Application Memory Authentication*

David Champagne, Reouven Elbaz and Ruby B. Lee

* D. Champagne, R. Elbaz and R. B. Lee, “The Reduced Address Space (RAS) for Application Memory Authentication”, In Proceedings of the 11th Information Security Conference (ISC’08), September 2008.

Introduction

- **Background:**
  - ✓ TPM, XOM, AEGIS, SP, SecureBlue want to provide **trust in an application's computations** and **protect private information**.
  - ✓ An adversary **corrupting** the memory space of an application can **affect the trustworthiness** of its computations.
- **Security Model:**
  - ✓ Threats:
    - Physical attacks: Tampering with bus data or memory chip
    - Software (SW) Attacks: Compromised OS
  - ✓ Assumptions:
    - Processor chip is the security perimeter
    - Application to protect is correctly written (no SW vulnerabilities)
    - On-chip engine can authenticate initial state of application
- **Objective**
  - ✓ Provide ***application memory authentication***: What the application reads from a memory location is what it last wrote there.

Outline

- Introduction to memory integrity trees

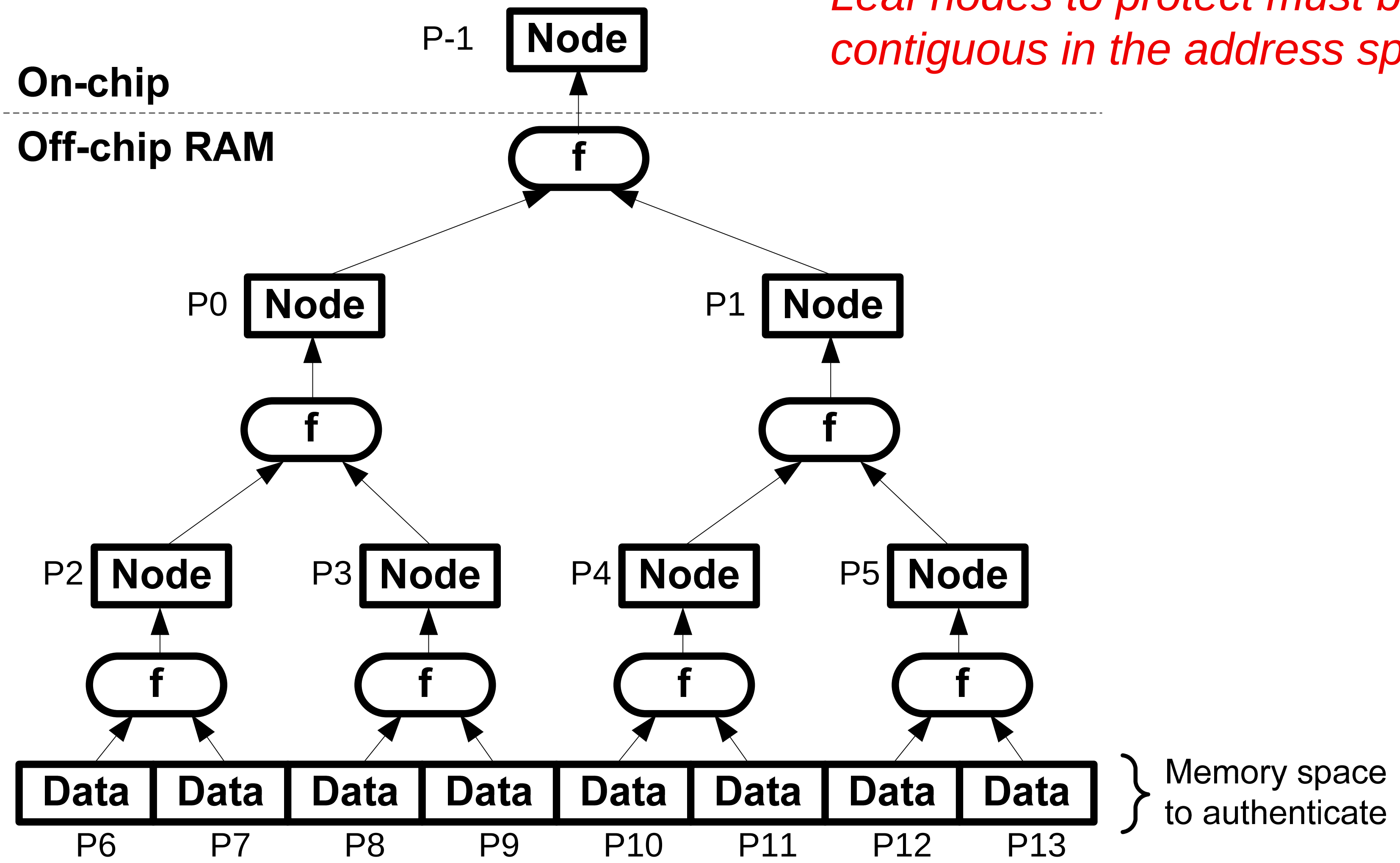
- Past Work:
  - ✓ Building a tree over the physical address space (PAS Tree)
  - ✓ Building a tree over the virtual address space (VAS Tree)

