Secure Copy Protection for Mobile Apps

Nils T. Kannengiesser
Technical University of Munich
Chair for Operating Systems (F13)
Munich, Germany
Nils.Kannengiesser@tum.de

Uwe Baumgarten
Technical University of Munich
Chair for Operating Systems (F13)
Munich, Germany
baumgaru@tum.de

Sejun Song
University of Missouri-Kansas City
Computing and Engineering
Kansas City, USA
sjsong@umkc.edu

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Content

Introduction

Foundations

Proposals

Conclusion / Outlook / Problems

Nils T. Kannengiesser

Introduction

Nils T. Kannengiesser
http://www.os.in.tum.de/personen/kannengiesser/
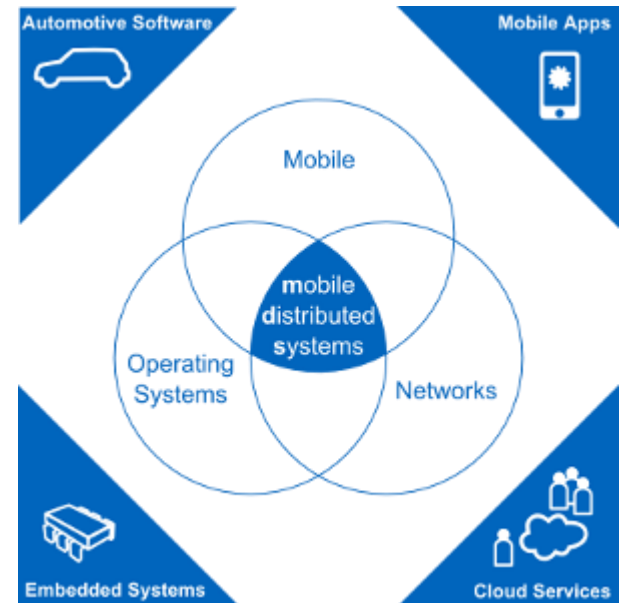
- Computer Science

- Research Assistant at TUM/F13

- Research field:
  Android Security / Copy Protection

- Teaching
  - Android Practical Course
  - "Computersysteme 2" training
  - Advisor of various theses

Chair for Operating Systems (F13)
- Cloud Services
- Automotive Software
- Mobile Apps
- Embedded Systems

Secure Copy Protection for Mobile Apps

Foundations

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Foundations

Main Research Question:

How can be achieved that an app can only be used on valid devices?

Additional notice:

These are early proposals towards the topic.

There are more assumptions than results available yet.

Nils T. Kannengiesser
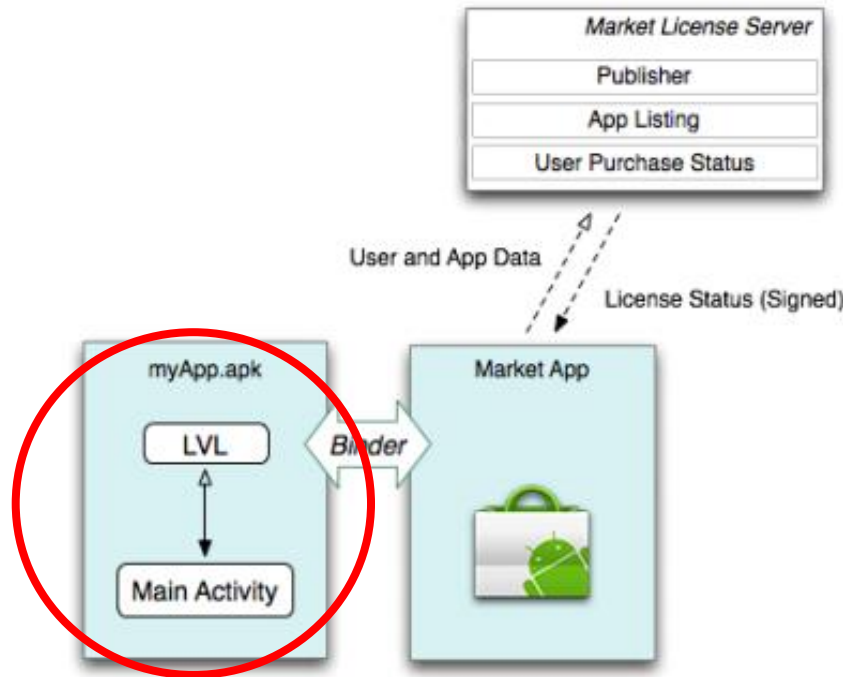
Secure Copy Protection for Mobile Apps

