Code obfuscation techniques

Eric Lafortune

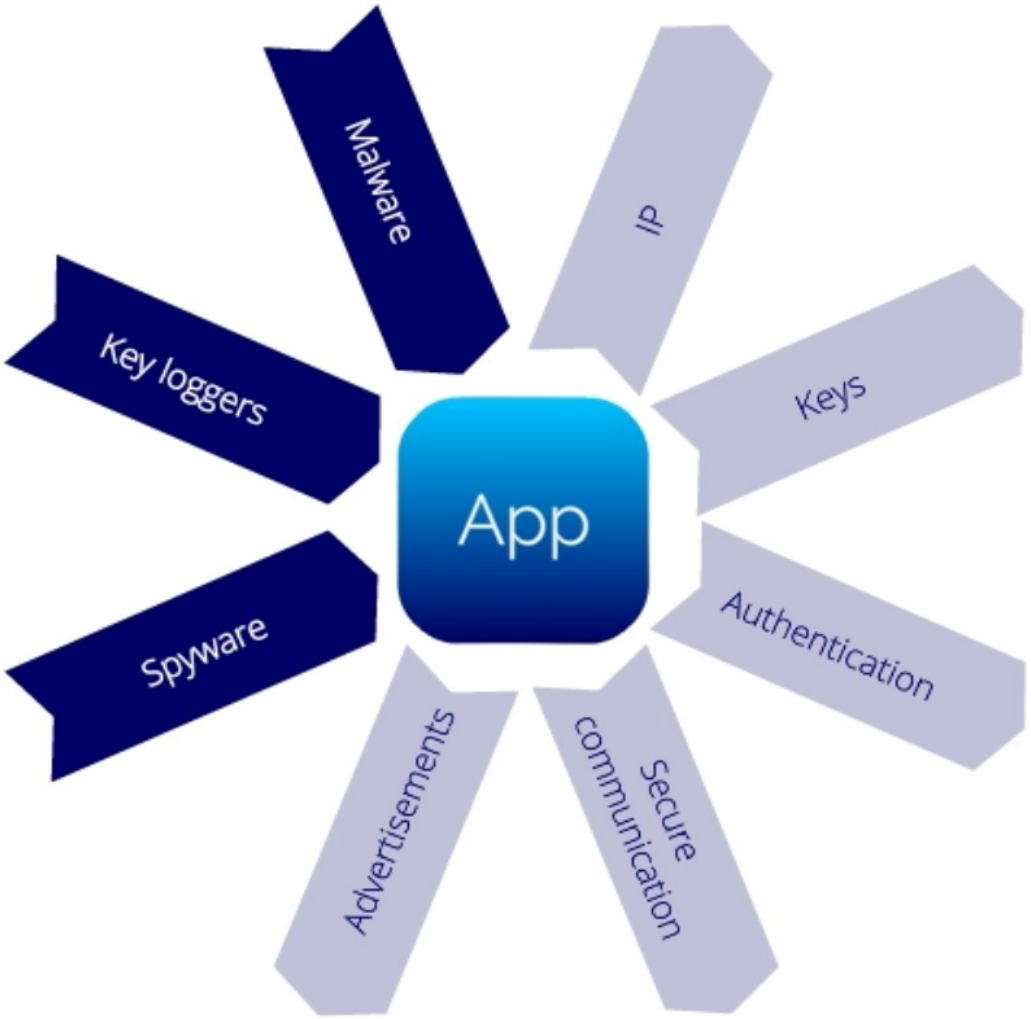
CTO at GuardSquare

Creator of ProGuard and DexGuard

Topics

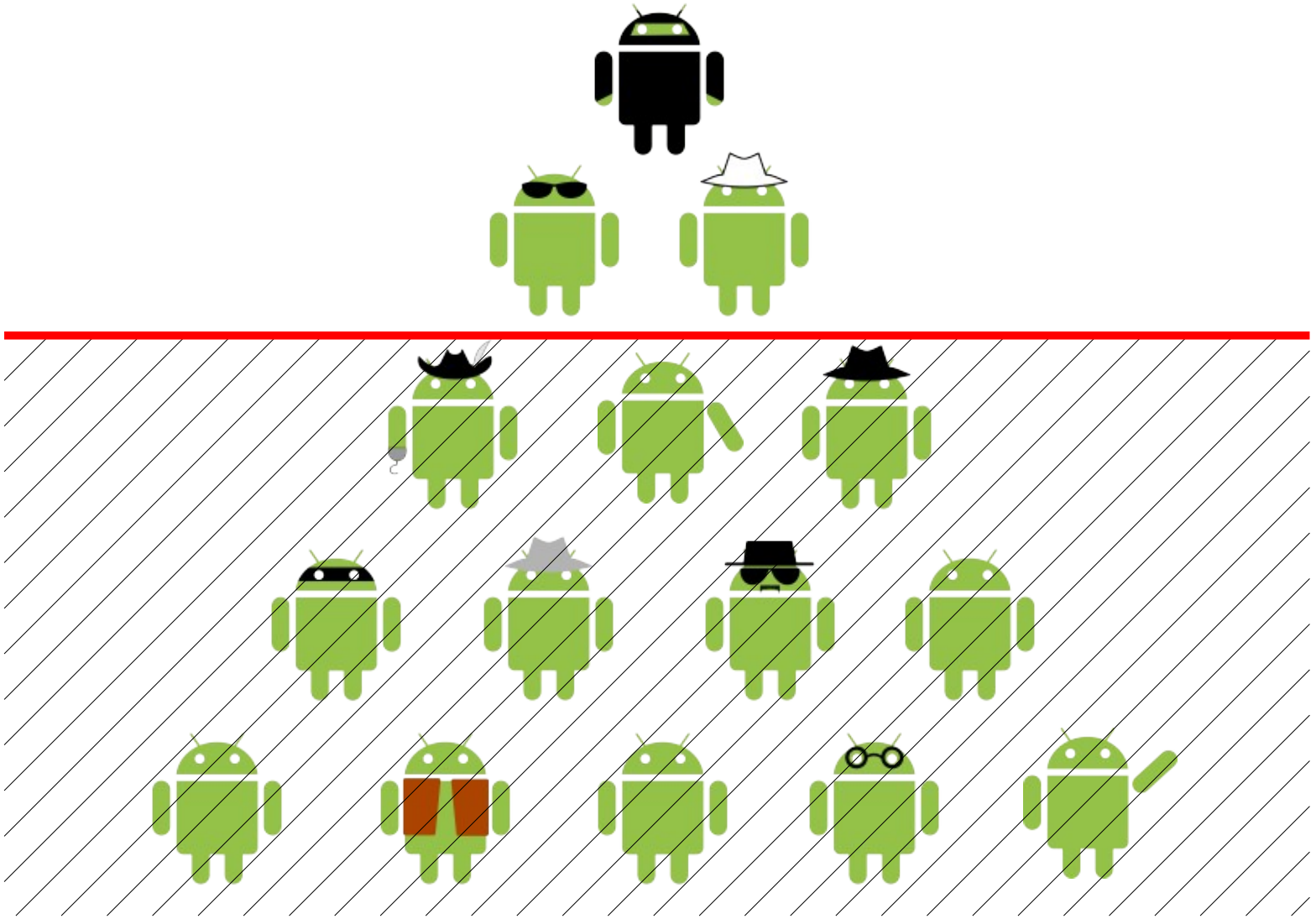
- Threats, attacks, and techniques
- Android applications and build process
- Obfuscation techniques

