Protecting Data on Smartphones & Tablets with Trusted Computing

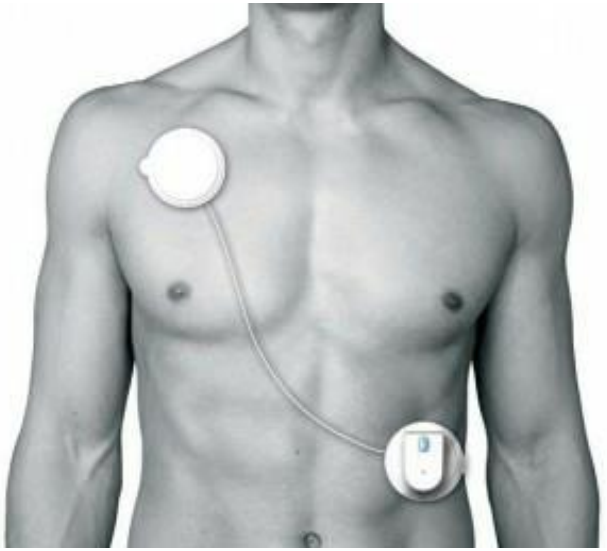
Stefan Saroiu

Microsoft Research (Redmond)

Smartphones have displaced PCs as the primary computing device

Smartphones Store Sensitive Data

Sensor Readings Have Value



Implications

- High value of smartphone data creates incentives for "bad" guys:
  - 3rd-parties want to steal data
  - 1st-parties want to fabricate/alter data

Data is under attack from malware, apps, or users

Smartphones and Tablets Are Easily Lost or Stolen

Implications

- Data loss due to device loss is common

- Attackers have easy access to device
  - Memory-based attacks are inexpensive
    - Cold-boot, bus snooping/monitoring, DMA

Cannot afford to neglect physical attacks

This Talk: Two Approaches

1. Software abstractions for mobile devices:
   - Firmware-TPM (trusted platform module)
   - Trusted sensors
   - Cloud-TPM: cross-device TPM-protection

2. New systems leveraging trusted hardware
   - Sentry: protect data against memory attacks
   - TLR: small secure runtime at the language-level

Acknowledgements

- Microsoft Research researchers & engineers:
  - Alec Wolman, Himanshu Raj, and many others (next slide)
- Microsoft Research interns:
  - Patrick Colp (U. of British Columbia)
  - He Liu (U. of California at San Diego, now with Google)
  - Chen Chen (ETH Zurich)
  - Nuno Santos (MPI-SWS, now with U. of Lisbon)
- External collaborators:
  - Jiawen Zhang, James Gleeson, Sahil Suneja, Eyal de Lara U. of Toronto
  - Krishna Gummadi (MPI-SWS)
  - Rodrigo Rodrigues (MPI-SWS, now with IST, Portugal)

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

(published at USENIX Security 2016)

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:

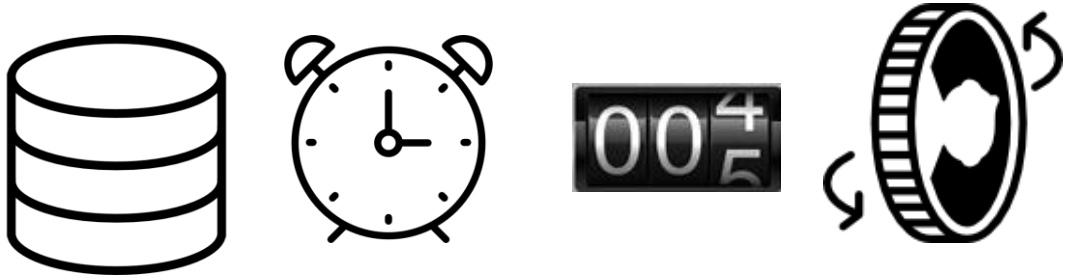
TrustZone®
Security Foundation by ARM®

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:

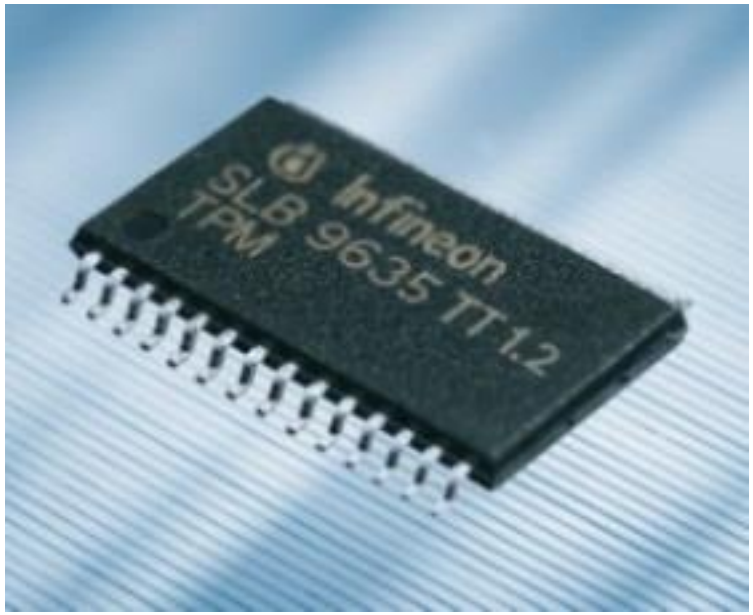
- 3 approaches to overcome TrustZone's limitations
  (lessons relevant to SGX also)
- Security analysis of fTPM vs TPM chips
- fTPM shipped millions of Microsoft Surface & WP

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

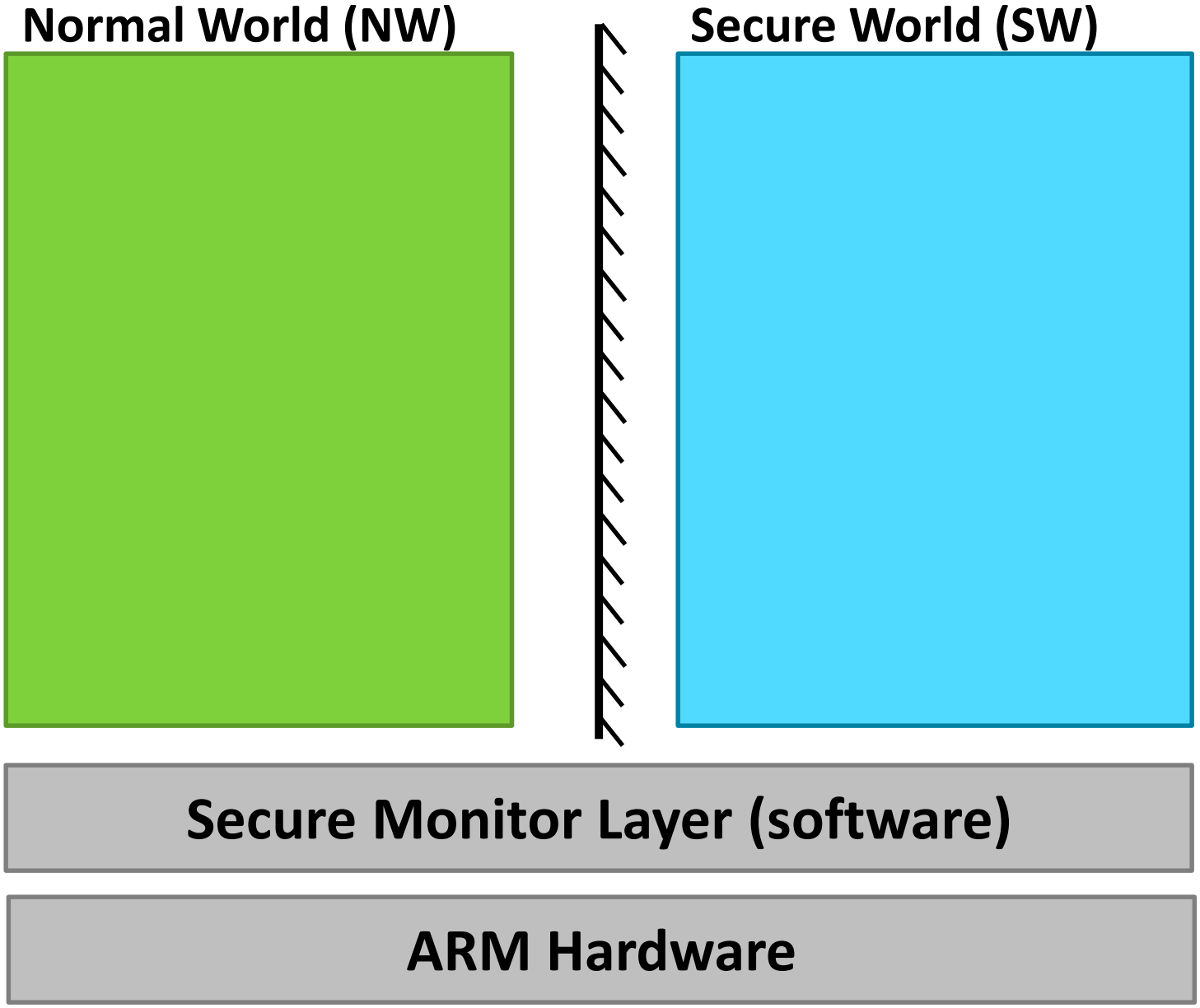
- Late 1999: TCPA is formed (IBM, HP, Intel, Microsoft, …)
- 2001: TPM specification 1.0 is released
  - Never adopted by any hardware AFAIK
- Late 2001: TPM 1.1 is released
- 2002: IBM Thinkpad T30 uses first discrete TPM chip
- 2003: TCPA morphs into TCG
- 2007: pin reset attack
- 2008: TPM 1.2
  -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

