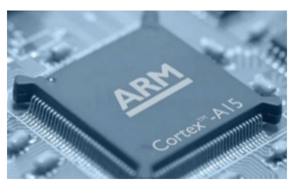
fTPM: A Software-only Implementation of a TPM Chip

Himanshu Raj, Stefan Saroiu, Alec Wolman, Ronald Aigner, Jeremiah Cox, Paul England, Chris Fenner, Kinshuman Kinshumann, Jork Loeser, Dennis Mattoon, Magnus Nystrom, David Robinson, Rob Spiger, Stefan Thom, David Wooten

Microsoft

Motivation

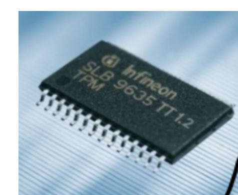
- Many systems in industry & research rely on TPMs
  - Bitlocker, trusted sensors, Chrome OS, etc…
- **Challenge**: Smartphones & tablets lack TPMs today
  - TPM: never designed to meet space, cost, power constraints
- Observation:



TrustZone®
Security Foundation by ARM®

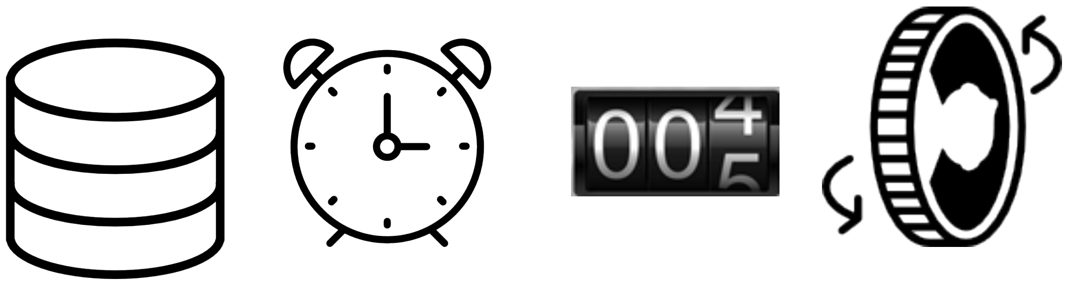
intel
INTEL® SOFTWARE GUARD EXTENSIONS (INTEL® SGX)

?



Big Problem

These CPU features omit several secure resources found on trusted hardware



Research Question

Can we overcome these limitations
to build systems whose security ~trusted hardware?

Answer: Yes

Contributions:

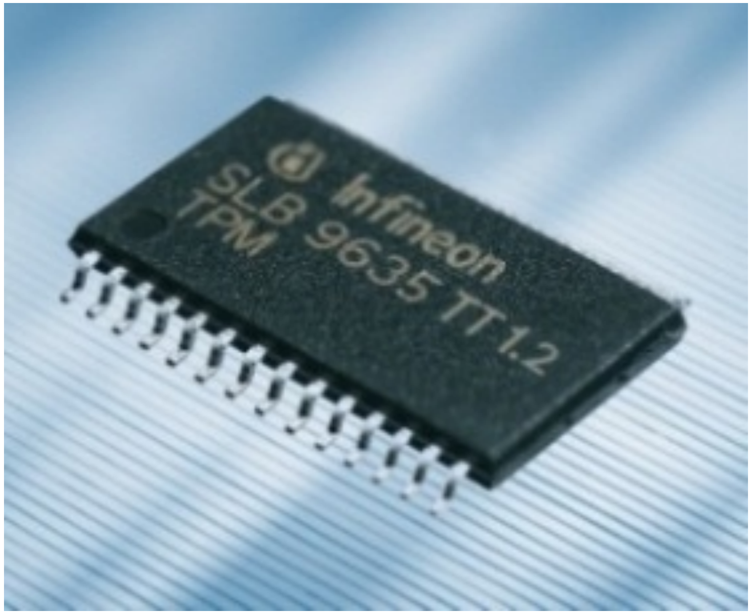
- 3 approaches to overcome TrustZone's limitations
  (lessons relevant to SGX also)
- Security analysis of fTPM vs TPM chips
- fTPM shipped millions of Microsoft Surface & WP

Outline

- Motivation
- Background on TPM
- ARM TrustZone and its shortcomings
- High-level architecture & threat model
- Overcoming TrustZone limitations: three approaches
- Performance evaluation
- Conclusions

What are TPMs?

- Hardware root of trust offering:
  - Strong machine identity
  - Software rollback prevention
  - Secure credentials store
  - Software attestation



What are TPMs good for?

- Shipped Products by Industry:
  - Protects “data-at-rest” (Google, Microsoft)
  - Prevents rollback (Google)
  - Virtual smart cards (Microsoft)
  - Early-Launch Anti-Malware (Microsoft)
- Research:
  - Secure VMs for the cloud [SOSP’11]
  - Secure offline data access [OSDI ‘12]
  - Trusted sensors for mobile devices [MobiSys ’11, SenSys ‘11]
  - Cloaking malware [Sec ‘11]