Threats

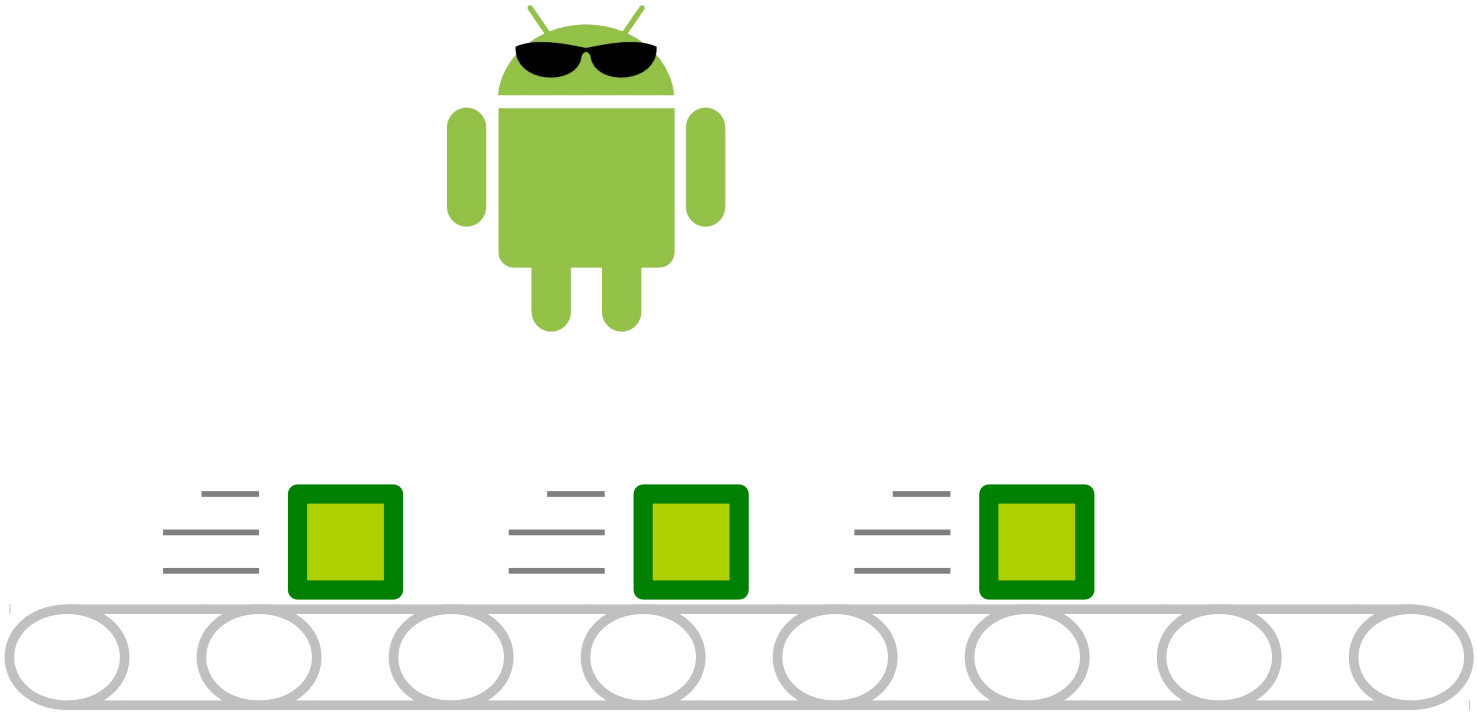

Malware
IP
Key loggers
Keys
App
Spyware
Authentication
Advertisements
Secure communication