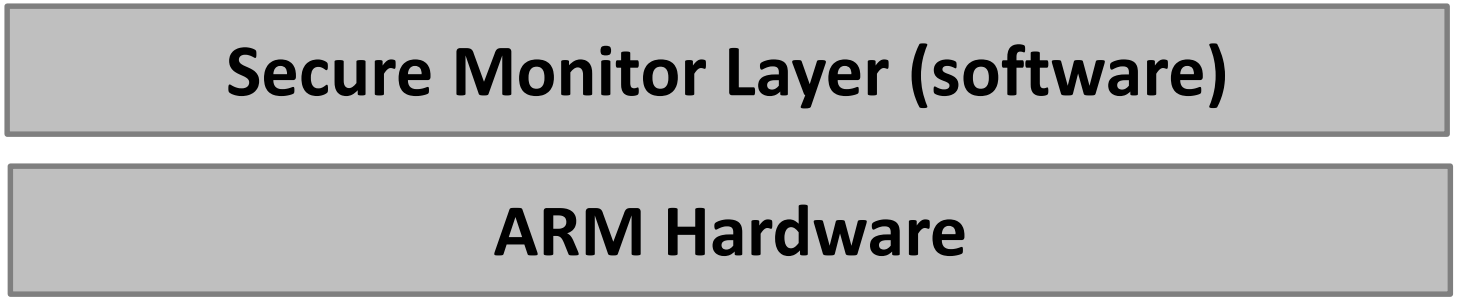
Outline

- 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

Booting Up

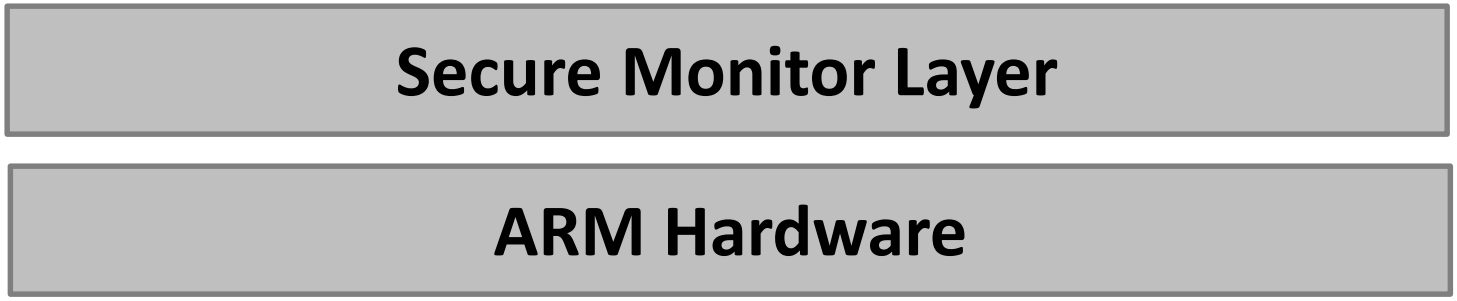


Booting Up

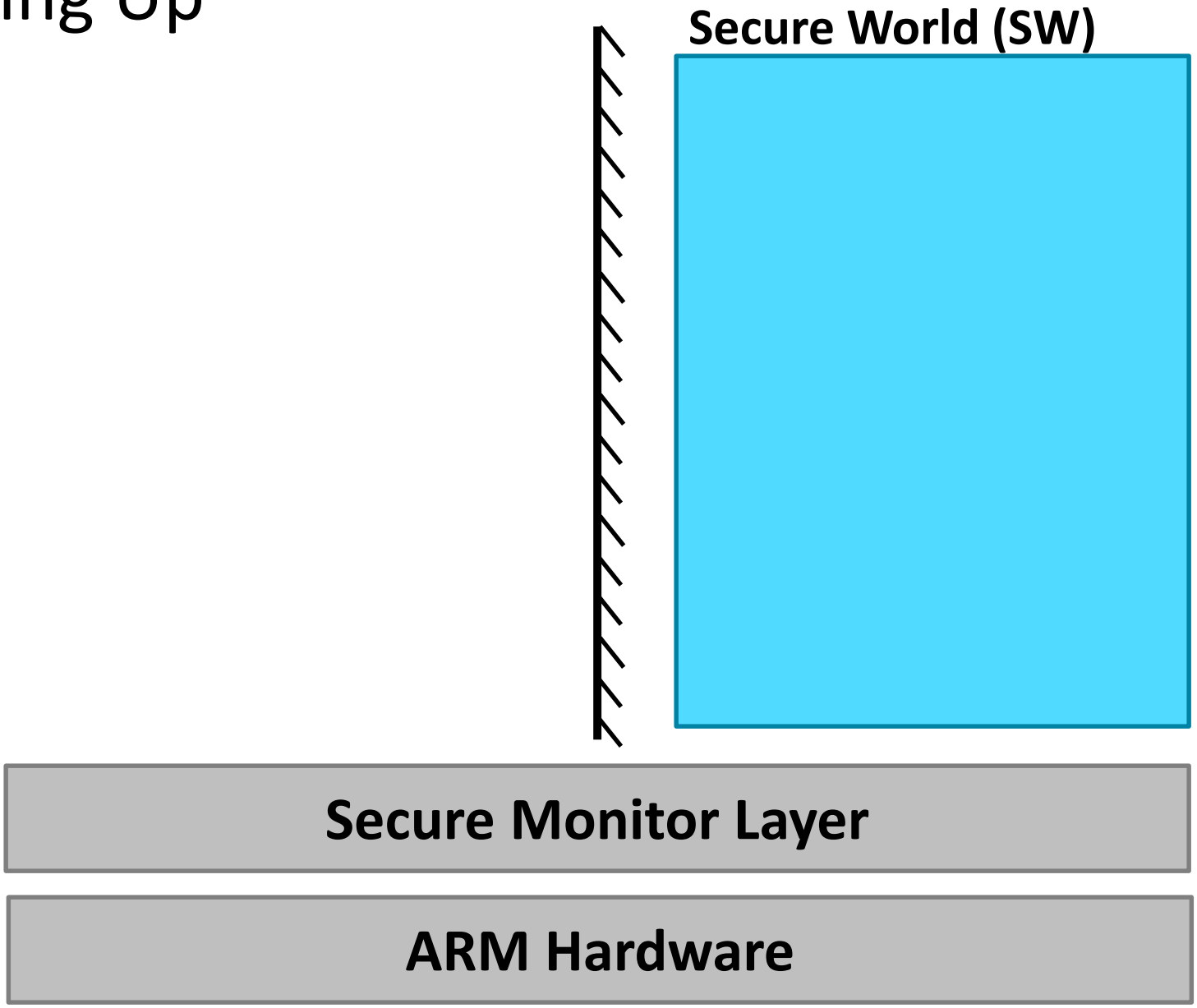
Allocates memory

Restricts its access to Secure World-only

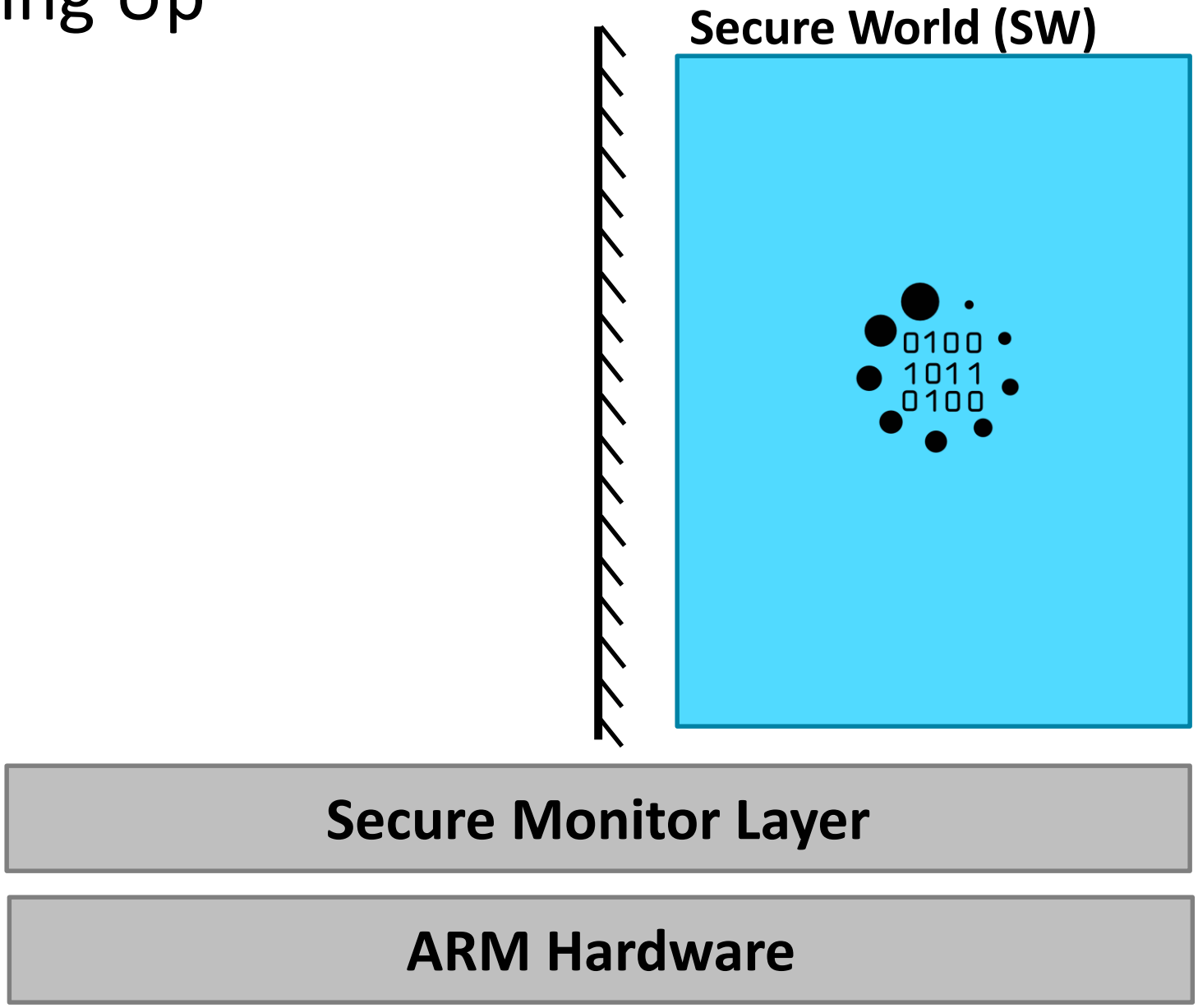
More setup…

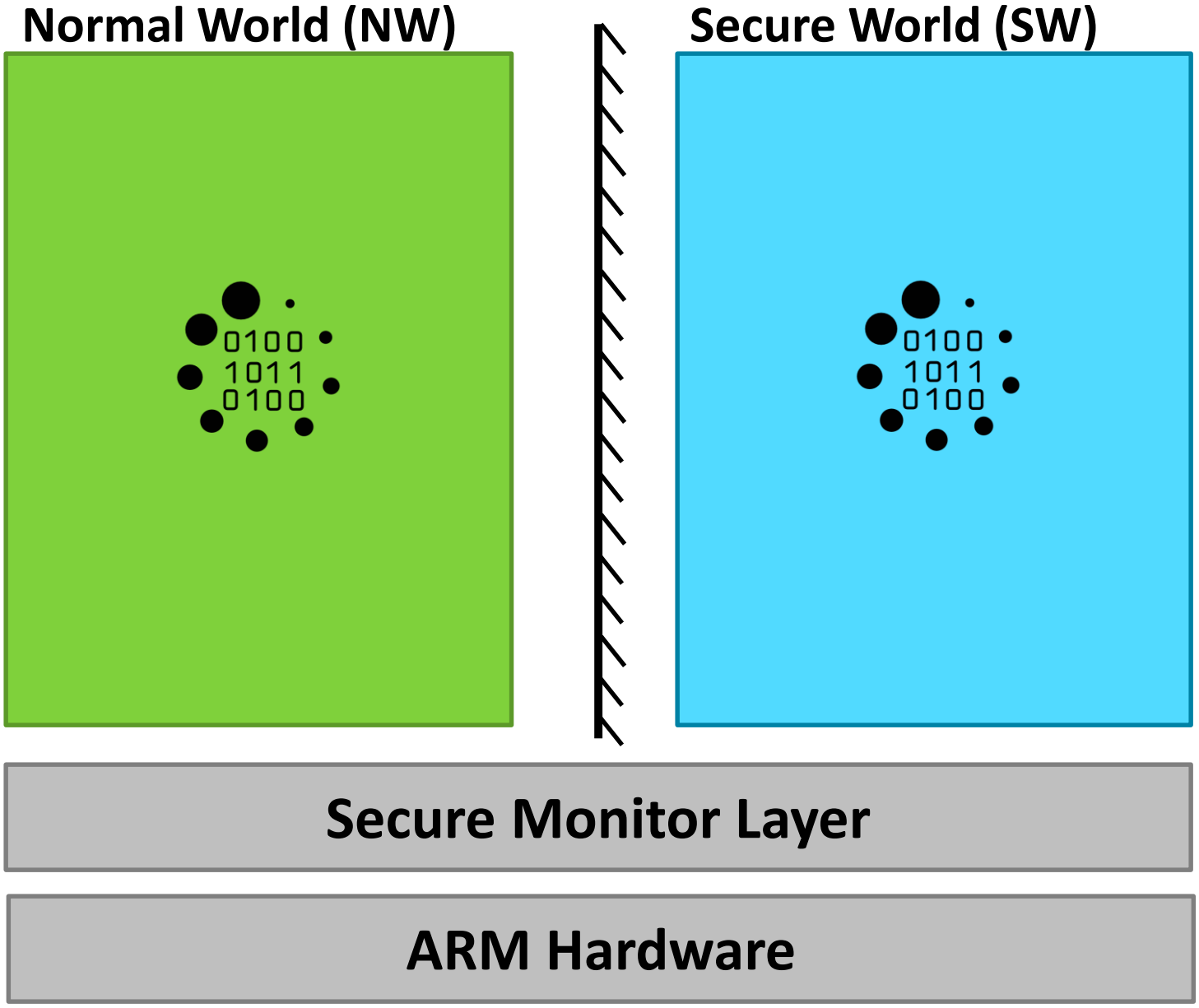


Booting Up



Booting Up

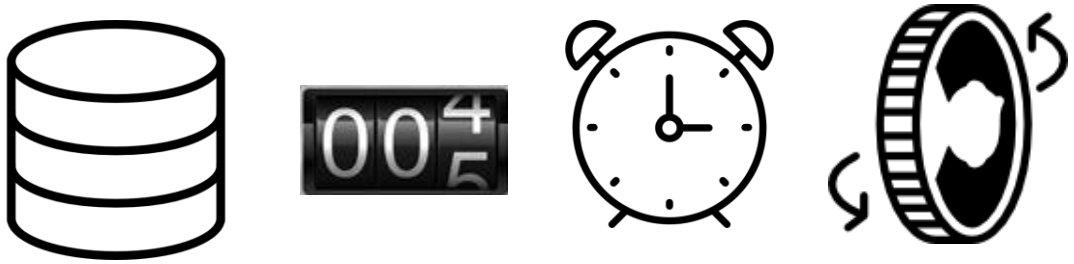

Secure World (SW)
0100
