



School of Computer Science and Engineering

BCI3002 – Disaster Recovery and Business Continuity Management

**Fall 2019-2020
Slot – C1+TC1**

J-COMPONENT REPORT

IMPLEMENTATION OF A SECURE PASSWORD MANAGER

FACULTY: PROF. ARIVOLI A.

Batch Details

| Sl No | Reg No | Name |
|-------|-----------|----------------------|
| 1 | 17BCI0157 | Charizma Gupta |
| 2 | 17BCI0181 | Rohith Rajeev Pillai |
| 3 | 17BCI0185 | Adrijeet Deb |

Table of Contents

| | |
|-------------------------------------------------------------|----|
| Abstract | 3 |
| Introduction | 3 |
| Literature Survey | 4 |
| Methodology..... | 7 |
| Modules description | 7 |
| Disaster Prevention and Authenticated Recovery methods..... | 8 |
| Implementation | 8 |
| Hardware and Software requirements | 8 |
| Implementation Snapshots..... | 8 |
| Vulnerability analysis [Potential Disasters] | 12 |
| SQL Injection..... | 12 |
| Session Hijacking..... | 12 |
| Brute Force Attack | 12 |
| Information Theft..... | 13 |
| Disaster Prevention | 13 |
| NoSQL Database..... | 13 |
| Code..... | 13 |
| Hash..... | 14 |
| Code..... | 15 |
| Encryption..... | 15 |
| Byte Substitution (SubBytes)..... | 16 |
| Shiftrows..... | 16 |
| MixColumns | 17 |
| Addroundkey..... | 17 |
| Decryption Process..... | 17 |
| Code..... | 18 |
| Biometric | 18 |
| Code..... | 19 |
| Future Works | 20 |
| Conclusion – Overall security of the application..... | 21 |
| References | 22 |

Abstract

One of the primary objectives of the Disaster Recovery process is protection and recovery of data. In our project, we have dived into the implementation of the protection of data through three factor authentication and additional security features like encryption and hashing. Through this Password Manager application named 'Secure', we have demonstrated how data backups can be digitally protected. The application allows users to store all their login credentials securely as a backup in case they are forgotten.

Users must login to the app with authorized credentials and once they're in, they can retrieve all the usernames and passwords they've stored or add new ones.

We plan on using a large array of security techniques like encryption and hashing along with the three-factor authentication to keep the data stored in an illegible manner. The authentication process involves passwords, OTP and Biometric authentication.

Introduction

One of the many disadvantages of living in the information age is that a lot of data is locked behind accounts. Certain websites require you to create an account to view/download much needed data. There's also social media and online forums all needing unique accounts, from the same individual, to be functional. This leads to a whole slew of accounts that one needs to memorize the details for.

Our project aims to bring a secure and convenient solution to this problem. We aim to create a password manager, an application that allows for the recording of login details.

Through this password manager, we aim to demonstrate disaster prevention and recovery authentication of important data that cannot afford to be compromised.

Three-factor authentication is an authentication method in which a computer user is granted access only after successfully presenting three pieces of evidence (or factors) to an authentication mechanism: knowledge (something the user and only the user knows), possession (something the user and only the user has), and inference (something the user and only the user is).

Our project implements this by using Passwords, OTPs in SMS form as well as Biometric in form of fingerprints.

Literature Survey

| Ref | Issue Addressed | Methodology/Algorithm | Advantages | Limitations |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| [1] | Three pillars of cybersecurity are authentication, security, and confidentiality. Users are presented with numerous solutions to enhance the security of login password methods. This is primarily done by using two-factor authentication methods. The combination of one-factor authentication schemes is two-factor authentication. | Due to the increasing need for privacy and security in this technological age, the increasing popularity and acceptance of two factor methods is driven. The effectiveness and acceptance of adapted security activities rely in large measure on their user-friendliness and ease of implementation. | Three factor authentications will become mainstream in the future. It is necessary to develop the technology more so that we can deploy it when it is required. | Much harder to implement than 2 factor authentications. Will require more compute power than 2-factor authentication. |
| [2] | Mobile equipment has now become an essential tool to provide the company's internal network and services with access to business resources. This is why wireless communication protection is becoming increasingly important. | This paper illustrates how to simultaneously improve the IKE process by implementing a mobile access system that is authenticated by multiple factor for mobile security communication. | Multi factor authentication is more secure than a single factor authentication. It is more robust and harder to crack. | It is quite complicated and not usable everywhere as it is hard to implement. Small organizations cannot provide multi-factor authentication. |

| | | | | |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [3] | Authentication of identity is the most important safety network and the first protection line for network security. | This article details the device configuration and authentication concept of shared authentication systems based on a mobile telephone SMS service based on the informative single-factor and two-factor authentication technology. | Two-factor authentication has still not been broken. It is one of the most widely used security models in the world. | It is hard to implement and requires additional computing power for utilizing the second factor. |
| [4] | A close look at password strength and the difficulty in breaking a password. This paper designs and implements a password strength checker called PwdStrength. | PwdStrength scores the user entered password against a number of factors and returns the score along with the classification of weak, fair, strong or invalid. The five factors are Length, Character set, Entropy, Predictability, Commonness | It checks for entropy or whether the password is a common one or not. It can also be frequently updated with respect to the common passwords list. | More secure alternatives to passwords exist. Human generated passwords are not random and easy to remember. Ease of access is not there if password are forced to be complicated |
| [5] | A Fingerprint lock system with three levels of security. User can unlock the system by using any two out of three levels. If an unauthorized person tries to unlock, alert message is sent to the admin in first attempt itself. | In case of authorized user, the system allows fingerprint sensor to validate the person followed by sending either PICTURE password or OTP via SIM using GSM module to the user registered mobile number saved in database (local SD card) in order to access the door. If the entered password matches, door will be opened automatically otherwise a message showing incorrect password will be displayed on TFT display and a notification will be sent to the owner that the security was tried to be breached. | <ul style="list-style-type: none"> Image password as a third security level Highly accuracy in term of security Relatively low cost so that everyone can use at their home, offices, shops, etc. Fingerprint enrollment is easier. Unlocking the system using any two security levels out of three security levels It helps to keep the place secure then other devices | Admin has to be alert on the notification from the system <ul style="list-style-type: none"> Users has to keeps their mobile phone secured, because hacker can hack their phone to get OTP and hacker may block the notification if they hack the phone |

