Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

DEPARTMENT OF INFORMATICS
SECURITY AND PRIVACY

Maximilian Blochberger

How to prevent cryptographic pitfalls by design

February 3, 2019

How to prevent cryptographic pitfalls by design

Goal

Raise awareness of cryptographic misuse

Disclaimer

Project pitch: iOS & macOS framework

DON'T PANIC!

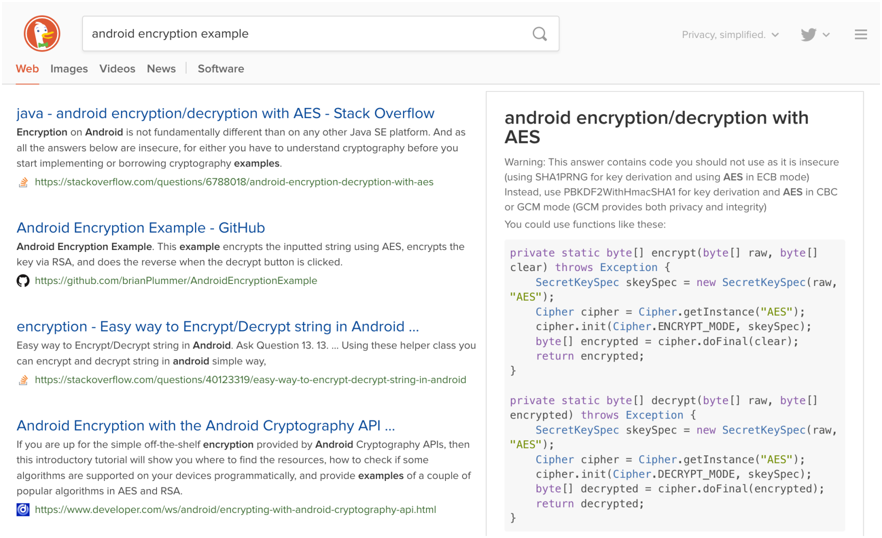
Scenario

Developer that values privacy intents to add encryption

Task: Encrypt a string

Android, Java Cryptographic Extensions (JCE), Bouncy Castle

Solution



java - 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**.
https://stackoverflow.com/questions/6788018/android-encryption-decryption-with-aes

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

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,
https://stackoverflow.com/questions/40123319/easy-way-to-encrypt-decrypt-string-in-android

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.
https://www.developer.com/ws/android/encrypting-with-android-cryptography-api.html

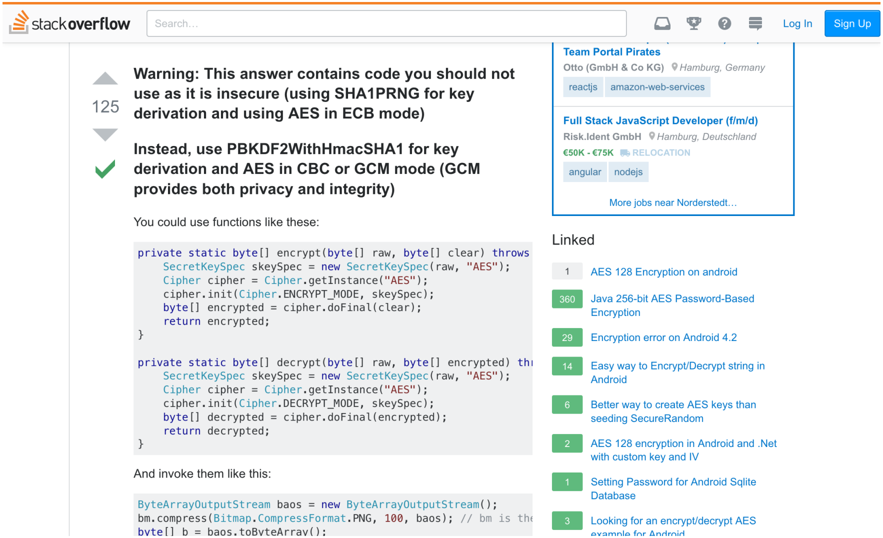
android encryption/decryption with AES

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:

```
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;
}
```