1011
0100
Secure Monitor Layer
ARM Hardware

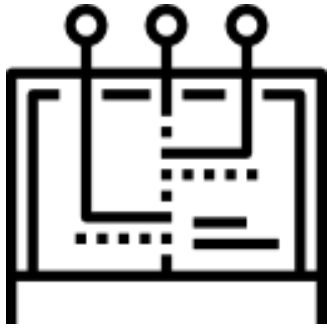

Normal World (NW)
Secure World (SW)
0100
1011
0100
0100
1011
0100
Secure Monitor Layer
ARM Hardware

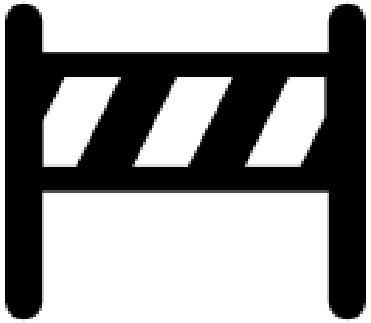
ARM TrustZone Properties

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

ARM TrustZone Limitations







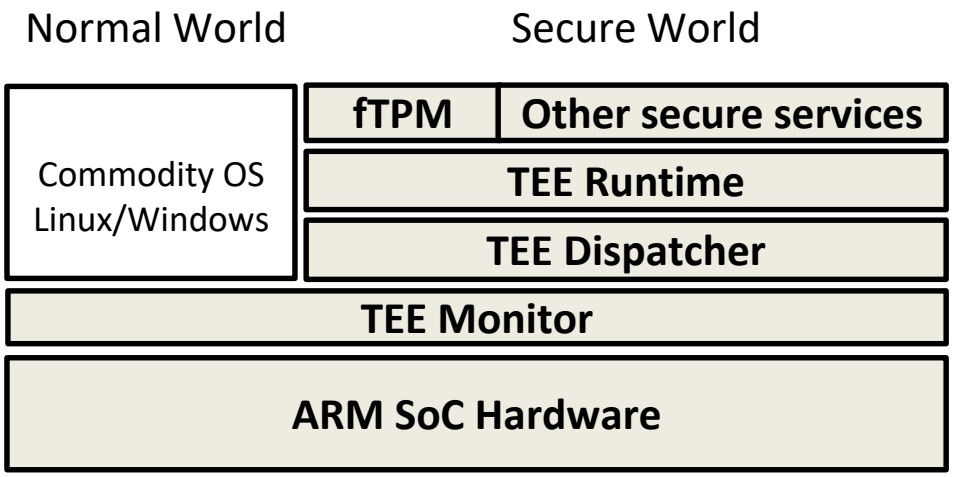
Lack of virtualization

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?