Attackers

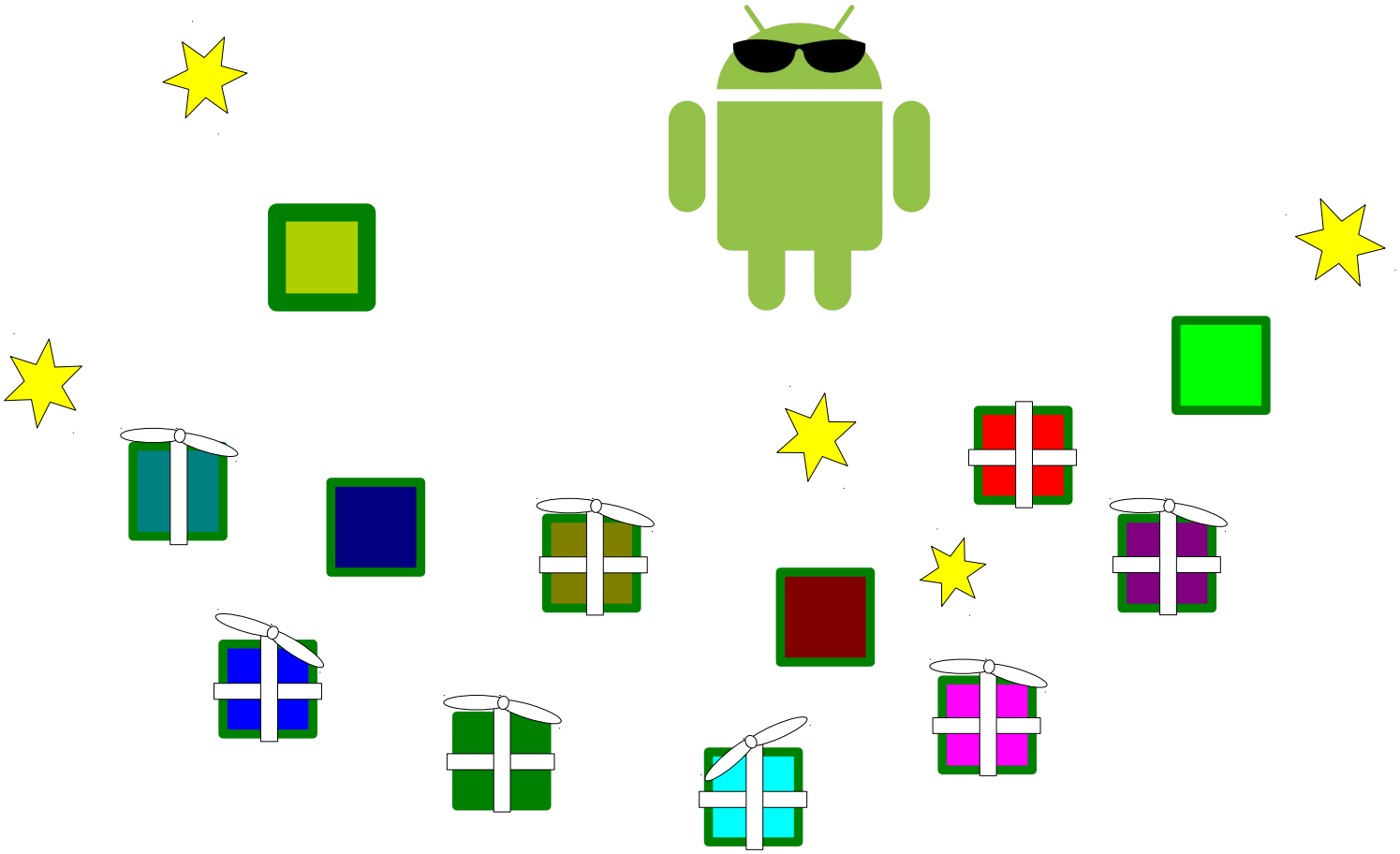
Attackers



Time

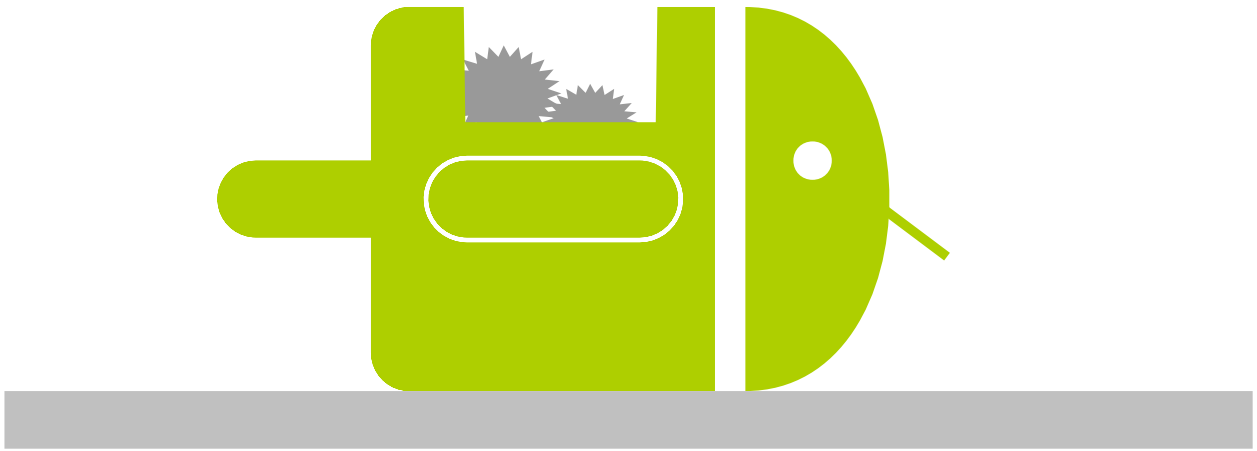


Choice



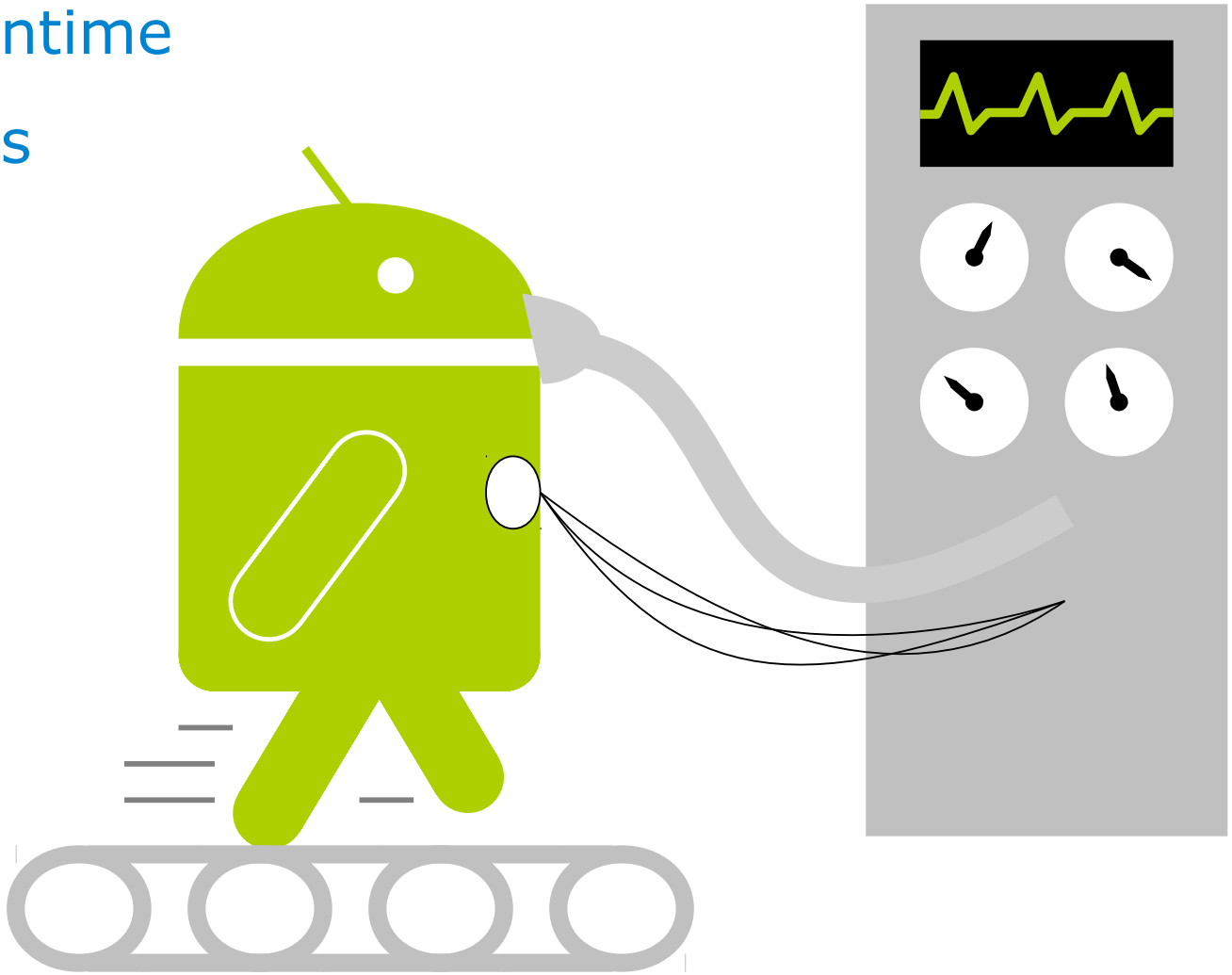
Static analysis

- Disassemblers: dexdump, baksmali
- Decompilers: dex2jar + jad, JD-GUI, JEB,...
- Resources: aapt, apktool,...



