Brandon Davidoff

12-4-2011

Dr. Li Yang

Securing Android Communication Using Cryptography

The scope of my Thesis project is to create an encryption decryption application for the android Smartphone. Earlier I said that I plan to use the encryption decryption application for text messaging and for local files, this has not changed, and has in fact been completed. The application is named "De-Cryptify" was developed on Eclipse with the Android SDK. I have used the same sources as before, Android Application Development by Wei-Meng Lee, and Beginning Cryptography with Java by David Hook. I used these sources to help with the coding for the android phone and for understanding cryptography in general. I have also used tutorials from Oracle to better understand cryptography.

The project as it now stands has completed the base requirements. It has a much more polished framework, with six activities and one broadcast receiver. The first De-Cryptify Activity remains unchanged, it opens up one of two options either En-Cryptify or De-Cryptify options. De-Cryptify Options still has only De-Cryptify a file option. It also has a new button for going back to the main menu which essentially closes the De-Cryptify Options activity. Had I more time this options menu would include more options based on the different types of encryption available.

The De-Cryptify file activity now has two buttons load file and back to De-Cryptify Options button. The first button loads an encrypted text file, its key and ivParameter spec, and then runs the decryption program and saves the decryptified file as a text document. The second button closes the activity. The following is the source code for the De-Cryptify File Activity.

```
package com.davidoffproductions.DCryptify;

import android.app.Activity;
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.File;

import java.security.*;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.os.Environment;
import android.widget.Toast;

public class Decryptifyfile extends Activity {
      private static final int READ_BLOCK_SIZE=100;
      @Override
      public void onCreate(Bundle savedInstanceState){
            super.onCreate(savedInstanceState);
            setContentView(R.layout.decryptifyfile);
            Button loadBtn = (Button) findViewById(R.id.btn_Decryptloadfile);
            Button menu = (Button) findViewById(R.id.btn_Ok);
            //The following button closes the activity opening the previous
activity
             menu.setOnClickListener(new View.OnClickListener(){
```

```
                public void onClick(View view){
                finish();
                }
            });
           //The following button loads the encrypted file and starts the
decryption process

          loadBtn.setOnClickListener(new View.OnClickListener() {

                public void onClick(View v) {

                     try{

                          //loads the secret key

                          File sdCard =
Environment.getExternalStorageDirectory();
                          File directory = new File
(sdCard.getAbsolutePath() + "/MyFiles");
                          File file = new File(directory,
"SecretKey.txt");
                          FileInputStream fIn = new
FileInputStream(file);

                          byte[] theKey = new byte[32];
                          fIn.read(theKey);




                          //loads the encrypted message

                          File sdCard2 =
Environment.getExternalStorageDirectory();
```

