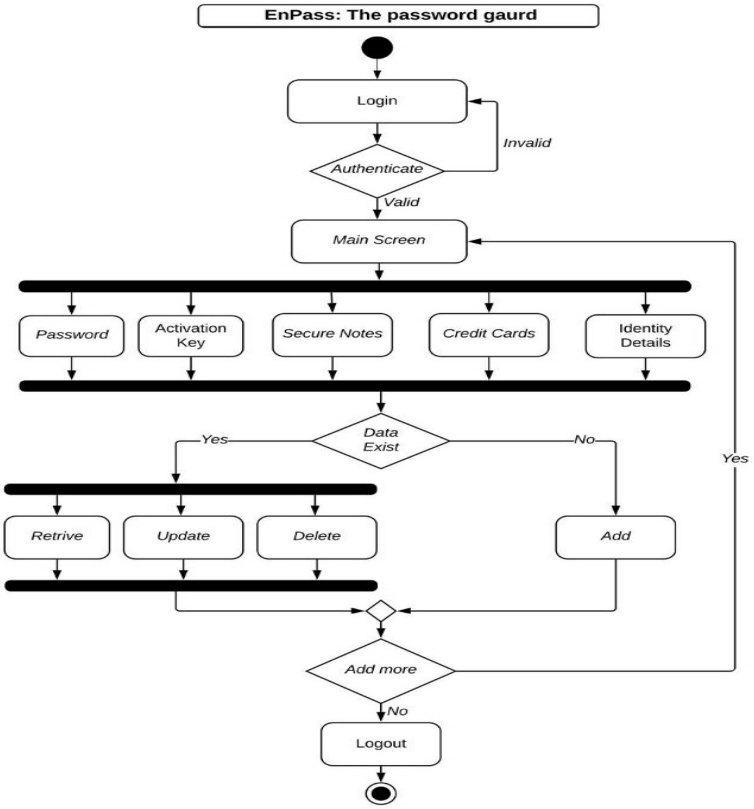


Figure 2.2: Flow of the System

IV. AES 256-BIT ENCRYPTION

AES was created because of the necessities of the U.S. government. In 1977, government offices depended on the Data Encryption Standard (DES) as their encryption algorithm. AES is a symmetric-key cipher which means it has the same security key used for both the receiver and the sender for the process of encryption and decryption of the data. By contrast, asymmetric-key cipher systems use a different keys one for encryption and other for decryption process of data. Asymmetric keys are best for external file transfers, whereas symmetric keys are better suited to internal encryption. The advantage of a symmetric-key cipher system like AES over the asymmetric-key cipher is their speed. Because a symmetric key algorithm requires less computational power than an asymmetric one, it's faster and more efficient to run. AES is also called as a block cipher. In this type of cipher, the information which is encrypted also known as plaintext which is divided into paragraphs called as blocks .The block size of AES consist of 128 bit, in which divide the data into a 4x4 array each consist of 16 bytes, hence there are 8 bits per byte, then the total of each block will became 128 bits. The size of the encrypted data remains the same: 128 bits of plaintext yields 128 bits of cipher text.

V. HOW DOES AES 256 WORKS?

The basic principle of all encryption is that each unit of data is replaced by a different one according to the security key. This process in AES is done by the process of substitution and permutation network because of which AES brings more security as it uses a process of Key Expansion technique in which by using the initial key, a series of new keys is generated called round keys. This becomes hard to break because the round key is generated by the modification of keys again and again. First, the initial key is added to the block using an XOR ("exclusive or") cipher, which is an operation built into processor hardware. Next the process of substitution takes place in which each byte is substituted with another, following a pre-determined table. Next, the Shiftrows takes place in which, the 1st row is not altered, in 2nd row 1-byte circular left shift is applied, in 3rd row 2-byte circular left shift is applied and in 3rd row 3-byte circular left shift is applied. In the forward Mixcolumn transformation each byte in a column is changed into a new data that is a function of those bytes in that column. Finally, AddRoundKey Transformation is performed where 128 bits state is bitwise XORed with the 128bits of round key. All the above process helps in yielding the cipher text. For AES decryption, the same process is carried out in reverse. Each stage of the AES encryption algorithm serves an important function. Using a modified key in each round becomes very hard to break. Substitution of the byte changes the data into non-linear manner, obscuring the relationship between the original and encrypted content. Shifting the rows and mixing the columns diffuses the data, transposing bytes to further complicate the encryption. Shiftrows transformation diffuses the data horizontally, while Mixcolumn transformation does so vertically. The result is a tremendously sophisticated form of encryption.

VI. IS AES 256 CRACKABLE?

AES 256 is virtually impenetrable using brute-force methods. In a 56-bit DES, for example, cipher key can be split in under a day, while AES would take billions of years to retrieve utilizing current registering innovation. Hackers would be foolish to even attempt this type of attack. Besides its advanced technology, the open nature of AES 256 makes it one of the best secure encryption protocols.

