Innovations in Computing Technology and Applications
Volume 2 | Year: 2017 | ISBN: 978-967-0194-95-0

Secure Authentication for Internet environment using Biometric and One-Time Password

Nabil Akid bin Abdul Mutalib[1] and Mazleena Salleh[2]

Faculty of Computing, University of Technology Malaysia, University of Technology Malaysia, UTM, 81310 Skudai, Johor, Malaysia

[1]beylakid@gmail.com, [2]mazleena@utm.my

Abstract— Authentication is important especially securing accounts in the Internet environment. Sensitive data such as bank accounts, Identification number, and passwords are being held online within every user. Accounts can be easily stolen because accounts are getting hacked, breached, and attacked unnoticed with the help of greater hardware speed. This happens because one layer of authentication is not secure enough to hold accounts on long term. By adding two-factor authentication method increases the protection of every account.

1. Introduction

Computing technology gave a great impact after decades of usage. The improvement helps people to develop better computing software for easier and better usage. After decades and great improvements towards the computer, the Internet came to existence. The Internet started off to help all computer users to communicate with each other. The Internet improves everyday with the help of greater computing system. As the amount of the Internet user improves, companies started taking the opportunity by using the Internet as a medium for users to use their services. Thus millions of accounts exist within the Internet that is stored inside the company's database. Most of the accounts that exist on the Internet are being protected by only a single factor. The single factor authentication method that is used by most companies is the Login ID and Password as shown in Figure 1.

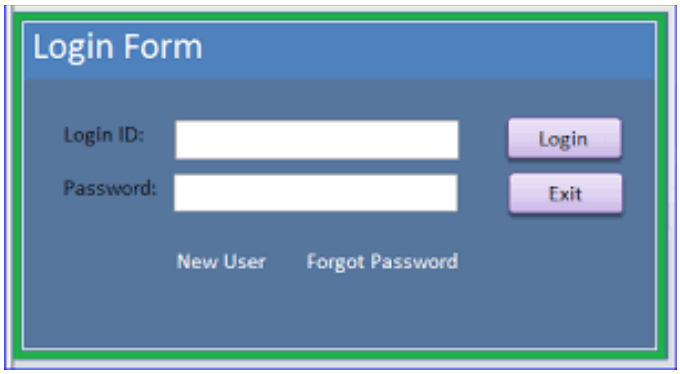


Figure 1. The Login ID and password authentication is the most used method.

These accounts may contain sensitive personal information such as bank account information, credit card number, and personal home address. One layer of protection is not secured enough to protect all accounts because attackers tend to implant viruses or keyloggers into many computers to track the user's Login ID and Password for personal gain. Thus by adding more protection layer thickens the wall. Using different authentication methods ensures such as combining both biometric and one time password. Biometric feature will be using the user characteristic such as the fingerprint while the OTP will be using generated passwords that will randomly change under a period of time. Attackers will

have a hard time trying to breach other authentication methods because of the differences between authenticating algorithms.

2. Method

2.1. Biometric Fingerprint

Fingerprint has been unique throughout life because no two fingerprints have ever been found to be the same. This applies to identical twins as well. It has been used over 140 years for sealing purposes. The identification of a fingerprint involves comparing the furrows on the fingertips and the pattern of ridges. The minutiae points (a characteristic of the ridge that occurs whenever a ridge splits into two, or ends) of a specimen print with a database of prints on file are also part of the identification as well [1]. The fingerprint optical scanner starts off by scanning the image of the finger. Then it determines the pattern of ridges and valleys in the image matches with the pattern of ridges and valleys in pre-scanned images. No image of a fingerprint has ever been saved, only a series of numbers (a binary code), which will be used for verification. The algorithm that is used can never be reversed back into the original image, thus no one can ever duplicate the fingerprint [2].

2.2. One-Time Password (OTP)

An OTP is an alphanumeric string of characters or automatically generated numbers that authenticates the user for a single session or transaction. A one-time password is not the same as static password. It is more secure than user-created password, which is typically weak. One-time password is mostly used as an additional authentication by adding another layer of security next to the login information. The generated number changes every 30 or 60 seconds, depending on how it has been configured [3].

