Strengthening Self-Checksumming via Self-Modifying Code

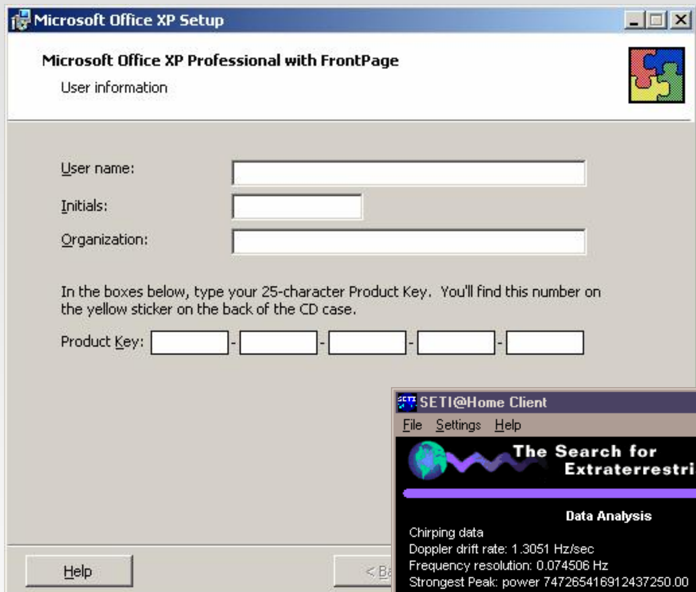
Jonathon T. Giffin, Mihai Christodorescu, Louis Kruger

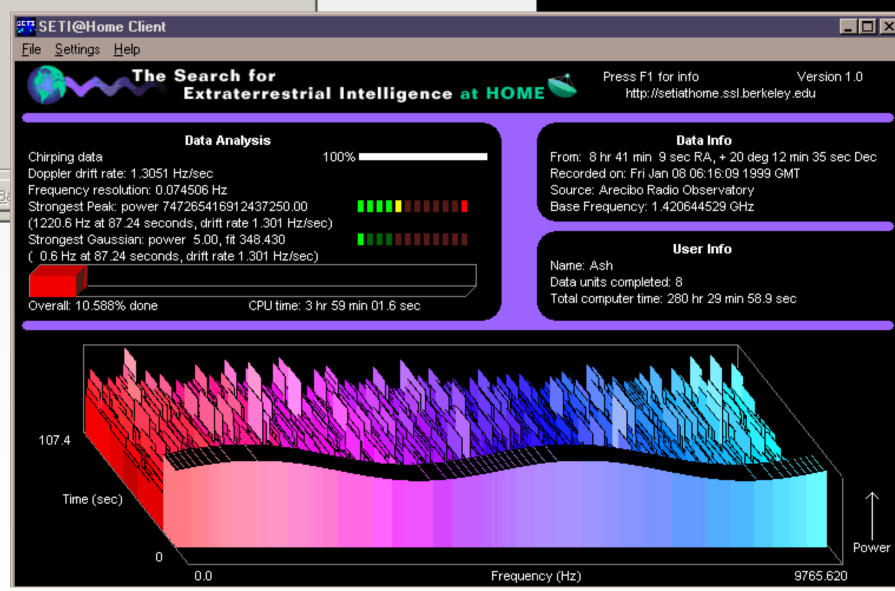
Computer Sciences Department
University of Wisconsin

{giffin,mihai,lpkruger}@cs.wisc.edu

Problem 1

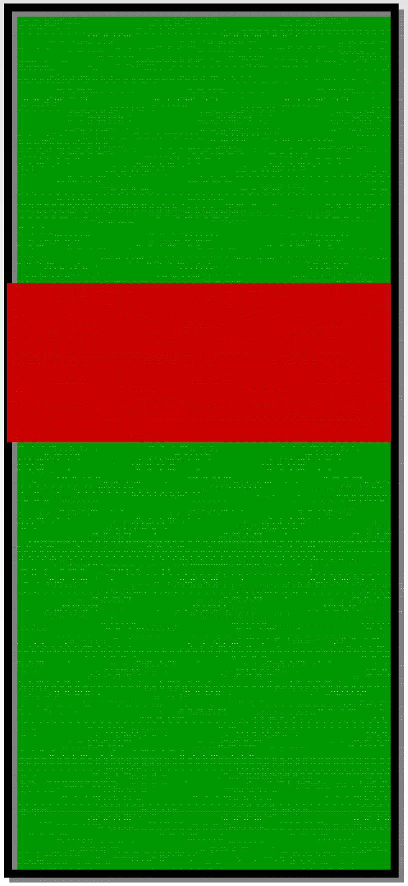
Detect malicious modifications to code





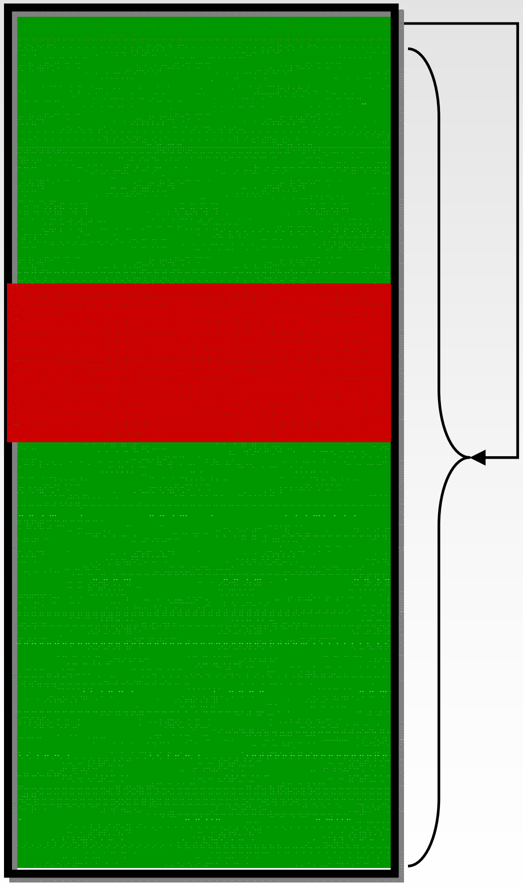
Problem 1

Detect malicious modifications to code



Solution: Self-Checksumming

Program contains code to checksum parts of its own code.

