Android Encryption

THCON 2022

14/04/2022

Agenda

- Introduction
- Android Data Encryption solutions
- File Base Encryption and Security Model
- Encryption with Secure Element
- Conclusion

Presentation

- **Jean-Baptiste Cayrou**
  - Security researcher @Synacktiv
  - Vulnerability research & exploitation
- **Synacktiv**
  - Offensive security company
  - Offices in Paris, Toulouse, Lyon and Rennes
  - ~100 Ninjas
  - We are hiring!!!

Introduction

- **Our smartphones contain a lot of sensitive data**
  - Email and conversations
  - Browsing history
  - Photos and videos
  - Bank accounts or cards
- **These data must be protected if the device is lost or stolen**
- **This talk focuses on cold boot case**
  - Scenario with best protections

Android Devices

- **Google develops the Android Open Source Project (AOSP)**
- **Android provides an architecture to help vendors to implement encryption**
  - Interface with Android code is generic
  - Vendors must write the low level part (the hardware support)
- **Final integration is performed by vendors**
  - This talk focuses on AOSP implementation guidelines

Android Data Encryption

Android Encryption

- **Data encryption is mandatory since Android 5.0 (2014)**
- **Only user data are encrypted**
- **Two kinds of encryption**
  - Full Disk Encryption (Android >= 4.4)
  - File Based Encryption (Android >= 7.0)
- **Most of encryption implementations use hardware security features**

Full Disk Encryption

- **Full Disk Encryption - FDE**
- **At boot, the system asks for a secret (PIN, Pattern, Password)**
- **Encryption is performed at block device level**
- **Will become deprecated**
  - Starting with Android 10 new devices must use File Base Encryption
  - Code will be removed in Android 13

File Base Encryption

- **Available since Android 7.0 (2016)**
- **Encryption is performed on files and not on the entire block device**
- **Device Encrypted (DE) storage**
  - Encryption key is usually bound to the HW but loaded at boot without user secret
  - Used to encrypt system data
- **Credential Encrypted (CE) storage**
  - Encryption key is usually bound to the HW and requires user credentials to be decrypted
  - Used to encrypt user data