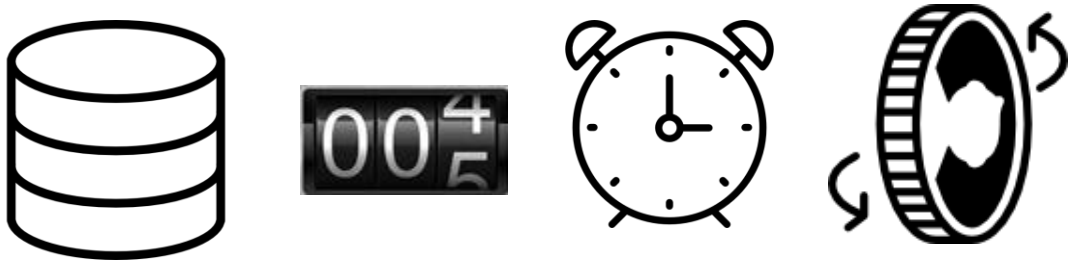
Goals	fTPM	TPM chip
Malicious software (e.g., malware, compromised OS)		
Time-based side-channel		
Cache-based side-channel		
Denial-of-Service		
Power analysis-based side-channel		
Memory attacks (e.g., coldboot, bus sniffing, JTAG)		

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

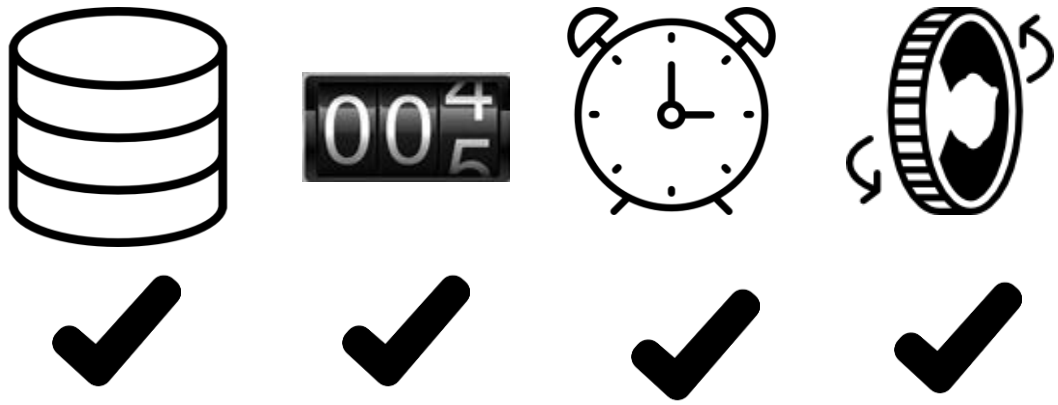
- eMMC controller present on many ARM SoCs
  - Has provisions for trusted storage
- Secure fuses: write-once, read-always registers
  - Can act as "seed" for deriving crypto keys
- Entropy for TrustZone can be added easily

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

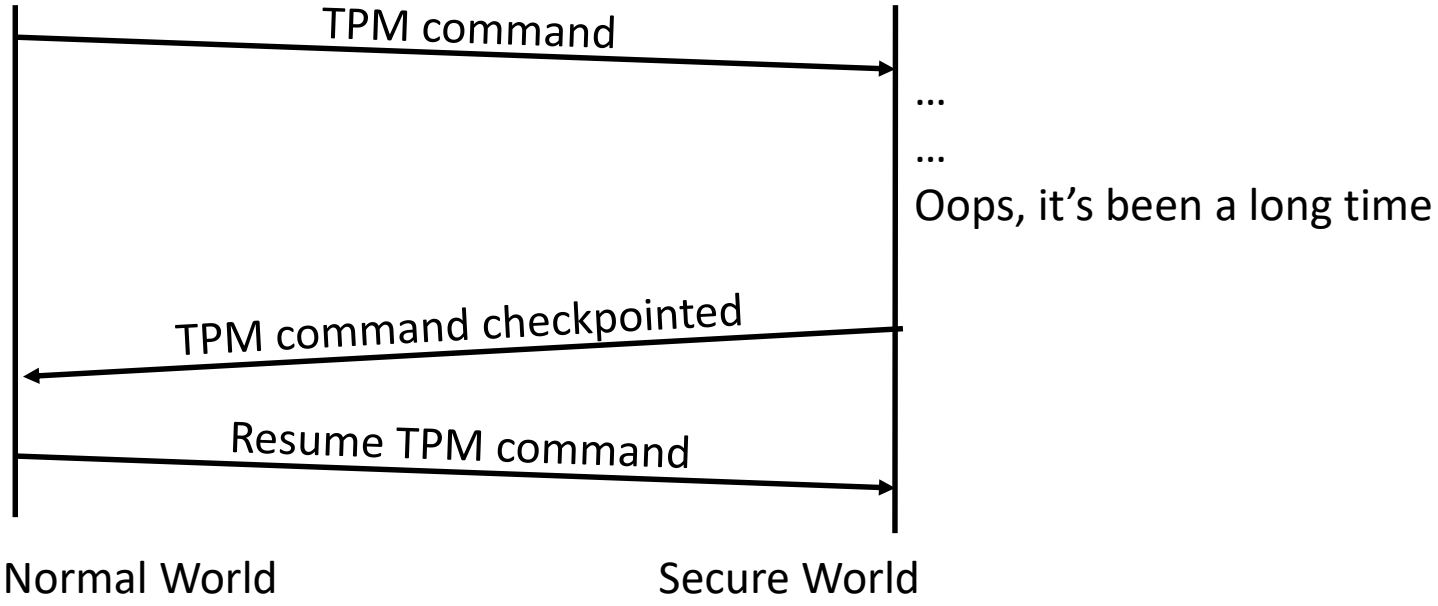
Do not affect TPM's security!

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

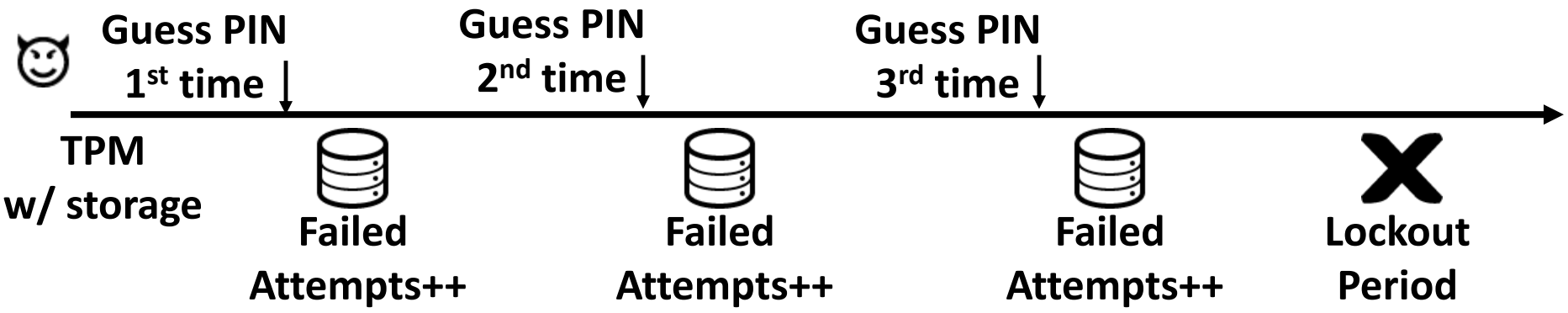


Three Approaches

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

Do not affect TPM's security!