| | | | | |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [6] | <p>Data breaches are drastically more common in today's society and so the need for true information security is greater than ever. So, to keep the privacy and confidentiality of people online, 3 factor authentications must be used</p> | <p>Using the MIT App inventor tool, an android app which employs facial recognition, username + password and One Time Password (mobile) authentication was created.</p> | <p>Using facial recognition in addition to both OTP and user details leads to three factor authentications. It's literally impossible to get through the facial recognition component without being the user.</p> <p>It doesn't take too long to get through the authentication step as all 3 can be done nearly simultaneously</p> | <p>This is a pretty simple implementation, and the researcher used a pre-built app development tool leading to less security overall.</p> |
| [7] | <p>This paper discussed the need for more than just 2 factor authentications. When you have only two out of the KNOW-HAVE-ARE triad, you're much more susceptible for attack.</p> | <p>Researchers tried to attack each factor of authentication on its own. OTP was attacked with Man in the middle attack, passwords were attacked with brute force and word list attacks, and so on</p> | <p>Having 1 factor is definitely not enough. But the researchers successfully demonstrated that two factors on its own also isn't solid enough in the case of any one factor being breached through by chance. 2 factors can very easily be reduced to 1 factor; however, 3 factor requires an extraordinary amount of luck to reduce to just 1 factor.</p> | <p>The research does not discuss generate any new ideas for 3 factor or further than 3 factor authentications, rather only discusses why 2 factors isn't good enough in today's society.</p> |

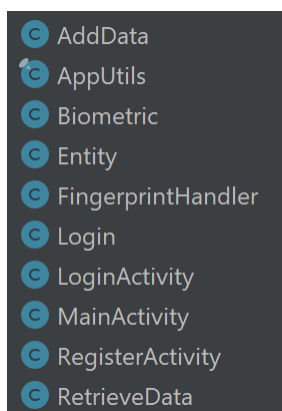
Methodology

Project architecture

Our password manager is an android app which will store any previously inputted details for future reference. The user will have to login initially with a master username and master password. Once they're inside, they can input any website name and the application will return their password for the chosen website.

We're using a firebase database for our backend with java (Android) front end.

Modules description



- **MainActivity:** Contains the OTP function. It asks for a phone number, generates the OTP, messages it to the phone and then autofill or can be manually filled for access to the rest of the application. The Generate OTP button sends the OTP containing SMS and the Verify and Sign In button authenticates the user.
- **RegisterActivity:** This is the sign-up page. It accepts the username, password and requests password reentry as confirmation to create new users for the application. The password is stored in hash form. The login button can be used to navigate to the login page.
- **LoginActivity:** The login page, accepts username and password to allow cross check with the database and allow access.
- **Login:** Creates the structure for the User table and helps define and add the attributes.
- **Add Data:** Users can store their login details of any website; it will be encrypted before storage for privacy and data protection.
- **Entity:** Creates the structure for the Entity table and helps define and add the attributes.
- **Biometric:** Contains the Biometric code. Defines what happens on success and failure or error in recognition.
- **FingerprintHandler:** Responsible for the Error Handling in the Biometric process.
- **Retrieve Data:** By inputting the correct identifier, users can extract their stored usernames and passwords from the database.

Disaster Prevention and Authenticated Recovery methods

- Password
- OTP
- Biometric
- Hash
- Encryption

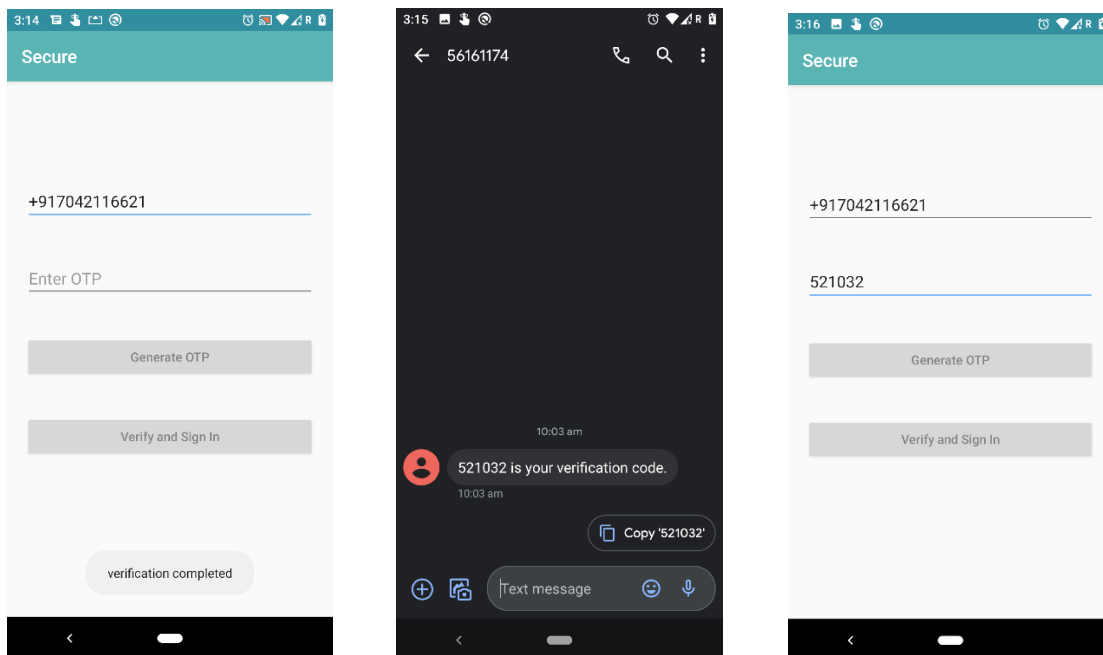
Implementation

Hardware and Software requirements

For the OTP feature, internet access is essential. For biometric features, the application must be using Android API 26 or above. The application is only compatible with Android Version Pie. Application currently only supports Android devices.