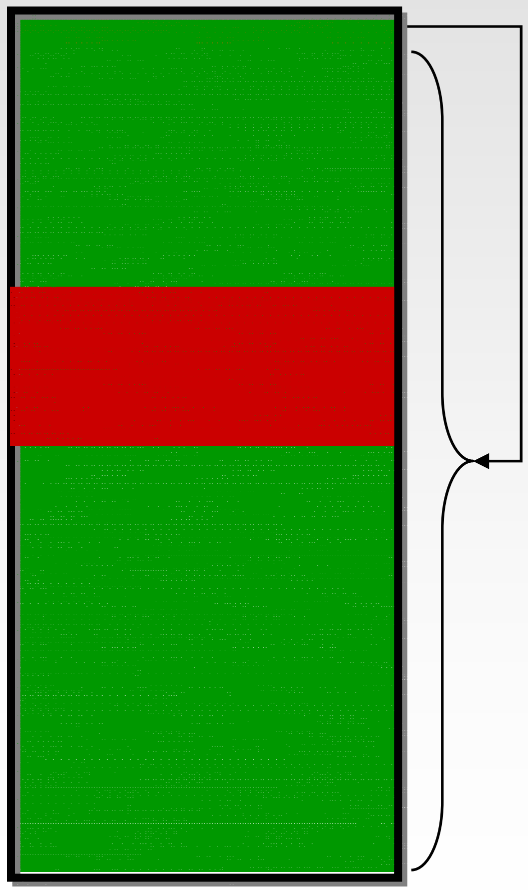


Solution: Self-Checksumming

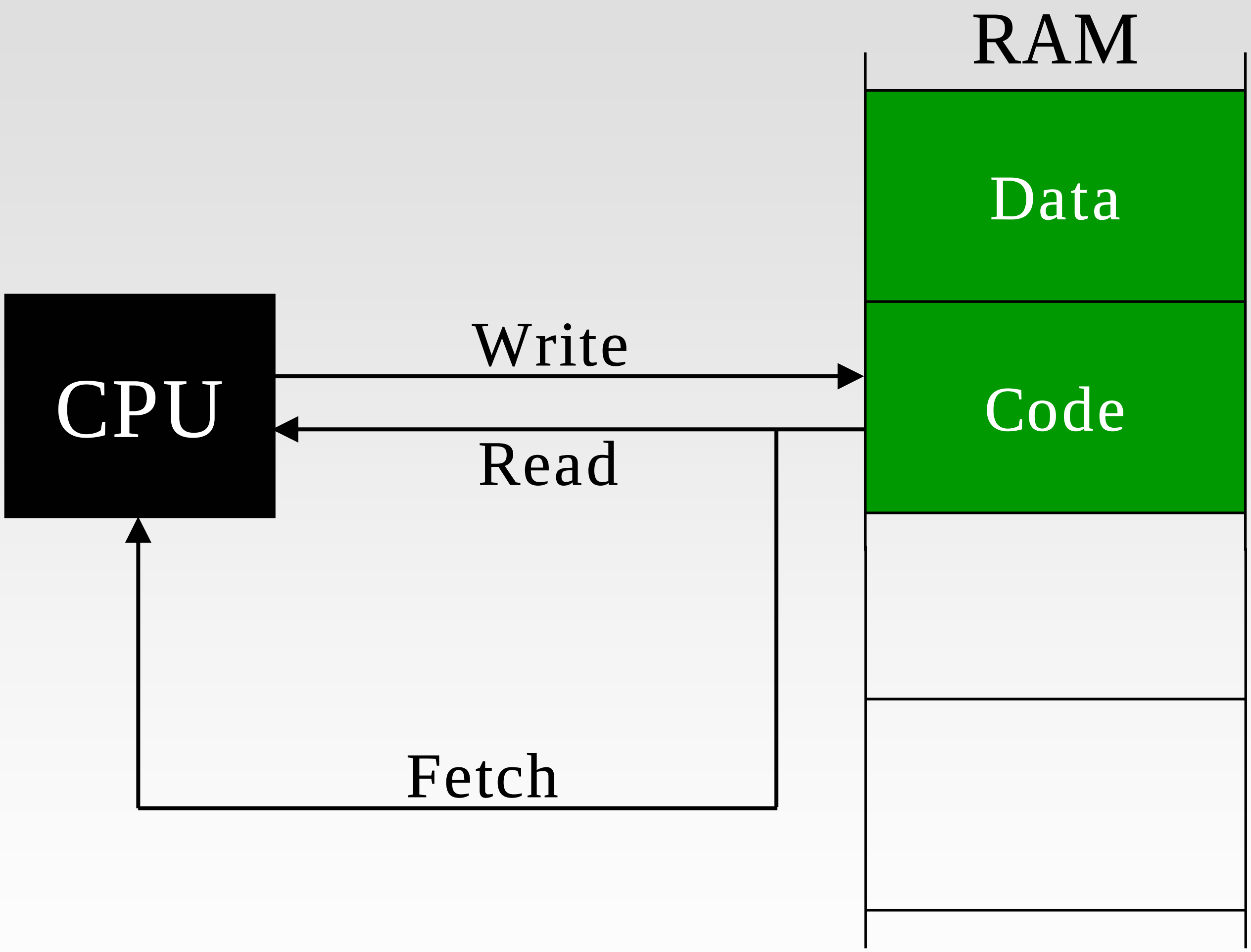
- Network of guards [Chang & Atallah 2001]
  - Many overlapping checksumming components
- Integrity Verification Kernels [Aucsmith 1996]
  - Multithreaded, self-modifying checksumming components
- Testers and correctors [Horne *et al.* 2001]

Problem 2

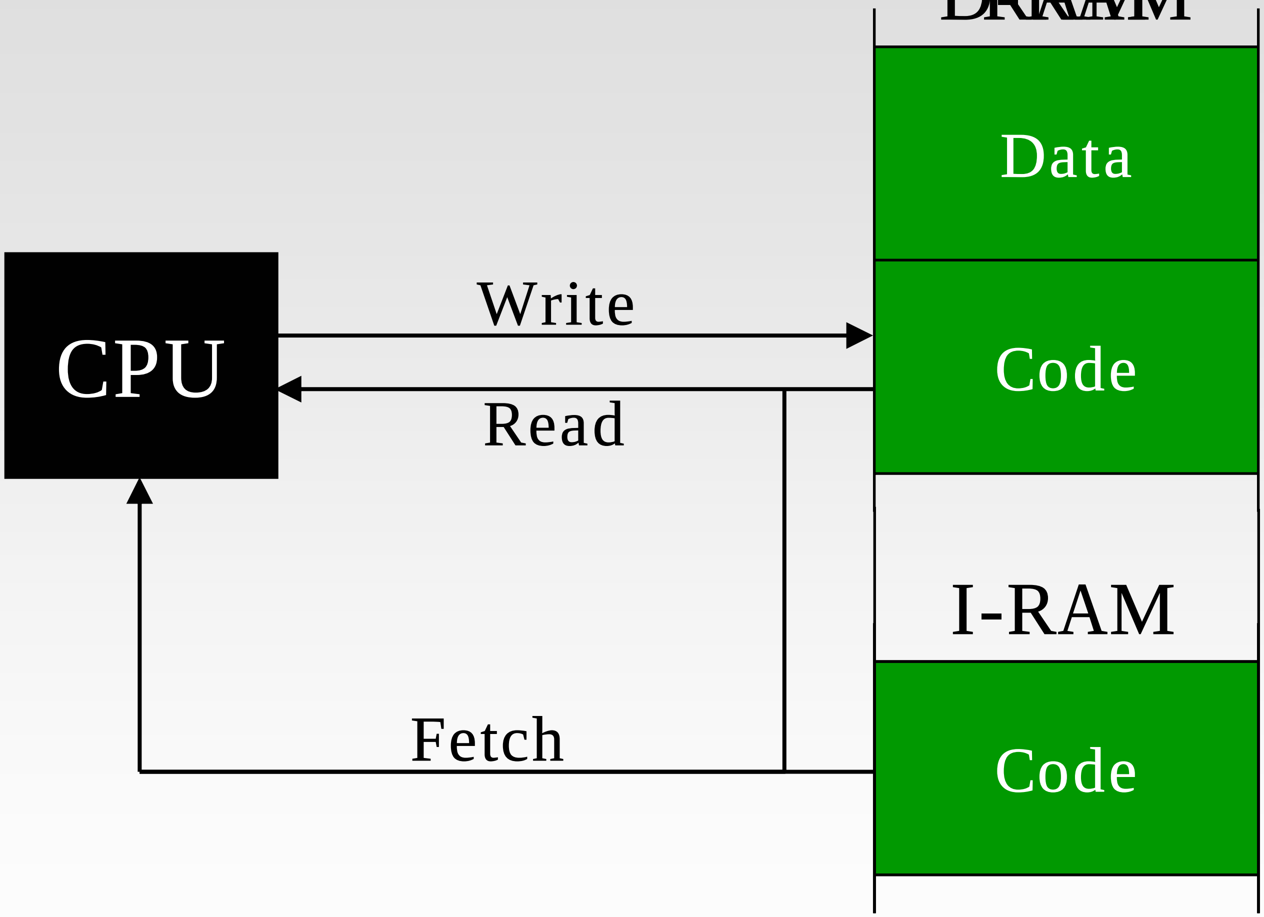
Is the checksummed & validated code actually the code executed?



Normal Memory Accesses



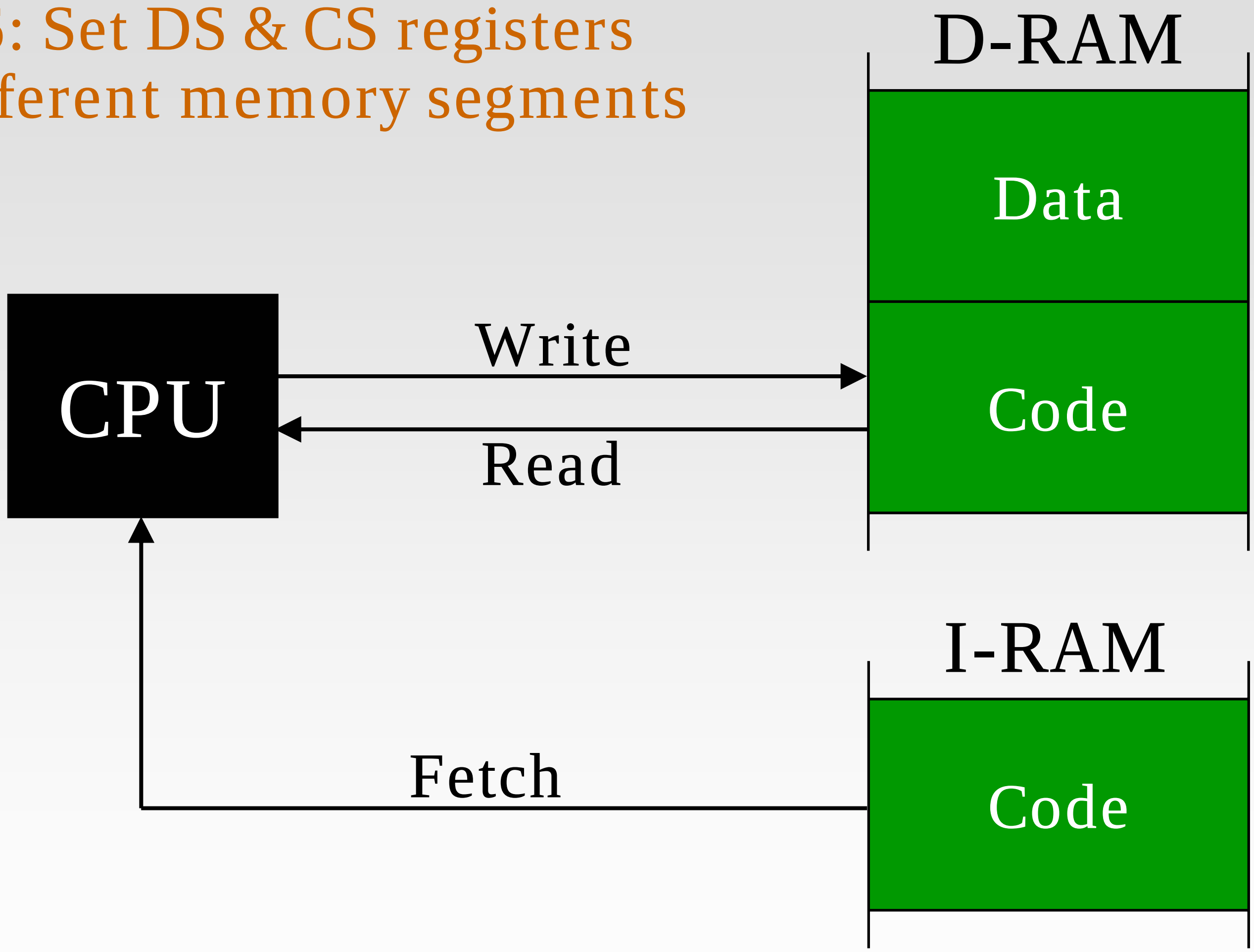
Page-Replication Attack



[Wurster *et al.* 2005]

Page-Replication Attack

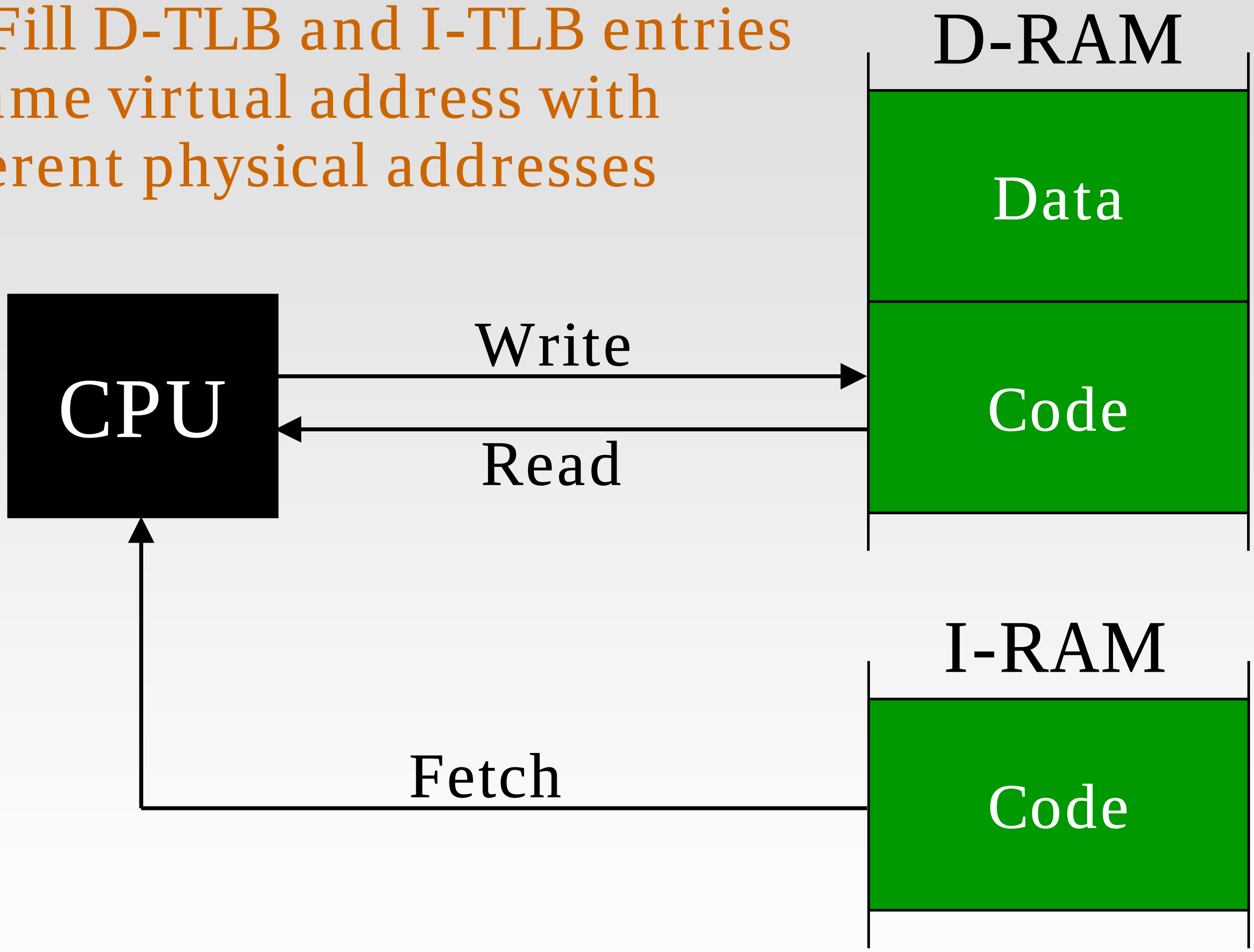
x86: Set DS & CS registers
to different memory segments



[Wurster *et al.* 2005]

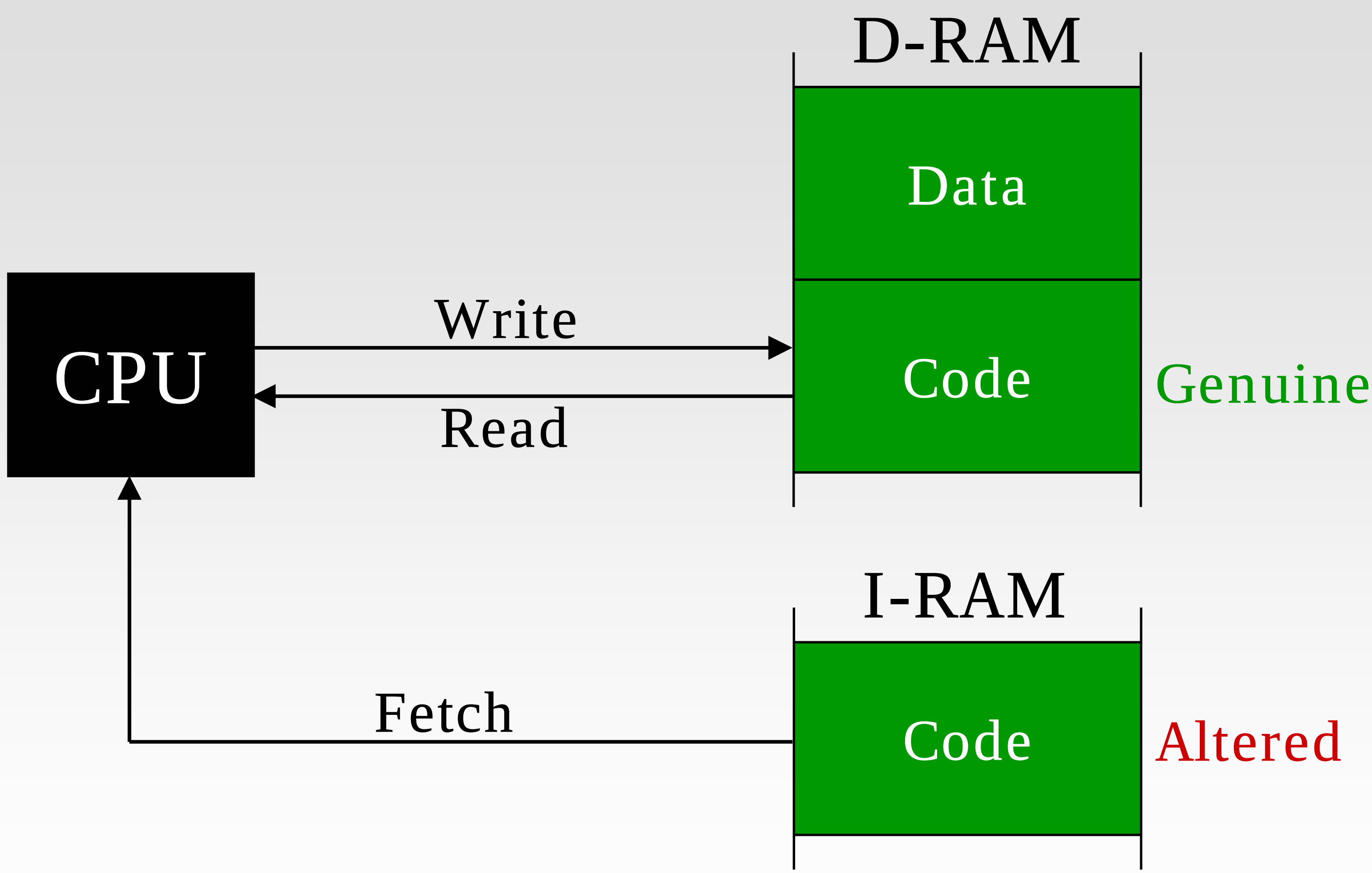
Page-Replication Attack

SPARC: Fill D-TLB and I-TLB entries at same virtual address with different physical addresses



[Wurster *et al.* 2005]

Page-Replication Attack



[Wurster *et al.* 2005]

Solution

Observation:

Writes to code affect program differently when a page-replication attack is underway

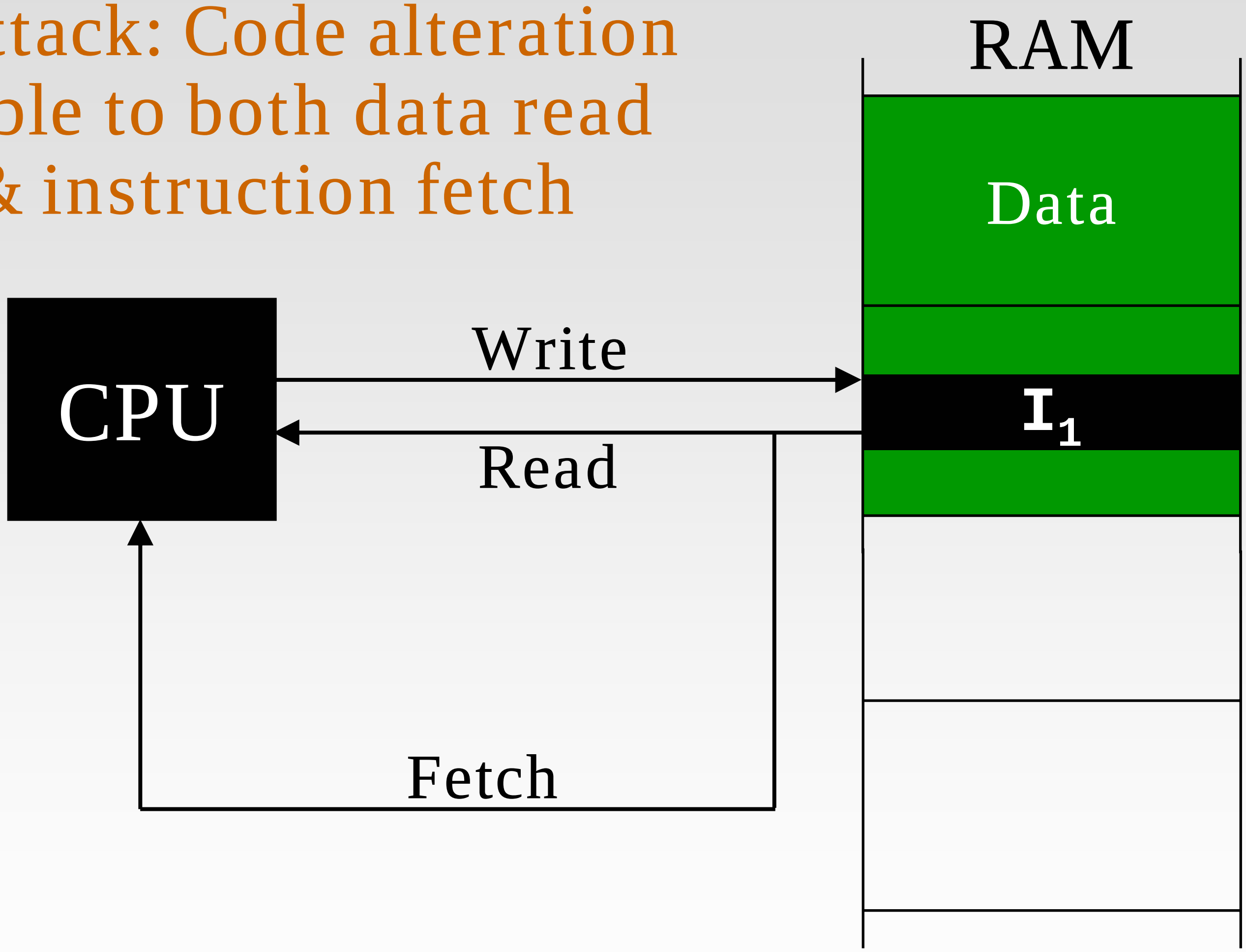
Use self-modifying code to detect page-replication attack

Solution

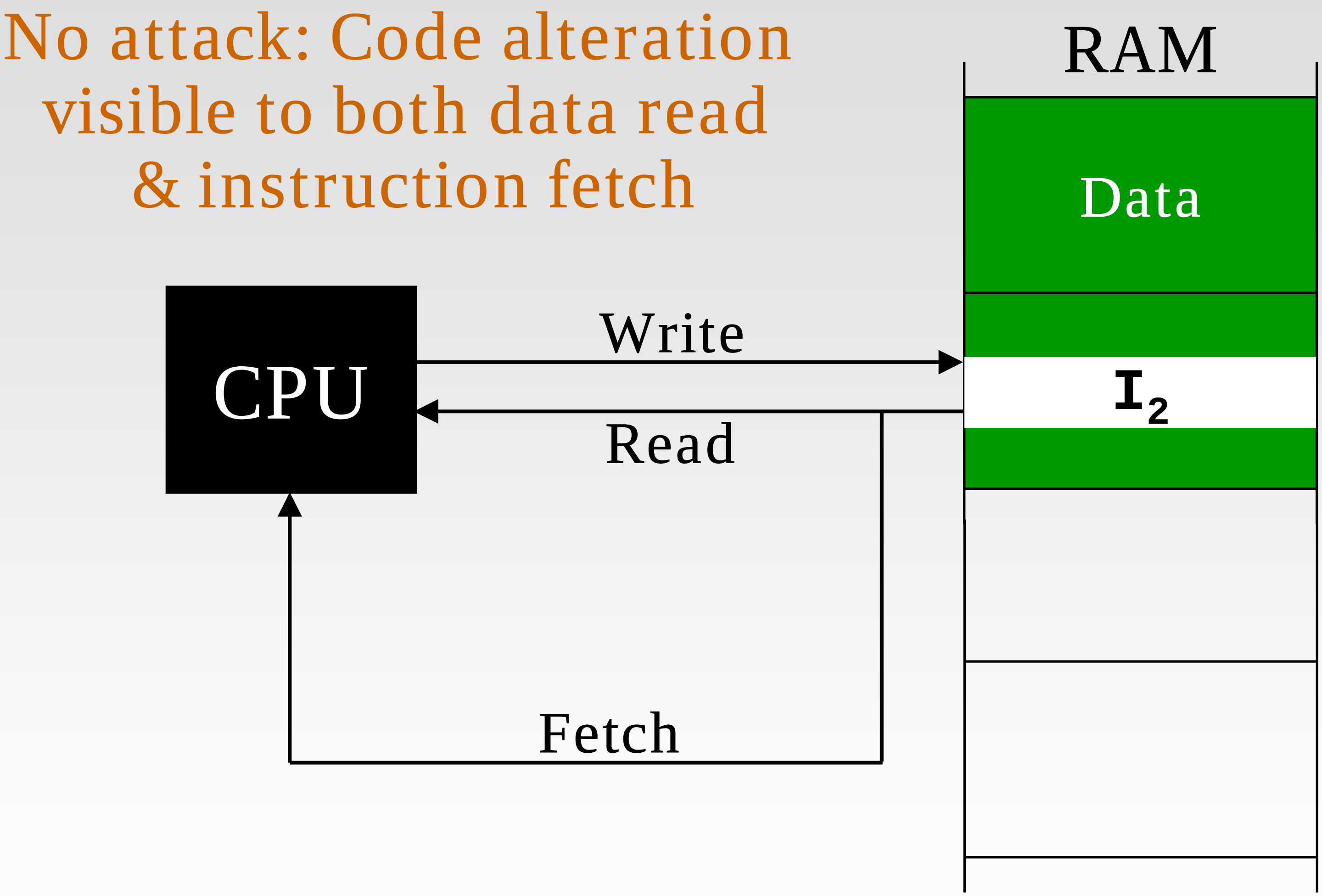
1. Overwrite instruction $\mathtt{I}_1$ at address *v* with new instruction $\mathtt{I}_2$ that alters control-flow
2. Read back the value at *v*
3. Execute the instruction at *v*

Attack Detection

No attack: Code alteration visible to both data read & instruction fetch

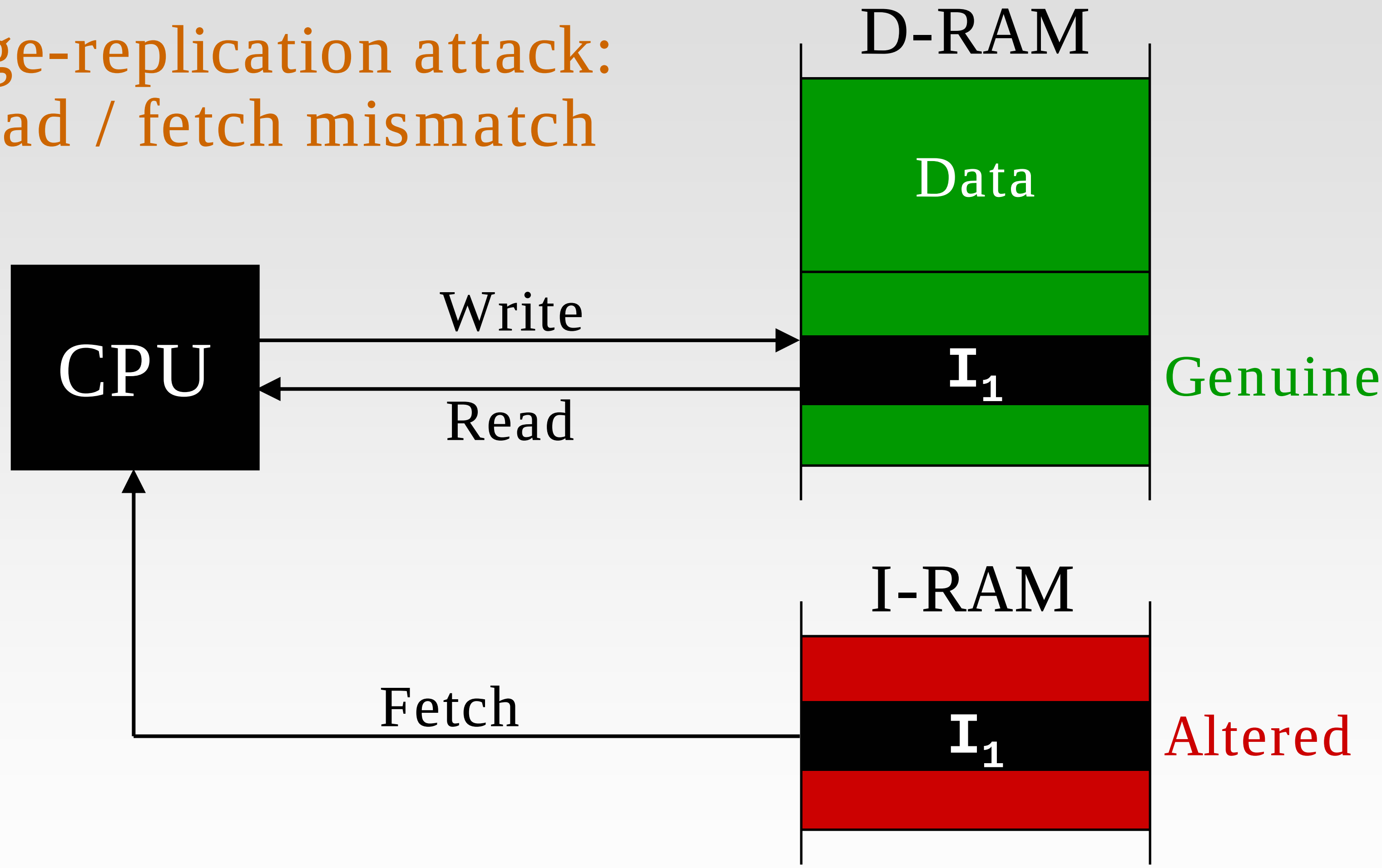


Attack Detection



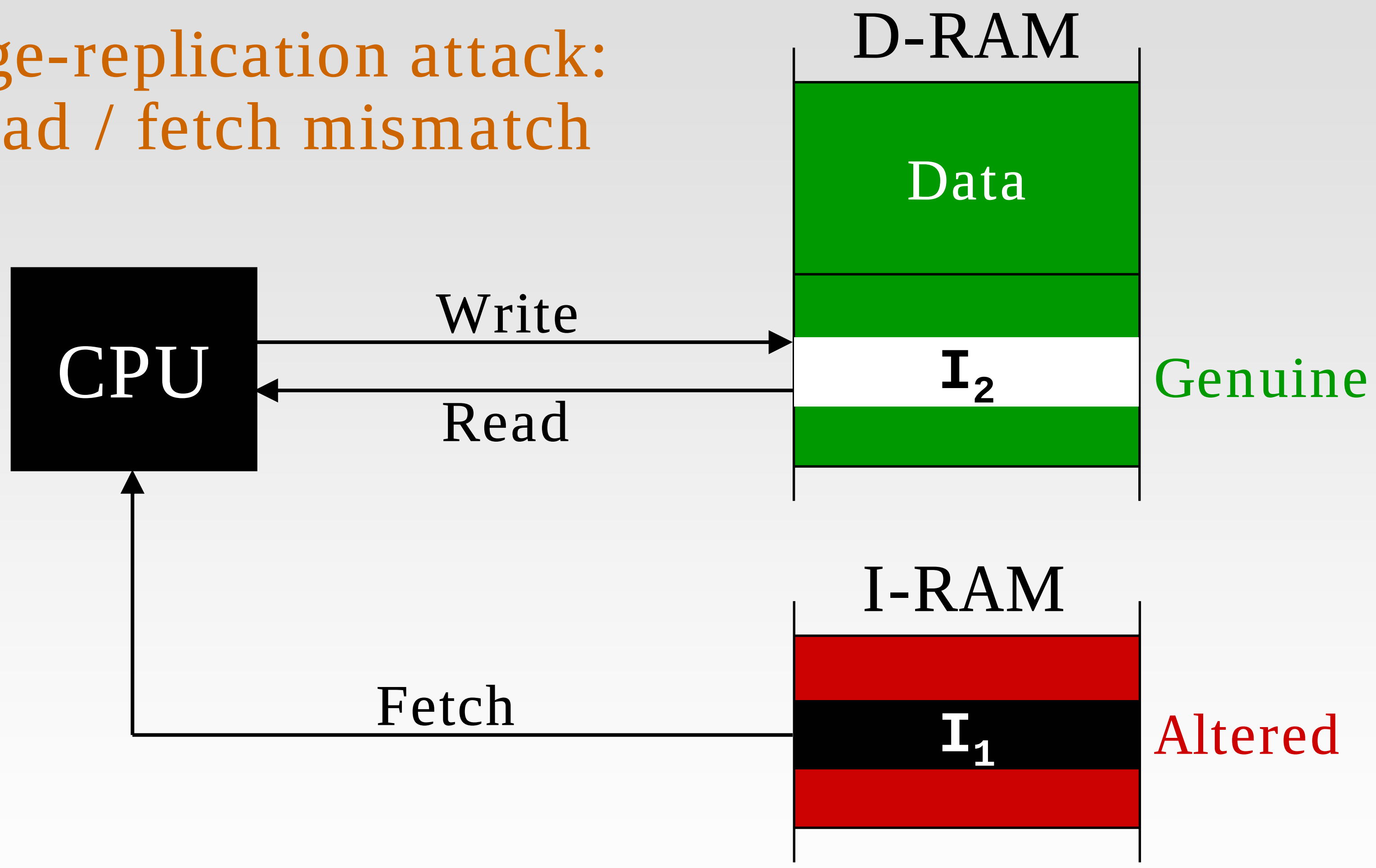
Attack Detection

Page-replication attack:
Read / fetch mismatch



Attack Detection

Page-replication attack:
Read / fetch mismatch



Solution

1. Overwrite instruction $\mathtt{I_1}$ at address *v* with new instruction $\mathtt{I_2}$ that alters control-flow
2. Read back the value at *v*
3. Execute the instruction at *v*

Value read

Control-flow path followed

	$\mathtt{I_1}$	$\mathtt{I_2}$
$\mathtt{I_1}$	Attack	Attack
$\mathtt{I_2}$	Attack	No Attack

Self-Checksumming and Reality

Jonathon T. Giffin, Mihai Christodorescu, Louis Kruger

Computer Sciences Department
University of Wisconsin

{giffin,mihai,lpkruger}@cs.wisc.edu

Taking Stock

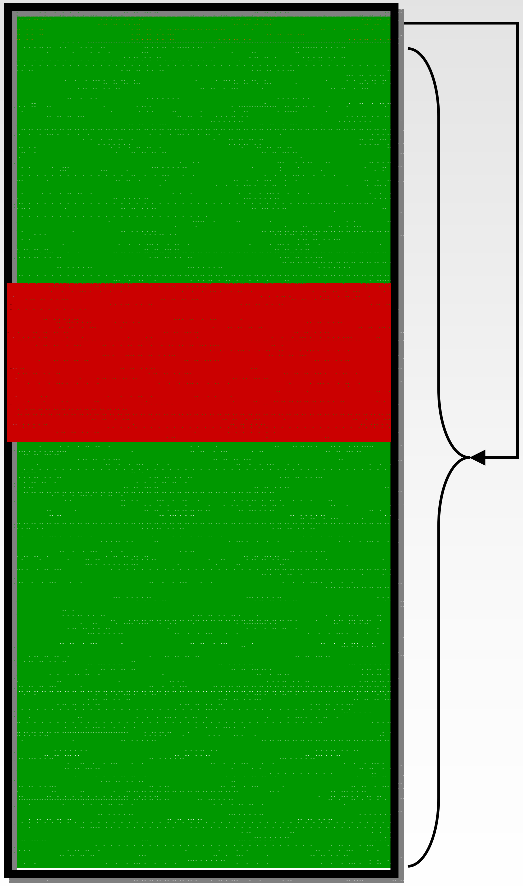
... So self-checksumming works again, right?

No.

Self-checksumming will always fail in current, realistic threat models.

Problem 3

Attackers first remove checksum code, then maliciously modify program



Solution: Redefine the Threat

The attacker cannot identify all relevant checksum code within the protected program.

"cannot identify" ➔ "cannot reverse engineer"

➔ Obfuscate

Solution: Redefine the Threat

- Network of guards [Chang & Atallah 2001]
  - – Many overlapping checksumming components
- Integrity Verification Kernels [Aucsmith 1996]
  - – Multithreaded, self-modifying checksumming components
- Testers and correctors [Horne *et al.* 2001]

Solution: Redefine the Threat

The attacker cannot identify all relevant checksum code within the protected program.

The attacker can reverse engineer & modify any non-checksumming code...

...but the attacker cannot reverse engineer & remove the checksum computation code.

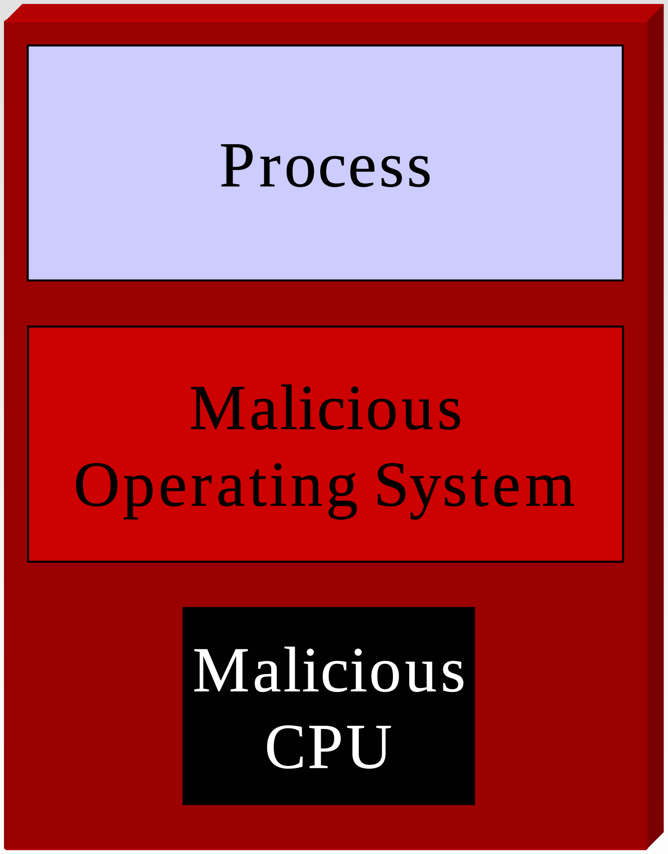
Realistic Threats

The attacker can understand and arbitrarily alter any code in the program.