Background: TPM Unseal

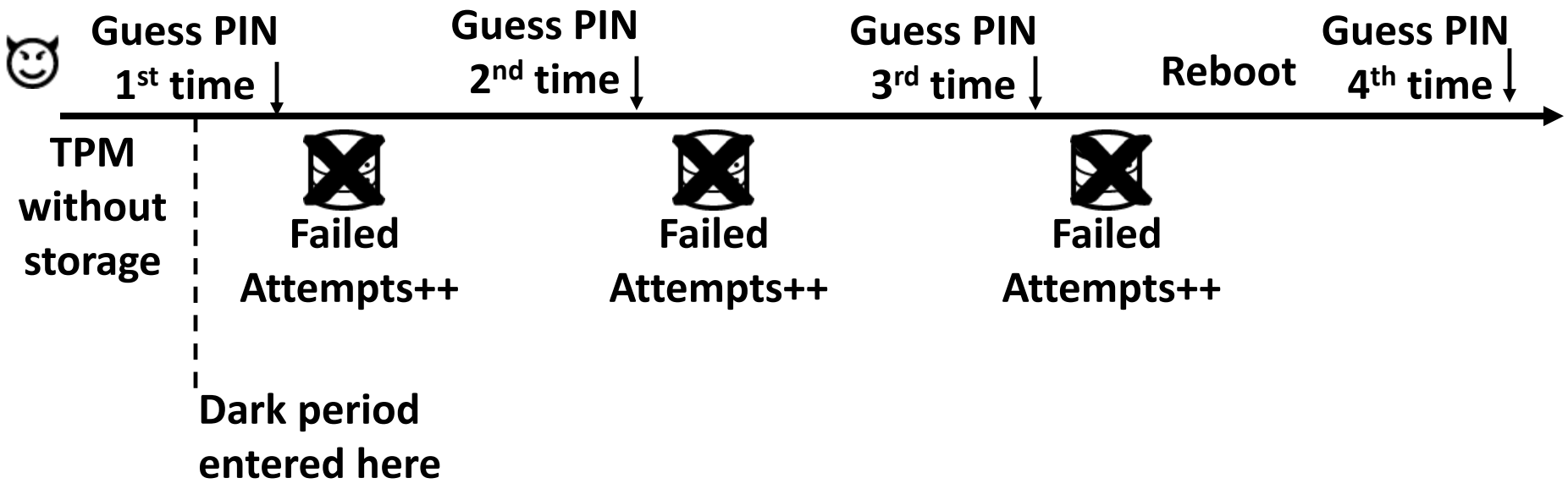


Problem: Dark Periods

- 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



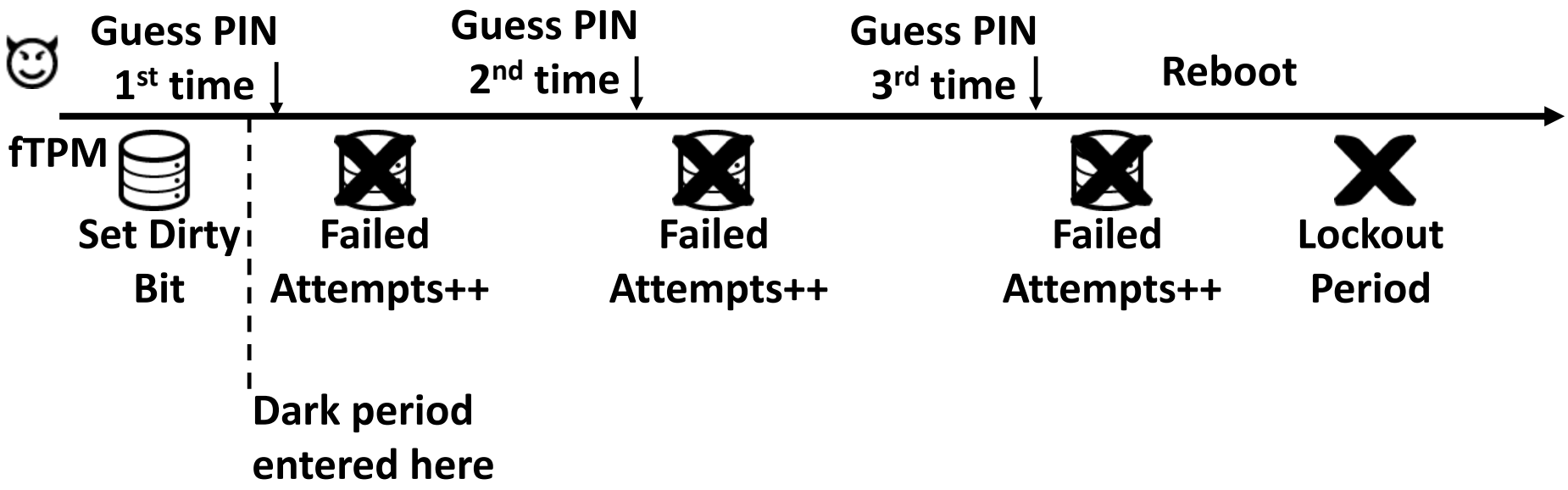
Solution: Dirty Bit

- 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

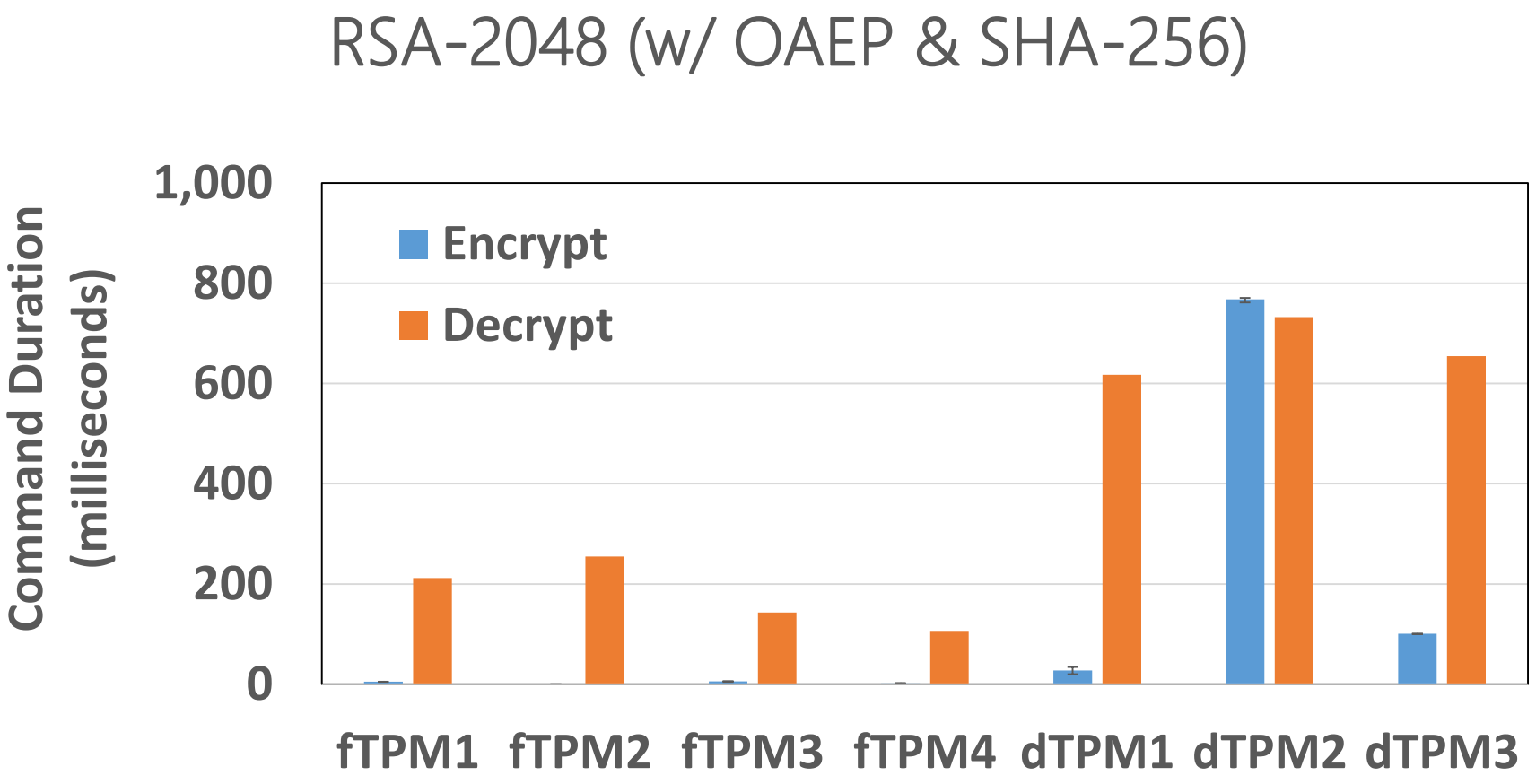
- Motivation
- Background on TPM
- ARM TrustZone and its shortcomings
- High-level architecture & threat model
- Overcoming TrustZone limitations: three approaches
- Performance evaluation
- Conclusions

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



fTPM: 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

This Talk: Two Approaches

1. Software abstractions for mobile devices:
   - Port TPM (trusted platform module) from PCs to smartphones
   - Trusted sensors
   - Cloud-TPM: cross-device TPM-protection

2. New systems leveraging trusted hardware
   - Sentry: protect data against physical attacks
   - TLR: small secure runtime at the language-level

Sentry: Protecting Data on Smartphones & Tablets from Memory Attacks

Patrick Colp

Jiawen Zhang
James Gleeson
Sahil Suneja
Eyal de Lara

Himanshu Raj
Stefan Saroiu
Alec Wolman

U. of British Columbia

U. of Toronto

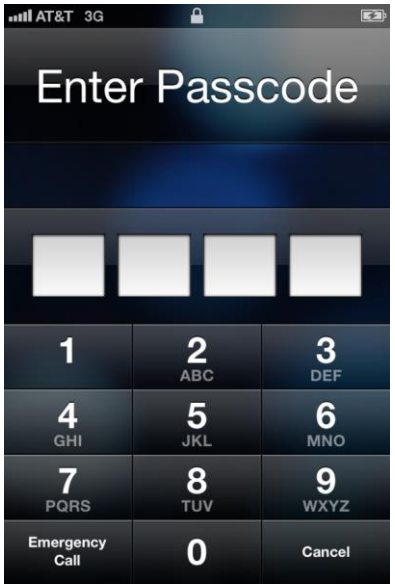
Microsoft Research

(published at ASPLOS 2015)

Smartphones Store Sensitive Data

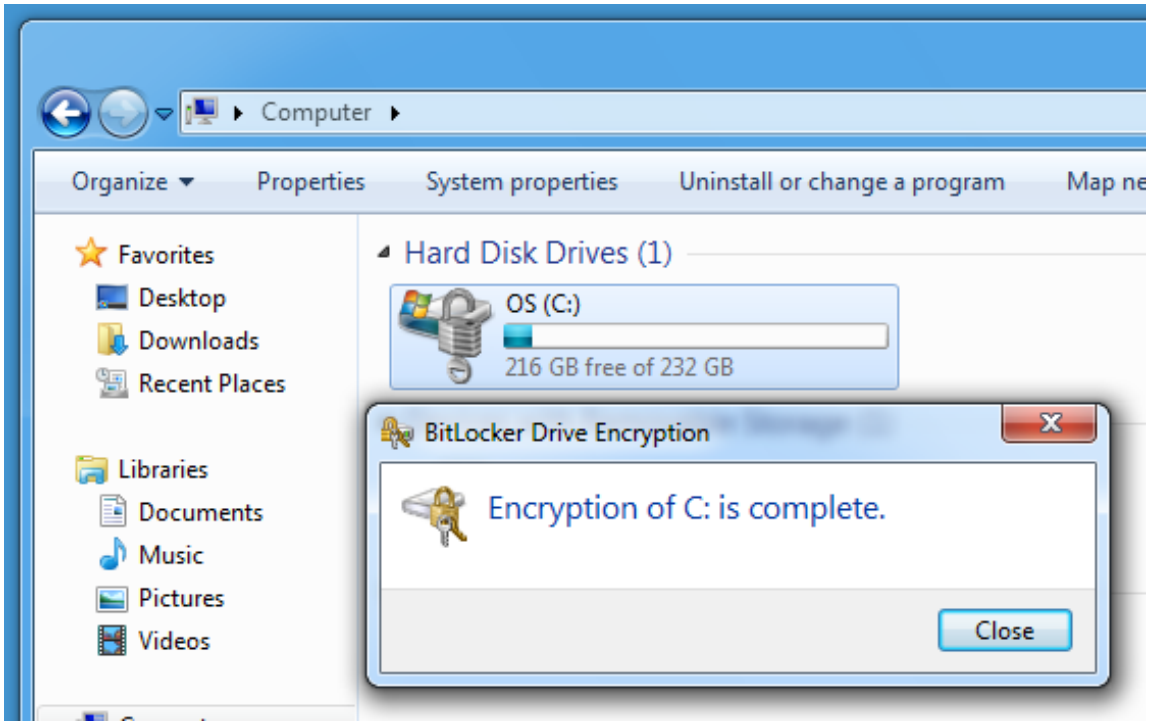
Smartphones and Tablets Are Easily Lost or Stolen

Industry Solution #1: PIN-unlock



Problem: Unencrypted data still resides in RAM!

