Android Security Symposium, Vienna, 9-11 September 2015

Secure Elements for You and Me: A Model for Programmable Secure Hardware in Mobile Ecosystems

Alexandra Dmitrienko

Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany

Joined work with

Marcos da Silva Ramos, Andre Rein

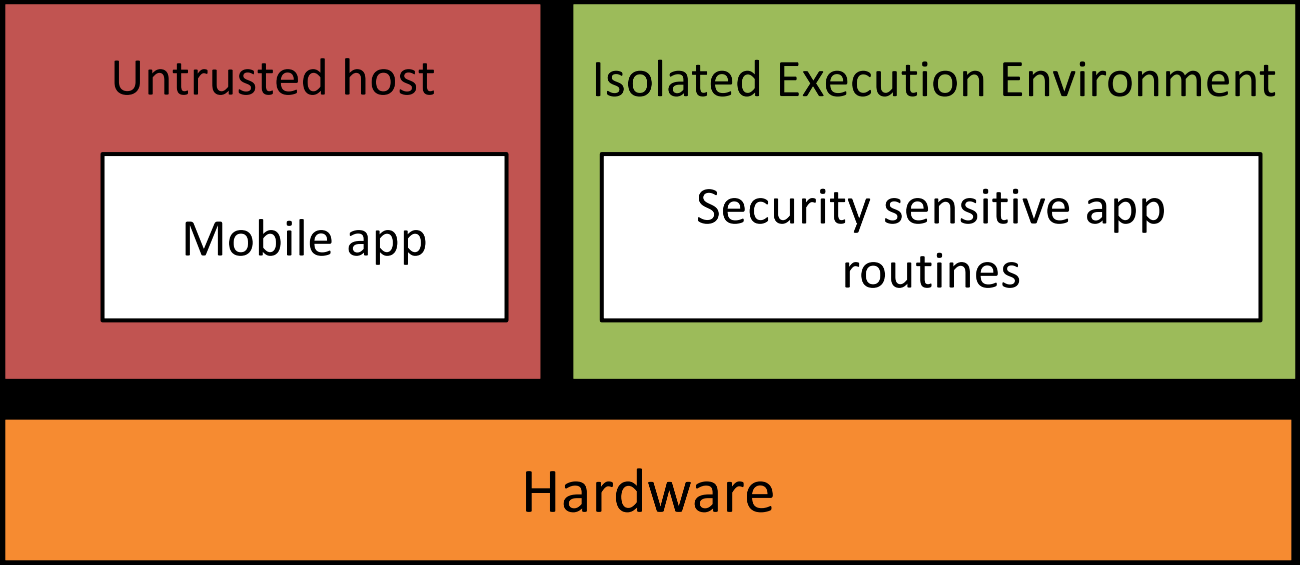
Fraunhofer SIT, Darmstadt, Germany

Stephan Heuser, Thien Duc Nguen, Ahmad-Reza Sadeghi

Technische Universität Darmstadt, Germany

Motivation

- Secure hardware can drastically improve security of mobile apps
  - Provides isolated execution environment for security sensitive app routines


Untrusted host
Mobile app
Isolated Execution Environment
Security sensitive app routines
Hardware

Secure Hardware

Processor-based Trusted Execution Environments (TEEs)

- ARM TrustZone, TI M-Shield

Secure Element (SE) on UICC (or SIM) cards

Secure Hardware

Embedded SE (eSE)

- Standalone or NFC-based

Secure Element (SE) on an SD memory card

Mobile Payments Apps with Secure Hardware Support

- SE-supported apps from OS vendors
  - Google Wallet and Apple Play
- SIM-based apps from Mobile Network Operators (MNOs)
  - Vodafone payment app, MyWallet from Telekom, etc.
- But, “[…] there is a major shift towards cloud based software solutions away from hardware based technologies […]” (Advanced Payments Report 2014, Edgar Dunn & Company)
  - E.g., solutions of PayPal and Visa rely on cloud-based services

Other Security Sensitive Apps

- Online banking, mobile ticketing, access control applications, etc.
    - Typically rely on software-based security mechanisms
        - SmartCard emulated in software
        - OS-level isolation and access control

Why?