File Base Encryption

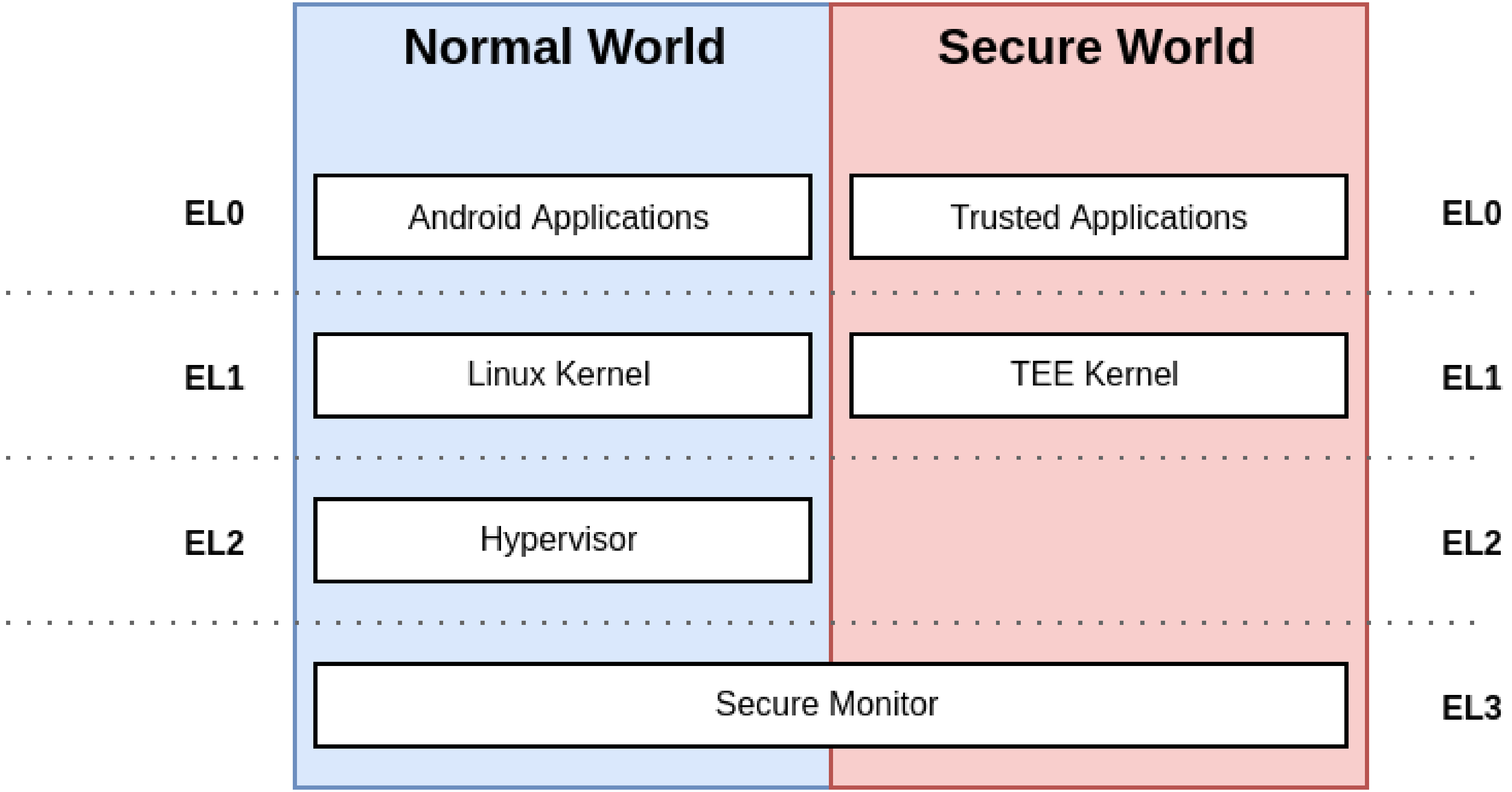
- **Android Direct Boot**
- **Start some applications before the user has unlocked the device**
  - Using the Device Encrypted storage key
  - E.g. the Alarm application

File Base Encryption and Security Model

ARM TrustZone

- **The CPU has two execution environments**
  - Secure World: Privileged mode. Run highly sensitive software
    - TEE - Trusted Execution Environment
    - Run Trusted Applications (TA)
      - Cryptographic keys, DRM, Banking data, biometric sensors
  - Normal World: Run Android kernel and applications
    - REE - Rich Execution Environment
- **If Normal Wold is compromised, cryptographic assets are still safe**