Industry Solution #2: Disk encryption



Full disk encryption: Protect data-at-rest

Adequate for laptops: Laptops often shutdown/hibernating

Inadequate for smartphones & tablets: These devices are always **on**

Imagine an attacker has possession of a stolen device and can't guess the PIN

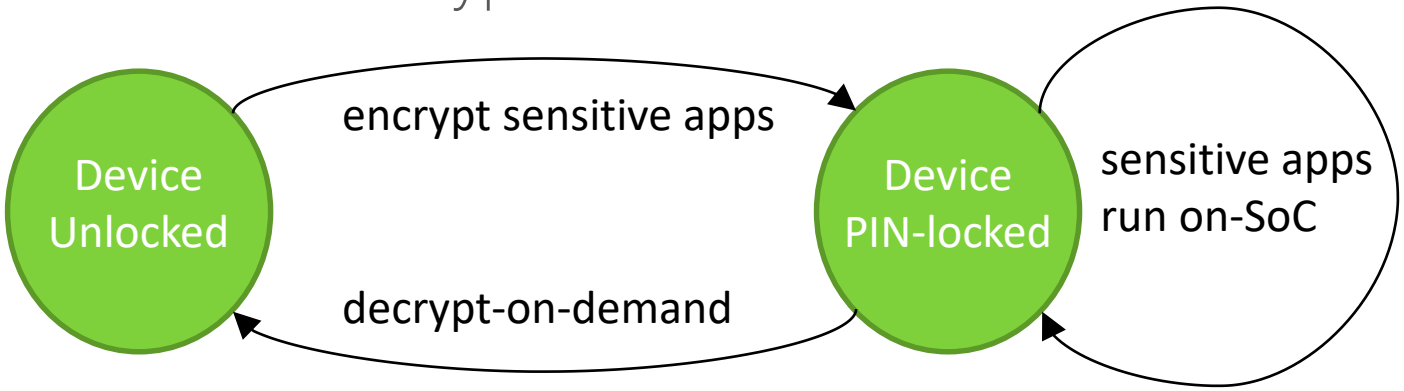
What can they do?

Memory Attacks

- Memory attacks allow attacker to gain access to sensitive data stored in memory
- Three classes of memory attacks:
  - Cold boot attacks
  - Bus monitoring attacks
  - DMA attacks
- Common aspect of attacks:
  - Physical possession of the device is required

Sentry: Keep Sensitive Data on SoC

- With Sentry, memory pages are stored:
  - Encrypted in DRAM
  - Decrypted on the ARM SoC (System-on-Chip)
- Key observation to reduce overhead
  - No need to encrypt when device is unlocked



Sentry's Lifecycle

Outline

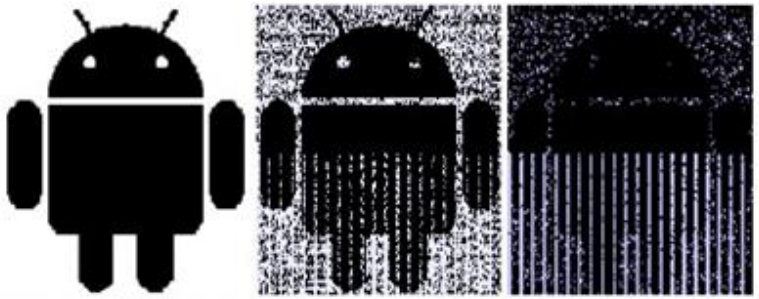
- Introduction
- Memory (RAM) attacks
- Threat model
- Sentry's system design
- Performance evaluation
- Related work & conclusions

Memory Attacks

- Three classes of memory attacks:
  - Cold boot attacks
  - Bus monitoring attacks
  - DMA attacks

Cold Boot Attacks

- DRAM contents don't disappear after power cut
  - Known as the data remanence effect, cooling extends time [Halderman et al., Usenix Security 2008]
- Two types of cold boot attacks
  - Remove DRAM from device and attach it to a reader
  - Reflash device with malicious firmware that reads (preserved) DRAM
- Recently demonstrated on Android [Müller et al., ACNS'13]



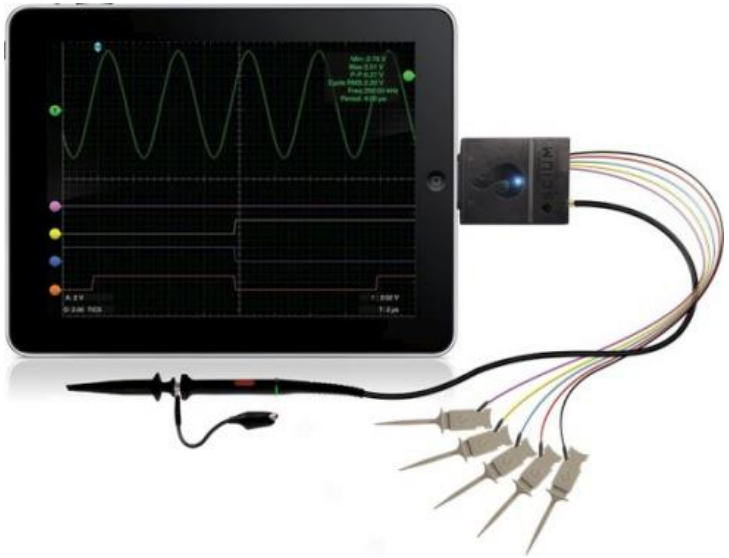
Modern Tegra3 NVidia Tablet

- 1 GB of DRAM, room temperature
- Three steps:
  1. Write unique 32-bit pattern into device's DRAM
  2. Mount various cold-boot attacks
  3. Measure fraction of bit pattern still preserved

Type of Attack	DRAM Preserved
OS Reboot (no power loss)	96.4%
Device Reflash (short power loss)	97.5%
2 Second Reset (long power loss)	0.1%

Bus Monitoring Attacks

- Place monitoring device on memory bus to record communication
- Cannot directly access memory contents, but can view all data read from or written to memory



DMA Attacks

- Attach malicious DMA-based peripheral to stolen tablet
  - Dump entire DRAM
- Today less prevalent because most smartphones and tablets lack DMA ports
  - But this could change

Outline

- Introduction
- Memory (RAM) attacks
- Threat model
- High-level system design
- Performance evaluation
- Related work & conclusions