Secure hardware is not freely programmable

Problem Description and Challenges

- Widely deployed Secure Elements (SE) and Trusted Execution Environments (TEEs) are controlled by their stakeholders
- No effective business models exist to allow third party app developers to interact with SE/TEE stakeholders
- Access control to secure hardware APIs is controlled either by their stakeholders or by OS vendors, but not by app developers

Our Goal

A framework to allow third party developers to develop for secure hardware and to deploy their code

- Incentives for secure hardware stakeholders to allow such access
- Simplified interaction of third party developers and secure hardware stakeholders
- Access control to secure hardware APIs independent from OS vendors

Core Ideas

Market place code distribution ecosystem

- Bridges small and mid-size app developers and large hardware stakeholders

DRM for secure hardware code

- Hardware stakeholders grant installation rights (to users) by issuing installation tokens
- Financial incentives for hardware stakeholders

Developer-centric access control to secure hardware APIs

- Distribute access policy in installation tokens

Involved Parties

Developer D
- Develops mobile apps and applets, trustlets or trusted apps

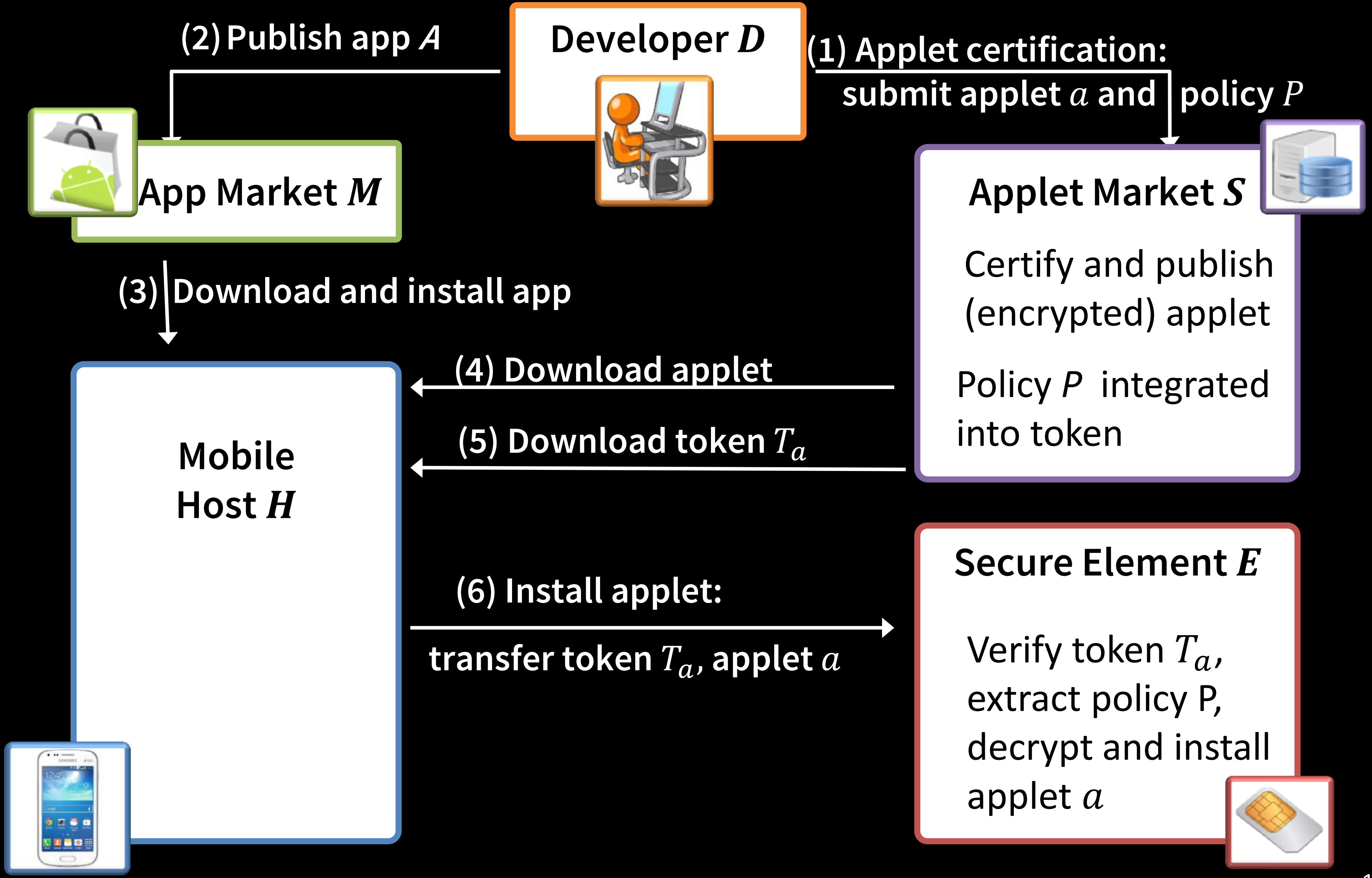
App Market M
- Usual app market where mobile apps are published