VII. IMPLEMENTATION

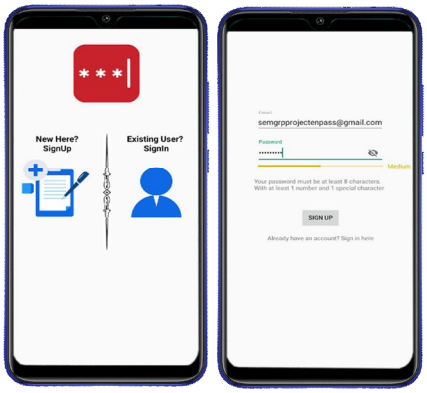


Figure 2.1: Home page Diagram

In figure 2.1, the user has to insert a valid Email-id and create a suitable password. If an EnPass account already exists, the user can click on the Existing User option and sign in by entering required Email-id and password

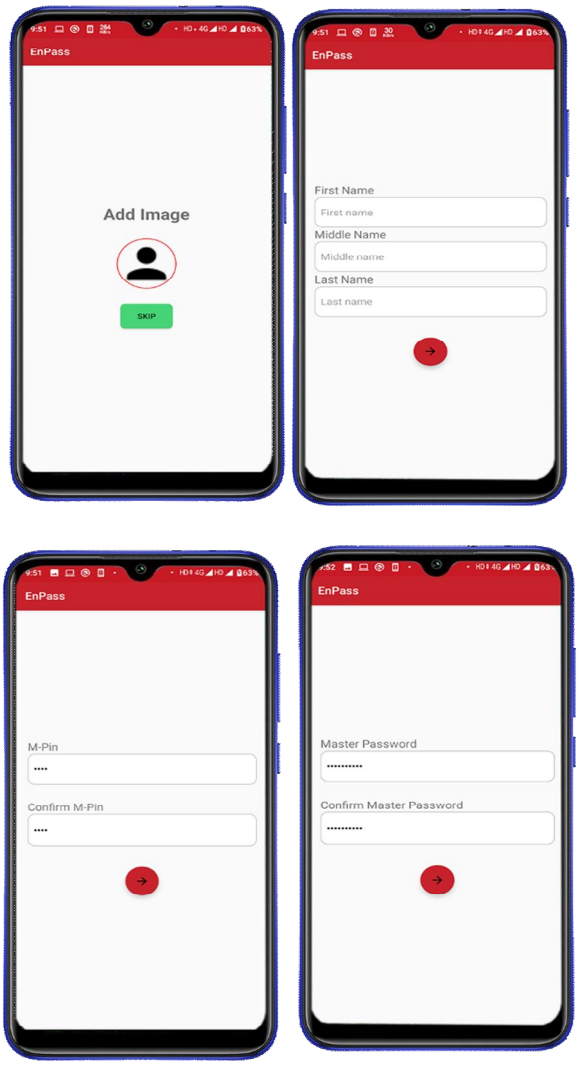


Figure 2.2: Registration Page Diagram

In figure 2.2, the user can set a profile image. Next, the user has to enter his/her full name and set an M-Pin and a Master Password.

Figure 2.3: Login Page Diagram

In figure 2.3, user can login the app by entering the M-Pin.

Figure 2.4:Main Menu Diagram

In figure 2.4, the user gets different options for storing different kind of credentials

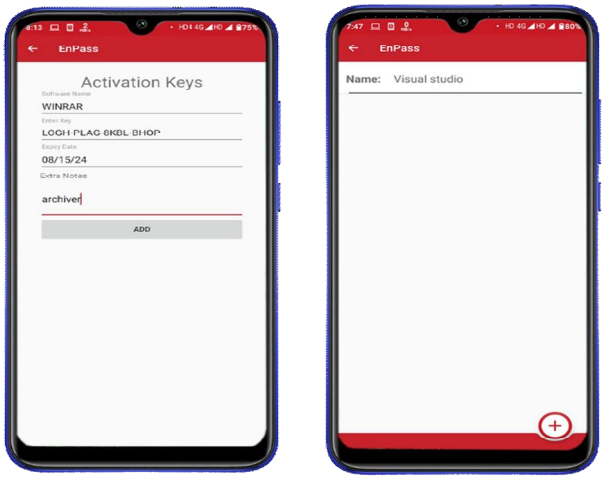


Figure 2.5: Storing Details Diagram

In figure 2.5, user can save various confidential details-Activation key, password, credit card details, notes etc.

Figure 2.6: Retrieving Page Diagram

In figure2.6, the user can easily access and retrieve the saved data by entering the Master Password.

