SMS Encryption

***Abstract*—In this lab we will try to create a secure SMS Android application. The secure SMS application that we designed utilizes encryption and decryption, which means that if there is some malware in the middle and tries to intercept or view our short message body it will get nothing but some random bytes (the so-called cipher text). (I have provided all source code for this lab, so you can simply copy and paste it into your project, but make sure to understand it fully, since I will only provide code fragments in the future.)**

# LAb Activity

Create a new Android application Project

Application Name: EncDecSMS

Project Name: EncDecSMS

Package Name: android.encdecsms

When creating a blank activity, specify

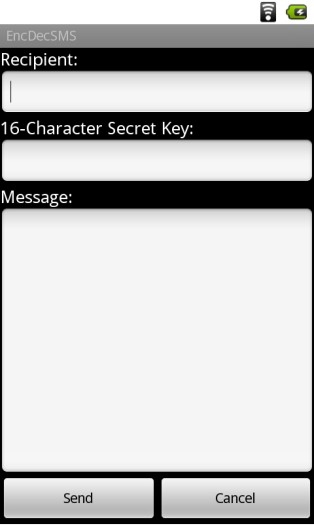
Activity Name: EncDecSMSActivity

Layout Name: main

Copy the following XML code to layout ==> main.xml:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="vertical" >      <LinearLayout          android:id="@+id/linearLayout1"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:orientation="vertical" >          <TextView              android:id="@+id/textView1"              android:layout\_width="wrap\_content"              android:layout\_height="wrap\_content"              android:text="Recipient:"              android:textAppearance="?android:attr/textAppearanceMedium" />      </LinearLayout>      <EditText          android:id="@+id/recNum"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:inputType="phone" >          <requestFocus />      </EditText>      <TextView          android:id="@+id/textView2"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="16-Character Secret Key:"          android:textAppearance="?android:attr/textAppearanceMedium" />      <EditText          android:id="@+id/secretKey"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:inputType="textPassword" />      <TextView          android:id="@+id/textView3"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Message:"          android:textAppearance="?android:attr/textAppearanceMedium" />      <EditText          android:id="@+id/msgContent"          android:layout\_width="match\_parent"          android:layout\_height="208dp"          android:layout\_weight="0.37"          android:inputType="textMultiLine" />      <LinearLayout          android:id="@+id/linearLayout2"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content" >          <Button              android:id="@+id/Send"              android:layout\_width="148dp"              android:layout\_height="wrap\_content"              android:layout\_weight="0.06"              android:text="Send" />          <Button              android:id="@+id/cancel"              android:layout\_width="wrap\_content"              android:layout\_height="wrap\_content"              android:layout\_weight="0.45"              android:text="Cancel" />      </LinearLayout>  </LinearLayout> |

The code above will create a layout as the following:

[](https://sites.google.com/site/mobilesecuritylabware/3-data-location-privacy/lab-activity/cryptography/cryptography-mobile-labs/encryption-decryption/2-lab-activity/lab-activity/mainlayout.jpg?attredirects=0)

Rightclick on layout 🡪 new 🡪 Android XML File, name it onreceive. Copy and paste the following code to onreceive.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      android:orientation="vertical" >      <TextView          android:id="@+id/textView1"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Sender:"          android:textAppearance="?android:attr/textAppearanceMedium" />      <LinearLayout          android:id="@+id/linearLayout1"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:orientation="vertical" >          <TextView              android:id="@+id/senderNum"              android:layout\_width="244dp"              android:layout\_height="wrap\_content"              android:textAppearance="?android:attr/textAppearanceMedium" />      </LinearLayout>      <TextView          android:id="@+id/textView2"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="16-Character Secret Key:"          android:textAppearance="?android:attr/textAppearanceMedium" />      <EditText          android:id="@+id/secretKey"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          android:inputType="textPassword" >          <requestFocus />      </EditText>      <TextView          android:id="@+id/textView3"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Received Encrypted Message:"          android:textAppearance="?android:attr/textAppearanceMedium" />      <TextView          android:id="@+id/encryptedMsg"          android:layout\_width="match\_parent"          android:layout\_height="130dp" />      <TextView          android:id="@+id/textView4"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Decrypted Message:"          android:textAppearance="?android:attr/textAppearanceMedium" />      <TextView          android:id="@+id/decryptedMsg"          android:layout\_width="match\_parent"          android:layout\_height="98dp"          android:layout\_weight="0.05" />      <LinearLayout          android:id="@+id/linearLayout2"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content" >          <Button              android:id="@+id/submit"              android:layout\_width="159dp"              android:layout\_height="wrap\_content"              android:text="Submit" />          <Button              android:id="@+id/cancel"              android:layout\_width="match\_parent"              android:layout\_height="wrap\_content"              android:text="Cancel" />      </LinearLayout>  </LinearLayout> |