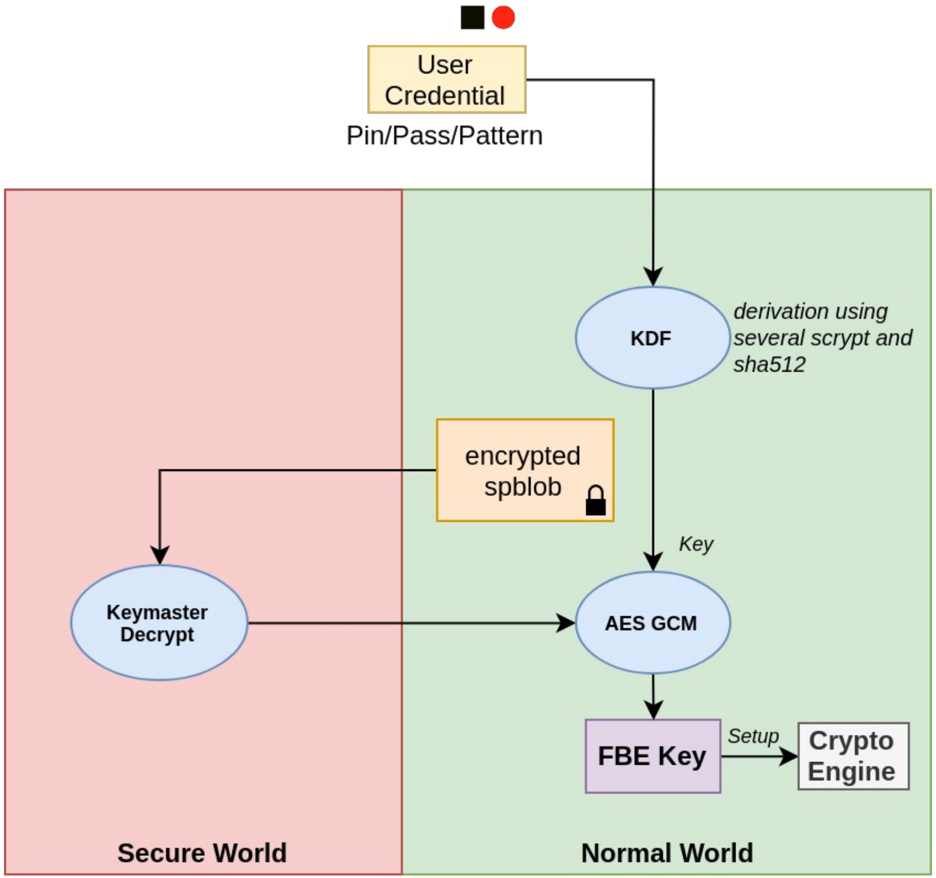
ARM TrustZone



Encryption Overview

- **Android Encryption logic is implemented by the SyntheticPasswordManager**
- **Based on an user secret (Pin, Password, Pattern)**
- **Cryptographic assets are protected by the TEE**
  - Theses assets are bound to the Hardware
  - They are safe even if the normal world is compromised
- **Key derivation must be performed on the device**
- **Request throttling to avoid online bruteforce**

