Establishing Software Root of Trust Unconditionally

(or, a First Rest Stop on the Never-Ending Road to Provable Security)

Virgil D. Gligor **Maverick S.-L. Woo**

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

NDSS 2019
San Diego, CA
February 27, 2019

Outline

I. What is it?

- Definition & relationships
- Unconditional solution

II. Why is it hard?

- 3 Problems
- RoT ≠ software-based, crypto attestation

III. How to do it?

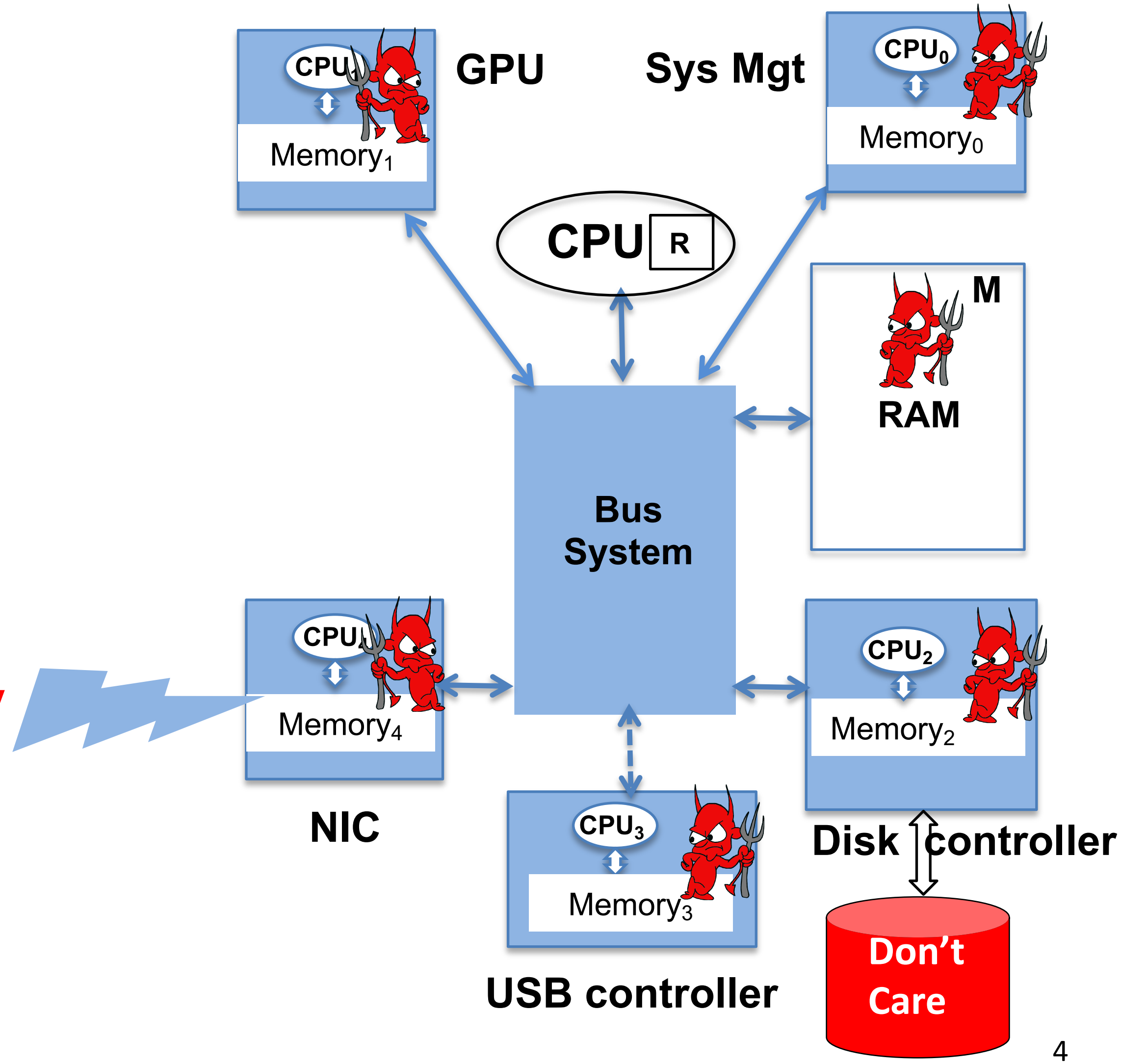
- **randomized polynomials**
  - k-independent (almost) universal hash families; ***and***
  - space-time optimal in **cWRAM**; ***and***
  - scalable optimal bounds

IV. Q & A

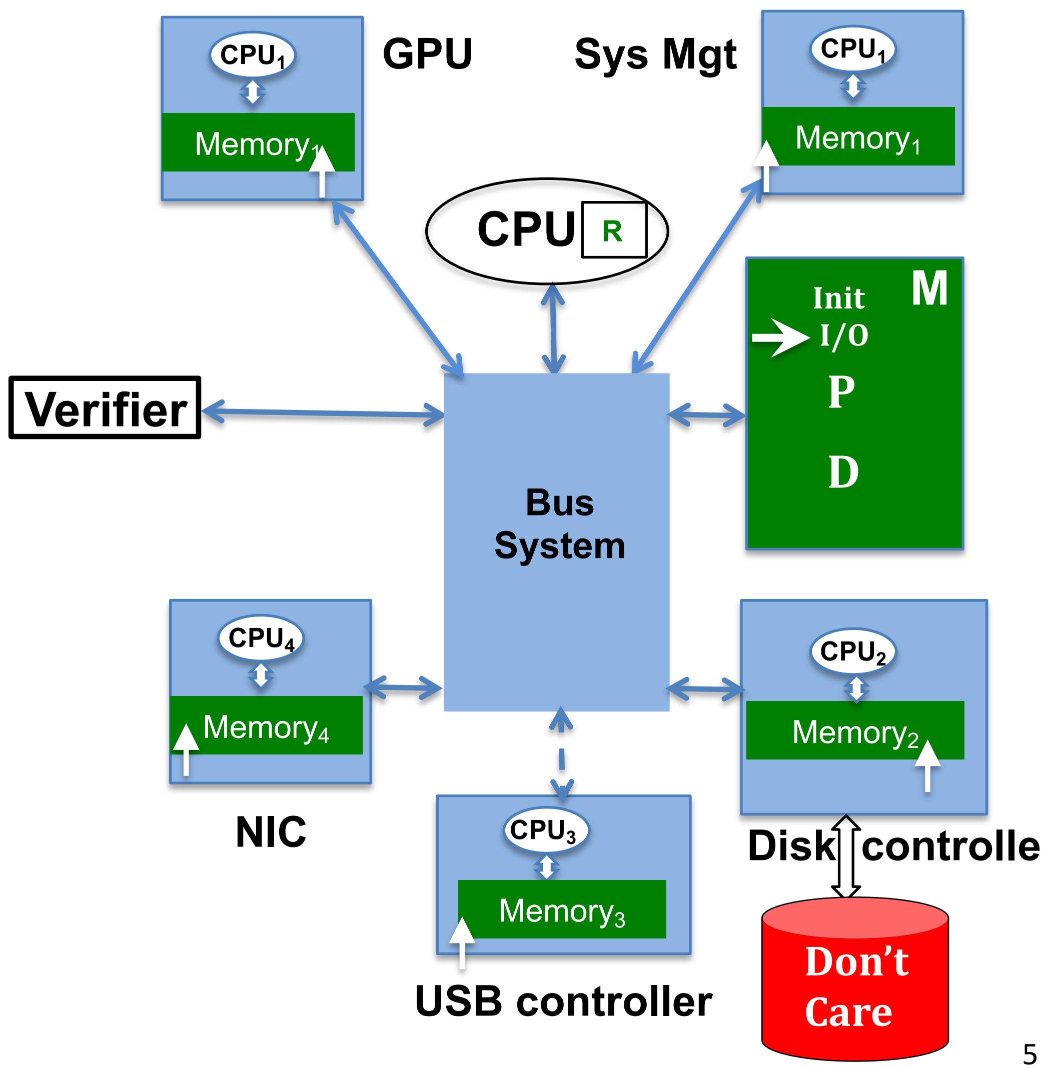
Full Paper is the CMU-CyLab TR 18-003
https://www.cylab.cmu.edu/_files/pdfs/tech_reports/CMUCyLab18003.pdf