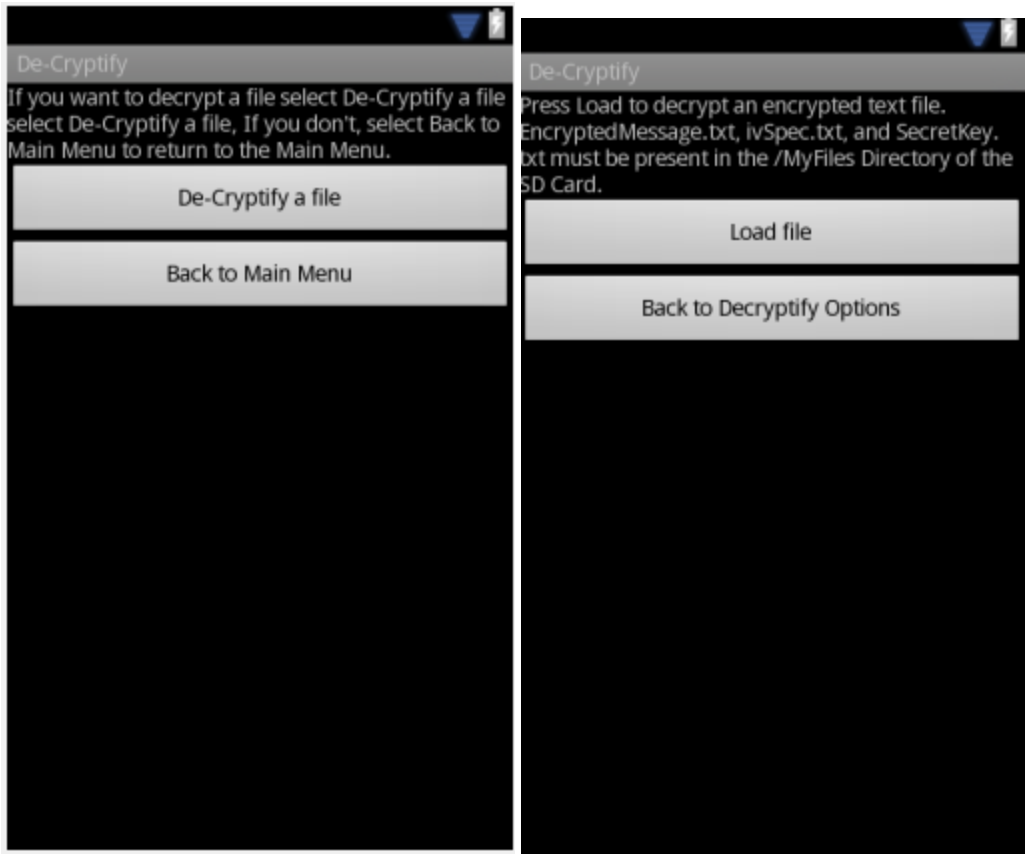
```
                                File directory2 = new File
(sdCard2.getAbsolutePath() + "/MyFiles");

                                File file2 = new File(directory2,
"EncryptedMessage.txt");

                                FileInputStream fIn2 = new
FileInputStream(file2);

                                InputStreamReader isr2 = new
InputStreamReader(fIn2);

                                char [] inputBuffer2 = new
char[READ_BLOCK_SIZE];

                                String s2 = "";

                                int charRead2;



                                while ((charRead2 =
isr2.read(inputBuffer2))>0){

                                      String readString2 =
String.copyValueOf(inputBuffer2, 0, charRead2);

                                      s2 += readString2;

                                      inputBuffer2 = new char
[READ_BLOCK_SIZE];

                                }


                                //loads the ivSpec


                                File sdCard4 =
Environment.getExternalStorageDirectory();

                                File directory4 = new File
(sdCard4.getAbsolutePath() + "/MyFiles");

                                File file4 = new File(directory4,
"ivSpec.txt");

                                FileInputStream fIn4 = new
FileInputStream(file4);



                                byte[] IV = new byte[16];

                                fIn4.read(IV);
```

```
                                //establishes all parameters


                                IvParameterSpec ivSpec = new
IvParameterSpec(IV);
                                SecretKeySpec aeskeySpec = new
SecretKeySpec(theKey, "AES");


                                //establishes cipher and begins decryption
process


                                Cipher cipher =
Cipher.getInstance("AES/CTR/NoPadding", "BC");
                                cipher.init(Cipher.DECRYPT_MODE, aeskeySpec,
ivSpec);




                                byte[] plainText =
cipher.doFinal(toByteArray(s2));
                                String value = new String(plainText);




                                Toast.makeText(getBaseContext(), "Decryption
successful", Toast.LENGTH_SHORT).show();


                                //saves the encrypted message to Decrpyted.txt
in the /MyFiles folder in the SD Card


                                File sdCard3 =
Environment.getExternalStorageDirectory();
                                File directory3 = new File
(sdCard3.getAbsolutePath() + "/MyFiles");
                                directory3.mkdirs();
                                File file3 = new File(directory3,
"Decrypted.txt");
```

