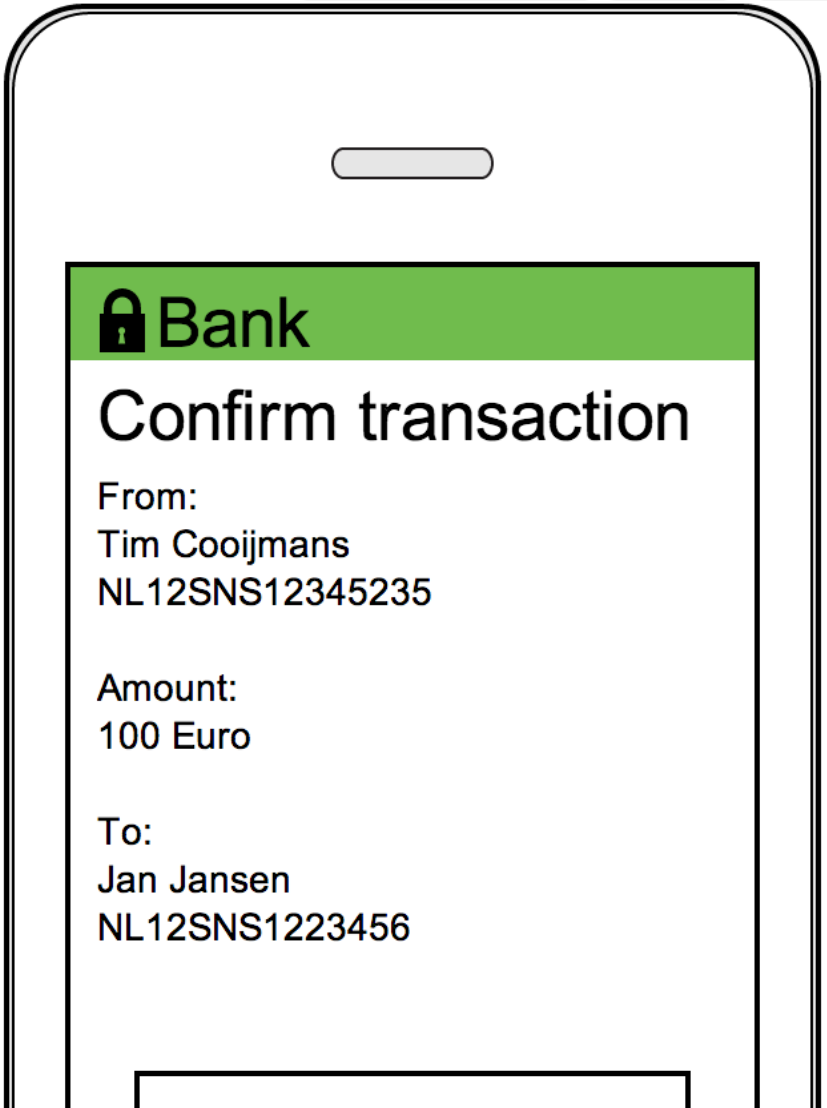
Analysis of Secure Key Storage Solutions on Android

Tim Cooijmans, **Joeri de Ruiter**, Erik Poll

Digital Security, Radboud University Nijmegen

Mobile payments

- App to transfer money or pay in a shop
- Transaction needs to be approved by user
- Typically done by signing transaction data
- How can this key be protected?



Secure key storage

- Secure environment to store keys
  - Symmetric: AES, 3DES
  - Asymmetric: RSA, ECC
- Cryptographic operations usually performed within secure environment
- Form of secure computations

