Dynamic Root of Trust and Trusted Execution

Prof. Dr.-Ing. Ahmad-Reza Sadeghi
System Security Lab
Technische Universität Darmstadt (CASED)
Germany
Winter Term 2016/17
Dynamic Root of Trust

- Problem of legacy
  - Incompatible or sloppy BIOS
  - Chain of trust is too long, too static

- AMD/Intel introduce **Dynamic RTM (DRTM)**
  - CPU resets into clean state to load OS/App
  - Essentially starting a new Chain of Trust
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
Dynamic Root of Trust

- Reset Chain of Trust, cut out BIOS and boot loader
- Boots clean OS out of compromised system (!)
“SENDER“ CPU instruction issues the process
  - Resets CPU and Chipset

Resets TPM PCRs 17-23

CPU executes Authenticated Code module (AC mod)
  - AC mod is signed by vendor (Intel)
  - AC mod is measured into PCR 17
  - AC mod reinitializes CPU and Chipset

Payload is measured by AC mod
  - Payload measurement stored in PCR 18
DRTM Security (1)

- **System Management Mode (SMM, “Ring -2”)**
  - Runtime, low-level hardware management
  - Arbitrary code execution while OS is suspended
  - BIOS is expected to initialize SMM and then lock access

- **Active Management Technology (AMT, “Ring -3”)**
  - RISC CPU inside the mainboard chipset
  - Dedicated link to your RAM and network card
  - Arbitrary code execution while OS is running
    - Even if disabled in BIOS
DRTM Security (2)

- SMM and AMT are not validated/supervised by TXT
  - BIOS is supposed to secure access and lock registers
  - Compromised SMM or AMT can manipulate AC or TXT payload
  - As of 2006, BIOS locks access to SM-RAM
  - TXT needs secure BIOS!
  - DRTM needs SRTM?!
DRTM Security (2)

- SMM and AMT are not validated/supervised by TXT
  - BIOS is supposed to secure access and lock registers
    - Compromised SMM or AMT can manipulate AC or TXT payload
    - As of 2006, BIOS locks access to SM-RAM
    - TXT needs secure BIOS!
    - DRTM needs SRTM!?
DRTM Security (2)

- SMM and AMT are not validated/supervised by TXT
  - BIOS is supposed to secure access and lock registers

- Compromised SMM or AMT can manipulate AC or TXT payload
- As of 2006, BIOS locks access to SM-RAM
- TXT needs secure BIOS!
- DRTM needs SRTM!?
DRTM Security (2)

- SMM and AMT are not validated/supervised by TXT
  - BIOS is supposed to secure access and lock registers

- Compromised SMM or AMT can manipulate AC or TXT payload
- As of 2006, BIOS locks access to SM-RAM
- TXT needs secure BIOS!
- DRTM needs SRTM!?
DRTM Security (2)

- SMM and AMT are not validated/supervised by TXT
  - BIOS is supposed to secure access and lock registers

- Compromised SMM or AMT can manipulate AC or TXT payload
- As of 2006, BIOS locks access to SM-RAM
- TXT needs secure BIOS!
- DRTM needs SRTM!? 
DRTM Security (2)

- SMM and AMT are not validated/supervised by TXT
  - BIOS is supposed to secure access and lock registers

- Compromised SMM or AMT can manipulate AC or TXT payload
- As of 2006, BIOS locks access to SM-RAM
- TXT needs secure BIOS!
- DRTM needs SRTM!?
SMM and AMT are not validated/supervised by TXT

- BIOS is supposed to secure access and lock registers

- Compromised SMM or AMT can manipulate AC or TXT payload
- As of 2006, BIOS locks access to SM-RAM
- TXT needs secure BIOS!
- DRTM needs SRTM!?
SMM and AMT are not validated/supervised by TXT

- BIOS is supposed to secure access and lock registers

- Compromised SMM or AMT can manipulate AC or TXT payload
- As of 2006, BIOS locks access to SM-RAM
- TXT needs secure BIOS!
- DRTM needs SRTM!? 
Trusted GRUB

- Root of Trust is the CPU reset to TXT mode
- Chain of Trust:
  - CPU
  - GRUB boot loader
  - (Hypervisor)
  - Operating System
  - ...  
- Similar problems as with Static Root of Trust
  - Long chain of trust 

TXT Example: TrustedGRUB

http://www.sirrix.com/content/pages/trustedgrub_en.htm
Flicker: Trusted Execution Environment

- Trusted Execution Environment (TEE) at runtime
  - Run only a small part in the TEE
  - Leads to small TCB
  - Must be isolated from other software

- Flicker
  - Suspend operating system
  - Switch to TXT mode
  - Measure and execute code
  - Resume operating system
Transaction Security (Flicker)

