Maximilian Blochberger

How to prevent cryptographic pitfalls by design
How to prevent cryptographic pitfalls by design

**Goal**
Raise awareness of cryptographic misuse

**Disclaimer**
Project pitch: iOS & macOS framework
DON’T PANIC!

**Scenario**
Developer that values privacy intents to add encryption
**Task:** Encrypt a string

Android, Java Cryptographic Extensions (JCE), Bouncy Castle
Solution

android encryption/decryption with AES - Stack Overflow

Encryption on Android is not fundamentally different than on any other Java SE platform. And as all the answers below are insecure, for either you have to understand cryptography before you start implementing or borrowing cryptography examples.


Android Encryption Example - GitHub

Android Encryption Example. This example encrypts the inputted string using AES, encrypts the key via RSA, and does the reverse when the decrypt button is clicked.

[https://github.com/brianPlummer/AndroidEncryptionExample](https://github.com/brianPlummer/AndroidEncryptionExample)

encryption - Easy way to Encrypt/Decrypt string in Android ...

Easy way to Encrypt/Decrypt string in Android. Ask Question 13. 13. ... Using these helper class you can encrypt and decrypt string in android simple way,


Android Encryption with the Android Cryptography API ...

If you are up for the simple off-the-shelf encryption provided by Android Cryptography APIs, then this introductory tutorial will show you where to find the resources, how to check if some algorithms are supported on your devices programmatically, and provide examples of a couple of popular algorithms in AES and RSA.


android encryption/decryption with AES

Warning: This answer contains code you should not use as it is insecure (using SHA1PRNG for key derivation and using AES in ECB mode). Instead, use PBKDF2WithHmacSHA1 for key derivation and AES in CBC or GCM mode (GCM provides both privacy and integrity).

You could use functions like these:

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.DECRYPT_MODE, skeySpec);
    byte[] decrypted = cipher.doFinal(encrypted);
    return decrypted;
}
```
Warning: This answer contains code you should not use as it is insecure (using SHA1PRNG for key derivation and using AES in ECB mode)

Instead, use PBKDF2WithHmacSHA1 for key derivation and AES in CBC or GCM mode (GCM provides both privacy and integrity)

You could use functions like these:

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.DECRYPT_MODE, skeySpec);
    byte[] decrypted = cipher.doFinal(encrypted);
    return decrypted;
}
```

And invoke them like this:

```java
ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bm.compress(Bitmap.CompressFormat.PNG, 100, baos); // bm is the
    byte[] b = baos.toByteArray();
```
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();

byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);

Code taken from https://stackoverflow.com/a/6788456/5082444
What could possibly go wrong?

private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();

byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);

Code taken from https://stackoverflow.com/a/6788456/5082444
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();

byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);

Code taken from https://stackoverflow.com/a/6788456/5082444
What could possibly go wrong?

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}
```

```java
byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();
```

```java
byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);
```

Insecure defaults

"AES/ECB/PKCS5PADDING"

Code taken from https://stackoverflow.com/a/6788456/5082444
What could possibly go wrong?

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}
```

byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr);
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();
byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);

Insecure defaults

"AES/ECB/PKCS5Padding"

Code taken from https://stackoverflow.com/a/6788456/5082444
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();

byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);

Code taken from https://stackoverflow.com/a/6788456/5082444
What could possibly go wrong?

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}
```

```java
byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();
```

```java
byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);
```

Outdated algorithms

Code taken from https://stackoverflow.com/a/6788456/5082444

February 3, 2019 | Maximilian Blochberger
What could possibly go wrong?

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}
```

Insecure key derivation

```java
byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();
```

```java
byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);
```

Code taken from https://stackoverflow.com/a/6788456/5082444

February 3, 2019 | Maximilian Blochberger
What could possibly go wrong?

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}
```

```java
byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();
```

```java
byte[] encryptedData = encrypt(key, b); // Not IND-CPA secure
byte[] decryptedData = decrypt(key, encryptedData);
```

Code taken from https://stackoverflow.com/a/6788456/5082444
What could possibly go wrong?

```java
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();

byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);
```

Not authenticated

---

Code taken from https://stackoverflow.com/a/6788456/5082444
Problem

- **98%** security-related snippets are insecure
  
  Fischer et al., 2017; Nadi et al., 2016; Das et al., 2014

- **Hard to get right**
  
  Nadi et al., 2016; Egele et al., 2013; …

- **Alternative APIs**
  
  - OpenSSL
  - Botan
  - Crypto++
  - NaCl / Libsodium
    
    Bernstein, Lange, and Schwabe, 2012
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

byte[] keyStart = "this is a key".getBytes();
KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(keyStart);
kgen.init(128, sr); // 192 and 256 bits may not be available
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();

byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);
private static byte[] encrypt(AesKey key, byte[] clear) throws Exception {

    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, key);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}

AesKey key = AesKey.deriveFrom("this is a key");

- Type-safe
- Implementation details hidden

byte[] encryptedData = encrypt(key, b);
byte[] decryptedData = decrypt(key, encryptedData);
import Tafelsalz

let password = Password("this is a key")!
let box = SecretBox(deriveKeyFrom: password)

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!

- Open-source framework
- iOS & macOS
- Swift
- Based on Libsodium
- License: ISC/MIT
import Tafelsalz

let password = Password("this is a key")!
let box = SecretBox(deriveKeyFrom: password)

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!

Secure memory
import Tafelsalz

let password = Password("this is a key")!
let box = SecretBox(deriveKeyFrom: password)

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!
import Tafelsalz

let password = Password("this is a key")!
let box = SecretBox(deriveKeyFrom: password)

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!

Fails if ciphertext has been tampered with
import Tafelsalz

let password = Password("this is a key")
let box = SecretBox(deriveKeyFrom: password)

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!

Problem

Key persistence is hard
Huber, Rasthofer, and Arzt, 2017
Utilizing Platform Capabilities

```swift
import Tafelsalz

let key = SecretBox.SecretKey()
let box = SecretBox(secretKey: key)

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!
```
Utilizing Platform Capabilities

```swift
import Tafelsalz

let key = SecretBox.SecretKey()
let box = SecretBox(secretKey: key)

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!
```

**Task:** Persist key
Utilizing Platform Capabilities

```swift
import Tafelsalz

let alice = Persona(uniqueName: "Alice")
let box = SecretBox(persona: alice)!

let encrypted = box.encrypt(plaintext: b)
let decrypted = box.decrypt(ciphertext: encrypted)!
```

Local identity management
- Named key (per app)
- Stored in Keychain (TPM-secured)
Cryptography is harder than it looks — Schneier, 2016

- Many things can go wrong
- Many things do go wrong
- StackOverflow, examples, documentation, ...

Tafelsalz

- Open-source framework for iOS & macOS
- Simple misuse-resistant API
- Supports platform capabilities

https://blochberger.github.io/Tafelsalz
Hands on

DCrypt

1. Check out project
2. Implement encryption & decryption
3. Implement unit tests
4. Does en-/decryption after relaunch still work?
5. Share encrypted files with others

https://github.com/AppPETs/DCrypt
Hands on

DCrypt

1. Check out project
2. Implement encryption & decryption
   → Symmetric encryption
3. Implement unit tests
4. Does en-/decryption after relaunch still work?
   → Credential storage
5. Share encrypted files with others
   → Password-based key derivation

https://github.com/AppPETs/DCrypt
References