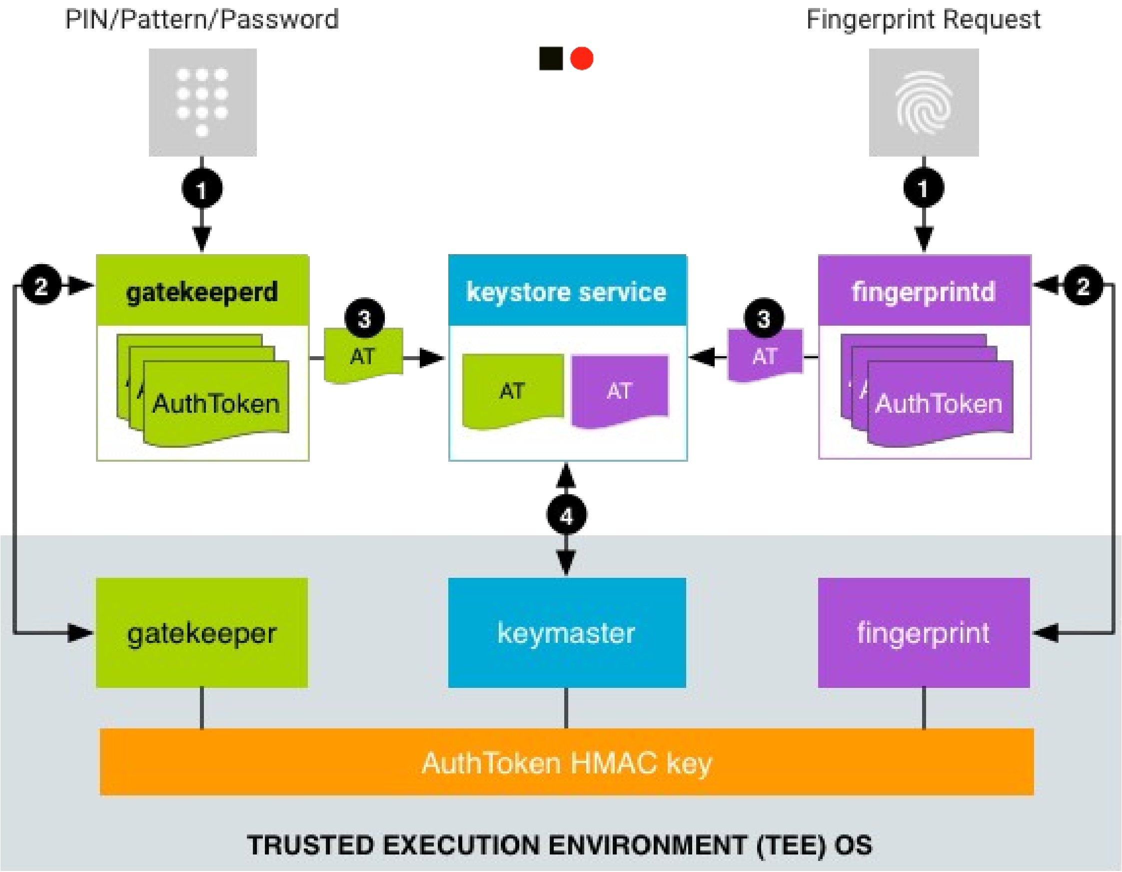
High level Encryption Workflow



Android authentication and keys

- **Gatekeeper: Authentication by Pin/Pass/Pattern**
  - Backend implementation in the TEE (Gatekeeper TA)
- **Fingerprint: Authentication by Fingerprint**
  - Backend implementation in the TEE (Fingerprint TA)
- **Keystore: Key management**
  - Backend implementation in the TEE (Keymaster TA)
- **Authentication tokens**
  - Signed by Gatekeeper TA or Fingerprint TA
  - Used by the Keystore to unwrap keys

