CE 874 - Secure Software Systems

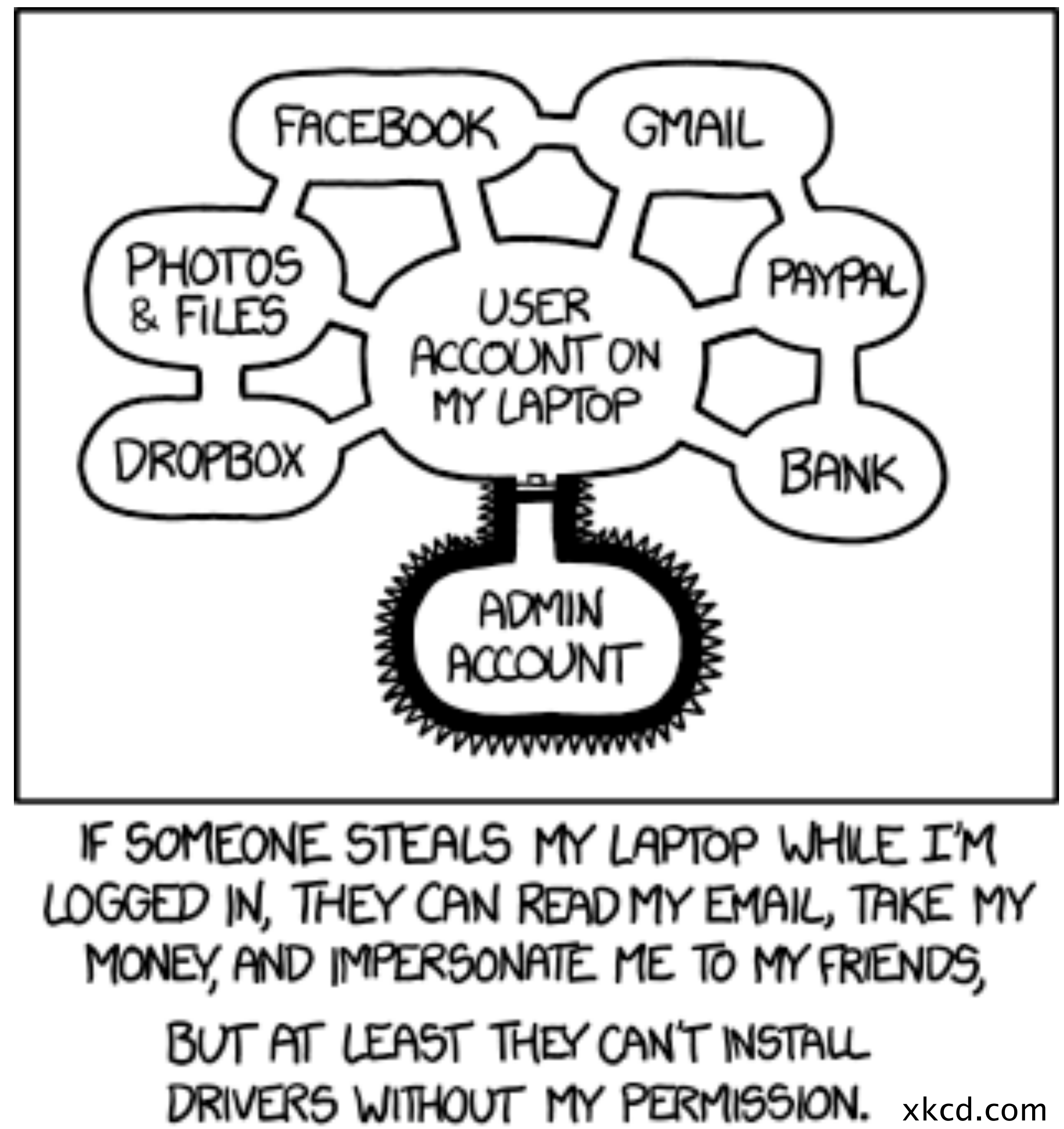
Secure Architecture III

Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology

Acknowledgments: Some of the slides are fully or partially obtained from other sources. A reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.

Secure Architecture

- How to come up with a secure architecture?
- What design principals is should be followed?
- What are the available mechanisms?
- How do you trust the code getting executed?

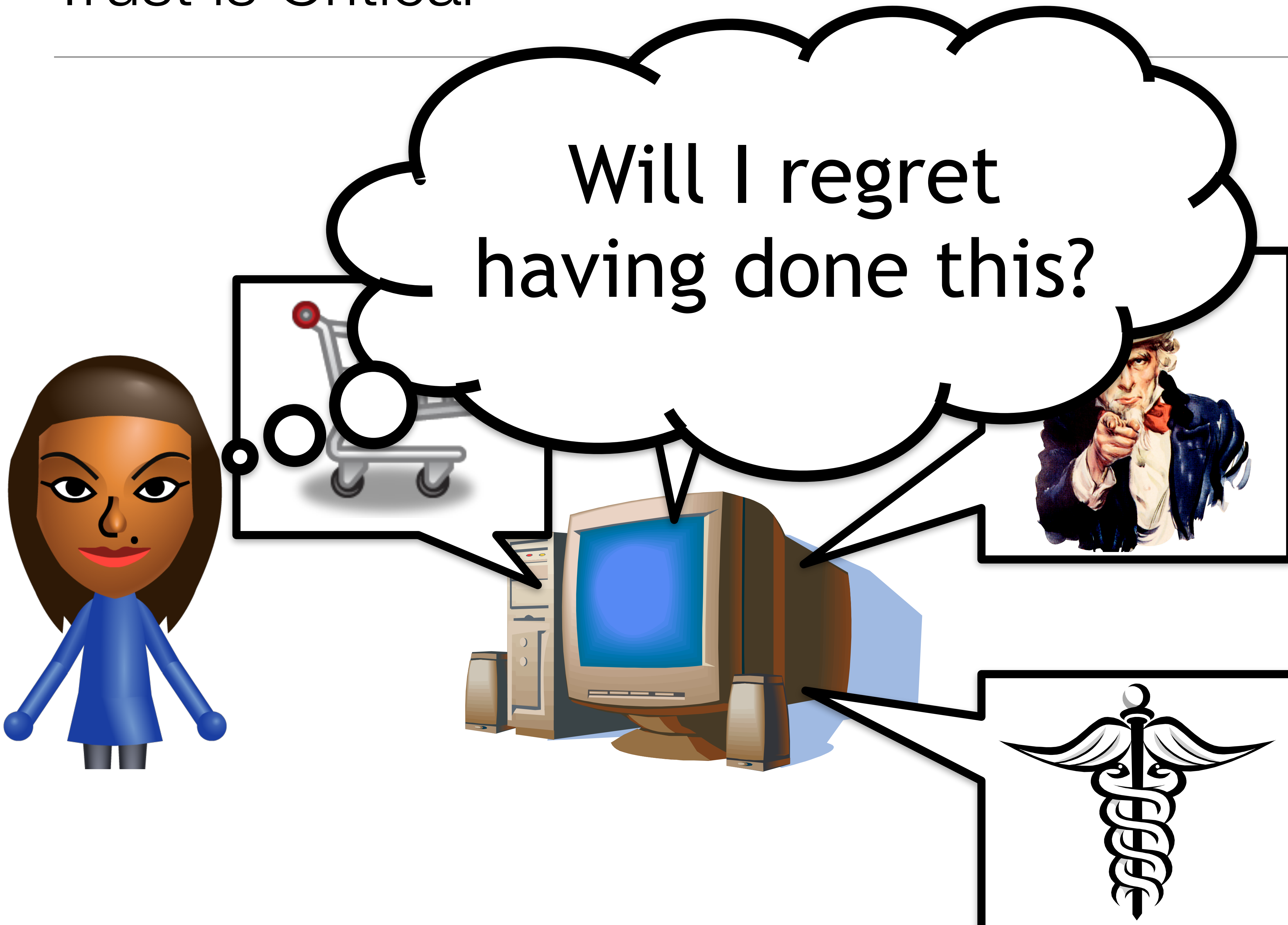


xkcd.com

Bootstrapping Trust in Commodity Computers,
Bryan Parno, Jonathan McCune, Adrian Perrig, IEEE S&P, 2010

A Travel Story

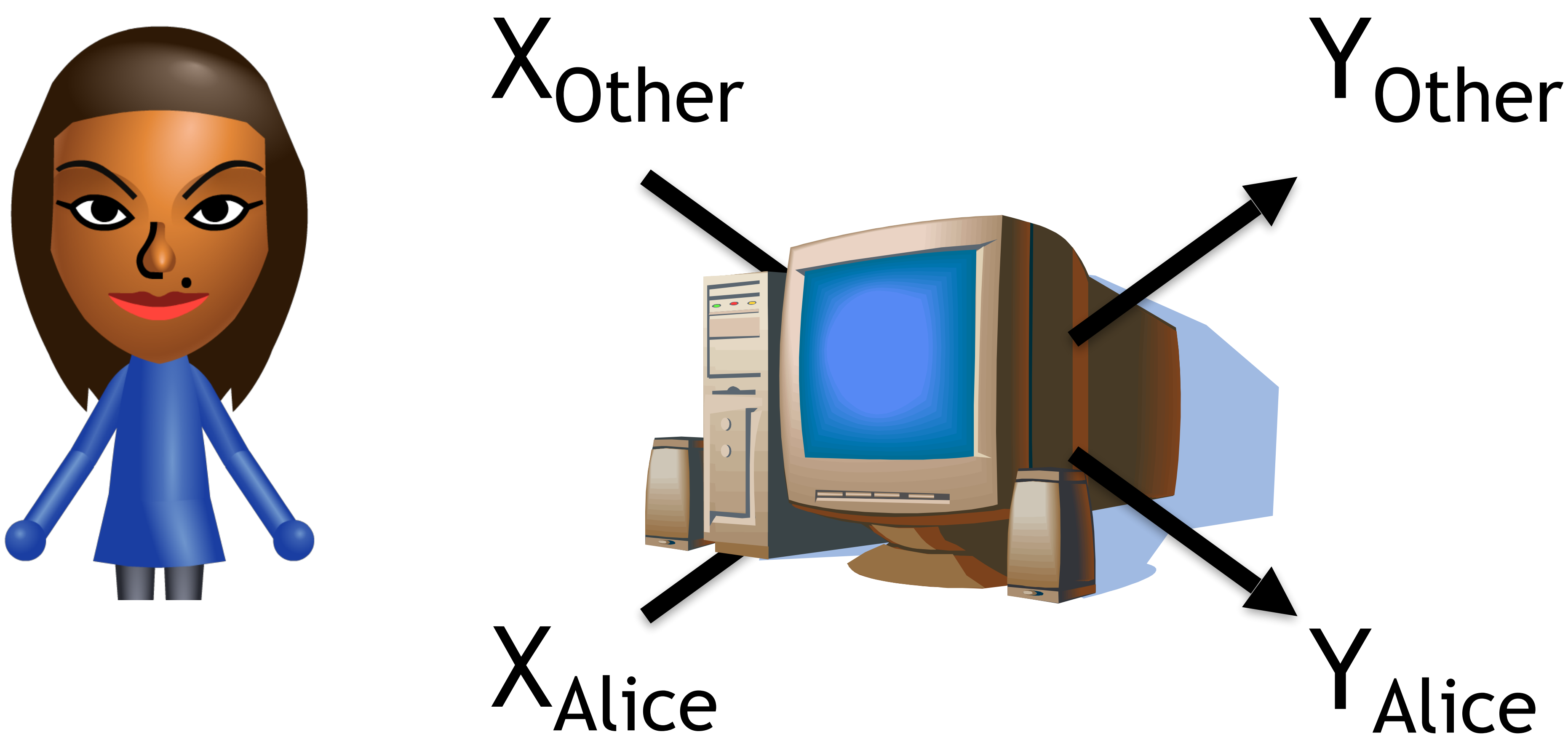
Trust is Critical



Does program P compute F?
Is F what the programmer intended?