[Madou *et al.* DRM 2005]

Root Problem

No trust base.



Root Problem

No trust base.

Self-checksumming will inherently and always fail in such an environment.

Root Problem

No trust base.

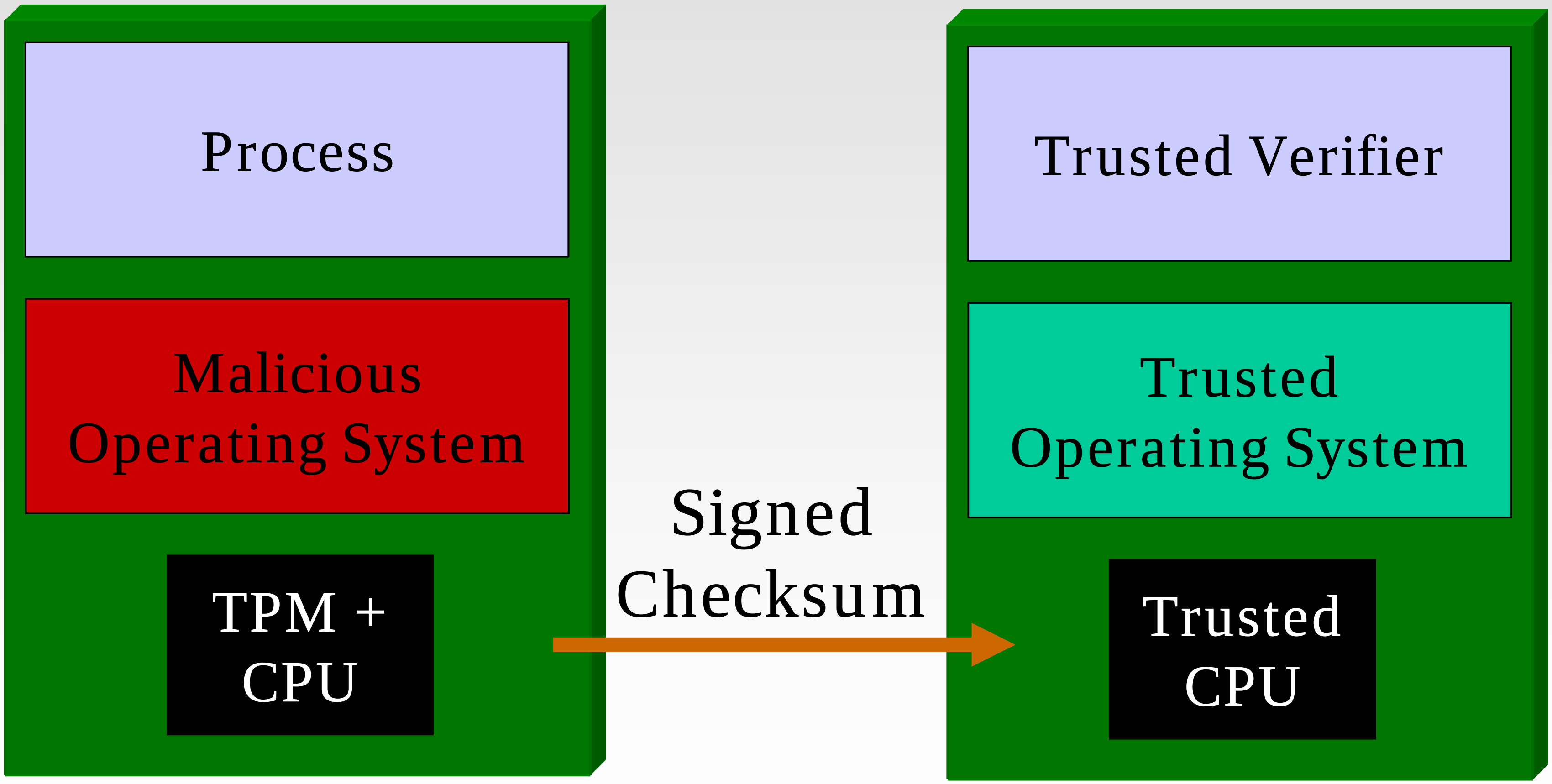
“Software alone never gets you assurance.”

“Need independent processor & address space.”

-- Brian Snow, 9:29 AM today

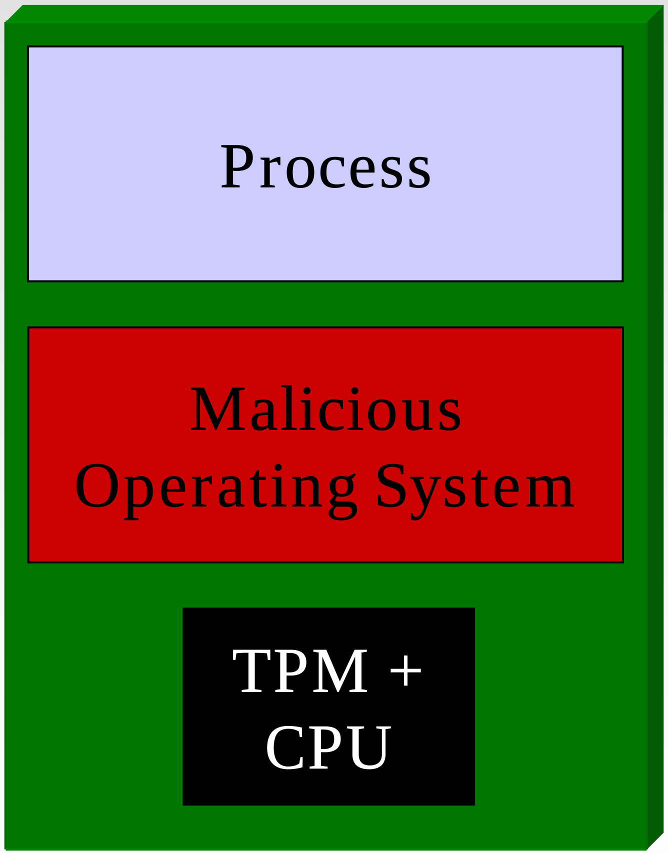
Solution

Trusted computing; remote verification



Solution

Trusted computing; remote verification



Trusted hardware alone is insufficient:

Malicious OS or malicious process can alter or remove local verification routines

Solution

Trusted computing; remote verification

Remote verification alone
is insufficient:

Malicious OS can again mount
page-replication attacks

Conclusions

- Strengthening self-checksumming via self-modifying code
  - Detects page-replication attack
- Fundamental attacks against self-checksumming remain valid
- Trusted hardware + remote verification needed for secure checksum validation

Questions?

Contact the authors:

Jonathon T. Giffin giffin@cs.wisc.edu

Mihai Christodorescu mihai@cs.wisc.edu

Louis Kruger lpkruger@cs.wisc.edu

Computer Sciences Department
University of Wisconsin