



Data Encryption Application Whitepaper

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.

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.

```
mKey = null;
try {
    kgen = KeyGenerator.getInstance("AES");
    mKey = kgen.generateKey();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

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.

```
// open stream to read origFilepath. We are going to save encrypted contents
to outfile
InputStream fis = new FileInputStream(origFilepath);
File outfile = new File(encFilepath);
int read = 0;
if (!outfile.exists())
    outfile.createNewFile();

FileOutputStream encfos = new FileOutputStream(outfile);
// Create Cipher using "AES" provider
Cipher encipher = Cipher.getInstance("AES");
encipher.init(Cipher.ENCRYPT_MODE, mKey);
CipherOutputStream cos = new CipherOutputStream(encfos, encipher);

// capture time it takes to encrypt file
start = System.nanoTime();
Log.d(TAG, String.valueOf(start));

byte[] block = new byte[mBlocksize];

while ((read = fis.read(block, 0, mBlocksize)) != -1) {
    cos.write(block, 0, read);
}
cos.close();
stop = System.nanoTime();

Log.d(TAG, String.valueOf(stop));
seconds = (stop - start) / 1000000; // for milliseconds
Log.d(TAG, String.valueOf(seconds));

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.

```
unsigned char cKeyBuffer[KEYSIZE/sizeof(unsigned char)];
unsigned char iv[] = "01234567890123456";

int opensslIsSeeded = 0;
if (!opensslIsSeeded) {
    if (!RAND_load_file("/dev/urandom", seedbytes)) {
        return -1;
    }
    opensslIsSeeded = 1;
}

if (!RAND_bytes((unsigned char *)cKeyBuffer, KEYSIZE )) {
}
```

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.

```
if (!(EVP_EncryptInit_ex(e_ctx, EVP_aes_256_cbc(), NULL, cKeyBuffer, iv ))) {
    ret = -1;
    printf( "ERROR: EVP_ENCRYPTINIT_EX\n");
}

// go through file, and encrypt
if ( orig_file != NULL ) {
    origData = new unsigned char[aes_blocksize];
    encData = new unsigned
char[aes_blocksize+EVP_CIPHER_CTX_block_size(e_ctx)]; // potential for
encryption to be 16 bytes longer than original

    printf( "Encoding file: %s\n", filename);

    bytesread = fread(origData, 1, aes_blocksize, orig_file);
    // read bytes from file, then send to cipher
    while ( bytesread ) {

        if (!(EVP_EncryptUpdate(e_ctx, encData, &len, origData,
bytesread))) {
            ret = -1;
            printf( "ERROR: EVP_ENCRYPTUPDATE\n");
        }
        encData_len = len;

        fwrite(encData, 1, encData_len, enc_file );
        // read more bytes
    }
}
```

```
        bytesread = fread(origData, 1, aes_blocksize, orig_file);
    }
    // last step encryption
    if (!(EVP_EncryptFinal_ex(e_ctx, encData, &len))) {
        ret = -1;
        printf( "ERROR: EVP_ENCRYPTFINAL_EX\n");
    }
    encData_len = len;

    fwrite(encData, 1, encData_len, enc_file );

    // free cipher
    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.

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.

Copyright © 2014 Intel Corporation. All rights reserved.

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