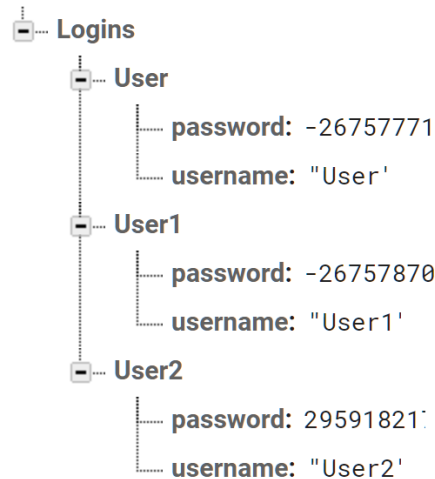
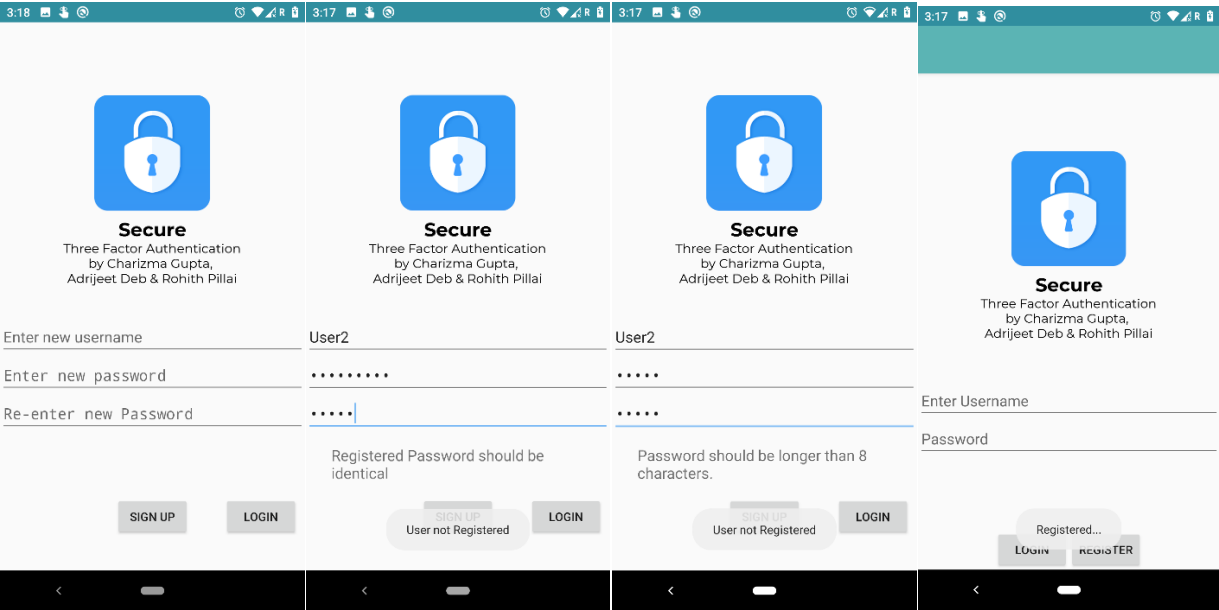
Implementation Snapshots

OTP



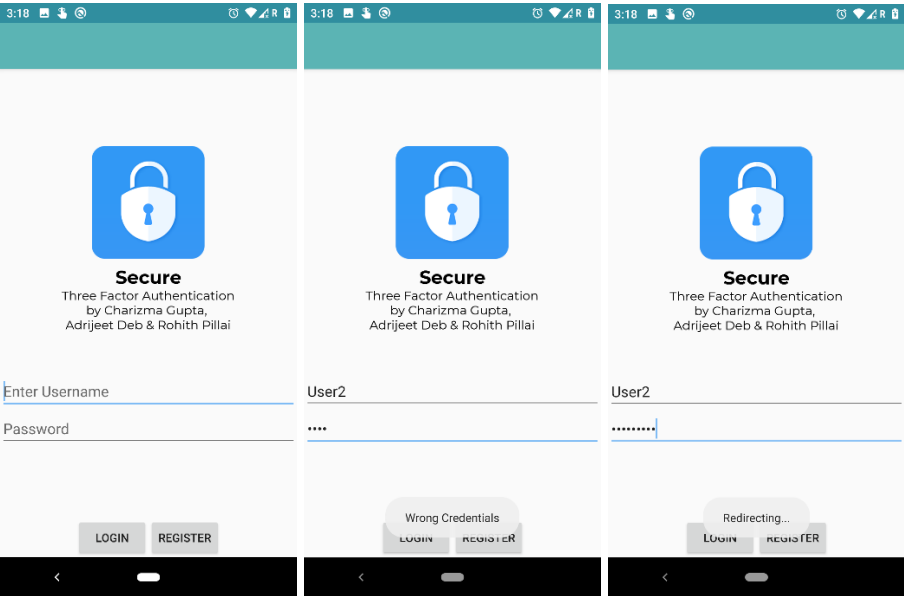
| Search by email address, phone number, or user UID | | | | | Add user | ↺ | ⋮ |
|----------------------------------------------------|-----------|-------------|-------------|------------------------------|-------------------|----------|-----|
| Identifier | Providers | Created | Signed In | User UID ↑ | | | |
| +919003433488 | 📞 | Jun 5, 2020 | Jun 5, 2020 | DN7I5SorzbZ7ee500JlcS4Zal3H2 | | | |
| +917042116621 | 📞 | Jun 5, 2020 | Jun 5, 2020 | rU6t87vt9Ac0PyjRM2Qybn9e3v32 | | | |
| | | | | | Rows per page: 50 | 1-2 of 2 | ⏪ ⏩ |