TPM: 1.0 → 1.1 → 1.2 → 2.0

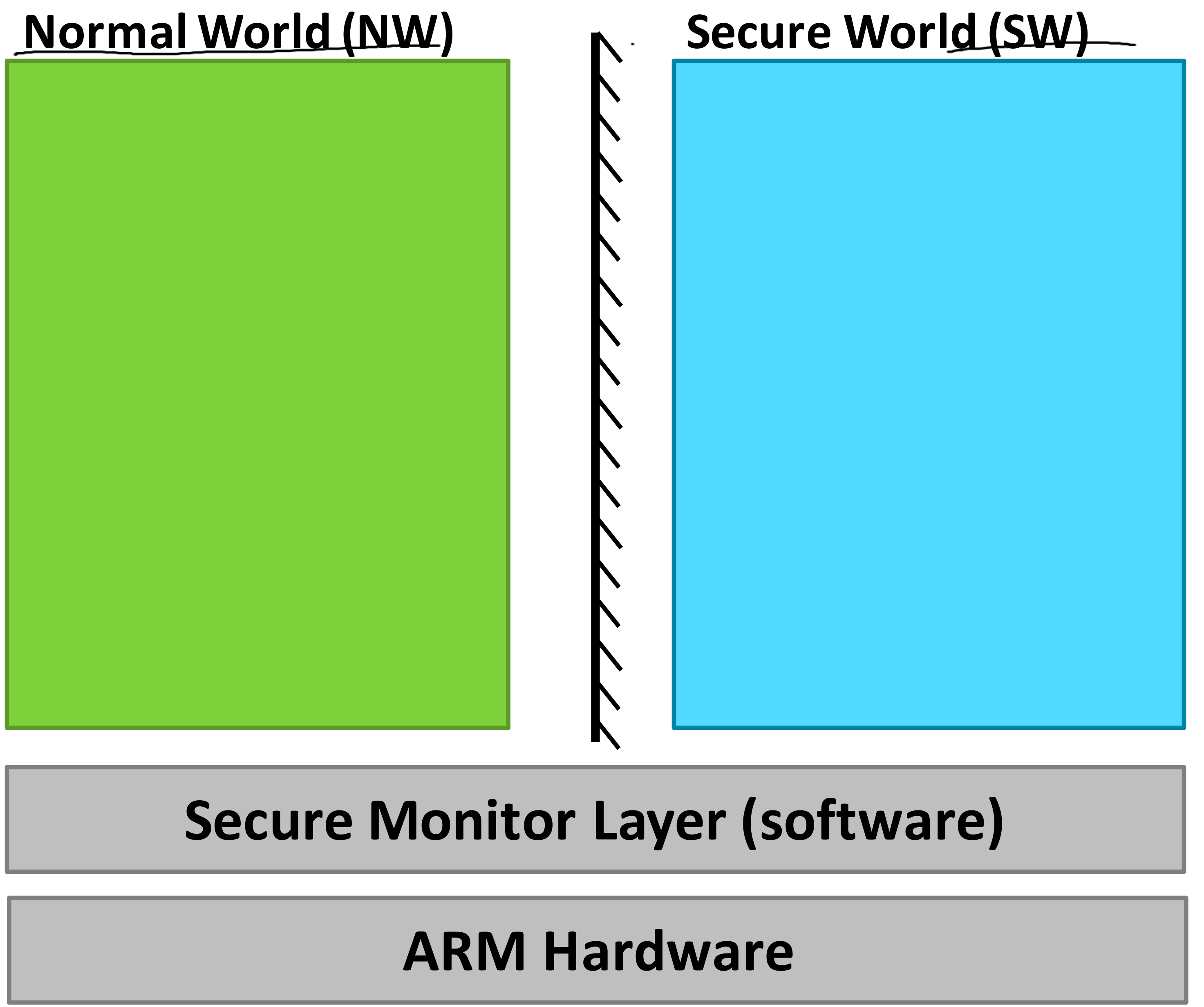
- **Late 1999**: TCPA is formed (IBM, HP, Intel, Microsoft, …)
- **2001**: TPM specification 1.0 is released
  - Never adopted by any hardware AFAIK
- **Late 2001**: TPM 1.1 is released
- **2002**: IBM Thinkpad T30 uses first discrete TPM chip
- **2003**: TCPA morphs into TCG
- **2007**: pin reset attack
- **2008**: TPM 1.2
  - Very popular, many hardware vendors built chips
- **2014**: TPM 2.0

New in TPM 2.0

- Newer cryptography
  - TPM 1.2: SHA-1, RSA
  - TPM 2.0: SHA-1, RSA, SHA-256, ECC
- TPM 2.0 provides a reference implementation
  - “the code is the spec”
- Much more flexible policy support
  - Read this as “more (useful) bells and whistles”

Outline

- Motivation
- Background on TPM
- ARM TrustZone and its shortcomings
- High-level architecture & threat model
- Overcoming TrustZone limitations: three approaches
- Performance evaluation
- Conclusions

