Debugging and Rapid Prototyping of NFC Secure Element Applications

Michael Roland

7 November 2013 • MobiCASE 2013 – NFC Workshop • Paris, France

This work is part of the project "High Speed RFID" within the EU program "Regionale Wettbewerbsfähigkeit OÖ 2007–2013 (Regio 13)" funded by the European regional development fund (ERDF) and the Province of Upper Austria (Land Oberösterreich).

NFC Research Lab Hagenberg • www.nfc-research.at
A research group of the University of Applied Sciences Upper Austria

Outline

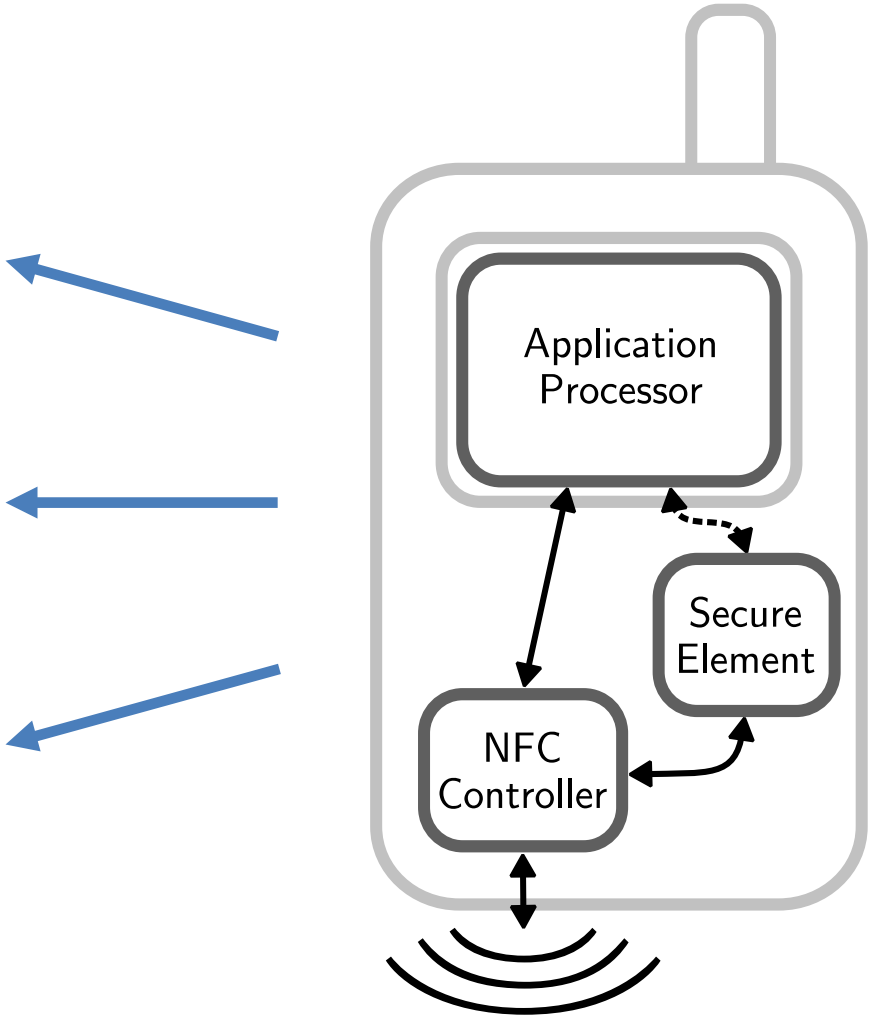
- Introduction
  - NFC in mobile devices
  - Card emulation
  - Java Card
- Debugging, Testing and Rapid Prototyping
  - Current issues
  - Ideal environment
  - Our approach towards a secure element emulator
- Discovered issues
- Conclusion