- Proposed Approach
  - ✓ A novel Reduced Address Space (RAS)
  - ✓ Building a tree over the RAS (RAS Tree)
  - ✓ Managing the RAS Tree with the Tree Management Unit (TMU)
  - ✓ Performance evaluation

- Conclusion

Addressing Nodes in an Integrity Tree

Leaf nodes to protect must be contiguous in the address space



On-chip

Off-chip RAM

P-1 Node

f

P0 Node

P1 Node

f

f

P2 Node

P3 Node

P4 Node

P5 Node

f

f

f

f

Data Data Data Data Data Data Data Data

P6 P7 P8 P9 P10 P11 P12 P13

Memory space to authenticate

$$parent\_position = \lfloor node\_position \div arity \rfloor - 1$$

Outline

- Introduction to memory integrity trees

- Past Work:
  - ✓ Building a tree over the physical address space (PAS Tree)
  - ✓ Building a tree over the virtual address space (VAS Tree)

- Proposed Approach
  - ✓ A novel Reduced Address Space (RAS)
  - ✓ Building a tree over the RAS (RAS Tree)
  - ✓ Managing the RAS Tree with the Tree Management Unit (TMU)
  - ✓ Performance evaluation

- Conclusion

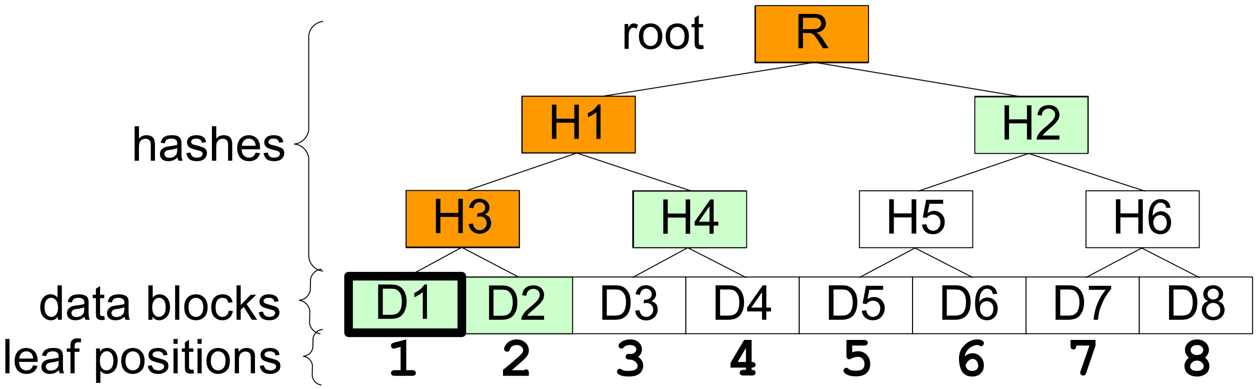
Physical Address Space (PAS) Tree

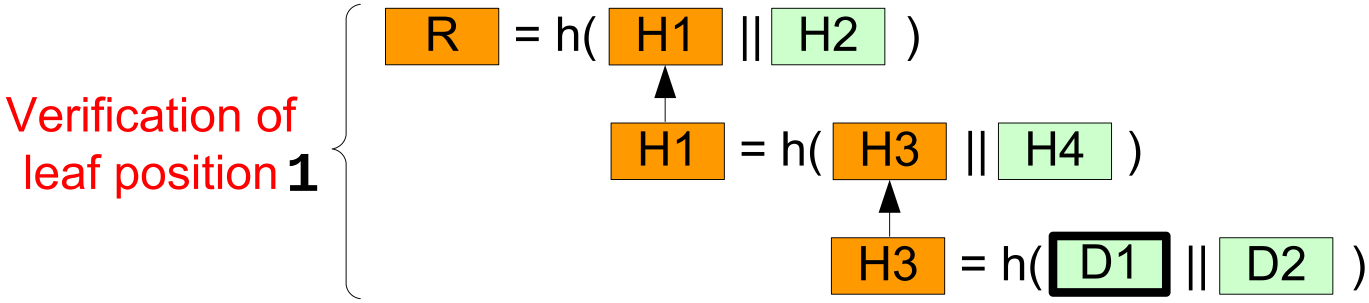
- Majority of past work implements a PAS Tree, where:
  - Tree nodes form a contiguous memory region in the PAS
  - An extra mechanism to protect paged data is needed