Applet Market S
- Maintained by the stakeholder of secure hardware

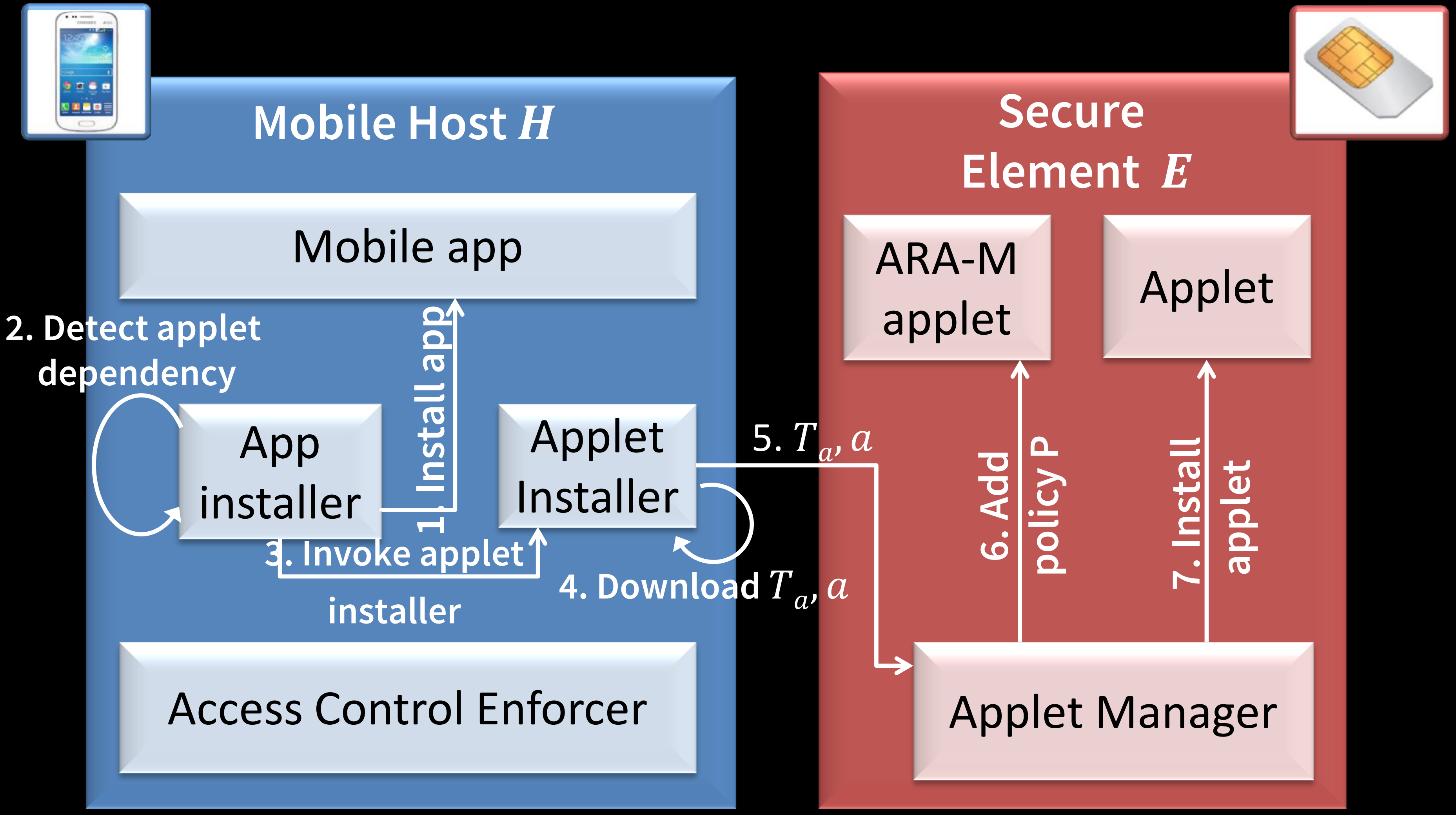
Secure Element SE
- Secure hardware which can run applets, trustlets, etc.

