A discussion about the misuse of cryptographic operations in Android applications

M.Sc. Rafael Will Macedo de Araujo

DCC – IME – USP

june 13th, 2014
Guide

1. Introduction
2. Definitions
3. Cryptography misuse
4. Case Studies
5. Conclusions
Our goals

- Discuss about the misuse of cryptographic functions by application programmers.
- Give (a very basic) background about how symmetric key cryptography works.
- Introduce some relevant (recent) works on the subject.
  - Analysis of real Android applications (1000+ apps):
    - (Enck, 2011) - ded decompiler.
    - (Egele, 2013) - CryptoLint.
  - (Atwater, 2014) - analyzes 90 public projects from Github (different programming languages and project types - not necessarily mobile).
Why is this discussion so important?

- Smartphones deal with lots of personal (sensitive) information.
- Anyone with basic programming skills is able to develop apps.
  - The Internet is full of tutorials.
  - Building your own app became easier (in the sense you do not need much technical background to make something usable, cool and fun!).
  - Easier ways to deploy apps (Google Play, Apple App Store, Nokia Ovi Store, etc).
ECB - *Electronic Codebook Mode*

**Figure**: Encryption and decryption in ECB mode.
CBC - *Cipher-block chaining*

**Figure:** Encryption and decryption in CBC mode.
**ECB vs CBC**

**Figure:** Comparison between images encrypted using ECB and CBC modes. (Source: [http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation](http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation))

“While the colour of each individual pixel is encrypted, the overall image may still be discerned as the pattern of identically coloured pixels in the original remains in the encrypted version.”
PBE - *Password-based Encryption*

Figure: *Password-based Encryption*. (Source: http://www.jasypt.org)
JCA - Java Cryptography Architecture

Figure: JCA overview. (Source: http://www.itcsolutions.eu)
Dalvik virtual machine

Figure: The DEX compiler converts the java .class file into a .dex file, which is of less size and more optimized for the Dalvik VM. (Source: http://markfaction.wordpress.com)
Guide

1. Introduction

2. Definitions

3. Cryptography misuse

4. Case Studies

5. Conclusions
It’s all about the programmer’s knowledge

- Cryptography allows programmers to provide basic tenets of confidentiality, integrity and authentication to the data managed by their software.

- However, implementation of cryptographic primitives by non-cryptographers almost always contain fatal flaws that render the implementation all but useless. *(Atwater, 2014)*
An example...

```java
//call the static factory method to create a DES instance
Cipher desCipher = Cipher.getInstance("DES/CBC/PKCS5Padding");

//call the static factory method to create a AES instance
Cipher aesCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

The **Cipher** class states that a block cipher mode can be requested. However, it fails to mention if no mode is requested, ECB mode will be used by default *(Egele, 2013)*.
Class Overview

This class provides access to implementations of cryptographic ciphers for encryption and decryption. Cipher classes can not be instantiated directly, one has to call the Cipher’s getInstance method with the name of a requested transformation, optionally with a provider. A transformation specifies an operation (or a set of operations) as a string in the form:

- "algorithm/mode/padding"
  or
- "algorithm"

`algorithm` is the name of a cryptographic algorithm, `mode` is the name of a feedback mode and `padding` is the name of a padding scheme. If `mode` and/or `padding` values are omitted, provider specific default values will be used.

A valid transformation would be:

```java
Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

When a block cipher is requested in in stream cipher mode, the number of bits to be processed at a time can be optionally specified by appending it to the mode name. e.g. "AES/CFB8/NoPadding". If no number is specified, a provider specific default value is used.

**Figure:** From:
public static final Cipher getInstance (String transformation)  

Creates a new Cipher for the specified transformation. The installed providers are searched in order for an implementation of the specified transformation. The first found provider providing the transformation is used to create the cipher. If no provider is found an exception is thrown.

**Parameters**

*transformation*  the name of the transformation to create a cipher for.

**Returns**

a cipher for the requested transformation.

**Throws**

*NoSuchAlgorithmException*  if no installed provider can provide the *transformation*, or it is *null*, empty or in an invalid format.

*NoSuchPaddingException*  if no installed provider can provide the padding scheme in the *transformation*.

**Figure:** From:  
CryptoLint (Egele, 2013)

- Based on Androguard (http://code.google.com/p/androguard).
- CryptoLint disassembles Android binaries in order to identify flows between cryptographic keys, initialization vectors (and similar cryptographic material) and the cryptographic operations themselves.
- While current developer tools can check a number of security properties, using cryptography correctly is not one of them. (Egele, 2013)
Common rules in cryptography

List of common rules in symmetric key cryptography, according to (Egele, 2013). Any application that violates one of the following, cannot be secure.

1. Do not use ECB mode for encryption.
2. Do not use a non-random IV for CBC encryption.
3. Do not use constant encryption keys.
4. Do not use constant salts for PBE.
5. Do not use fewer than 1000 iterations for PBE.
6. Do not use static seeds to seed SecureRandom().
(Egele, 2013) evaluated 145,095 apps from the official Google Play marketplace.