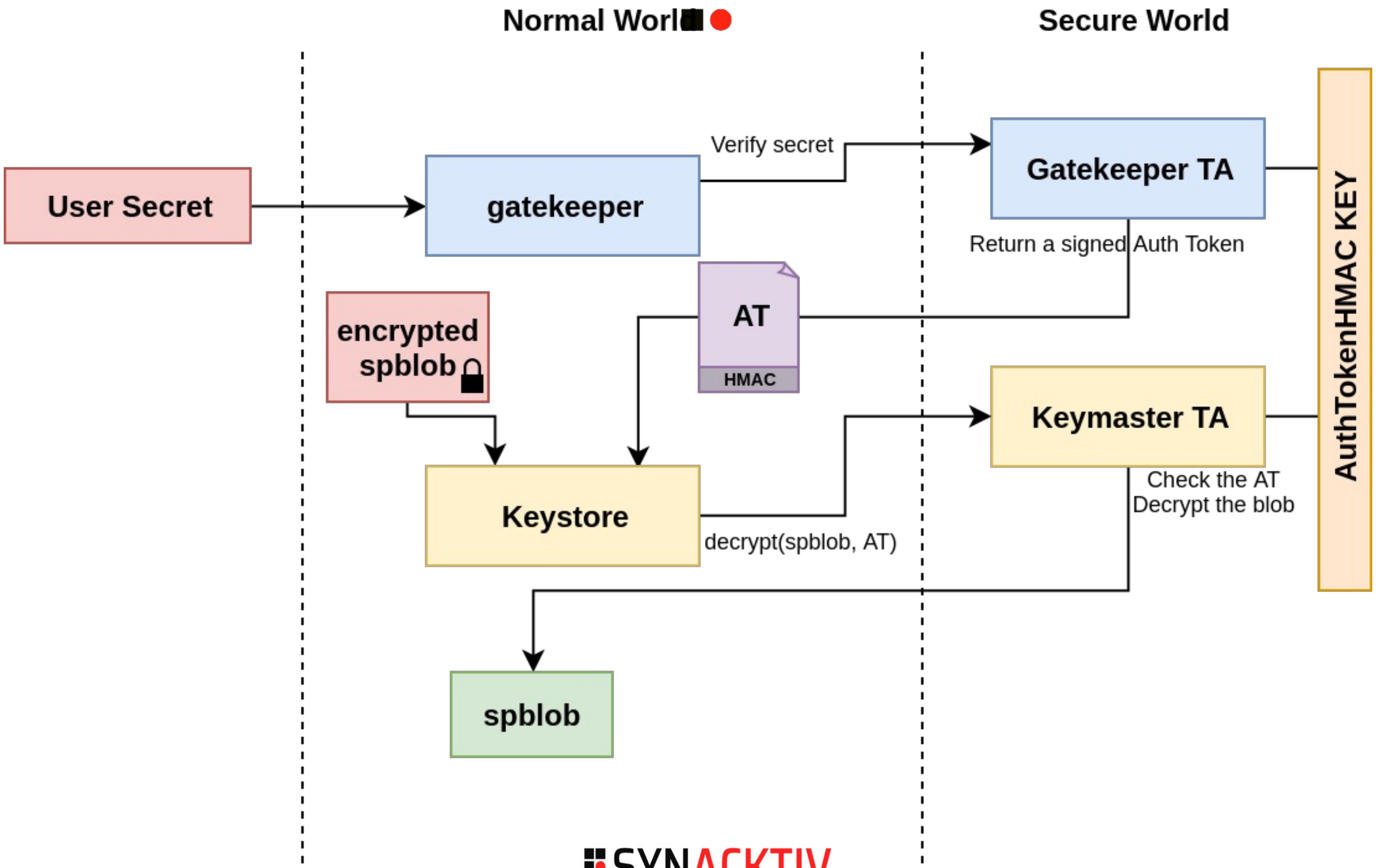
AuthToken and Keys



AT=AuthToken

https://source.android.com/security/authentication

spblob TEE decryption



Attacker point of view

- **Several vulnerabilities are needed to break the encryption**
- **For online brute force (on the device)**
  - Compromise the Normal World
  - Bypass the TEE anti bruteforce mechanism
- **For offline brute force (out of the device)**
  - Compromise the Normal World
  - Compromise the Secure World to extract spblob encryption key
- **Even through all assets were extracted, the brute force hash rate will be limited by the derivation functions (scrypt, sha512)**

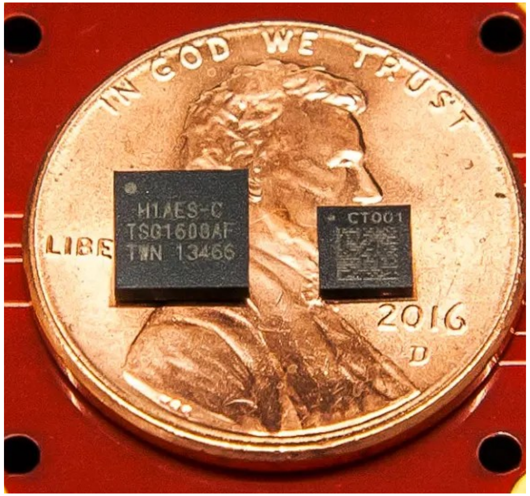
Attack Surface

- **Vulnerabilities in early boot stages break secrets protection**
- **TEE attack surface is big**
  - TEE Kernel
  - Indirect path using other Trusted Applications
  - Secure Monitor