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security
Transaction Security (Flicker)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)
Transaction Security (Flicker)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)
Transaction Security (Flicker)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)
Transaction Security (Flicker)

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security
Transaction Security (Flicker)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)
Transaction Security (Flicker)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)
Transaction Security (Flicker)

- Use TXT only for small critical functions like signature
- Output can be signed or otherwise bound to TPM

- Strong isolation for legacy OS without virtualization overhead
- Simple and stable apps Integrity = Security

(Flicker: McCune, Parno, Perrig, Reiter, Isozaki. An Execution Infrastructure for TCB Minimization, EuroSys 2008)
Transaction Security (Flicker)

- Flicker framework is available for Intel TXT, AMD SVM, Windows, Linux
- First prototype apps:
  - Transaction confirmation, e.g., in online shopping
  - Software-Smartcard
    - Cryptographic operations executed in “TXT-mode”
    - PKCS#11 Interface
      - Usable with legacy software
      - E.g., Mail signing with Thunderbird
Disadvantages of TXT/SVM

- DRTM with TXT/SVM still contains huge software stack
  - How to verify measurements
  - Multiple open problems in cloud scenario
    - Provider doesn’t want to reveal software configuration
    - Host machine identification through TPM keys

- Isolating only the security critical part is desirable
  - Flicker never evolved to a practical solution
    - Substantial modification of the OS required
    - Switch to and from TXT/SVM is time consuming
    - Execution stalls entire system (interrupts disabled)
This slide set is based on slides provided by Prof. N. Asokan, Aalto University, Finland
Overview

- General model for mobile platform security
  - Key hardware security techniques and general architecture
- Example(s)
  - ARM TrustZone
  - Trusted Platform Module
What is a TEE?

**Trusted Execution Environment**

- Processor, memory, storage, peripherals
- Isolated and integrity-protected
- Chances are that:
  - You have devices with hardware-based TEEs in them!
  - But you don’t have (m)any apps using them
- From the “normal” execution environment (Rich Execution Environment)
TEE Overview

1. Platform integrity ("boot integrity")
2. Secure storage
3. Isolated execution
4. Device identification
5. Device authentication
Platform Integrity

Boot code certificate
- Boot code hash

Certified by device manufacturer: \(\text{Sig}_{SK_M}(H(\text{boot code}))\)

Legend
- Trust anchor (Hardware)
- Trust anchor (Code)
- TEE code
- External certificate

Mobile device hardware TCB

Verification root

Cryptographic mechanisms

Volatile memory
- Device key \(K_D\)
- Base identity
- Non-volatile memory
- TEE management
- Trusted Application (TA)
- Secure storage and isolated execution
- Device identification

Platform integrity

Launch boot code
Platform Integrity

Boot code certificate
- Boot code hash

Certified by device manufacturer: \(\text{Sig}_{SK_M}(H(\text{boot code}))\)

Device manufacturer public key: \(PK_M\)

Mobile device hardware TCB

Verification root

Cryptographic mechanisms

Volatile memory
- Boot sequence

Stores measurements for authenticated boot

Device key \(K_D\)

Signature verification algorithm

Secure storage and isolated execution

Device identification

Launch boot code

Legend
- Trust anchor (Hardware)
- Trust anchor (Code)
- TEE code
- External certificate

Platform integrity

Device manufacturer public key: \(PK_M\)

Signature verification algorithm

Stores measurements for authenticated boot
Secure Storage

Sealed-data = AuthEnc_{K_D}(data | ...)

Legend
- Trust anchor (Hardware)
- Trust anchor (Code)
- TEE code
- External certificate

Mobile device hardware TCB
- Verification root
- Volatile memory
- Boot sequence
- Trusted Application (TA)
- TEE management

Cryptographic mechanisms
- Device key K_D
- Non-volatile memory
- Encryption algorithm
- Protected memory
- Rollback protection
- Platform integrity
- Device identification
Isolated Execution

TEE code
Secure storage and isolated execution
Mobile device hardware TCB

Legend

Trust anchor (Hardware)
Trust anchor (Code)
TEE code
External certificate

Certified by device manufacturer

TA code certificate
TA code hash

Verification root
Cryptographic mechanisms
Volatile memory

Device key $K_D$
Non-volatile memory

Controls TA execution
Platform integrity

Boot sequence

Trusted Application (TA)
TEE management

Secure storage and isolated execution

Device identification

Platform integrity

Base identity

TEE Entry from Rich Execution Environment
Device Identification

Mobile device hardware TCB

Verification root

Cryptographic mechanisms

- Volatile memory
- Boot sequence
- Trusted Application (TA)
- TEE management
- Device key $K_D$
- Non-volatile memory
- Secure storage and isolated execution

Legend

- Trust anchor (Hardware)
- Trust anchor (Code)
- TEE code
- External certificate