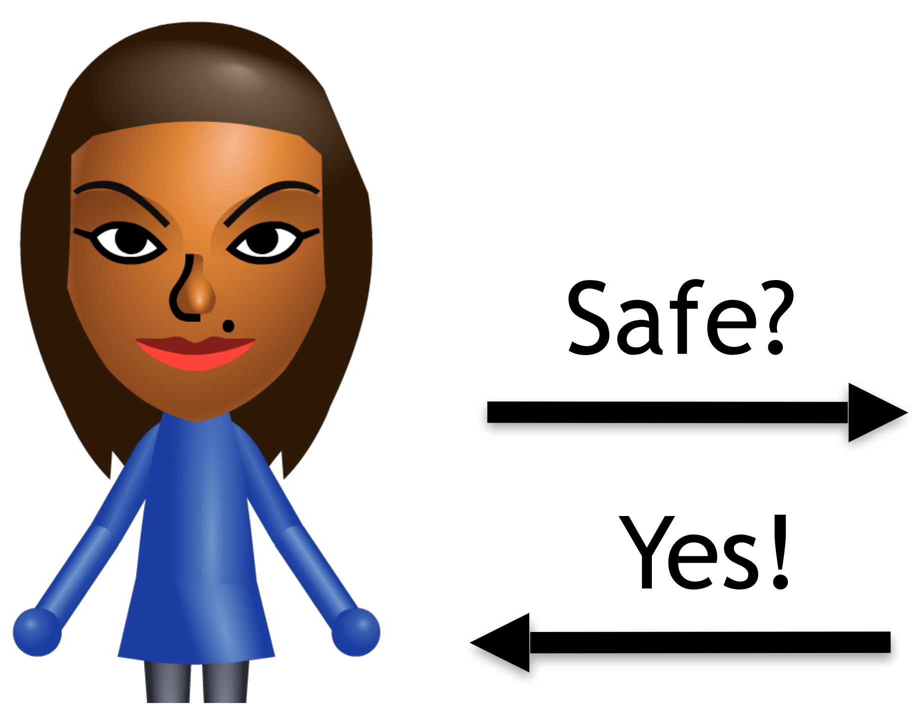
Bootstrapping Trust

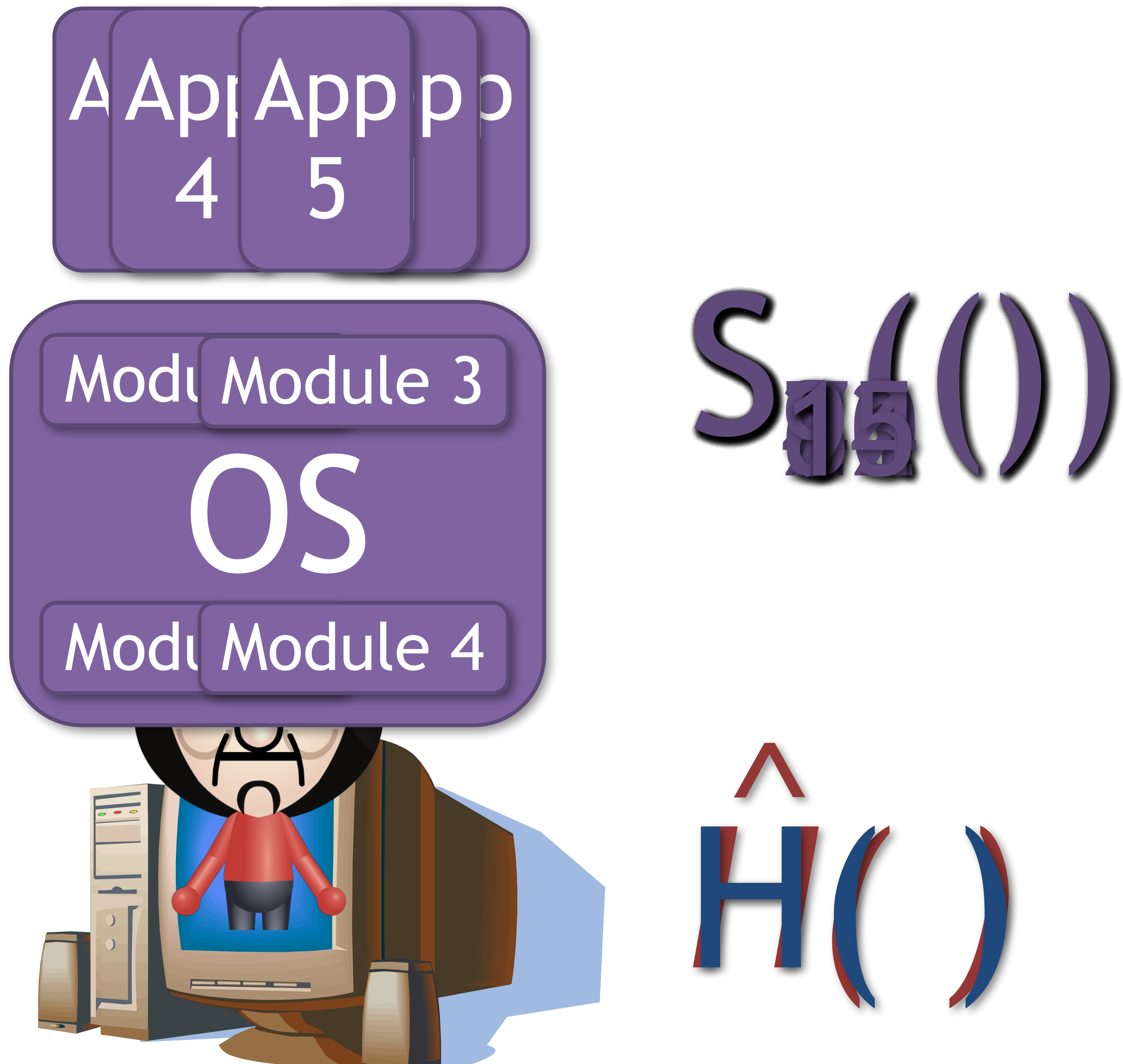
What F will this machine compute?



Bootstrapping Trust is *Hard*!

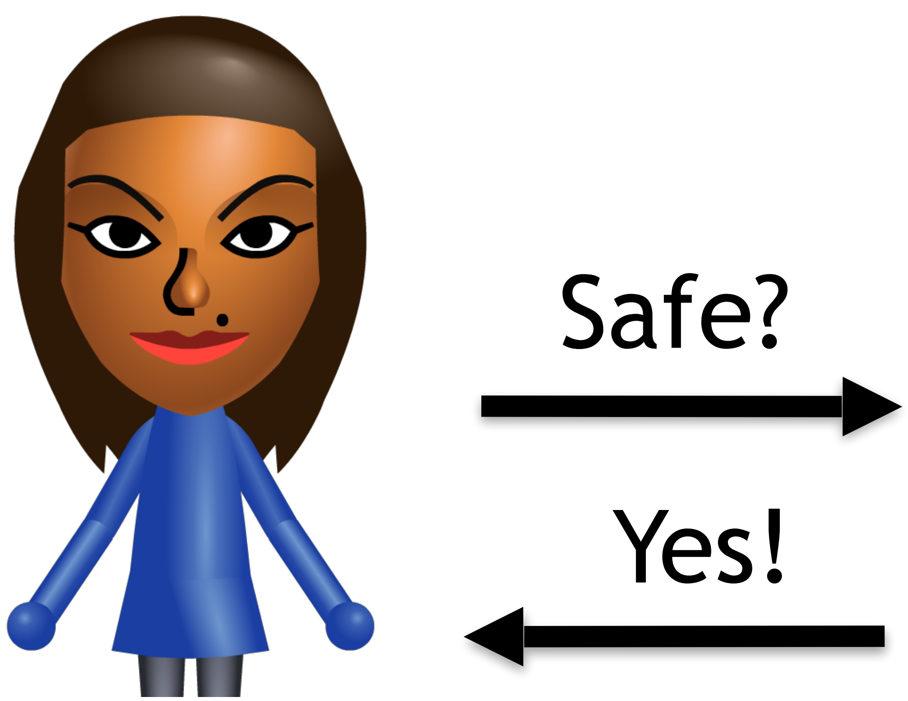
- Challenges:
  - Hardware assurance
  - Ephemeral software
  - User Interaction





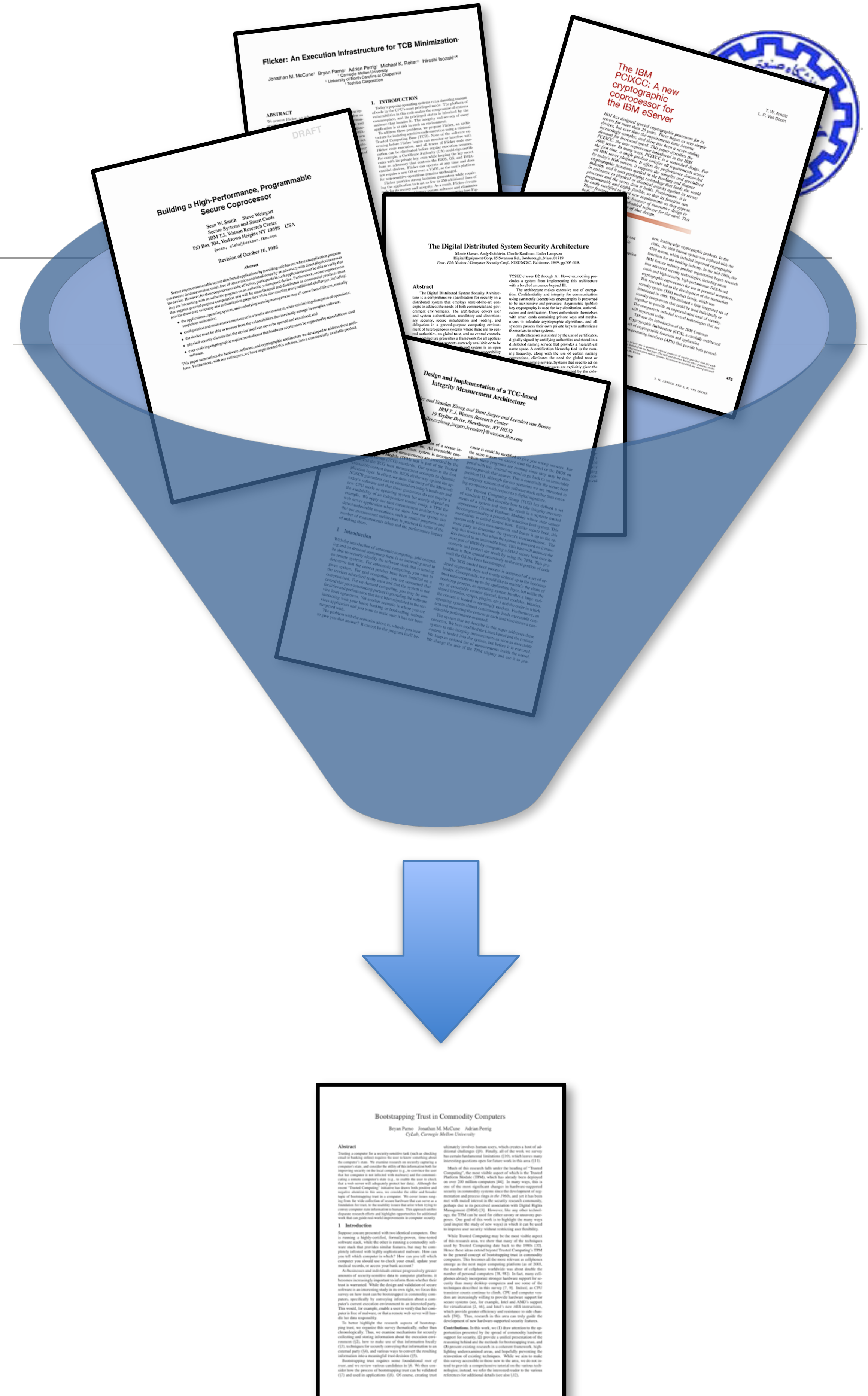
Bootstrapping Trust is *Hard*!

- Challenges:
  - Hardware assurance
  - Ephemeral software
  - User Interaction



In the paper…

- Bootstrapping foundations
- Transmitting bootstrap data
- Interpretation
- Validation
- Applications
- Human factors
- Limitations
- Future directions
- … and much more!

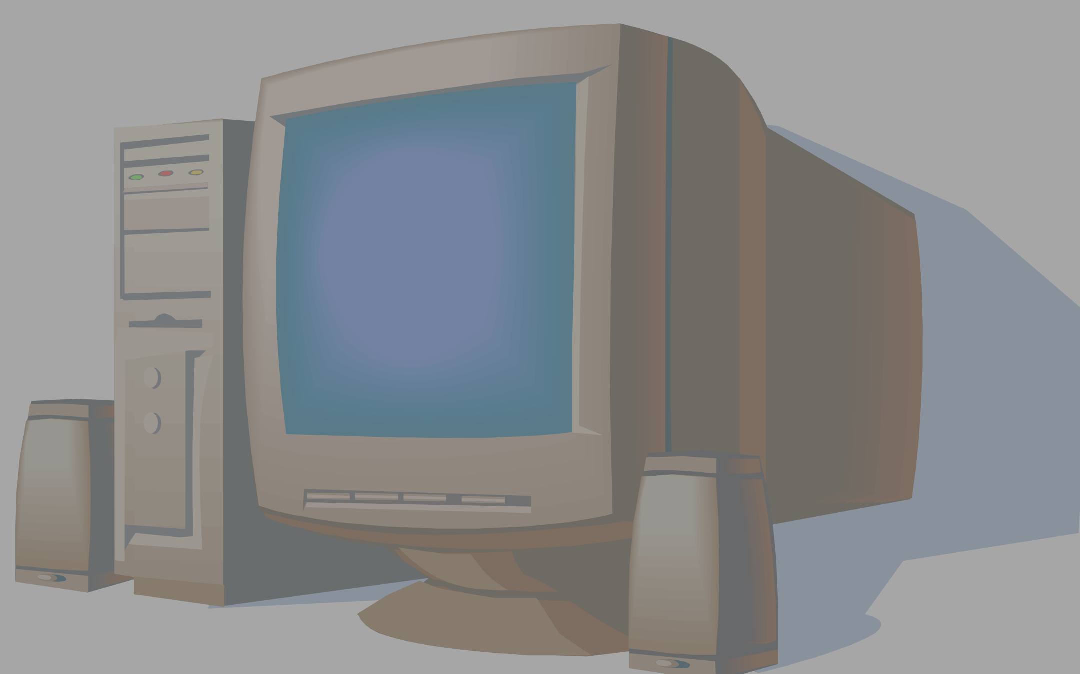


 [Parno'10]

1) Establish Trust in Hardware

- Hardware is durable

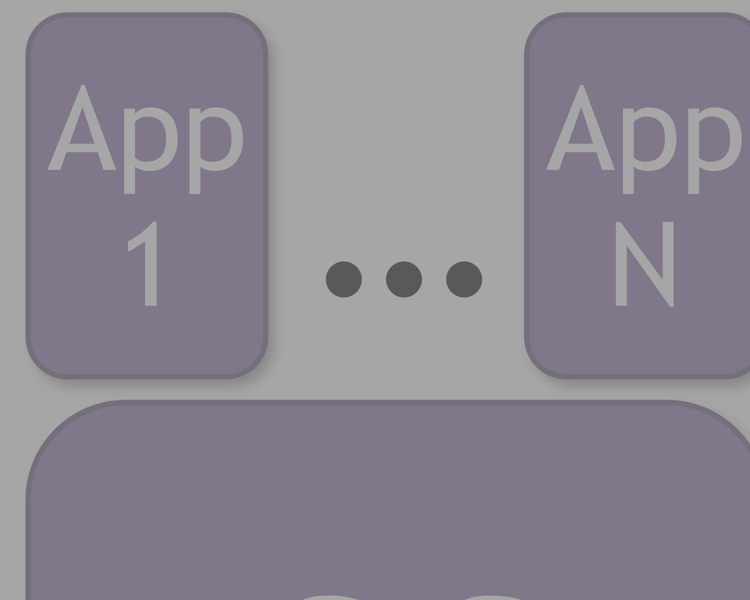
<u>Open Question</u>: Can we do better?



 [Parno’10]

2) Establish Trust in Software

- Software is ephemeral
- We care about the software currently in control
- Many properties matter:
    - Proper control flow
    - Type safety

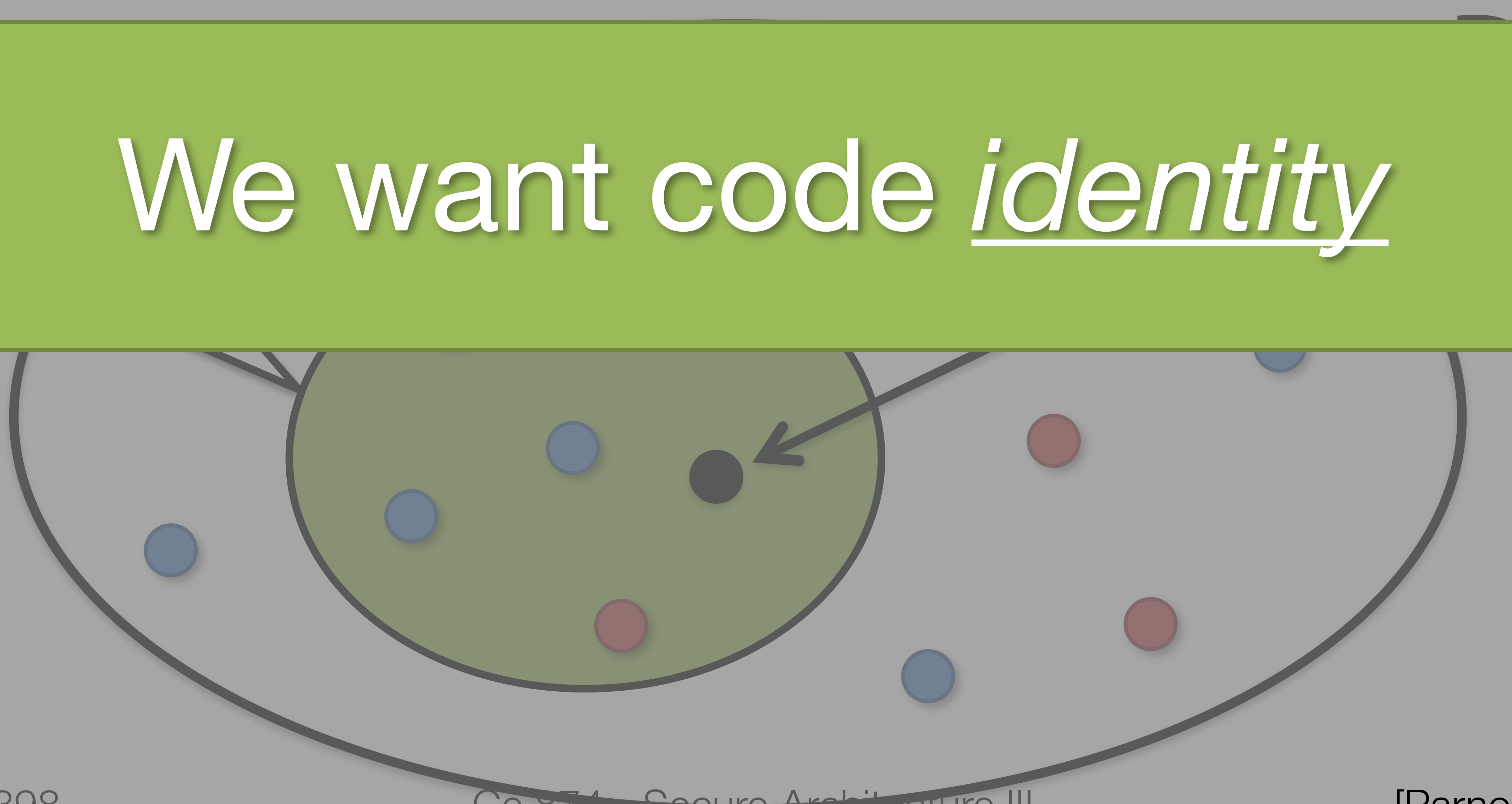


Which property matters most?

A Simple Thought Experiment

- Imagine a perfect algorithm for analyzing control flow
  - Guarantees a program always follows intended control flow
- Does this suffice to bootstrap trust?

No!



We want code *identity*

[Parno'10]

What is Code Identity?

- An attempt to capture the behavior of a program
- Current state of the art is the collection of:
  - Program binary
  - Program libraries
  - Program configuration files
  - Initial inputs
- Often condensed into a hash of the above

Function f

Inputs to f

Code Identity as Trust Foundation

- From code identity, you may be able to infer:
  - Proper control flow
  - Type safety
  - Correct information flow

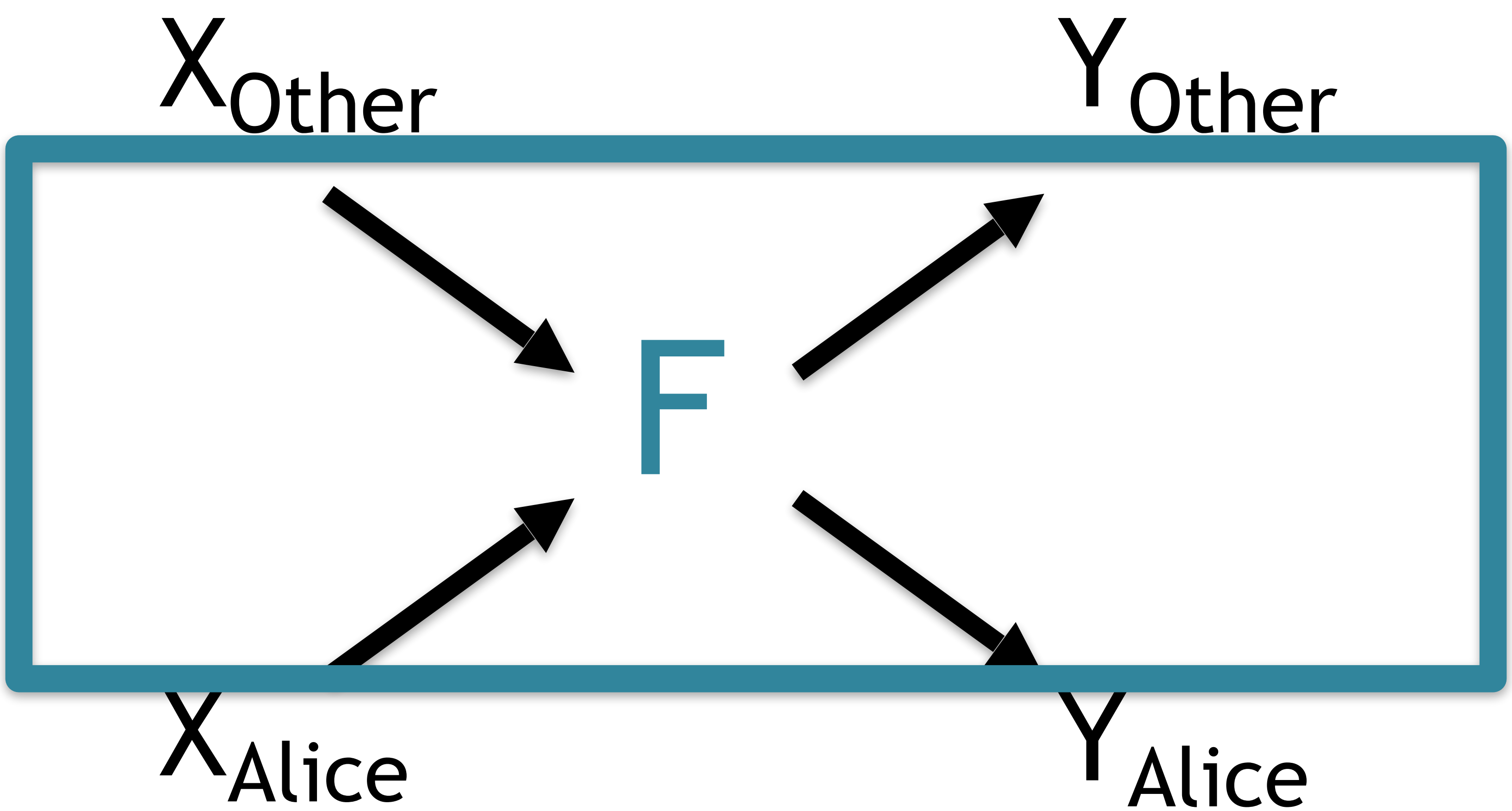
    …
- Reverse is not true!

What Can Code Identity Do For You?

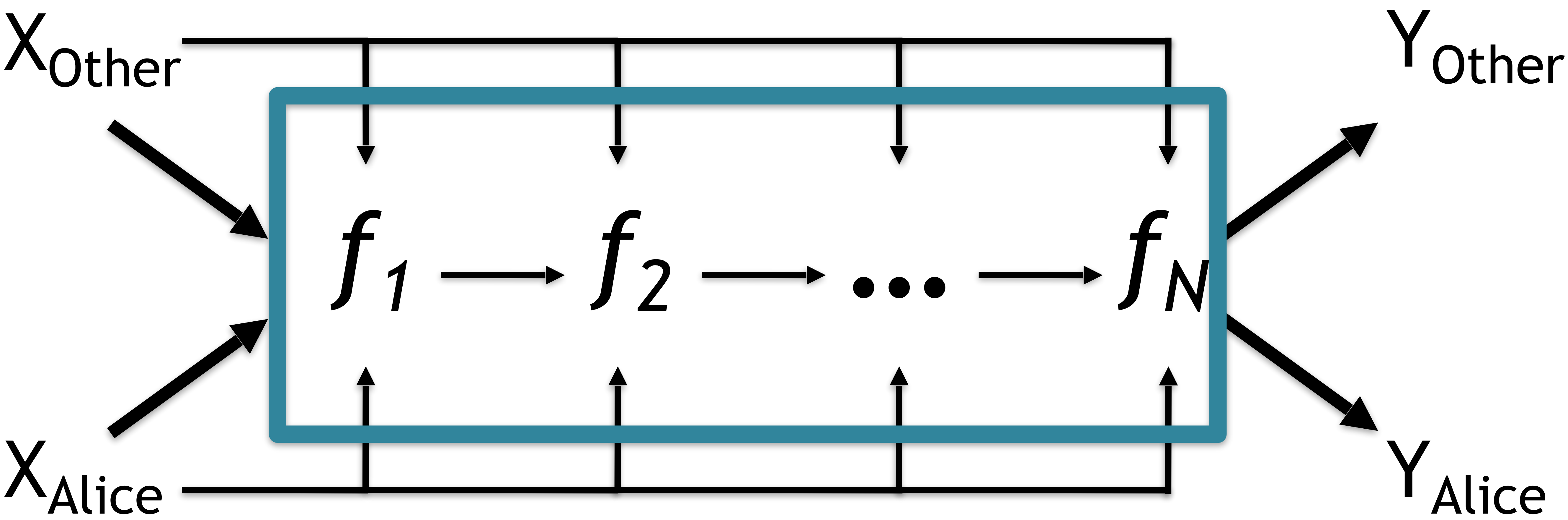
- Research applications
  - Secure the boot process
  - Count-limit objects
  - Improve security of network protocols
  - Thwart insider attacks
  - Protect passwords
  - Create a Trusted Third Party

- Commercial applications
  - Secure disk encryption (e.g., Bitlocker)
  - Improve network access control
  - Secure boot on mobile phones
  - Validate cloud computing platforms

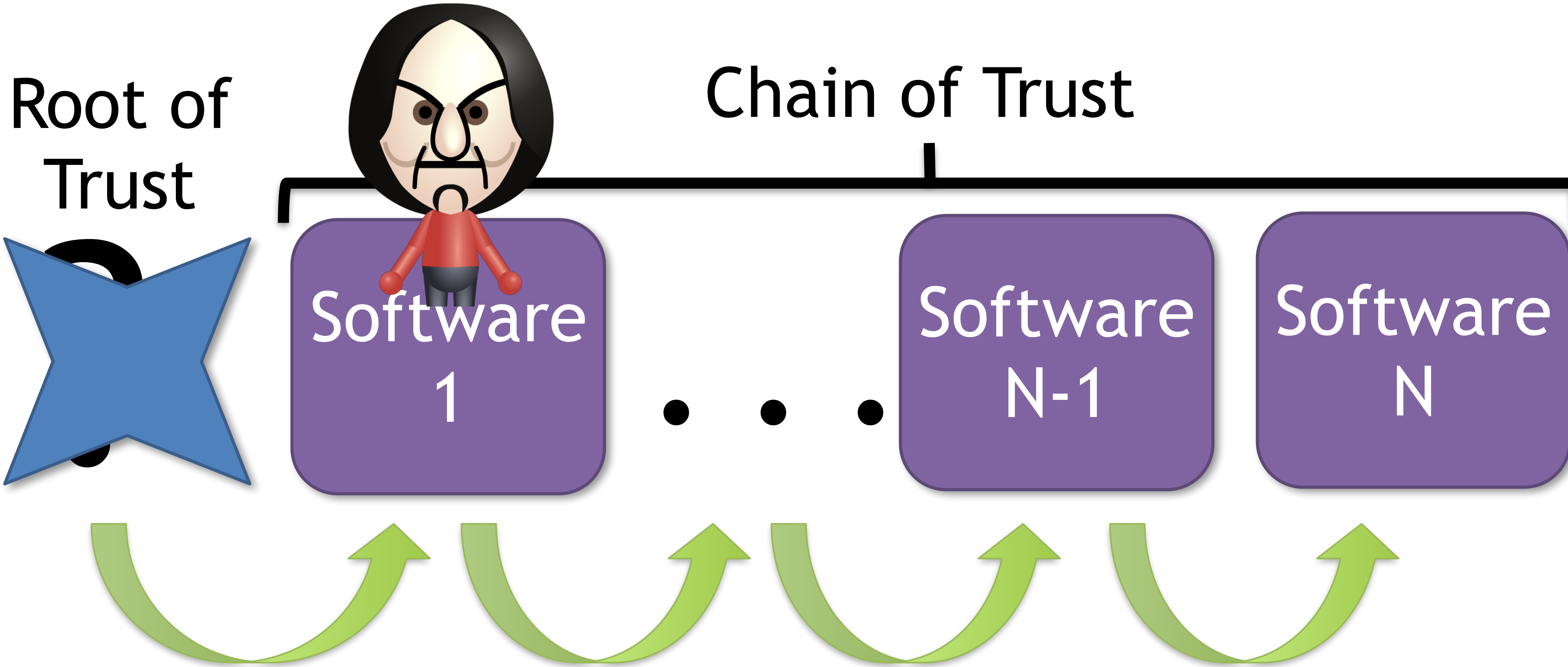
Establishing Code Identity



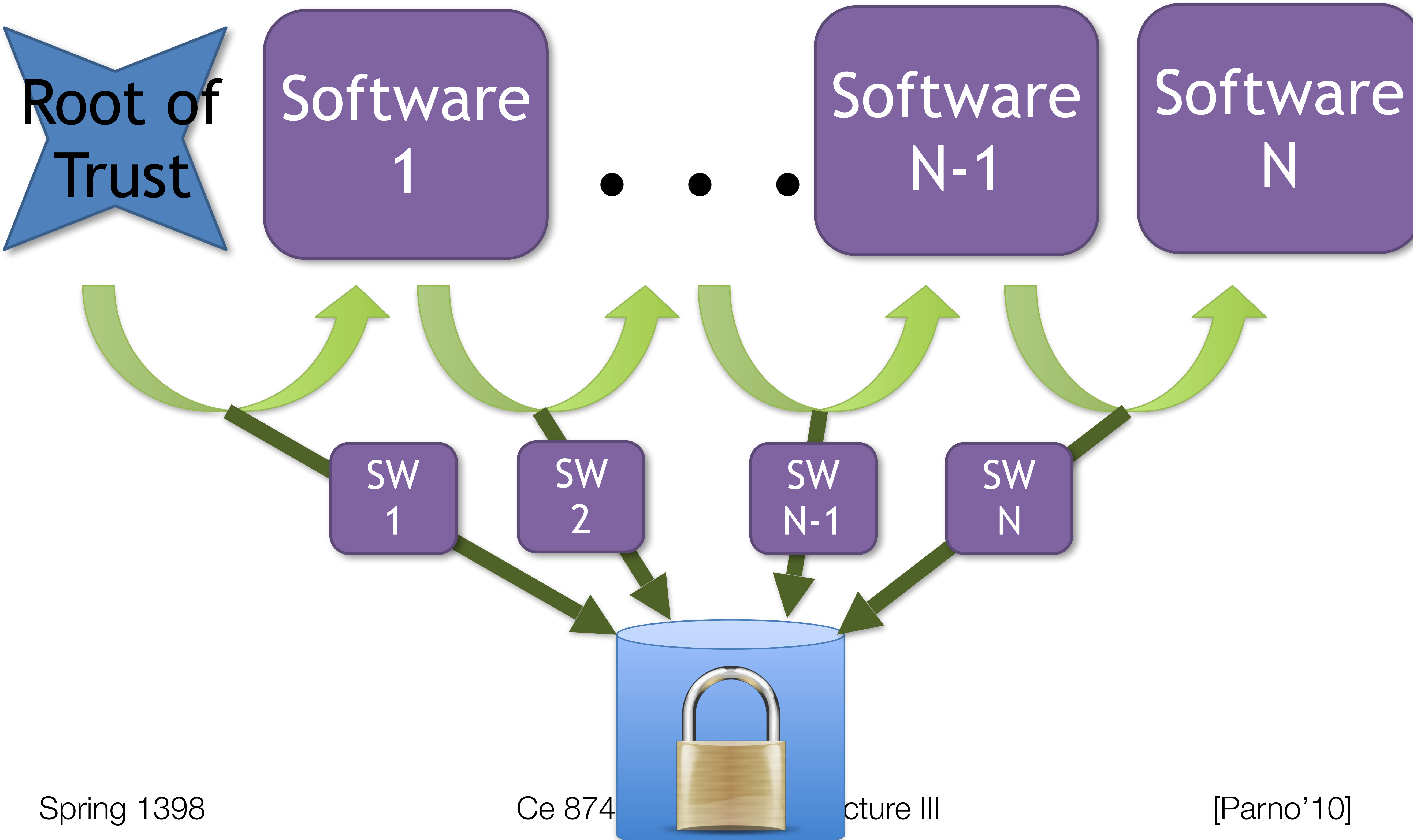
Establishing Code Identity



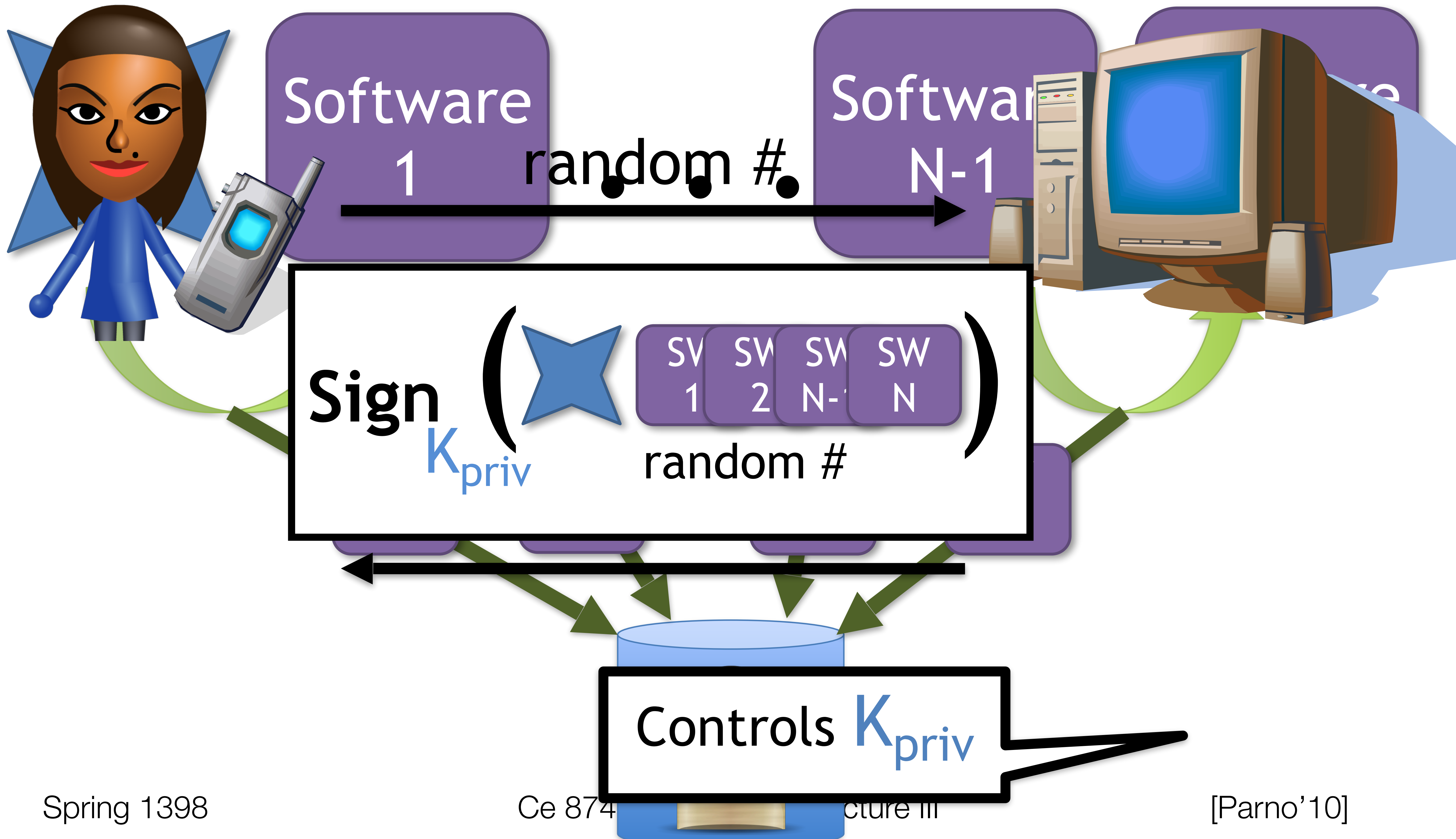
Establishing Code Identity



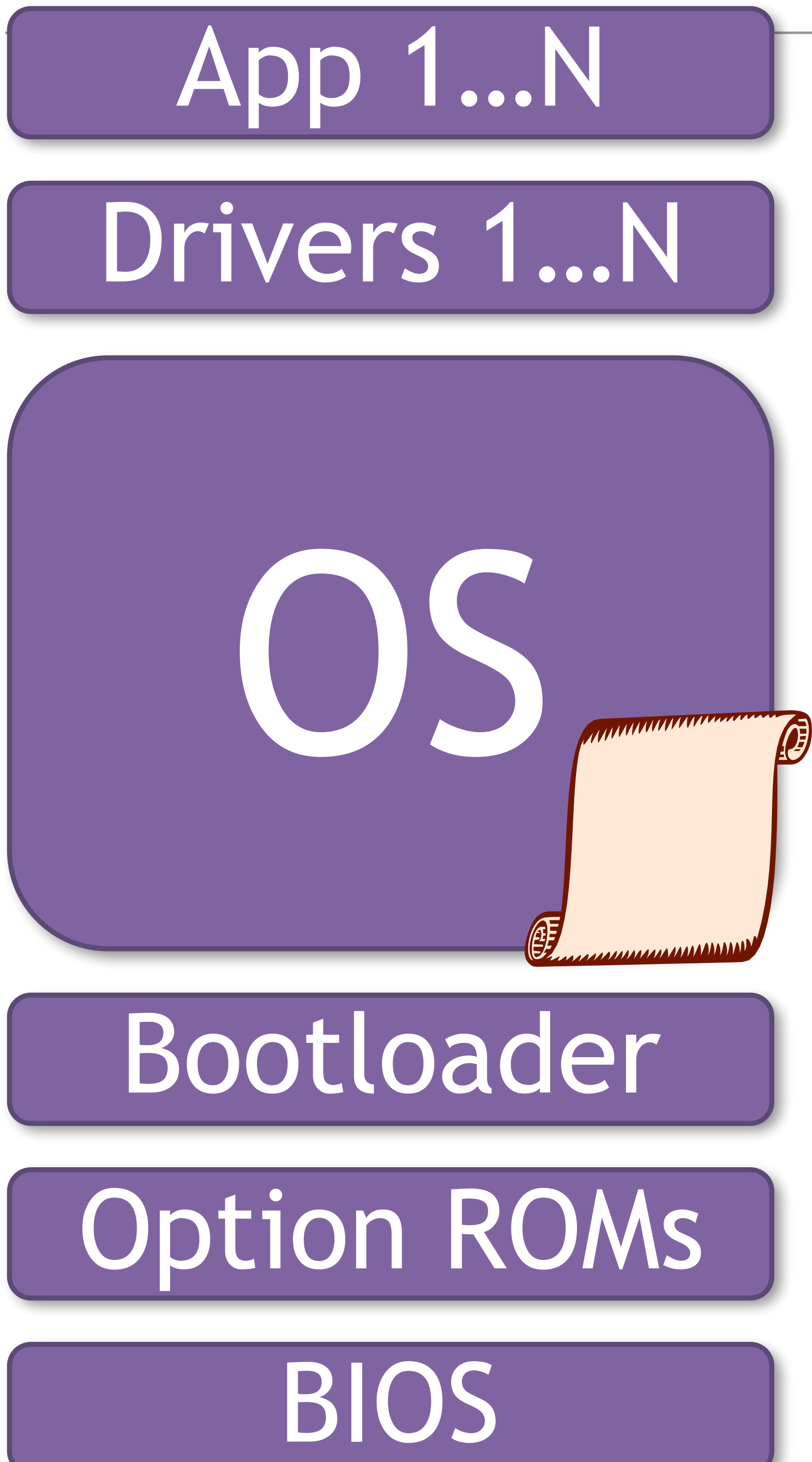
Trusted Boot: Recording Code Identity



Attestation:
Conveying Records to an External Entity



Interpreting Code Identity



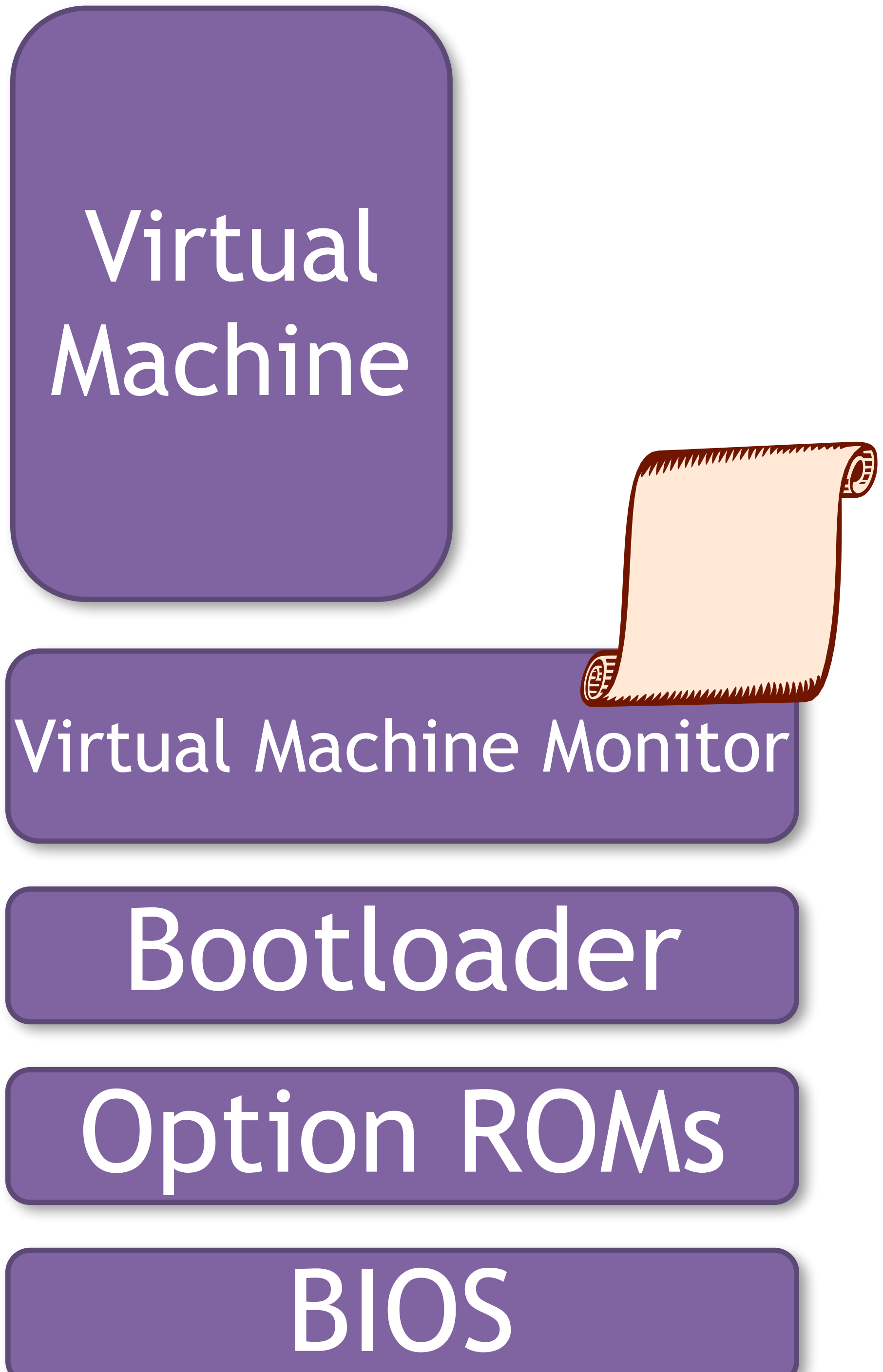
Traditional

[Gasser et al. ‘89], [Sailer et al. ‘04]

Policy Enforcement

[Marchesini et al. ‘04], [Jaeger et al. ’06]

Interpreting Code Identity



Traditional

[Gasser et al. ‘89], [Sailer et al. ‘04]

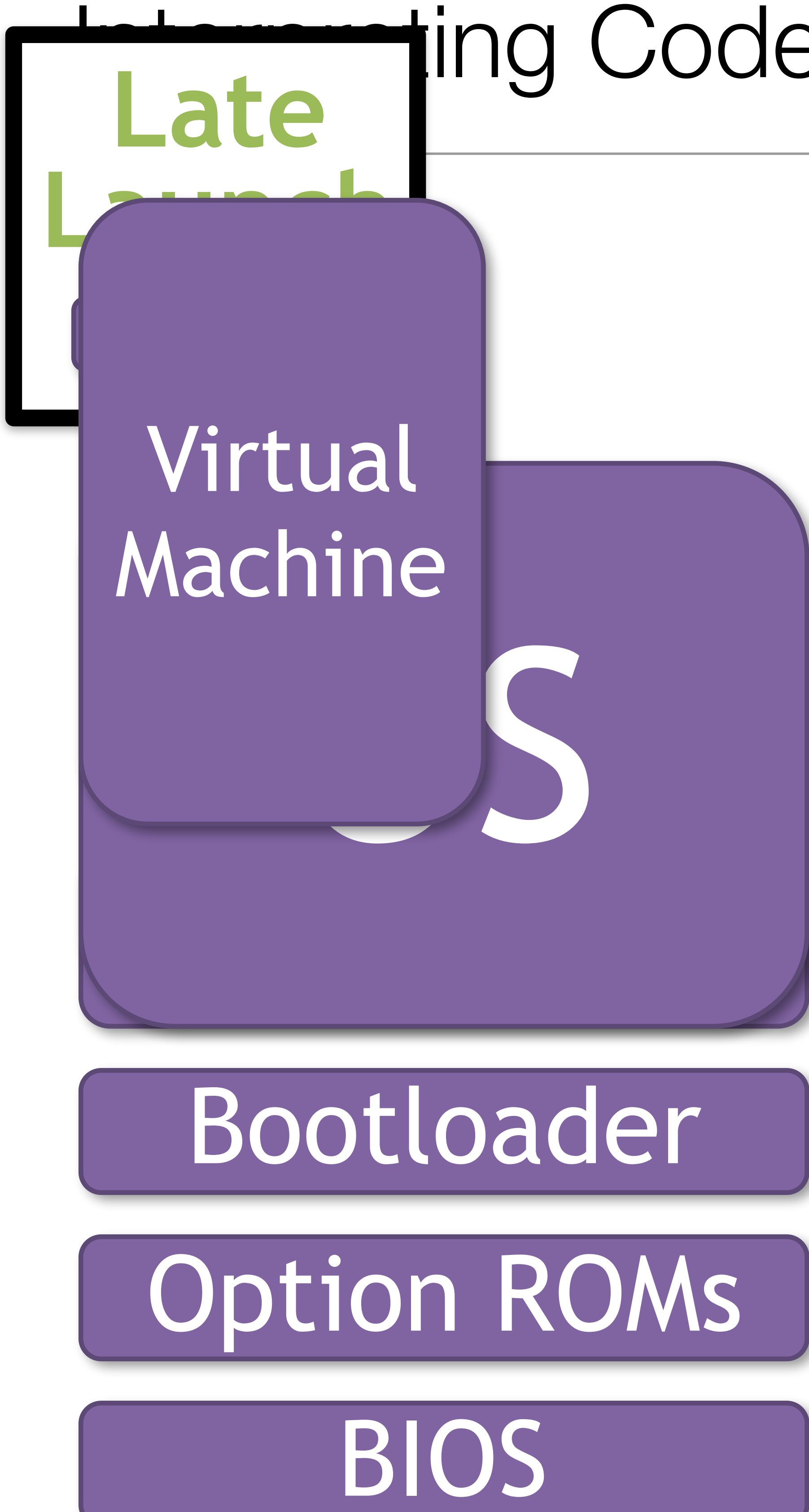
Policy Enforcement

[Marchesini et al. ‘04], [Jaeger et al. ’06]

Virtualization

[England et al. ‘03], [Garfinkel et al. ‘03]

Interpreting Code Identity



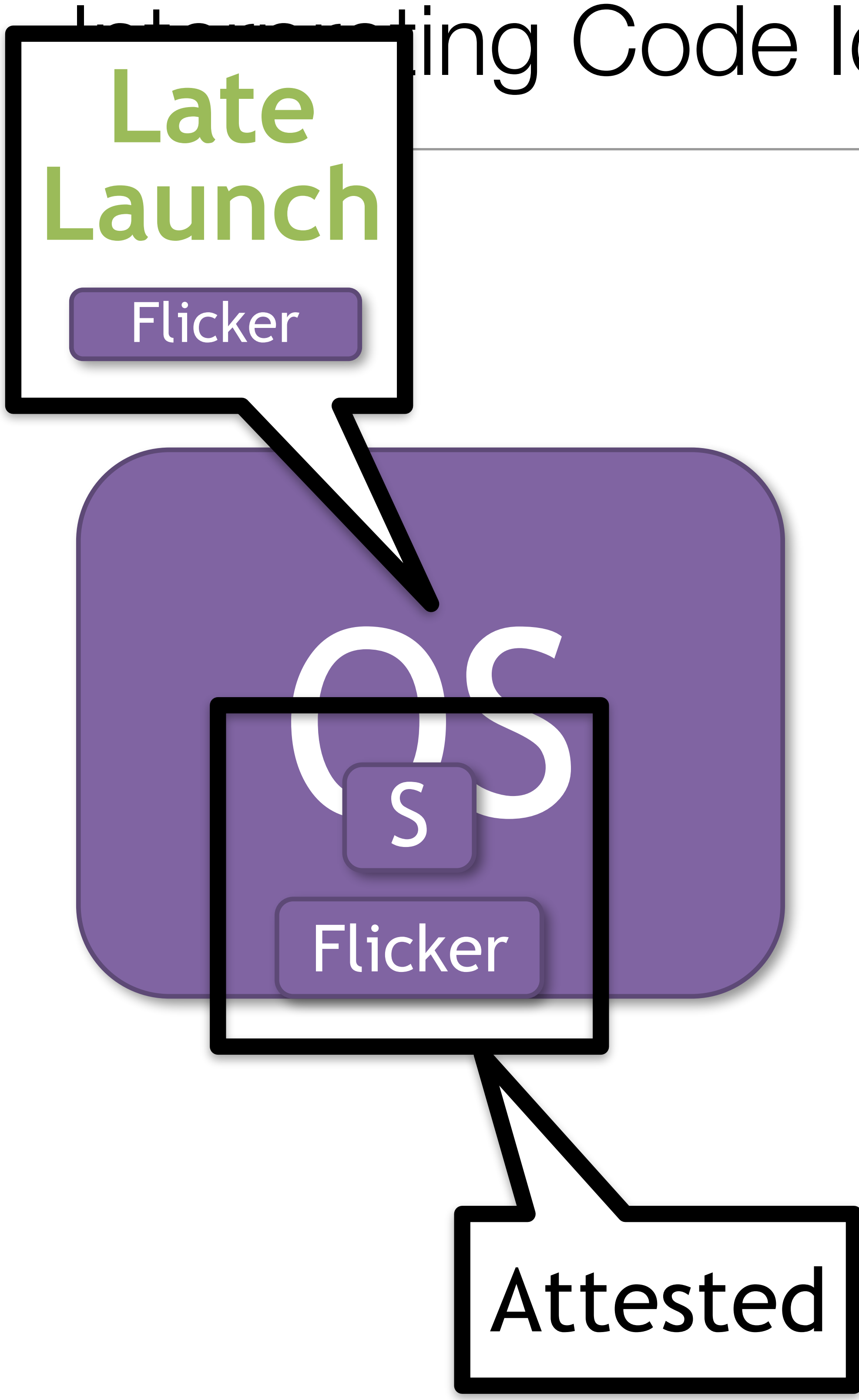
Traditional
[Gasser et al. ‘89], [Sailer et al. ‘04]

Policy Enforcement
[Marchesini et al. ‘04], [Jaeger et al. ’06]

Virtualization
[England et al. ‘03], [Garfinkel et al. ‘03]

Late Launch
[Kauer et al. ‘07], [Grawrock ‘08]

Interpreting Code Identity



Traditional
[Gasser et al. ‘89], [Sailer et al. ‘04]

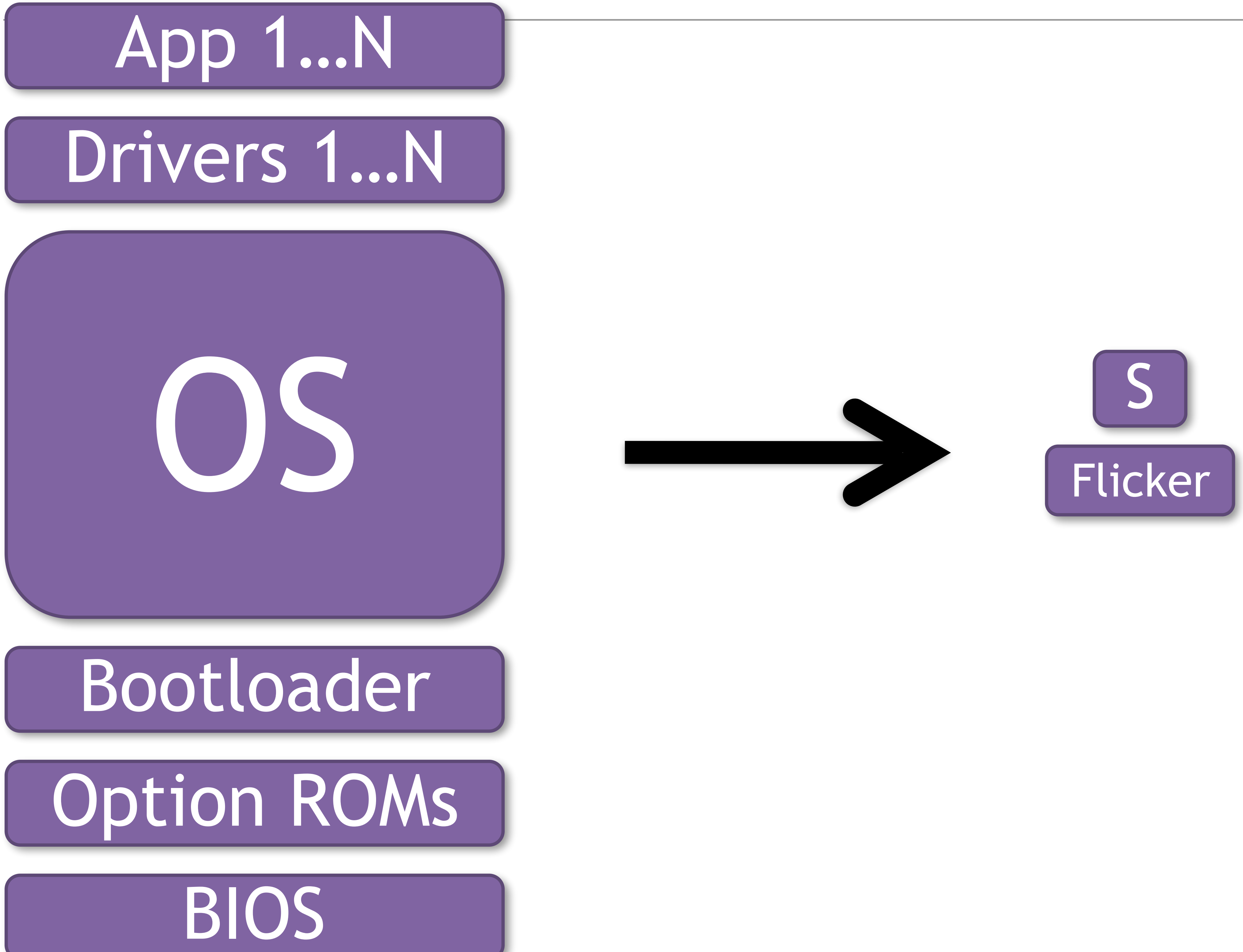
Policy Enforcement
[Marchesini et al. ‘04], [Jaeger et al. ’06]

Virtualization
[England et al. ‘03], [Garfinkel et al. ‘03]

Late Launch
[Kauer et al. ‘07], [Grawrock ‘08]

Targeted Late Launch
[McCune et al. ‘07]

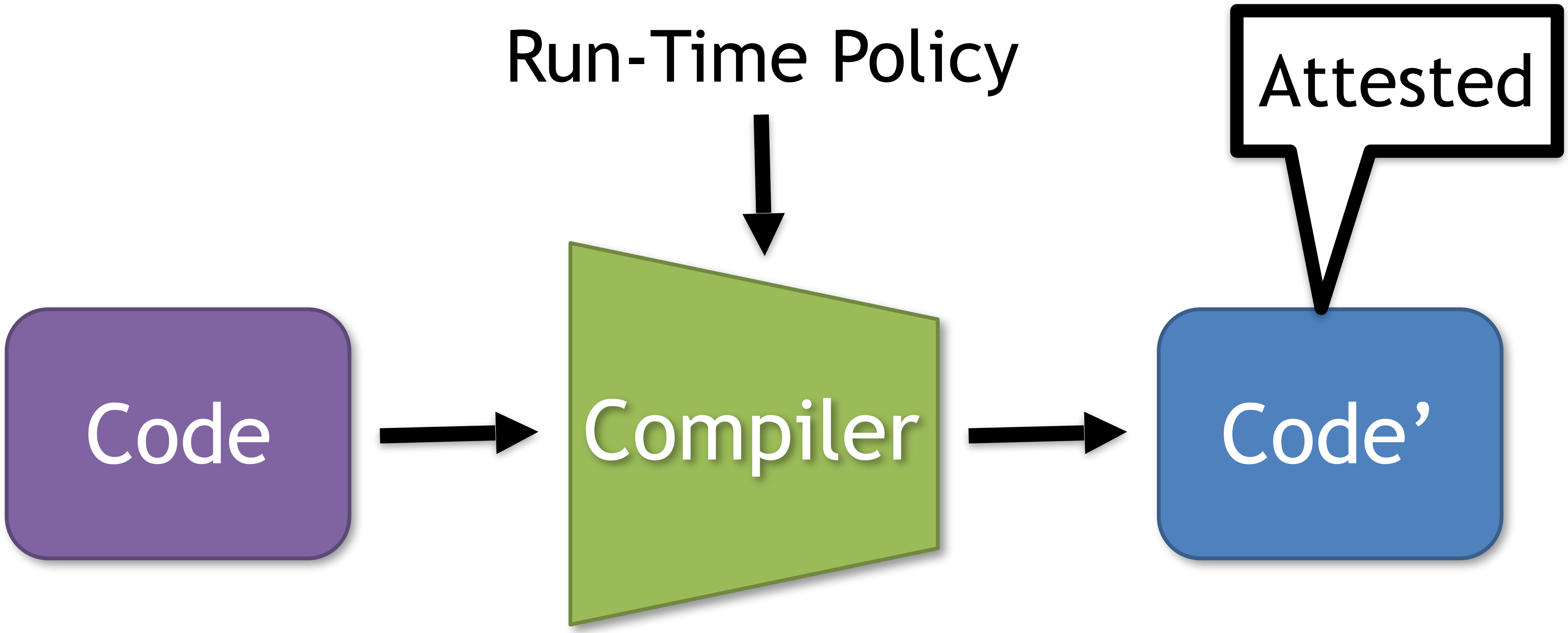
Interpreting Code Identity



Load-Time vs. Run-Time Properties

- Code identity provides load-time guarantees
- What about run time?
- Approach #1: Static transformation

[Erlingsson et al. ‘06]

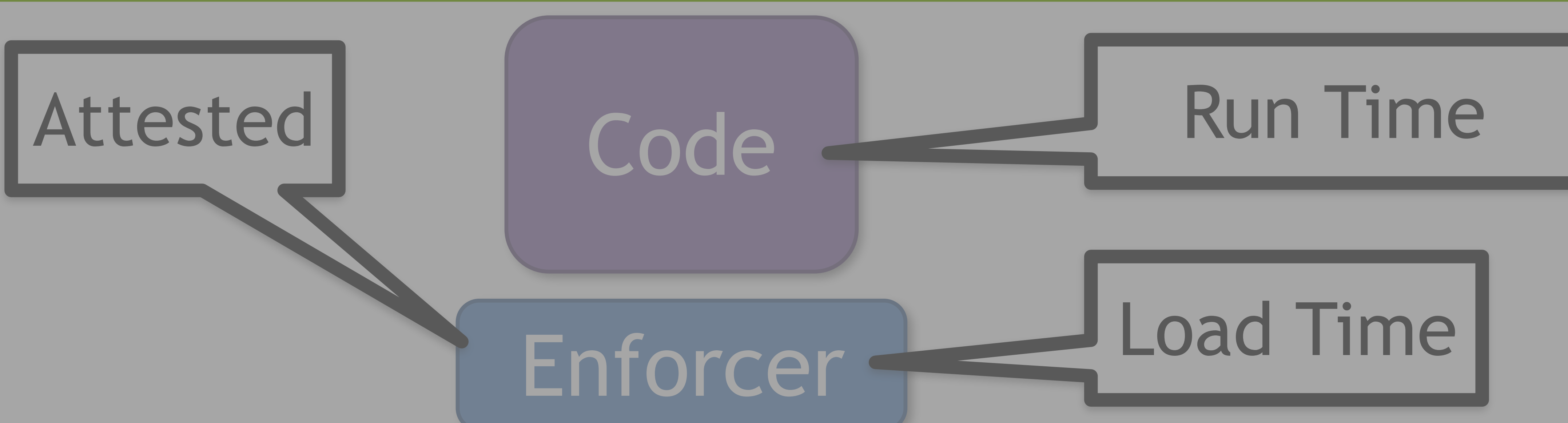


Load-Time vs Run-Time Properties

- Code identity provides load-time guarantees
- What about run time?

<u>Open Question</u>:

How can we get complete run-time properties?



Roots of Trust

Cheaper

Open Question:

What functionality do we need in hardware?

responding

defenses

detailed HW knowledge

[Weingart '87]
[White et al. '91]
[Yee '94]
[Smith et al. '99]
...

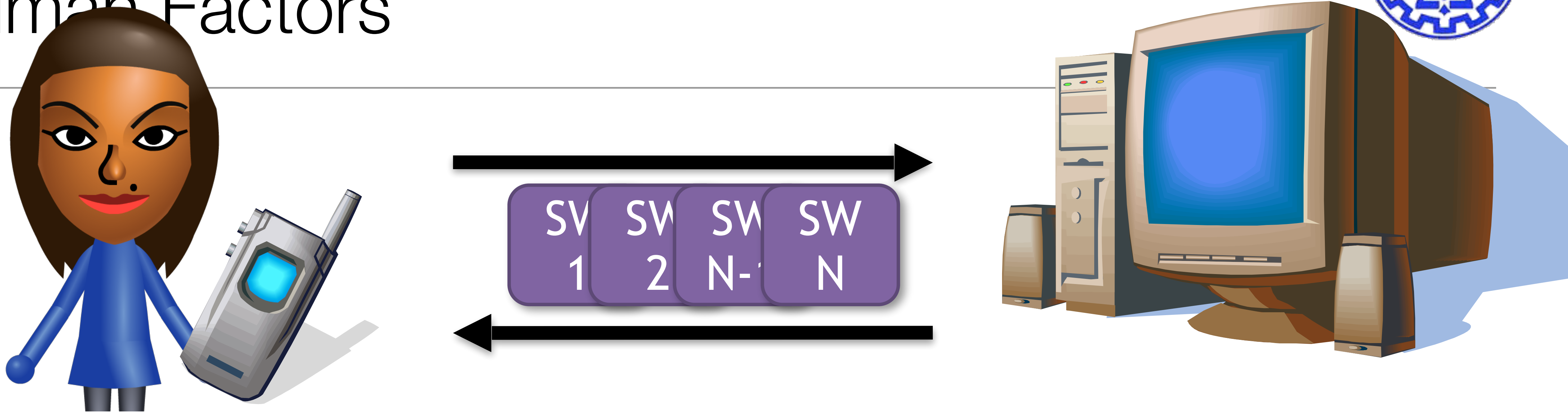
[ARM TrustZone '04]
[TCG '04]
[Zhuang et al. '04]
...

[Chun et al. '07]
[Levin et al. '09]

[Spinellis et al. '00]
[Seshadri et al. '05]
...

[Parno'10]

Human Factors



Open Questions:

Conclusions

- **Code identity** is critical to bootstrapping trust
- Assorted hardware **roots of trust** available
- Many **open questions** remain!

A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping, Seunghun Han, Wook Shin, Jun-Hyeok Park, and HyoungChun Kim, Usenix Security 2018

Trusted Computing Group (TCG)

- Defines global industry specifications and standards
- Is supportive of a hardware root of trust
  - Trusted Platform Module (TPM) is the core technology
  - TCG technology has been applied to Unified Extensible Firmware Interface (UEFI)

Trusted Platform Module (TPM) (1)

- Is a tamper-resistant device
- Has own processor, RAM, ROM, and non-volatile RAM
    - It has own state separated from the system
- Provides cryptographic and accumulating measurements functions
    - Measurement values are accumulated to Platform Configuration Registers (PCR #0~#23)

Trusted Platform Module (TPM) (2)

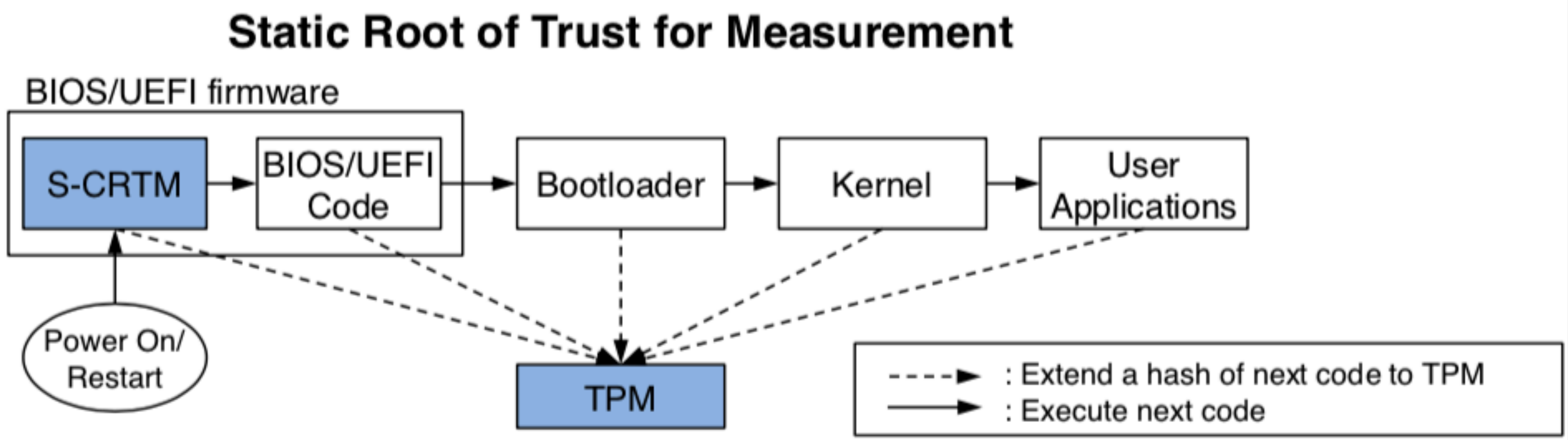
- Is used to determine the trustworthiness of a system by investigating the values stored in PCRs
  - A local verification or remote attestation can be used
- Is used to limit access to secret data based on specific PCR values
  - “Seal” operation encrypts secret data with the PCRs of the TPM
  - “Unseal” operation can decrypt the sealed data only if the PCR values match the specific values

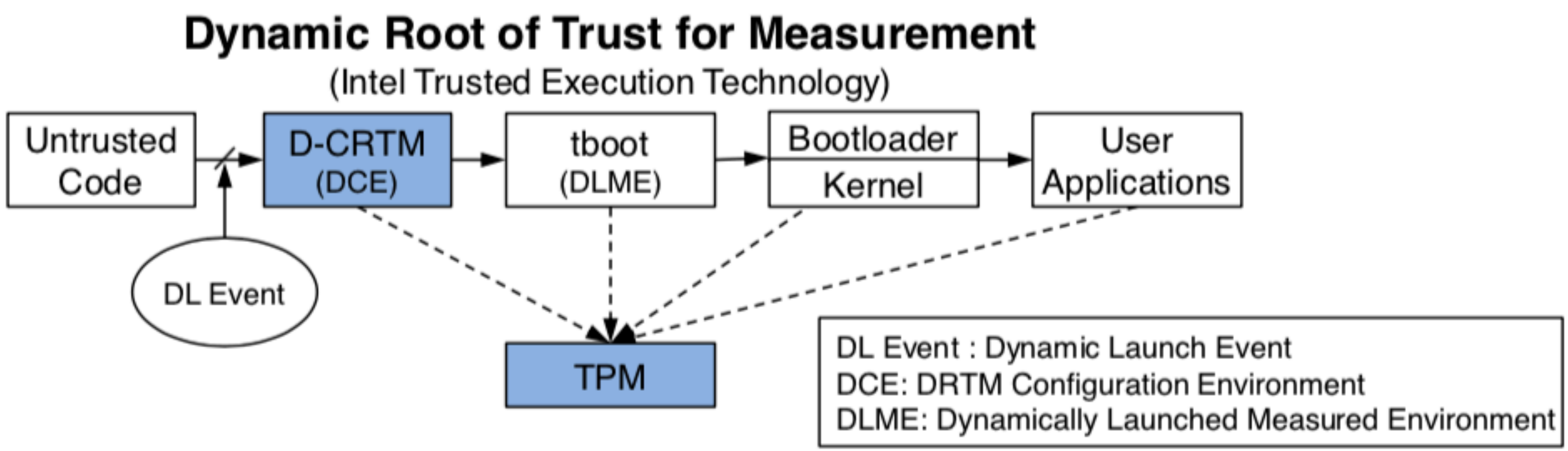
Root of Trust for Measurement (RTM)

- Sends integrity-relevant information (measurements) to the TPM
  - TPM accumulates the measurements to a PCR with the previously stored value in the PCR
    - **Extend: PCRnew = Hash(PCRold || Measurementnew)**
- The CPU controlled by Core RTM (CRTM)
  - The CRTM is the first set of instructions when a new chain of trust is established

Static and Dynamic RTM (SRTM and DRTM)

- SRTM is started by static CRTM (S-CRTM) when the host platform starts at **POWER-ON** or **RESTART**
- DRTM is started by dynamic CRTM (D-CRTM) at runtime **WITHOUT** platform **RESET**
- They extend measurements (hashes) of components to PCRs **BEFORE** passing control to them


Static Root of Trust for Measurement
BIOS/UEFI firmware
S-CRTM
BIOS/UEFI Code
Bootloader
Kernel
User Applications
Power On/ Restart
TPM
: Extend a hash of next code to TPM
: Execute next code


Dynamic Root of Trust for Measurement
(Intel Trusted Execution Technology)
Untrusted Code
D-CRTM (DCE)
tboot (DLME)
Bootloader
Kernel
User Applications
DL Event
TPM
DL Event : Dynamic Launch Event
DCE: DRTM Configuration Environment
DLME: Dynamically Launched Measured Environment

PCR Protection

- PCRs contains measurement results of a system
- They **MUST NOT** be reset by disallowed operations
  - Static PCRs (PCR #0~#15) can be reset only if the host resets
  - Dynamic PCRs (PCR #17~#19) can be reset only if the host initializes the DRTM
- If PCRs are reset by attackers, they can reproduce specific PCR values by replaying hashes
  - They can steal the secret and deceive the local and remote
- verification