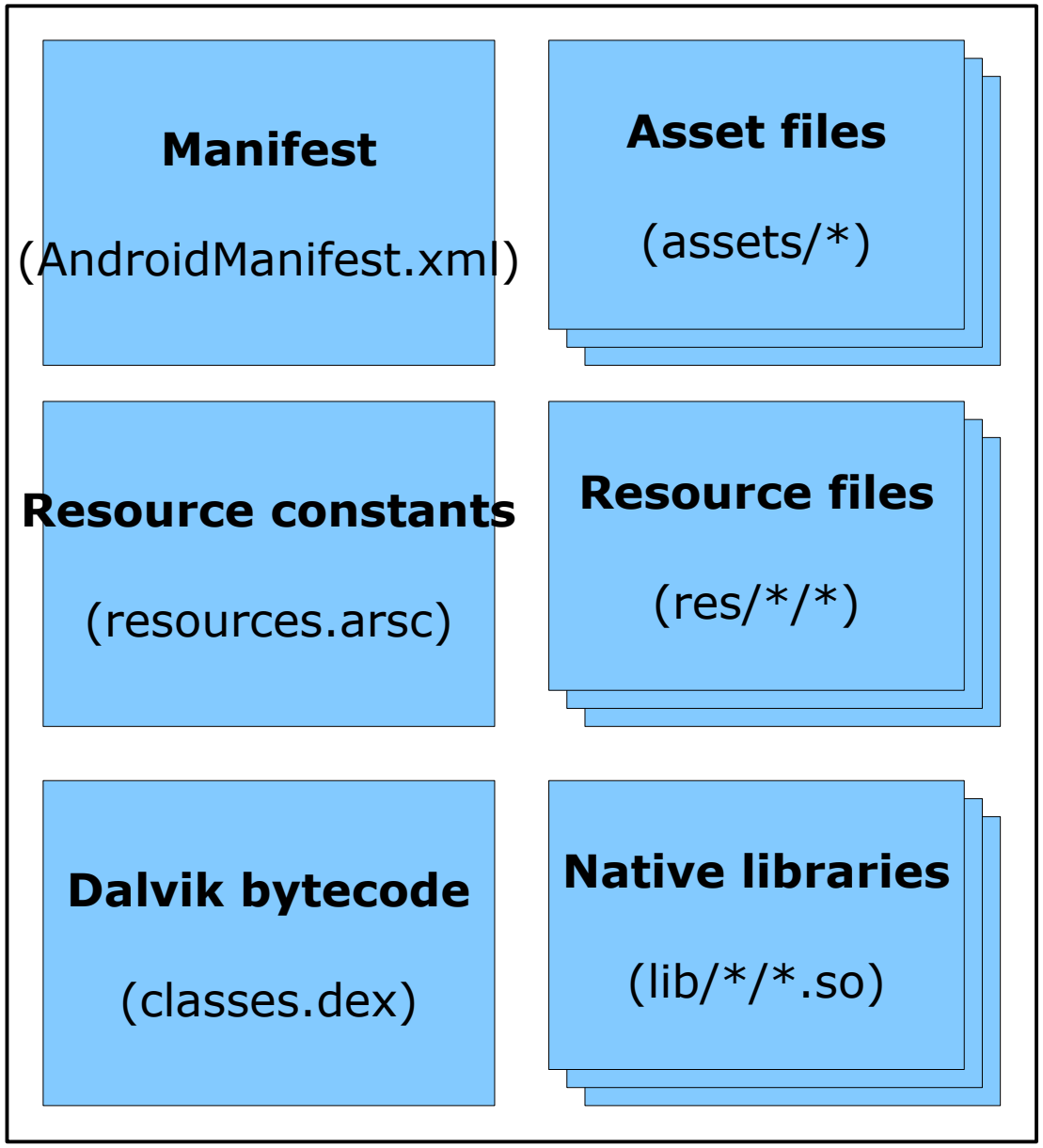
Dynamic analysis

- Debuggers: adb,...
- Subverted runtime
- Cracking tools

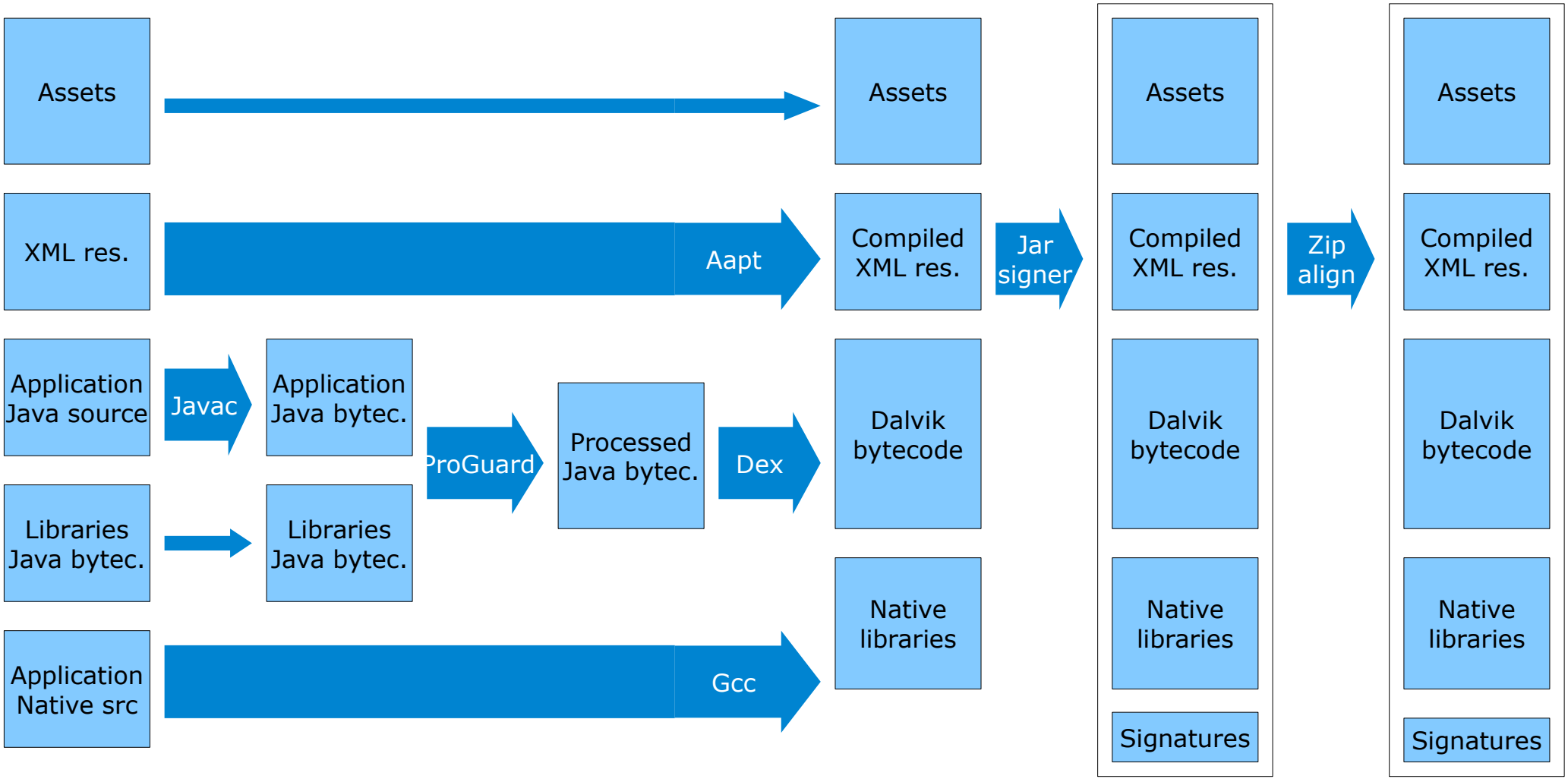


Android application

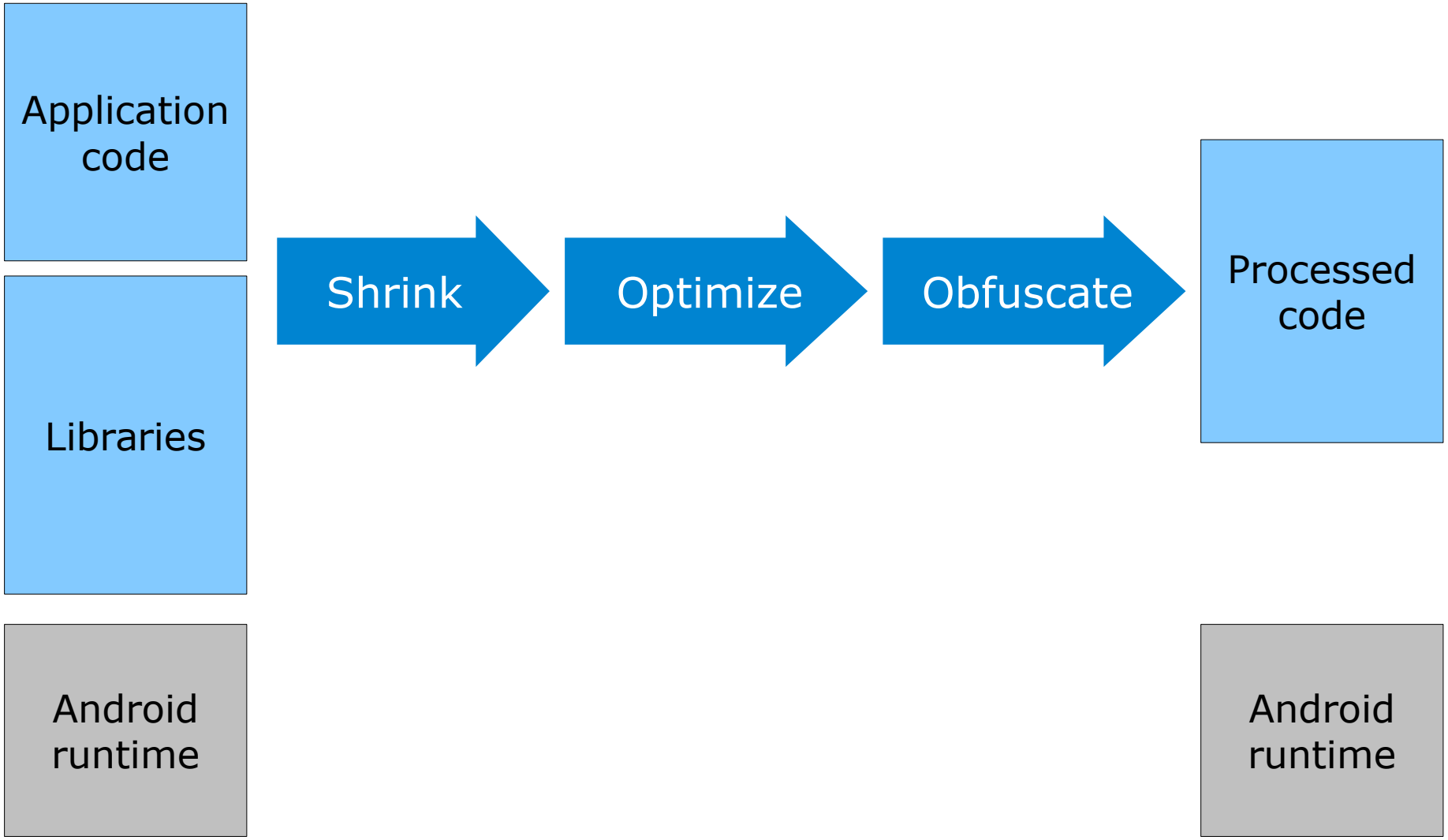
*.apk



Android build process

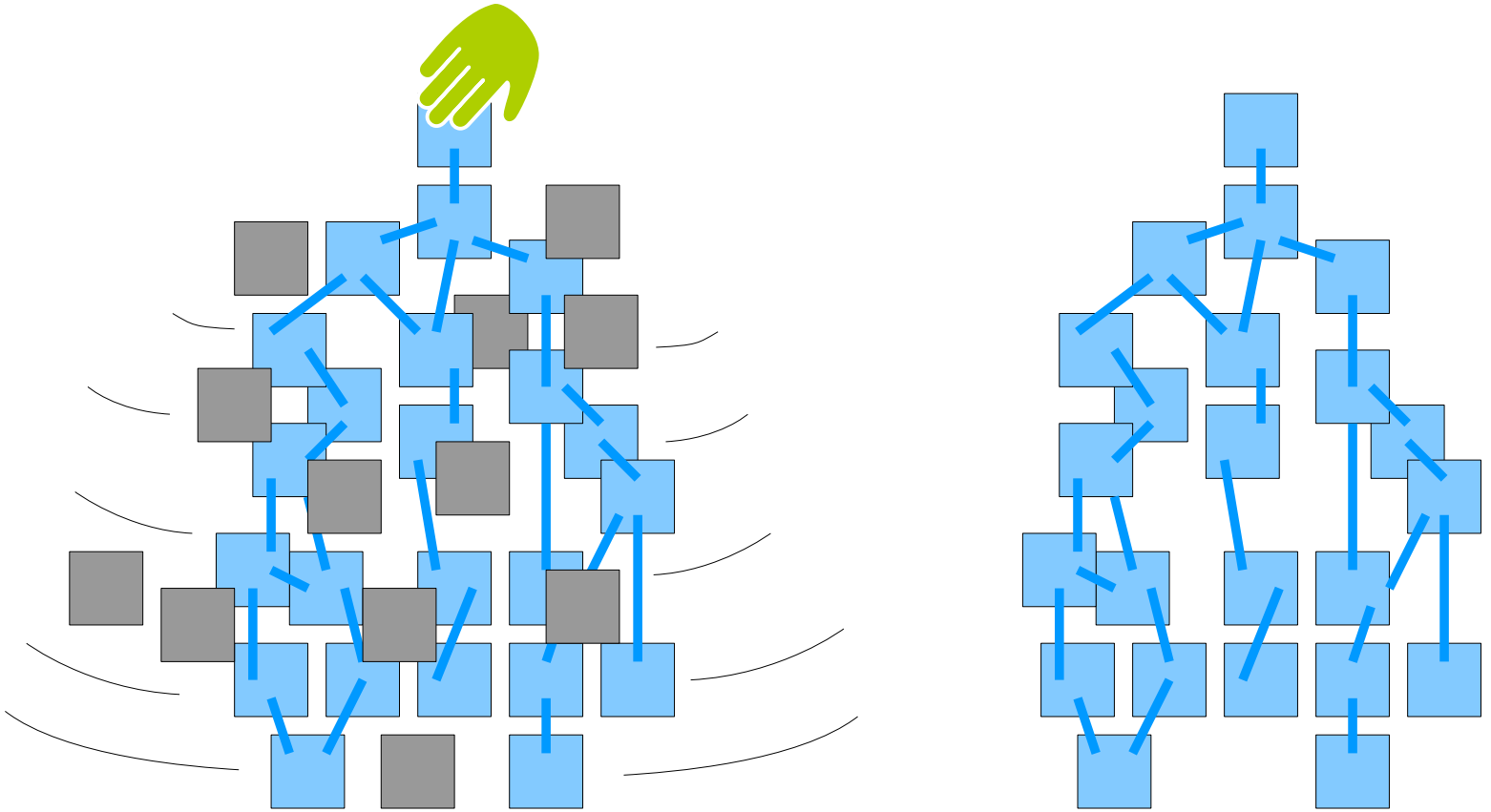


ProGuard


Application code
Libraries
Android runtime
Shrink
Optimize
Obfuscate
Processed code
Android runtime

Shrinking

- Classes, fields, methods



Optimization

At the bytecode instruction level:

- Dead code elimination
- Constant propagation
- Method inlining
- Class merging
- Remove logging code
- Peephole optimizations
- Devirtualization
- ...

Obfuscation

Traditional name obfuscation:

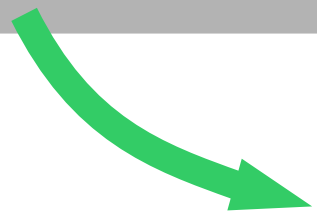
- Rename identifiers:
  class/field/method names
- Remove debug information:
  line numbers, local variable names,...

Name obfuscation

```
public class MyComputationClass {
    private MySettings  settings;
    private MyAlgorithm algorithm;
    private int         answer;

    public int computeAnswer(int input) {
        …
        return answer;
    }
}
```

Name obfuscation

```
public class MyComputationClass {
    private MySettings  settings;
    private MyAlgorithm algorithm;
    private int         answer;

    public int computeAnswer(int input) {
        …
        return answer;
    }
}
```



```
public class a {
    private b   a;
    private c   b;
    private int c;

    public int a(int a) {
        …
        return c;
    }
}
```

