



## **Data Encryption Application Whitepaper**

***Peng Wang, Atom Software Applications Engineer***

**Application Origination:**  
**Intel SSG**

### **Introduction**

Mobile security is getting personal. Smartphones and tablets contain some of our most precious memories and personal details, including access to financial transactions and other sensitive information. Cybercriminals now develop five new threats every second. With the sharp growth in smartphones and tablets, the bad guys are aggressively attacking these devices and your information with a new range of insidious threats.

Here at Intel, we've begun creating small software applications to provide to independent software vendors (ISVs) as examples of how to use the security features on the platforms based on Intel® Architecture.

The purpose of this paper is to give developers a quick start by describing the basic Android\* data encryption API and showing how to encrypt application data for the Android OS.

We recommend that you try out the features and compile the code as you read through the paper.

## Data Encryption Code and Explanations

The DataEncryption app has three major features: data encryption, progress bar, and performance indicators.

### *DataEncryption Class declaration*

DataEncryption class extends the standard Activity class. It has private attributes for the logging, dialog, and UI components.

```
public class DataEncryption extends Activity {

    private final String TAG = "SECURITY_DATA_ENCRYPTION";
    private final String FLAG_SUC = "SECURITY_DATA_ENCRYPTION_SUCCESSFULLY";
    private final String FLAG_FAIL = "SECURITY_DATAENCRYPTION_FAILED";

    private Boolean mDoneEncryption = false;
    private long mTimeTook = 0;

    ProgressDialog progDialog;
    ProgressThread progThread;
    int delay = 40; // Milliseconds of delay in the update loop
    int maxBarValue = 200; // Maximum value of horizontal progress bar
    int typeBar;

    private Handler mHandler = new Handler();
    private String mIndicatorText;
    private TextView mIndicator;
    private SDPUtility mUtility;
    private ImageButton encryptionButton;
```

## ***onCreate Member Function***

The onCreate member function is called by the framework after an activity is created. Since the DataEncryption class extends the Activity class as you saw above, it is treated like an Activity when started. The initialization for the application occurs here.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.encrypted);

    mUtility = new SDPUtility();
    mHandler.postDelayed(mUpdateTimeTask, 100);
    mIndicator = (TextView) findViewById(R.id.encryptedIndicator);

    typeBar = 0;
    encryptionButton = (ImageButton) findViewById(R.id.encryptedButton);
    encryptionButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            showDialog(typeBar);
            new Thread(new Runnable() {
                public void run() {
                    encryptVideo();
                }
            }).start();
        }
    });
} catch (Exception e) {
    Log.d(TAG, "Exception occurs in security encryption test.", e);
}
}
```

## ***encrypt() function***

The encrypt function does the actual work of data encryption. It first creates an input and output file stream. It then creates the “AES” cipher object and encryption key. Finally, the encryption function reads from the input file stream and writes the encrypted data to the output file stream.

```

private void encryptVideo() {
    FileInputStream fis;
    try {
        fis = new FileInputStream(new
File("/sdcard/h264.mp4"));

        File outfile = new File("/sdcard/h264_enc.mp4");
        int read = 0;
        if (!outfile.exists())
            outfile.createNewFile();
        File decfile = new File("/sdcard/h264_dec.mp4");
        if (!decfile.exists())
            decfile.createNewFile();
        FileOutputStream fos = new FileOutputStream(outfile);
        FileInputStream encfis = new FileInputStream(outfile);
        FileOutputStream decfos = new
FileOutputStream(decfile);
        Cipher encipher = Cipher.getInstance("AES");
        Cipher decipher = Cipher.getInstance("AES");
        KeyGenerator kgen = KeyGenerator.getInstance("AES");

        SecretKey skey = kgen.generateKey();
        encipher.init(Cipher.ENCRYPT_MODE, skey);
        CipherInputStream cis = new CipherInputStream(fis,
encipher);

        decipher.init(Cipher.DECRYPT_MODE, skey);
        CipherOutputStream cos = new
CipherOutputStream(decfos, decipher);
        long start = System.nanoTime();
        Log.d("security", String.valueOf(start));
        while ((read = cis.read()) != -1) {
            fos.write((char) read);
            fos.flush();
        }
        long stop = System.nanoTime();
        Log.d("security", String.valueOf(stop));
        long seconds = (stop - start) / 1000000000; // for
seconds

        Log.d("security", String.valueOf(seconds));
        fos.close();
        mTimeTook = seconds;
        mDoneEncryption = true;

    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (NoSuchPaddingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InvalidKeyException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

## Progress Bar

It often takes a while to encrypt big data files, so it is very useful to have a progress bar to inform the user when the encryption is complete. The progress bar is on a different thread and has both “Spinner” and “horizontal” styles.

```
@Override
protected Dialog onCreateDialog(int id) {
    switch(id) {
        case 0: // Spinner
            progressDialog = new ProgressDialog(this);
            progressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
            progressDialog.setMessage("encrypting data...");
            progThread = new ProgressThread(handler);
            progThread.start();
            return progressDialog;
        case 1: // Horizontal
            progressDialog = new ProgressDialog(this);

progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            progressDialog.setMax(maxBarValue);
            progressDialog.setMessage("encrypting data...");
            progThread = new ProgressThread(handler);
            progThread.start();
            return progressDialog;
        default:
            return null;
    }
}

final Handler handler = new Handler() {
    public void handleMessage(Message msg) {
        int total = msg.getData().getInt("total");
        progressDialog.setProgress(total);
        if (mDoneEncryption) {
            dismissDialog(typeBar);
            progThread.setState(ProgressThread.DONE);
        }
    }
};
```

## Performance Indicator

Performance is also very important for the data encryption. The DataEncryption app also provides some key indicators such as CPU utilization, memory utilization, and total time spent.

```
private Runnable mUpdateTimeTask = new Runnable() {
    public void run() {
        DecimalFormat df = new DecimalFormat("#.##");
        String cpuUsage = df.format(mUtility.readUsage() * 100) + "%";
        String memUsage = Long
            .toString(mUtility
                .readMem((ActivityManager)
                    getSystemService(ACTIVITY_SERVICE)))
            + "M";

        mIndicatorText = "CPU usage: " + cpuUsage + "\nMemory usage: "
            + memUsage;
        if(mDoneEncryption){
            mIndicatorText = mIndicatorText + "\nTotal Time Spent: " +
mTimeTook + " Seconds";
        }

        mIndicator.setText(mIndicatorText);

        mHandler.postAtTime(this, SystemClock.uptimeMillis() + 2000);
    }
};
```

## Conclusion

By implementing code like the samples described in this paper, you can quickly learn how to encrypt data for Intel Android platforms and know how to measure the basic encryption performance.

## About the author

Peng Wang is a member of the Intel Software and Solutions Group (SSG), Developer Relations Division, Atom High Touch Software Enabling team. Before joining SSG, Peng was leading the integration and validation team for the Ultra Mobile Group.

### Notices

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel and the Intel logo are trademarks of Intel Corporation in the US and/or other countries.

Copyright © 2012 Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.