

# Introduction to Computer Security Lecture Slides

© 2024 by [Mihai Chiroiu](#) & [Florin Stancu](#)

is licensed under [Attribution-NonCommercial-ShareAlike 4.0  
International](#)

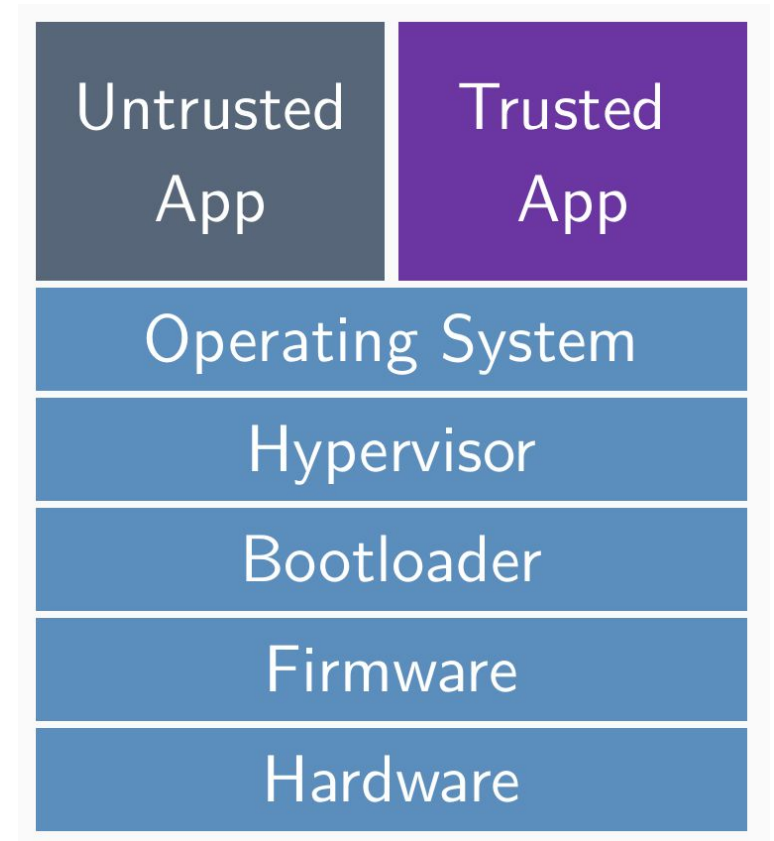
# Hardware Security

# This lecture

- Trusted Computing / Root of Trust
- Hardware-assisted security: TPM, ARM TrustZone, Intel SGX...
- Hardware Security Modules (HSMs)
- Cold boot / Evil Maid
- Side-channel attacks & other attacks

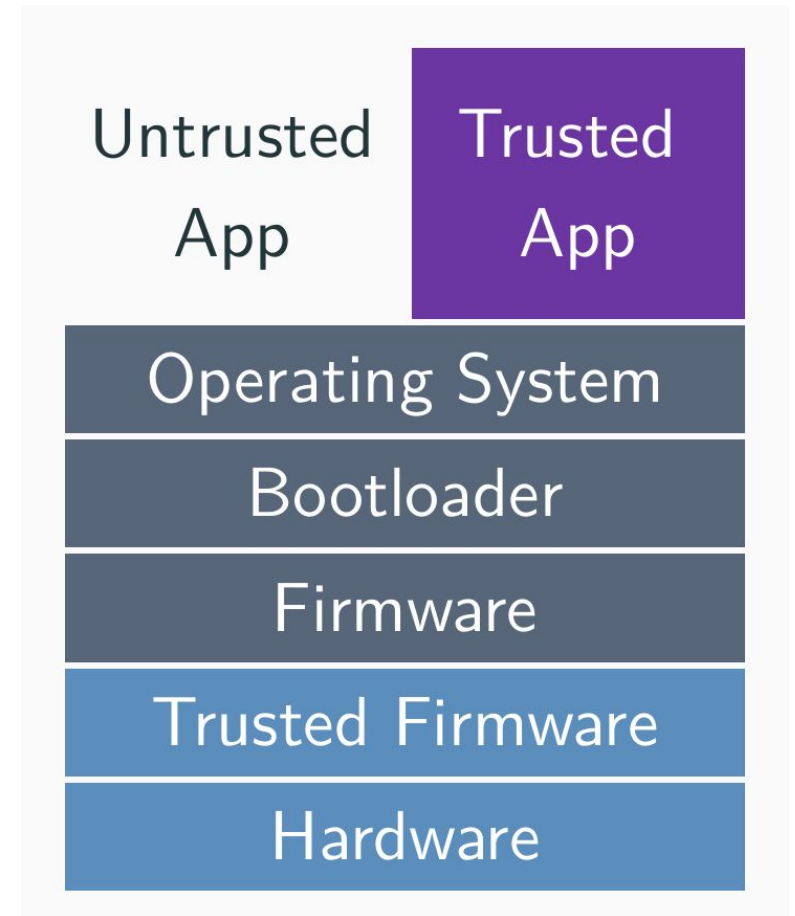
# Trustworthy Systems

- Trustworthy software + hardware!
- Boot process:
  - System Firmware (e.g., BIOS)
  - Bootloader
  - Hypervisor + OS Kernel
- Trusted Computing Base
  - OS Kernel: ~20 millions LoC!
  - Bug probability?
    - It's over 9000!