- *Problem*: with untrusted OS, ***branch splicing*** attack can be carried out

Branch splicing attack: splicing of memory data via page table corruption

On-Chip Root Recomputation Example 1/2





Verification of leaf position **1**

R = h(H1 || H2)

H1 = h(H3 || H4)

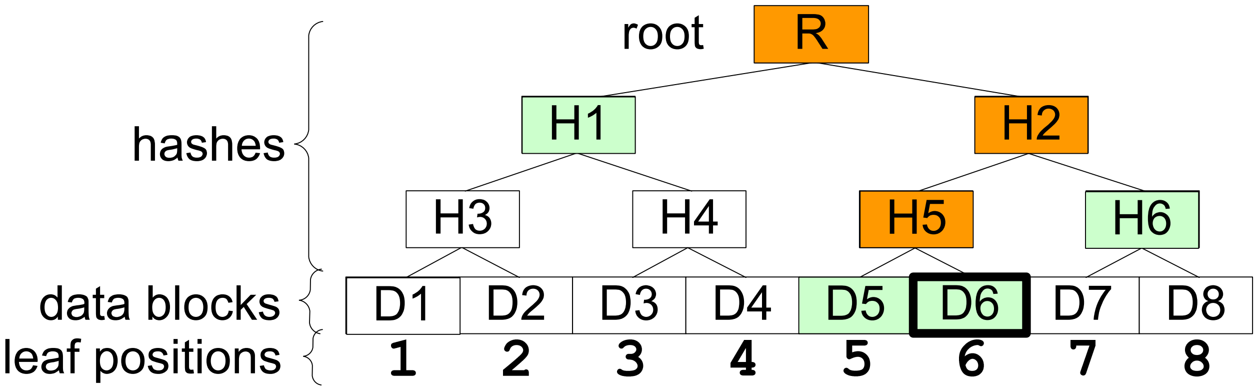
H3 = h(D1 || D2)

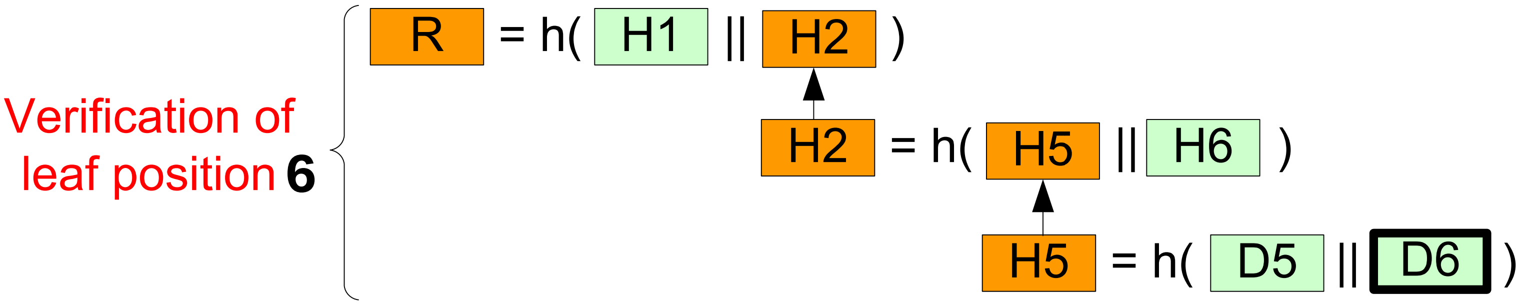
🟧 X recomputed on-chip

🟩 X fetched from memory

▣ X leaf to verify (fetched from memory)