2.3. Related Works

Computer-related companies such as Google, Microsoft, and Authy created authenticator for all users for safety purposes. Google authenticator generates a code as an additional password for users. It even works without any connectivity, even on airplane mode. Microsoft Azure authenticator lets the user choose to verify their status by providing two methods that are one-time password or fingerprint. Users are allowed verify themselves with either method. Authy 2-Factor authenticator uses one-time password, SMS one-time password, and Yes or No Approval. Authy's authenticator can be used with many applications as well. All three authenticator provides one-time password method. Azure adds biometric as an alternative method. Authy adds SMS one-time password or Yes or No approval as an alternative method. Table 1 shows the comparison between related works.

Table 1. Comparison between related works

Existing Applications	**Google Authenticator**	**Azure Authenticator**	**Authy2-factor Authenticator**
Characteristic	Separate application for second authentication	Separate application for second authentication	Separate application for second authentication
Methodology	• HMAC-based One-time password • Time-based One-time password	• HMAC-based One-time password • Time-based One-time password • Biometric Touch ID	• Time-based One-time password • SMS-based One-time Password • Yes or No approval

Existing Applications	Google Authenticator	Azure Authenticator	Authy2-factor Authenticator
Compatibility	• Android • iOS • BlackBerry OS	• Android • iOS • Windows Phone	• Android • iOS • BlackBerry OS • Mac OSX • Windows • Linux
Advantages	Wide range of application support	Biometric feature is implemented	Wide range of application support
Disadvantages	Vulnerable against phishing	Redundant administrators will increase the attack surface	Vulnerable against phishing

All three of the mobile application uses the same method that is as the second application authenticator and One-time Password. Only Azure Authenticator provides Biometric Touch ID that is compatible with Apple's iOS. Authy 2-factor Authentication provides one of the easiest methods in the application that are 'Yes' or 'No' approval method.

In terms of compatibility, Authy provides the widest range of support such as Android, iOS, Blackberry OS, Mac OSX, Windows, and Linux. Google and Authy support a wide range of application. This can be easy for users if they want to use the desired authenticator. All three of the applications have their own weaknesses as well.

Google Authenticator is vulnerable towards phishing attacks. An attack on Azure can be increased if there are redundant administrators. Authy's weakness is the One-time Password time limit for every request made by the user.

2.4. Mobile Devices With Fingerprint Reader

Latest mobile devices have been equipped with integrated fingerprint reader especially the company's flagship models. Apple iPhone 5s was the first iPhone to be equipped with a fingerprint reader. Apple named the fingerprint reader as Touch ID. Samsung first mobile device with a fingerprint reader was Samsung Galaxy S5 but it was filled with major flaws. An improved version was implemented onto Samsung Galaxy S6. Google managed to implement fingerprint readers onto their mobile devices as well such as Google Nexus 6P. Google named their fingerprint reader as Nexus Imprint. Table 2 shows the comparison between existing mobile devices.

Table 2. Comparison between mobile devices

Mobile Devices	Apple iPhone 5s	Samsung Galaxy S6	Google Nexus 6P
Display	4.0"	5.1"	5.7"
Battery Life	Up to 10 hours	Up to 17 hours	Up to 23 hours
Operating System	Apple iOS	Google Android	Google Android
Fingerprint scanner	Touch ID	Fingerprint Reader	Nexus Imprint
Features	• 1.2MP front camera • 8MP back camera • Microphone	• 5MP front camera • 16MP back camera • Microphone	• 8MP front camera • 12MP back camera • Microphone
Built-in GPS	Yes	Yes	Yes

By comparing all of the biometric-enabled mobile devices, Apple iPhone 5s is a better preference because it was the first mobile device to use fingerprint scanner equipped with advanced technology. Though it is built with an advanced technology, Samsung Galaxy S6 has a better battery life and back camera out of all three. Google Nexus 6P may have the highest battery life but due to the size of it for being too big. Samsung Galaxy S6 is already enough for the biometric purpose such as the fingerprint and it uses the Android environment.

2.5. Existing Software Technology

There are many ways to develop the application. Both Phonegap and Appcelrator Titanium use HTML, CSS, and Javascript as programming language. Android Studio uses Java as the only programming language. Android Studio is mainly used for Android mobile devices because of it has a native relation with it. Table 3 shows the comparison between software technologies

Table 3. Comparison between software technologies

Software	PhoneGap	Appcelerator Titanium	Android Studio
Programming Language	HTML, CSS, Javascript	HTML, CSS, Javascript	Java
Advantages	• Open Source and Free • Give opportunity to all developers • Same result as native apps • A lot of plugins	• Good • Javascript can be used on multiple platform • Provide value-adds to 3rd party component	• Faster and more powerful • Better outlook for the future
Disadvantages	• No guarantee success • Risk of incompatible plugins • Bad performance	• Manage target platform manually • Required skill on proprietary technology	• Beta version • Overwhelming for beginners

In the result, a developer will need to identify a suitable platform to create a mobile application. For the PhoneGap work, a developer will have to learn on language compatibility because of the language that required. On the other hand, Appcelerator Titanium is better when building a mobile application. Android Studio is suitable for developer that has basic knowledge on Android.