# Trustworthy Systems (2)

- Protect critical data / applications!
- Minimize Trusted Computing Base?
  - Approach: Turn a portion of a platform into a trustworthy environment!
- Trusted Execution Environments!
- Applications: confidential cloud, banking, medical privacy, industrial equipment etc.



# Security Properties

- Isolated Execution
  - Trusted Execution Environments (TEE)
- Secure Storage
  - Integrity, confidentiality
- Attestation (remote and local)
  - Data given only to the trusted machine
- Secure Provisioning
  - Channel for sending data
- Trusted Path
  - Communication channel for peripherals (Secure I/O)

# Trusted Platform Module (TPM)

# Trusted Computing

- The Trusted Computing Platform Alliance (TCPA) The Trusted Computing Platform Alliance (TCPA) – TPM v1
  - Established by the 5 founders in 1999: Intel, AMD, IBM, HP and MSFT
- The Trusted Computing Group (TCG) – TPM v2
  - Established in March 2003 as continuation of TCPA
- “For years Bill Gates has dreamed of finding a way to make the Chinese pay for software, TC looks like being the answer to his prayer.” by Ross Anderson



# Trusted Platform Module

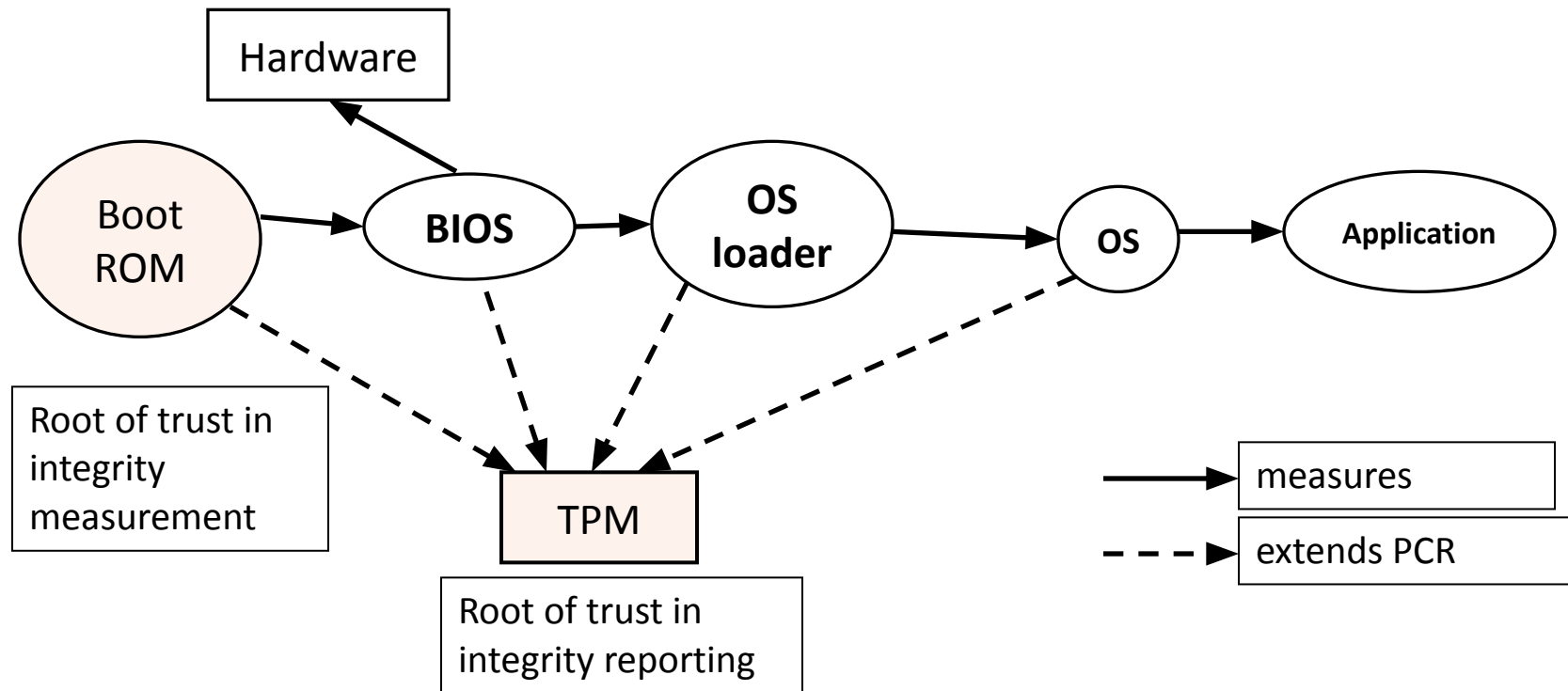
- Cryptographic Coprocessor
- Provides multiple Roots of Trust:
  - Root of trust for measurement (RTM) a trusted implementation of a hash algorithm
  - Root of trust for storage (RTS) a trusted implementation for one or more secret keys —the storage root key (SRK)
  - Root of trust for reporting (RTR) a trusted implementation for a secret key representing a unique platform identity – the endorsement key (EK)