NFC in a Mobile Device

Reader/writer mode
Read and write NFC tags

Peer-to-peer mode
Exchange data with other NFC devices

Card emulation mode
Emulate contactless smartcards for use with existing readers



Card Emulation Mode

Secure Element

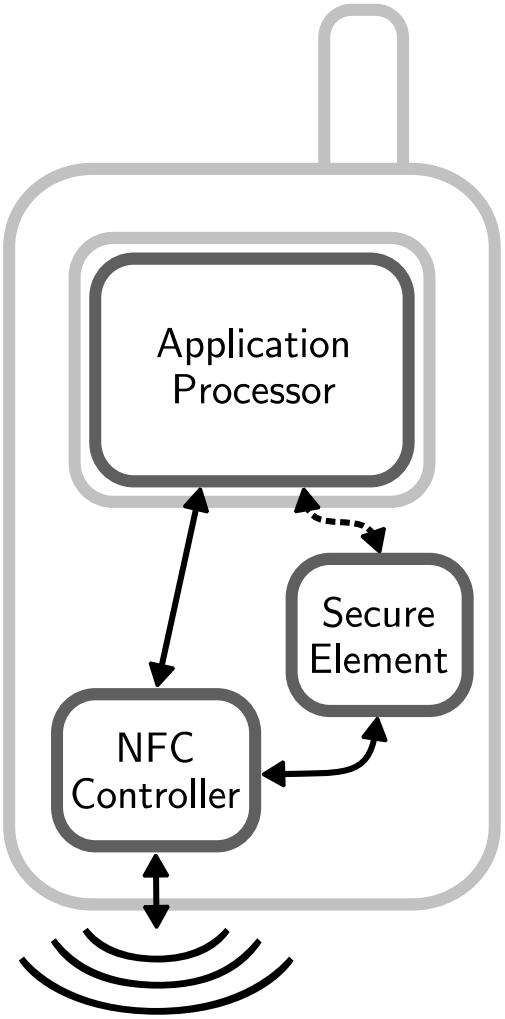
Emulation by secure smartcard chip

- different types: embedded SE, UICC (SIM card), microSD
- complex ecosystem with many players
- difficult/impossible to access by developers and small service providers