Foundations

- For monetization of apps it is reasonable to use some kind of copy protection (= less piracy)
- Google released the **License Verification Library** in 2010

Nils T. Kannengiesser
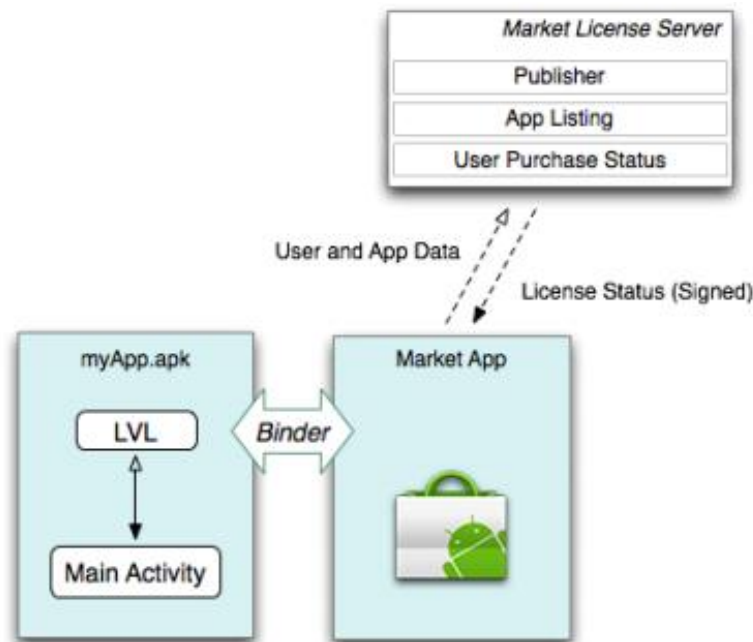
Foundations

- It got cracked the same year by changing a condition within any application

Ref. http://www.androidpolice.com/2010/08/23/exclusive-report-googles-android-market-license-verification-easily-circumvented-will-not-stop-pirates/

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps
Foundations

- The problem: Reengineering (usual) Android Apps is quite readily
  - Tools: APKtool (baksmali/smali), dex2jar & JD-GUI

- The resulting code is an assembly dialect (smali) or even Java code containing **all class and variable names**

Example: LicenseValidator.smali
[…] .field private static final LICENSED:I = 0x0
.field private static final LICENSED_OLD_KEY:I = 0x2
.field private static final NOT_LICENSED:I = 0x1
[…]
.sparse-switch
0x0 -> :sswitch_d3
0x1 -> :sswitch_de
[…]
Ref. Example Source Code - http://www.androidpolice.com/2010/08/23/exclusive-report-googles-android-market-license-verification-easily-circumvented-will-not-stop-pirates/
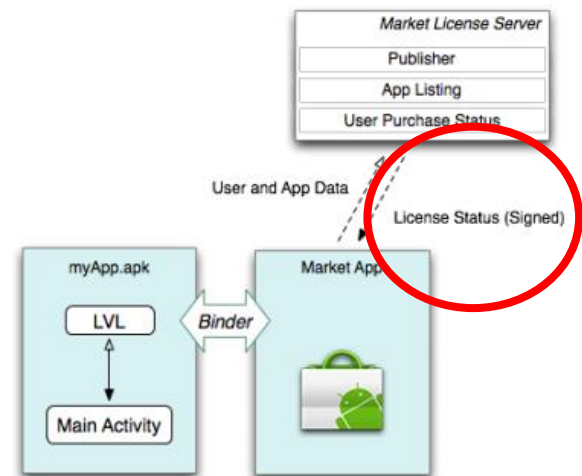
Nils T. Kannengiesser

Foundations

- Later they included another tool in their IDE:
  ProGuard (Obfuscator* ; disabled by default)
  ➔ main reason that lots of apps are still easy to reengineer

- Since then Google updated the LVL, of course. In their latest release they added e.g. signed replies.