Some limitations

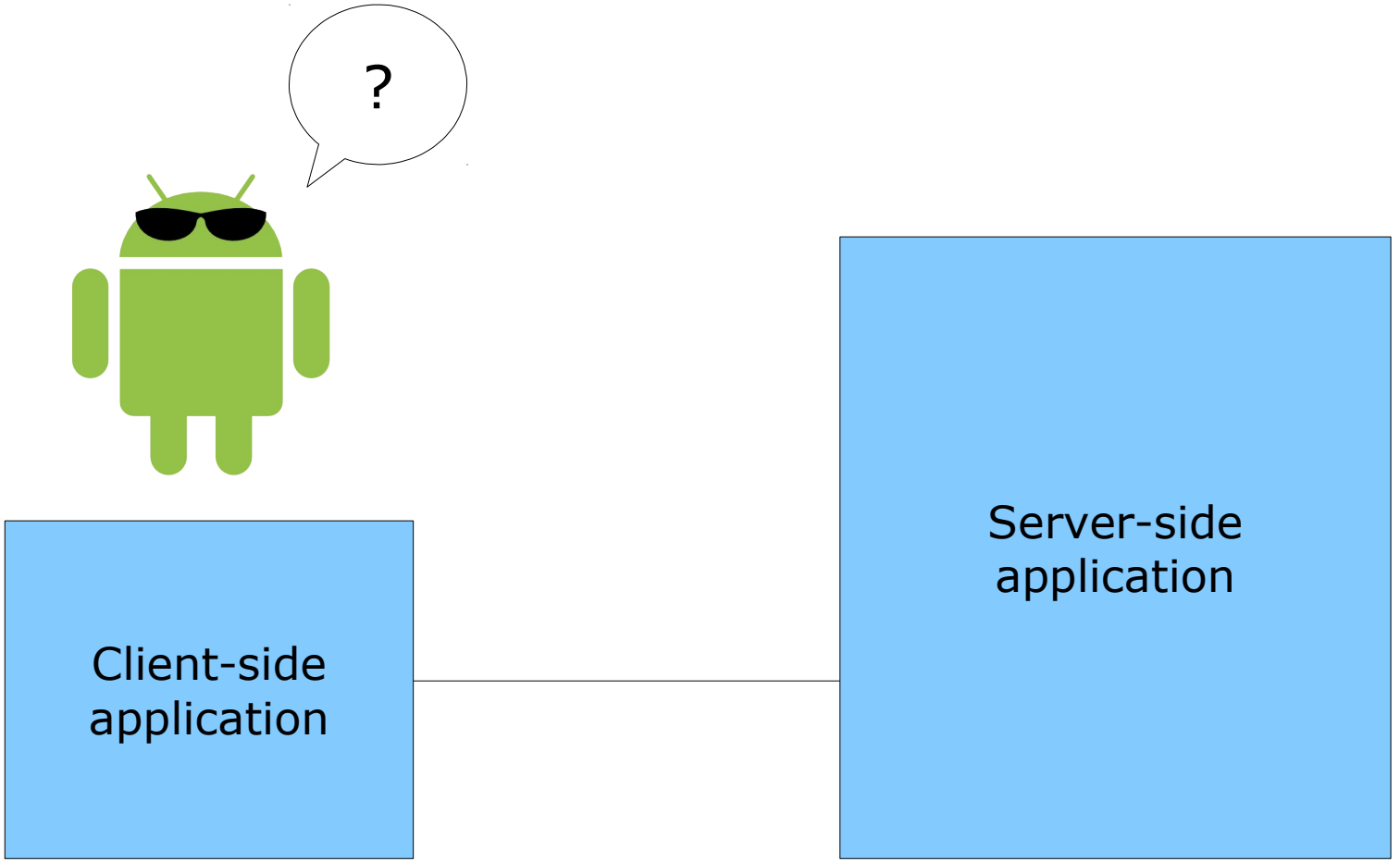
```
public class MyVerificationClass {
    public int checkSignatures() {
        …
        return activity
            .getPackageManager()
            .checkSignatures(“mypackage1”, “mypackage2”);
    }
}
```

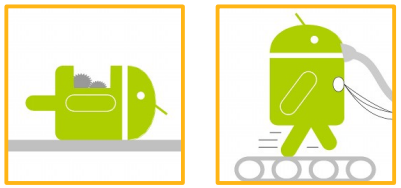
Some limitations

```
public class MyVerificationClass {
    public int checkSignatures() {
        …
        return activity
            .getPackageManager()
            .checkSignatures(“mypackage1”, “mypackage2”);
    }
}
```

```
public class a {
    public int a() {
        …
        return a
            .getPackageManager()
            .checkSignatures(“mypackage1”, “mypackage2”);
    }
}
```

Server-side code





String encryption

```
String KEY = “Secret key”;
```

```
String KEY = new String(Base64.decode(“U2VjcmV0IGtleQo=”));
```