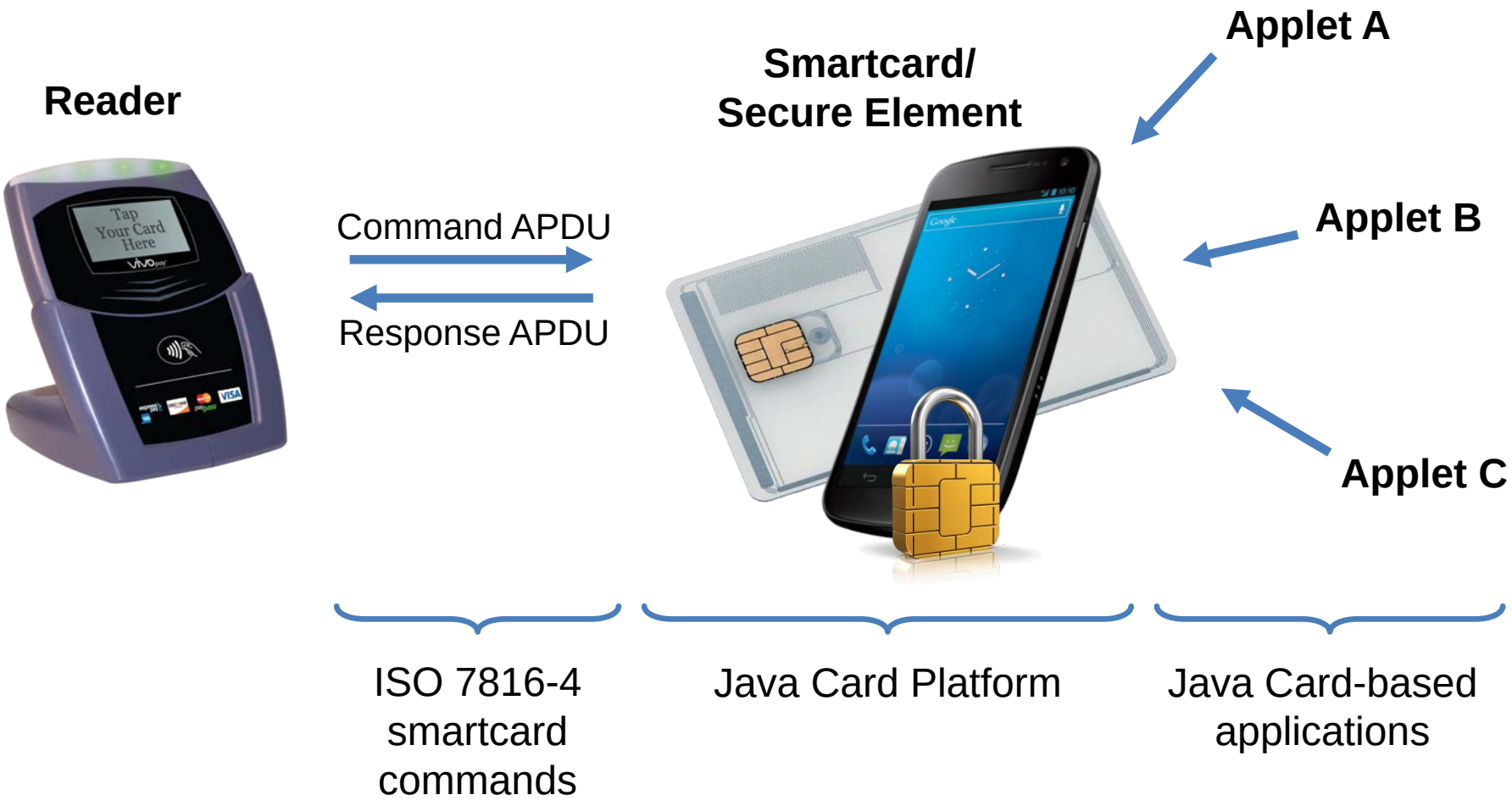
Host-based Card Emulation

Emulation by app on application processor

- flexible and open
- ***no*** secure smartcard chip
- secure element in the cloud possible



Smartcards and Secure Elements



Java Card Platform

- Java Card Runtime Environment
  - Java Card Virtual Machine
  - Java Card API
  - Specific security features
- Java specifically designed for smartcards
  - Small footprint designed for tiny devices
    - Limited memory
    - Limited processing power
  - Limited subset of Java language
    - Reduced set of primitive data types: **`boolean`**, **`byte`**, **`short`**, optional: **`int`**
    - Some Java language constructs not supported
    - Most of Java core API not supported
    - No multi-threading
  - Smartcard-specific classes for application life-cycle management, APDU processing, cryptography, …

Java Card Virtual Machine

- All applications run in one VM
- VM lifetime = smartcard lifetime
  - VM runs from card production until card destruction
  - Code and data memory backed by persistent memory
  - Applications run across power-cycles of the card (from installation until deinstallation)
- Security: application firewalling
  - Strict separation between application contexts
  - Applications cannot access each other's data (unless explicitly granted permission)

Java Card Applets

- One application consists of one or more applets
- Applet is application's entry-point object
- Contains several life-cycle methods invoked by JCRE
  - **`install()`**: create and initialize applet instance
  - **`select()`**: notify applet that it has been selected for command exchange
  - **`process()`**: forward received command APDU to applet
  - **`deselect()`**: notify applet that it has been deselected

Debugging and Testing Java Card Applications

- On smartcard hardware
  - Testing with real smartcard and reader infrastructure
  - Problem: smartcard is secure chip
    - Does not provide debug interface
    - No on-chip debugging
    - No source-level debugging
- In smartcard simulator
  - Source-level debugging (with some simulators)
  - Problem: limited capabilities of most simulators
    - e.g. limited API support, etc.
  - Problem: no communication with real reader infrastructure

Rapid Prototyping of Java Card Applications

- On smartcard hardware
  - Problem: complex secure element ecosystem
    - Difficult/impossible to get access (particularly for prototyping)
- In smartcard simulator
  - Problem: not usable for show-casing of applications