* Example:
aa.smali
[…]
.field private static final c:I = 0x1
[…]
.sparse-switch
0x0 -> :sswitch_d3
0x1 -> :sswitch_de
[…]

Ref. Graphic http://developer.android.com/google/play/licensing/overview.html

Nils T. Kannengiesser
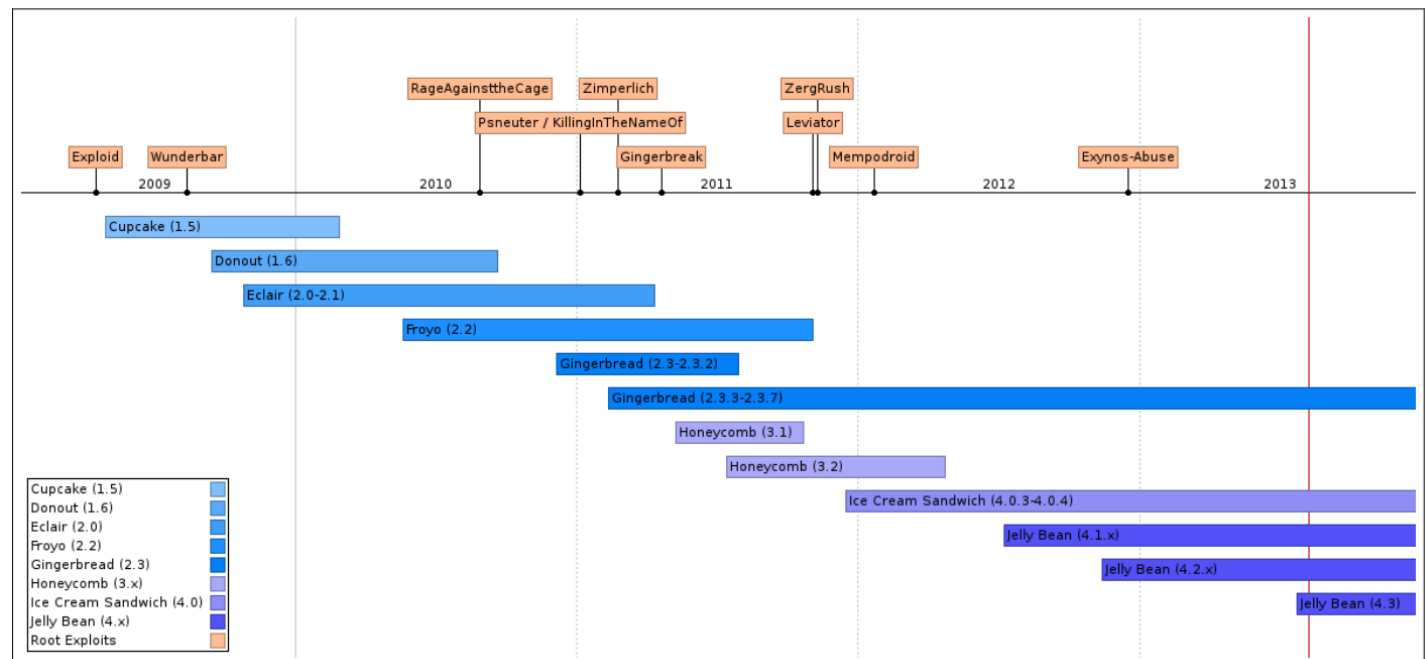
Secure Copy Protection for Mobile Apps

Foundations

Why is reengineering actually a problem? My phone can't be rooted. There are lots of exploits and it can be assumed that every Android version is insecure and it's going to be insecure. It's possible to access APKs* and private files (as root user).

*

Android

Application

Pac**k**age

File

Ref. Graph. Janosch Maier, Bachelor's thesis

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps
Foundations

Current LVL summary:

- Our current assumption is that <u>the LVL is still insecure</u>, since an app can be decompiled and changed to work properly – with certain time effort maybe (cf. obfuscation tools).

- "actual enforcement and handling of the license [...] are up to you" by Google

- There are even deobfuscation tools in current development (by hackers), which might cause problems soon.

- **Their newest LVL release is under current investigation by us.**

Ref. Quote http://developer.android.com/google/play/licensing/overview.html

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Foundations

- Possible solutions (for instance) and related work:

Trusted Execution Environments	Secure Elements	Enhanced Operating Systems
ARM's TrustZone	Giesecke & Devrient's MSC	NSA's SEAndroid
	Any SIM card	Fraunhofer's TrustDroid
	Embedded (phone's NFC chip)	

Advantages
Highly secure   Secure ; HW attached easily   No additional HW required

Disadvantages
New hardware   Limited security (cf. root access)   Less secure (exploits?)
New OS/drivers

Secure Copy Protection for Mobile Apps

Proposals

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Proposals: Hardware

- We decided to go with *Secure Elements*, since our solution should be available for a huge variety of phones
- Due to available partnerships the **M**obile **S**ecurity **C**ard (MSC) by Giesecke & Devrient was picked for our research purposes

Ref. Graphic http://www.gi-de.com/de/about_g_d/press/press_releases/global_press_release_7234.jsp

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Proposals: Hardware

- Since we want to be able to offer it for current smartphones as well, we plan to make use of a MicroSD adapter.

