



Sample Code: Data Encryption Application

Submitted by [Christopher Bird \(Intel\)](#) on January 17, 2014



Share



Tweet



g+Share

[Download as PDF](#)[Download sample code](#)

Android* >

Forum >

[Translate](#) [Disclaimer](#)

Christopher Bird, Software Applications Engineers

Application Origination:
Intel SSG

Introduction

[Encryption](#) is important because it allows you to securely protect data that you don't want

anyone else to have access to. Encryption has been a trending topic in the security community. As more mobile devices store valuable information than ever before, encryption has become crucial to ensure information security.

This paper introduces data encryption APIs that are available through either Java* or OpenSSL*. Both solutions work on the Android* OS.

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



Encrypt

Settings

CPU usage: 8.82%
Memory usage: 1004M
Total Time Spent: 487 ms

Data Encryption Code and Explanations

If you want to encrypt data on Android, you have two options: Java Crypto API and OpenSSL API. We will show you how to encrypt data using both ways.

Java Crypto API

Using Java Crypto API on Android is very straightforward. First, you will need to generate a key for the encryption. There is a `KeyGenerator` class in the `javax.crypto` package that can do this for you.

```
1 mKey = null;
2 try {
3     kgen = KeyGenerator.getInstance("AES");
4     mKey = kgen.generateKey();
5
6 } catch (NoSuchAlgorithmException e) {
7     e.printStackTrace();
8 }
```

Then you can use the generated key to encrypt the data file. This can be done by feeding chunks of bytes to an AES Cipher created by `javax.crypto`.

```
01 // open stream to read origFilepath. We are going to save encrypt
02 InputStream fis = new FileInputStream(origFilepath);
03 File outfile = new File(encFilepath);
04 int read = 0;
05 if (!outfile.exists())
```

```

06         outfile.createNewFile();
07
08         FileOutputStream encfos = new FileOutputStream(outfile);
09         // Create Cipher using "AES" provider
10         Cipher encipher = Cipher.getInstance("AES");
11         encipher.init(Cipher.ENCRYPT_MODE, mKey);
12         CipherOutputStream cos = new CipherOutputStream(encfos, enci
13
14         // capture time it takes to encrypt file
15         start = System.nanoTime();
16         Log.d(TAG, String.valueOf(start));
17
18         byte[] block = new byte[mBlocksize];
19
20         while ((read = fis.read(block,0,mBlocksize)) != -1) {
21             cos.write(block,0, read);
22         }
23         cos.close();
24         stop = System.nanoTime();
25
26         Log.d(TAG, String.valueOf(stop));
27         seconds = (stop - start) / 1000000; // for milliseconds
28         Log.d(TAG, String.valueOf(seconds));
29
30         fis.close();

```

OpenSSL API

Encrypting data in OpenSSL on Android requires writing native C code that can be accessed in Java through JNI calls. It requires more work, but you will get better performance in return.

First, let's generate the key and the iv.

```
01 unsigned char cKeyBuffer[KEYSIZE/sizeof(unsigned char)];
02 unsigned char iv[] = "01234567890123456";
03 int opensslIsSeeded = 0;
04 if (!opensslIsSeeded) {
05     if (!RAND_load_file("/dev/urandom", seedbytes)) {
06         return -1;
07     }
08     opensslIsSeeded = 1;
09 }
10
11 if (!RAND_bytes((unsigned char *)cKeyBuffer, KEYSIZE )) {
12 }
```

Then, we can use the generated key (cKeyBuffer) to encrypt a file. Initialize EVP by feeding it your key and iv. Then feed chunks of bytes to the EVP_EncryptUpdate function. The final chunk of bytes from your file will need to be fed to the EVP_EncryptFinal_ex function.

```
01 if (!(EVP_EncryptInit_ex(e_ctx, EVP_aes_256_cbc(), NULL, cKeyBuf:
02     ret = -1;
03     printf( "ERROR: EVP_ENCRYPTINIT_EXn");
04 }
05
06 // go through file, and encrypt
07 if ( orig_file != NULL ) {
08     origData = new unsigned char[aes_blocksize];
09     encData = new unsigned char[aes_blocksize+EVP_CIPHER_CTX_
10
11     printf( "Encoding file: %sn", filename);
12
13     bytesread = fread(origData, 1, aes_blocksize, orig file);
```

```

14 // read bytes from file, then send to cipher
15 while ( bytesread ) {
16
17
18     if (!(EVP_EncryptUpdate(e_ctx, encData, &len, origData, l
19         ret = -1;
20         printf( "ERROR: EVP_ENCRYPTUPDATEn");
21     }
22     encData_len = len;
23
24     fwrite(encData, 1, encData_len, enc_file );
25     // read more bytes
26     bytesread = fread(origData, 1, aes_blocksize, orig_file)
27 }
28 // last step encryption
29 if (!(EVP_EncryptFinal_ex(e_ctx, encData, &len))) {
30     ret = -1;
31     printf( "ERROR: EVP_ENCRYPTFINAL_EXn");
32 }
33 encData_len = len;
34
35 fwrite(encData, 1, encData_len, enc_file );
36
37 // free cipher
38 EVP_CIPHER_CTX_free(e_ctx);

```

Conclusion

By implementing code like the samples described in this paper, you can quickly learn how to use both Java Crypto API and OpenSSL API to encrypt data on Intel® processor-based platforms running Android.

About the author

Christopher Bird is a member of the Intel Software and Solutions Group (SSG), Developer Relations Division, Intel® Atom™ Processor Innovative Technologies Engineering team.

Related Articles and Resources:

- [Intel Security and Encryption](#)
- [Intel Developer Zone - Android](#)

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

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

For more complete information about compiler optimizations, see our [Optimization Notice](#).

Categories: [Android*](#) , [Developers](#) , [Android*](#)

Comments (1)

[^Top](#)

[PATRICK N.](#) said on Wed, 03/05/2014 - 02:26



Very neat stuff! Though I am not yet fully familiar with android coding yet, the java programming definitely makes up for any of my worries. Even so, i would wish to learn of any other ways of encrypting my data with android other than the two specified ways mentioned...thanks for the info.

Add a Comment

[^Top](#)

(For technical discussions visit our [developer forums](#). For site or software product issues [contact support](#).)

Please [sign in](#) to add a comment. Not a member?

[Join today >](#)

Rate Us ☆☆☆

Look for us on:



English >

