Lab 4: Decryption and encrypted messenger app

Objective:

This lab will keep describing the knowledge of cryptography. Particularly, you will code for the decryption of the message and finish the app AES encryption and decryption of message. Furthermore, you will design the encrypted messenger app by using explicit intent. The lab will introduce you about Android BROADCAST package that allows you to program send and receive message via mobile platforms. Some suggestions about a real messenger app will be investigated.

Outline of the lab 4:

- ✓ Task 1: AES encryption and decryption
- ✓ Task 2: Demo message app and coding Explicit Intent for encrypted app
- ✓ Task 3: Introduction of BROADCAST package
- ✓ Task 4: Suggestions of a real security messenger app

Task 1. AES encryption and decryption message app:

Task: You will make an app that demonstrates how a message is encrypted and decrypted through AES algorithm. Input is the message text and the password you type in. Output is to display the cipher text (when encrypting) and raw text with the right password (when decrypting).

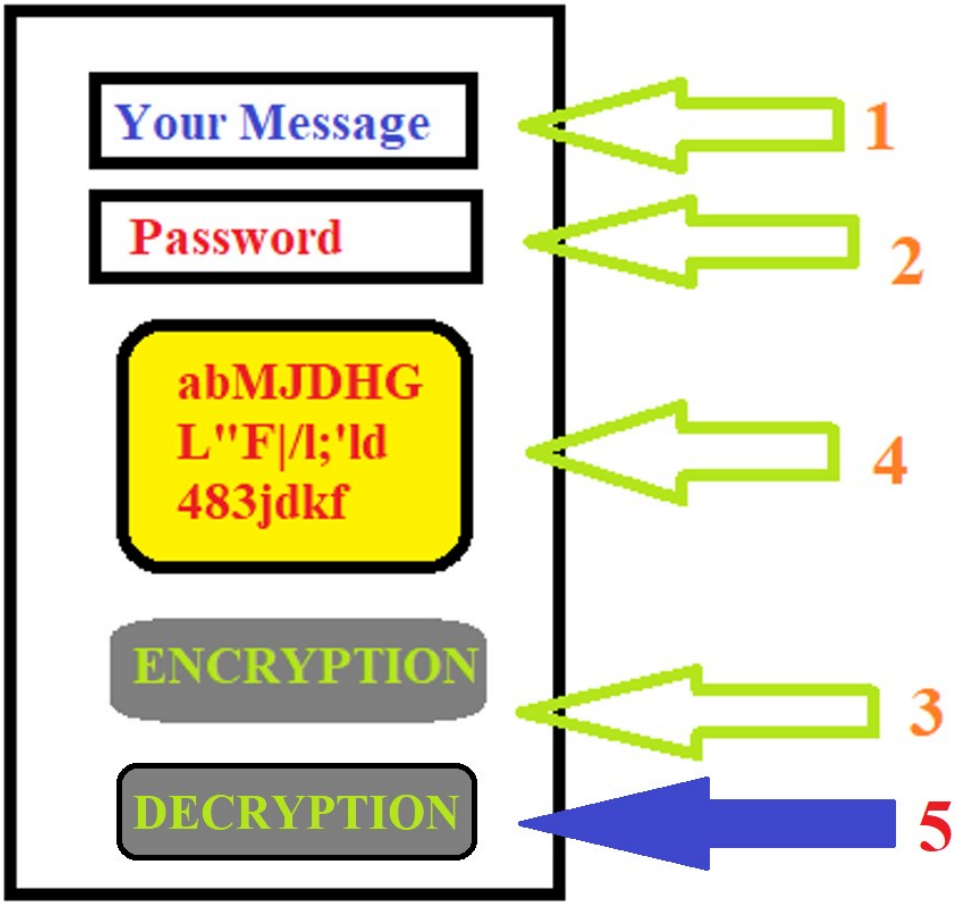
The user interface will have components as follows. We have two edit-texts including the message and the password edit-texts to get the text input. Below the above two edit-texts, we have a text-view that displays the encrypted message (Cipher text) and decrypted message (Raw message). In the text-view, the background colour is yellow, and the text is red. Below the text-view, we have two buttons with labelled “ENCRYPTION” and “DECRYPTION”.

You will code in Android environment for this app. When you click the encryption button, text-view will display the cipher text. Then you type the

password in the password edit-text. When clicking decryption button, if the typed password is right, the text-view will display the raw message. If the password is not right, the screen will toast the message “Wrong Password”.