I. What is it?

Controlled by a Powerful Adversary



Root of Trust (RoT) Establishment

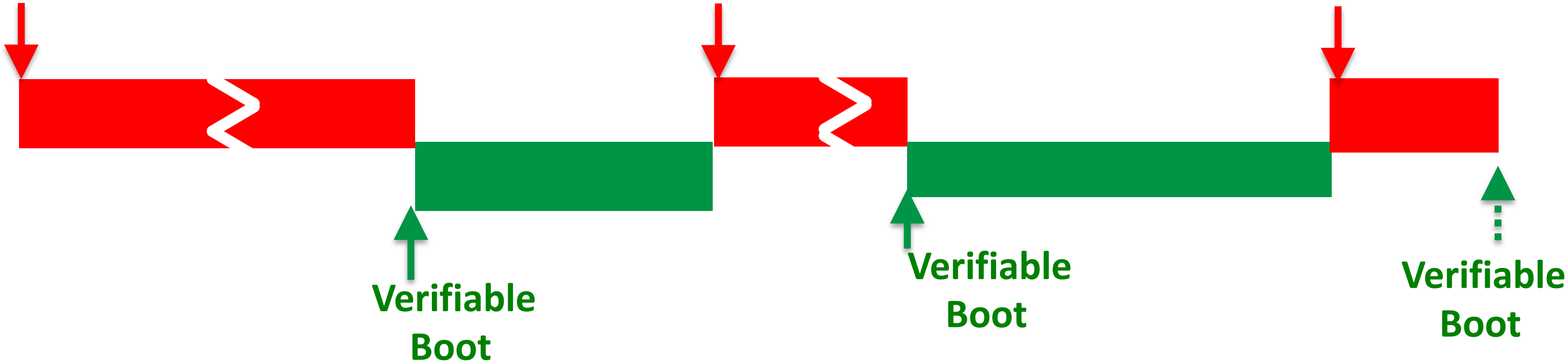


Secure State: ***RoT state*** (***chosen content***) *satisfies security predicate* ***P***

Verifiable boot:

either boot code in a ***secure state***

or detect unknown content



Verifiable boot => Secure State => ***RoT State***

Trusted Recovery => . . .

Access Control Models => . . .

. . .

*Unconditional Solution**

- **no** Secrets, **no** Trusted HW Modules, **no** Bounds on Adversary's Power

- need **only**
  - *random bits*
  - *device specifications*.

Importance?

- **no dependencies** on the unknown & unknowable
- a defender has a **provable advantage** over ***any*** adversary
- **outlives technology** advances.

---

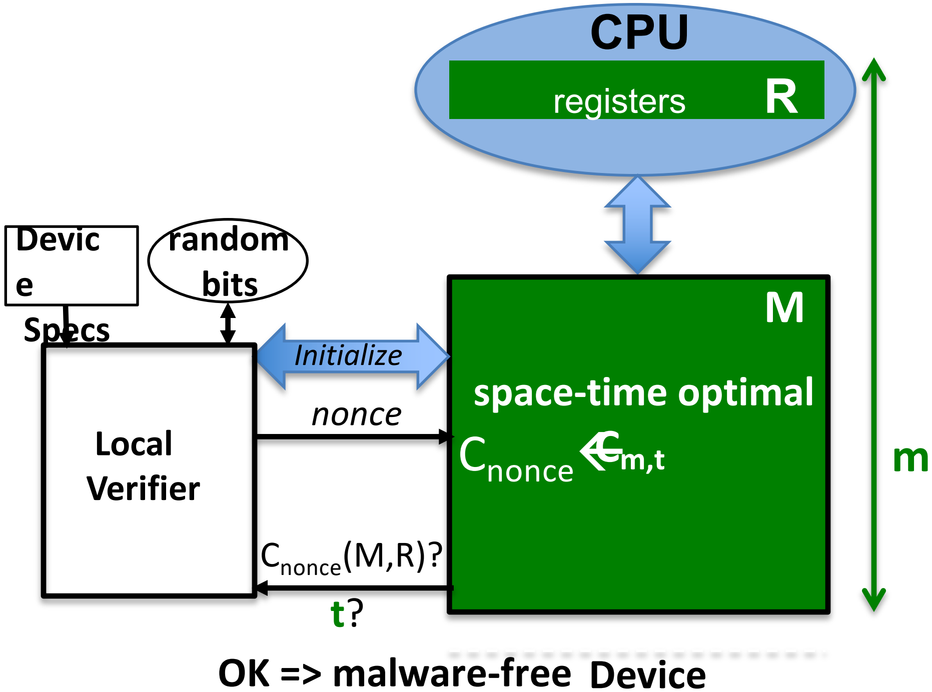
*I know of **no other unconditional solution** to any software security problem

I. What is it?

II. Why is it hard?

1. space-time optimal $C_{m,t}$ $\lesssim_{\text{Trusted Verifier}}$ malware-free Device

- non-asymptotic bounds
- on Device Specs; e.g., ISA ++
  (a *realistic model of computation?*)



Complexity theory?

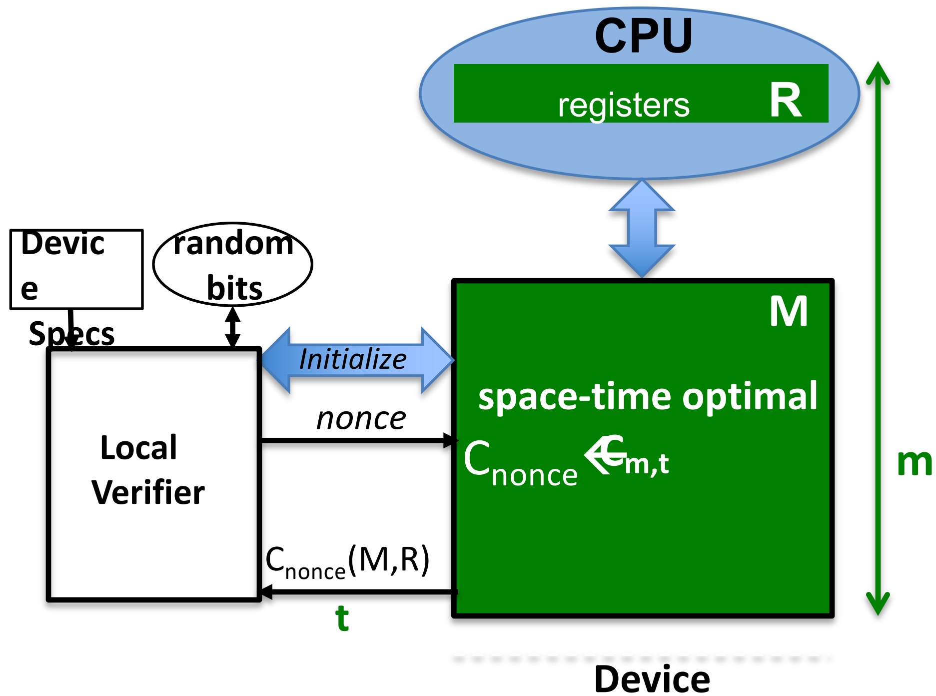
- *non-asymptotic bounds*? **Very few**
- *on Device Specs*? **None**

e.g., **Horner's rule** for polynomial evaluation
uniquely optimal in <u>infinite</u> fields: **2d** (**×**,**+**)
not optimal in <u>finite</u> fields,
nor on <u>*any*</u> Device ISA++

1. space-time optimal $C_{m,t}$ $\lesssim$ malware-free Device

Trusted
Verifier

- non-asymptotic bounds
- on Device Specs



1. space-time optimal $C_{m,t}$ $\lesssim$ malware-free Device

Trusted Verifier

- non-asymptotic bounds
- on Device Specs
- adversary execution?

Complexity Theory?

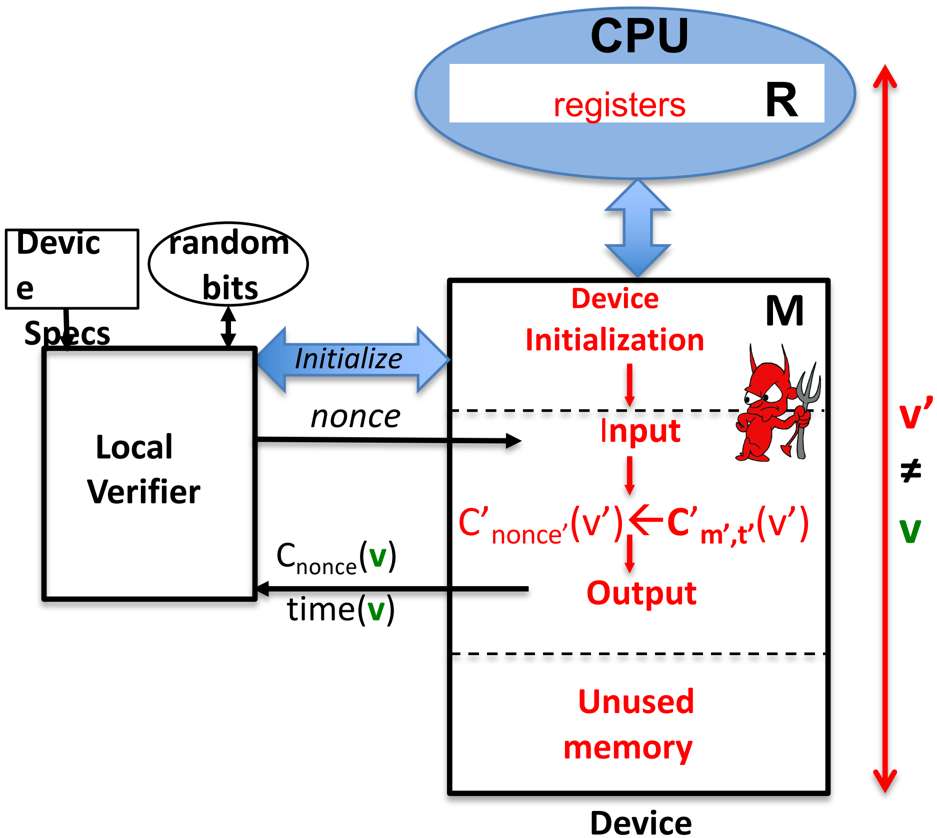
- **no help.**
- **how could it help**?

