

Biometric And Password Authenticated Secure Android Notes

Kushagra Pathak¹, Rahulkumar Ankola², Gauravi Mittal³, Aditi Akhauri⁴

¹Computer Science and Engineering with specialization in Information Security, ²Computer Science and Engineering, ³Computer Science and Engineering with specialization in Information Security, ⁴Computer Science and Engineering with specialization in Information Security

¹ kushagra.pathak2020@vitstudent.ac.in, ² rahulkumar.ankola2020@vitstudent.ac.in,
³ gauravi.mittal2020@vitstudent.ac.in, ⁴ aditi.akhauri2020@vitstudent.ac.in

1. Abstract

We will build an effective system for securing notes using fingerprint authentication. In contrast to the traditional method of requiring a password to enter, this system can also be referred to as Keyless Authentication. This system does not require registration, but only the phone's owner has access to these messages because it looks for the owner's print. This system can be used for private notes, personal diaries, or significant notes; it can be given numerous titles, but it performs the same function of capturing notes and keeping them hidden from anybody other than the phone's owner. This app will not work if your phone does not have a biometric function. The user can create new notes, update existing notes, and delete notes.

Keywords: AES, Secure Notes,

2. Introduction

Everyone needs a safe space to store their valuable information like passwords and private notes. Regular notes applications available on the store are not secure enough. Although some applications provide a login security before one can access the notes stored by the user, it is not very hard for hackers to retrieve the username and password and login to the system.

SecureNotes is an application with which the user can access their notes with username or password which are encrypted with AES-128 which makes it infinitely tougher for hackers to get the actual username and password. Apart from this, the user also has an option to access the notes with fingerprint authentication which means that none other than the phone owner itself can access the notes with this feature.

3. Literature Survey

[1] This application uses fingerprint technology to verify authenticity in which capacitive fingerprint scanners produce a picture of hills and valleys that form fingerprints. But instead of hearing the print using light, capacitors use electrical energy. The sensor is made of one or more semiconductor chips that contain a series of tiny cells and that help in the protection of the data stored in these applications.

Another study used fingerprint authentication in digital signature based on X.509 certification infrastructure. A unique feature of this study is that users have been able to download third-party algorithms to customize protocols.

[2] In this paper, it is proposed how to prevent ransomware on the Android platform. The proposed method can monitor file events that occurred when ransomware accessed and copied files. This process can detect and remove ransomware using CPU and I / O usage and information stored in the DB. This proposed method can detect ransomware modified patterns without obtaining information about ransomware. In addition, it can be used in the kernel and source framework for Android so that it can detect ransomware faster than other applications running at the app level. In addition, it can continuously monitor ransomware without downloading separately or updating.

[3] In this paper, the authors review the fingerprint recognition technology, a popular biometric security feature, to improve web login.

authentication mobile application. In this paper they focus on summarizing research work done on methods of matching fingerprints, recognition strategies, and performance analysis. In this paper they also review some of the research problems and tools used under the mobile application development.

[4] The authors use fingerprint recognition technology to improve the traditional way of attendance. The paper focuses on a smart attendance system where an android device will be used as a smart ID card. Employee's mobile phone is connected to the system through its MAC address. The software is developed in Android Studio connected with the SQL database.

[5] Mobile banking services have become one of the most important online applications, being offered to many banks around the world. The end user can manage the accounts or make certain payments without having to go to the actual bank office. This is why security concerns about securities should be accounted for and the bank should offer a variety of integrated and sign-in and payment methods, in order to increase confidence in their mobile banking services. This research paper will introduce some of the two areas: mobile banking and fingerprint verification process. During the research, they developed a Java-based mobile application to simulate access to Mobile Banking to access login and payment options. The authors also tested the app for this app and, as a result, found it to be 100% secure and 100% efficient and easy to use.

[6] The proposed system can overcome the security issues faced in the present situation. The 3 level security in the system can help the user for accurate security. The main reason for the proposed system is to secure the user living place, working place or to keep their valuable things, documents in a protected way.

In case of authorized user, the system allows fingerprint sensor to validate the person followed by sending either PICTURE password or OTP via SIM to the user registered mobile number saved in database (local SD card) in order to access.

[7] Biometrics systems will be ubiquitous in the community, such as education, government, smart cities etc. Technologies that have made smart phones an integral part of our daily lives are GSM, GPRS, 3G, 4G, WiMAX, Blue-tooth and Wi-Fi.

However, this rapid growth of technology and the widespread use of smartphones make them less susceptible to malware and other security.

breaking the attack. In this paper, we are reviewing fingerprint recognition technology, a popular biometric security feature, to develop a web-based authentication application. In this paper the authors focus on summarizing research work done on methods of matching fingerprints, observation techniques, and performance analysis. In this paper they also review some of the research issues and tools used under the development of mobile applications.

[8] In this research they compared the accuracy of fingerprint authentication and behavioral biometric authentication using 2D and 3D gestures. Each method is evaluated through a separate Android application. The collected Preliminary results from 6 participants each of whom used the apps for 5 days suggest more and comprehensive data collection and in-depth data analysis are necessary to better understand the capability of gesture based behavioral biometrics in user authentication.

[9] In this paper they have used C# using the .NET functions, using the AES encryption standards and the Android-specific tools in the Xamarin platform provided for Visual Studio. This application implies creating a software that uses a hardware plug-in of the Android-powered device. It's an interesting application for code writing, because it includes the use of information transfer between the device components.

4. Work Flow

(i) We sortlisted 3 project ideas.

(ii) Then we selected “Biometri and password Authenticated Secure Android Notes”

(iii) Review 0

(iv) Thought of adding additional feature of security questions

(v) Literature survey for review 1

(vi) Implemented XML for the application

(vii) Implemented Biometric security

(viii) Review 2

(ix) Database added

(x) Implemented AES for password login

(xi) Added security Questions

(xii) Final Presentation

5. System Design

5.1 Algorithm for AES:

AES works with data bytes instead of bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of input data at a time.

The number of cycles depends on the length of the key as follows:

128 bit key - 10 rounds

192 bit key - 12 cycles

256 bit key - 14 rounds

Creation of rotating keys:

The Key Schedule algorithm is used to calculate all the round keys from the key. The first key is therefore used to create many different round keys that will be used in the corresponding encryption round.

Encryption:

AES considers each block as a 16 byte grid (4 byte x 4 byte = 16) in a large column system.

```
[b0 | b4 | b8 | b12 |  
| b1 | b5 | b9 | b13 |  
| b2 | b6 | b10 | b14 |  
| b3 | b7 | b11 | b15]
```

Each round has 4 steps:

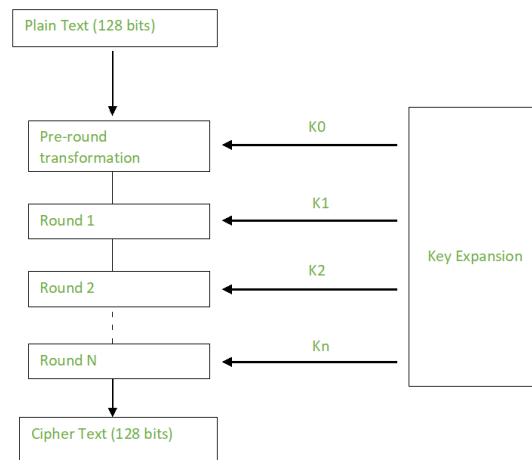
SubBytes

ShiftRows

MixColumns

Add a round key

The last round does not have a MixColumns cycle.



SubBytes changes and ShiftRows and MixColumns make concessions to the algorithm.

SubBytes:

This step uses replacement.

In this step each byte is replaced by another byte. It is made using a viewing table also called the S-box. These changes are made in such a way that the byte is never changed by itself and can also be changed in another way which is to recommend the current byte. The result of this step is a 16 byte matrix (4 x 4) as before.

The next two steps apply permission.

ShiftRows:

This step is as sound as it sounds. Each line is moved by a certain number of times.

The first line was not moved

The second row is moved once to the left.

The third row is moved twice to the left.

The fourth row is moved three times to the left.

(Left circle rotation is performed.)

$[b_0 | b_1 | b_2 | b_3] \rightarrow [b_0 | b_1 | b_2 | b_3]$

$[b_4 | b_5 | b_6 | b_7] \rightarrow [b_5 | b_6 | b_7 | b_4]$

$[b_8 | b_9 | b_{10} | b_{11}] \rightarrow [b_{10} | b_{11} | b_8 | b_9]$

$[b_{12} | b_{13} | b_{14} | b_{15}] \rightarrow [b_{15} | b_{12} | b_{13} | b_{14}]$

MixColumns:

This step is basically a multiplication of the matrix. Each column is repeated with a specific matrix and thus the location of each byte in the column is changed as a result.

This step was skipped in the last round.

$[c_0] = [2 \ 3 \ 1 \ 1] [b_0]$

$[c_1] = [1 \ 2 \ 3 \ 1] [b_1]$

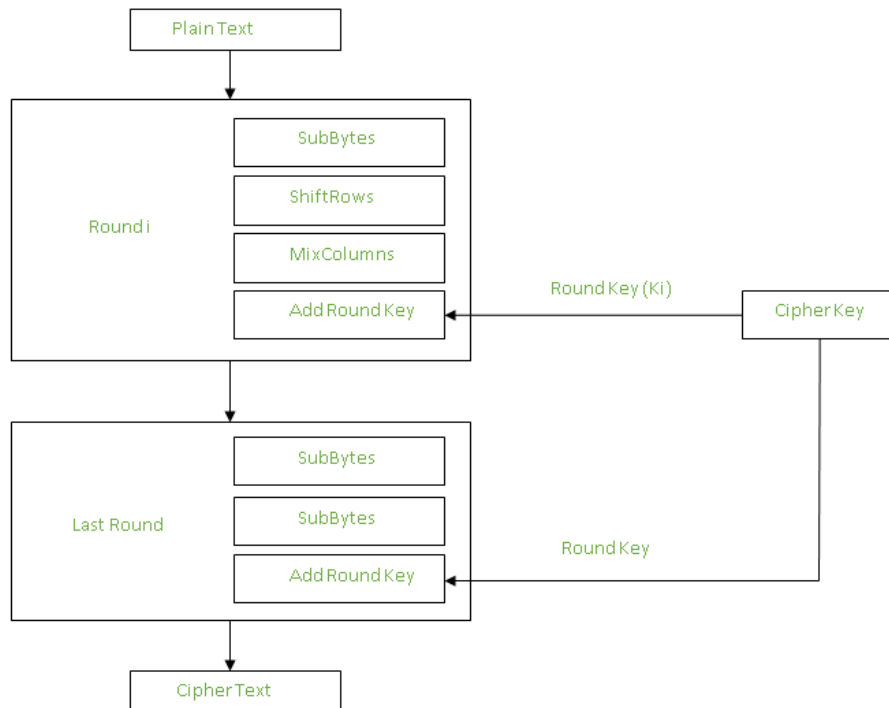
$[c_2] = [1 \ 1 \ 2 \ 3] [b_2]$

$$[c3] = [3 \ 1 \ 1 \ 2] [b3]$$

Add Round Keys:

Now the result of the previous phase XOR-ed result with the corresponding round key. Here, 16 bits are not considered as a grid but as 128 bits of data.

After all these rounds 128 bits of encrypted data is restored as output. This process is repeated until all encrypted data enters this process.



Decryption:

Stages in rounds can be easily reversed as these sections have an argument which when done returns the changes. Each 128 blocks go through 10, 12 or 14 rounds depending on key size.

The stages of each cycle in text writing are as follows:

Add a round key

Inverse MixColumns

ShiftRows

Contrasting SubByte

The encryption process is the encryption process performed in retrospect to explain the steps with significant differences.

Inverse MixColumns:

This step is similar to the MixColumns step in encryption, but differs in the matrix used to perform the function.

$$\begin{array}{cccc}
 [c_0] & & [2 & 3 & 1 & 1] & [b_0] \\
 |c_1| & = & |1 & 2 & 3 & 1| & |b_1| \\
 |c_2| & & |1 & 1 & 2 & 3| & |b_2| \\
 [c_3] & & [3 & 1 & 1 & 2] & [b_3]
 \end{array}$$

Inverse SubBytes:

The Inverse S-box is used as a viewing table and is used for adjustable bytes during the secret release.

Summary:

A set of AES commands are now integrated with the CPU (provides GB / s output) to improve the speed and security of applications that use AES encryption and decryption. Although it has been 20 years since its launch we have failed to break the AES algorithm as it does not even happen with current technology. So far the only risk left is the use of the algorithm.

5.2 Algorithm for Biometric Implementation:

To add biometric verification to your application using the Biometric library, complete the following steps:

In the module file of your build.gradle app, add dependencies to the androidx.biometric library.

At work or in the clipboard of the biometric login box, display the box using the logic.

Verify using biometric or lock screen

You can use a secret key that allows authentication using biometric credentials or screen lock credentials (PIN, pattern, or password). When setting up this key, specify the verification time. During this time, your application can perform many cryptographic functions without the user having to re-verify.

To encrypt sensitive information after a user has authorized using biometric or screen lock information, complete the following steps:

Determine how the user is authorized

After verifying user authentication, you can check if the user is authorized using device details or biometric data by dialing `getAuthenticationType()`.

Show login command

To display a system command that asks a user to authorize using biometric credentials, use the Biometric library. This chat provided by the system is compatible with all applications that it uses, creating in-depth user information that is very reliable.

5.3 Implementation

Sperate SQL databases are created for storing username-password, security question and its answer, and for notes data. This is done with extending the database class to SQLiteOpenHelper helper class.

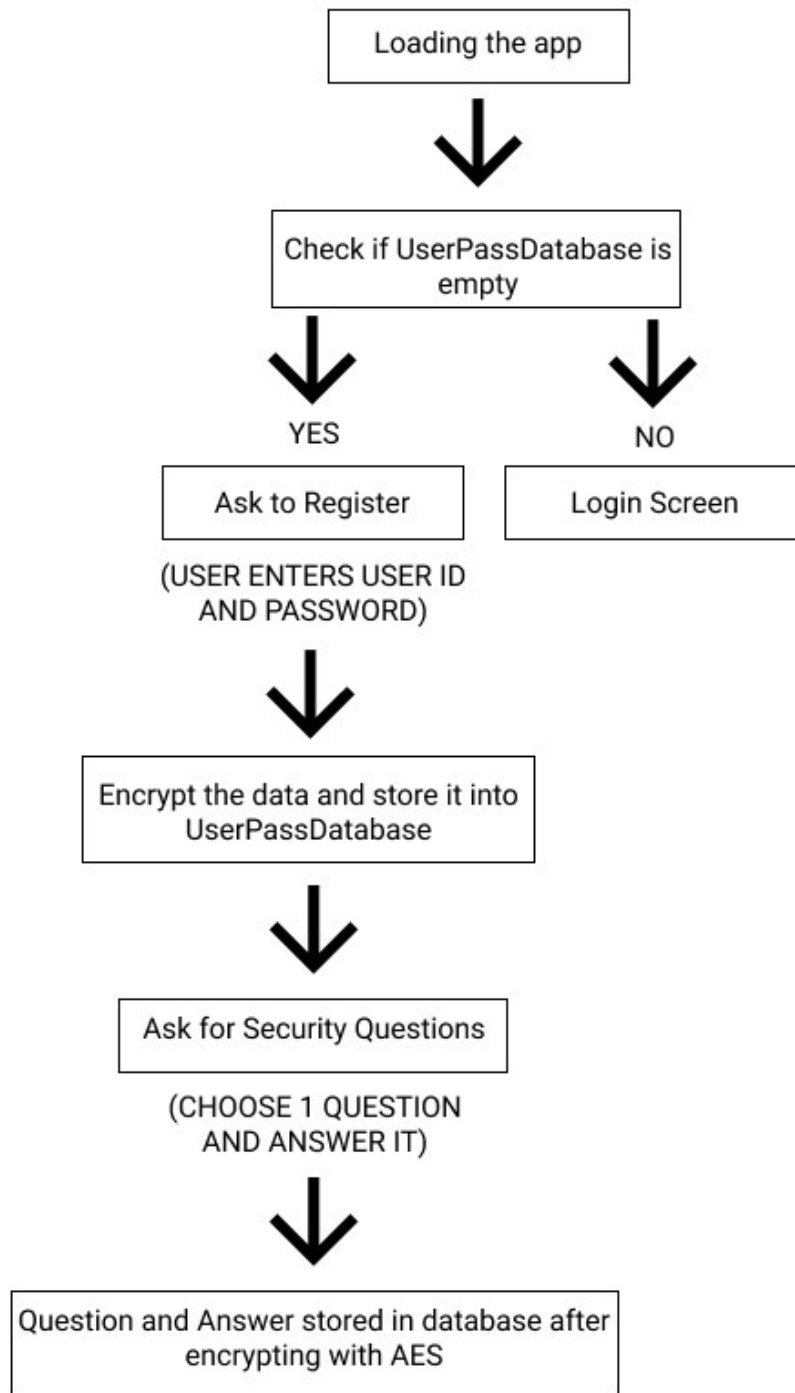
On start the system checks if the UserPassDatabase is empty. If yes, it asks the user to register for the first time. On submission, username and password both are encrypted with AES-128 and the ciphered username password is added to the database. Same is done on submission of security question. The question and answer are encrypted and inserted into QuestionsDatabase.

While logging in, it decrypts the data first and then checks with the inserted values.

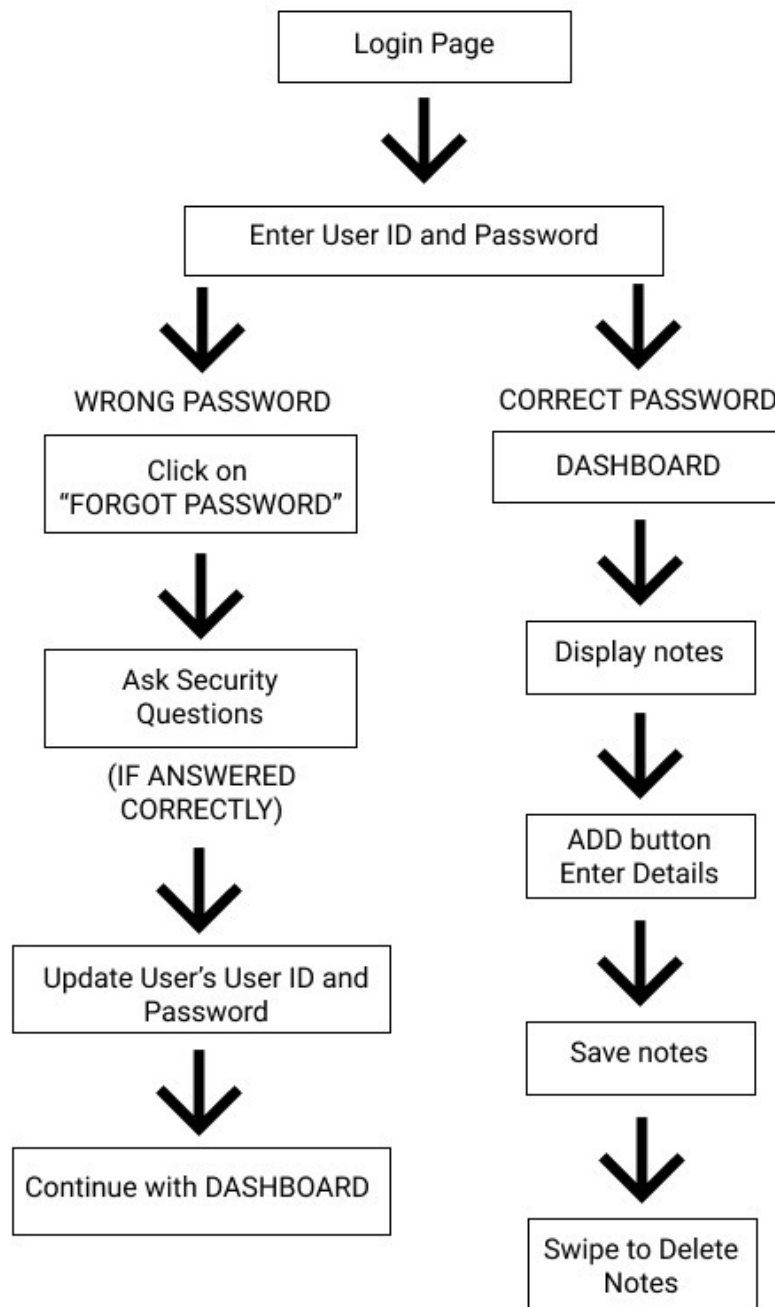
Notes title and description is added to NotesDatabase directly on creation or updation.

Notes are displayed in a RecyclerView in the activity_dashboard_screen.xml rather than a ListView for a quick and efficient display of data. MyAdapter is created to set the RecyclerView with all the notes available.

6. Flowchart



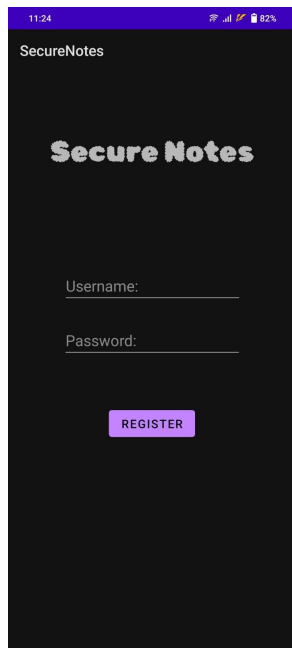
After the user has got his/her User ID and password , flow will go to login page



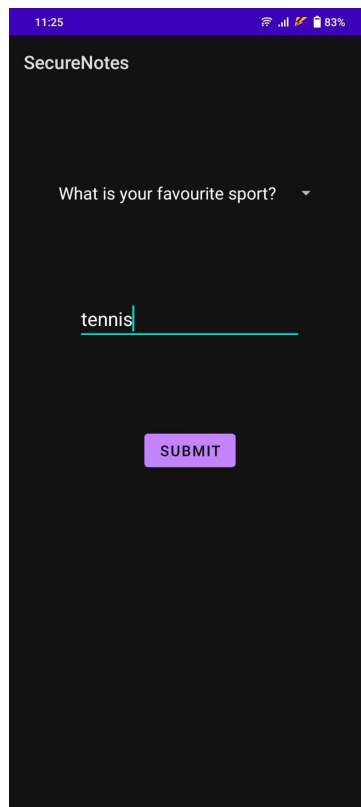
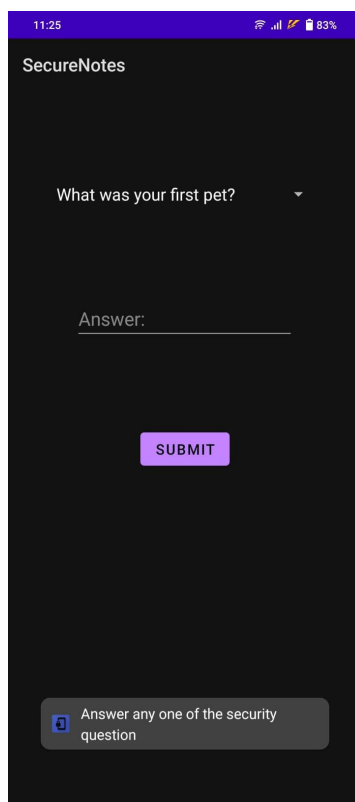
Hence, the above 2 flow charts explain the complete flow of our model which we have implemented.

7. Experimental Results and Analysis

7.1 App is opened for the first time

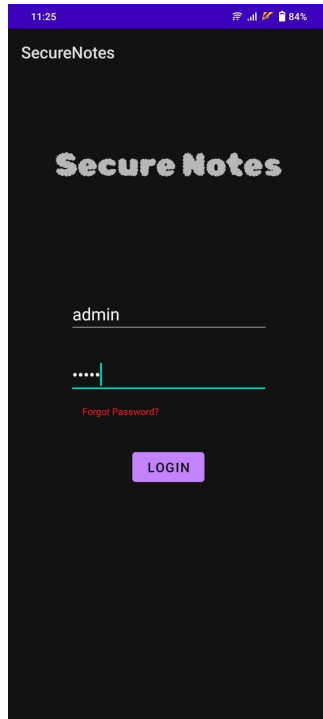


7.2 Security Question after setting username and password



7.3 Login page once the username, password and security question is set

7.3.1 Correct credentials



SecureNotes

Secure Notes

admin

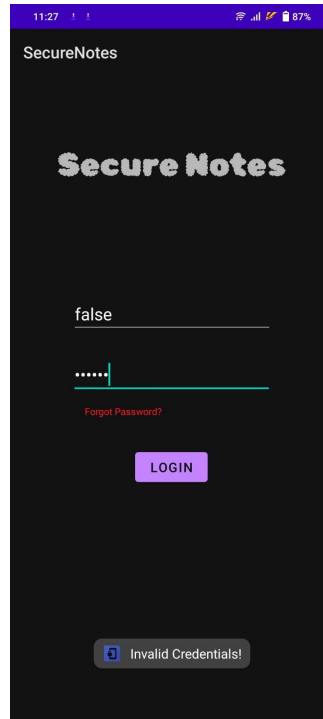
.....

Forgot Password?

LOGIN

This screenshot shows the login page of the SecureNotes application. The username field contains 'admin' and the password field contains six dots. A 'Forgot Password?' link is visible below the password field. The 'LOGIN' button is at the bottom.

7.3.2 Incorrect credentials



SecureNotes

Secure Notes

false

.....

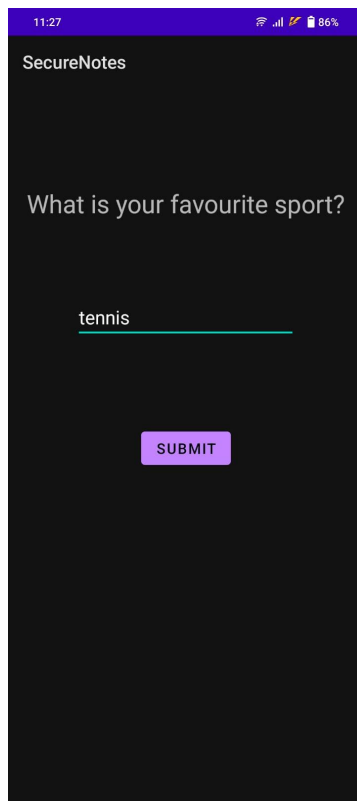
Forgot Password?

LOGIN

Invalid Credentials!

This screenshot shows the login page of the SecureNotes application. The username field contains 'false' and the password field contains six dots. A 'Forgot Password?' link is visible below the password field. The 'LOGIN' button is at the bottom. Below the login button, a message 'Invalid Credentials!' is displayed in a grey box.

7.3.2.1 Forgot password



SecureNotes

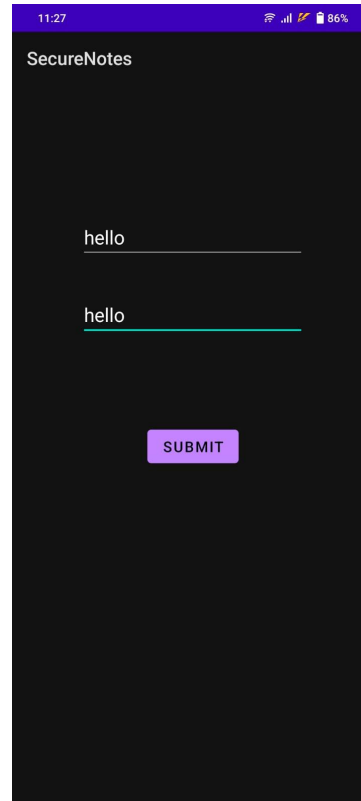
What is your favourite sport?

tennis

SUBMIT

This screenshot shows the forgot password page of the SecureNotes application. The question 'What is your favourite sport?' is displayed. The answer field contains 'tennis'. A 'SUBMIT' button is at the bottom.

7.3.2.2 Reset username and password



SecureNotes

hello

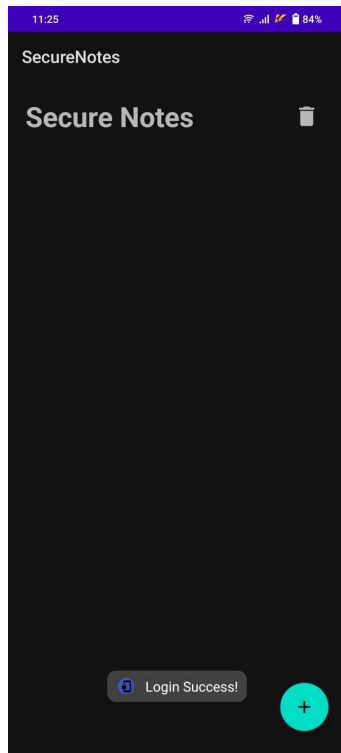
hello

SUBMIT

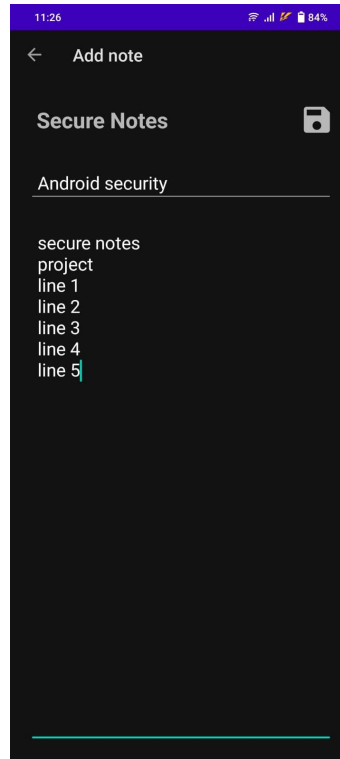
This screenshot shows the reset username and password page of the SecureNotes application. The new username field contains 'hello' and the new password field contains 'hello'. A 'SUBMIT' button is at the bottom.

7.4 Dashboard Screen after successful login

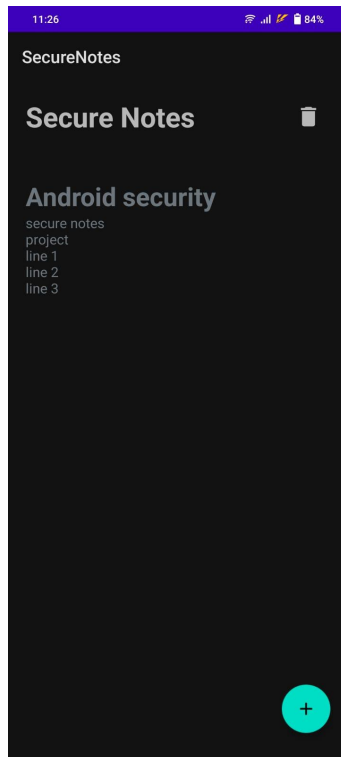
7.4.1 No notes added



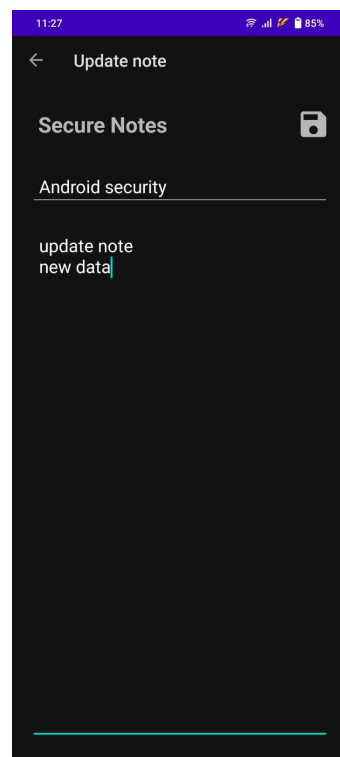
7.4.2 Add note



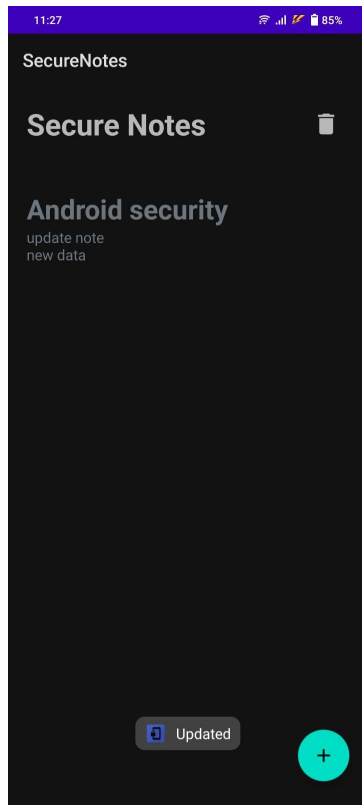
7.4.3 After Adding notes



7.4.4 Update note



7.4.5 Dashboard after updating note



8. Requirements

8.1 Software Requirements

- Windows XP, Windows 7
- Android Studio

8.2 Hardware Requirements

- Processor – i3
- Hard Disk – 5 GB
- Memory – 1GB RAM
- Android Phone with 2 prerequisites
 1. Marshmallow and Higher
 2. Build in Biometric.

9. Drawbacks

9.1 Drawbacks of AES algorithm

- It uses too simple algebraic structure.
- Every block is always encrypted in the same way.
- Hard to implement with software.
- AES in counter mode is complex to implement in software taking both performance and security into considerations.

9.2 Drawbacks of biometric authentication

- Costs – Significant investment needed in biometrics for security
- Data breaches – Biometric databases can still be hacked
- Runs only on Build in Biometric phones.
- Tracking and data – Biometric devices like facial recognition systems can limit privacy for users
- Bias – Machine learning and algorithms must be very advanced to minimize biometric demographic bias
- False positives and inaccuracy – False rejects and false accepts can still occur preventing select users from accessing systems
- Creating a phony fingerprint
- The scourge of ‘MasterPrints’ fooling popular smart devices

10. Future Work

We can even add more features to this application like recycle bin, search bar, notification reminder, adding date with notes, etc.

11. Conclusion

Through this report, you understood what a Finger-Print Authentication Notes app with additional layer of AES is. We discussed the application’s flow,drawbacks,software/hardware requirements,and understood how to implement Fingerprint Secured Android Notes on Android Studio and addition of additional layer of AES algorithm for more security.

12. References

- [1] Vaibhav Garg , Siddharth Yadav , Rishabh Kamal, “Android Notes using FingerPrint Authentication”,International Journal of Science and Research (IJSR) ISSN: 2319-7064 SJIF (2019): 7.583 , Volume 10 , Issue 5, May 2021.
- [2] Jeong Hyun Yi, Aziz Mohaisen, Sean Yang, Ching-Hsien Hsu, "Advances in Mobile Security Technologies", *Mobile Information Systems*, vol. 2016, Article ID 9501918, 2 pages, 2016. <https://doi.org/10.1155/2016/9501918>.
- [3] Akanksha Bali, Shivangi Goswami, and Shagun Sharma, “Biometrics Security in Mobile Application Development and its Applications,” International Journal of Scientific and Technical Advancements, Volume 5, Issue 1, pp. 51-60, 2019.
- [4] H. Adal, N. Promy, S. Srabanti and M. Rahman, "Android based advanced attendance vigilance system using wireless network with fusion of bio-metric fingerprint authentication," 2018 20th International Conference on Advanced Communication Technology (ICACT), 2018, pp. 217-222, doi: 10.23919/ICACT.2018.8323702.
- [5] L. Sharma and M. Mathuria, "Mobile banking transaction using fingerprint authentication," 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 1300-1305, doi: 10.1109/ICISC.2018.8399016.
- [6] N. Meenakshi, M. Monish, K. J. Dikshit and S. Bharath, "Arduino Based Smart Fingerprint Authentication System," 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), 2019, pp. 1-7, doi: 10.1109/ICIICT1.2019.8741459.
- [7] Akanksha Bali¹ , Shivangi Goswami² , Shagun Sharma³ , “ Biometrics Security in Mobile Application Development & its Applications” , Computer Engineering Department, YCET, Jammu, J&K, India-181205
- [8] Diana K. Anguiano Cervantes* , Ghouri Mohammad Saaduddin† , Yanyan Li† , Mengjun Xie† * California State University–San Marcos 2016 IEEE Conference on Communications and Network Security (CNS) DOI: 10.1109/CNS.2016.7860517
- [9] Naomi Estera Costea ,Department of Computer Science and Information Technology, Elisa Valentina Moisi ,Department of Computer Science and Information Technology, 2019 15th International Conference on Engineering of Modern Electric Systems (EMES) , DOI: 10.1109/EMES.2019.8795100