3. Result

6.5 System Architecture Design

There are two types of system architecture for this method. One for the biometric authentication and the second one would be for one time password generator. The process for biometric authentication starts off by the enrollment of the selected finger. The sample will then be extracted by taking some pattern and ridges that is converted into binary codes to become a template. The template itself will be stored into the fingerprint database. After

registering, if the user decided to use the system, the user will need to place their registered finger onto the fingerprint reader. The sample taken by the fingerprint reader will be compared with the one inside the fingerprint database. If it's valid, the user will be granted to obtain the one time password. If not, then the user will not be granted to obtain the one time password. The process for one-time password starts off by registering the desired account using QR code. Once registered, users can obtain the generated code from the database. The provided code then will be used inside the original account as the second factor authentication. If the number entered is valid, users will be granted full access towards the account. If the number entered is invalid or already expired, users will not be granted any access towards the account. The one-time password system architecture design can be seen in Figure 2.

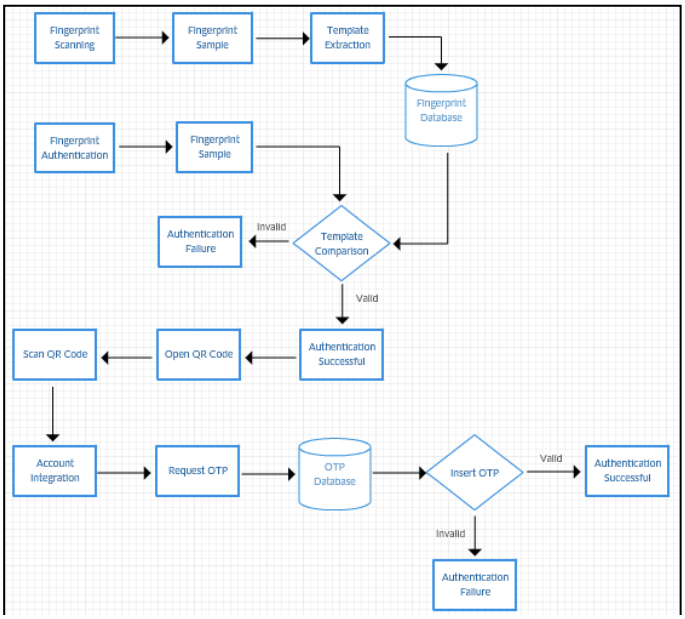
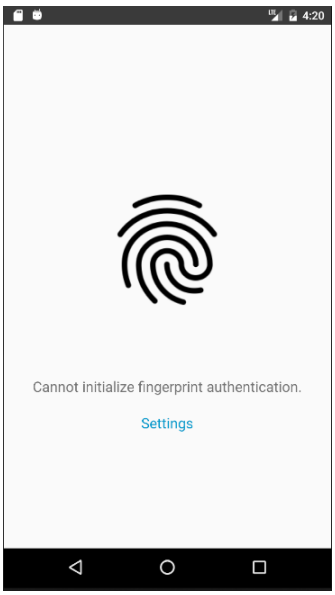


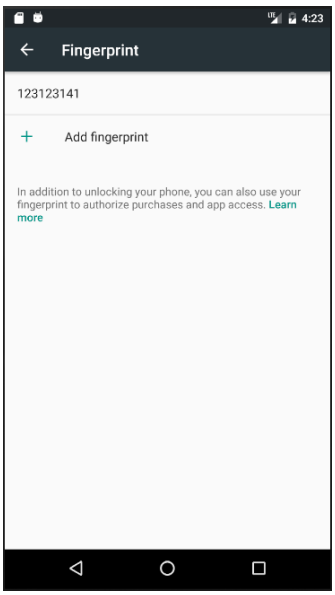
Figure 2. One-time password system architecture design