e.g., **malware beats m-t bounds**

=> C_{nonce}(v) becomes *unpredictable*

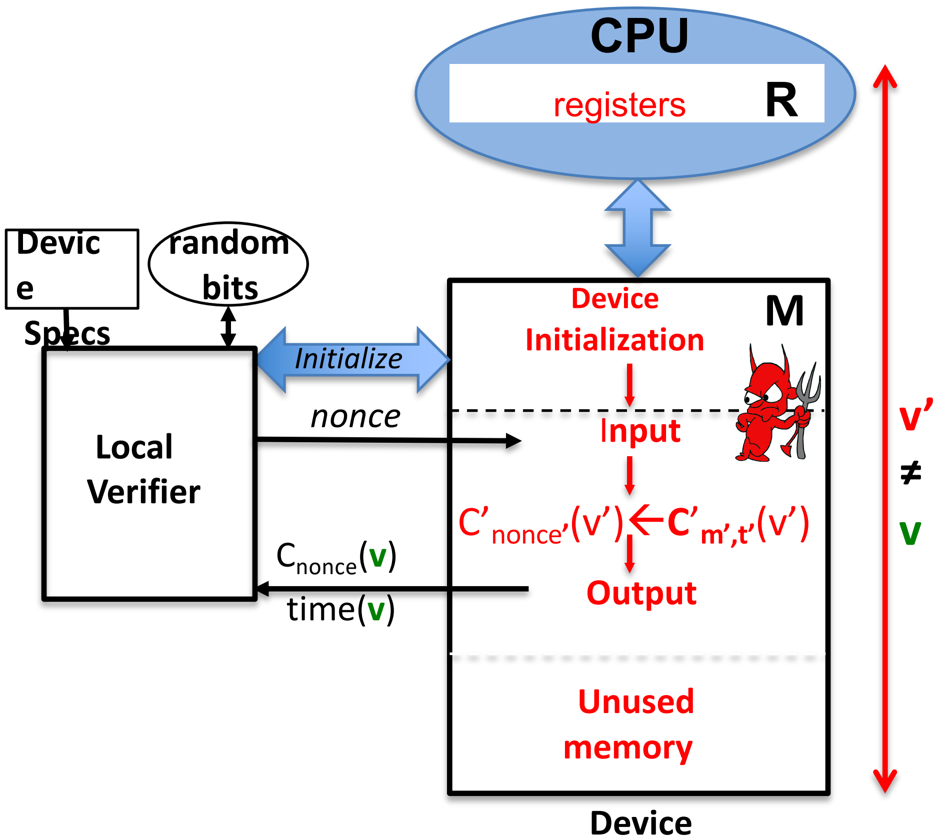
Engineering Solution?

e.g., see - segmented memory



1. space-time optimal $C_{m,t}$ $\lesssim_{\text{Trusted Verifier}}$ malware-free Device

- non-asymptotic bounds
- on Device Specs
- adversary execution

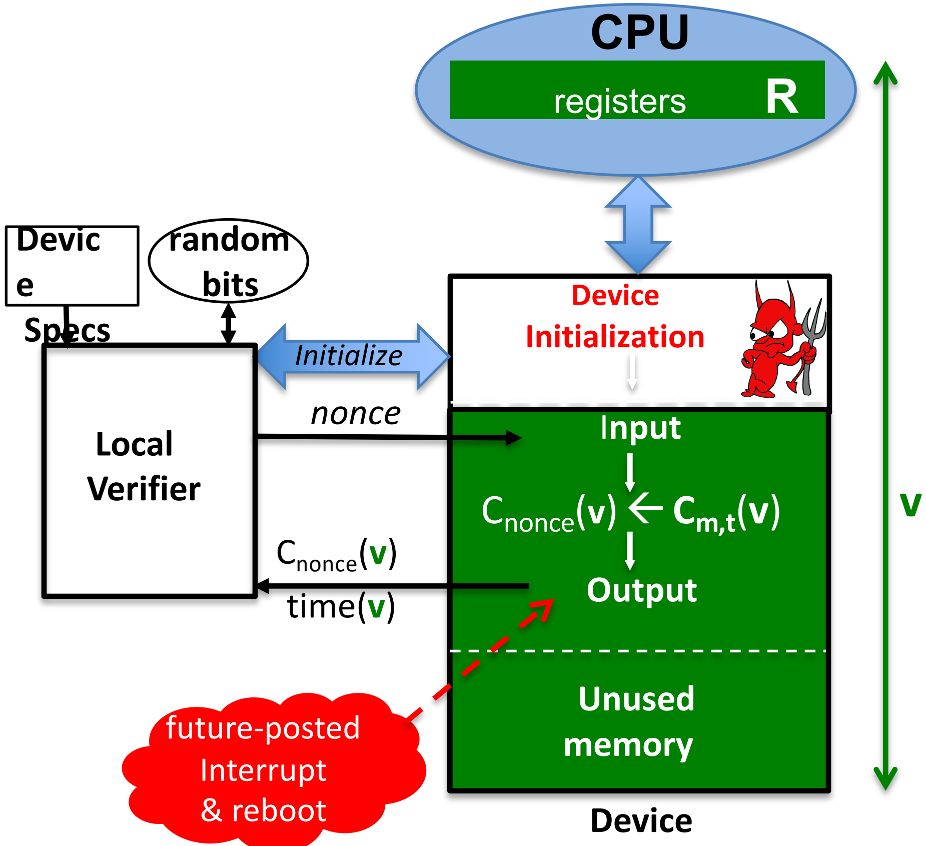


1. space-time optimal $C_{m,t}$ $\lesssim$ malware-free Device

Trusted
Verifier

- non-asymptotic bounds
- on Device Specs
- adversary execution

Reduction is insufficient !



1. space-time optimal $C_{m,t}$ $\lesssim$ malware-free Device ✓

Trusted Verifier

- non-asymptotic bounds
- on Device Specs
- adversary execution

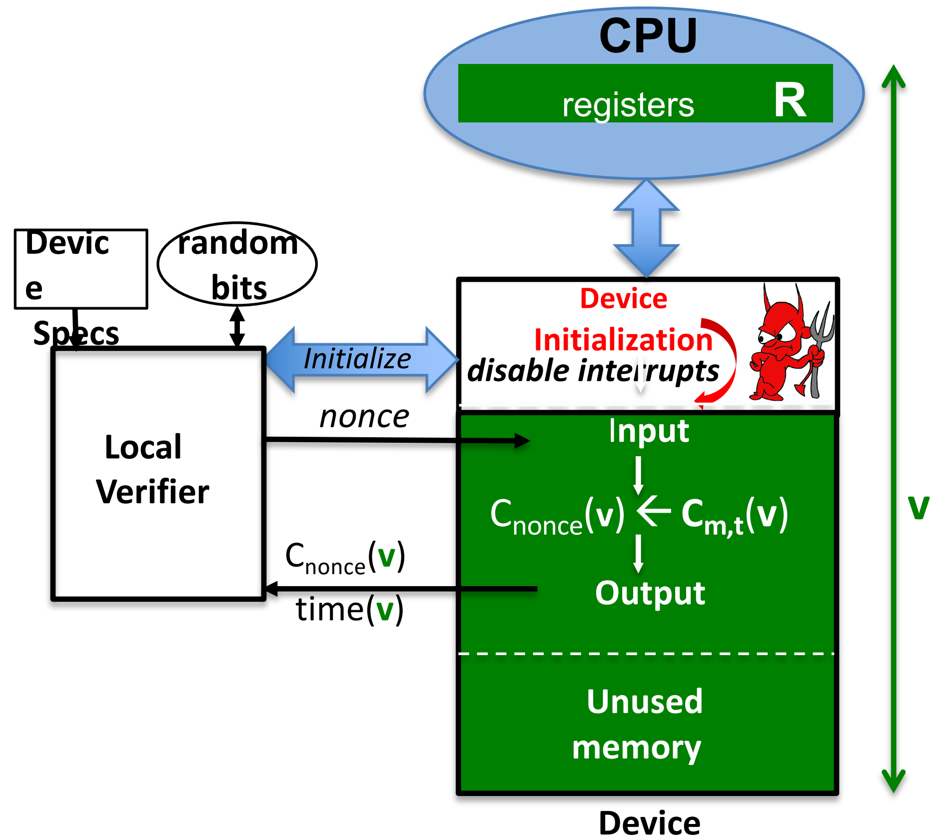
Reduction is insufficient !