Multiple assigned identities (Certified by device manufacturer)

Identity certificate

- Base identity
- Assigned identity

Platform integrity

Device identification

One fixed device identity
Device Authentication (and Remote Attestation)

Legend

- Trust anchor (Hardware)
- Trust anchor (Code)
- TEE code
- External certificate

Mobile device hardware TCB

- Verification root
- Cryptographic mechanisms
- Volatile memory
- Device key $K_D$

External trust root

Device certificate
- Identity
- Device public key $PK_D$

Device authentication

Platform integrity

Secure storage and isolated execution

Boot sequence

Trusted Application (TA)

Non-volatile memory

TEE management
Device Authentication
(and Remote Attestation)

Legend
- Trust anchor (Hardware)
- Trust anchor (Code)
- TEE code
- External certificate

Mobile device hardware TCB

Verification root

Cryptographic mechanisms

Volatile memory

Device key $K_D$

Device certificate

Identity

Device public key $PK_D$

External trust root

Platform integrity

Secure storage and isolated execution

Boot sequence

Trusted Application (TA)

TEE management

Issued by device manufacturer

Used to protect/derive signature key

Sign system state in remote attestation
Hardware Security Mechanisms (recap)

1. Platform integrity
   - Secure boot
   - Authenticated boot

2. Secure storage

3. Isolated execution
   - Trusted Execution Environment (TEE)

4. Device identification

5. Device authentication
   - Remote attestation

**Legend**

- Trust anchor (Hardware)
- Trust anchor (Code)
- TEE code
- External certificate

**Mobile device hardware TCB**

- Verification root
- Cryptographic mechanisms
  - Volatile memory
  - Trusted application
  - TEE mgmt layer
  - Secure storage and isolated execution
- Device key $K_D$
- Base identity
- Non-volatile memory

**Device certificate**

- Identity
- Device pub key $PK_D$

**Platform integrity**

- Launch boot code
  - TEE Entry from Rich Execution Environment

**Platform integrity**

- Device authentication
Device hardware and firmware with TEE support

### TEE System Architecture

- **Architectures with single TEE**
  - ARM TrustZone
  - TI M-Shield
  - Smart card
  - Crypto co-processor
  - Trusted Platform Module (TPM)

- **Architectures with multiple TEEs**
  - Intel SGX
  - TPM (and “Late Launch”)
  - Hypervisor

---

![Diagram of TEE System Architecture](image-url)
External Secure Element (TPM, smart card)

Embedded Secure Element (smart card)

Processor Secure Environment (TrustZone, M-Shield)

Legend:
SoC: system-on-chip
OTP: one-time programmable
System on chip (SoC)

- On-chip memory
- Boot ROM
- Main CPU
- Memory controller
- Memory controller
- Interrupt controller
- Off-chip/main memory (DDR)
- Peripherals (touchscreen, USB, NFC...)

SoC internal bus (carries status flag)

Secure World and Normal World

TrustZone hardware architecture

ARM TrustZone Architecture
ARM TrustZone Architecture

System on chip (SoC)

- On-chip memory
- Modem
- Access control hardware
- Boot ROM
- Main CPU
- Access control hardware
- Memory controller
- Access control hardware
- Memory controller
- Interrupt controller
- Off-chip/main memory (DDR)
- Peripherals (touchscreen, USB, NFC...)

SoC internal bus (carries status flag)

TrustZone system architecture

- Normal world (REE)
  - Mobile OS
  - App
  - App

- Secure world (TEE)
  - Trusted OS
  - Trusted app
  - Trusted app

Device hardware

- TEE entry

TrustZone hardware architecture
TrustZone Overview

Normal World (NW)

User mode

- User

Privileged mode

- Supervisor

Secure World (SW)

- User

- Supervisor

- Monitor

Secure Monitor Call (SMC)

SCR.NS := 1

SCR.NS := 0

Boot sequence

Address space controllers

TZ-aware MMU

Legend:
MMU: memory management unit

On-chip ROM

On-chip RAM

Main memory

physical address range

SW RW
NW NA

SW RO
NW WO

SW RW
NW RW
1. Boot begins in Secure World Supervisor mode (set access control)

Boot sequence

Secure World Supervisor

On-chip ROM

On-chip RAM

Main memory (off-chip)
1. Boot begins in Secure World Supervisor mode (set access control)

Secure World Supervisor

On-chip ROM

On-chip RAM

Main memory (off-chip)
TrustZone Example (1/2)

1. Boot begins in Secure World Supervisor mode (set access control)

   Boot sequence → Secure World Supervisor

   code (trusted OS) device key

2. Copy code and keys from on-chip ROM to on-chip RAM

   Secure World Supervisor

   On-chip ROM

   On-chip RAM

   Main memory (off-chip)
TrustZone Example (1/2)