The code above will create a layout as the following:

[](https://sites.google.com/site/mobilesecuritylabware/3-data-location-privacy/lab-activity/cryptography/cryptography-mobile-labs/encryption-decryption/2-lab-activity/lab-activity/onreceive.jpg?attredirects=0)

In the Project explorer under EncDecSMS folder ==> src (right click on src) ==> new ==> class   
In the Package name enter the same name which you had used while creating new project i.e. android.encdecsms

For Name enter EncDecSMSActivity.java, copy and paste the following code: 

|  |
| --- |
| package android.encdecsms;  import java.security.Key;  import java.util.ArrayList;  import javax.crypto.Cipher;  import javax.crypto.spec.SecretKeySpec;  import android.app.Activity;  import android.os.Bundle;  import android.telephony.SmsManager;  import android.view.View;  import android.widget.Button;  import android.widget.EditText;  import android.widget.Toast;  public class EncDecSMSActivity extends Activity {  /\*\* Called when the activity is first created. \*/  EditText recNum;  EditText secretKey;  EditText msgContent;  Button send;  Button cancel;  @Override  public void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.main);  recNum = (EditText) findViewById(R.id.recNum);  secretKey = (EditText) findViewById(R.id.secretKey);  msgContent = (EditText) findViewById(R.id.msgContent);  send = (Button) findViewById(R.id.Send);  cancel = (Button) findViewById(R.id.cancel);  // finish the activity when click Cancel button  cancel.setOnClickListener(new View.OnClickListener() {  public void onClick(View v) {  finish();  }  });  // encrypt the message and send when click Send button  send.setOnClickListener(new View.OnClickListener() {  public void onClick(View v) {  String recNumString = recNum.getText().toString();  String secretKeyString = secretKey.getText().toString();  String msgContentString = msgContent.getText().toString();  // check for the validity of the user input  // key length should be 16 characters as defined by AES-128-bit  if (recNumString.length() > 0 && secretKeyString.length() > 0  && msgContentString.length() > 0  && secretKeyString.length() == 16) {  // encrypt the message  byte[] encryptedMsg = encryptSMS(secretKeyString,  msgContentString);  // convert the byte array to hex format in order for  // transmission  String msgString = byte2hex(encryptedMsg);  // send the message through SMS  sendSMS(recNumString, msgString);  // finish  finish();  } else  Toast.makeText(  getBaseContext(),  "Please enter phone number, secret key and the message. Secret key must be 16 characters!",  Toast.LENGTH\_SHORT).show();  }  });  }  public static void sendSMS(String recNumString, String encryptedMsg) {  try {  // get a SmsManager  SmsManager smsManager = SmsManager.getDefault();  // Message may exceed 160 characters  // need to divide the message into multiples  ArrayList<String> parts = smsManager.divideMessage(encryptedMsg);  smsManager.sendMultipartTextMessage(recNumString, null, parts,  null, null);  } catch (Exception e) {  e.printStackTrace();  }  }  // utility function  public static String byte2hex(byte[] b) {  String hs = "";  String stmp = "";  for (int n = 0; n < b.length; n++) {  stmp = Integer.toHexString(b[n] & 0xFF);  if (stmp.length() == 1)  hs += ("0" + stmp);  else  hs += stmp;  }  return hs.toUpperCase();  }  // encryption function  public static byte[] encryptSMS(String secretKeyString,  String msgContentString) {  try {  byte[] returnArray;  // generate AES secret key from user input  Key key = generateKey(secretKeyString);  // specify the cipher algorithm using AES  Cipher c = Cipher.getInstance("AES");  // specify the encryption mode  c.init(Cipher.ENCRYPT\_MODE, key);  // encrypt  returnArray = c.doFinal(msgContentString.getBytes());  return returnArray;  } catch (Exception e) {  e.printStackTrace();  byte[] returnArray = null;  return returnArray;  }  }  private static Key generateKey(String secretKeyString) throws Exception {  // generate secret key from string  Key key = new SecretKeySpec(secretKeyString.getBytes(), "AES");  return key;  }  } |

After that, we create another two new classes and name them as "DisplaySMSActivity.java" and "SmsBroadCastReceiver.java".