VIII. CONCLUSION

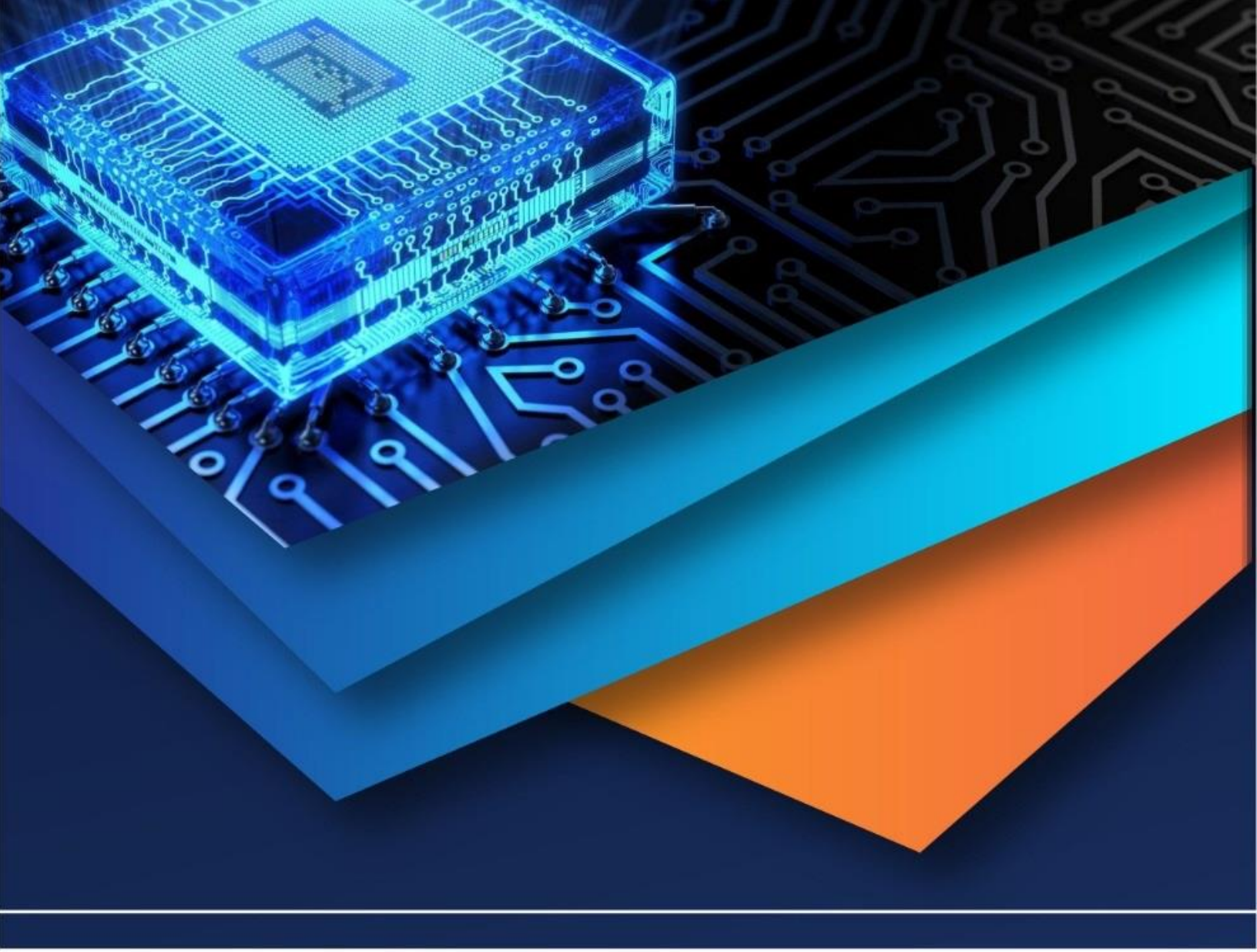
This system is mainly proposed for storing the passwords and credentials in a secure way. This system will help the users to store their passwords, ID details, card details, activation keys, and also secure notes. This application uses AES-256 encryption technique therefore the password will be stored in an encrypted form. So hereby, by using this implemented system user will be able to store and manage their confidential details more securely.

IX. ACKNOWLEDGMENT

We would like to express our acknowledgement to those people, without whose contribution, support and guidance this report would not have seen the light of the day. We would like to extend our gratitude to many people who helped to bring this research project to function. First, we would like to thank our guide, Asst. Professor Ms. Farhin Mansur for providing us the opportunity. We are so profoundly thankful for her assistance, polished methodology, important direction and backing all through this undertaking and through whole program of study that we need more words to communicate our deep and sincere appreciation. At long last, we should offer our prideful thanks to our parents for offering unfailing help and ceaseless support during the time of study and through the procedure. This achievement would not have been possible without them.

REFERENCES

[1] "SPHINX: A Password Store that Perfectly Hides Passwords from Itself", Maliheh Shirvanian∗, Stanislaw Jarecki†, Hugo Krawczyk‡ and Nitesh Saxena∗, ∗University of Alabama at Birmingham, †University of California at Irvine, IEEE 37th International Conference on Distributed Computing Systems, 2017.
[2] "NoKey - A Distributed Password Manager", Florian Zinggeler, Supervised by: Simon Tanner, Gino Brunner, Prof. Dr. Roger Wattenhofer, Distributed Computing Group Computer Engineering and Networks Laboratory ETH Zurich, June 7, 2018.
[3] "Password Assistance", Moritz Horsch, Johannes Braun, and Johannes Buchmann, Open identity summit, 2017.
[4] LastPass, homepage. https://www.lastpass.com/.
[5] 1Password, homepage. https://1password.com/.
[6] Dashlane, homepage. https://www.dashlane.com/.
[7] KeePass, homepage. https://keepass.info/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY