

# Trusted Computing Platforms

**Jonathan M. McCune**  
**jonmccune@cmu.edu**

**2010.09.13**

# What is Trusted Computing?

- **Controversial topic**
  - “Treacherous Computing” according to R. Stallman
- **Origin of Trusted Platform Module (TPM) chip**
  - “Fritz Chip” according to Ross Anderson
- **Trusted not to run “unauthorized” programs**
  - By software vendors?
  - By users?
- **Advocates**
  - Make computers safer (fewer viruses, malware, ...)
- **Opponents**
  - Too much power and control into software vendors
- **You**
  - Make an informed decision

# Goals for Today

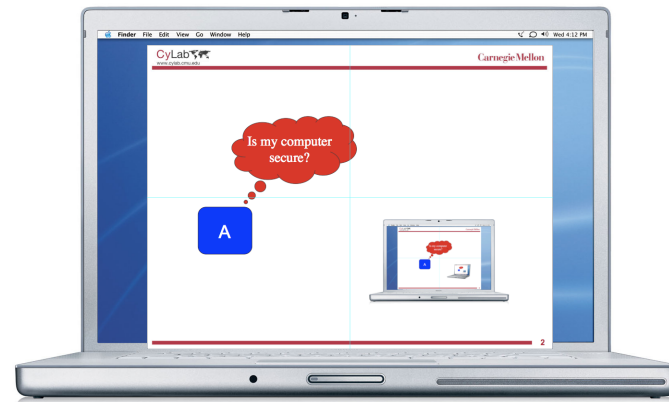
- **Introduce Trusted Computing landscape**
- **Get the facts right about the technology**
- **Understand industry trends**
- **Understand some relevant research**
- **Be informed when discussing policy next time**

# Trusted vs Trustworthy

- A trusted system or component is one whose failure can break the security policy
- A trustworthy system or component is one that will not fail
- Trusted Computing as used in the title of this lecture is really intended to mean “Towards Trustworthy Computing”

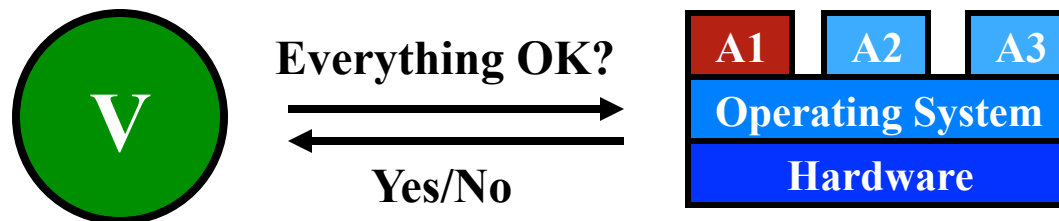
Is my computer  
secure?

A



# Externally Verifiable?

- **Desirable property: Remotely verify trustworthy device operation**



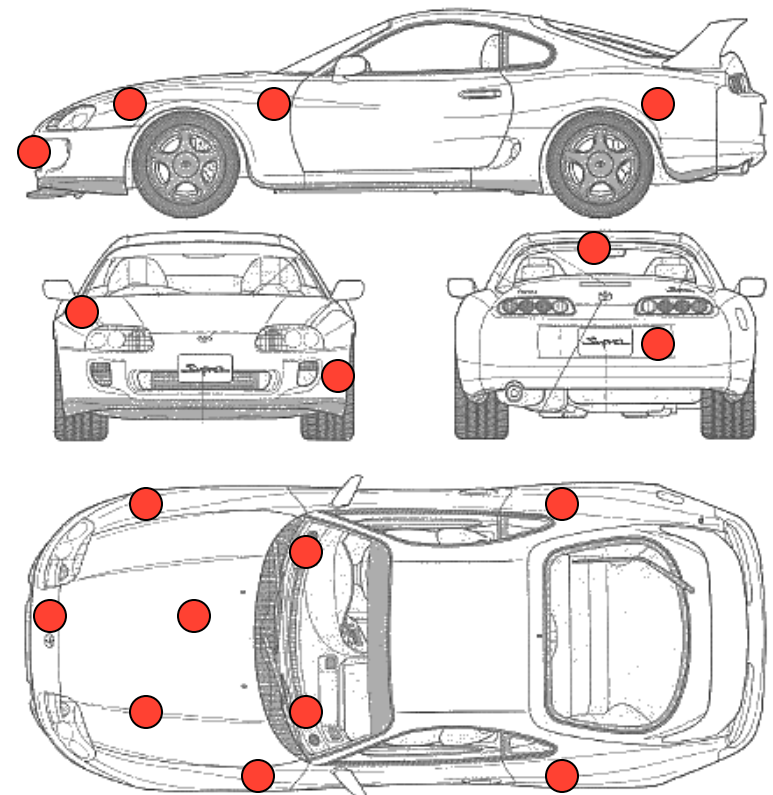
# Embedded Systems Example

- **Computers are everywhere**
  - Uses electricity? Likely to have a CPU.
  - Additional devices are emerging (e.g., thermometers)
- **Embedded processors enable new features, BUT**
  - Features increase complexity
  - Complexity results in bugs
  - Bugs require software updates to fix (today, anyways)
- **Trend: embedded systems become networked**
  - Network access enables many features

**Scary: Embedded systems with network access and code update features**

# Example: Vehicular Embedded Networks

- **Technology trends**
  - **Steady increase in number and complexity of processing units**
    - Regenerative braking, GPS, in-car entertainment, safety systems
  - **Car communication systems**
    - DSRC, cellular technologies, BlueTooth, USB, OnStar
- **“Ford rolls out software fix for hybrid brakes” CNN  4/2010**
- **Security challenges:**
  - **Vehicular malware!**



# Example: Tuning Protection

## ■ Problem

- Individuals alter engine controller software to get more power from engine

## ■ Consequences

- Engine damage
  - Who is liable for engine damages?
- Next-gen vehicle-to-vehicle safety systems
  - Who is liable for crashes?

## ■ Challenge

- How can we verify the software currently running in the engine controller?



# Challenge

- **How can we build secure systems with the following properties**
  - **Highly distributed system**
  - **Large-scale**
  - **Networked devices using wireless communication**
  - **Resource-constrained environment (low-power isn't just for batteries)**
  - **Non-expert users!**
  - **Protects against powerful remote adversary**
- **This is hard!**

# Attestation to the Rescue!

- Attestation enables us to verify what software is executing on a potentially untrusted device
- Software code integrity is an extremely powerful property for building secure systems
- Example: Tuning protection using attestation



What SW is running?



Hash(Software)



# Outline

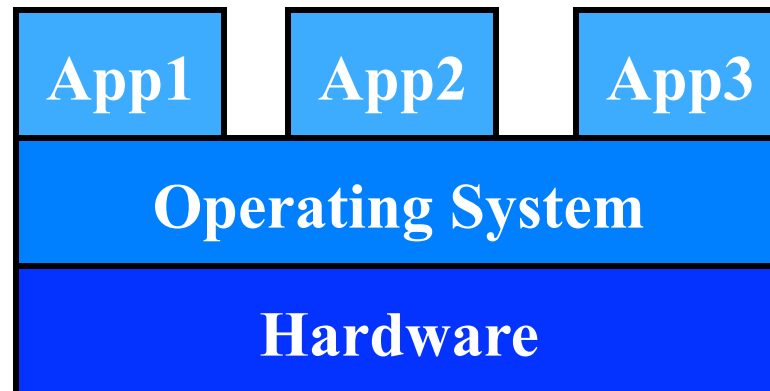
- **Review of some current approaches for building secure (trustworthy) systems**
- **Commodity TPM-based attestation**
  - **Static Root of Trust (version 1.1b)**
  - **Second-generation TCG (version 1.2)**
  - **AMD/Intel secure virtual machine extensions**
- **Software-based attestation**
  - **Promising research direction**
  - **Enables verifiable computing without dedicated hardware**

# Adversary Model

- **Axiom: Every system has at least one more flaw**  
☺
- **We assume remote adversary who can launch network-based attacks**
  - Adversary can compromise OS and/or applications
  - Adversary may control network communication
- **We trust local hardware, local hardware attacks are even harder to defend against**
- **Practical model, as remote attacks constitute majority of threats against commodity systems**

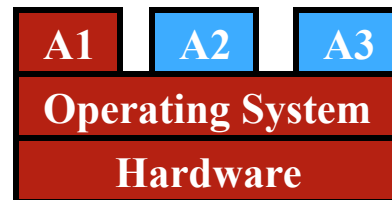
# Security Properties to Consider

- **Trustworthy device operation**
  - How can we trust operations that our devices perform?
- **Questions to consider**
  - How can we trust App1?
  - What if App2 has a security vulnerability?
  - What if Operating System has a security vulnerability?



# Some Current Approaches

- Program code in ROM
- Secure boot
- Virtual-machine-based isolation
  
- Evaluation metric: size of Trusted Computing Base (TCB)
- We visualize components in TCB in red:



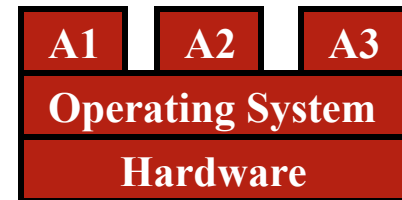
# Program Code in ROM

- **Approach: keep entire program in ROM**
- **Advantages**
  - **Simplicity**
  - **Adversary cannot inject any additional software**
- **Disadvantages**
  - **Cannot update software (without exchanging ROM)**
  - **Adversary can still use control-flow attack**
  - **Entire system is in TCB, no isolation**
- **Verdict**
  - **Impractical for current systems, ability to update code for enhancing features or fixing bugs is critical**



# “Secure” Boot

- **Each component of the boot process verifies following component to be loaded**
  - **Example: digital signature on each boot component; boot loader contains public key and verifies digital signature on OS, etc.**
- **Advantages**
  - **Only approved software can be loaded (assuming no vulnerabilities)**
- **Disadvantages**
  - **Adversary only needs to compromise single component**
  - **Entire system is in TCB, no isolation**
- **Verdict: Entire system is still part of TCB, Relatively weak security guarantee**



# Virtual-machine-based Isolation

- Approach: Isolate applications by executing them inside different Virtual Machines

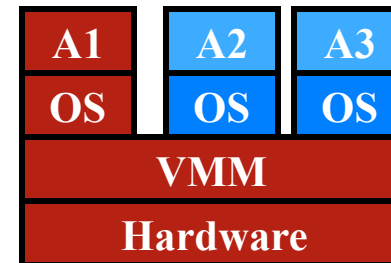
- Advantages

- Smaller TCB
- Isolation between applications

- Disadvantages

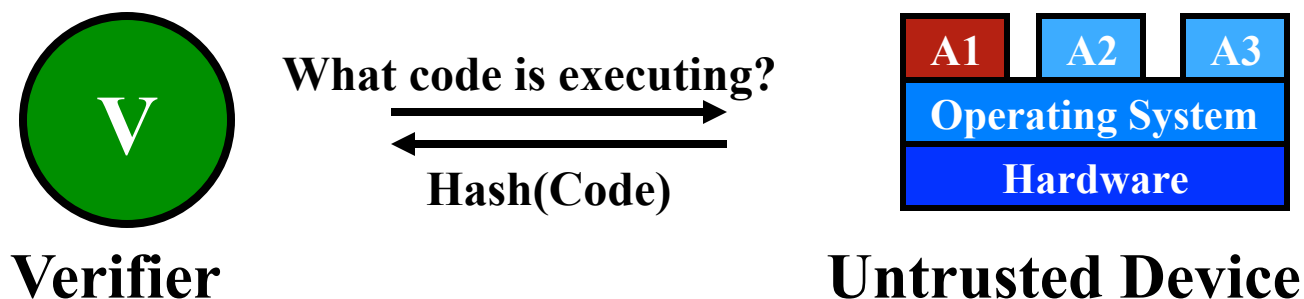
- VMM is still large and part of TCB
- Relatively complex, not well suited for average user

- Verdict: Smaller TCB, step in right direction



# Remote Attestation

- **Attestation enables verifier to establish trust in untrusted device**
  - **Attestation tells verifier what code is executing on device**
  - **If intended code is executing on untrusted device, verifier can trust its operation**



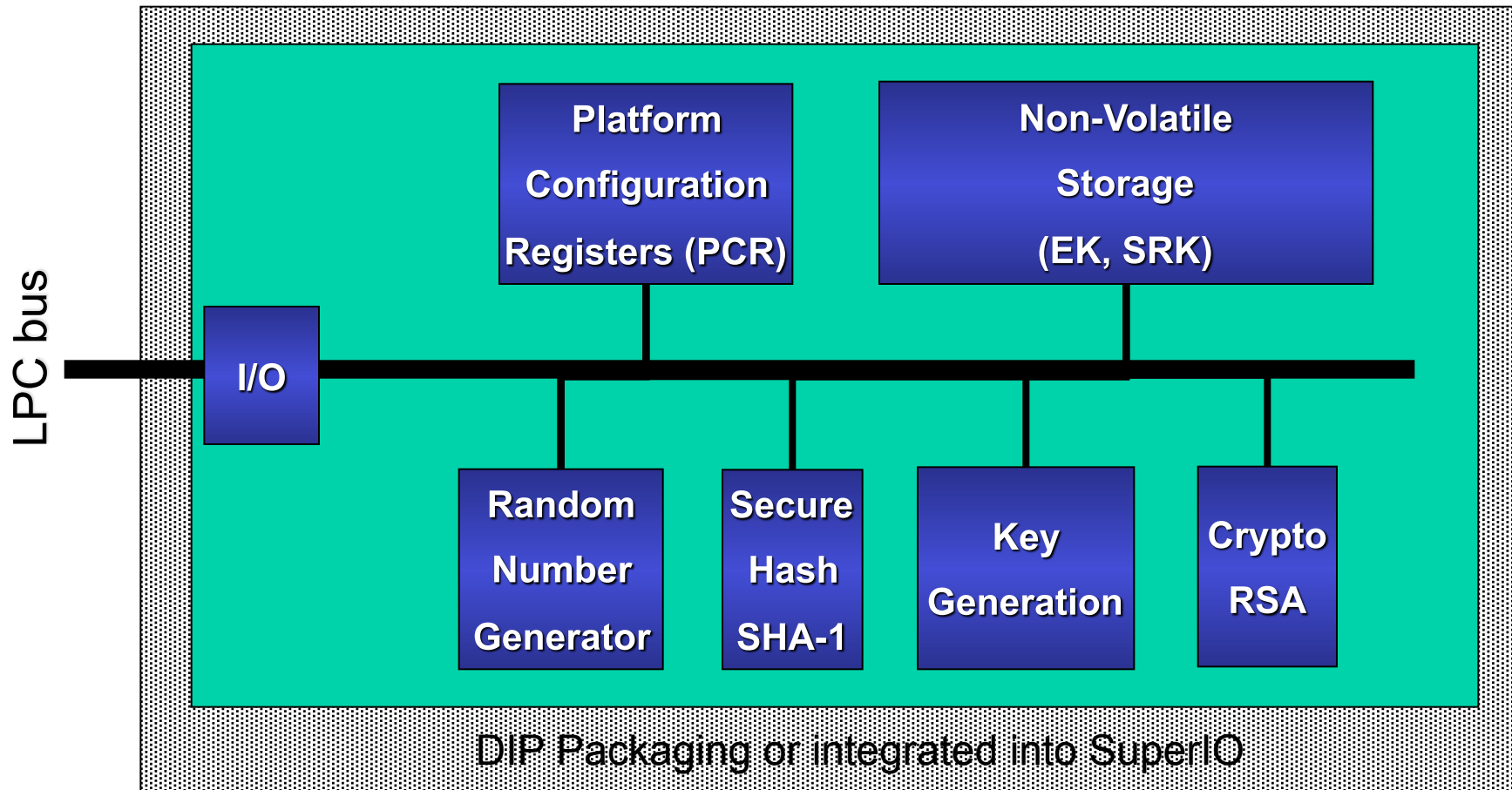
# Hardware-based Attestation

- **Leverages hardware support for attestation**
- **Trusted Computing Group (TCG) proposed Trusted Platform Module (TPM) chip**
  - **Already included in many platforms (200M+)**
  - **Cost per chip less than \$1**
- **Modern microprocessors provide special instructions that interact with TPM chip**
  - **AMD SVM: SKINIT instruction**
  - **Intel TXT/LT: GETSEC[SENDER] instruction**

# Trusted Computing Group (TCG)

- Open organization to “develop, define, and promote open standards for hardware-enabled trusted computing and security technologies.”
  - Desktops, laptops, servers, cell phones, PDAs, ...
  - Industry consortium by AMD, IBM, Intel, HP, Microsoft, ...
- These secure platform primitives include
  - Platform integrity measurements
  - Measurement attestation
  - Sealed storage
- Can enable
  - **Trusted boot** (not secure boot)
  - **Attestation**, which lets a remote verifier check integrity of software
- Goals:
  - Ensure absence of malware
  - Detect spyware, viruses, worms, ...

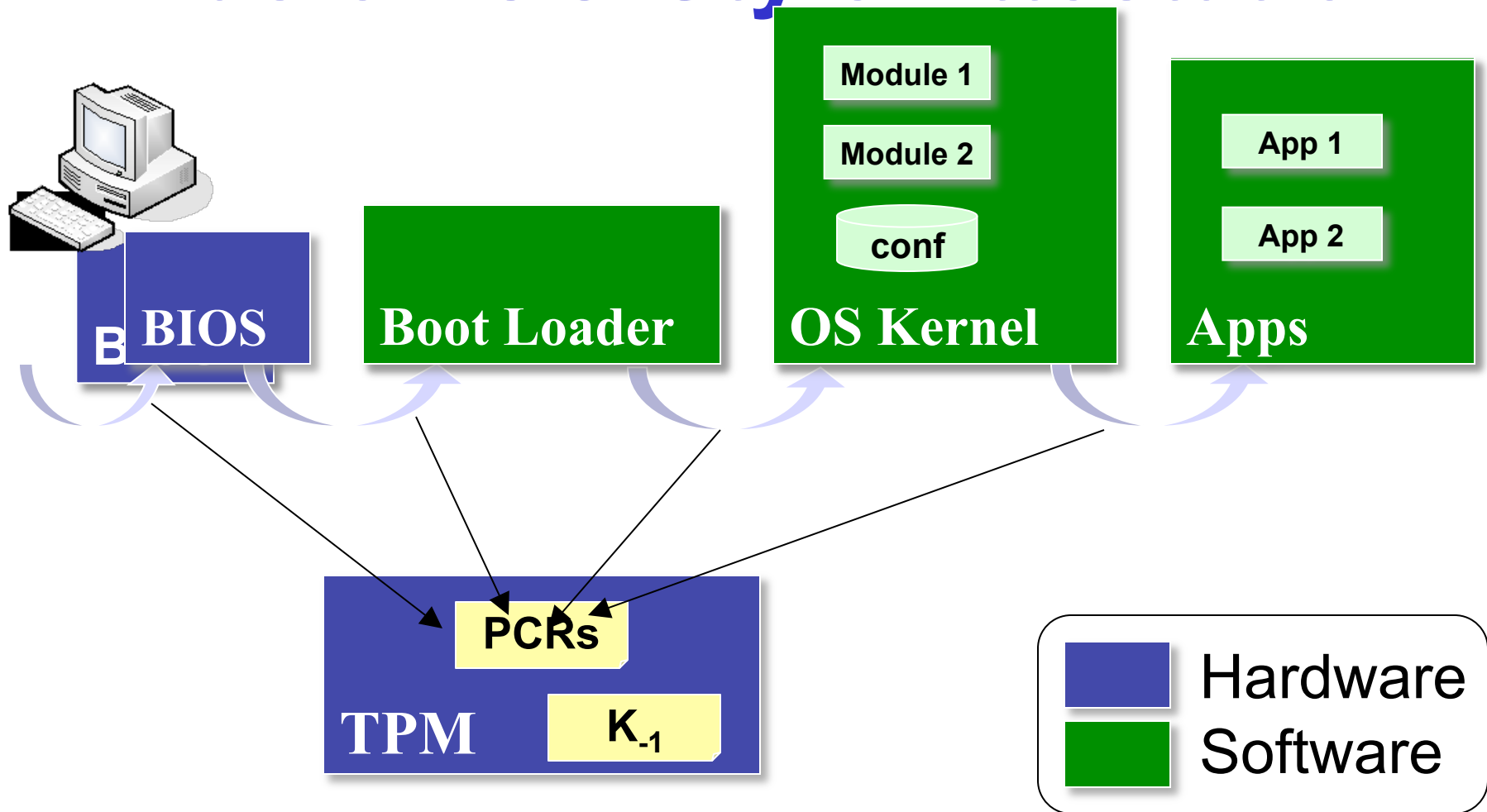
# TCG Trusted Platform Module (TPM)