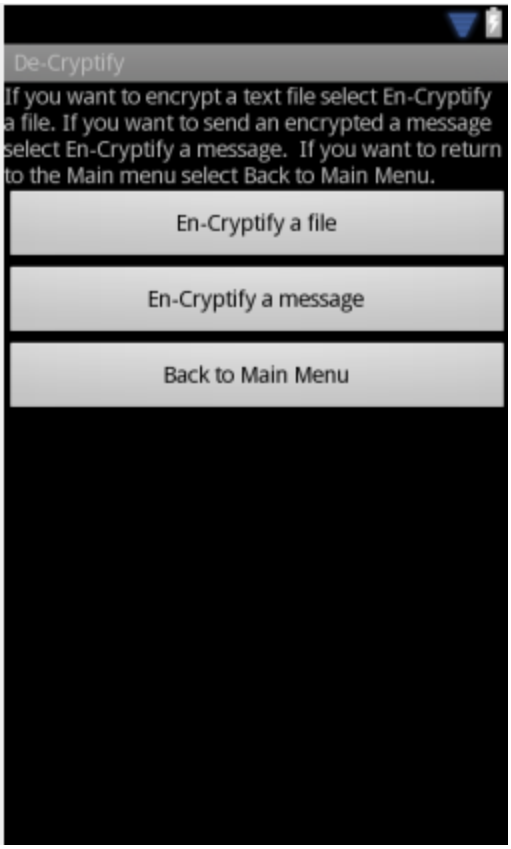
```
                                FileOutputStream fOut = new
FileOutputStream(file3);


                                OutputStreamWriter osw = new
OutputStreamWriter(fOut);
                                osw.write(value);
                                osw.flush();
                                osw.close();


                                Toast.makeText(getBaseContext(), "Save
successful", Toast.LENGTH_SHORT).show();


                            }catch (NoSuchAlgorithmException e) {


      Toast.makeText(getBaseContext(),"AlgorithmException",
Toast.LENGTH_SHORT).show();
                        } catch (NoSuchPaddingException e) {
                            Toast.makeText(getBaseContext(),"Padding",
Toast.LENGTH_SHORT).show();
                        } catch (InvalidKeyException e) {


      Toast.makeText(getBaseContext(),"InvalidKeyException",
Toast.LENGTH_SHORT).show();
                        }


                            catch(Exception e) {
                                  Toast.makeText(getBaseContext(),"There seems to
be a problem", Toast.LENGTH_SHORT).show();
                            }
                                                    }


                    });




      }
```

```
	//handles string and byte conversions of text message during the
encryption process.


	public static String toString(byte[] bytes, int length){
		char[]	chars = new char [length];
		for (int i = 0; i != chars.length; i++){
			chars[i] = (char)(bytes[i] & 0xff);
		}
		return new String(chars);
	}
	public static String toString (byte[] bytes){
		return toString(bytes, bytes.length);
	}
	public static byte[] toByteArray(String string){
		byte[] bytes = new byte[string.length()];
		char[] chars = string.toCharArray();
		for(int i = 0; i != chars.length; i++){
			bytes[i] = (byte)chars[i];
		}
		return bytes;
	}
}
```

The De-Cryptify File requires that an encrypted message text file be placed in the MyFiles folder, encrypted message text files can be created in the En-Cryptify File, and En-Cryptify Message activities. The De-Cryptify File activity also requires an ivParameterSpec and a Key which are also generated in the same activities as the encrypted message text file. It exports the decryptified message in a text file in the MyFiles folder.



En-Cryptify Options still has the two options to En-Cryptify File, En-Cryptify Message. It also has an option to go back to the menu which finishes the En-Cryptify Option Activity. En-Cryptify File has two options the first is Load File which loads a file by the name of MessageToBeEncrypted.txt. It loads the text of the file and encrypts it creating a key, an ivSpecParameter and saves the encrypted text to a file.




```
package com.davidoffproductions.DCryptify;

import android.app.Activity;

import android.os.Bundle;

import android.os.Environment;

import android.view.View;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileInputStream;
```

```
import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.security.Key;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.spec.IvParameterSpec;

import android.widget.Button;

import android.widget.Toast;



public class Encryptifyfile extends Activity  {

      private static final int READ_BLOCK_SIZE=100;

      @Override

      public void onCreate(Bundle savedInstanceState){

            super.onCreate(savedInstanceState);

            setContentView(R.layout.encryptifyfile);

            Button loadBtn = (Button) findViewById(R.id.btn_Encryptloadfile);