Registration






User Database Table

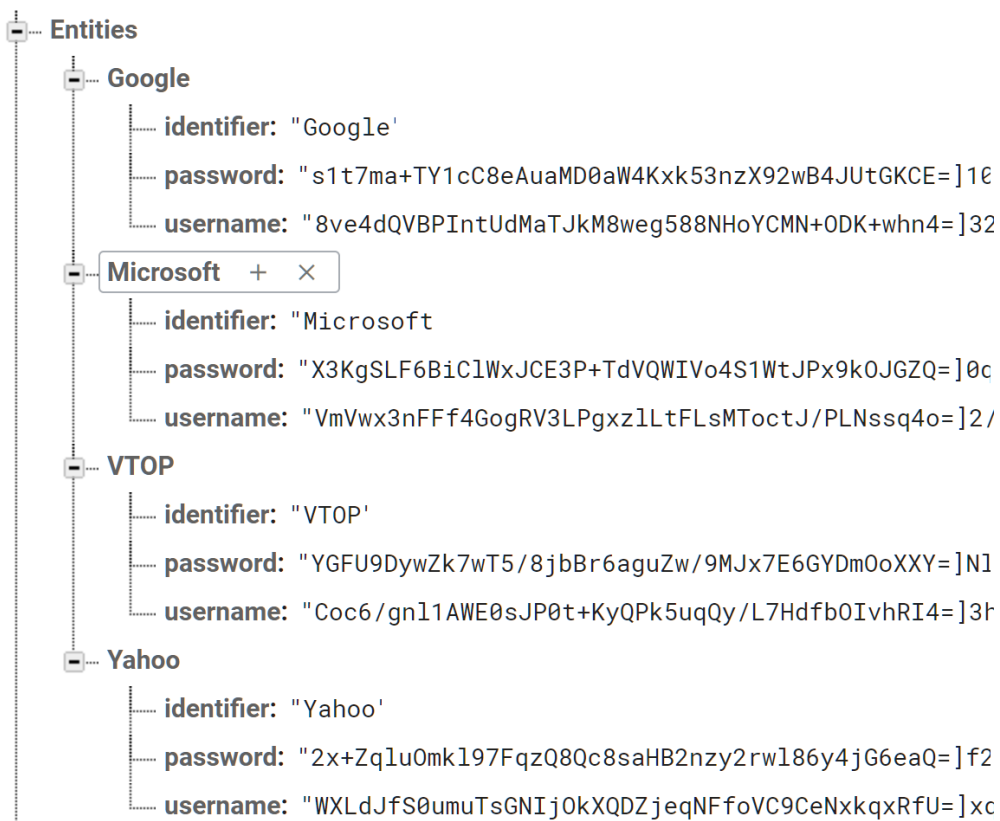
Login



Storing Data

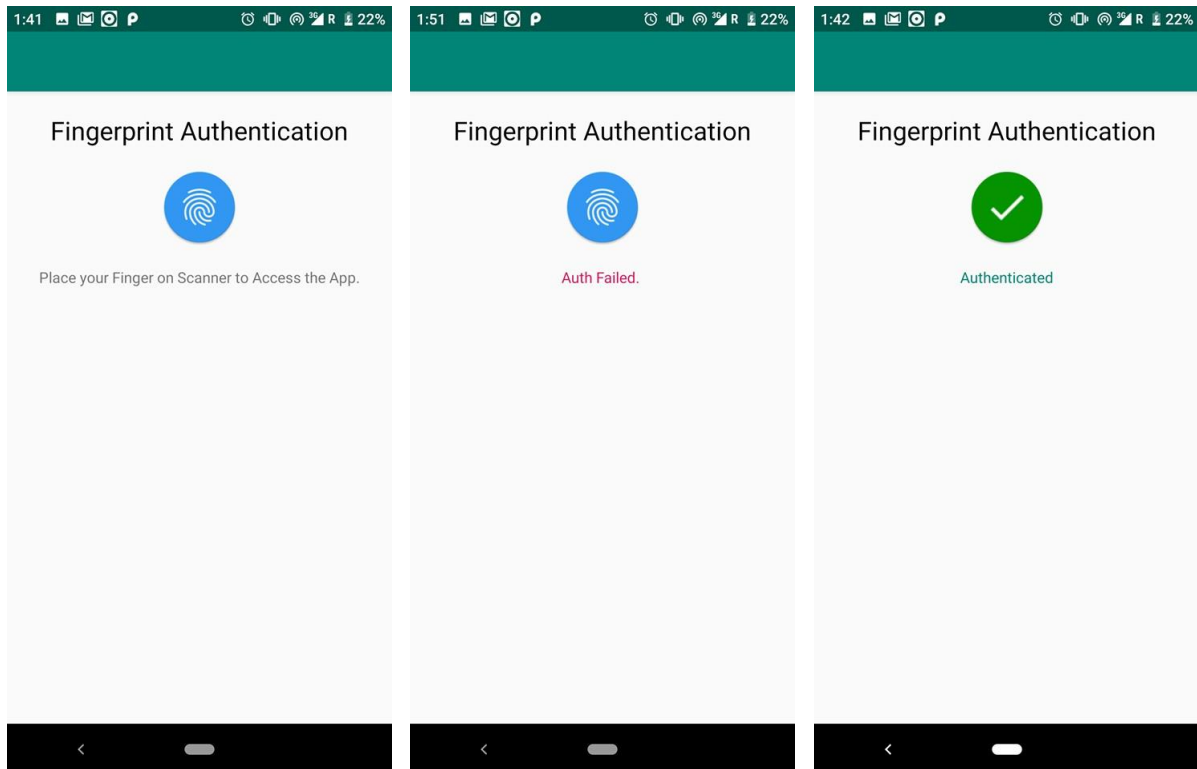
| 3:18 | 3:19 | 3:19 |
|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Add Data | Add Data | Add Data |
|  |  |  |
| Secure Three Factor Authentication by Charizma Gupta, Adrijeet Deb & Rohith Pillai | Secure Three Factor Authentication by Charizma Gupta, Adrijeet Deb & Rohith Pillai | Secure Three Factor Authentication by Charizma Gupta, Adrijeet Deb & Rohith Pillai |
| Data Identifier | VTOP | VTOP |
| Enter Username | 17BCI0157 | 17BCI0157 |
| Password | | |
| ADD DATA RETRIEVE DATA | ADD DATA RETRIEVE DATA | ADD DATA RETRIEVE DATA data inserted successfully |

Data Storage Database Table

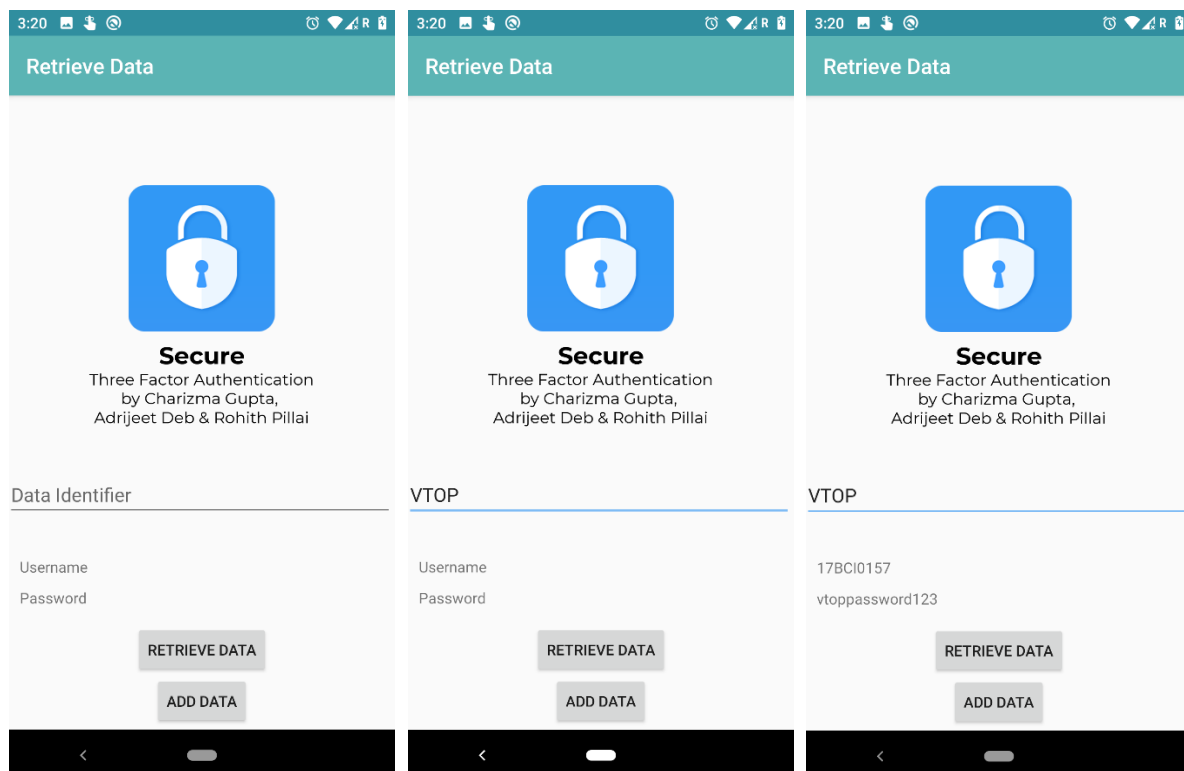


Retrieving Data (Biometric)

Biometric Authentication



Retrieval



Vulnerability analysis [Potential Disasters]

Since our application focuses on the storage of highly sensitive data, any breach of the database can lead to massive leaks. Any user whose details get stolen would normally lose a lot of their private info as the malicious user could login and view all their passwords.