For DisplaySMSActivity.java, copy and paste the following code:

|  |
| --- |
| package android.encdecsms;  import java.security.Key;  import javax.crypto.Cipher;  import javax.crypto.spec.SecretKeySpec;  import android.os.Bundle;  import android.view.View;  import android.widget.Button;  import android.widget.EditText;  import android.widget.TextView;  import android.widget.Toast;  import android.app.Activity;  public class DisplaySMSActivity extends Activity {  EditText secretKey;  TextView senderNum;  TextView encryptedMsg;  TextView decryptedMsg;  Button submit;  Button cancel;  String originNum = "";  String msgContent = "";  @Override  public void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.onreceive);  senderNum = (TextView) findViewById(R.id.senderNum);  encryptedMsg = (TextView) findViewById(R.id.encryptedMsg);  decryptedMsg = (TextView) findViewById(R.id.decryptedMsg);  secretKey = (EditText) findViewById(R.id.secretKey);  submit = (Button) findViewById(R.id.submit);  cancel = (Button) findViewById(R.id.cancel);  // get the Intent extra  Bundle extras = getIntent().getExtras();  if (extras != null) {  // get the sender phone number from extra  originNum = extras.getString("originNum");  // get the encrypted message body from extra  msgContent = extras.getString("msgContent");  // set the text fields in the UI  senderNum.setText(originNum);  encryptedMsg.setText(msgContent);  } else {  // if the Intent is null, there should be something wrong  Toast.makeText(getBaseContext(), "Error Occurs!",  Toast.LENGTH\_SHORT).show();  finish();  }  // when click on the cancel button, return  cancel.setOnClickListener(new View.OnClickListener() {  public void onClick(View v) {  finish();  }  });  // when click on the submit button decrypt the message body  submit.setOnClickListener(new View.OnClickListener() {  public void onClick(View v) {  // user input the AES secret key  String secretKeyString = secretKey.getText().toString();               //key length should be 16 characters as defined by AES-128-bit  if (secretKeyString.length() > 0  && secretKeyString.length() == 16) {  try {  // convert the encrypted String message body to a byte  // array  byte[] msg = hex2byte(msgContent.getBytes());  // decrypt the byte array  byte[] result = decryptSMS(secretKey.getText()  .toString(), msg);  // set the text view for the decrypted message  decryptedMsg.setText(new String(result));  } catch (Exception e) {  // in the case of message corrupted or invalid key  // decryption cannot be carried out  decryptedMsg.setText("Message Cannot Be Decrypted!");  }  } else  Toast.makeText(getBaseContext(),  "You must provide a 16-character secret key!",  Toast.LENGTH\_SHORT).show();  }  });  }  // utility function: convert hex array to byte array  public static byte[] hex2byte(byte[] b) {  if ((b.length % 2) != 0)  throw new IllegalArgumentException("hello");  byte[] b2 = new byte[b.length / 2];  for (int n = 0; n < b.length; n += 2) {  String item = new String(b, n, 2);  b2[n / 2] = (byte) Integer.parseInt(item, 16);  }  return b2;  }  // decryption function  public static byte[] decryptSMS(String secretKeyString, byte[] encryptedMsg)  throws Exception {  // generate AES secret key from the user input secret key  Key key = generateKey(secretKeyString);  // get the cipher algorithm for AES  Cipher c = Cipher.getInstance("AES");  // specify the decryption mode  c.init(Cipher.DECRYPT\_MODE, key);  // decrypt the message  byte[] decValue = c.doFinal(encryptedMsg);  return decValue;  }  private static Key generateKey(String secretKeyString) throws Exception {  // generate AES secret key from a String  Key key = new SecretKeySpec(secretKeyString.getBytes(), "AES");  return key;  }  } |