# TPM Capabilities

- Platform Configuration Registers (PCR)
  - Can be read
  - Can only be extended, not writable
- Migratable vs non-migratable keys
  - EK and SRK never leave TPM
- Secure Storage (binding / sealing)
- Attestation (Local / Remote)

# Trusted/Secure Boot

- After boot, PCRs contain hash chain of booted software

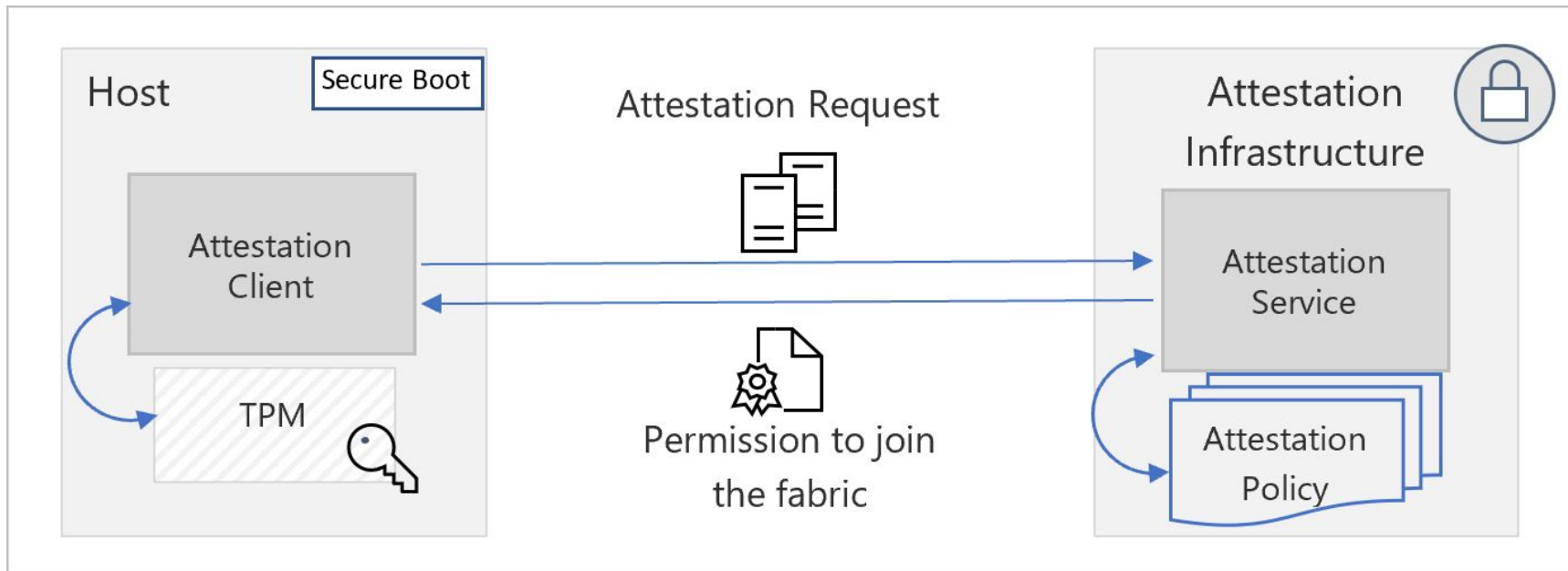


# Secure Storage

- Binding/Unbind
  - Store small keys to the TPM;
  - Can only be retrieved by having the same measurements at boot!
- Sealing/Unsealing
  - Sealing is an extension to binding;
  - Only non-migratable storage keys can be used to seal data;
  - The encrypted data is always bound to a specific platform!

# Attestation (remote and local)

- System boots, measures all code & data into PCR;
- Client requests the TPM to sign using the Attestation Key;
- Client supplies report to remote service for secret provisioning.