SQL Injection

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web application vulnerabilities. The OWASP organization (Open Web Application Security Project) lists injections in their OWASP Top 10 2017 document as the number one threat to web application security.

Our app uses a NoSQL database and hence is not vulnerable to these.

Session Hijacking

The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token.

Because http communication uses many different TCP connections, the web server needs a method to recognize every user's connections. The most useful method depends on a token that the Web Server sends to the client browser after a successful client authentication. A session token is normally composed of a string of variable width and it could be used in different ways, like in the URL, in the header of the http requisition as a cookie, in other parts of the header of the http request, or yet in the body of the http requisition.

The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server.

However, our app has many checks even if you do infiltrate a session. You must know pin and have the correct biometric to progress.

Brute Force Attack

A brute-force attack consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly. The attacker systematically checks all possible passwords and passphrases until the correct one is found.

Our passwords have a check so that they're always 8 characters or more which makes brute force very hard to use to crack the passwords.

Information Theft

Information theft can happen through Phishing, Pharming, Malicious software, Unsecure app design, Weak passwords and Discarded computers and mobile devices.

No matter the method, our app data is very well protected. All inputted data is stored with military grade AES encryption. Passwords and pins are duly hashed and no data can be retrieved without biometric authentication.

Disaster Prevention

To counter the database breach problem, we're storing only the hashes [from the `.hashCode()` function] of each user's password in our Login table. When a user enters their details, the password is hashed and that is then compared with the table (filled with hash codes) to see whether the password matches. This works on the Deterministic principle; the same message must always produce the same hash. So, storing the hash for the password is just a more objectively secure way of storing the password. We're also encrypting all the passwords stored in each user's password list using AES encryption as to counter a breach into that portion of the database.

NoSQL Database

We used a NoSQL database called firebase for our project. It eliminates the possibility of an SQL injection attack.

Firebase is a Backend-as-a-Service — BaaS — that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform. Firebase frees developers to focus crafting fantastic user experiences. You don't need to manage servers. You don't need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you'll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can't be everything to everybody. But it gets pretty close. Firebase Storage has its own system of security rules to protect your GCloud bucket from the masses, while granting detailed write privileges to your authenticated clients.

OTP

This the 2nd factor of our application. OTP is generated and sent to user to verify their identity. Once the identity is verified, we move on to the last step of our authentication. An OTP is more secure than a static password, especially a user-created password, which can be weak and/or reused across multiple accounts. OTPs may replace authentication login information or may be used in addition to it in order to add another layer of security.

Code

```
btnGenerateOTP.setOnClickListener((v) -> {  
    phoneNumber=etPhoneNumber.getText().toString();  
  
    PhoneAuthProvider.getInstance().verifyPhoneNumber(  
        phoneNumber,                // Phone number to verify  
        60,                          // Timeout duration  
        TimeUnit.SECONDS,            // Unit of timeout  
        activity, MainActivity.this, // Activity (for callback binding)  
        mCallback);                 // OnVerificationStateChangedCallbacks  
});
```

```

auth = FirebaseAuth.getInstance();
mCallback = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

    @Override
    public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential) {
        Toast.makeText( context: MainActivity.this, text: "verification completed",Toast.LENGTH_SHORT).show();
        SigninWithPhone(phoneAuthCredential);
    }

    @Override
    public void onVerificationFailed(FirebaseException e) {
        Toast.makeText( context: MainActivity.this, text: "verification failed",Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onCodeSent(String s, PhoneAuthProvider.ForceResendingToken forceResendingToken) {
        super.onCodeSent(s, forceResendingToken);
        verificationCode = s;
        Toast.makeText( context: MainActivity.this, text: "Code sent",Toast.LENGTH_SHORT).show();
    }

};

```

```

private void SigninWithPhone(PhoneAuthCredential credential) {
    auth.signInWithCredential(credential)
        .addOnCompleteListener((task) -> {
            if (task.isSuccessful()) {
                startActivity(new Intent( packageContext: MainActivity.this, LoginActivity.class));
                finish();
            } else {
                Toast.makeText( context: MainActivity.this, text: "Incorrect OTP",Toast.LENGTH_SHORT).show();
            }
        });
}

```

Hash

Hashing is the transformation of a string of characters into a usually shorter fixed-length value or key that represents the original string. For security reasons, we store passwords in hashed form. This guards against the possibility that someone who gains unauthorized access to the database can retrieve the passwords of every user in the system. Hashing performs a one-way transformation on a password, turning the password into another String, called the hashed password. “One-way” means that it is practically impossible to go the other way - to turn the hashed password back into the original password. There are several mathematically complex hashing algorithms that fulfil these needs. By default, the Personalization module uses the MD5 algorithm to perform a one-way hash of the password value and to store it in hashed form.

The hashed password value is not encrypted before it is stored in the database. When a member attempts to log in, the Personalization module takes the supplied password, performs a similar one-way hash and compares it to the database value. If the passwords match, then login is successful.