- Function tests using the MSC are still open due to unavailability of the adapter

"Mini MicroSD Reader for Android Smartphones & Tablets", Kickstarter project

Ref. Graphic http://www.kickstarter.com/projects/andyfei/mini-microsd-reader-for-android-smartphones-and-ta

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Proposals: Idea

- The basic idea is to ***bind apps to their owners*** (Apps <> MSC).
- We assume that the source code is available to be able to modify it.
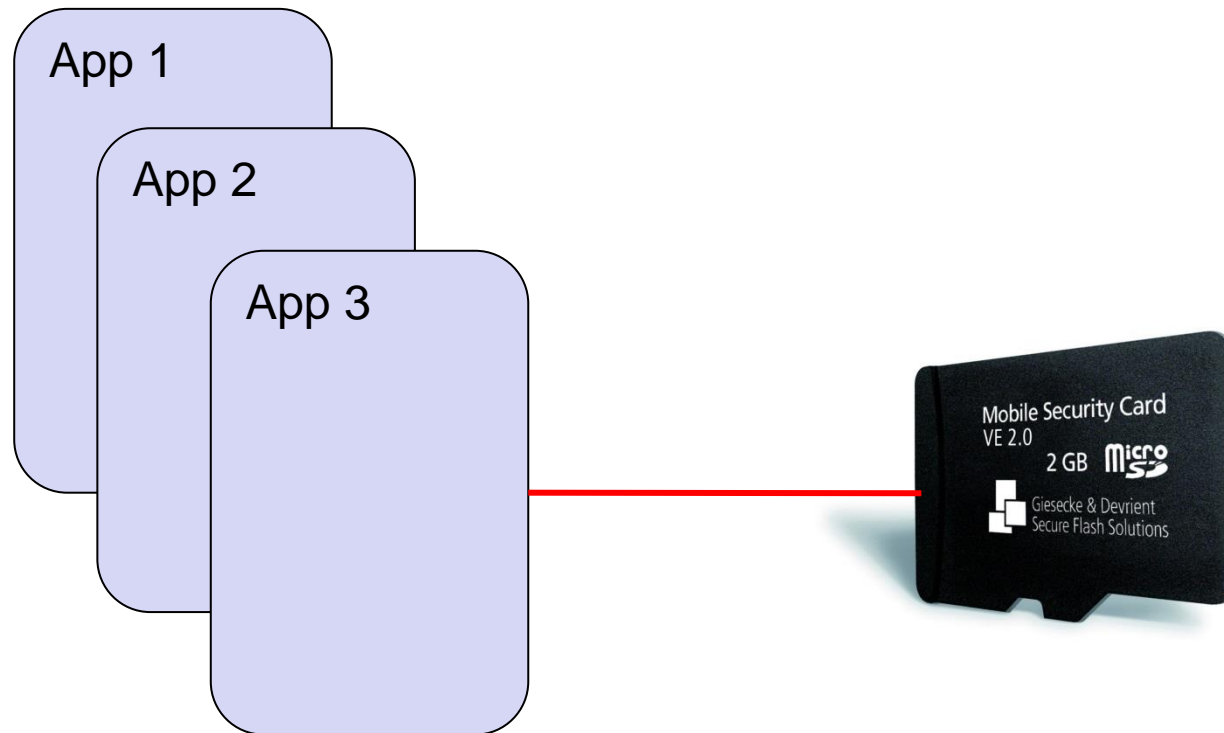
Ref. Graphic http://www.gi-de.com/de/about_g_d/press/press_releases/global_press_release_7234.jsp

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Proposals

The initial goal of every copy protection mechanism is to identify a valid user/device.

- **Device identification**

Current API functions for the device ID might be spoofed.

Therefore we propose to implement our **own identification algorithms** based on "Short Term"- and "Long Term"-information.