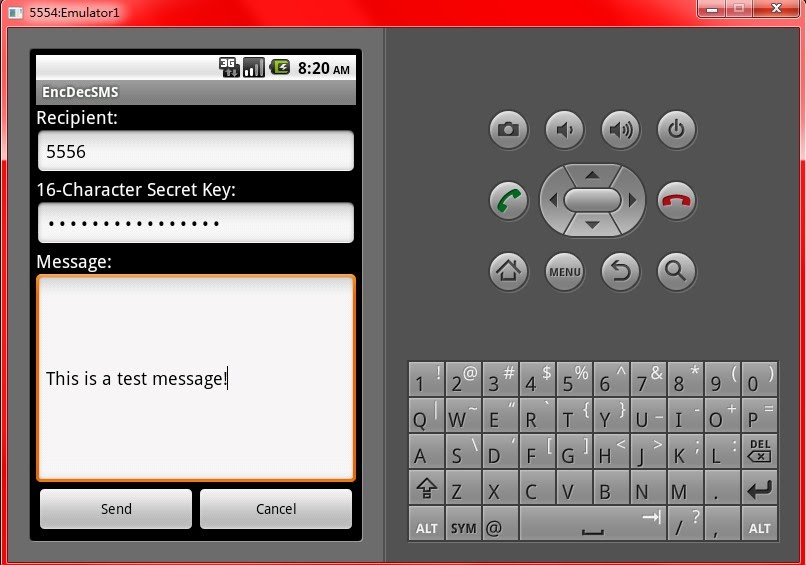
For SmsBroadCastReceiver.java, copy and paste the following code:

|  |
| --- |
| package android.encdecsms;  import android.content.BroadcastReceiver;  import android.content.Context;  import android.content.Intent;  import android.os.Bundle;  import android.telephony.SmsMessage;  public class SmsBroadCastReceiver extends BroadcastReceiver {  @Override  public void onReceive(Context context, Intent intent) {  Bundle bundle = intent.getExtras();  // Specify the bundle to get object based on SMS protocol "pdus"  Object[] object = (Object[]) bundle.get("pdus");  SmsMessage sms[] = new SmsMessage[object.length];  Intent in=new Intent(context,DisplaySMSActivity.class);  in.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);  in.addFlags(Intent.FLAG\_ACTIVITY\_SINGLE\_TOP);  String msgContent = "";  String originNum = "";  StringBuffer sb = new StringBuffer();  for (int i = 0; i < object.length; i++) {  sms[i] = SmsMessage.createFromPdu((byte[]) object[i]);  // get the received SMS content  msgContent = sms[i].getDisplayMessageBody();    //get the sender phone number  originNum = sms[i].getDisplayOriginatingAddress();    //aggregate the messages together when long message are fragmented  sb.append(msgContent);    //abort broadcast to cellphone inbox  abortBroadcast();  }    //fill the sender's phone number into Intent  in.putExtra("originNum", originNum);    //fill the entire message body into Intent  in.putExtra("msgContent", new String(sb));    //start the DisplaySMSActivity.java  context.startActivity(in);    }  } |

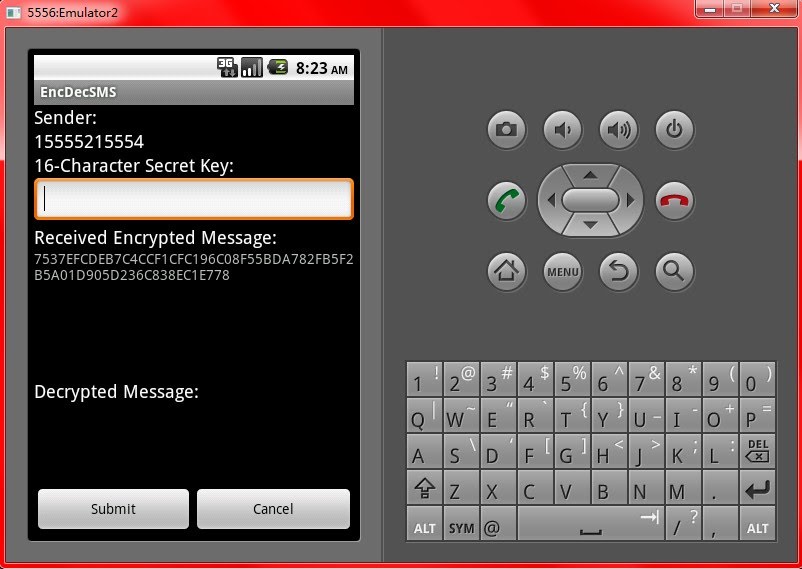
Lastly, go to AndroidManifest.xml. Copy and pate the following code: ( placed under main project folder i.e. EncDecSMS==>AndroidManifest.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"      package="android.encdecsms"      android:versionCode="1"      android:versionName="1.0" >      <uses-sdk android:minSdkVersion="8" />      <uses-permission android:name="android.permission.RECEIVE\_SMS"></uses-permission>      <uses-permission android:name="android.permission.SEND\_SMS" />      <application          android:icon="@drawable/ic\_launcher"          android:label="@string/app\_name" >          <activity              android:name=".EncDecSMSActivity"              android:label="@string/app\_name" >              <intent-filter>                  <action android:name="android.intent.action.MAIN" />                  <category android:name="android.intent.category.LAUNCHER" />              </intent-filter>          </activity>            <activity android:name="DisplaySMSActivity">  </activity>          <receiver android:name=".SmsBroadCastReceiver">              <intent-filter android:priority="1000">                  <action android:name="android.provider.Telephony.SMS\_RECEIVED"/>              </intent-filter>          </receiver>      </application>  </manifest> |