PCR protection mechanisms work properly

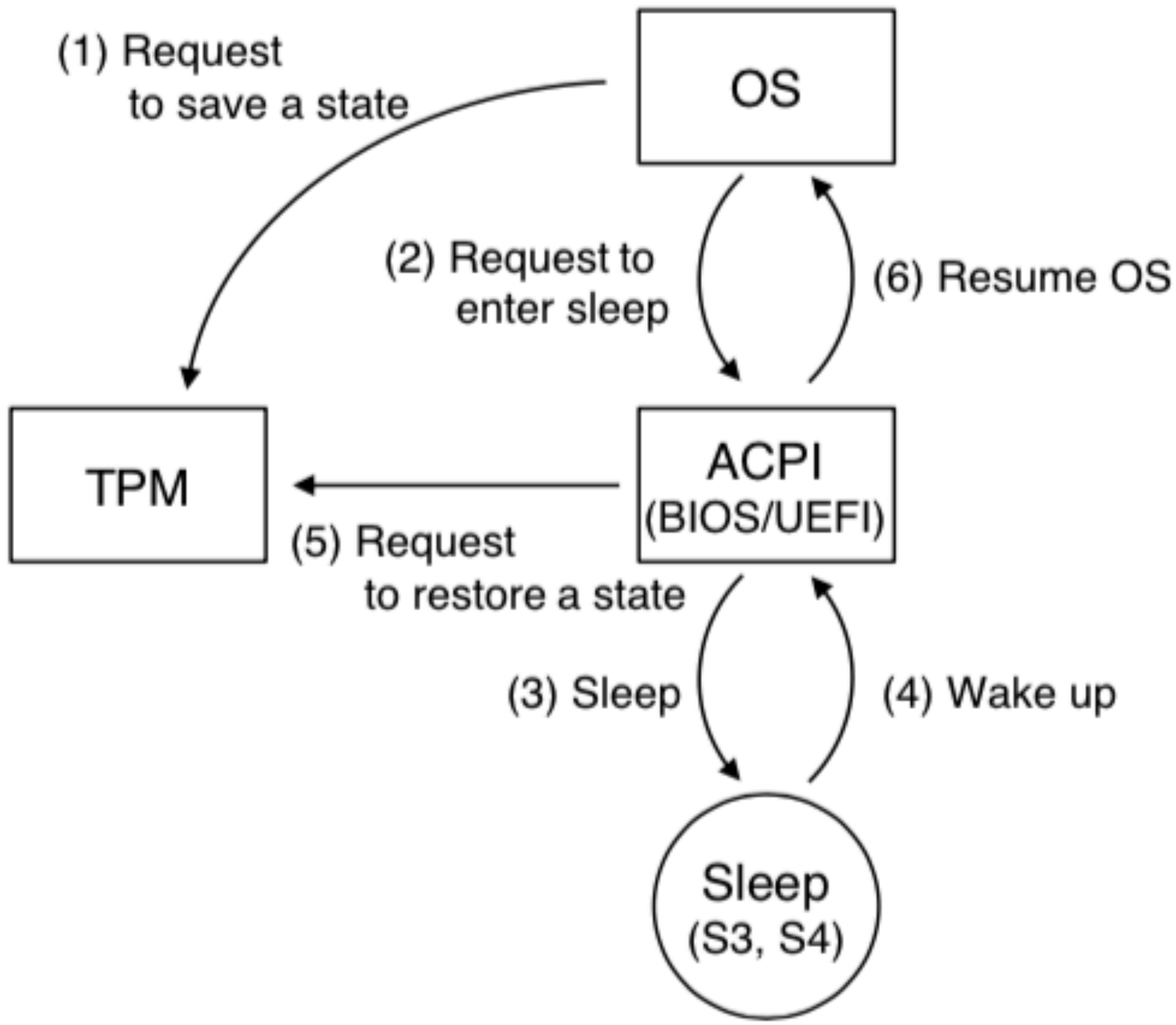
UNTIL YESTERDAY

Assumptions and Threat Model

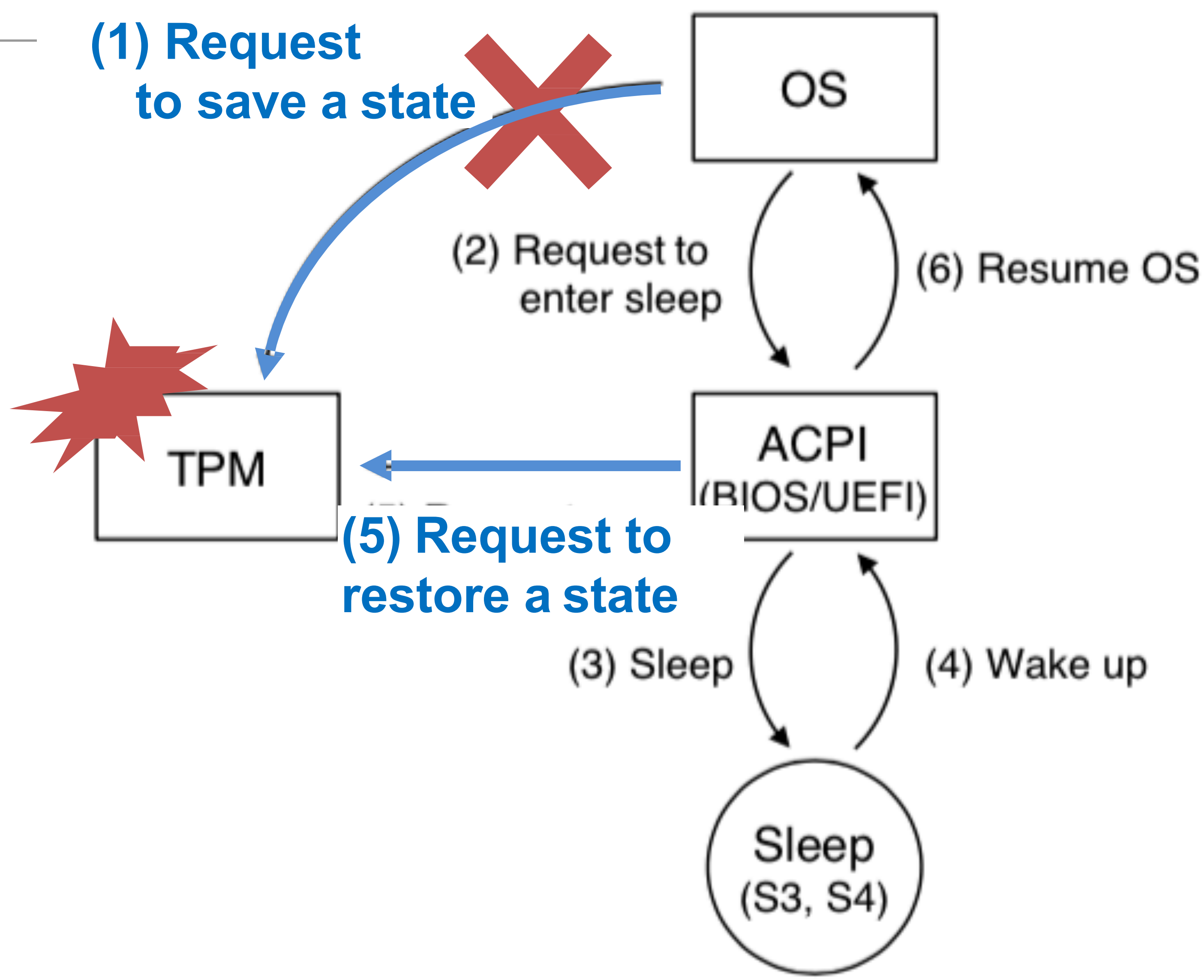
- The system measures boot components using the SRTM and DRTM
  - The measurement results stored in PCRs are verified by a remote verifier
  - The modifications of boot components are detected
- The attackers already gain a root privilege and try to compromise the whole system
  - They try to hide the breach and retain the root privilege
  - They cannot access the system circuit physically
  - They cannot flash the firmware with arbitrary code

Advanced Configuration and Power Interface (ACPI)

- Defines power states and hardware register sets
  - Global states
    - G0 (Working), G1 (Sleeping), G2 (Soft-off), G3 (Mechanical-off)
  - Sleeping states
    - S0 and S1: Working and Power on Suspend
    - S2: Same as S1, CPU is powered off
    - S3: Sleep, **All devices are powered off except RAM**
    - S4: Hibernation, All devices are powered off



ACPI Sleep Process with TPM



The Grey Area vulnerability (CVE-2018-6622)

The Grey Area Vulnerability (CVE-2018-6622)

What is the "corrective action"?

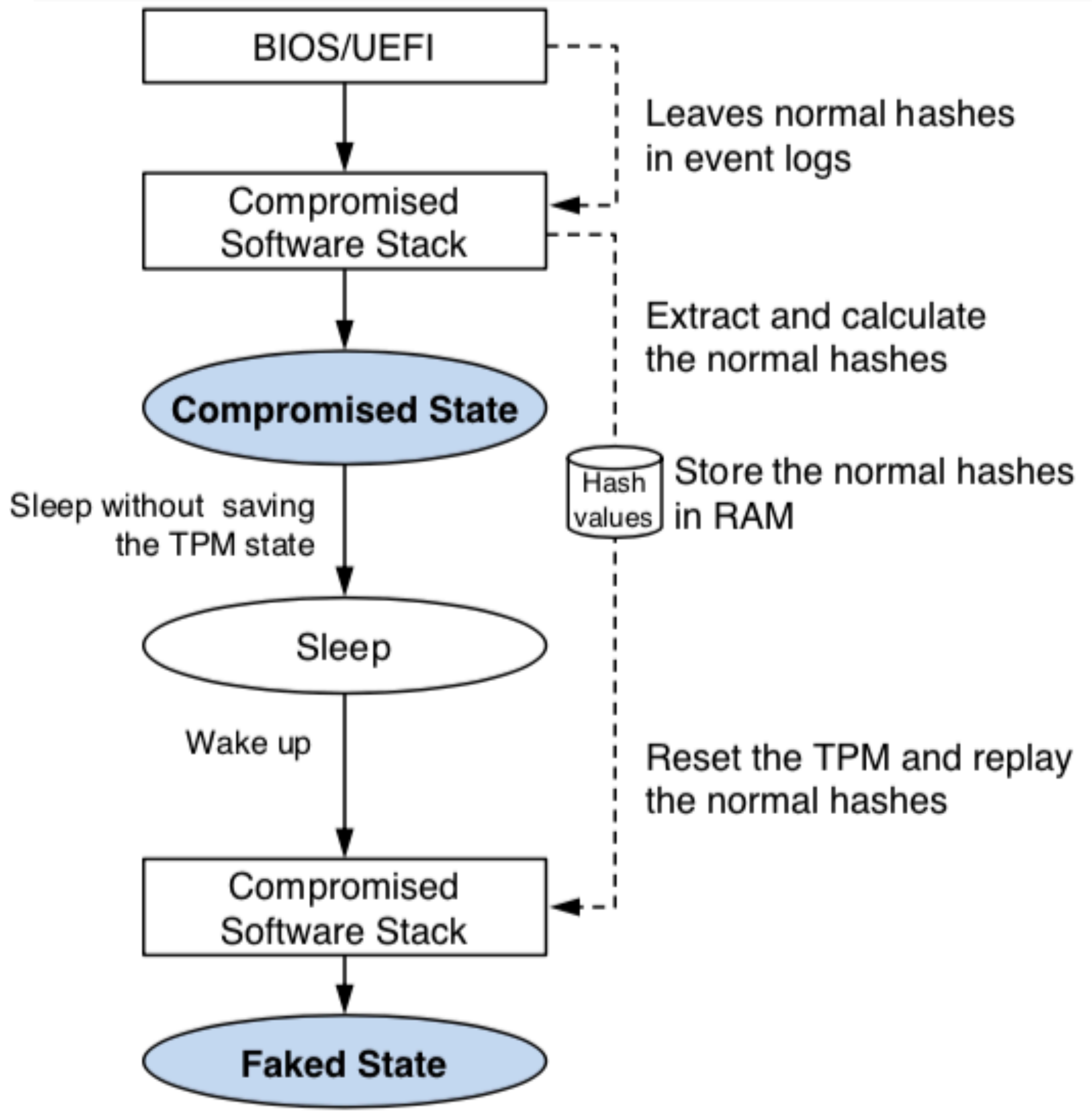
If the TPM receives Startup(STATE) that was not preceded by Shutdown(STATE), then there is no state to restore and the TPM will return TPM_RC_VALUE. The CRTM is expected to take corrective action to prevent malicious software from manipulating the PCR values such that they would misrepresent the state of the platform. The CRTM would abort the Startup(State) and restart with Startup(CLEAR).

This means "reset the TPM"

The startup behavior defined by this specification is different than TPM 1.2 with respect to Startup(STATE). A TPM 1.2 device will enter Failure Mode if no state is available when the TPM receives Startup(STATE). This is not the case in this specification. It is up to the CRTM to take corrective action if it the TPM returns TPM_RC_VALUE in response to Startup(STATE).

Trusted Platform Module Library Part1: Architecture

Exploit of the Grey Area Vulnerability



Evaluation – The Grey Area Vulnerability

PC No.	Vendor	CPU (Intel)	PC and mainboard model	BIOS Ver. and release date	TPM Ver.	TPM vendor and firmware Ver.	SRTM attack
1	Intel	Core i5-5300U	NUC5i5MYHE	MYBDEWi5v.86A, 2017.11.30	2.0	Infineon, 5.40	Y
2	Intel	Core m5-6Y57	Compute Stick STK2mv64CC	CCSKLm5v.86A.0054, 2017.12.26	2.0	NTC, 1.3.0.1	Y
3	Dell	Core i5-6500T	Optiplex 7040	1.8.1, 2018.01.09	2.0	NTC, 1.3.2.8	Y
4	GIGABYTE	Core i7-6700	Q170M-MK	F23c, 2018.01.11	2.0	Infineon, 5.51	Y
5	GIGABYTE	Core i7-6700	H170-D3HP	F20e, 2018.01.10	2.0	Infineon, 5.61	Y
6	ASUS	Core i7-6700	Q170M-C	3601, 2017.12.12	2.0	Infineon, 5.51	Y
7	Lenovo	Core i7-6600U	X1 Carbon 4th Generation	N1FET59W (1.33), 2017.12.19	1.2	Infineon, 6.40	N
8	Lenovo	Core i5-4570T	ThinkCentre m93p	FBKTCPA, 2017.12.29	1.2	STMicroelectronics, 13.12	N
9	Dell	Core i5-6500T	Optiplex 7040	1.8.1, 2018.01.09	1.2	NTC, 5.81.2.1	N
10	HP	Xeon E5-2690 v4	z840	M60 v02.38, 2017.11.08	1.2	Infineon, 4.43	N
11	GIGABYTE	Core i7-6700	H170-D3HP	F20e, 2018.01.10	1.2	Infineon, 3.19	N

Countermeasures – The Grey Area Vulnerability

- **Disable S3 sleeping state** option in BIOS menu
  - Brutal, but simple and effective
- Revise TPM 2.0 specification to **enter failure mode** if there is no state to restore
- Revise TPM 2.0 specification to **define "corrective action" in detail**
  - A long time to revise and apply to the TPM or BIOS/UEFI firmware, but fundamental solutions

Countermeasures – The Lost Pointer Vulnerability

- **Apply our patch to tboot**
  - https://sourceforge.net/p/tboot/code/ci/521c58e51eb5be105a29983742850e72c44ed80e/
- **Update tboot to the latest version**

Conclusion

- Two vulnerabilities that can subvert the TPM using S3 sleeping state were found
  - The Grey Area Vulnerability: CVE-2018-6622
  - The Lost Pointer Vulnerability: CVE-2017-16837
- Attackers can deceive the local and remote verification with the vulnerabilities
  - They also can unseal the seal secret and steal it
- We have contacted manufacturers and contributed a patch to tboot project to solve the vulnerabilities

Acknowledgments/References

- [Parno’10] Bootstrapping Trust in Commodity Computers, Bryan Parno, Jonathan McCune, Adrian Perrig, Slides IEEE S&P, 2010
- [Han’18] A Bad Dream: Subverting Trusted Platform Module While You Are Sleeping, Seunghun Han, Wook Shin, Jun-Hyeok Park, and HyoungChun Kim, Slides, Usenix Security 2018