Solution?

control flow integrity *after* $\mathbf{C_{nonce}}$ ends

=>

control flow integrity *before* $\mathbf{C_{nonce}}$ starts!

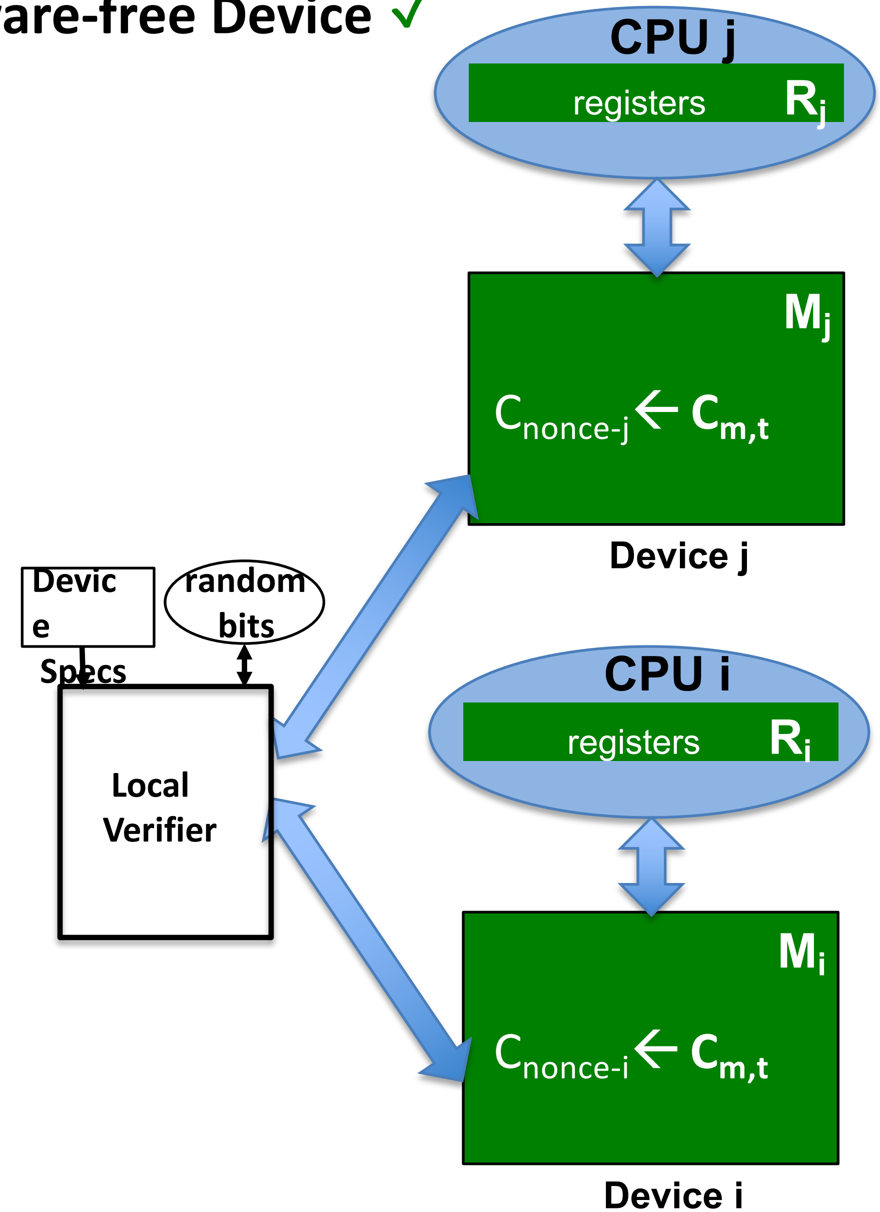


1. space-time optimal $C_{m,t}$ $\lesssim$ malware-free Device ✓
Trusted Verifier

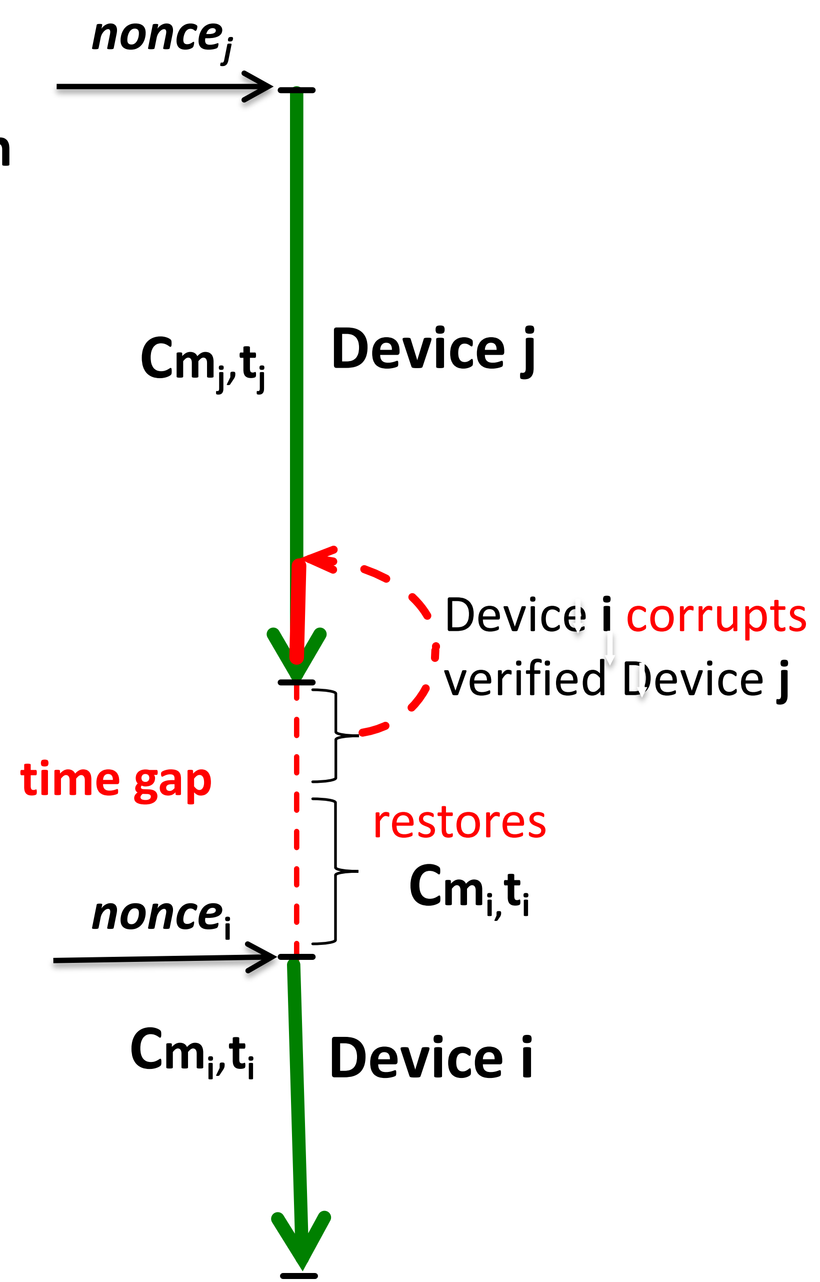
- non-asymptotic bounds
- on Device Specs
- adversary execution

2. Verifiable Control Flow ✓

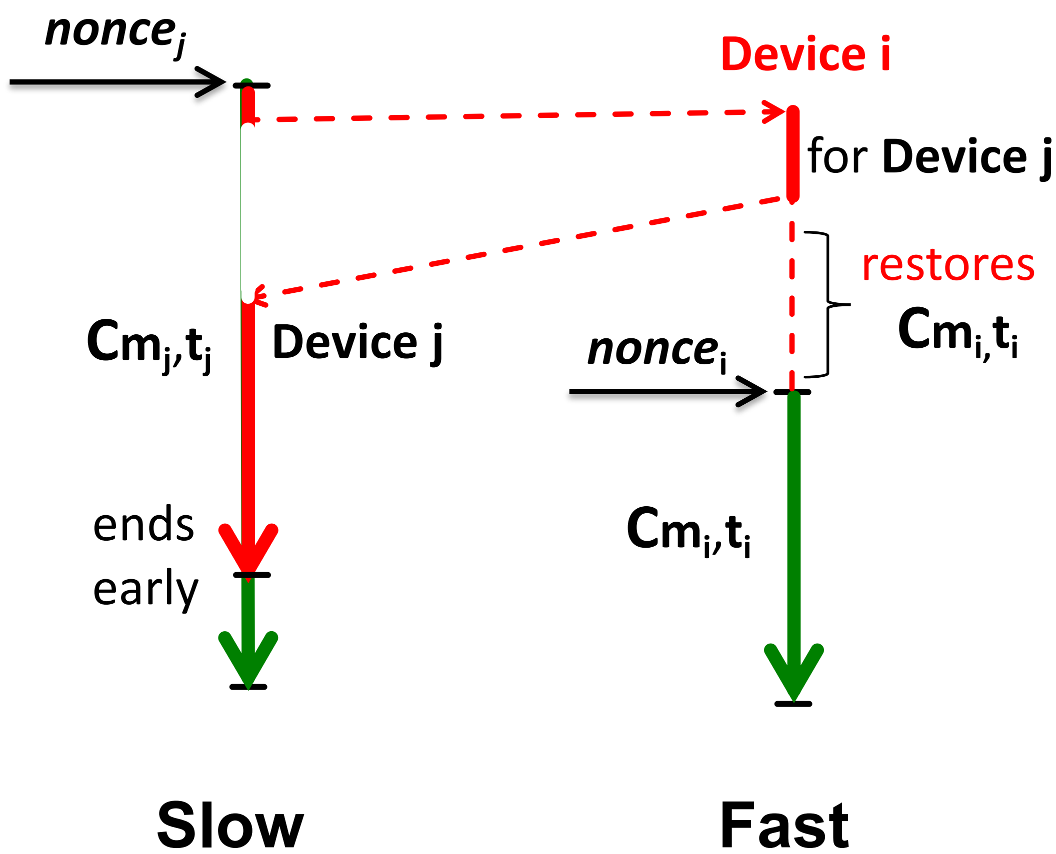
3. Two Devices, or more?



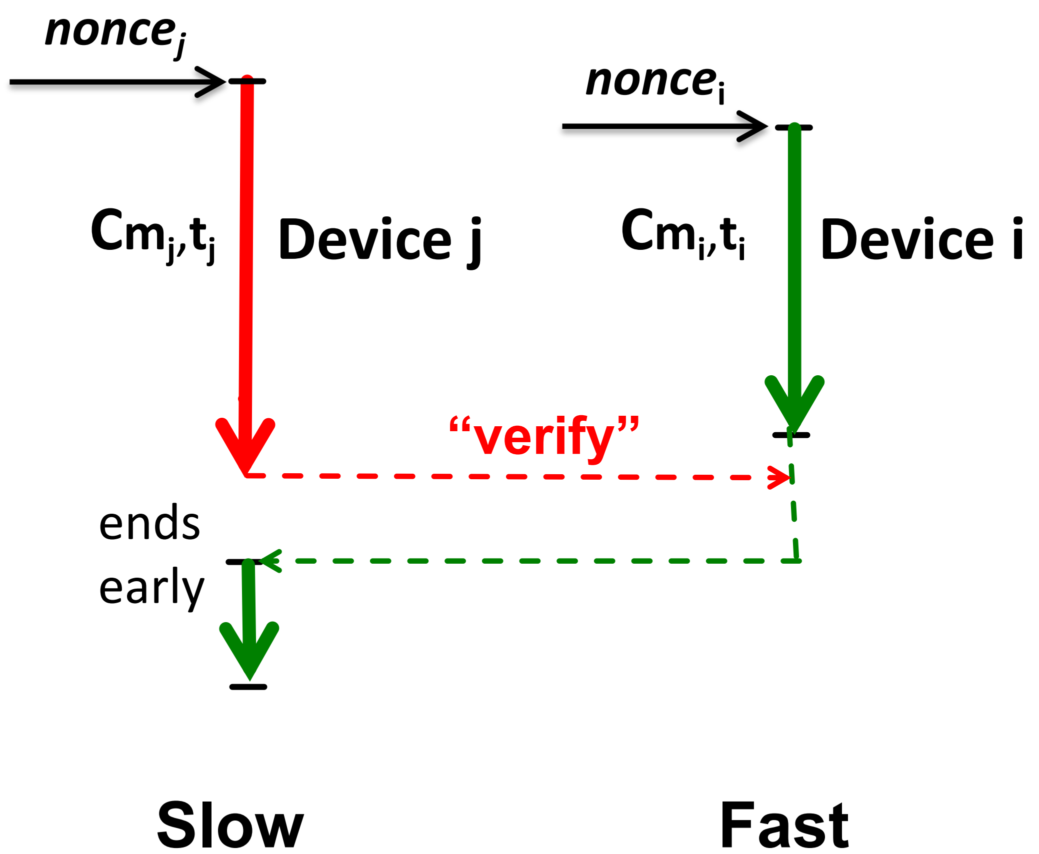
- **sequential verification fails**



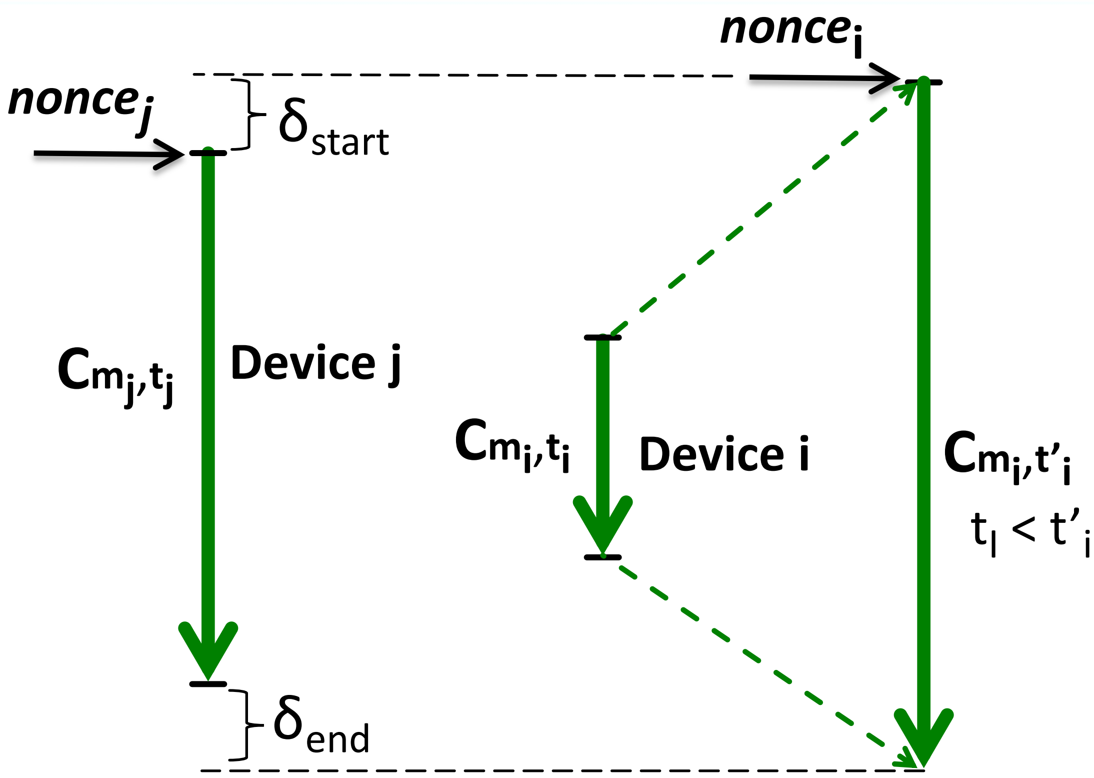
- **ordinary concurrency fails**

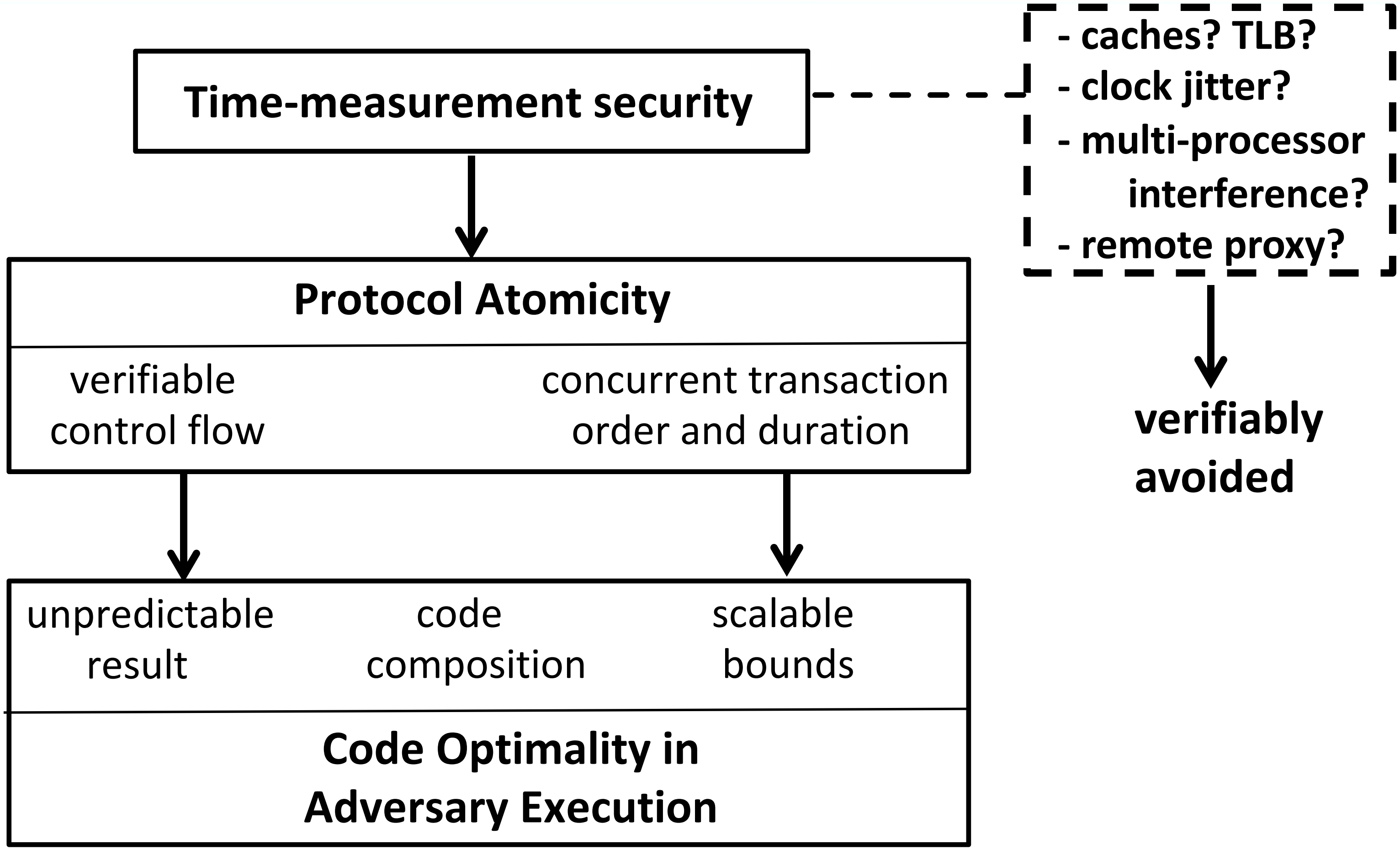


- **ordinary concurrency fails**

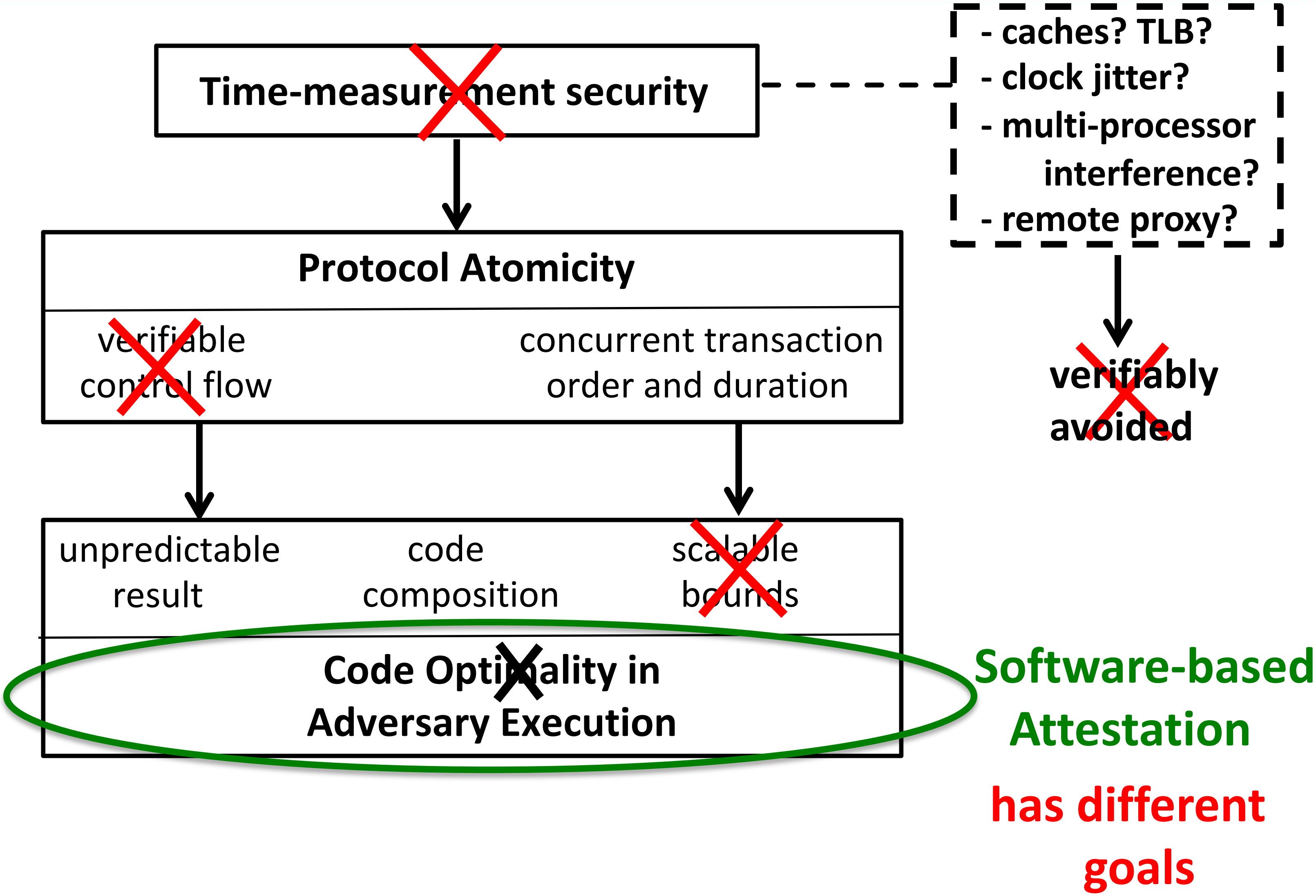


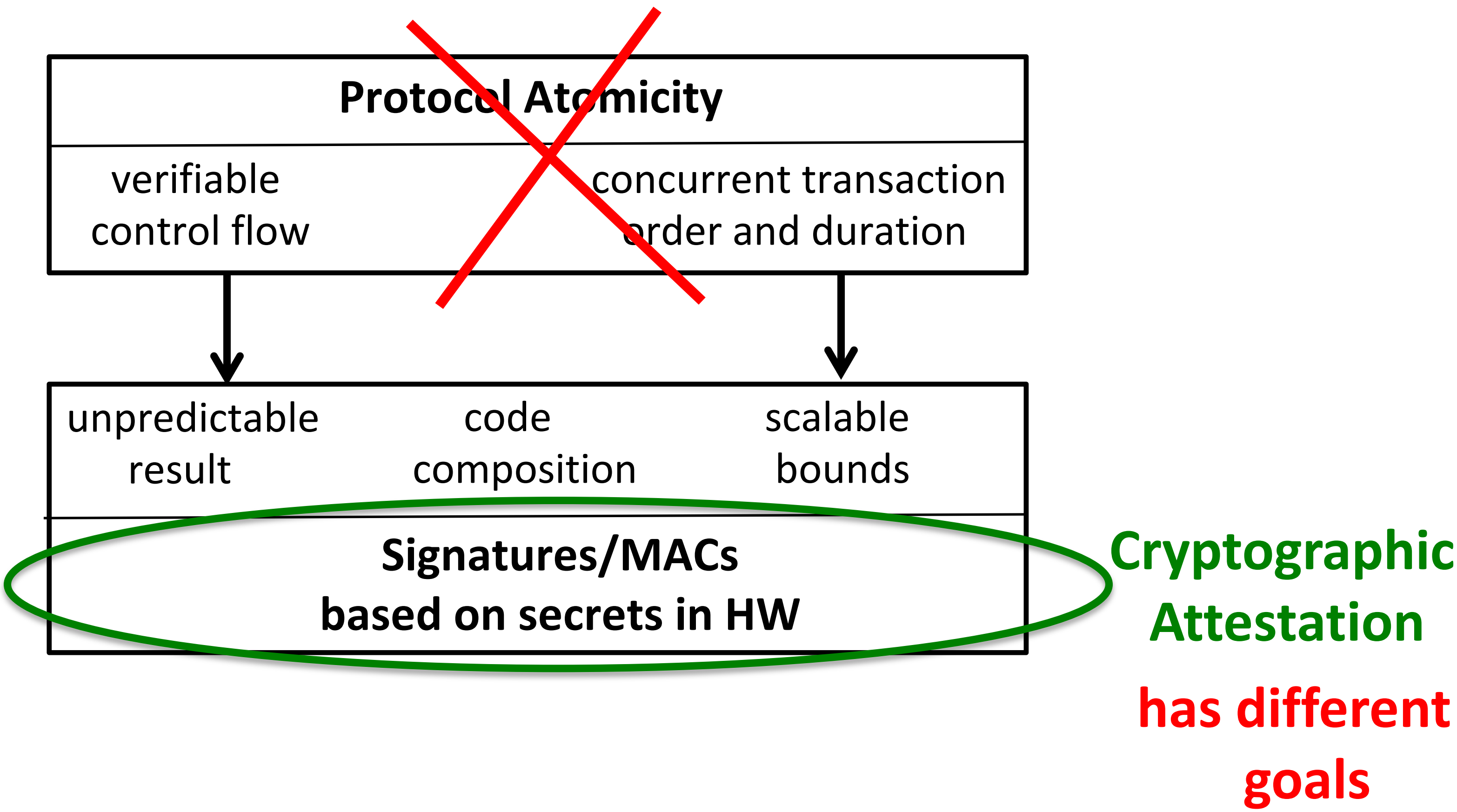
- concurrent verification w/ scalable bounds