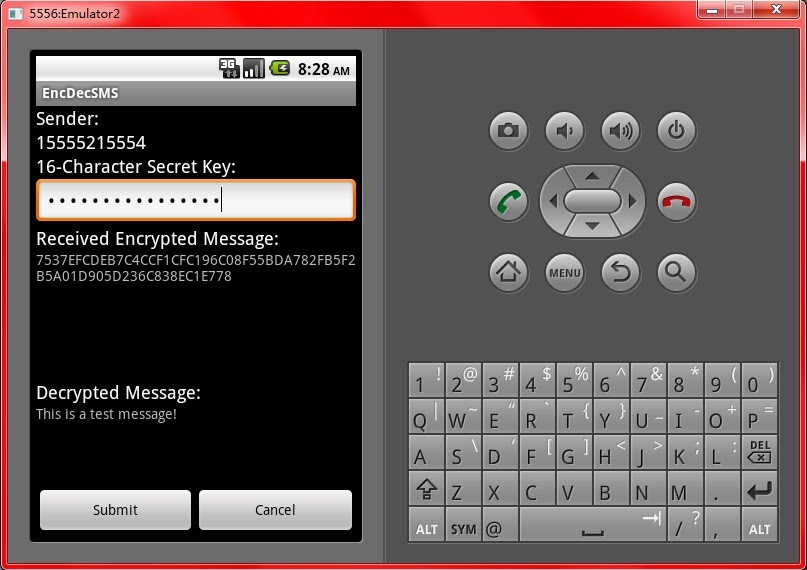
Open two emulators on your eclipse. Run the Android Project "EncDecSMS" on both emulators. Emulator1 serves as a sender which sends encrypted short message to Emulator 2. Emulator 2 serves as receiver which receives the short message from Emulator 1 and decrypts the received cipher text message. Upon starting the activity, since we will use Emulator 1 to send the short message, we should press "Cancel" button on Emulator 2 in order for Emulator 2 to receive the message. On Emulator 1 we fill in the corresponding information such as recipient, 16-character AES secret key and the message body. Here, one thing should be noticed: the AES secret key should be a 16-character input. This is because the AES encryption algorithm we use is a 128-bit block cipher which must uses a 128-bit, a 192-bit or a 256-bit secret key. Here we set it to only accept a 16-character (128-bit) secret key for simplicity and we use "1111111111111111" as secret key for demonstration.

[](https://sites.google.com/site/mobilesecuritylabware/3-data-location-privacy/lab-activity/cryptography/cryptography-mobile-labs/encryption-decryption/2-lab-activity/lab-activity/em1.jpg?attredirects=0)

After filling these information, the user can press "Send" button and then the message is sent to Emulator 2. Upon receiving, the Emulator 2 is as the following:

[](https://sites.google.com/site/mobilesecuritylabware/3-data-location-privacy/lab-activity/cryptography/cryptography-mobile-labs/encryption-decryption/2-lab-activity/lab-activity/em2.jpg?attredirects=0)

On Emulator 2 we can see that, the sender's phone number and the received encrypted messages has been filled on those text fields. Then, the user should input the secret key in the input box in order to use it to decrypt the message. The secret key should be the same as the one which is used to encrypt the message, the one "1111111111111111". After filling in a correct secret key in the box, press "Submit" button and the decrypted message should appear.

[](https://sites.google.com/site/mobilesecuritylabware/3-data-location-privacy/lab-activity/cryptography/cryptography-mobile-labs/encryption-decryption/2-lab-activity/lab-activity/res.jpg?attredirects=0)

If the message is corrupted or the secret key is wrong, then the message cannot be decrypted correctly.

# Report

Please include step-by-step screenshots in the lab report, with your name in English as the username, and your student ID as the password.