You will do this app in three phases. Phase 1 is to design the user interface by drawing with any tools (Paint is recommended). Phase 2 is to make the user interface as designed and code for the app. Phase 3 is to run the app and debug the app.

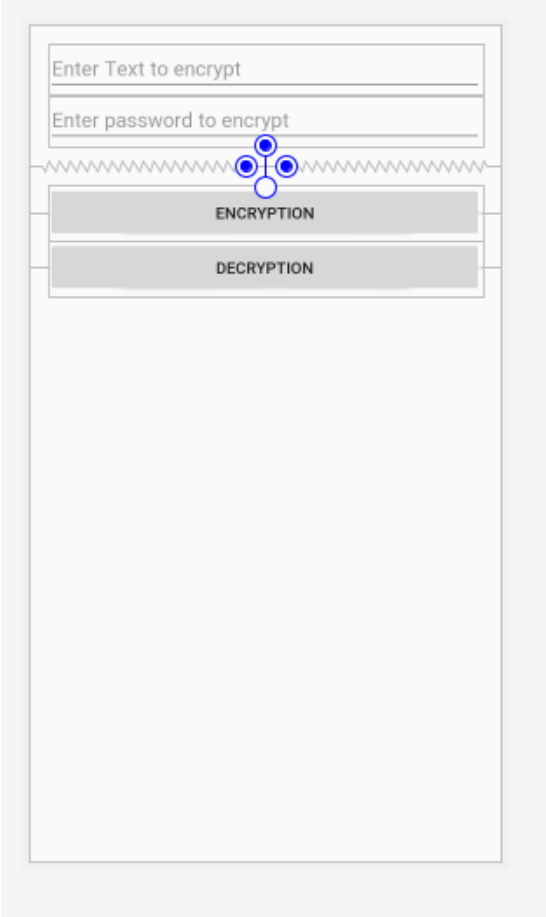
Phase 1: The design of the user interface like this.



Phase 2: This phase includes 2 small tasks making user interface and coding.

First, we can make a new project with the name is “**Enc Dec**”

Then, you will make the user interface. Remember to do the requirements from the task like colouring for the background of text view or labelling for the buttons. The user interface should be like this. The text-view is hard to see because we design in relative-layout mode and design text-view with properties for layout_width and layout_height are “wrap_content”.



Subsequently, you code for the app.

This is the code for xml file activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/inputText"
        android:textColor="#000"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Text to encrypt"
        android:inputType="text" />

    <EditText
        android:id="@+id/inputPass"
        android:textColor="#000"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter password to encrypt"
        android:layout_below="@id/inputText"
        android:inputType="text" />

    <TextView
        android:id="@+id/outputText"
        android:textSize="25sp"
        android:textColor="#F30723"
        android:background="#EAF863"
        android:layout_centerHorizontal="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/inputPass" />

    <Button
        android:id="@+id/encBtn"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@id/outputText"
        android:layout_centerHorizontal="true"
        android:text="@string/encButton"/>

    <Button
        android:id="@+id/decBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/encBtn"
        android:layout_centerHorizontal="true"
        android:text="@string/decButton"/>