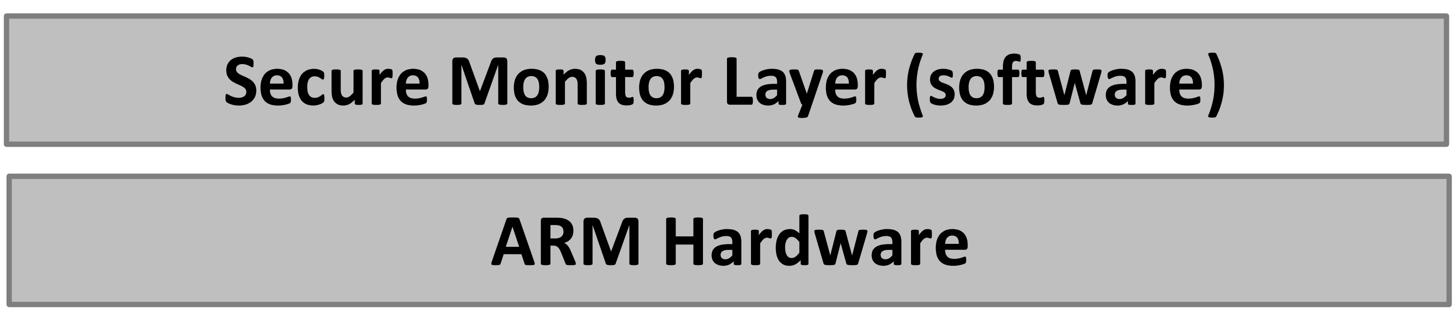


Normal World (NW)
Secure World (SW)
Secure Monitor Layer (software)
ARM Hardware

Booting Up

ARM Hardware

Booting Up



Booting Up

Allocates memory

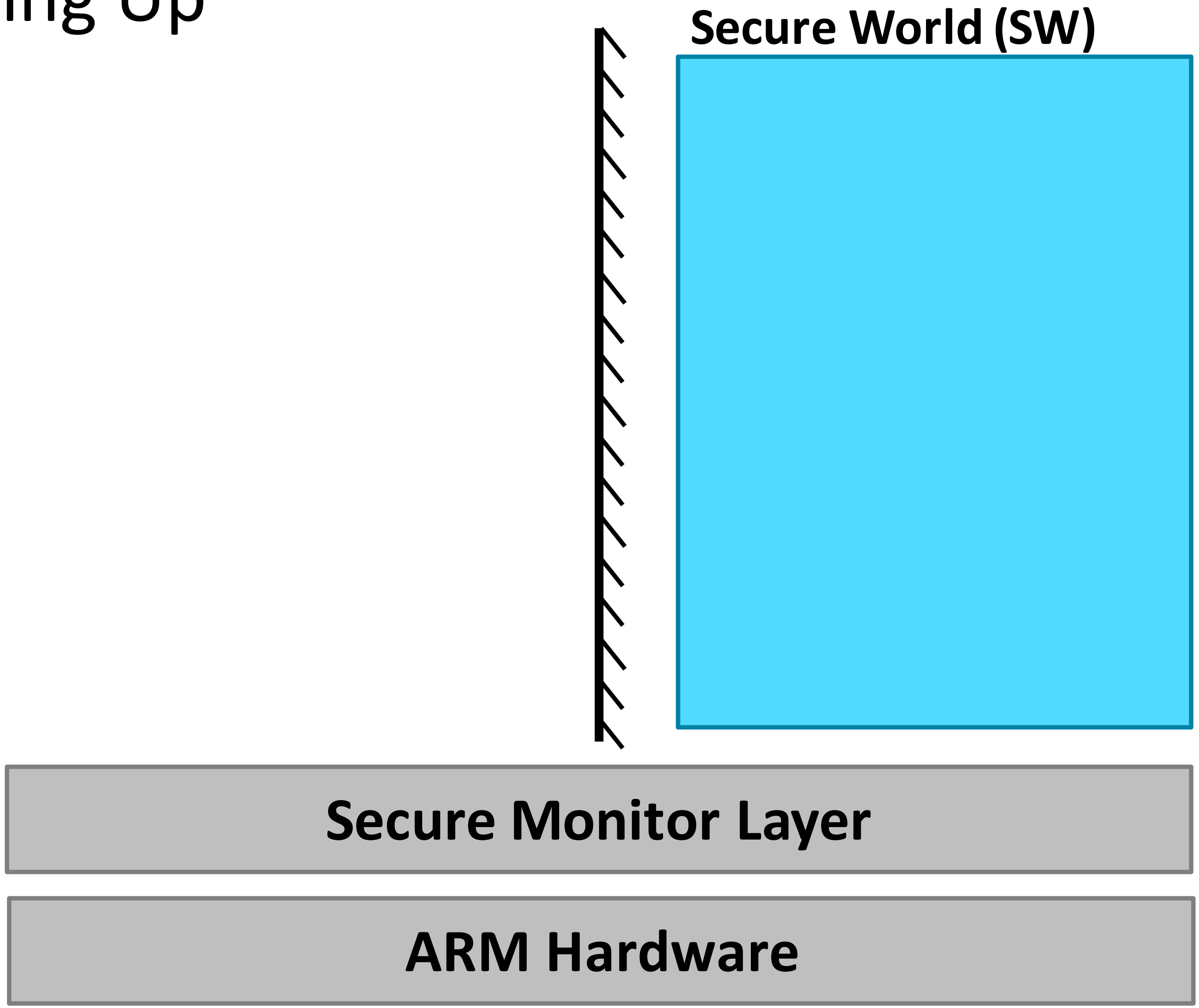
Restricts its access to Secure World-only