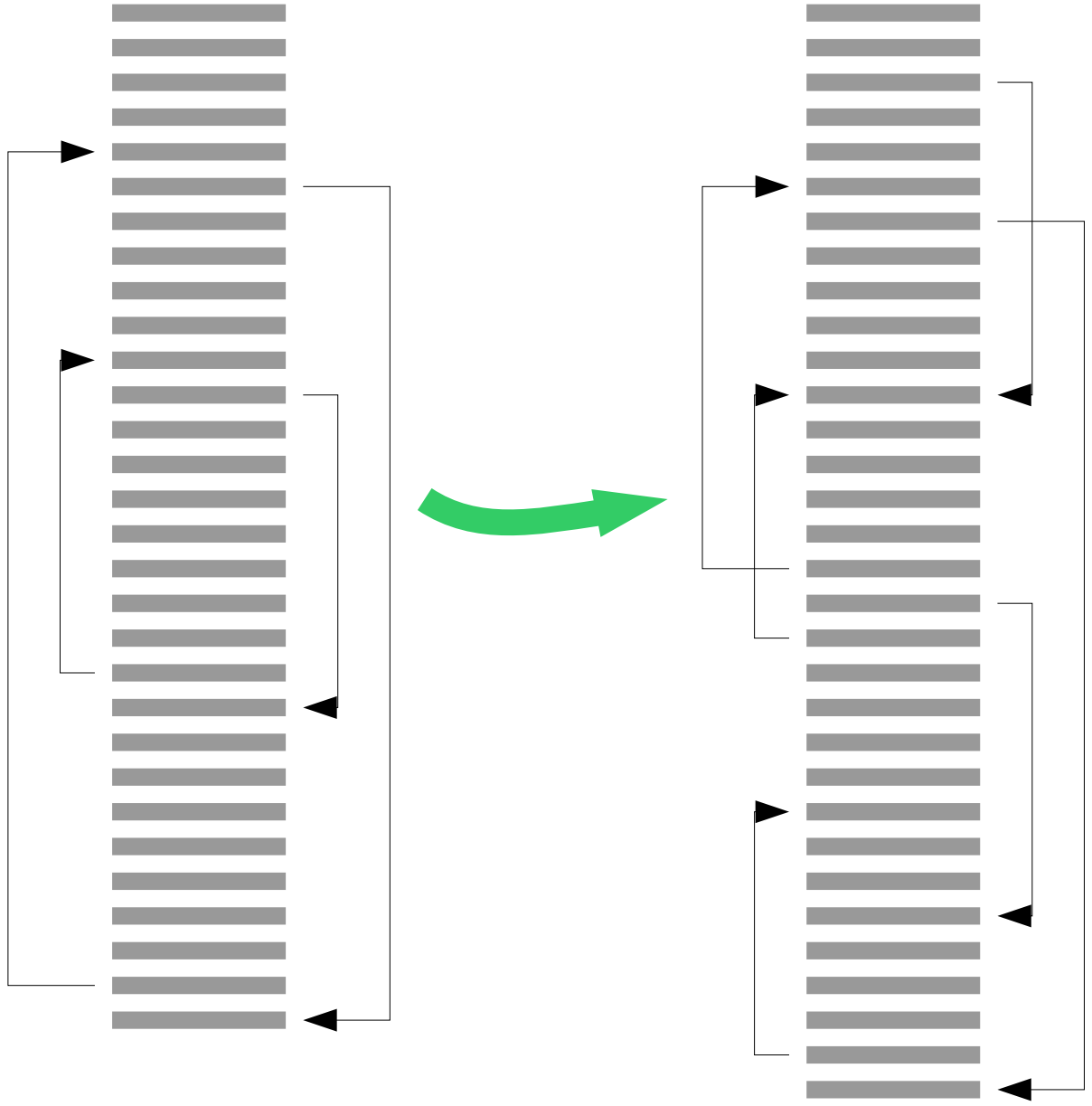
Reflection

java.lang.reflect

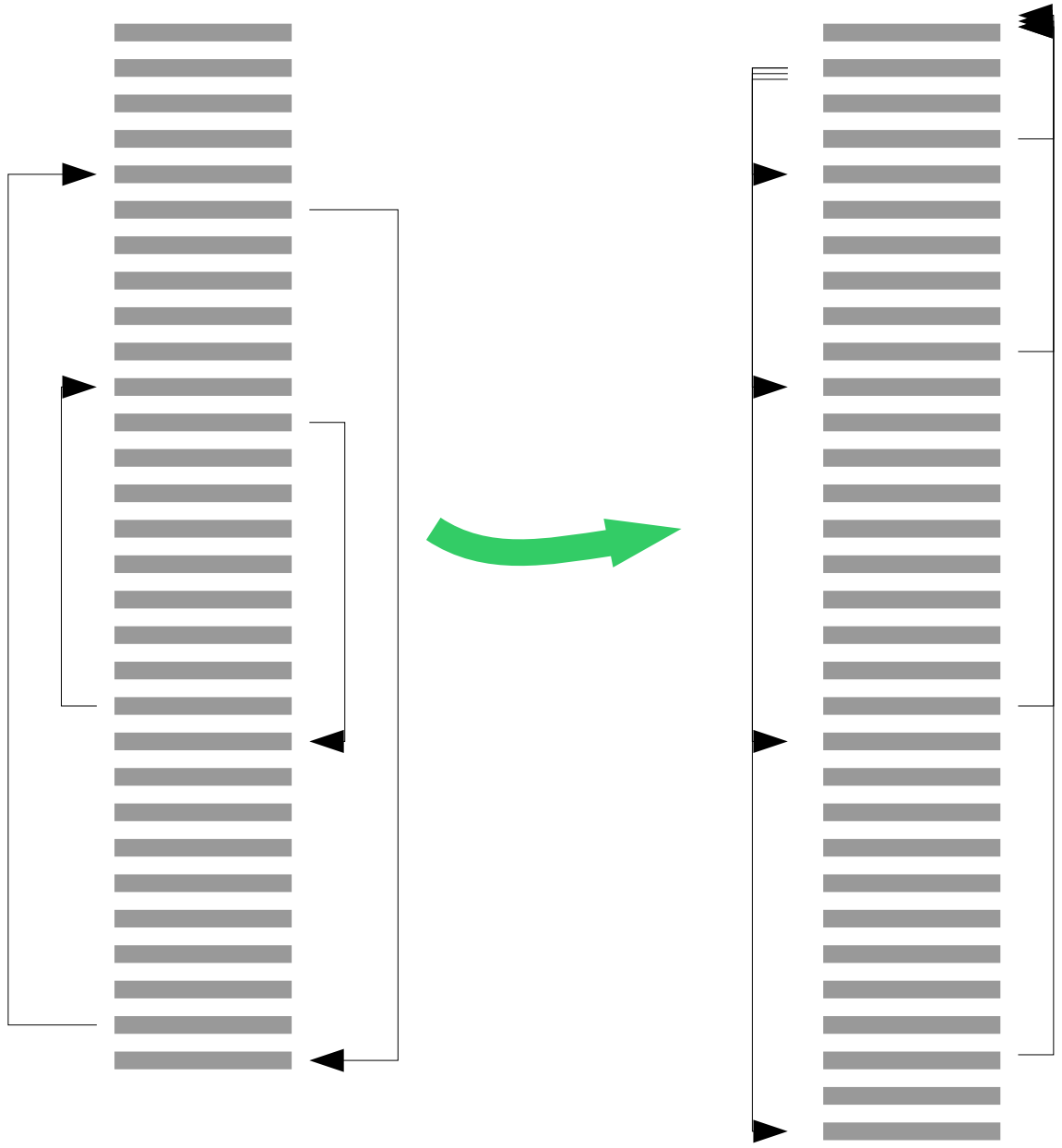
```
System.out.println("Hello world!");
```

```
Class clazz =
    Class.forName("java.io.PrintStream");

Method method =
    clazz.getMethod("println",
                    new Class[] { String.class });

method.invoke(nul, new Object[] { "Hello world!" });
```

Control flow obfuscation



Control flow flattening



Opaque predicates

```
boolean flag = true;
```

```
boolean flag = System.currentTimeMillis() > 0;
```

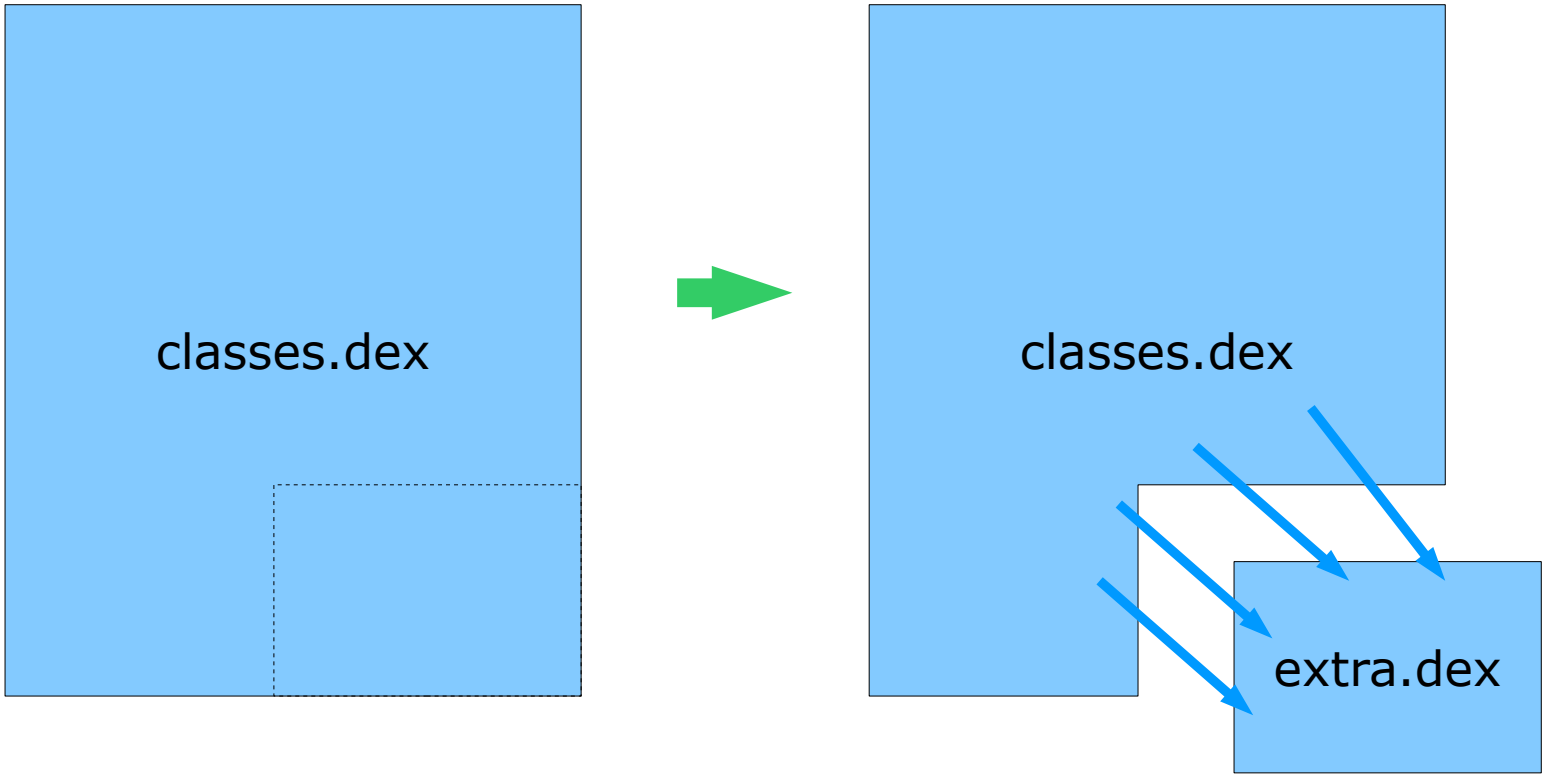
Arithmetic obfuscation

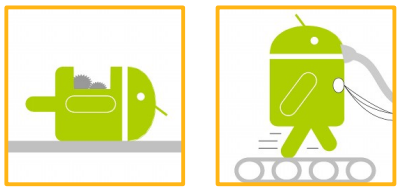
```
int c = a + b;
```

```
int c = (a ^ b) + 2 * (a & b);
```