- 15,134 (about 10.4%) use cryptographic functionality.
  - 11,748 (77.6%) apps were successfully analyzed.
- Only 1,421 (about 12.1%) did not violate any rules.
## Most violated rules

<table>
<thead>
<tr>
<th># Apps</th>
<th>Violated rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,656</td>
<td>Uses ECB mode (BouncyCastle default) (R1)</td>
</tr>
<tr>
<td>3,644</td>
<td>Uses constant symmetric key (R3)</td>
</tr>
<tr>
<td>2,000</td>
<td>Uses ECB (Explicit use) (R1)</td>
</tr>
<tr>
<td>1,932</td>
<td>Uses constant IV (R2)</td>
</tr>
<tr>
<td>1,636</td>
<td>Uses less than 1000 iterations for PBE (R5)</td>
</tr>
<tr>
<td>1,629</td>
<td>Seeds SecureRandom with static (R6)</td>
</tr>
<tr>
<td>1,574</td>
<td>Uses static salt for PBE (R4)</td>
</tr>
<tr>
<td>1,421</td>
<td>No violation</td>
</tr>
</tbody>
</table>

*Table: Violations of cryptographic security rules (Egele, 2013)*
## Apps violating two rules

<table>
<thead>
<tr>
<th># Apps</th>
<th>Rules violated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,905</td>
<td>Rule 1 &amp; Rule 3</td>
</tr>
<tr>
<td>1,588</td>
<td>Rule 1 &amp; Rule 6</td>
</tr>
<tr>
<td>1,247</td>
<td>Rule 4 &amp; Rule 5</td>
</tr>
<tr>
<td>866</td>
<td>Rule 2 &amp; Rule 3</td>
</tr>
<tr>
<td>109</td>
<td>Rule 1 &amp; Rule 2</td>
</tr>
<tr>
<td>24</td>
<td>Rule 1 &amp; Rule 5</td>
</tr>
<tr>
<td>11</td>
<td>Rule 3 &amp; Rule 5</td>
</tr>
<tr>
<td>5</td>
<td>Rule 2 &amp; Rule 5</td>
</tr>
<tr>
<td>2</td>
<td>Rule 1 &amp; Rule 4</td>
</tr>
<tr>
<td>2</td>
<td>Rule 3 &amp; Rule 4</td>
</tr>
</tbody>
</table>

**Table:** Applications violating two rules *(Egele, 2013)*
### Frequently used symmetric encryption schemes

<table>
<thead>
<tr>
<th># Occurrences</th>
<th>Symmetric encryption scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,878</td>
<td>AES/CBC/PKCS5Padding</td>
</tr>
<tr>
<td>4,803</td>
<td>AES*</td>
</tr>
<tr>
<td>1,151</td>
<td>DES/ECB/NoPadding</td>
</tr>
<tr>
<td>741</td>
<td>DES*</td>
</tr>
<tr>
<td>501</td>
<td>DESede*</td>
</tr>
<tr>
<td>473</td>
<td>DESede/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>468</td>
<td>AES/CBC/NoPadding</td>
</tr>
<tr>
<td>443</td>
<td>AES/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>235</td>
<td>AES/CBC/PKCS7Padding</td>
</tr>
<tr>
<td>221</td>
<td>DES/ECB/PKCS5Padding</td>
</tr>
<tr>
<td>220</td>
<td>AES/ECB/NoPadding</td>
</tr>
<tr>
<td>205</td>
<td>DES/CBC/PKCS5Padding</td>
</tr>
<tr>
<td>155</td>
<td>AES/ECB/PKCS7Padding</td>
</tr>
<tr>
<td>104</td>
<td>AES/CFB8/NoPadding</td>
</tr>
</tbody>
</table>

**Table:** Distribution of frequently used symmetric encryption schemes. Schemes marked with * are used in ECB mode by default *(Egele, 2013)*
Some analyzed applications

- **Social gaming platform** (50M ~ 100M installations)
  - Sends/receives data to a server using DES/ECB (explicitly).
  - Static encryption key.

- **Bookmark manager** (1M ~ 5M installations)
  - Stores user’s Google credential password in the property file, using DES/ECB.
  - Static encryption key.

- **Password management application** (100K ~ 500K installations)
  - Open source app with a publicly available GIT repository.
    - Version 1: AES/ECB, but uses 2 bytes of random data in the password.
    - Version 2: AES/CBC, but uses a static IV.
    - Version 3: AES/CBC, with random generated IV using the regular number generator instead of SecureRandom().
Popular libraries

- **AdMob** (one of the most popular advertising libraries included in Android apps)
  - Uses AES/CBC to encrypt location and identifiers. ✓
  - Uses `SecureRandom()` to generate a random IV. ✓
  - Uses a constant encryption key to do that. ✗

- **Scoreloop** (provides functionality to integrate social capabilities to mobile apps)
  - Uses AES/CBC. ✓
  - Uses constant IV. ✗
Guide

1. Introduction
2. Definitions
3. Cryptography misuse
4. Case Studies
5. Conclusions
Conclusions

- 88% of Android apps that make use of cryptographic functionality have some issue.
- Undocumented insecure default configuration is a contributing factor.
- Programmers with no background in cryptography will not use cryptographic functionality correctly.
References


Any questions?

Thanks!

http://www.ime.usp.br/~rwill