Legend: ← dependency


Time-measurement security
- caches? TLB?
- clock jitter?
- multi-processor interference?
- remote proxy?
verifiably avoided
Protocol Atomicity
verifiable control flow
concurrent transaction order and duration
unpredictable result
code composition
scalable bounds
Code Optimality in Adversary Execution
Software-based Attestation
has different goals


Protocol Atomicity
verifiable control flow
concurrent transaction order and duration
unpredictable result
code composition
scalable bounds
Signatures/MACs based on secrets in HW
Cryptographic Attestation
has different goals

I. What is it?

II. Why is it hard?

III. How to do it

Solution Overview

Randomized Polynomials

new - k-independent uniform coefficients, independent of input x

↓

new kind - k-independent (almost) universal hash function family

and

new - (**m**, **t**)-*optimal* in the concrete Word Random Access Machine (**cWRAM**)

and

new - optimal bounds **m** and **t** are scalable; e.g., no mandatory **m•t** tradeoffs

Overview of the cWRAM ISA++

- ***Constants:*** ***w**-bit word*, up to ***2 operands****/instruction*
instructions execute in ***unit time***

- ***Memory***: ***M*** words
- **Processor registers *R***: GPRs, PC, PSW, Special Processor + Flag & I/O Registers
- ***Addressing***: immediate, relative, direct, indirect
- ***Architecture features:*** caches, virtual memory, TLBs, pipelining, multi-core processors

- ***ISA: all (un)signed integer instructions***
  - All Loads, Stores, Register transfers
  - All Unconditional & Conditional Branches, all branch types
    - *all predicates with 1 or 2 operands*
  - *Halt*
  - All *Computation* Instructions:
    - addition, subtraction, logic, $shift_{r/l}(R_i, \alpha)$, $rotate_{r/l}(R_i, \alpha)$, . . .
    - *variable* $shift_{r/l}(R_i, R_j)$, *variable* $rotate_{r/l}(R_i, R_j)$, . . .
    - multiplication (1 register output). . .
    - *mod* (aka., division-with-remainder) . . .

random bits

↓

$\{ r_0 \ldots r_{k-1}, x \} \xleftarrow{\$} \mathbf{Z_p}$

nonce

↓

$$\mathbf{H}r_0\ldots r_{k-1},x(\mathbf{v}) = \sum_{i=d}^{0} (\mathbf{s_i} \oplus \mathbf{v_i}) \cdot x^i \pmod{p}, \quad \mathbf{s_i} = \sum_{j=0}^{k-1} r_j (i+1)^j \pmod{p}$$

$$d = |\mathbf{v}| - 1$$

k-independent almost universal hash function family

$$\mathbf{C}_{nonce}(\mathbf{v}) = \mathbf{H}r_0\ldots r_{k-1},x(\mathbf{v}) = \mathbf{H}_{d,k,x}(\mathbf{v})$$

m-t *optimal* bounds on **cWRAM: m** = **k** + 22, **t** = (6**k** - 4)6**d**

Scalable bounds: k↑ => m↑, t↑ and **d↑ => t↑**

Foundation

Theorem 1

Let w > 3, and *p* be a prime, $2 < p < 2^{w-1}$.
Horner's rule for *one-time honest evaluation* of $P_d(\cdot)$ in cWRAM

$$P_d(\cdot) = \sum_{i=d}^{0} a_i \cdot x^i \pmod{p} = (\ldots(a_d \cdot x + a_{d-1}) \cdot x + \ldots + a_1) \cdot x + a_0 \pmod{p}$$

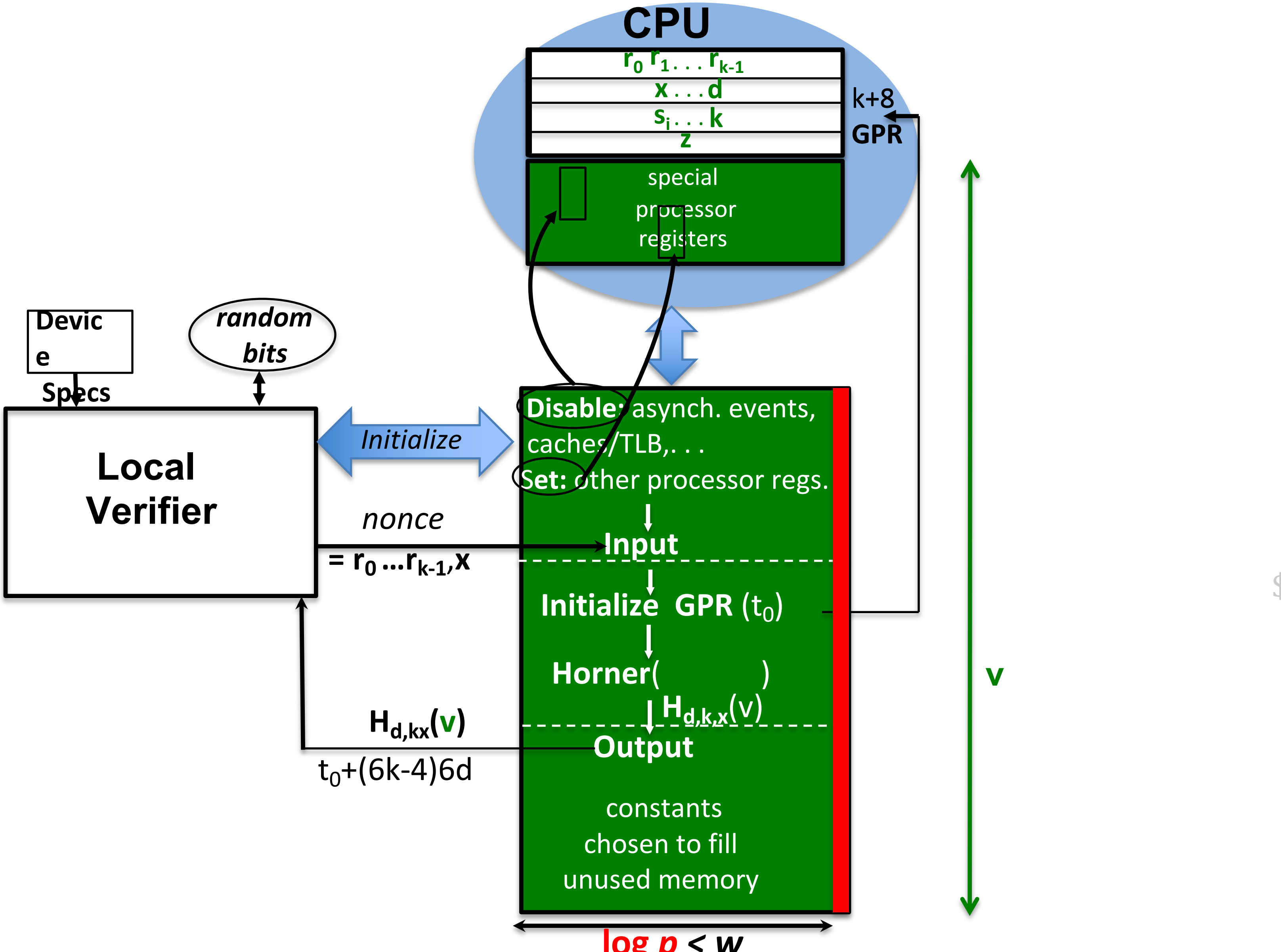
is *uniquely (m, t)-optimal* if the cWRAM execution space & time are *simultaneously minimized*; i.e., $m = d+11$, $t = 6d$.