Dynamic class loading

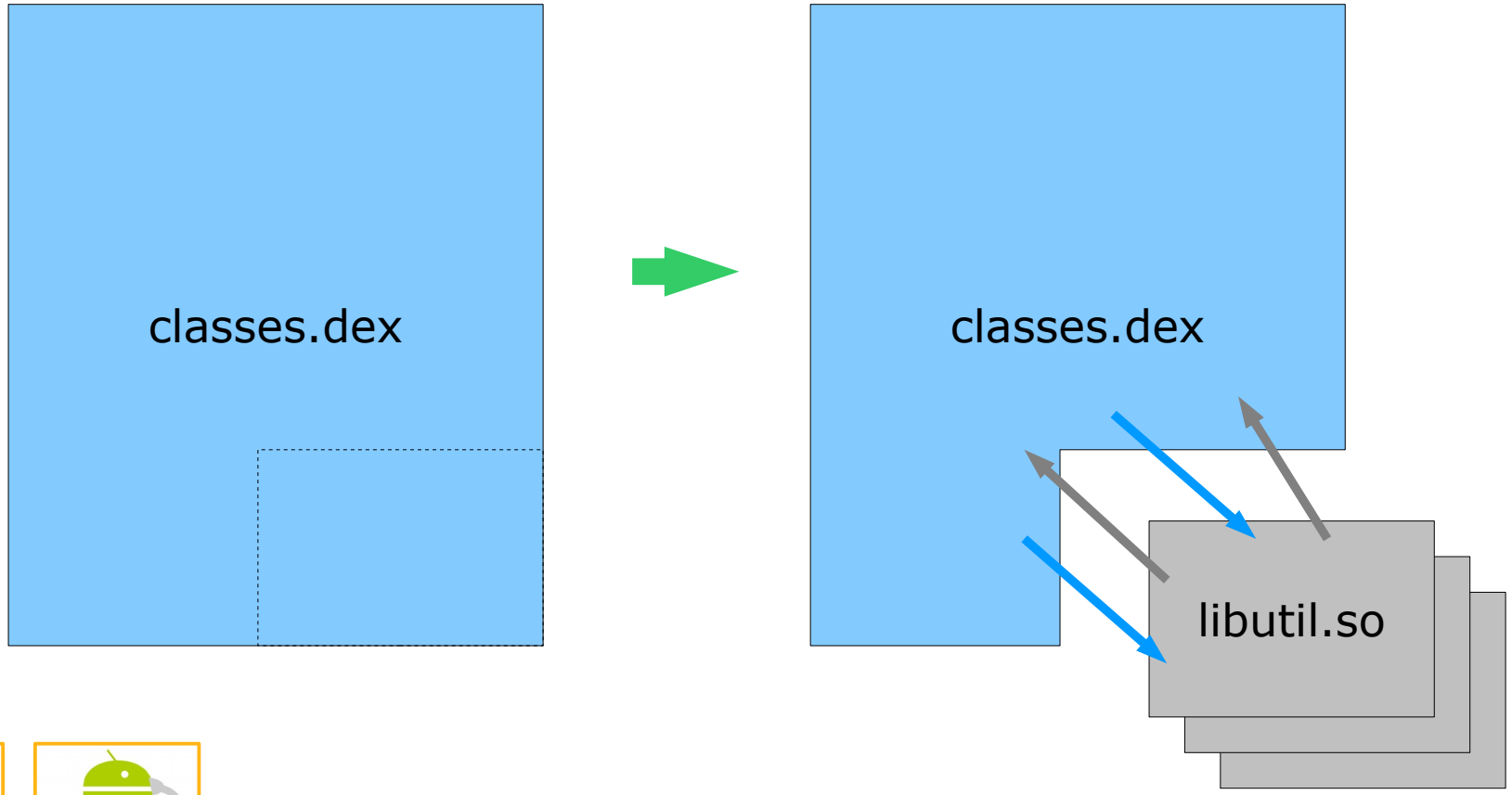
http://android-developers.blogspot.be/2011/07/custom-class-loading-in-dalvik.html

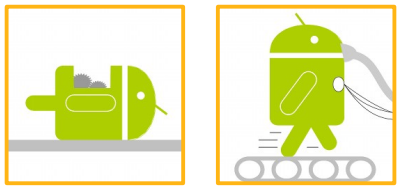




Native code

Java Native Interface





Data encryption

- Standard cryptography APIs

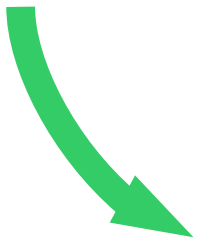
```
Cipher cipher =
    Cipher.getInstance("AES/CFB/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE,
    key, initializationVector);

byte[] decrypted = cipher.doFinal(encrypted);
```

- SQLCipher

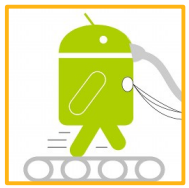
White-box cryptography

Research: www.whiteboxcrypto.com

```
final int KEY = 42;

int decryptValue(int encryptedValue) {
    return encryptedValue * KEY;
}
```



```
int decryptValue(int encryptedValue) {
    encryptedValue += 5;
    encryptedValue *= 21;
    encryptedValue -= 105;
    encryptedValue += encryptedValue;
    return encryptedValue;
}
```



Real examples: DES, AES

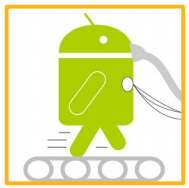
Application checks

- Check application certificate:

```
byte[] certificateData = activity
    .getPackageManager()
    .getPackageInfo(activity.getPackageName(),
                    PackageManager.GET_SIGNATURES)
    .signatures[0]
    .toByteArray();
```

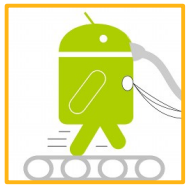
- Check debug flag:

```
boolean debug =
    (activity.getApplicationInfo().flags &
     ApplicationInfo.FLAG_DEBUGGABLE) != 0;
```



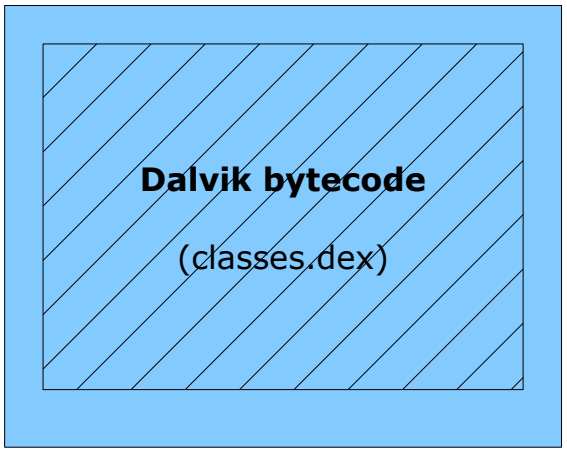
Environment checks

- Emulator detection
- Root detection

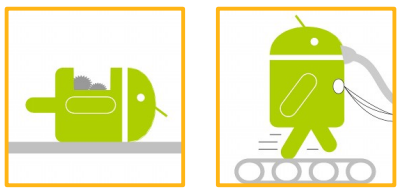


Packing

Files



Memory



Summary

Nothing is unbreakable, but you can raise the bar:

- String encryption
- Reflection
- Control flow obfuscation
- Dynamic class loading
- Native code
- Data encryption
- Application checks
- Environment checks
- Packing

Questions?

ProGuard

Android

Shrinking
Optimization
Obfuscation

Cryptography

Protection

Java bytecode

DexGuard

Dalvik bytecode