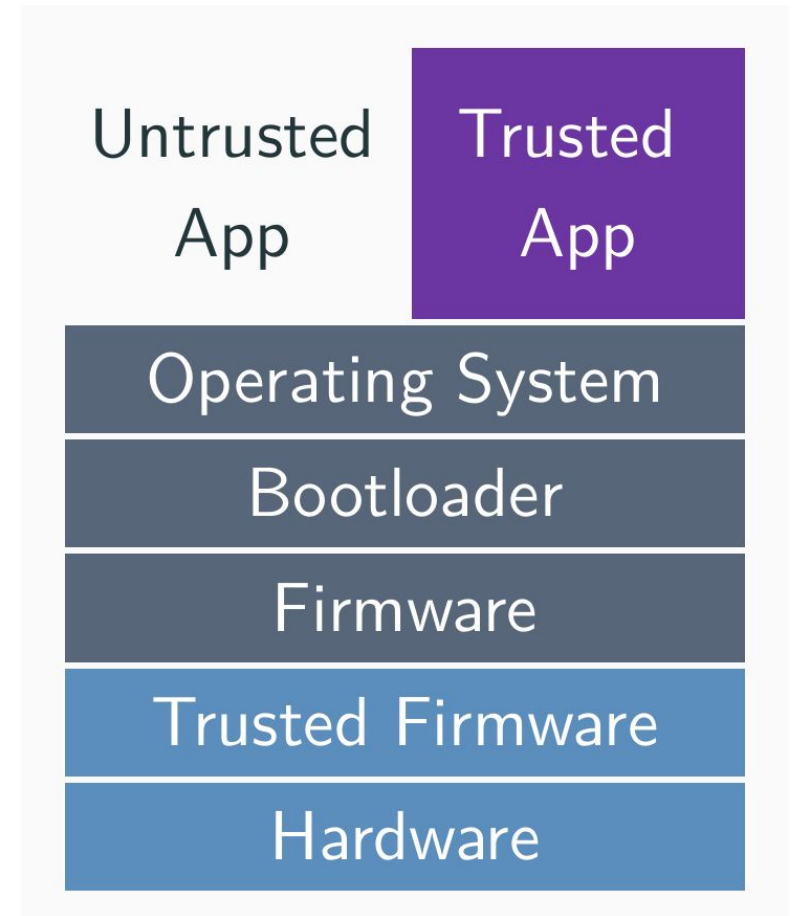
# Attestation issues

- Attestation: remote service validates the signed PCRs (using the a public key derived from the Endorsement Key)
  - How to do this for local attestation (untrusted physical system)?
- It only validates the loaded code, not the running one!
  - Buffer overflows in RAM => p0wned!
- Privacy? (shouldn't identify the user / machine!)
  - Anonymous attestation protocols (e.g., Intel EPID)
- Vendor lock-in / discrimination / DRM == bad...

# Trusted Execution Environments

# RE: TEEs

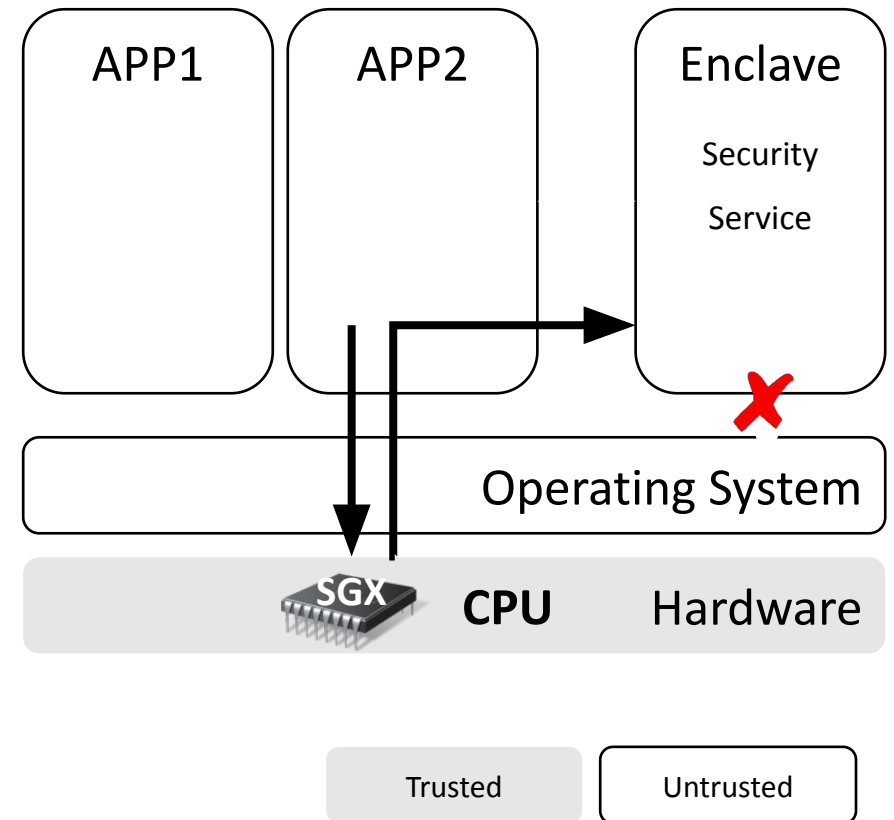
- Isolate memory & execution state
  - guaranteed protection even from higher privileged attackers
- Implementation approaches:
  - Software-based de-privileging (e.g., binary instrumentation) – performance & complexity issues
  - Hardware-assisted (e.g., CPU architectural extensions) – faster!
- HW platforms: Intel TxT, ARM TrustZone, Intel SGX (RIP), AMD SEV, Intel TDX





# Intel SGX

- Security critical code + data is isolated in enclaves
- Only the hardware (CPU) is trusted
  - Transparent memory encryption
- Enclaves cannot harm the system
- Only run unprivileged code (Ring 3)
- Memory protection (RAM encrypted)