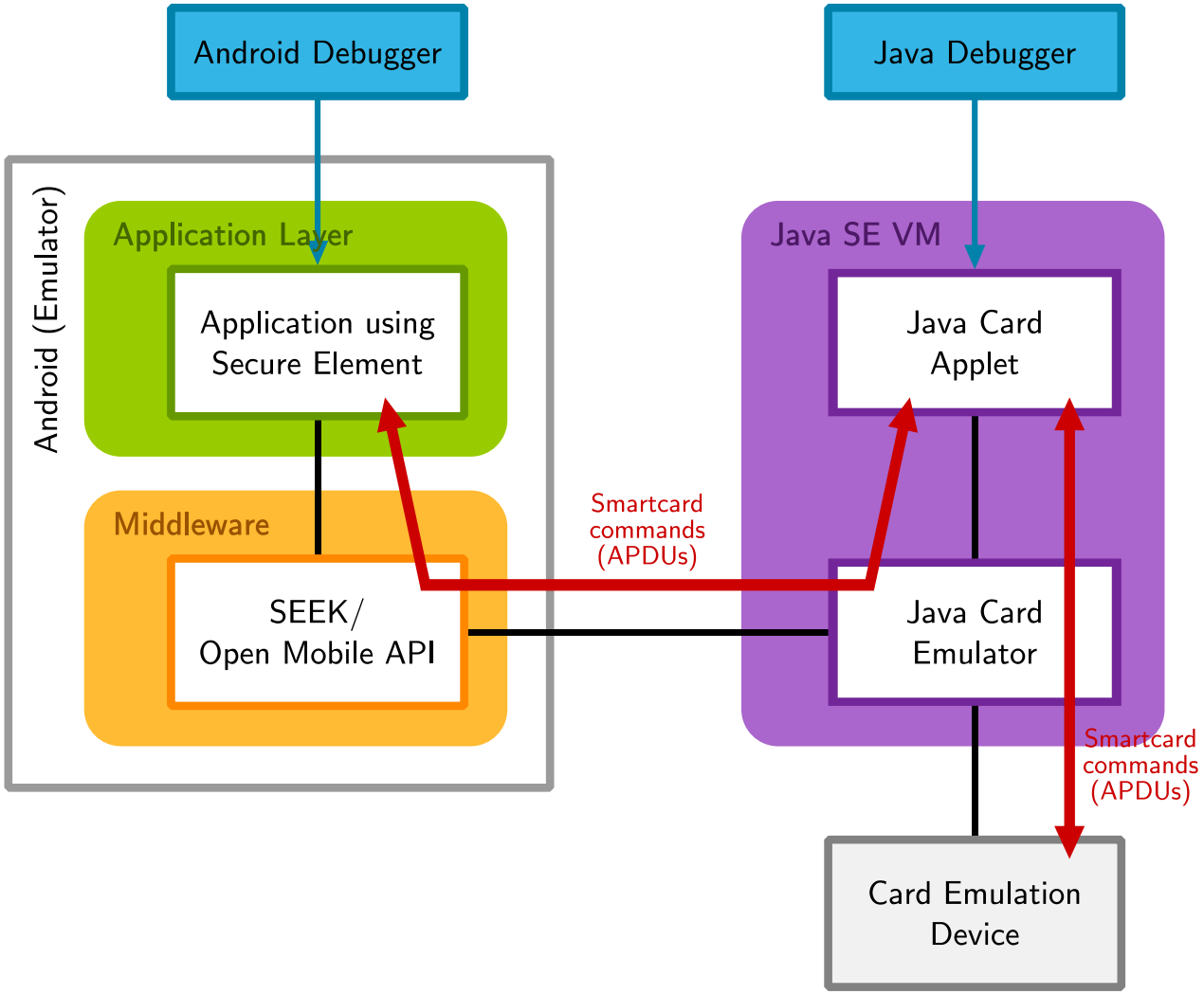
Ideal Environment for Debugging and Rapid Prototyping

- Emulation of complete run-time environment
  - Complete Java Card API (at least comparable to real cards)
  - Same application life-cycle as with real card
- Source-level debugging with existing debugger tools
  - Use same tools as for debugging regular Java applications
- In-place testing, debugging and prototyping
  - Test and debug in combination with other application components
    - Connect emulator to real smartcard readers
    - Connect emulator to software through smartcard/secure element APIs
  - Drop-in replacement for secure element for rapid prototyping
    - Integrate emulator into mobile device platform as “secure element”
    - Trade openness for security