More setup…

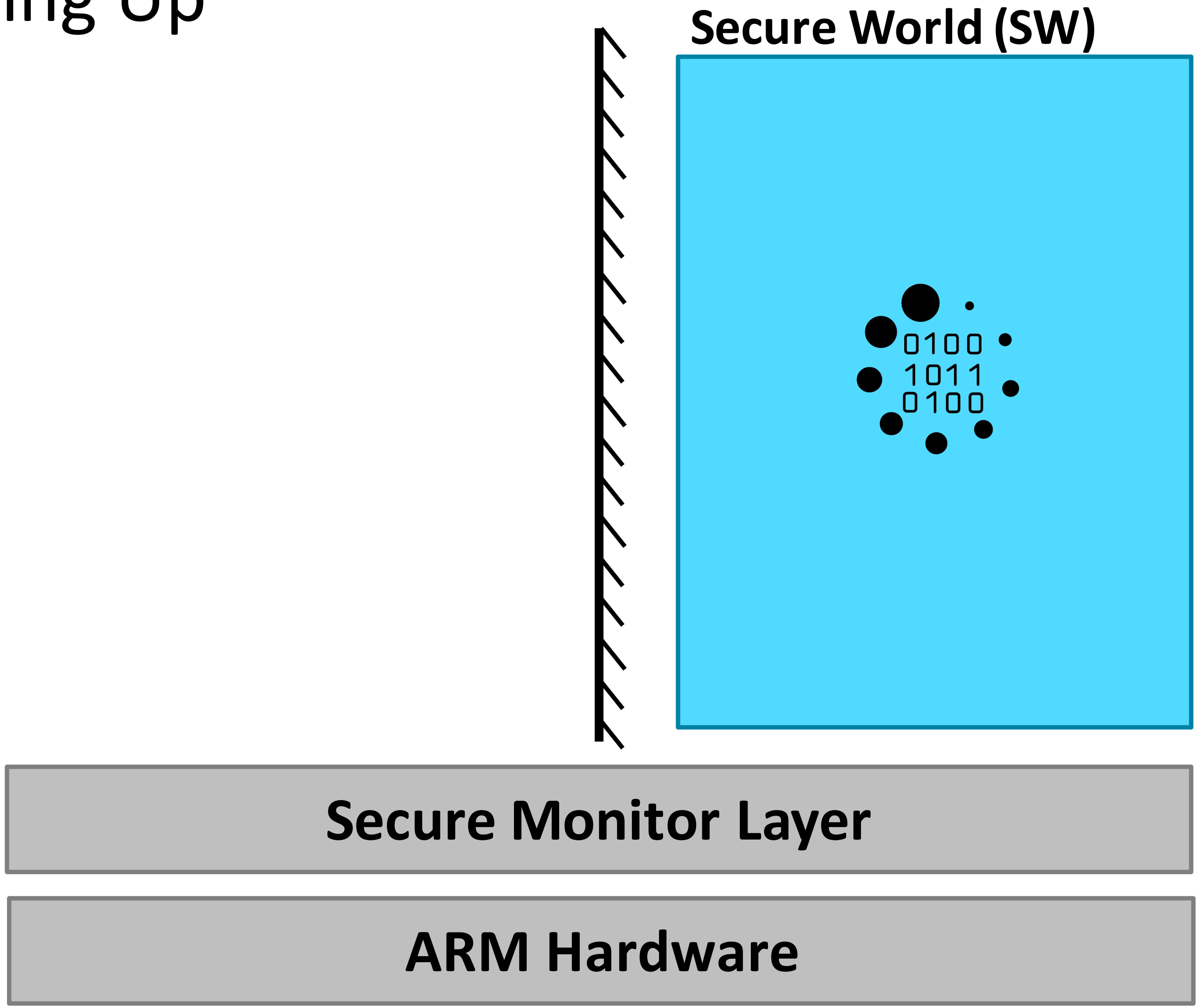
Secure Monitor Layer

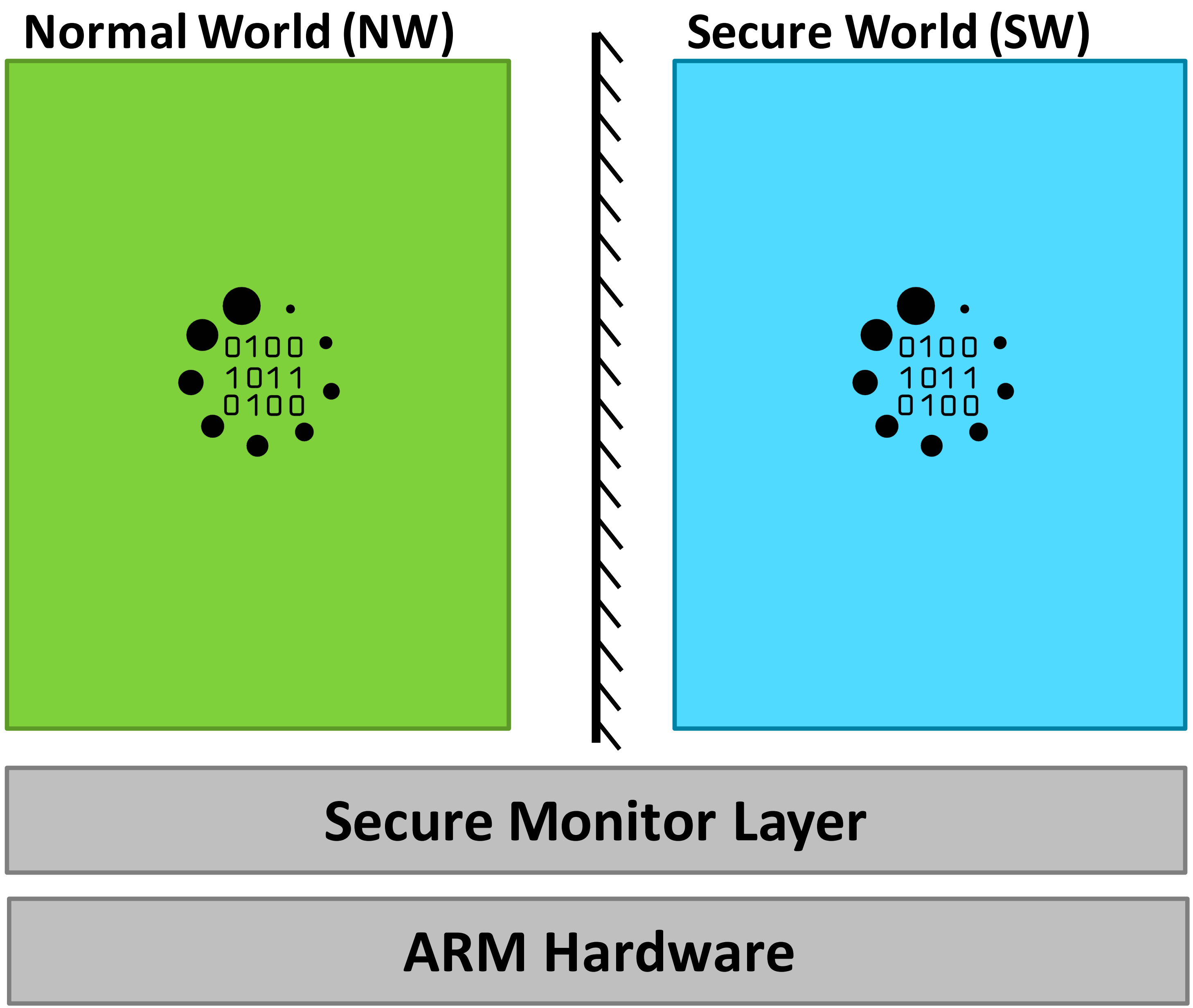
ARM Hardware

Booting Up



Booting Up

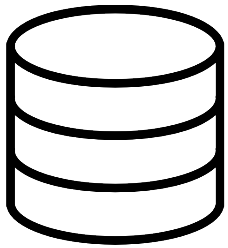

Secure World (SW)
0100
1011
0100
Secure Monitor Layer
ARM Hardware

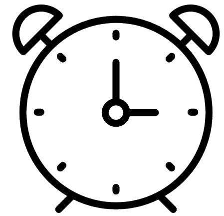

Normal World (NW)
0100
1011
0100
Secure World (SW)
0100
1011
0100
Secure Monitor Layer
ARM Hardware