# Intel SGX (2)

- Untrusted Operating System manages the Enclaves as normal processes!
- Enclaves may make normal system calls, e.g.: I/O, network, hardware access
  - WARNING: their operation + results are not to be trusted!
- Designed for Multi-Core systems
  - Multi-threaded execution of enclaves
  - Parallel execution of enclaves and untrusted code
  - Enclaves are interruptible

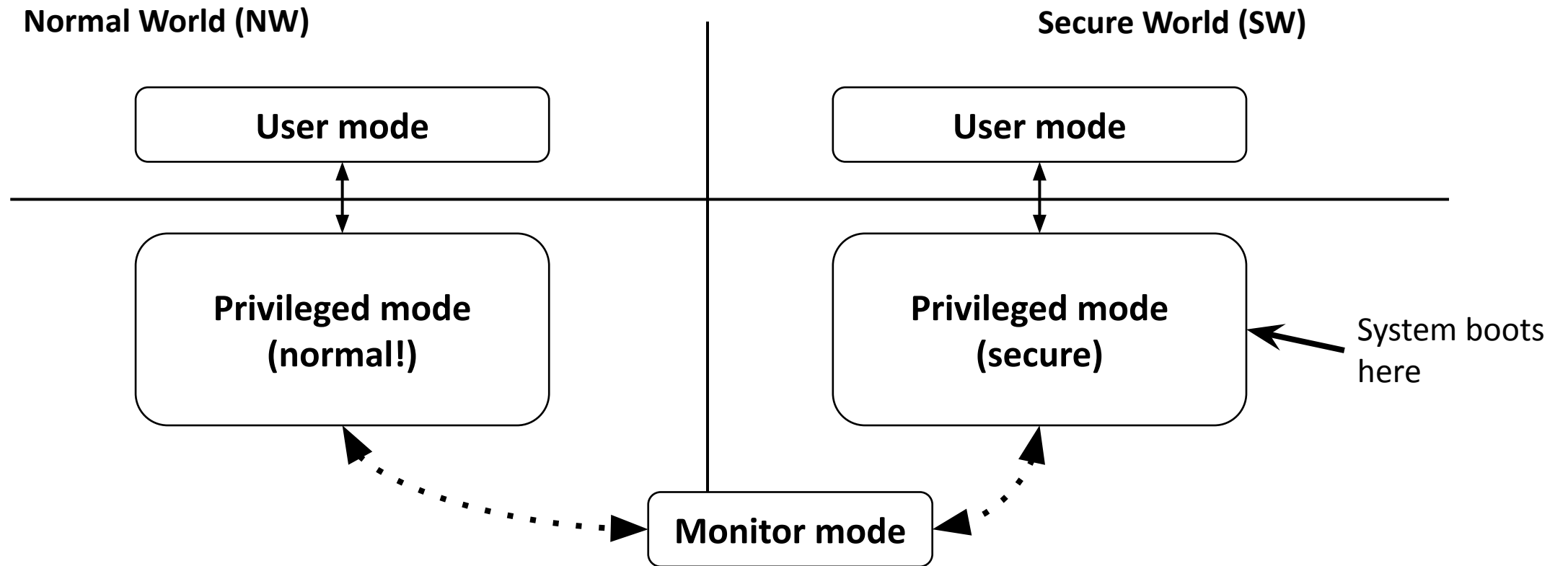
# Attestation (remote and local)

- Yes, both are built-in with SGX!
- An enclave can request a HW-signed REPORT
- In local attestation, one enclave can attest its TCB to another one
- Remote attestation requires Intel service contract
  - due to privacy-enabled group signature protocol (EPID)
  - Newer versions (for server CPUs) offer alternate ECDSA-based signatures!

# Trusted Path?

- Enclaves run in user space!
- Accessing devices (keyboard, display) from enclaves?
  - Need to go through untrusted kernel!
- Potential solutions: proxy hardware, hypervisor...
- Open research question...