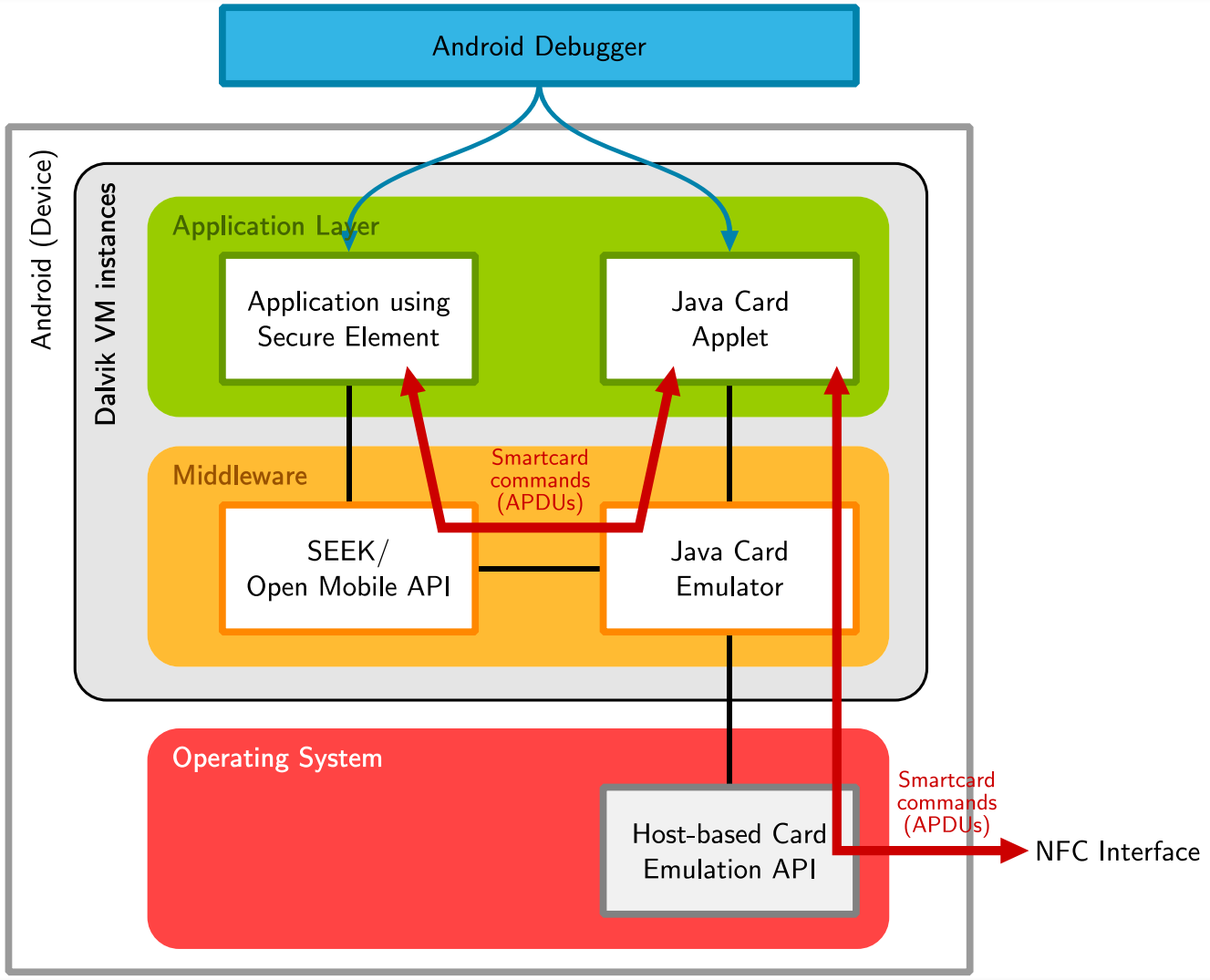
Secure Element Emulator

- Idea: extend existing open-source Java Card simulator
- jCardSim
  - Implements large portions of Java Card API
  - Runs on top of standard Java virtual machine (no Java Card VM)
    - Source-level debugging with standard Java debugger
- TBD
  - In-place testing/debugging
    - Connect with smartcard reader hardware
    - Integrate in mobile device as “secure element”
  - jCardSim only supports short simulation cycles
    - JCRE state and application state do not persist across simulation sessions
  - jCardSim lacks some core functionality
    - Atomic transaction mechanism
    - Logical channel management
  - Integration into Android
    - Run on top of Dalvik VM instead of Java VM