            Button menu = (Button) findViewById(R.id.btn_Ok);

            //the button below closes the activity opening the previous
activity
```

```
            menu.setOnClickListener(new View.OnClickListener(){
            public void onClick(View view){
            finish();
            }
        });
            //the button below initializes the load and encrypt phase
            loadBtn.setOnClickListener(new View.OnClickListener() {
                                public void onClick(View v) {
                        try
                        {
                                //loades MessageToBeEncrypted.txt
                                File sdCard2 =
Environment.getExternalStorageDirectory();
                                File directory2 = new File
(sdCard2.getAbsolutePath() + "/MyFiles");
                                File file2 = new File(directory2,
"MessageToBeEncrypted.txt");
                                FileInputStream fIn2 = new
FileInputStream(file2);
                                InputStreamReader isr2 = new
InputStreamReader(fIn2);
```

```
                              char [] inputBuffer2 = new
char[READ_BLOCK_SIZE];
                              String message = "";
                              int charRead2;

                              while ((charRead2 =
isr2.read(inputBuffer2))>0){
                                    String readString2 =
String.copyValueOf(inputBuffer2, 0, charRead2);
                                    message += readString2;
                                    inputBuffer2 = new char
[READ_BLOCK_SIZE];
                              }
                              //generates an 256 AES encryption key with
Bouncy Castle provider.
                              KeyGenerator generator =
KeyGenerator.getInstance("AES", "BC");
                        generator.init(256);
                        Key encryptionKey = generator.generateKey();
                        //creates the IvParameterivSpec, referencing the
createCtrIvForAES method
                        SecureRandom random = new SecureRandom();
```

```
                                IvParameterSpec ivSpec = createCtrIvForAES(1,
random);

                                //creates a cipher and starts encrypting the
message.

                                Cipher cipher =
Cipher.getInstance("AES/CTR/NoPadding", "BC");
                                String input = message;
                                cipher.init(Cipher.ENCRYPT_MODE, encryptionKey,
ivSpec);

                                byte[] test2 = encryptionKey.getEncoded();

                                byte[] vars2 = ivSpec.getIV();

                                //creates a byte array of the message

                                byte[] cipherText =
cipher.doFinal(toByteArray(input));
```

```
                                //converts the byte array into a string

                                String test = totheString(cipherText);



                                //saves the key

                                File sdCard =
Environment.getExternalStorageDirectory();
                                File directory = new File
(sdCard.getAbsolutePath() + "/MyFiles");
                                directory.mkdirs();
                                File file = new File(directory,
"SecretKey.txt");

                                FileOutputStream fOut = new
FileOutputStream(file);



                                fOut.write(test2);
                                fOut.flush();
```