Secure key storage on Android

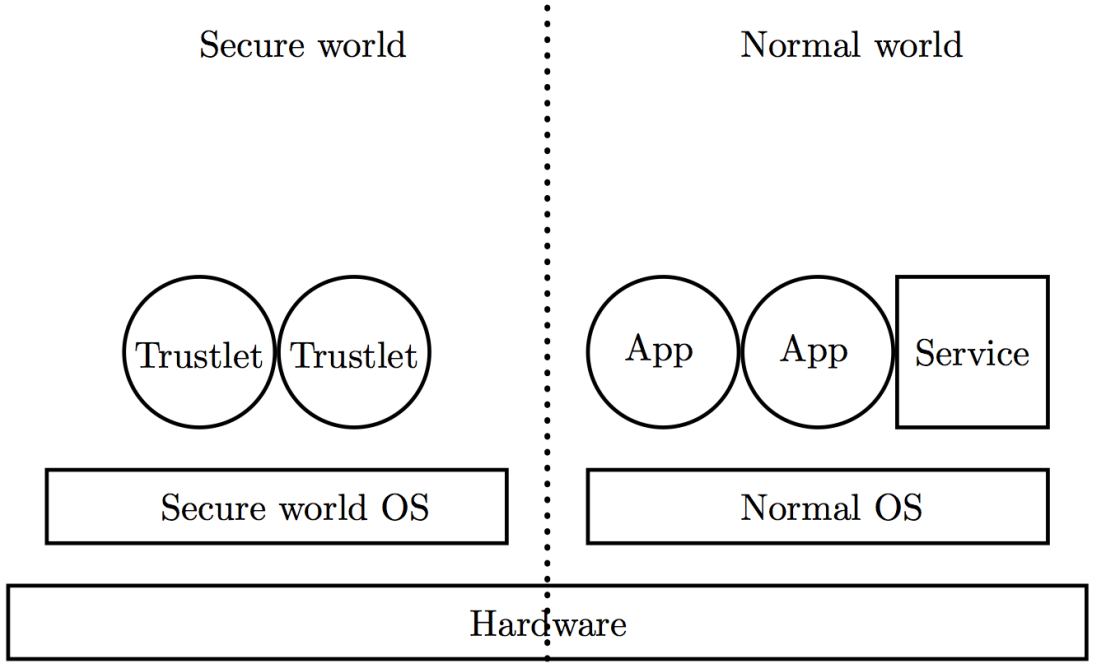
- Biggest OS for mobile devices
- Open environment
  - Easy to inspect or make changes
- More public documentation available
- Focused on Nexus devices

Security mechanisms

- Access control
  - File system
  - Unique user id per app
- Trusted execution environment (TEE)
  - Complete app in secure world
  - Cryptographic operations in secure world
- Password-protected storage
  - Stored password
  - User-provided password

TrustZone Technology

- ARM processor feature
- Default in processor designs
- Two execution environments
  - Normal world
  - Secure world
- Virtualisation



Secure key storage solutions: Bouncy Castle

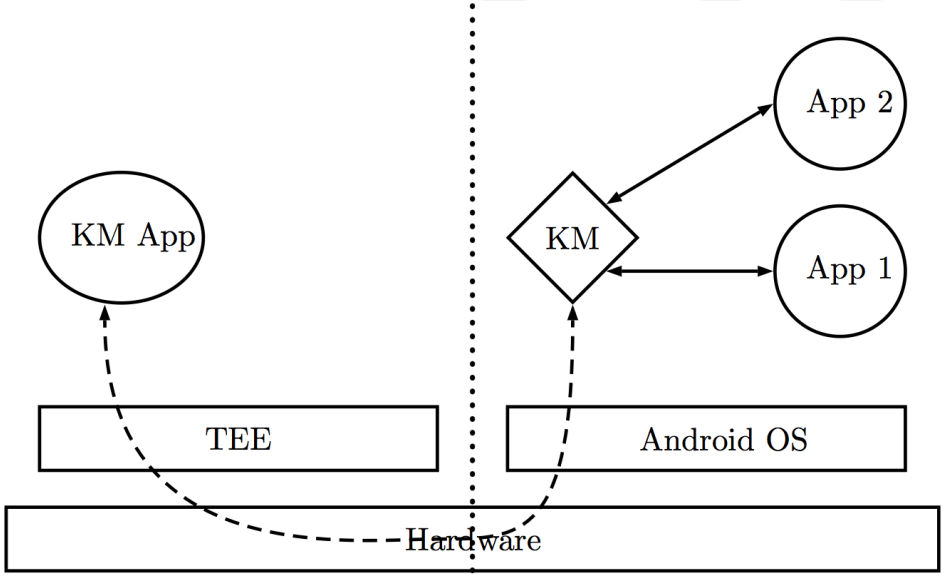
- Cryptographic library for Java
- Stripped down for Android
- Software only
  - Stored password
  - User-provided password