MAC 09-11-22-02-24-20
Googleaccount, Contact details, …

Usual phone locations,
available WiFi networks, …

Nils T. Kannengiesser

Proposals

Beside the identification we need *to add new security features* to bind an app and MSC together. Some are based on initial ideas by Google.

- **Content protection (1)**

APK files are usually not encrypted. After reengineering it's possible to extract resources (audio etc.).

We propose to encrypt all files, while the keys are received from the MSC during runtime.

The same applies to received files in future.

Benefit:
Basic protection of files.

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps
Proposals

Beside the identification we need to add new security features:

- **Content protection (2)**

Strings will be replaced by MSC calls, while the MSC returns the string on (validated) requests.

e.g.
MSCauth(…)
[… other code …]
Print: MSC("msg1")

Benefit:
Attackers are no longer able to look for strings in reengineered code nor is it possible to deobfuscate it easily.

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps
Proposals

Beside the identification we need to add new security features:

- **Content protection (3)**

Most URLs should be removed from the original source code and replaced by MSC calls. The MSC will provide a temporary URL later on, while keeping the original URL secret.

e.g.

www.os.in.tum.de/scripts.php?p1=222

www.os.in.tum.de/MSC?appid=1&l=1d2e2&p1=222

www.os.in.tum.de/MSC?appid=1&l=1d2e3&p1=222

Benefit:
Real URLs hidden. It's not possible to use the app without the MSC.
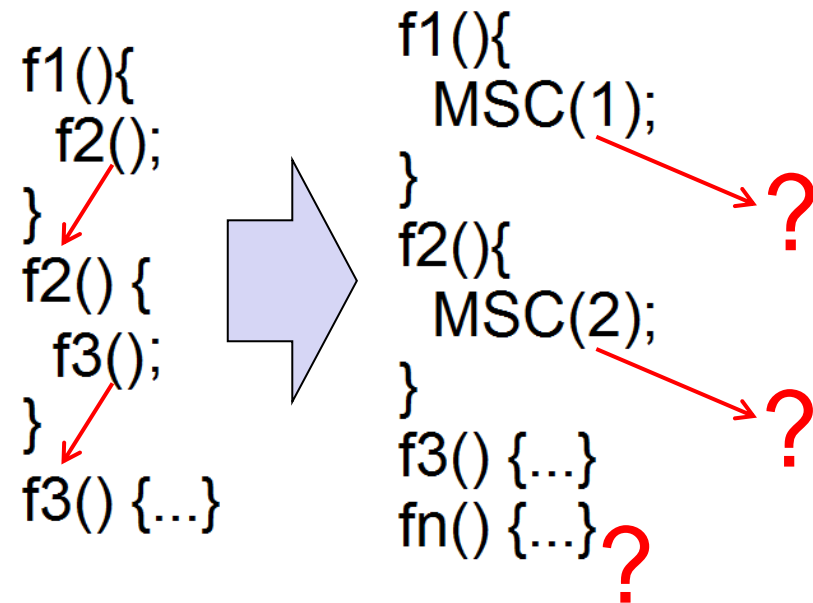
Nils T. Kannengiesser

Proposals

Beside the identification we need to add new security features:

- **Obfuscate execution**

Add nonsense functions based on available functions to the code, while replacing function calls with MSC calls. The MSC replies with the required method name.

Attacker's view after reengineering:

Benefit:
Attackers will be confused by lots of nonsense code.
Reengineering level increased to difficult (live debugging tools?)

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Proposals:

- Also there is the requirement to update the stored information in the MSC according to the used app by using a MSC proxy and a special License server. The connection will be encrypted.