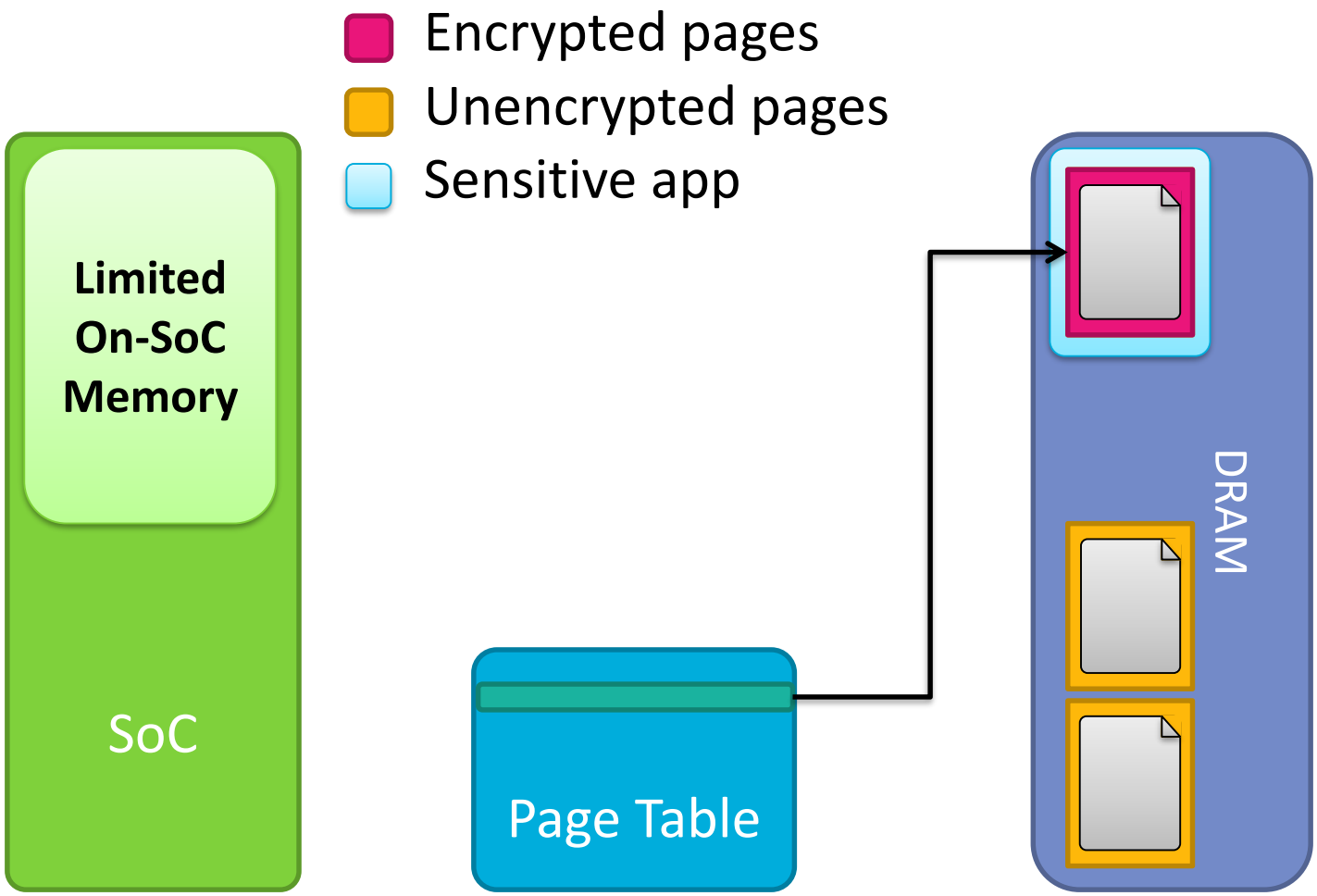
Threat Model

- In-scope:
  - Cold boot, bus monitoring, DMA attacks
- Out-of-scope:
  - JTAG attacks
  - Sophisticated physical attacks
  - Code-injection attacks
  - Physical side-channel attacks

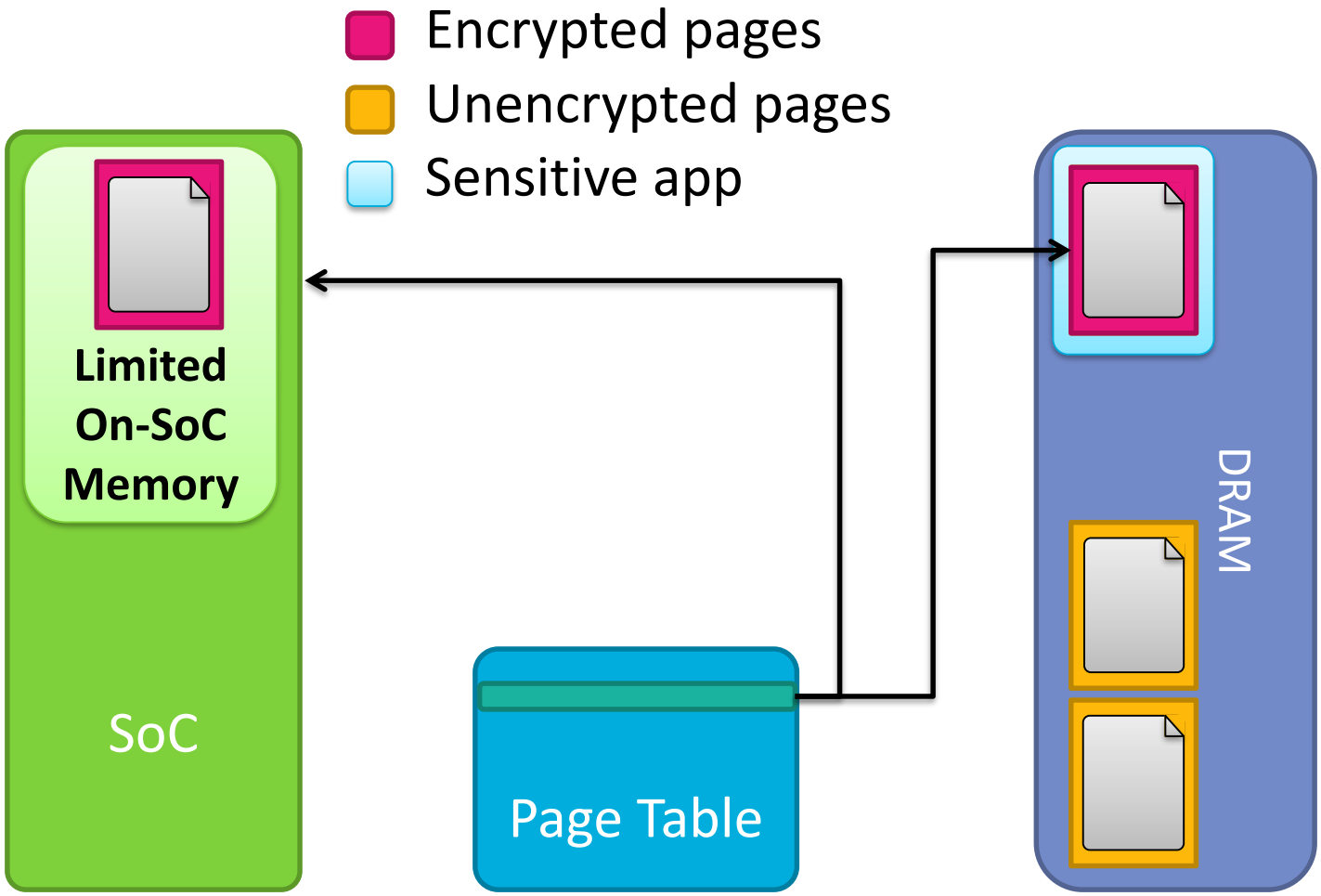
Outline

- Introduction
- Memory (RAM) attacks
- Threat model
- Sentry's system design
- Performance evaluation
- Related work & conclusions

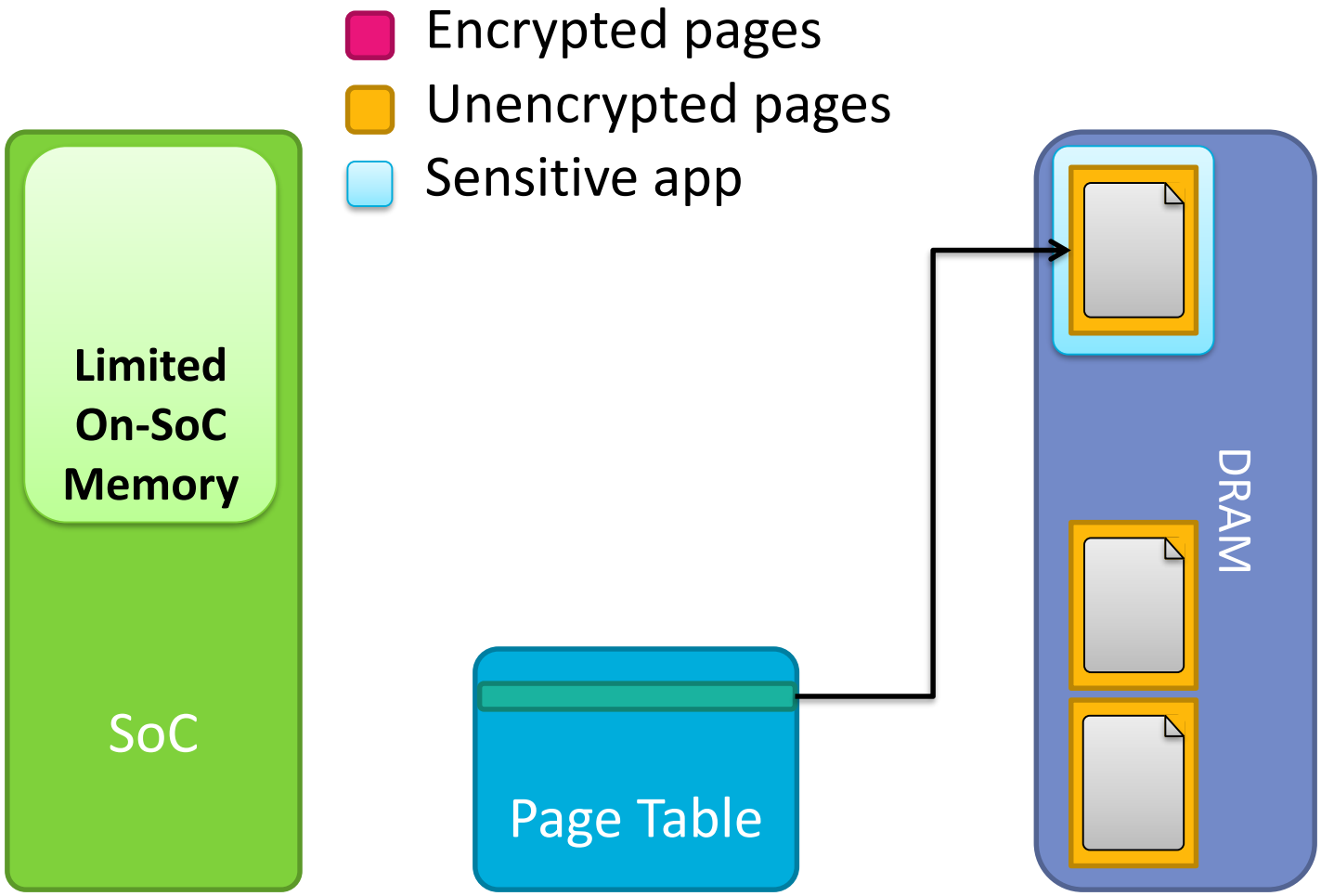
Sentry in Action: Upon Device Lock



Sentry in Action: Sensitive Apps Running in Background (Locked Device)



Sentry in Action: Upon Device Unlock



Sentry's Challenges

1. Where on SoC can code and data be kept?
2. How can crypto be done in-place on the SoC?
3. How do we guarantee no data "leaks" to DRAM?
4. How do we secure freed memory pages?
5. How do we bootstrap?
6. What are minimum on SoC requirements?