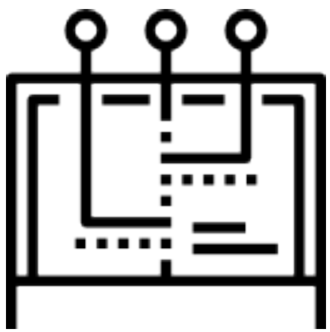
ARM TrustZone Properties

- Isolated runtime that boots first
- Curtained memory
- Ability to map interrupts delivered to Secure World
  - Secure monitor dispatches interrupts

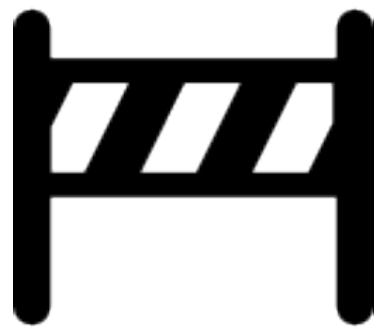
ARM TrustZone Limitations







Lack of virtualization

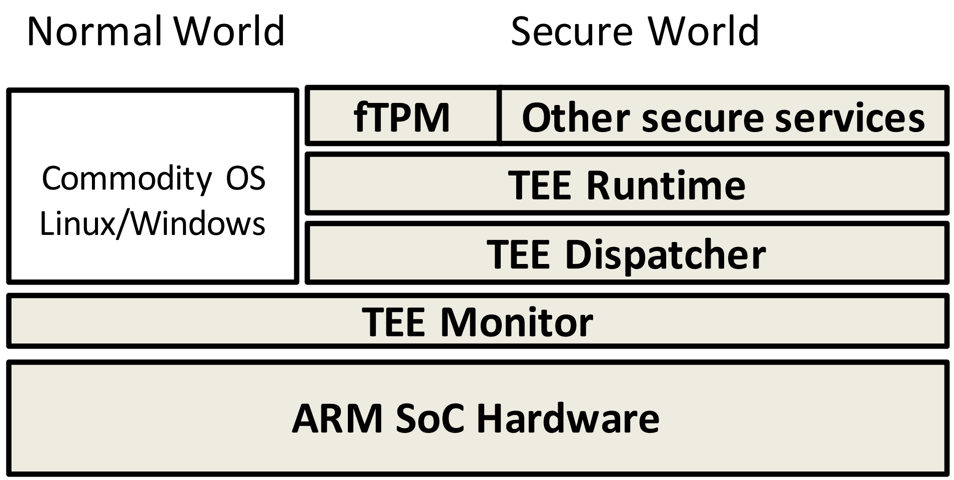


Lack of accessibility

Outline

- Motivation
- Background on TPM
- ARM TrustZone and its shortcomings
- High-level architecture & threat model
- Overcoming TrustZone limitations: three approaches
- Performance evaluation
- Conclusions

High-Level architecture



- TEE: trusted execution environment (small codebase)
  - Monitor, dispatcher, runtime
- Most hardware resources mapped to Normal World
  - For better perf.

Threat Model: What Threats are In-Scope?

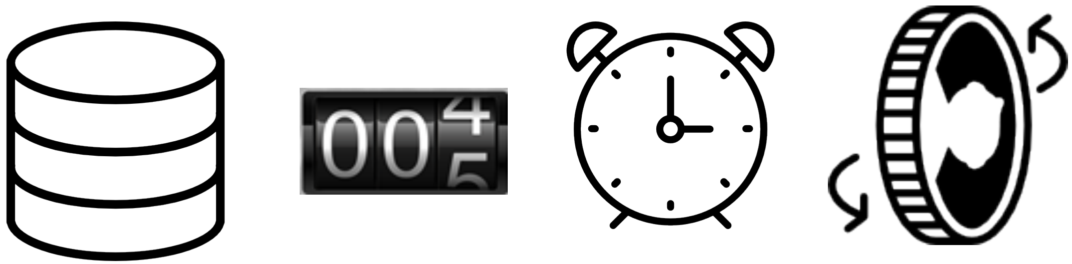
Goals	fTPM	TPM chip
Malicious software (e.g., malware, compromised OS)		
Time-based side-channel		
Cache-based side-channel		
Denial-of-Service		
Power analysis-based side-channel		
Memory attacks (e.g., coldboot, bus sniffing, JTAG)		

See “Memory Attacks” (ASPLOS 2015)

Outline

- Motivation
- Background on TPM
- ARM TrustZone and its shortcomings
- High-level architecture & threat model
- Overcoming TrustZone limitations: three approaches
- Performance evaluation
- Conclusions

ARM TrustZone Limitations



Helpful observation: huge ARM eco-system out there

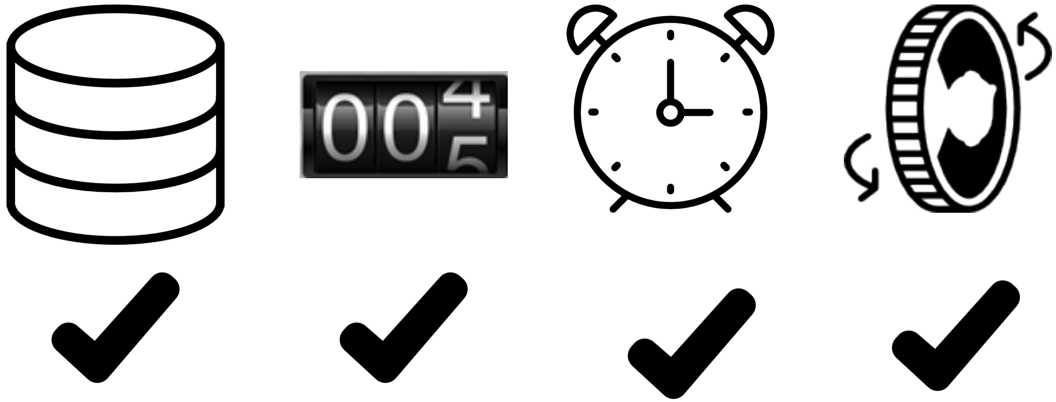
- eMMC controller present on many ARM SoCs
  - Has provisions for trusted storage
- Secure fuses: write-once, read-always registers
  - Can act as “seed” for deriving crypto keys
- Entropy for TrustZone can be added easily