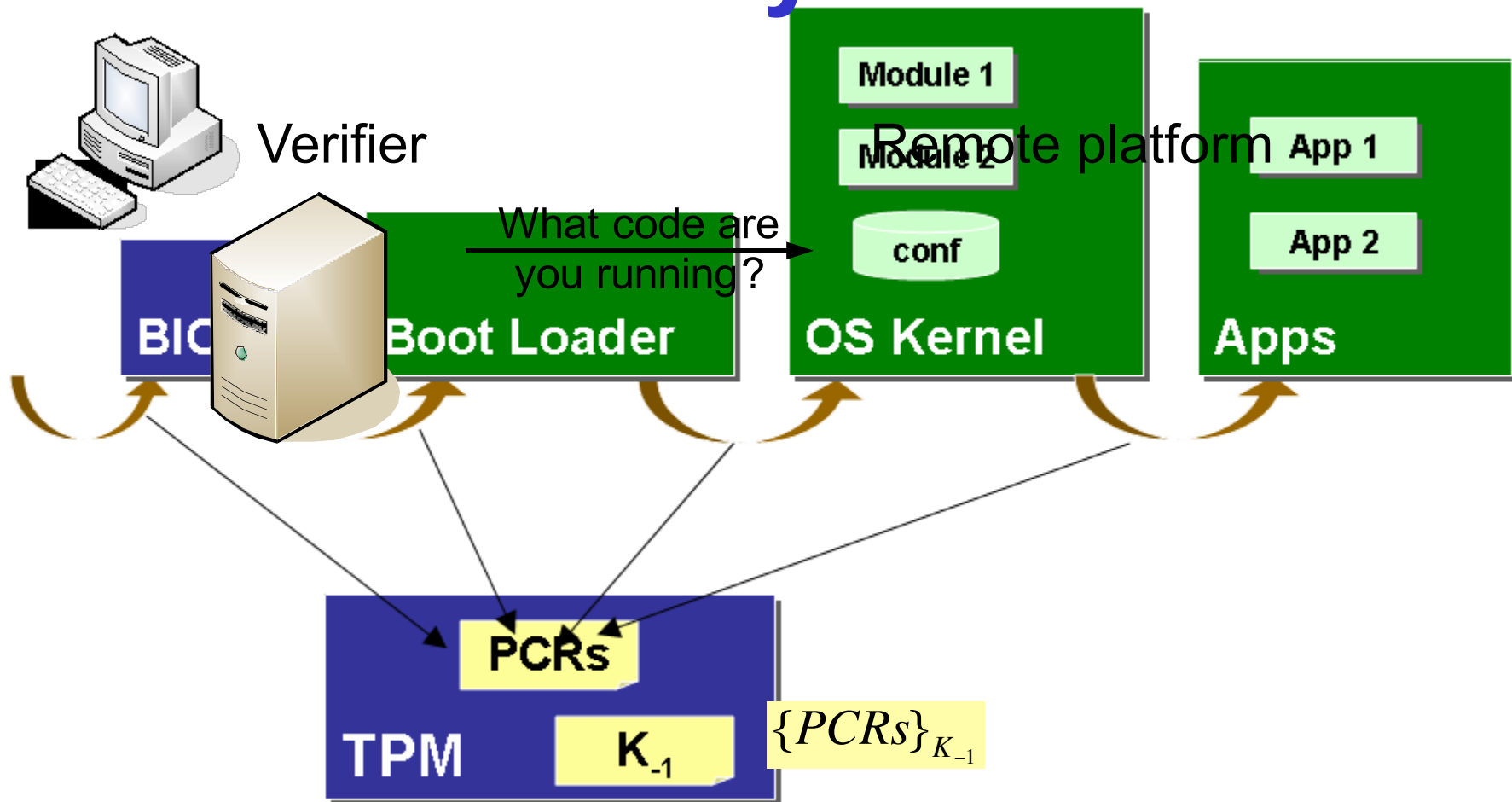
# Basic TPM Functions

- **PCRs store integrity measurement chain**
  - $\text{PCR}_{\text{new}} = \text{SHA-1}(\text{PCR}_{\text{old}} || \text{SHA-1}(\text{data}))$
- **On-chip storage for Storage Root Key  $K^{-1}_{\text{SRK}}$**
- **Manufacturer certificate, e.g.,  $\{K_{\text{TPM}}\}K^{-1}_{\text{IBM}}$**
- **Remote attestation (PCRs + AIK)**
  - Attestation Identity Keys (AIKs) for signing PCRs
  - Attest to value of integrity measurements to remote party
- **Sealed storage (PCRs + SRK)**
  - Protected storage + unlock state under a particular integrity measurement (data portability concern)

# Basic TCG-Style Attestation



# Basic TCG-Style Attestation



# Example: TCG on Linux

- Integrity Measurement Architecture (IMA) by IBM
- Measurement principles
  - Whole system measurements
  - Measure all executable content **on-demand**
    - Too expensive to measure whole system
    - Content is added dynamically
  - Measure content **before** execution
    - Only measured content can introduce and measure new content
  - Place as little trust as necessary in measurement system

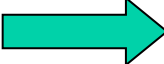
# Linux Modifications

- **Added support to Linux kernel**
  - To measure dynamic linker
  - To measure each executable
- **Added support to dynamic linker**
  - To measure each shared library
- **Added support to kernel module loader**
  - To measure kernel modules
- **Kernel keeps a measurement cache**
  - **Files are only measured once!**
  - Unless modified (opened for writing)
- **What are they forgetting?**

# Linux Application Measurements

cat /proc/tcg/measurements

```
#000: 276249898F406BE176E3D86EDD5A3D20D03EEB11 [remeasure] linuxrc
#001: 9F860256709F1CD35037563DCDF798054F878705 [remeasure] nash
#002: 4CC52A8F7584A750303CB2A41DEA637917DB0310 [clean] insmod
#003: 84ABD2960414CA4A448E0D2C9364B4E1725BDA4F [clean] init
#004: 194D956F288B36FB46E46A124E59D466DE7C73B6 [clean] ld-2.3.2.so
#005: 7DF33561E2A467A87CDD4BB8F68880517D3CAECB [clean] libc-2.3.2.so
#006: 93A0BBC35FD4CA0300AA008F02441B6EAA425643 [clean] rc.sysinit
#007: 66F445E31575CA1ABEA49F0AF0497E3C074AD9CE [clean] bash
#008: F4F6CB0ACC2F1BEE13D60330011DF926D24E5688 [clean] libtermcap.so.2.0.8
#009: 346443AAD8E7089B64B2B67F1A527F7E2CA2D1E5 [clean] libdl-2.3.2.so
#010: 02385033F849A2A4BFB85FD52BCEA27B45497C6C [clean] libnss_files-2.3.2.so
#011: 6CB3437EC500767328F2570C0F1D9AA9C5FEF2F6 [clean] initlog
#012: FD1BCAEF339EAE065C4369798ACAADFF44302C23 [clean] hostname
#013: F6E44B04811CC6F53C58EEBA4EACA3FE9FF91A2E [clean] consoletype
#014: 12A5A9B6657EFEE7FD619A68DA653E02A7D8C661 [clean] grep
#015: 3AF36F2916E574884850373A6E344E4F2C51DD60 [clean] sed
#016: CE516DE1DF0CD230F4A1D34EFC89491CAF3D50E4 [clean] libpcre.so.0.0.1
#017: 5EE8CD72AAD26191879E01221F5E051CE5AAE95F [clean] setsysfont
#018: 8B15F3556E892176B03D775E590F8ADF9DA727C5 [clean] unicode_start
#019: F948CF91C7AF0C2AB6AD650186A80960F5A0DAB1 [clean] kbd_mode
#020: FF02DD8E56F0B2DCFB3D9BF392F2FCE045EFE0BC [clean] dumpkeys
#021: C00804432DFBC924B867FC708CB77F2821B4D320 [clean] loadkeys
#022: DE3AC70601B9BA797774E59BEC164C0DDF11982D [clean] setfont
#023: 7334B75FDF47213FF94708D2862978D0FF36D682 [clean] gzip
#024: AEC13AA4FF01F425ACACF0782F178CDFE3D17282 [clean] minilogd
#025: 09410DDC5FE2D6E7D8A7C3CF5BB4D51ED6C4C817 [clean] sleep
```