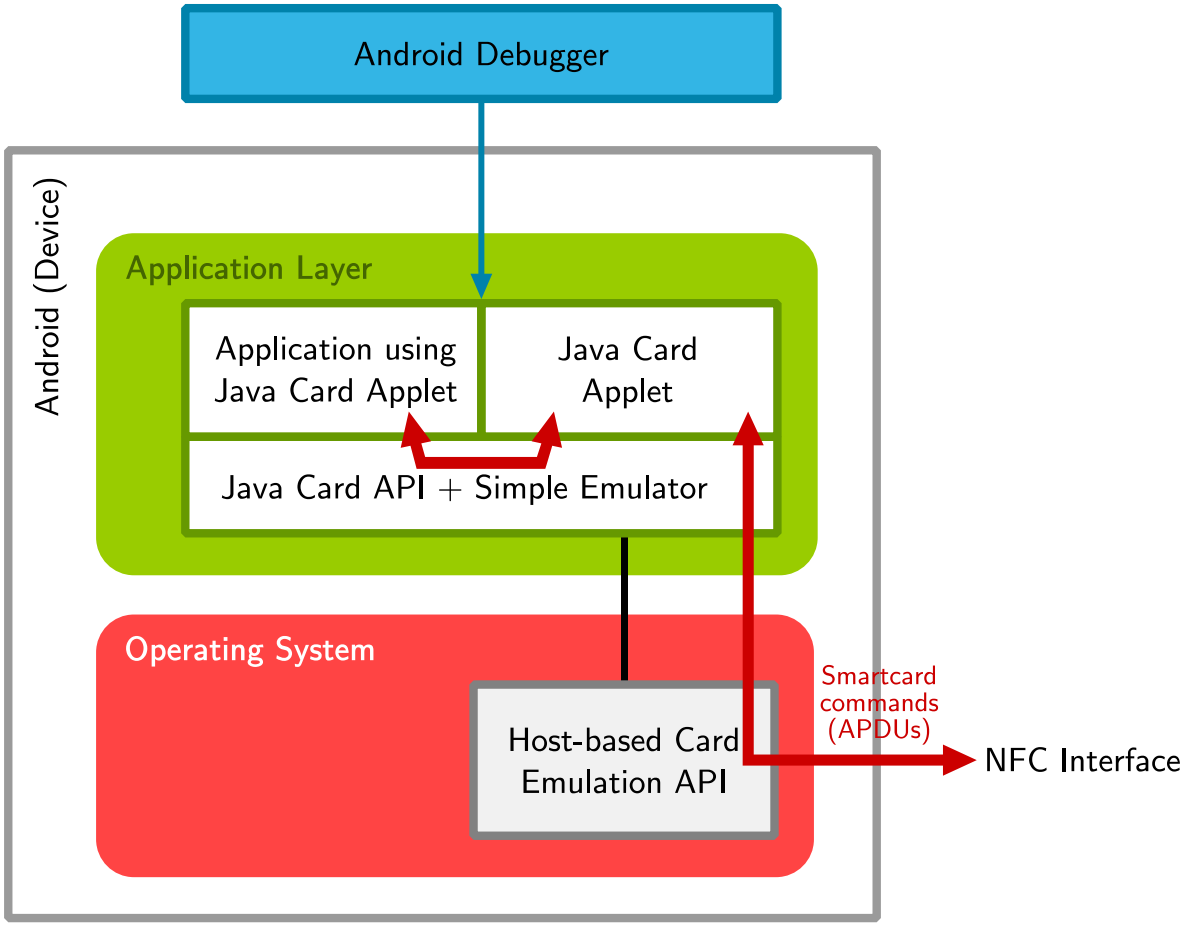
Integration with Android Emulator



Integration with Android NFC Device



Our first prototype



Issues discovered with our prototype

- Java Card atomic transaction mechanism
  - Card tearing not an issue
    - Regular smartcard: tearing causes power-loss/reset
    - Emulator: execution completes after interruptions of the RF connection
      - Transaction mechanism is not necessary in that case
  - Roll-back of transactions is not possible
    - Java does not have an atomic transaction mechanism by default
    - Variables/objects involved in transactions cannot easily be rolled-back to a defined boundary
- State of JCRE and applications is not persistent
  - Significant differences between lifetime of Dalvik/Java VM and Java Card VM
  - If emulator environment is terminated (app closed, device rebooted, etc) the state of the JCRE and all Java Card applets is lost
  - Upon restarting the emulator all applets start at the beginning of their life-cycle

→ *Methods for extracting and re-implanting application state necessary!*

Conclusion

- Developed new concepts for in-system debugging of secure element applications / Java Card applications
  - In-place debugging with other system components
    - Secure element API
    - RF interface
  - Drop-in replacement for secure element for rapid prototyping
    - Open environment but less/no security
- Created a first proof-of-concept prototype
  - Successful emulation and in-place debugging of simple applet
    - Even single-stepping through code possible
  - Identified several unresolved issues based on the prototype
    - Different life-cycle of Java Card VM in comparison to other Java VMs
    - Use of persistent memory technology for storing objects

Dr. Michael Roland

Research Associate, NFC Research Lab Hagenberg
University of Applied Sciences Upper Austria

michael.roland (at) fh-hagenberg.at

This work is part of the project "High Speed RFID" within the EU program "Regionale Wettbewerbsfähigkeit OÖ 2007–2013 (Regio 13)" funded by the European regional development fund (ERDF) and the Province of Upper Austria (Land Oberösterreich).

NFC Research Lab Hagenberg • www.nfc-research.at
A research group of the University of Applied Sciences Upper Austria