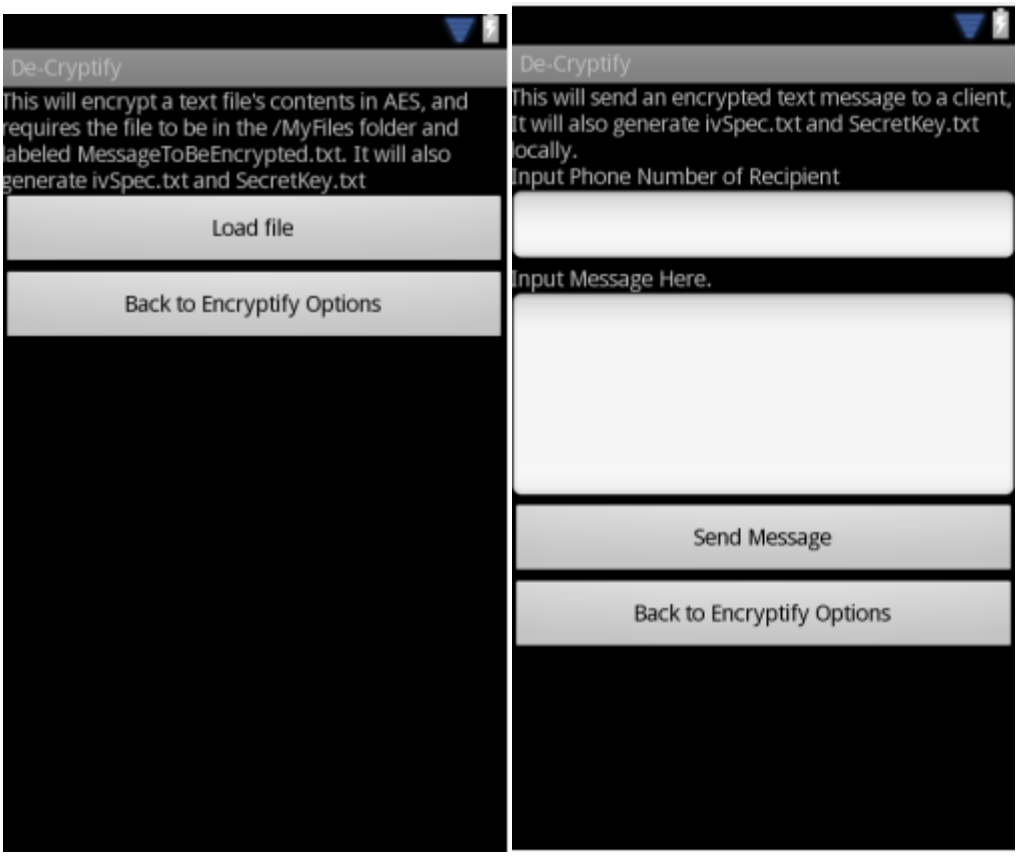
```
                                fOut.close();



                                //saves the ivSpec



                                File sdCard3 =
Environment.getExternalStorageDirectory();
                                File directory3 = new File
(sdCard3.getAbsolutePath() + "/MyFiles");
                                directory3.mkdirs();
                                File file3 = new File(directory3,
"ivSpec.txt");
                                FileOutputStream fOut3 = new
FileOutputStream(file3);
                                fOut3.write(vars2);
                                fOut3.flush();
                                fOut3.close();
                                //saves the encrypted Message



                                File sdCard4 =
Environment.getExternalStorageDirectory();
                        File directory4 = new File (sdCard4.getAbsolutePath()
+ "/MyFiles");
```

```
                        directory4.mkdirs();

                        File file4 = new File(directory4,
"EncryptedMessage.txt");

                        FileOutputStream fOut4 = new FileOutputStream(file4);



                        OutputStreamWriter osw4 = new
OutputStreamWriter(fOut4);

                        osw4.write(test);

                        osw4.flush();

                        osw4.close();



                        Toast.makeText(getBaseContext(), "File Encrypted and
Saved.", Toast.LENGTH_SHORT).show();

                        }

                        catch (Exception e){

                              Toast.makeText(getBaseContext(), "There seems
to be a problem", Toast.LENGTH_SHORT).show();

                        }

                  }});

      }

      //This portion of code is present in the Beginning Cryptography with
Java book and is evoked to create
```

```
	//The IvParameter Spec
	public static IvParameterSpec createCtrIvForAES(int messageNumber,
SecureRandom random){
		byte[]	ivBytes = new byte[16];
		random.nextBytes(ivBytes);
		ivBytes[0] = (byte)(messageNumber >> 24);
		ivBytes[1] = (byte)(messageNumber >> 16);
		ivBytes[2] = (byte)(messageNumber >> 8);
		ivBytes[3] = (byte)(messageNumber >> 0);
		for (int i = 0; i != 7; i++){
			ivBytes[8 + i] = 0;
		}
		ivBytes[15] = 1;
		return new IvParameterSpec(ivBytes);
	}
	//This portion of code is present in the Beginning Cryptography with
Java book and is evoked to
	//convert strings and bytes.
public static String totheString(byte[] bytes, int length){
	char[]	chars = new char [length];
	for (int i = 0; i != chars.length; i++){
```

```
            chars[i] = (char)(bytes[i] & 0xff);
      }
      return new String(chars);
}
public static String totheString (byte[] bytes){
      return totheString(bytes, bytes.length);
}
public static byte[] toByteArray(String string){
      byte[] bytes = new byte[string.length()];
      char[] chars = string.toCharArray();
      for(int i = 0; i != chars.length; i++){
            bytes[i] = (byte)chars[i];
      }
      return bytes;
}
}
```

En-Cryptify message is very similar code to the above although instead of loading an encrypted message from a file and saving it to a file, it allows the user to type their own message to be encrypted into an EditText to be sent to another client. It still saves the key and ivSpecParameter locally, and the recipient still needs those files to decryptify the message.



My original plan while dealing with the transmission and decryption of the text message was to have the decryption takes place on the broadcast receiver. This became impractical since the broadcast receiver picks up every text message sent, if I had sent the message the key and ivparameterspec, the message would be treated the same as the encrypted message, making the application distinguish between the messages would have posed a challenge, in addition to this ensuring that the client has the key and ivParameterSpec before they receive the message for the broadcast receiver would have made this more difficult. In the end I decided that the decryption should take place in the De-Cryptify file section and that the broadcast receiver should save the encrypted text message rather than try to decode it automatically.

For this project I have accomplished what I had intended to do, create an application that would send encrypted text messages, and encrypt text files. The project uses AES encryption with the Bouncy

Castle provider, and creates a key and ivSpecParameter locally. The project can also Decrypt messages and files created in the En-Cryptify Message, and En-Cryptify Files respectively. If I had more time I would have liked to included more types of encryption, a better way of handling keys, and implementation of passwords.