ARM Eco-system Offers eMMC

- eMMC controllers can setup one partition as Replay-Protected Memory Block (RPMB)

- RPMB primitives:
  - One-time programmable authentication keys:
    - fTPM uses “seed” from secure fuse to generate auth. keys
    - fTPM writes auth. keys to eMMC controller upon provisioning
  - Authenticated reads and writes (uses internal counters)
  - Nonces

ARM TrustZone Limitations



eMMC & Secure fuses

Entropy

Timer & changed semantics of TPM commands

Three Approaches

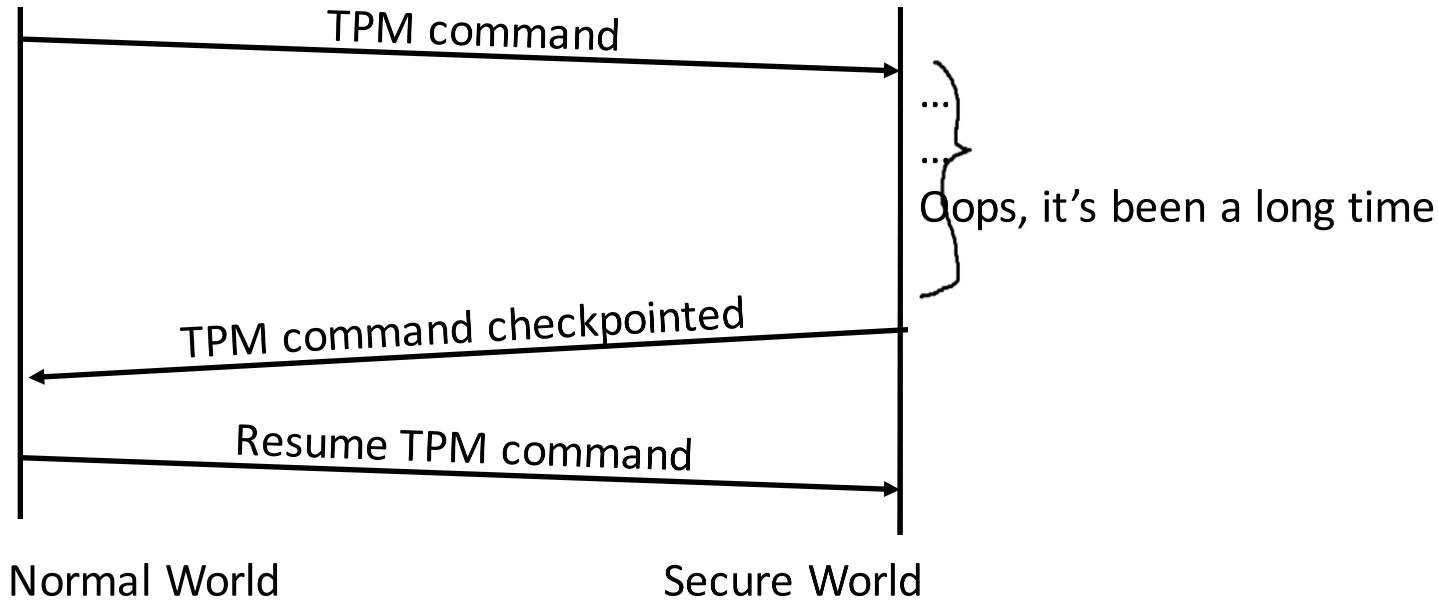
1. Provision additional trusted hardware
2. Make design compromises
3. Change semantics of TPM commands

Do not affect TPM's security!

Problem: Long-Running Commands

- Design requirements:
  - Code running in secure world must be minimal
    - e.g., TEE lacks pre-emptive scheduler
  - fTPM commands cannot be long-lived
    - Commodity OS “freezes” during fTPM command
- Creating RSA keys can take 10+ seconds on slow mobile devices!!!

Solution: Cooperative Checkpointing

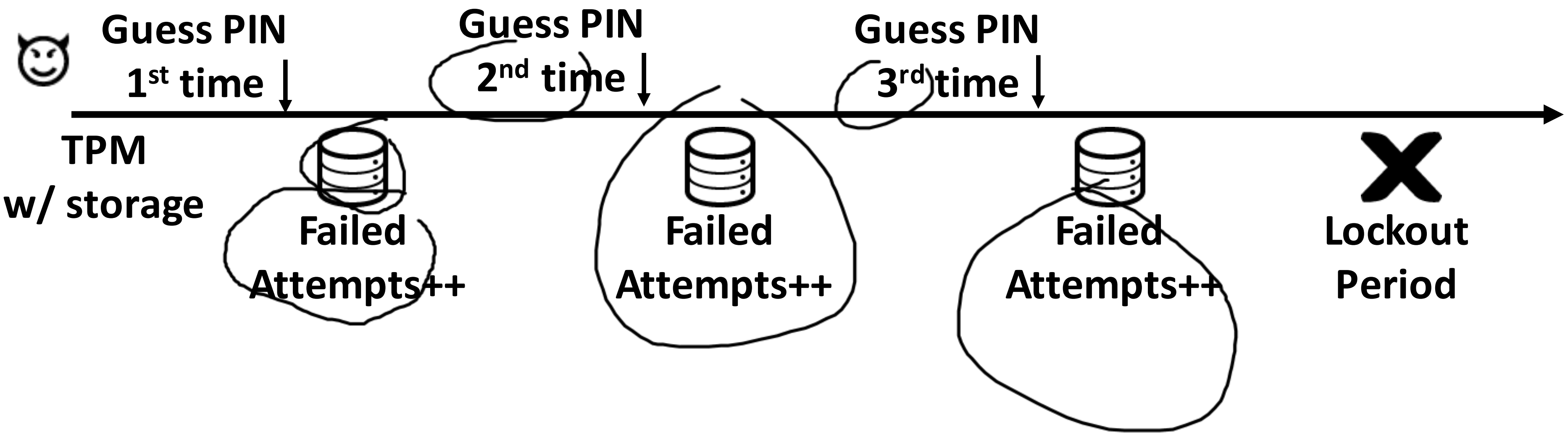


Three Approaches

1. Provision additional trusted hardware
2. Make design compromises
3. Change semantics of TPM commands

Do not affect TPM's security!