The general contract of `hashCode()` states:

Whenever it is invoked on the same object more than once during an execution of a Java application, `hashCode()` must consistently return the same value, provided no information used in equals comparisons on the object is modified. This value needs not remain consistent

from one execution of an application to another execution of the same application

If two objects are equal according to the equals(Object) method, then calling the hashCode() method on each of the two objects must produce the same value

It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results. However, developers should be aware that producing distinct integer results for unequal objects improves the performance of hash tables

Code

```
String u=ed.getText().toString();
int p=ed1.getText().toString().hashCode();
int pi=ed3.getText().toString().hashCode();

Login e=new Login(u, p, pi);
ref.child(e.username).setValue(e);
```

```
// Write a message to the database
String i=ed.getText().toString();
String u=AppUtils.encrypt(ed1.getText().toString());
String p=AppUtils.encrypt(ed2.getText().toString());
int pi= ed3.getText().toString().hashCode();

Entity e=new Entity(i, u, p, pi);
ref.child(e.identifier).setValue(e);
Toast.makeText(context, AddData.this, text: "data inserted successfully",
```

```
if(ed5.getText().toString().hashCode() !=Integer.valueOf(pi)) {
    ed1.setText("incorrect pin");
    ed2.setText("incorrect pin");
} else{
    ed1.setText(AppUtils.decrypt(u));
    ed2.setText(AppUtils.decrypt(p));
}
```

Encryption

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

The features of AES are as follows –

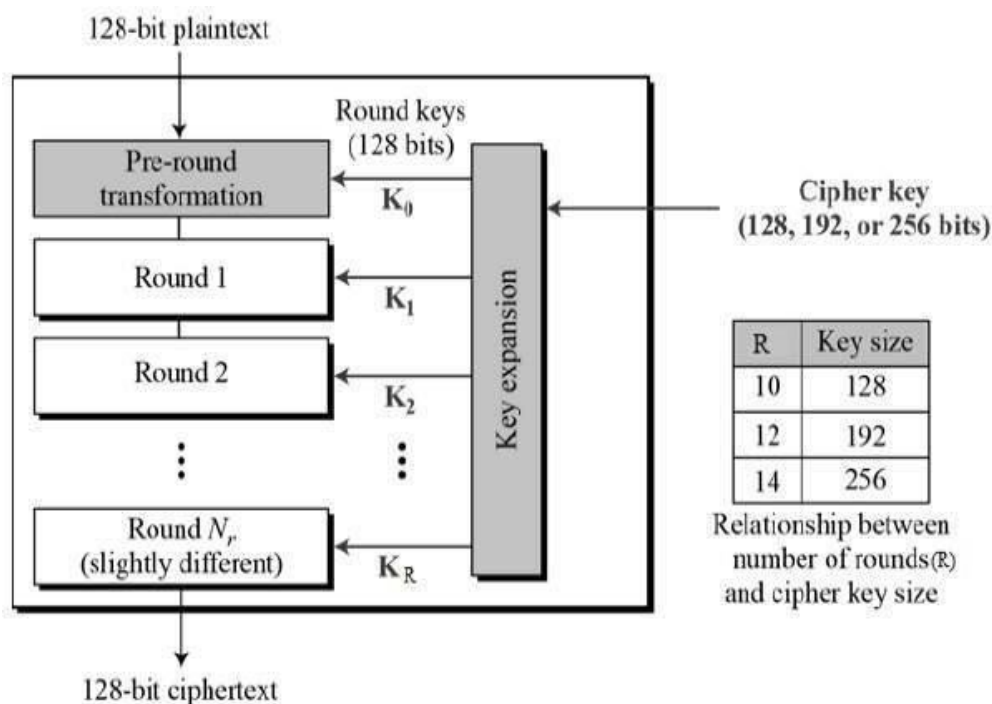
- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys

- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix –

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. The schematic of AES structure is given in the following illustration –



Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted on the right side of row. Shift is carried out as follows –

- First row is not shifted.
- Second row is shifted one (byte) position to the left

- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.

The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

AddRoundKey

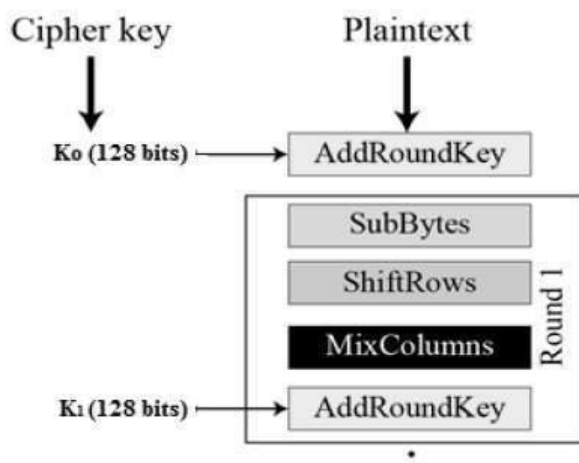
The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order –

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms need to be separately implemented, although they are very closely related.



Code

```
AppUtils.java × RetrieveData.java × activity_retrieve_data.xml × AddData.java × activity_main.xml × Entity.java ×
28 }
29
30 @
31 static String encrypt(String plaintext) {
32     byte[] salt = generateSalt();
33     SecretKey key = deriveKey(salt);
34
35     try {
36         Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
37         byte[] iv = generateIv(cipher.getBlockSize());
38         IvParameterSpec ivParams = new IvParameterSpec(iv);
39         cipher.init(Cipher.ENCRYPT_MODE, key, ivParams);
40         byte[] cipherText = cipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));
41
42         if(salt != null) {
43             return String.format("%s%s%s",
44                 toBase64(salt),
45                 DELIMITER,
46                 toBase64(iv),
47                 DELIMITER,
48                 toBase64(cipherText));
49         }
50
51         return String.format("%s%s",
52             toBase64(iv),
53             DELIMITER,
54             toBase64(cipherText));
55     } catch (GeneralSecurityException e) {
56         throw new RuntimeException(e);
57     }
58 }
```

```
AppUtils.java × RetrieveData.java × activity_retrieve_data.xml × AddData.java × activity_main.xml × Entity.java ×
58
59 @
60 static String decrypt(String ciphertext) {
61     String[] fields = ciphertext.split(DELIMITER);
62     if(fields.length != 3) {
63         throw new IllegalArgumentException("Invalid encrypted text format");
64     }
65     byte[] salt = fromBase64(fields[0]);
66     byte[] iv = fromBase64(fields[1]);
67     byte[] cipherBytes = fromBase64(fields[2]);
68     SecretKey key = deriveKey(salt);
69
70     try {
71         Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
72         IvParameterSpec ivParams = new IvParameterSpec(iv);
73         cipher.init(Cipher.DECRYPT_MODE, key, ivParams);
74         byte[] plaintext = cipher.doFinal(cipherBytes);
75         return new String(plaintext, StandardCharsets.UTF_8);
76     } catch (GeneralSecurityException e) {
77         throw new RuntimeException(e);
78     }
79
80 private static byte[] generateSalt() {
81     byte[] b = new byte[PKCS5_SALT_LENGTH];
82     random.nextBytes(b);
83     return b;
84 }
85 }
```

Biometric

Biometrics are biological measurements — or physical characteristics — that can be used to identify individuals. Fingerprint mapping, facial recognition, and retina scans are all forms of biometric technology, but these are just the most recognized options.

Because physical characteristics are relatively fixed and individualized — even in the case of twins — they are being used to replace or at least augment password systems for computers, phones, and restricted access rooms and buildings.

Biometrics scanners are becoming increasingly sophisticated but yet examples of fingerprint

cloning are everywhere. One example from the Black Hat cybersecurity conference demonstrated that a fingerprint can be cloned reliably in about 40 minutes with \$10 worth of material, simply by making a fingerprint impression in molding plastic or candle wax.