</RelativeLayout>
```

Add code for strings.xml

```
<resources>
    <string name="app_name">Enc Dec</string>
    <string name="encButton">Encryption</string>
    <string name="decButton">Decryption</string>
</resources>
```

This is code for Java file MainActivity.java

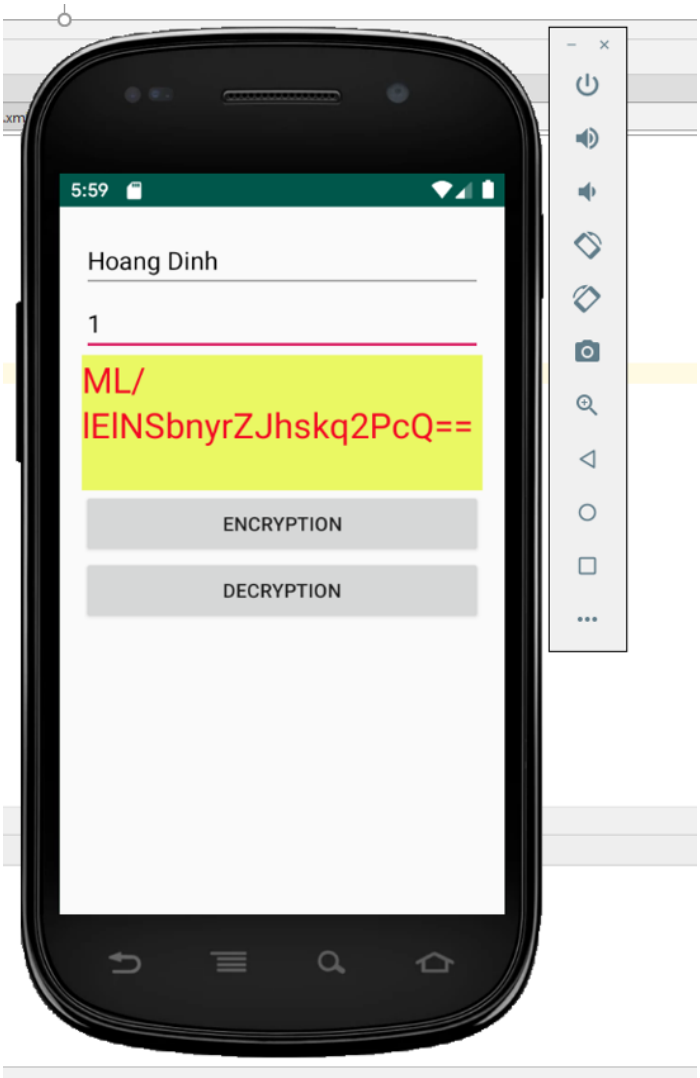
```
package com.example.encdec;

import android.app.Activity;
import android.os.Bundle;
import android.util.Base64;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
```

```
import java.security.MessageDigest;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class MainActivity extends Activity {
    EditText inputTxt, inputPw;
    TextView outputTxt;
    Button encBt, decBt;
    String outputString, encryptedValue, decryptedValue;
    String AES="AES";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        inputTxt = (EditText) findViewById(R.id.inputText);
        inputPw = (EditText) findViewById(R.id.inputPass);
        outputTxt = (TextView) findViewById(R.id.outputText);
        encBt = (Button) findViewById(R.id.encBtn);
        decBt = (Button) findViewById(R.id.decBtn);

        encBt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    outputString =
encrypt(inputTxt.getText().toString(),
inputPw.getText().toString());
                    outputTxt.setText(outputString);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