Encryption with Secure Element

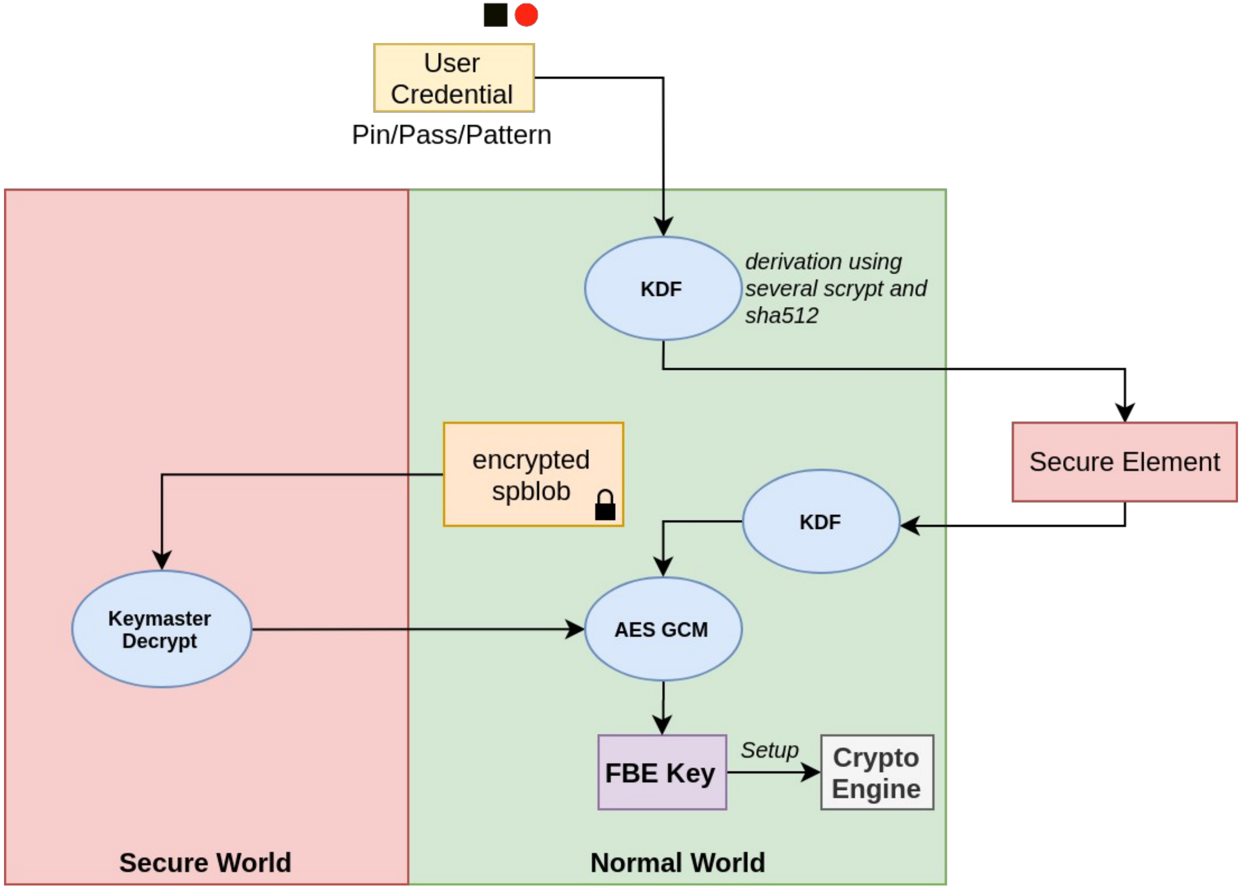
Secure Element

- **Some devices improve protection using SE**
- **Secure Element: External secure chip**
  - Microcontroller with high level security design
  - Connected to CPU by I2C or SPI bus
- **Used to store secrets**
  - HW crypto features (AES, Hashes)
  - Minimal attack surface
- **Tamper-resistant**
- **Protection against hardware attacks**



Google Titan and Titan M

Encryption with SE



Attack Surface

- ~~**Vulnerabilities in early bootstages break assets protection**~~
- ~~**TEE attack surface is big**~~
  - ~~TEE Kernel~~
  - ~~Other Trusted Applications~~
  - ~~Secure Monitor~~
- **Secrets are now safe even with a main CPU BootRom vulnerability!**
- **Secure Element attack surface is very limited!**

Conclusion

- **The encryption model proposed by Android is well designed and built upon hardware protections**
- **A single vulnerability should not break encryption**
  - Except BootRom vulnerabilities if no SE
- **SE offers a physical separation with strong security design**
- **Weaknesses**
  - After a complete boot, FBE keys are manipulated by the kernel ...
  - Final implementation is done by vendors
    - No guarantee that Android guidelines are respected

References

- Android Encryption
- Android Authentication
- SyntheticPasswordManager.java

SYNACKTIV

https://www.linkedin.com/company/synacktiv
https://twitter.com/synacktiv
Our publications: https://synacktiv.com