 PCR<sub>10</sub>

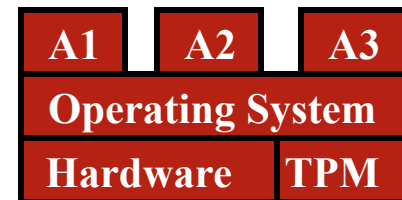
# Linux Application Measurements

```
cat /proc/tpm/pcrs
```

```
PCR-00: 0A 2A B1 F6 56 EA ED 4C 53 F0 C7 9D 5E 05 61 37 51 B7 1C E5
PCR-01: 5F DB 12 AD B3 34 7D D6 90 63 46 72 D8 DE 02 1C F3 3C 00 F7
PCR-02: EB B3 BA AE E7 57 4B B6 37 AA AB 67 0F 9A C1 BC EB 6F 80 F3
PCR-03: 04 FD EC DD 50 1D AF 0F 62 4C 1F 99 60 12 CF 30 44 FF 46 10
PCR-04: 28 E3 E8 F0 CA 34 ED DD 58 AA 7E 71 F6 FC AE 08 C3 88 EB 05
PCR-05: E7 23 99 CD A3 1D 37 E4 35 61 B7 1A 85 68 3B 66 7F 51 B6 B4
PCR-06: 04 FD EC DD 50 1D AF 0F 62 4C 1F 99 60 12 CF 30 44 FF 46 10
PCR-07: 04 FD EC DD 50 1D AF 0F 62 4C 1F 99 60 12 CF 30 44 FF 46 10
PCR-08: DC 0E 38 C4 F4 46 F7 BC DF C8 83 CA CC 86 E2 69 50 C5 0E 66
PCR-09: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-10: 50 48 FF 78 06 63 CB BF A5 F6 43 0B DA 41 1A 15 74 C3 1A 92
PCR-11: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-12: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-13: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-14: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-15: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

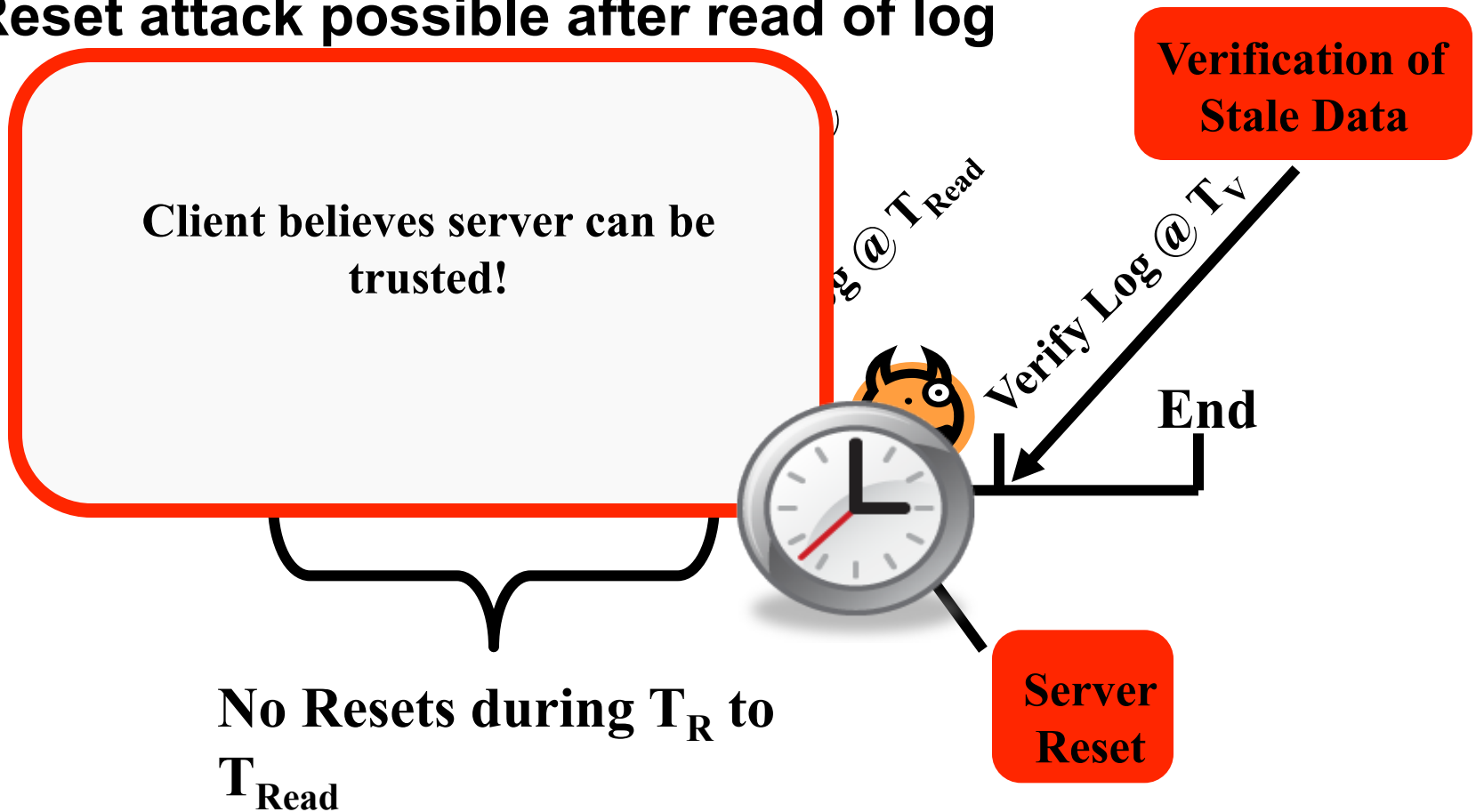
# Shortcomings of TCG-style Attestation

- Integrity measurements are done at **load-time** not at run-time
  - Time-of-check-time-of-use (TOCTOU) problem
  - Cannot detect any dynamic attacks!
- **Coarse-grained, measures entire system**
  - Accumulates hundreds of integrity measurements
  - Every host is different, different firmware versions, different drivers, different patches, different apps, different spyware, ...
  - What does a PCR mean in this context?
  - TCB includes entire system!
- **No guarantee of execution**
- **Requires special hardware: TPM chip**



# Time of Check, Time of Use (TOCTOU)

- Reset attack possible after read of log



# 1. Attesting to Current State

- **Attestation only attests to what code was loaded.**
- **Does not guarantee that the same code is running at the time of check.**
- **Can we attest to the current state of a running system?**
  - **... or is there a better way?**

## 2. Encrypted viruses

- **Suppose malicious music file exploits bug in Windows Media Player.**
  - **Music file is encrypted.**
  - **TCG prevents anyone from getting music file in the clear.**
  - **Can anti-virus companies block virus without ever seeing its code in the clear?**

# 3. TPM Compromise

- **Suppose one TPM Endorsement Private Key is exposed**
  - **Impacts entire attestation infrastructure:**
    - Now, can attest to anything without running it.
  - **Certificate Revocation is critical for TCG Attestation.**
  - **Identifying the TPM in this machine is also critical**

# 4. Private attestation

- **Attestation should not reveal platform ID.**
- **Private attestation:**
  - **Remote server can validate trustworthiness of attestation**
  - **... but cannot tell what machine it came from.**
- **TCG Solutions:**
  - **Privacy CA: online trusted party**
  - **Group sigs: privacy without trusted infrastructure**