Sentry's Challenges

1. Where on SoC can code and data be kept?
2. How can crypto be done in-place on the SoC?
3. How do we guarantee no data "leaks" to DRAM?
4. How do we secure freed memory pages?
5. How do we bootstrap?
6. What are minimum on SoC requirements?

See ASPLOS 2015 paper for rest of answers

On-SoC Storage

- Internal RAM (iRAM)
  - Some devices ship with small iRAM (e.g., 256 KB)
- L2 Cache Locking
  - ARM cache controllers offer cache locking
    - Aimed at embedded systems for performance predictability
- Safe against cold-boot attacks
  - Unflashable firmware erases iRAM
- Safe against bus monitoring attacks
- Safe against DMA attacks
  - iRAM is DMA-able; need TrustZone-based DMA protections

Outline

- Introduction
- Memory (RAM) attacks
- Threat model
- Sentry's system design
- Performance evaluation
- Related work & conclusions

Performance & Energy Questions

- What is Sentry's overhead?
  - Upon locking and unlocking a device
  - While decrypting on-demand on running apps
  - When running sensitive app in background
  - For protecting OS subsystem (dm-crypt)

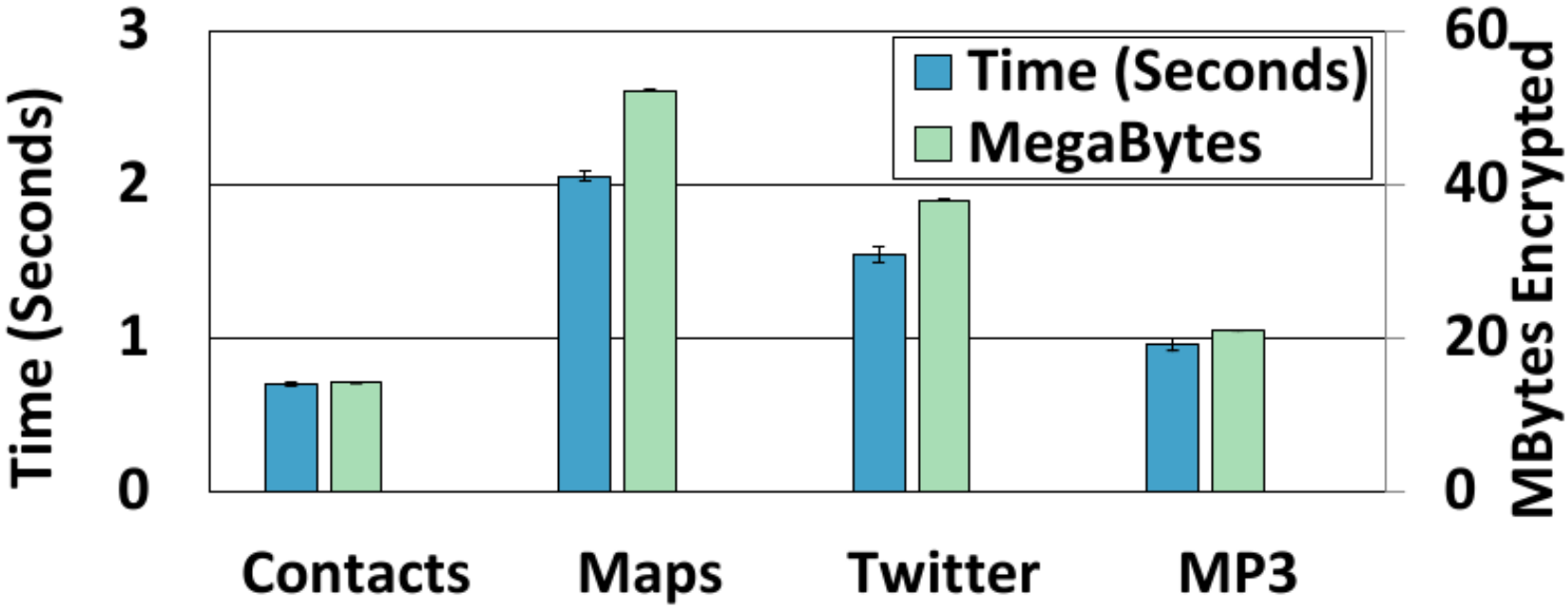
- What is Sentry's impact to the rest of system?
  - Portion of L2 cache allocated to Sentry

Performance & Energy Questions

- What is Sentry's overhead?
  - Upon locking and unlocking a device
  - While decrypting on-demand on running apps
  - When running sensitive app in background
  - For protecting OS subsystem (dm-crypt)

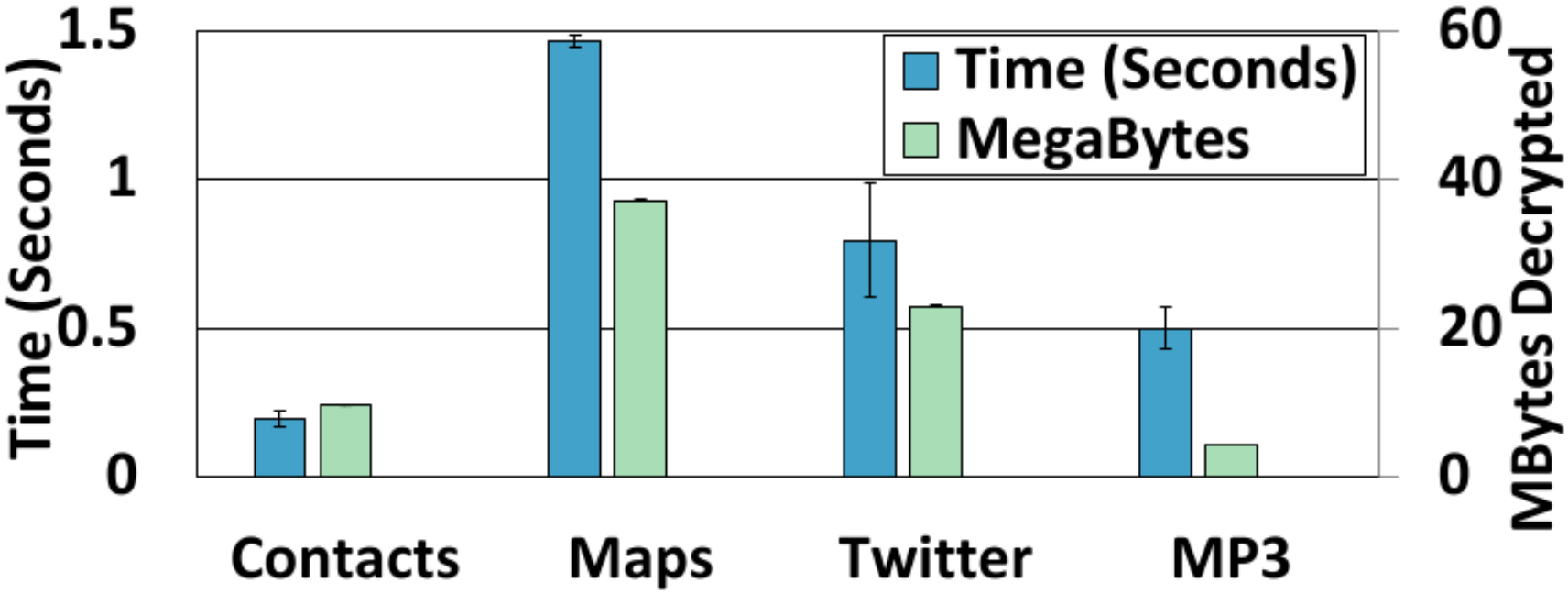
- What is Sentry's impact to the rest of system?
  - Portion of L2 cache allocated to Sentry

Performance Overhead on Lock



0.7-2.1 seconds overhead per application

Performance Overhead on Unlock



Minimum state required for apps to operate
0.2-1.5 seconds overhead per application

Outline

- Introduction
- Memory (RAM) attacks
- Threat model
- Sentry's system design
- Performance evaluation
- Related work & conclusions

Related Work

- Intel SGX
- On-chip AES schemes for x86:
  - AESSE [Eurosec'10]
  - TRESOR [Usenix Sec'11]
- Encrypted RAM
  - Cryptkeeper [ICTHS'10]
  - Encrypt-on-cache-evict [DATE'08]
- Cloud-backed encrypt-on-lock
  - ZIA [Mobicom'02]
  - Transient Authentication [Mobisys'03]
  - Clean OS [OSDI'12]

Sentry: Conclusions

- Smartphones/tablets are vulnerable to memory attacks
- Sentry protects these devices by keeping sensitive data encrypted in DRAM
- ARM offers cache-locking and iRAM to hold sensitive data on-SoC

Overall Summary

1. Software abstractions for mobile devices:
   - Firmware-TPM (trusted platform module)
   - Trusted sensors
   - Cloud-TPM: cross-device TPM-protection

2. New systems leveraging trusted hardware
   - Sentry: protect data against memory attacks
   - TLR: small secure runtime at the language-level

Questions?

- ssaroiu@microsoft.com