Answer to A. M. Ostrowski's 1954 question:

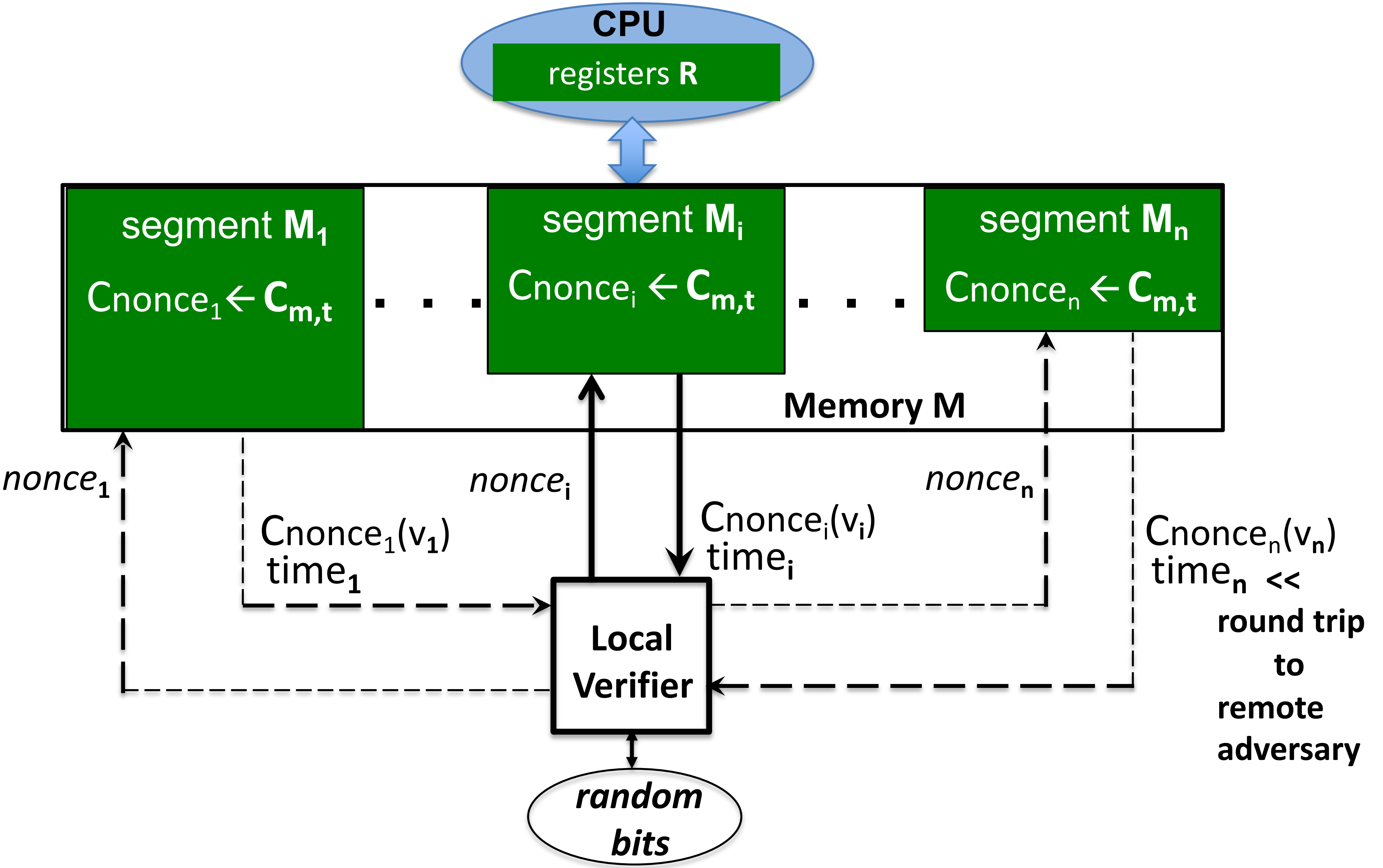
"Is Horner's rule optimal for polynomial evaluation?"

with non-asymptotic bounds in a realistic model of computation (**cWRAM**)

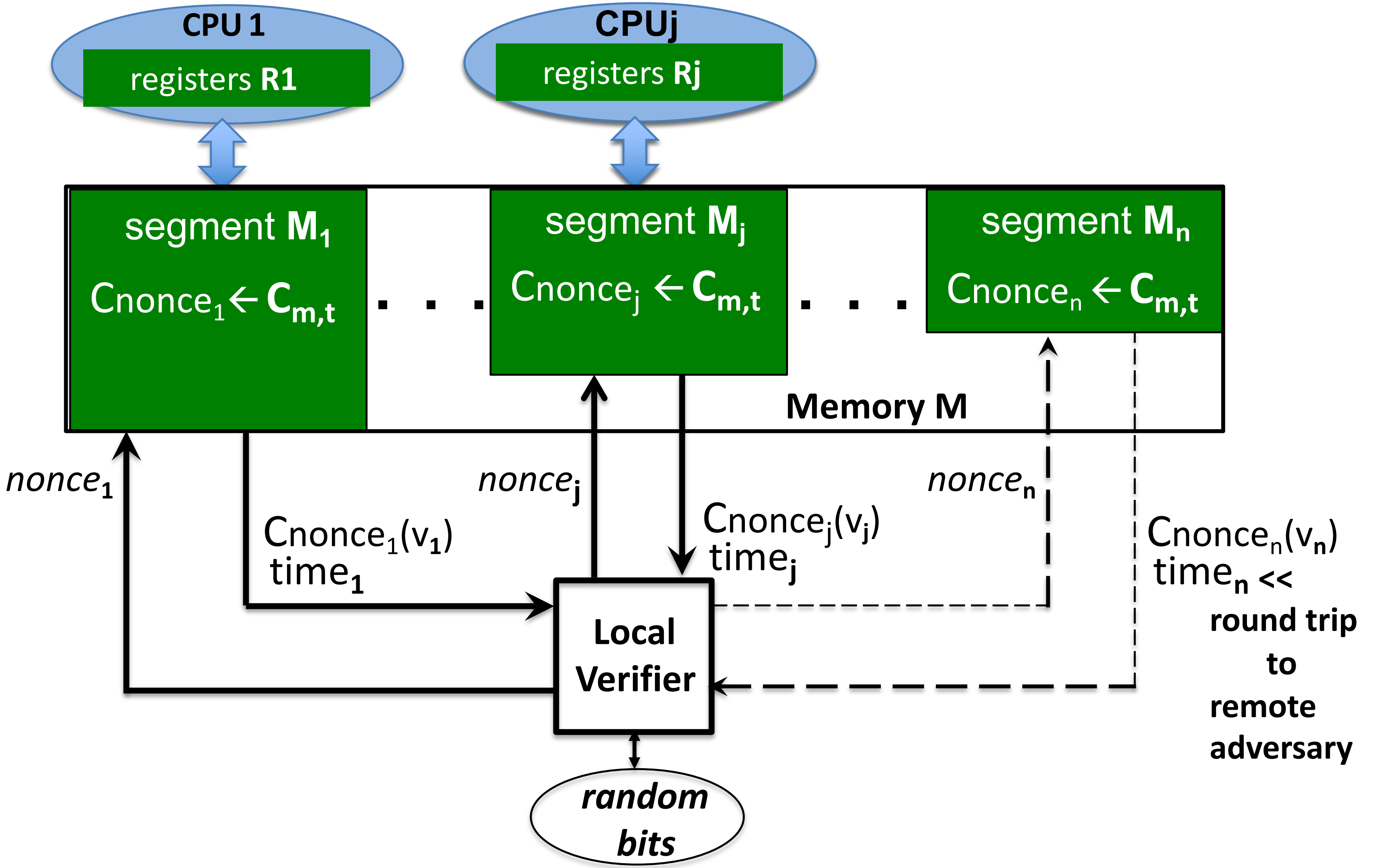
IV. Q & A



OK => **malware-free Device** → **2nd Pass w/ ordinary UHF**



verifier's random choice of segment i


CPU 1
registers R1
CPUj
registers Rj
segment M_1
$Cnonce_1 \leftarrow C_{m,t}$
segment M_j
$Cnonce_j \leftarrow C_{m,t}$
segment M_n
$Cnonce_n \leftarrow C_{m,t}$
Memory M
$nonce_1$
$Cnonce_1(v_1)$ $time_1$
$nonce_j$
$Cnonce_j(v_j)$ $time_j$
$nonce_n$
$Cnonce_n(v_n)$ $time_n$ << round trip to remote adversary
Local Verifier
random bits

Implementation Notes

(Appendix C of CMU-CyLab TR 18-003)

Optimal Code: ($s_i \oplus v_i$) , **loop control** – simple on most real processors

Horner-rule step? (recall: *p* is largest prime in w bits)

multiply	*multiply*	*multiply-add mod* 2^w
mod p	*div p*	
add	*multiply p &*	*reduction mod p @ end*
mod p	*subtract*	
	add	$\Rightarrow$ *8 + 3 (reduction) instructions*
	div p	
	multiply p &	
	subtract	
m, T	**M, T**	**M, t**

different encodings => different results => SINGLE CHOICE!