# ARM TrustZone

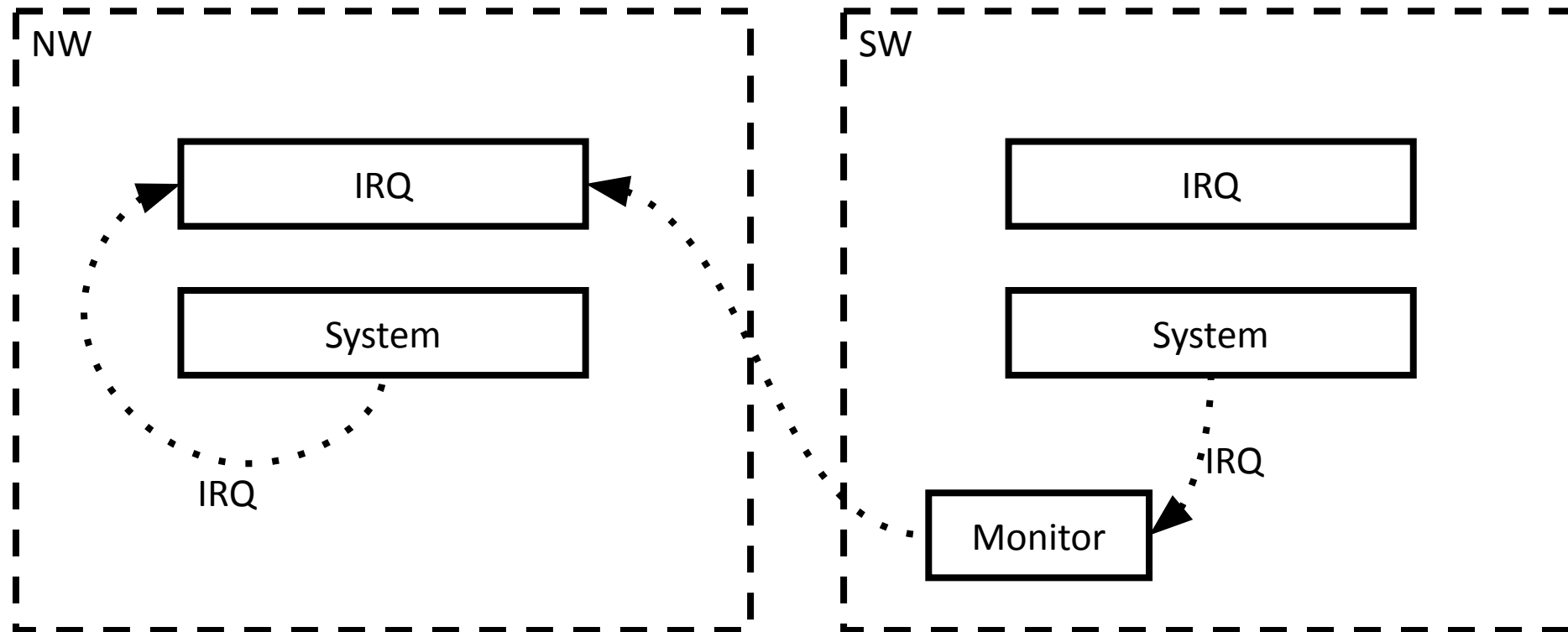


# TrustZone Discussion

- Includes Secure Boot!
- Used in modern mobile devices
  - TEE enclaves for banking cards, eSIMs, password manager etc.
- Issues:
  - Vulnerabilities in the TEEOS (CVE-2015-4421)
  - Closed system for third party application development on most devices...
  - Only one compartment for TCB!
- Provides a Trusted I/O Path!

# Trusted Path

- The ability to trap IRQ and FIQ directly to the monitor



# Comparison

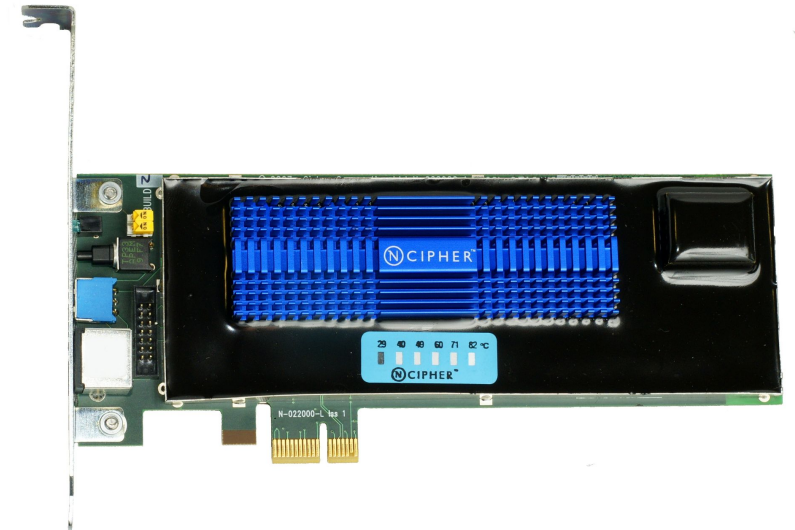
	Isolated Execution	Secure Storage	Remote Attestation	Secure provisioning	Trusted Path
TPM	No	Yes (limited)	Yes	Yes	No
TrustZone	Yes	Yes	Yes	Yes	Yes
SGX	Yes	Yes	Yes	Yes	??



# Hardware Security Modules (HSM)

# Hardware Security Modules

- A [cheap] piece of hardware and associated software/firmware that usually attaches to the inside of a PC or server and provides at least the minimum of cryptographic functions:
  - Strong random number generation
  - A secure time source
  - Tamper-resistance
- Used to keep secrets (and die with them!)



# Smart cards

- Hardware-based, tamper-proof public-key cryptography certificate stores
  - Small/cheap devices used for authentication / signing transactions / key encryption etc.
- Examples include: bank cards, SIM cards, passports / ID cards, building access cards, electronic car keys etc.
- And most of them run JavaCard :D



# Security Keys

- Portable cryptographic devices
- Used as second factor of authentication (something *physical* you have)
  - way more secure than mobile phones!
- Connected / wireless (Bluetooth / NFC) tokens
- Stolen / lost => enrol backups!
- YubiKey, NitroKey, Google Titan



# Hardware Vulnerabilities

# Hardware vs Software

- Hardware => unfixable... (buy another?)
- Move to software => increase complexity...
- Approaches:
  - Upgradable firmware
  - CPU instructions, microcode!
- Vulnerabilities => malicious code injection...

# Physical Attacks

- Worst case scenario
- Evil Maid Attack
  - TPMs
  - Hardware Key Loggers
- USB Rubber Ducky
- Tamper resistant hardware
  - Can't touch this!
  - E.g., HSMs
  - Self-destructing in 10...9...



# Side Channel Vulnerabilities

- Hardware behavior can give out extra information!
  - Extract algorithm parameters (key)
- Cache / timing, power analysis, electromagnetic, acoustic etc.
- Cause: hardware / software
- Mitigation: software
  - Constant time/power algorithms
  - Cache invalidation
  - CPU microcode update to make vulnerable instruction clear prediction cache



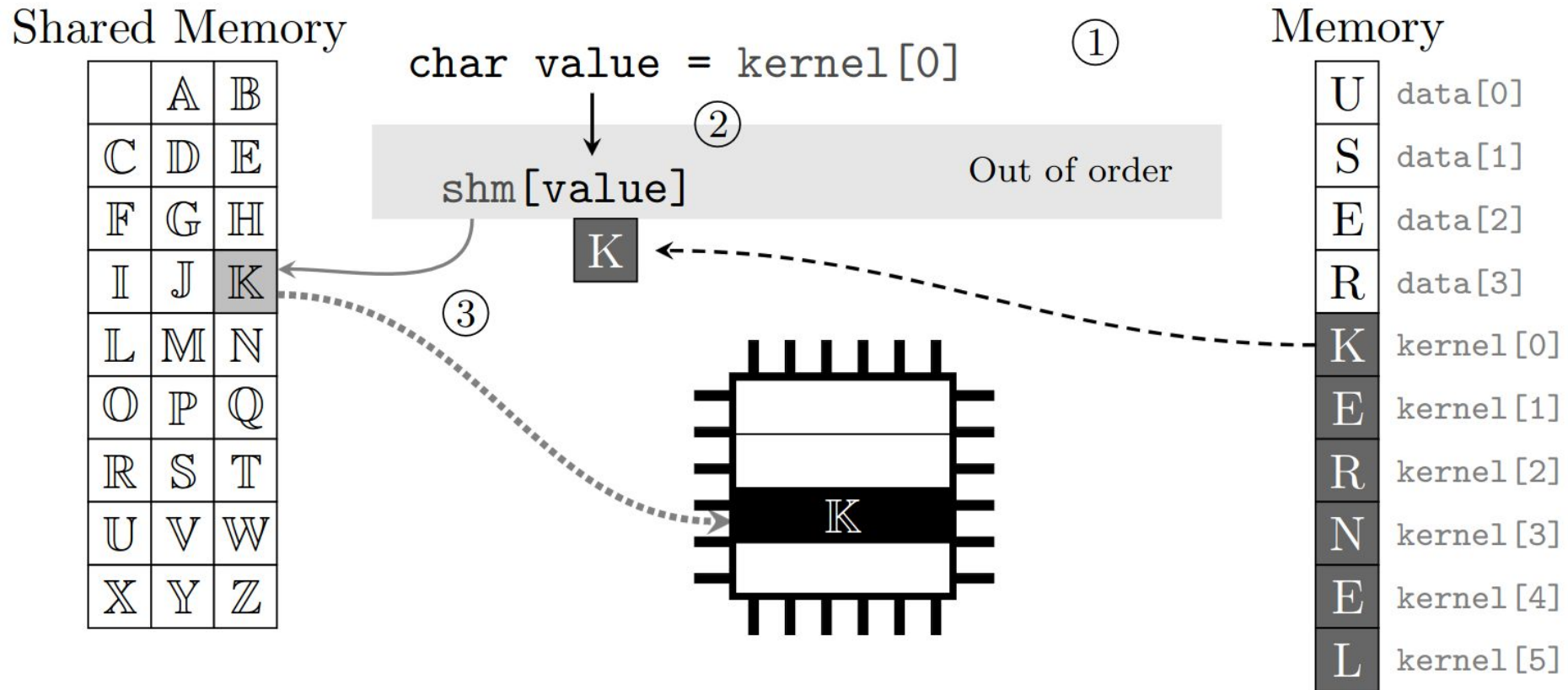
# Timing attack examples

- Password comparison stopping at first character...
- Asymmetric crypto math implementations:
  - Modular exponentiation depends on number of '1' bits + chosen ciphertext  
=> measure execution time => reveal key using statistical correlation
- Solution: hardened software algorithms!
  - Make all branches have same duration / power consumption / etc.