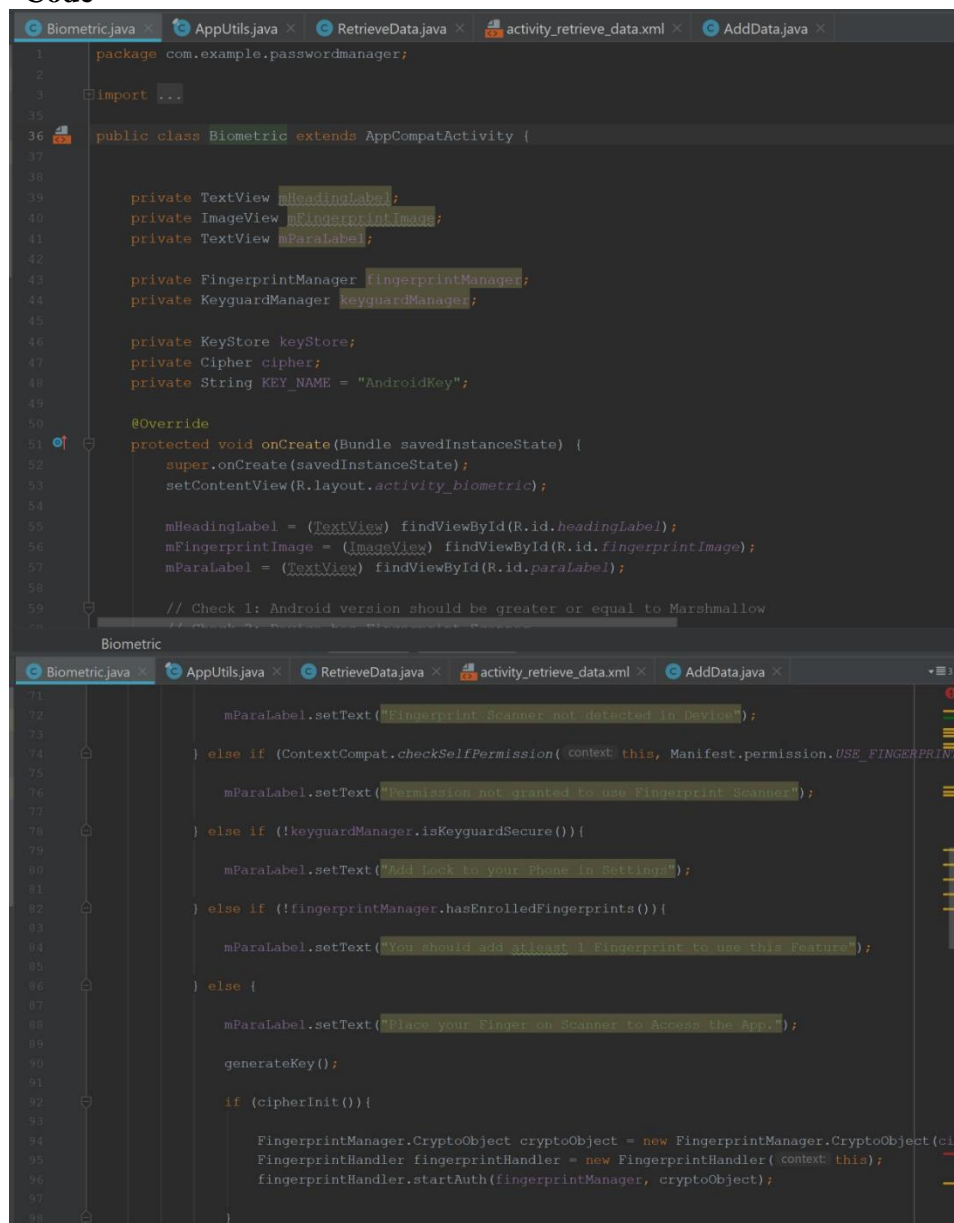
Germany's Chaos Computer Club spoofed the iPhone's TouchID fingerprint reader within two days of its release. The group simply photographed a fingerprint on a glass surface and used it to unlock the iPhone 5s.

Biometric authentication is convenient, but privacy advocates fear that biometric security erodes personal privacy. The concern is that personal data could be collected easily and without consent. A more immediate problem is that databases of personal information are targets for hackers. For example, when the U.S. Office of Personnel Management was hacked in 2015, cybercriminals made off with the fingerprints of 5.6 million government employees, leaving them vulnerable to identity theft.

Storing biometric data on a device – like the iPhone's TouchID or Face ID – is considered safer than storing it with a service provider, even when the data is encrypted.

Our app uses the mobile stored finger print for higher security. Fingerprint must be given before any password is retrieved.