3.1. User Interface

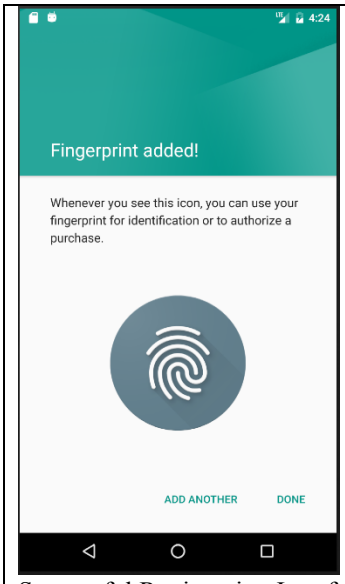
The application's user interface is shown in Figure 4 and the description is shown in Table 4.



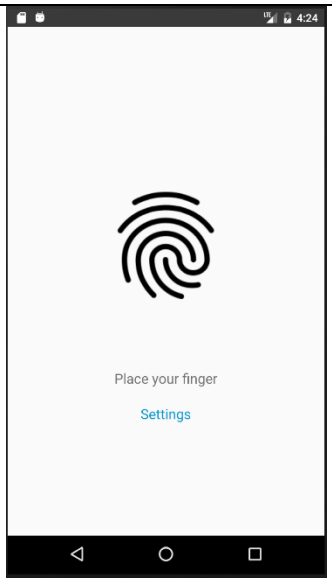
Unregistered Fingerprint Interface



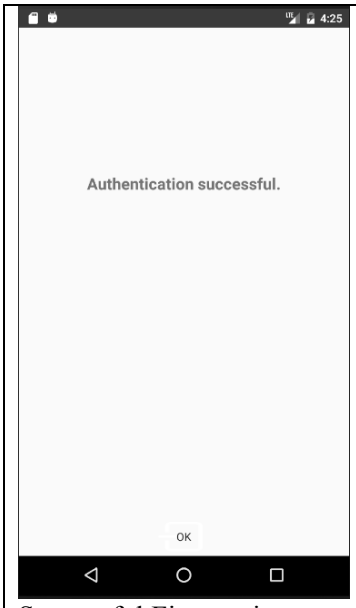
Add Fingerprint Interface



Successful Registration Interface

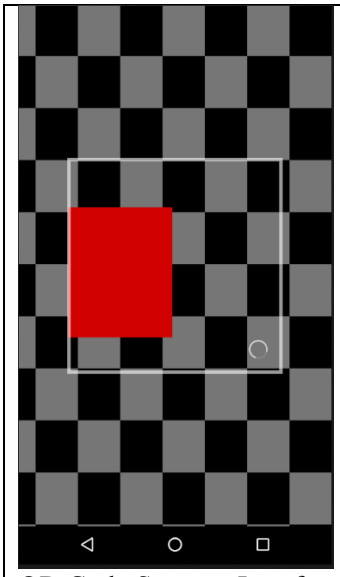


Main Interface For Fingerprint Scanning

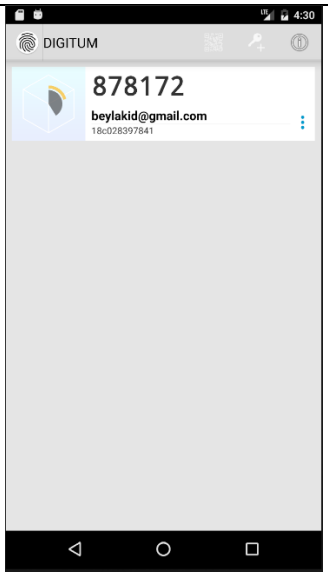


Successful Fingerprint Authentication Interface

One Time Password Interface



QR Code Scanner Interface



Main One-Time Password Interface

Figure 3. User interface and layout

Table 4. Description of the user interface and layout

User Interface	Description
Fingerprint Main Interface	The main page of the unregistered fingerprint authenticator
Add Fingerprint Interface	The native security settings to add fingerprints
Successful Registration Interface	The native security settings for successful registered fingerprints
Main Interface For Fingerprint Scanning	The main page of registered fingerprint authenticator
Successful Fingerprint Authentication Interface	The successful page after fingerprint authentication
One Time Password Main Interface	The main page of the One Time Password Authenticator
QR Code Scanner Interface	The QR Code Scanner for registering accounts
Main One-Time Password Interface	The main page of the One Time Password Authenticator with registered Account