Mobile Host
- Mobile device, such as a smartphone or tablet

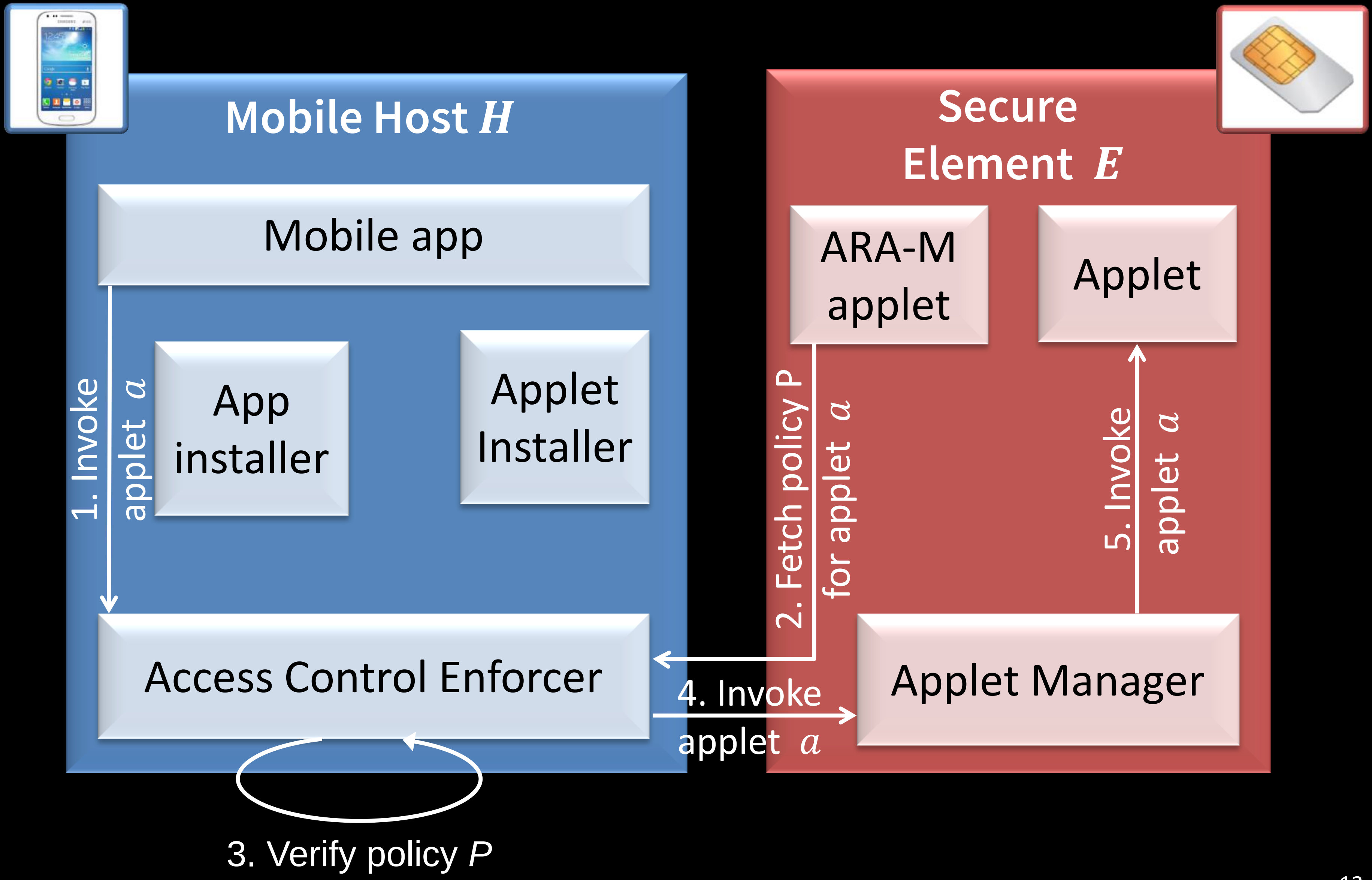
System Architecture


(2) Publish app A
Developer D
(1) Applet certification: submit applet a and policy P
App Market M
Applet Market S
Certify and publish (encrypted) applet
Policy P integrated into token
(3) Download and install app
(4) Download applet
(5) Download token T_a
Mobile Host H
(6) Install applet:
transfer token T_a, applet a
Secure Element E
Verify token T_a, extract policy P, decrypt and install applet a

Applet Installation



Applet Invocation



Associated Challenge

- Market-driven code provisioning scheme may stipulate rapid development of various SE applets
- Limited resources of hardware-based SEs limit number of installed applets
- Current resource quota mechanisms for SEs are not effective in case of market-driven code provisioning

Our solution: SE resource management

SE Resource Management

Applet Manager performs resource management by

- monitoring available SE resources
- maintaining statistics of applet usage
- de-installing of rarely used applets in case SE is out of resources
- installing applets on-demand in case previously de-installed applet is invoked

- On-demand applet installation
  - Transparent to mobile applications invoking the applet
  - May impact runtime performance

Implementation

Module	Size (LoC)	Language	Dependencies
Applet Manager	791	Java/Android	SpongyCastle Crypto API
jCardSim4Android*	4923	Java/Android	SmartCardIO