Secure key storage solutions: Bouncy Castle

- Cryptographic library for Java
- Stripped down for Android
- Software only
  - Stored password
  - User-provided password

Secure key storage solutions: AndroidKeyStore

- Available since 4.3
- Service running in background
- No access to key
- Hardware-based secure storage
  - Qualcomm devices
  - Texas Instruments devices
- Software-fallback



Attacker models

- Malicious app attacker
  - Attacks other apps
- Root attacker
  - Full access to file system
- Intercepting root attacker
  - Full access to file system and user input

Security requirements

- Device-binding
  - Key can only be used on one device
- App-binding
  - Key can only be used by one app on one device
- User-consent
  - Key can only be used with explicit consent from user

Method

- KeyStorageTest app
  - Check if algorithms are bound to device
  - Generate keys
  - Generate signatures
- Try to create a valid signature
  - Using another app
  - On another device
  - Without asking for user consent

Devices

Phone	Processor	Android version	TrustZone
Nexus 4	Qualcomm	4.4.2	Yes
Nexus 5	Qualcomm	4.4.2	Yes
Galaxy Nexus	Texas Instruments	4.3	Yes*
Nexus S	Samsung	2.3.6	No
Nexus S	Samsung	4.1.2	No
Moto G	Qualcomm	4.3	No
Moto G	Qualcomm	4.4.2	Yes

* not enabled by default

Bouncy Castle using stored password

- Password stored in app specific directory

	Malicious app	Root	Intercepting root
Device-binding	✔	✘	✘
App-binding	✔	✘	✘
User-consent	✔	✘	✘

Bouncy Castle using user-provided password

- Password not stored on device
- Password need to contain enough entropy
- Correctness of password can be verified using Bouncy Castle's integrity check

	Malicious app	Root	Intercepting root
Device-binding	✓	✓*	✗
App-binding	✓	✓*	✗
User-consent	✓	✓*	✗

AndroidKeyStore

- Storage handled by service in Android
- Two files on filesystem
  - Key parameters and encrypted private key

    `10101_USRPKEY_TestKeyPair`
  - Certificate

    `10101_USRCERT_TestKeyPair`
- Files can be moved by root user

AndroidKeyStore using TEE

- Qualcomm: Nexus 4 and 5, Moto G
- Texas Instruments: Galaxy Nexus
- Trustlet running in TEE for key operations
- Device specific key
- File formats different

	Malicious app	Root	Intercepting root
Device-binding	✓	✓	✓
App-binding	✓	✗	✗
User-consent	✓	✗	✗

AndroidKeyStore using software-fallback

- No encryption if no device PIN/password set

	Malicious app	Root	Intercepting root
Device-binding	✓	✗	✗
App-binding	✓	✗	✗
User-consent	✓	✗	✗

- Encrypted using device PIN/password

	Malicious app	Root	Intercepting root
Device-binding	✓	✓*	✗
App-binding	✓	✗	✗
User-consent	✓	✗	✗

Discussion

- All methods safe against malicious app
- Bouncy Castle
  - If user-provided password is used app- and device-binding provided
- AndroidKeyStore
  - No app-binding
  - No posibility for user-consent
  - Possible to create oracle
- User-consent
  - No control over what is signed

Recommendations

- Use Bouncy Castle with user-provided password if possible
  - Be careful with passwords
- Educate users about the dangers of root
- AndroidKeyStore
  - Include user id integrity in key files
  - Offer possiblity to use user-provided passwords
- Improve TEE implementations
  - Check apps that request key operations
  - What-you-see-is-what-you-sign

Conclusions

- Bouncy Castle provides reasonable security when using user-provided password
- AndroidKeyStore
  - TEE-based store provides device-binding
  - No app-binding so effectiveness is limited
  - Can be improved by using more TEE-features
    - Confirmation dialogs
    - Integrity checking

Conclusions

- Bouncy Castle provides reasonable security when using user-provided password
- AndroidKeyStore
  - TEE-based store provides device-binding
  - No app-binding so effectiveness is limited
  - Can be improved by using more TEE-features
    - Confirmation dialogs
    - Integrity checking

Thank you for your attention!