Code



```
1 package com.example.passwordmanager;
2
3 import ...
4
5
6 public class Biometric extends AppCompatActivity {
7
8
9     private TextView mHeadingLabel;
10    private ImageView mFingerprintImage;
11    private TextView mParaLabel;
12
13    private FingerprintManager fingerprintManager;
14    private KeyguardManager keyguardManager;
15
16    private KeyStore keyStore;
17    private Cipher cipher;
18    private String KEY_NAME = "AndroidKey";
19
20    @Override
21    protected void onCreate(Bundle savedInstanceState) {
22        super.onCreate(savedInstanceState);
23        setContentView(R.layout.activity_biometric);
24
25        mHeadingLabel = (TextView) findViewById(R.id.headingLabel);
26        mFingerprintImage = (ImageView) findViewById(R.id.fingerprintImage);
27        mParaLabel = (TextView) findViewById(R.id.paraLabel);
28
29        // Check 1: Android version should be greater or equal to Marshmallow
30
31
32        mParaLabel.setText("Fingerprint Scanner not detected in Device");
33    } else if (ContextCompat.checkSelfPermission(context, this, Manifest.permission.USE_FINGERPRINT)
34        mParaLabel.setText("Permission not granted to use Fingerprint Scanner");
35    } else if (!keyguardManager.isKeyguardSecure()) {
36        mParaLabel.setText("Add lock to your Phone in Settings");
37    } else if (!fingerprintManager.hasEnrolledFingerprints()) {
38        mParaLabel.setText("You should add atleast 1 Fingerprint to use this Feature");
39    } else {
40        mParaLabel.setText("Place your Finger on Scanner to Access the App.");
41        generateKey();
42        if (cipherInit()) {
43            FingerprintManager.CryptoObject cryptoObject = new FingerprintManager.CryptoObject(cipher);
44            FingerprintHandler fingerprintHandler = new FingerprintHandler(context, this);
45            fingerprintHandler.startAuth(fingerprintManager, cryptoObject);
46        }
47    }
48 }
```

```

10 import android.widget.ImageView;
11 import android.widget.TextView;
12 import androidx.core.content.ContextCompat;
13
14 @TargetApi(Build.VERSION_CODES.M)
15 public class FingerprintHandler extends FingerprintManager.AuthenticationCallback {
16
17     private Context context;
18
19     public FingerprintHandler(Context context) {
20
21         this.context = context;
22     }
23
24     @Override
25     public void startAuth(FingerprintManager fingerprintManager, FingerprintManager.CryptoObject cryptoObject) {
26
27         CancellationSignal cancellationSignal = new CancellationSignal();
28         fingerprintManager.authenticate(cryptoObject, cancellationSignal, flags: 0, callback: this, handler: null);
29     }
30
31     @Override
32     public void onAuthenticationError(int errorCode, CharSequence errString) {
33
34         this.update( "There was an Auth Error. " + errString, false);
35     }
36
37     @Override
38     public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
39
40         this.update( "Authenticated", true);
41         Biometric b = new Biometric();
42         context.startActivity(new Intent(context, RetrieveData.class));
43     }
44
45     private void update(String s, boolean b) {
46
47         TextView paraLabel = ((Activity)context).findViewById(R.id.paraLabel);
48         ImageView imageView = ((Activity)context).findViewById(R.id.fingerprintImage);
49
50         paraLabel.setText(s);
51
52         if(b == false) {
53
54             paraLabel.setTextColor(ContextCompat.getColor(context, R.color.colorAccent));
55
56         } else {
57
58             paraLabel.setTextColor(ContextCompat.getColor(context, R.color.colorPrimary));
59             imageView.setImageResource(R.mipmap.action_done);
60         }
61     }
62
63 }

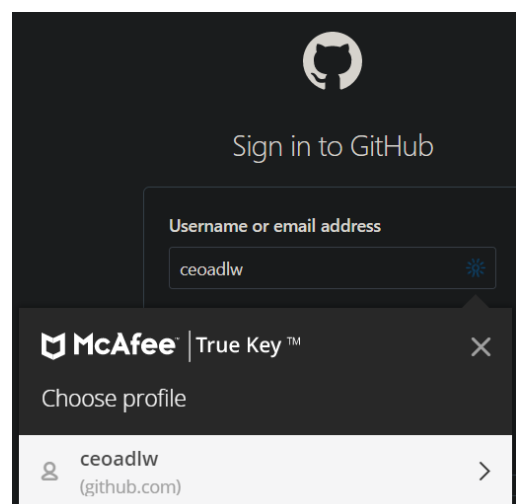
```

Future Works

Passwords are mostly used for logging into websites. As such this can be used to create a browser extension that will enable the user to log into websites seamlessly without searching for his password in the password manager.

As a proof of concept, I have attached this screenshot:

Along with that we can also create a pseudo random password generator for the user to generate new and strong passwords. It will be supported up to 64 bits and allow combinations of special characters, numbers, lower and upper characters. This will help users move away from using the same password twice. The randomness of the passwords can be based on a number of factors. It can be done on the basis of the weather in combination with memory being used in real time of the user's system. This will allow randomness to a great extent as the weather in the same



town may be same but the memory used in user's system will be different always.

This can also be integrated for login into apps from the Microsoft Store by using the Windows Hello API. This will allow apps within the Windows ecosystem to be more secure and less vulnerable.

Future enhancements of this project could also include the database being stored on a remote server. Hackers are a very real threat nowadays. With new technologies arriving on the scene every other day, hackers get lots of opportunities to exploit vulnerabilities. This makes server security a high priority.

Having a self-owned server that gets timely updates with a team monitoring your server 24/7 makes the chances of a hacker targeting your server is next to zero.

UI enhancements could be made for the user to be able to see a table of what they've stored instead of having to enter identifier as that can be forgotten over time.

Conclusion – Overall security of the application.

We have made a NoSQL based password managing android application which is immune to sql injection attacks, hashes usernames and passwords that are stored making the stealing of accounts through a data breach nearly impossible, allows for the registration of new users and the creation of PINs unique to each user.

The app is also extremely scalable as it uses a NoSQL database (Firebase) which allows for quick horizontal expansion (more users), effectively making our app ready to go out to the public provided we get a secure remote server to store our databases in.

Data can only be accessed through an identifier and PIN that only the authorized user would know and even if they managed to bypass the application and crack into the database, infiltrators will be greeted by data protected through military grade AES encryption.

References

- [1] Kennedy, W., & Olmsted, A. (2017, December). Three factor authentication. In 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST) (pp. 212-213). IEEE.
- [2] Hang, Q., Guo, H., Zou, X., & Hou, Q. (2012, August). Design and implementation of mobile access system based on multi-factor authentication. In 2012 International Conference on Computer Science and Information Processing (CSIP) (pp. 131-134). IEEE.
- [3] Yang, S., & Meng, J. (2018, October). Research on Multi-factor Bidirectional Dynamic Identification Based on SMS. In 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) (pp. 1578-1582). IEEE.
- [4] Katha Chanda, "Password Security: An Analysis of Password Strengths and Vulnerabilities", *International Journal of Computer Network and Information Security (IJCNIS)*, Vol.8, No.7, pp.23-30, 2016. DOI: 10.5815/ijcnis.2016.07.04
- [5] 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), CHENNAI, India, 2019, pp. 1-7. doi: 10.1109/ICIICT1.2019.8741459
- [6] A. Bissada and A. Olmsted, "Mobile multi-factor authentication," 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), Cambridge, 2017, pp. 210-211.
- [7] Nath, Asoke & Mondal, Tanushree. (2016). Issues and Challenges in Two Factor Authentication Algorithms. *International Journal of Latest Trends in Engineering and Technology(IJLTET)*. 6. 318-327.
- [8] Lu, C. C., & Tseng, S. Y. (2002, July). Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter. In *Proceedings IEEE International Conference on Application-Specific Systems, Architectures, and Processors* (pp. 277-285). IEEE.
- [9] Osvik, D. A., Bos, J. W., Stefan, D., & Canright, D. (2010, February). Fast software AES encryption. In *International Workshop on Fast Software Encryption* (pp. 75-93). Springer, Berlin, Heidelberg.
- [10] Aishwariya, G., Kokilapriya, S., Adhithya, S., & Selvarani, A. G. (2017). Fingerprint Recognition for Android Application Data Retrieval. *International Journal of Scientific Research in Science and Technology*, 3, 253-256.
- [11] Kupershtein, L. M., Voytovych, O. P., Kaplun, V. A., & Prokopchuk, S. O. (2018). The database- oriented approach to files protection in android operation system. *Вісник Хмельницького національного університету. Технічні науки*, (1), 18-22.