SmartCardIO**	728	Java/Android	
Applet Installer	1124	Java/Android	SpongyCastle Crypto API, Communication API
SE Stakeholder	1390	Java 6	BouncyCastle Crypto API, Communication API
Developer	656	Java 6	BouncyCastle Crypto API, Communication API
Communication API	545	Java 6, Java/Android	

*Port of open-source project jCardSim (http://jcardsim.org/) to Android
** Port of javax.smartcardio classes from Open-JDK v7 to Android

External SE Deployment

- Host on mobile device
- Emulated SE on a smartwatch (or any other wearable)

Performance Evaluation

	Applet installation*, ms	Applet de-installation*, ms	Applet execution**, ms
H and E on the same smartphone	46.27 ± 19.19	15.76 ± 7.83	38.43 ± 18.47
H on the smartphone, E on the smartwatch	415.27 ± 78.00	205.36 ± 72.54	150.34 ± 50.07
Hardware-based SE			24.47 ± 1.86

* Applet size 10953 Bytes

** Send 4 bytes to the applet and reply with 10 bytes

- Smartphone: Samsung Galaxy S3 (Android 4.4)
- Smartwatch: Samsung Galaxy Gear SM-V700 (Android 4.2)
- Hardware-based SE: Mobile Security Card SE 1.0 (G&D)

Summary and Further Work

Our model for programmable secure hardware

- Provides incentives for SE/TEE stakeholders
- Bridges small-size app developers and large stakeholders
- Developer-centric access control to secure hardware APIs compatible to Global Platform specifications

Further work:

- Support for Trusted Execution Environments
- Support for multiple types of secure hardware on a single platform

E
?
R
S
D
F