# Cache/timing/transient execution

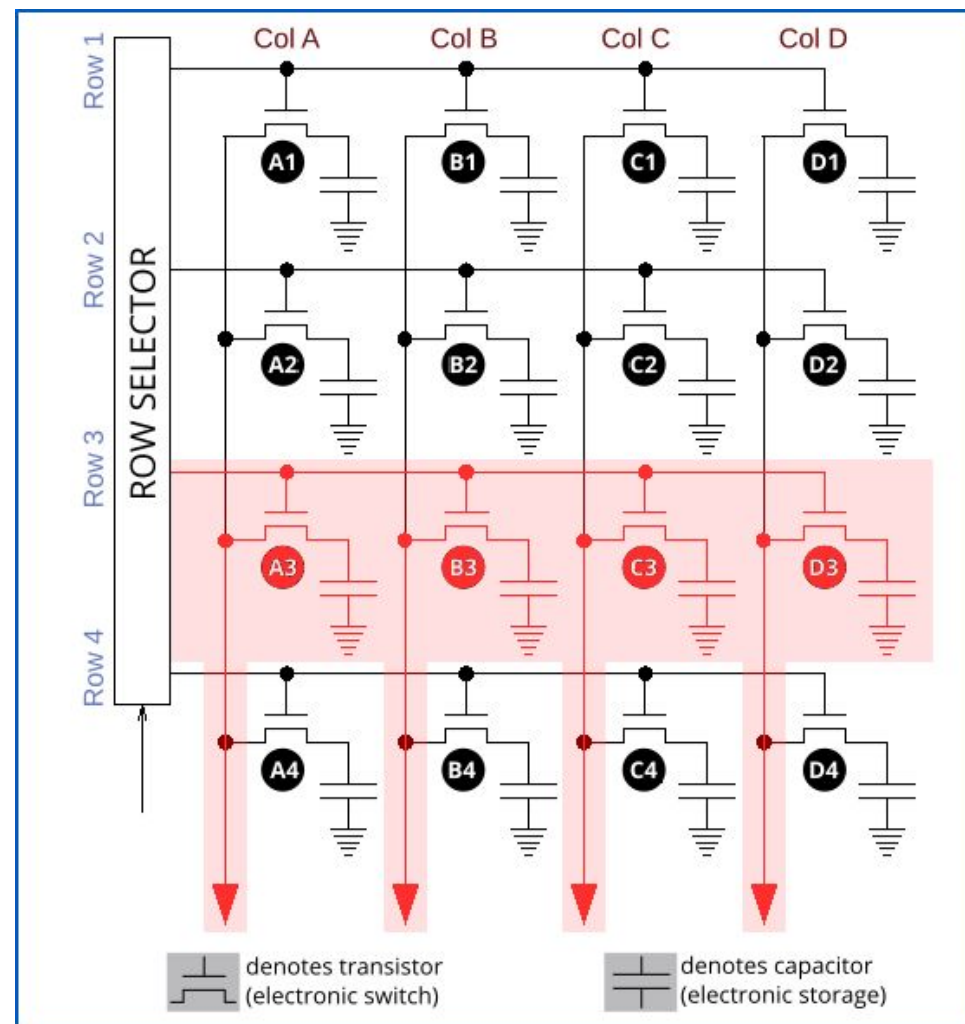
- Hardware optimizations can be a source of side channels:
  - Cache misses >> hits (time)
- Micro-architectural transient execution vulnerabilities
  - Speculative execution / branch prediction attacks
  - Spectre & Meltdown (2017-2018)
  - Foreshadow – extract data from SGX (in decrypted cache)
- Fix: compiler instrumentation + CPU microcode!
- Mitigation impact: between 5-30% (depending on workload)

# Meltdown concept



# RowHammer

- DRAM: optimized for cost!
  - 1 transistor per byte
  - Organized in rows
- Each row must be periodically refreshed!
- Starvation + electrical currents may charge / discharge adjacent rows!
- Fortunately, not entirely practical from JavaScript!



# References

1. <http://repository.cmu.edu/cgi/viewcontent.cgi?article=1096&context=cylab>
2. IBM Press - A Practical Guide to Trusted Computing, Jan 2008
3. Trusted Computing Introduction  
<https://www.cs.ox.ac.uk/files/1873/RR-08-11.PDF>
4. <http://asokan.org/asokan/Padova2014/tutorial-mobileplatsec.pdf>
5. <http://resources.infosecinstitute.com/uefi-and-tpm/>
6. [http://fc16.ifca.ai/preproceedings/25\\_Lang.pdf](http://fc16.ifca.ai/preproceedings/25_Lang.pdf)

# References

7. Real-time Kernel Protection from the ARM TrustZone Secure World  
<http://www.cse.unsw.edu.au/~cs9242/15/exam/paper2.pdf>
8. Samsung KNOX TrustZone implementation:  
[http://www.samsung.com/ro/business-images/insights/2015/An\\_Overview\\_of\\_the\\_Samsung\\_KNOX\\_Platform\\_V1.11-0-0.pdf](http://www.samsung.com/ro/business-images/insights/2015/An_Overview_of_the_Samsung_KNOX_Platform_V1.11-0-0.pdf)
9. RowHammer: <https://arxiv.org/pdf/2211.07613.pdf>