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps
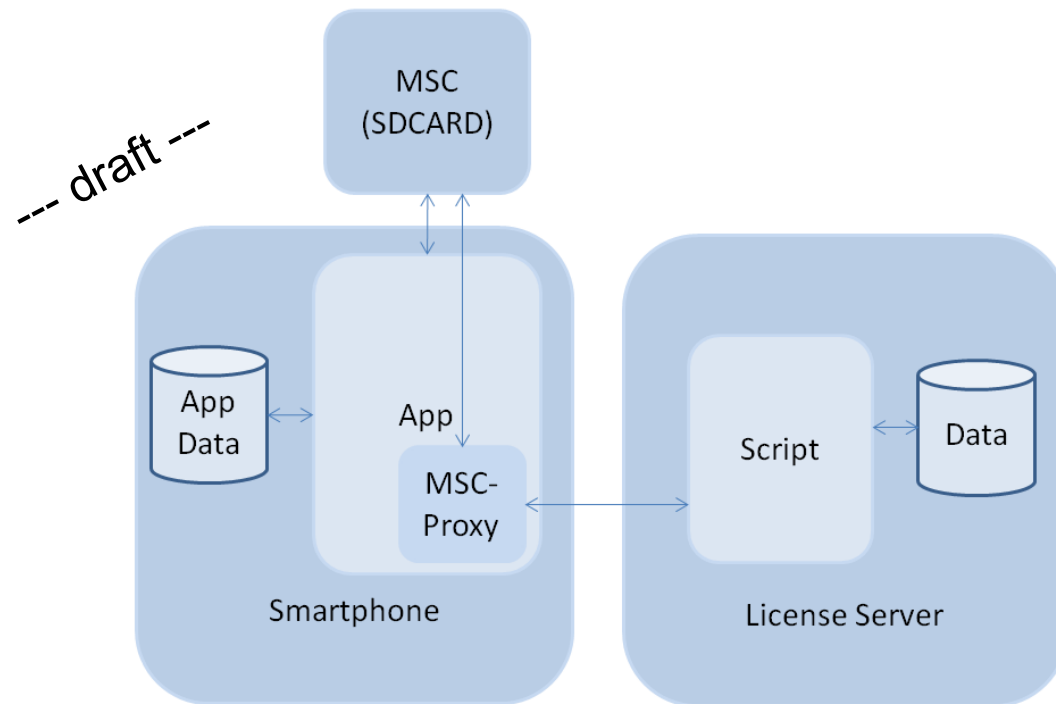
Proposals

Beside these methods, we can try to integrated known methods:

- Integrity checks
- Use methods to break disassemblers (e.g. wrong OPcode)

Ref. Google and http://stackoverflow.com/questions/6481668/protecting-executable-from-reverse-engineering

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Conclusion

Nils T. Kannengiesser

Conclusion / Problems / Outlook

The goal is to create additional developer tools.

We **are assuming** that the proposed methods will **increase the security** and make piracy much more difficult. The next step is to verify open questions (Performance/MSC, MicroSD Adapter/MSC, MSC & devices, […]), start developing, testing and in the end a hacking contest to prove its security benefit.
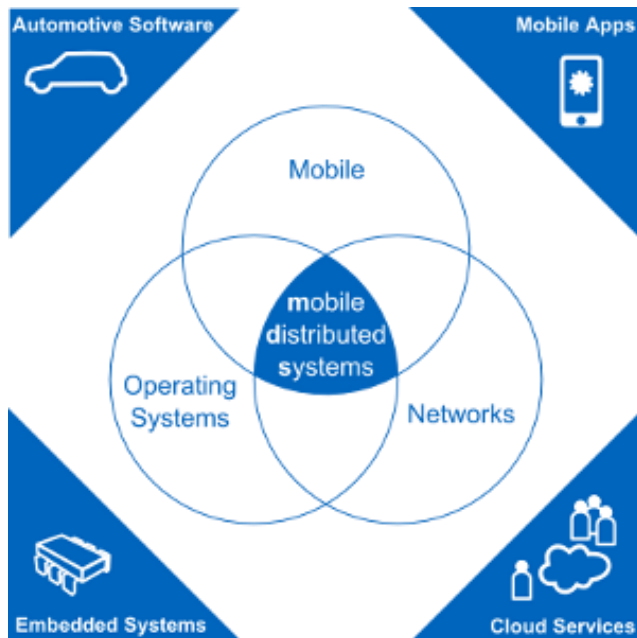
There **might be issues on the performance** of such secure elements, which are unknown currently, too. We want to define methods requiring as less performance as possible to decrease the impact on the runtime performance of apps.

In general Secure Elements might be the key element to provide security in a non-secure environment.

Secure Copy Protection for Mobile Apps

Last but not least …

We are always looking for new partners and cooperations – for research projects as well as for teaching purposes.

Research Topics

Android Practical Course

Deutsches Herzzentrum München

Nils T. Kannengiesser

Secure Copy Protection for Mobile Apps

Thank you for your attention

Nils.Kannengiesser@TUM.de

Android Logo used in this presentation

Portions of this presentation (Android image) are modifications based on work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License.

The logos by our partners are copyrighted by those.

Some images might belong to other owners and might be copyrighted. See references. All other content is copyrighted by the university / chair.

Nils T. Kannengiesser