3.2. Code Implementation

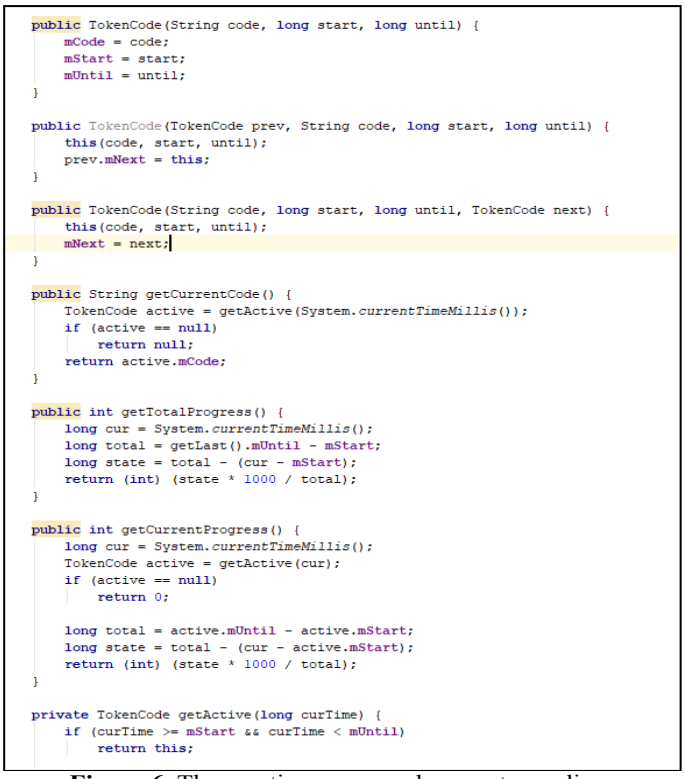
The implementation for authentication key and cipher calculation is for the fingerprint authentication while the implementation for one time password generator is for one time password authentication. All 4 of them are shown in Figure 4, Figure 5, Figure 6, and Figure 7.

```
@TargetApi(23)
private boolean generateKey() {
    mKeyStore = null;
    KeyGenerator keyGenerator;

    //Get the instance of the key store.
    try {
        mKeyStore = KeyStore.getInstance("AndroidKeyStore");
        keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES, "AndroidKeyStore");
    } catch (NoSuchAlgorithmException |
            NoSuchProviderException e) {
        return false;
    } catch (KeyStoreException e) {
        return false;
    }

    //generate key.
    try {
        mKeyStore.load(null);
        keyGenerator.init(new
                KeyGenParameterSpec.Builder(KEY_NAME,
                KeyProperties.PURPOSE_ENCRYPT |
                        KeyProperties.PURPOSE_DECRYPT)
                .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
                .setUserAuthenticationRequired(true)
                .setEncryptionPaddings(
                        KeyProperties.ENCRYPTION_PADDING_PKCS7)
                .build());
        keyGenerator.generateKey();

        return true;
    } catch (NoSuchAlgorithmException
            | InvalidAlgorithmParameterException
            | CertificateException
            | IOException e) {
        return false;
    }
}
```

Figure 4. The authentication key coding

```
@TargetApi(23)
private boolean cipherInit() {
    boolean isKeyGenerated = generateKey();

    if (!isKeyGenerated) {
        mCallback.onAuthFailed(AuthErrorCodes.NON_RECOVERABLE_ERROR, ERROR_FAILED_TO_GENERATE_KEY);
        return false;
    }

    try {
        mCipher = Cipher.getInstance(
                KeyProperties.KEY_ALGORITHM_AES + "/"
                        + KeyProperties.BLOCK_MODE_CBC + "/"
                        + KeyProperties.ENCRYPTION_PADDING_PKCS7);
    } catch (NoSuchAlgorithmException |
             NoSuchPaddingException e) {
        mCallback.onAuthFailed(AuthErrorCodes.NON_RECOVERABLE_ERROR, ERROR_FAILED_TO_GENERATE_KEY);
        return false;
    }

    try {
        mKeyStore.load(null);
        SecretKey key = (SecretKey) mKeyStore.getKey(KEY_NAME, null);
        mCipher.init(Cipher.ENCRYPT_MODE, key);
        return true;
    } catch (KeyPermanentlyInvalidatedException e) {
        mCallback.onAuthFailed(AuthErrorCodes.NON_RECOVERABLE_ERROR, ERROR_FAILED_TO_INIT_CHIPPER);
        return false;
    } catch (KeyStoreException | CertificateException
             | UnrecoverableKeyException | IOException
             | NoSuchAlgorithmException | InvalidKeyException e) {
        mCallback.onAuthFailed(AuthErrorCodes.NON_RECOVERABLE_ERROR, ERROR_FAILED_TO_INIT_CHIPPER);
        return false;
    }
}

@TargetApi(23)
@Nullable
private FingerprintManager.CryptoObject getCryptoObject() {
    return cipherInit() ? new FingerprintManager.CryptoObject(mCipher) : null;
}
```