Solution



Team Portal Pirates
Otto (GmbH & Co KG) Hamburg, Germany
reactjs amazon-web-services

Full Stack JavaScript Developer (f/m/d)
Risk.Ident GmbH Hamburg, Deutschland
€50K - €75K RELOCATION
angular nodejs

More jobs near Norderstedt…

125

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:

```
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) thr
    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:

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
bm.compress(Bitmap.CompressFormat.PNG, 100, baos); // bm is the
byte[] b = baos.toByteArray();
```

Linked

1	AES 128 Encryption on android
360	Java 256-bit AES Password-Based Encryption
29	Encryption error on Android 4.2
14	Easy way to Encrypt/Decrypt string in Android
6	Better way to create AES keys than seeding SecureRandom
2	AES 128 encryption in Android and .Net with custom key and IV
1	Setting Password for Android Sqlite Database
3	Looking for an encrypt/decrypt AES example for Android

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`

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);
```

Typing?

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);
```

Obscure choices

”AES”, ”DES”, ”RSA”, ”RC2”, ...

¯_(ツ)_/¯

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);
```

Insecure defaults

”AES/ECB/PKCS5PADDING”

¯_(ツ)_/¯

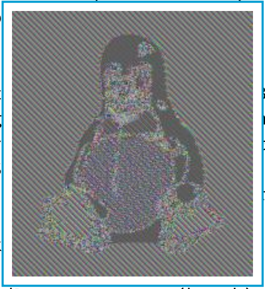
Code taken from `https://stackoverflow.com/a/6788456/5082444`

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 encryp
}

byte[] keyStart                    ytes();
KeyGenerator kg                    nstance(”AES”);
SecureRandom sr                    ance(”SHA1PRNG”);
sr.setSeed(keyS
kgen.init(128,                     s may not be available
SecretKey skey
byte[] key = sk

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



Insecure defaults

”AES/**ECB**/PKCS5PADDING”

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

`https://commons.wikimedia.org/w/index.php?title=File:Tux_ecb.jpg&oldid=109528640`

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
SecretKey skey = kgen.generateKey();
byte[] key = skey.getEncoded();

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

Static parameters

Keys, Nonces/IVs, Seeds, Passwords, ...

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);
```

Outdated algorithms

SHA1, MD5, DES, ...

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;
}
```

Insecure key derivation

```
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);
```

$\cancel{IIII}\ I$

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);
```

Not IND-CPA secure

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

𝍸 III

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

Repairing

```
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);
```

Repairing

```
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);
```

Tafelsalz

```
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

Tafelsalz

```
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

Tafelsalz

```
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)!
```

Type-safe

Compiler vs. runtime checks

Tafelsalz

```
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

Tafelsalz

```
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)!
```

Still static

Problem

Key persistence is hard

Huber, Rasthofer, and Arzt, 2017

Utilizing Platform Capabilities

```
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

```
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

```
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)

Summary

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

Felix Fischer et al. “Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security.” In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2017, pp. 121–136.

Sarah Nadi et al. “Jumping through hoops: why do Java developers struggle with cryptography APIs?” In: *ICSE*. ACM, 2016, pp. 935–946.

Somak Das et al. *IV=0 Security: Cryptographic Misuse of Libraries*. Tech. rep. 2014.

Manuel Egele et al. “An empirical study of cryptographic misuse in android applications.” In: *ACM Conference on Computer and Communications Security*. ACM, 2013, pp. 73–84.

Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. “The Security Impact of a New Cryptographic Library.” In: *LATINCRYPT*. Vol. 7533. Lecture Notes in Computer Science. Springer, 2012, pp. 159–176.

Stephan Huber, Siegfried Rasthofer, and Steven Arzt. *Extracting All Your Secrets: Vulnerabilities in Android Password Managers*. 2017. URL: `http://sit4.me/pw-manager`.

Bruce Schneier. “Cryptography Is Harder than It Looks.” In: *IEEE Security & Privacy* 14.1 (2016), pp. 87–88.