        decBt.setOnClickListener(new View.OnClickListener() {
            @Override
```

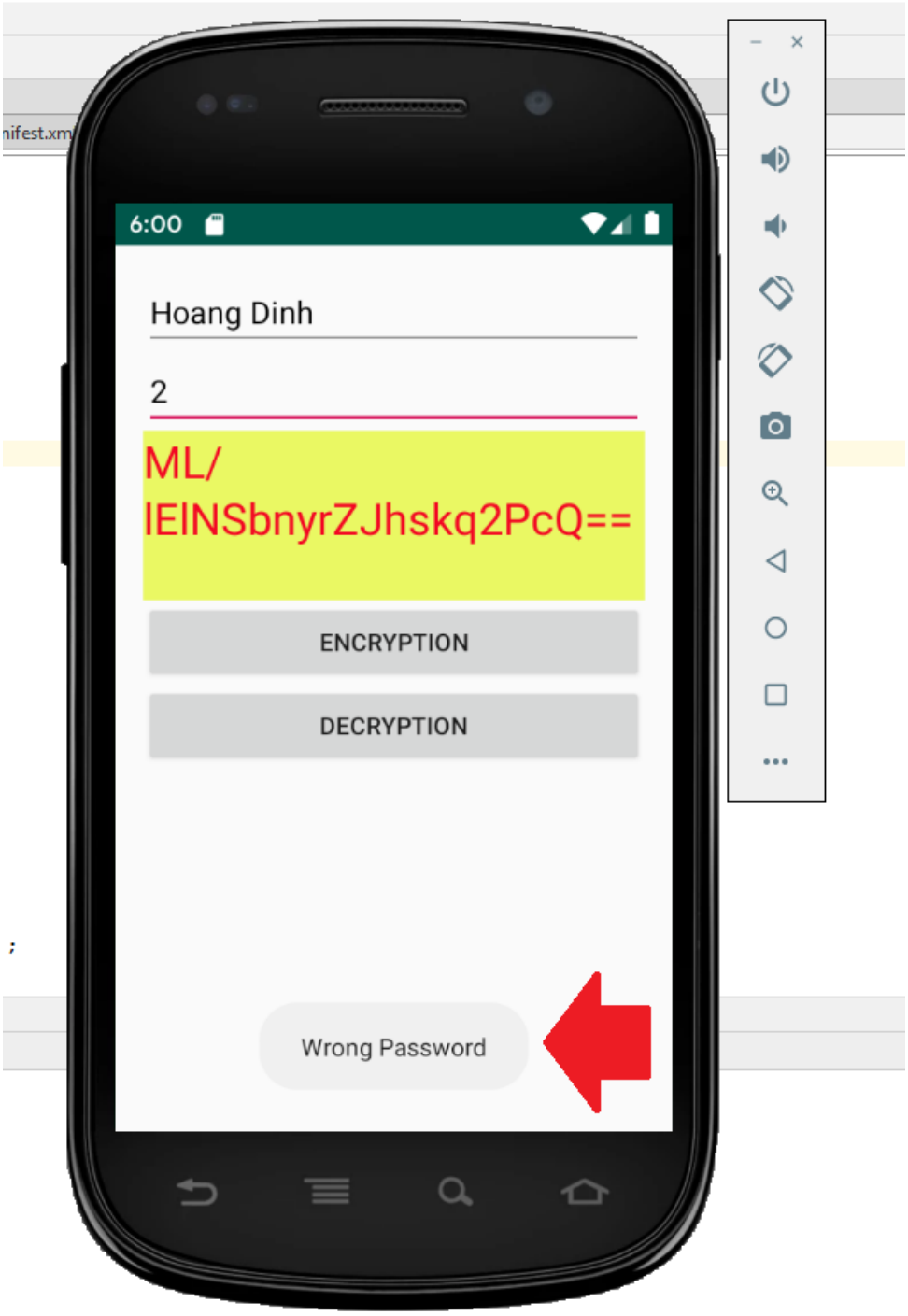
```
            public void onClick(View v) {
                try {
                    outputString = decrypt(outputString,
inputPw.getText().toString());
                } catch (Exception e) {
                    Toast.makeText(MainActivity.this, "Wrong
Password", Toast.LENGTH_LONG).show();
                    e.printStackTrace();
                }
                outputTxt.setText(outputString);
            }
        });
    }
    private String encrypt (String Data, String pass) throws
Exception {
        SecretKeySpec key = generateKey(pass);
        Cipher c = Cipher.getInstance(AES);
        c.init(Cipher.ENCRYPT_MODE, key);
        byte[] encVal = c.doFinal(Data.getBytes());
        encryptedValue = Base64.encodeToString(encVal,
Base64.DEFAULT);
        return encryptedValue;
    }

    private String decrypt (String outputString, String pass)
throws Exception {
        SecretKeySpec key = generateKey(pass);
        Cipher c = Cipher.getInstance(AES);
        c.init(Cipher.DECRYPT_MODE, key);
        byte[] decodedValue = Base64.decode(outputString,
Base64.DEFAULT);
        byte[] decValue = c.doFinal(decodedValue);
        decryptedValue = new String(decValue);
        return decryptedValue;
    }