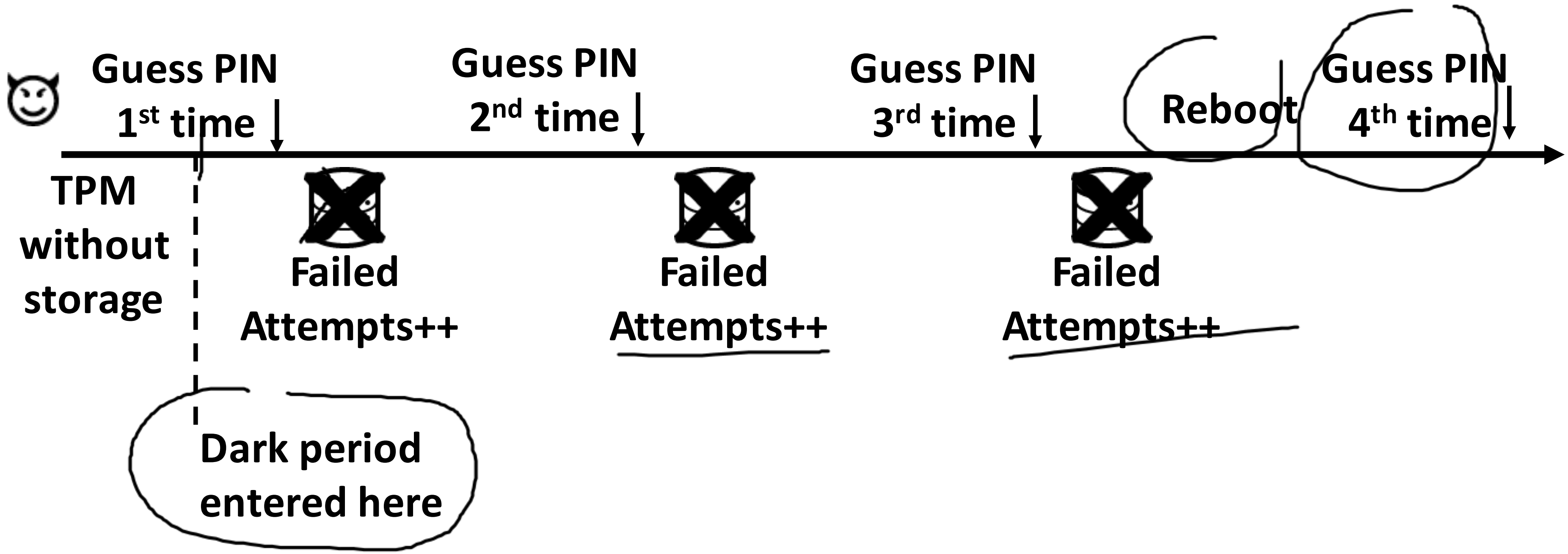
Background: TPM Unseal



Problem: Dark Periods

- During dark periods:
  - Problem: storage unavailable
  - Danger: TPM Unseal commands not safe
- Example of dark period: During boot:
  - Firmware (UEFI) finished running and unloaded
  - OS loader is running (OS not fully loaded)

Possible Attack during Dark Period



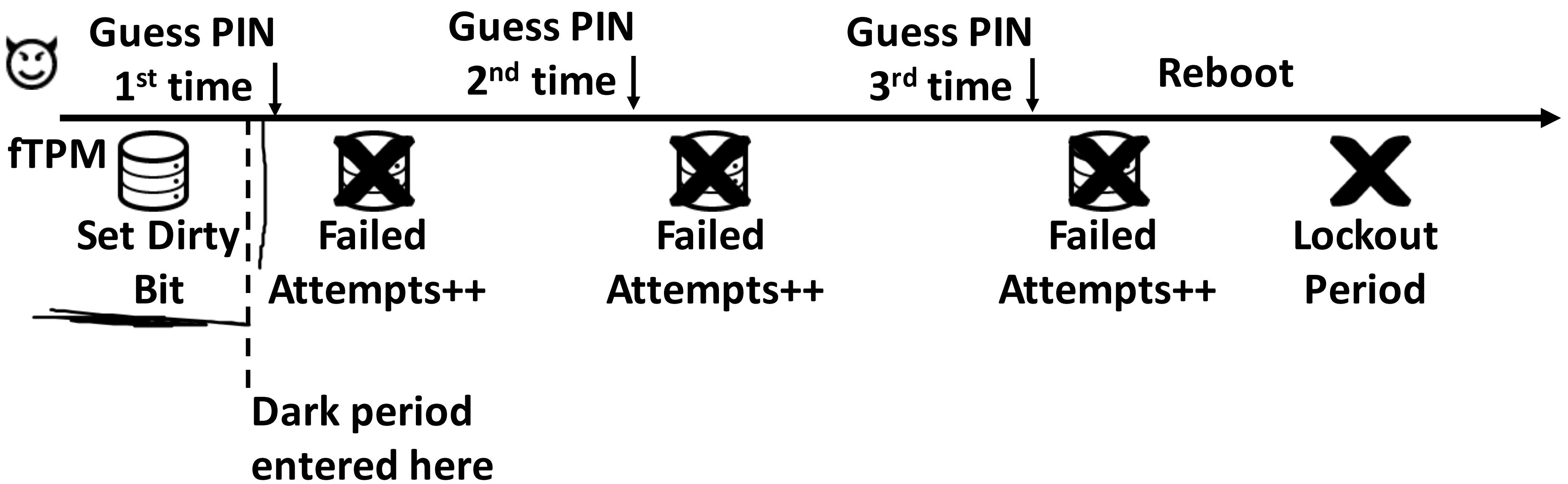
Solution: Dirty Bit

- Write dirty bit to storage before enter dark period
- If dark period exited, dirty bit is cleared