1. Boot begins in Secure World Supervisor mode (set access control)

   Boot sequence
   
   Secure World Supervisor
   
   code (trusted OS) device key
   
2. Copy code and keys from on-chip ROM to on-chip RAM

   Secure World Supervisor
   
   On-chip ROM
   
   SW NA
   
   NW NA
   
3. Configure address controller (protect on-chip memory)

   Secure World Supervisor
   
   On-chip RAM
   
   SW RW
   
   NW NA
   
   Main memory (off-chip)
   
   SW RW
   
   NW NA
1. Boot begins in Secure World Supervisor mode (set access control)

Boots sequence

2. Copy code and keys from on-chip ROM to on-chip RAM

3. Configure address controller (protect on-chip memory)

4. Prepare for Normal World boot
1. Boot begins in Secure World Supervisor mode (set access control)

2. Copy code and keys from on-chip ROM to on-chip RAM

3. Configure address controller (protect on-chip memory)

4. Prepare for Normal World boot

---

**Secure World Supervisor**

**On-chip ROM**
- SW NA
- NW NA

**On-chip RAM**
- SW RW
- NW NA

**Main memory (off-chip)**
- SW RW
- NW RW
TrustZone Example (2/2)

5. Jump to Normal World Supervisor for traditional boot

An ordinary boot follows:
Set up MMU, load OS, drivers...
5. Jump to Normal World Supervisor for traditional boot

Secure World Supervisor \rightarrow Normal World Supervisor

SCR.NS\rightarrow1

An ordinary boot follows: 
Set up MMU, load OS, drivers...

6. Set up trusted application execution

Normal World Supervisor

User

On-chip ROM

SW NA
NW NA

On-chip RAM

SW RW
NW NA

Main memory (off-chip)

SW RW
NW RW
5. Jump to Normal World Supervisor for traditional boot

Secure World Supervisor → Normal World Supervisor

SCR.NS→1

An ordinary boot follows:
Set up MMU, load OS, drivers...

6. Set up trusted application execution

Normal World User → Normal World Supervisor

On-chip ROM

SW NA
NW NA

On-chip RAM

SW RW
NW NA

Main memory (off-chip)

SW RW
NW RW

trusted app and parameters
5. Jump to Normal World Supervisor for traditional boot

6. Set up trusted application execution

7. Execute trusted application

An ordinary boot follows: Set up MMU, load OS, drivers...

trusted app and parameters
5. Jump to Normal World Supervisor for traditional boot

6. Set up trusted application execution

7. Execute trusted application
Secure Boot vs. Authenticated Boot

Secure Boot:
- Firmware
  - Boot block
    - OS Kernel
      - checker
        - pass/fail

Authenticated Boot:
- Firmware
  - Boot block
    - OS Kernel
      - measurer
        - state
Secure Boot vs. Authenticated Boot

**Why?**

**How will you implement a checker?**
- hardcode \( H(\text{boot code}) \) as reference value in checker (in Firmware)?

**Why?**

State can be:
- bound to stored secrets (sealing)
- reported to external verifier (remote attestation)
Mobile TEE Deployment

- TrustZone support available in majority of current smartphones
- Mainly used for manufacturer internal purposes
  - Digital rights management, Subsidy lock...

- **APIs for developers?**
// create RSA key pair
Context ctx;
KeyPairGeneratorSpec spec = new KeyPairGeneratorSpec.Builder(ctx);
spec.setAlias("key1")
…
spec.build();

KeyPairGenerator gen = KeyPairGenerator.getInstance("RSA", "AndroidKeyStore");
gen.initialize(spec);
KeyPair kp = gen.generateKeyPair();

// use private key for signing
AndroidRsaEngine rsa = new AndroidRsaEngine("key1", true);
PSSSigner signer = new PSSSigner(rsa, …);
signer.init(true, …);
signer.update(signedData, 0, signedData.length);
byte[] signature = signer.generateSignature();

Android Key Store Implementation

- Selected devices
  - Android 4.3
  - Nexus 4, Nexus 7
- Keymaster operations
  - GENERATE_KEYPAIR
  - IMPORT_KEYPAIR
  - SIGN_DATA
  - VERIFY_DATA

Persistent storage on Normal World

What Protects Hardware Platform Security?

A well-known scientist (some say it was Bertrand Russell) once gave a public lecture on astronomy. He described how the earth orbits around the sun and how the sun, in turn, orbits around the center of a vast collection of stars called our galaxy. At the end of the lecture, a little old lady at the back of the room got up and said: "What you have told us is rubbish. The world is really a flat plate supported on the back of a giant tortoise." The scientist gave a superior smile before replying, "What is the tortoise standing on?" "You're very clever, young man, very clever," said the old lady. "But it's tortoises all the way down!“

- Stephen Hawking, in A Brief History of Time