    private SecretKeySpec generateKey (String pass) throws
```

```
Exception {
        final MessageDigest digest =
MessageDigest.getInstance("SHA-256");
        byte[] bytes = pass.getBytes("UTF-8");
        digest.update(bytes, 0, bytes.length);
        byte[] key = digest.digest();
        SecretKeySpec secretKeySpec = new SecretKeySpec(key,
"AES");
        return secretKeySpec;
    }
}
```

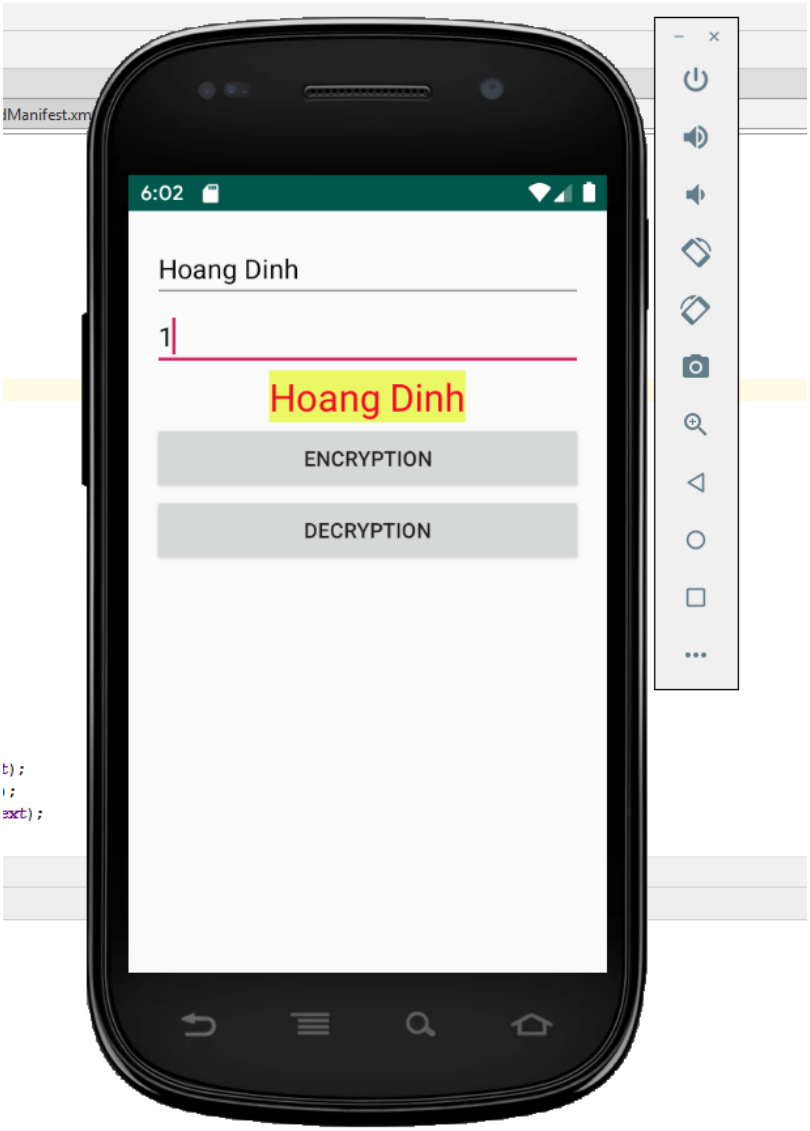
Phase 3: run and debug the app



If you enter “2” in password edit-text to decrypt, then seeing the Wrong Password



Enter 1 in password edit-text to decrypt, then we have the right message.

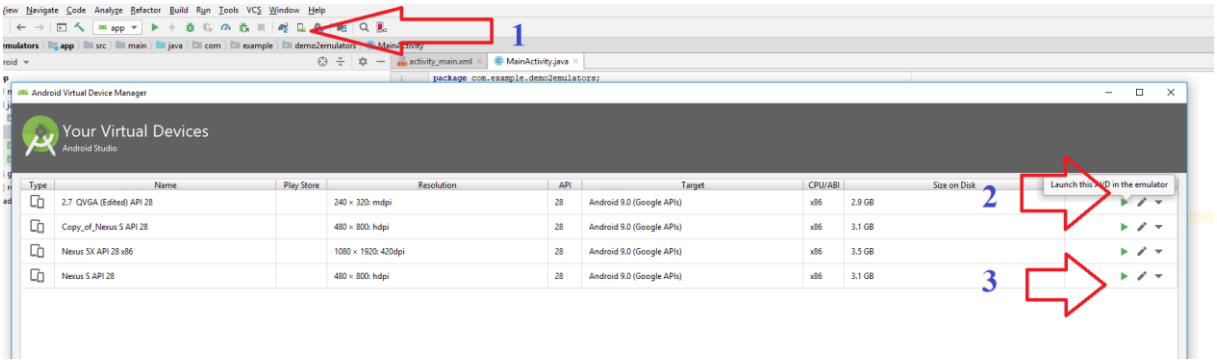


Task 2: Demo message app and coding Explicit Intent for encrypted app

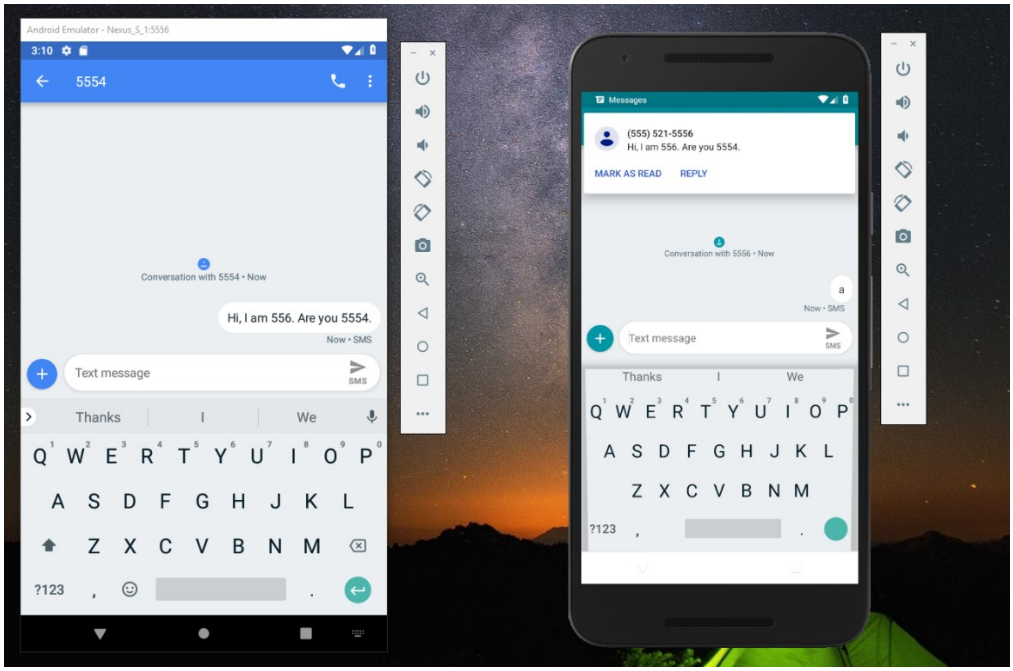
Demo message app:

This is the demonstration of how two emulators will send and receive the messages. This likes the real messenger app.

First, open two emulators by opening AVD manager then launch two emulators. If you have one in AVD manager. You can create one more as guided from lab 1.



With two emulators, you can send and receive messages between them, like this.



Suggest for coding Explicit Intent for encrypted app:

We can imagine that we use explicit intent to work with two activities like simulating with two emulators above.

From the first task, we can make the explicit intent for the app as follows.

- Explicit Intent for encryption app. You click the encryption button new activity starts (new Explicit Intent).
- The new activity is the decryption page. It includes a text-view to display the encrypted message; a Password edit-text to get your password input to decrypt the cipher text; a text-view to show the decrypted message; a decryption button to decrypt the message; a go back button to come back the first page;

- It requires your symmetric password (AES method). You enter the password. If the password is wrong (different from the password for encryption), then toast on the screen the message "Wrong password! Please try it again". If the password is right, then the raw message will appear. You click the go back then immediately going back to the first page.

Task 3: Introduction of BROADCAST package

Introduction of Broadcast in Android: To send and receive the message between two mobile phones by using Broadcast package.
In android, we can send SMS from our android application in two ways either by using **SMSManager api** or **Intents based**.

1. Android Send SMS using SMSManager API
In android, to send SMS using SMSManager API we need to write the code like as shown below.

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage(phoneNum, null, message, null, null);
```