Figure 5. The cipher key calculation coding



```
public TokenCode(String code, long start, long until) {
    mCode = code;
    mStart = start;
    mUntil = until;
}

public TokenCode(TokenCode prev, String code, long start, long until) {
    this(code, start, until);
    prev.mNext = this;
}

public TokenCode(String code, long start, long until, TokenCode next) {
    this(code, start, until);
    mNext = next;
}

public String getCurrentCode() {
    TokenCode active = getActive(System.currentTimeMillis());
    if (active == null)
        return null;
    return active.mCode;
}

public int getTotalProgress() {
    long cur = System.currentTimeMillis();
    long total = getLast().mUntil - mStart;
    long state = total - (cur - mStart);
    return (int) (state * 1000 / total);
}

public int getCurrentProgress() {
    long cur = System.currentTimeMillis();
    TokenCode active = getActive(cur);
    if (active == null)
        return 0;

    long total = active.mUntil - active.mStart;
    long state = total - (cur - active.mStart);
    return (int) (state * 1000 / total);
}

private TokenCode getActive(long curTime) {
    if (curTime >= mStart && curTime < mUntil)
        return this;
```

Figure 6. The one time password generator coding

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.widget.Button;
import org.digitum.otp.MainActivity;

public class AuthSuccessScreen extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_auth_success_screen);

        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener((v) → {

                startActivity(new Intent(AuthSuccessScreen.this, MainActivity.class));
        });

    }
```

Figure 7. The biometric and one time password integration coding

3.3. Activities Testing

The aim of the application is to provide extra protection towards accounts. Every activity that is provided within the system is then tested. The result of the activities and testing are shown in the tables below. The tables are arranged in order of the activity sequence.

Table 5. Fingerprint results

No.	Test Procedure	Expected Results	Results
1.	Click on 'DIGITUM' icon to launch the application	The main page activity will be displayed	Success
2.	On the Main Page before registering	The interface will display please register fingerprints	Success
3.	Settings button	The button will redirect to native fingerprint settings	Success
4.	Add Fingerprint	Place finger on the Home key	Success
5.	On the Main Page after registering	Redirect to authentication successful page	Success
6.	Ok button	The button will redirect to OTP page	Success
7.	Unregistered fingerprint	"Cannot recognize your fingerprint. Please try again." Message will be displayed	Success
8.	Invalid 5 attempt of fingerprint scanning	"Please wait for 60 seconds before trying again." Message will be displayed	Success

Table 6. OTP results

No.	Test Procedure	Expected Results	Results
1.	On the Main Page before registering	No OTP keys is installed will be displayed	Success

2.	Enable Camera Access	The camera will open for QR Code scanning	Success
3.	QR Code scanning	Capture the QR code for the desired account	Success
4.	On the Main Page after registering	A button will appear to generate the one time password code	Success
5.	More settings button	Edit or Delete button will appear	Success
6.	Edit button	Users are able to change name and emails.	Success
7.	Delete button	Delete account	Success

4. Discussion

Two-Factor Authentication was designed and developed for security purposes. The usage of combining both fingerprint security and the one time password provides a new level of security for users. Finally, this application will be an open source, free to use, and can be operated anywhere.

References

1. iAccess World, Information Sharing Center [online] http://www.iaccessworld.com/downloads/login- register-reset-password-with-security-questions/ [Accessed: May 14, 2017]
2. Bioelectronix, What is Biometrics? [online] http://www.bioelectronix.com/what-is biometrics.html [Accessed: May 14, 2017]
3. I. Wigmore, "One-Time Password {OTP}" [online] http://searchsecurity.techtarget.com/definition/one- time-password-OTP [Accessed: May 14, 2017]