- If machine reboots during dark period, bit remains dirty
  - Possibility #1: Legitimate user reboots machine
  - Possibility #2: Attacker attempts to guess PIN

- Solution: Upon fTPM bootup, if bit dirty enter lockout

Dirty Bit Stops Attack



Outline

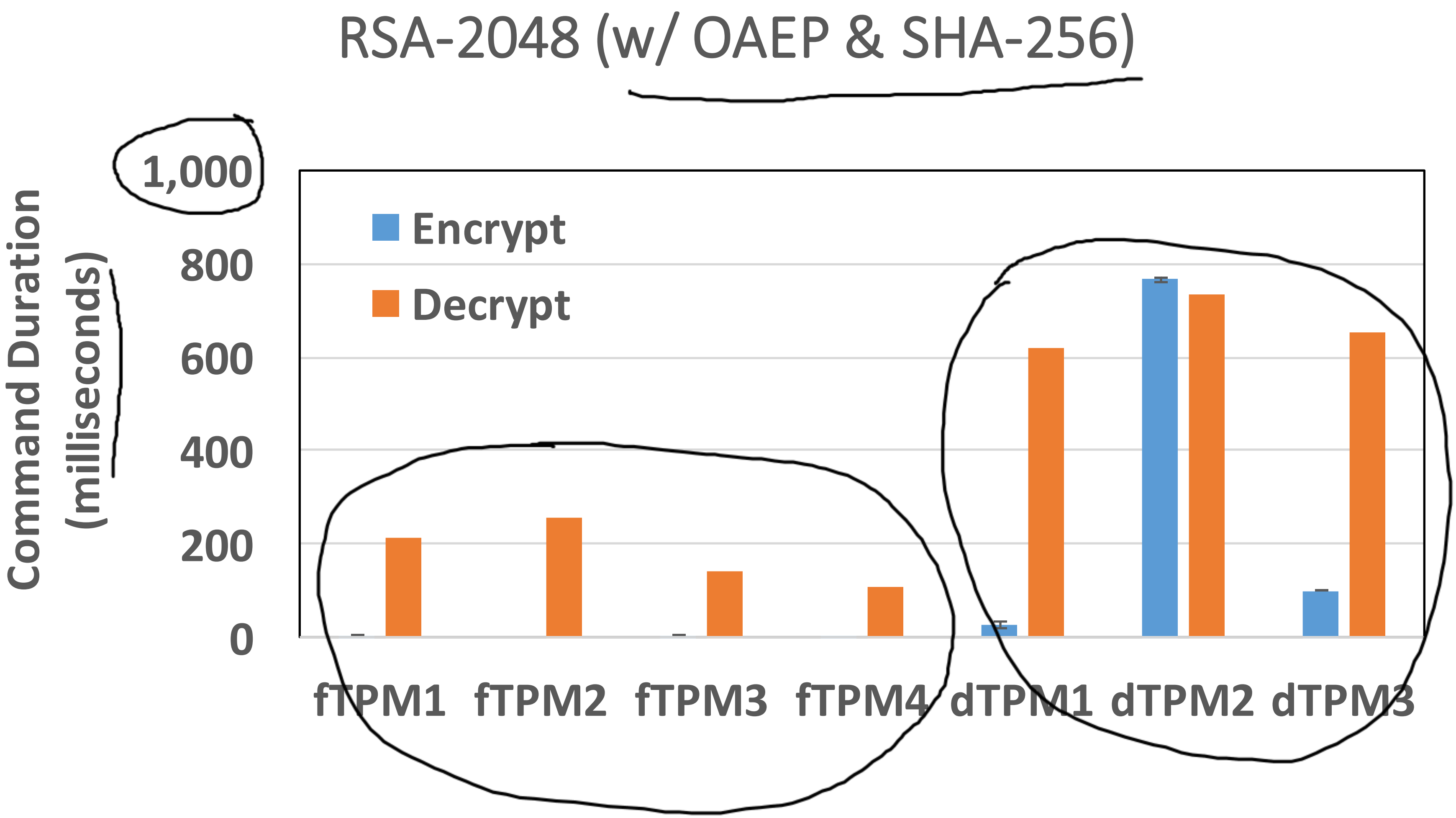
- Motivation
- Background on TPM
- ARM TrustZone and its shortcomings
- High-level architecture & threat model
- Overcoming TrustZone limitations: three approaches
- Performance evaluation
- Conclusions

Methodology

fTPM1	1.2 GHz Cortex-A7
fTPM2	1.3 GHz Cortex-A9
fTPM3	2 GHz Cortex-A57
fTPM4	2.2 GHz Cortex-A57
dTPM1	
dTPM2	
dTPM3	

- Instrumented and measured various TPM commands
  - Create RSA keys, seal, unseal, sign, verify, encrypt, decrypt

Result: fTPMs much faster than dTPMs



Conclusions

- fTPM leverages ARM TrustZone to build TPM 2.0 running in-firmware
- Three approaches to build fTPM:
  - Additional hardware requirements
  - Design compromises
  - Modify TPM semantics
- fTPMs offer much better performance than dTPMs

Discussion of SGX Limitations

- Lack of trusted storage, secure counters, and clock
  - Due to fundamental process limitations
- Lack of Intel eco-system (unlike ARM):
  - Intel needs to decide to equip their devices with eMMC
- One plus: SGX encrypts memory
  - No need to worry about memory attacks
- One minus: SGX can only run ring-3 code
  - No secure interrupts available
  - More concerns about side-channel attacks

Questions?

- ssaroiu@microsoft.com