SMSManager API required **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

2. Android Send SMS using Intent
In android, Intent is a messaging object which is used to request an action from another app components such as activities, services, broadcast receivers and content providers. To know more about Intent object in android check this Android Intents with Examples.

To send SMS using Intent object, we need to write the code like as shown below.

```
Intent i = new Intent(Intent.ACTION_VIEW);
i.setDataAndType(Uri.parse("smsto:"),"vnd.android-dir/mms-sms");
i.setType("vnd.android-dir/mms-sms");
i.setData(Uri.parse("smsto:"));
i.putExtra("address", new String(txtMobile.getText().toString()));
i.putExtra("sms_body",txtMessage.getText().toString());
startActivity(Intent.createChooser(i, "Send sms via:"));
```

Even for Intent, it required a **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

The link below help you understand about the app send and receive message
https://www.tutlane.com/tutorial/android/android-send-sms-with-examples

With receive the message, we will do similar, but we will use the permission in manifest.

```
<uses-permission android:name="android.permission.RECEIVE_SMS">
```

I recommend you can find out the send and receive message app in this below link:
https://google-developer-training.github.io/android-developer-phone-sms-course/Lesson%202/2_p_sending_sms_messages.html
and you can get more significant information from this trusted website for Android developer.
https://stackoverflow.com/questions/6869961/how-to-send-and-receive-sms-from-android-app

Task 4: Suggestions of a real security messenger app

After coding for the send and receive message app from previous app, you can integrate Task 1 and Task 3 to build the new messenger app with cryptography mode. This task can be developed in your research project.