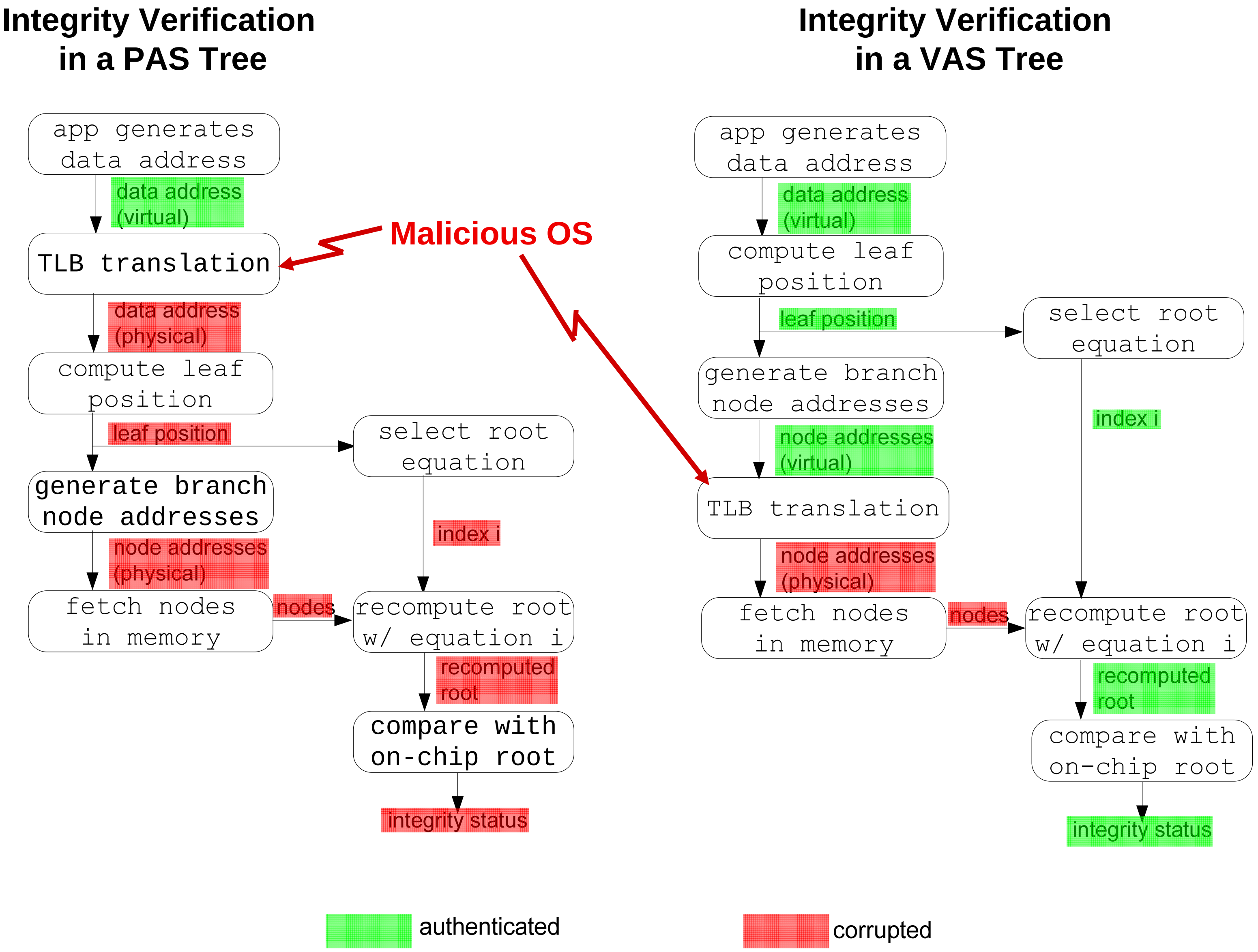
On-Chip Root Recomputation Example 2/2





🟧 X	recomputed on-chip
🟩 X	fetched from memory
▣ X	leaf to verify (fetched from memory)

The Branch Splicing Attack



Integrity Verification in a PAS Tree

app generates data address

data address (virtual)

TLB translation

data address (physical)

compute leaf position

leaf position

select root equation

generate branch node addresses

node addresses (physical)

index i

fetch nodes in memory

nodes

recompute root w/ equation i

recomputed root

compare with on-chip root

integrity status

Malicious OS

Integrity Verification in a VAS Tree

app generates data address

data address (virtual)

compute leaf position

leaf position

select root equation

generate branch node addresses

index i

node addresses (virtual)

TLB translation

node addresses (physical)

fetch nodes in memory

nodes

recompute root w/ equation i

recomputed root

compare with on-chip root

integrity status

🟩 authenticated

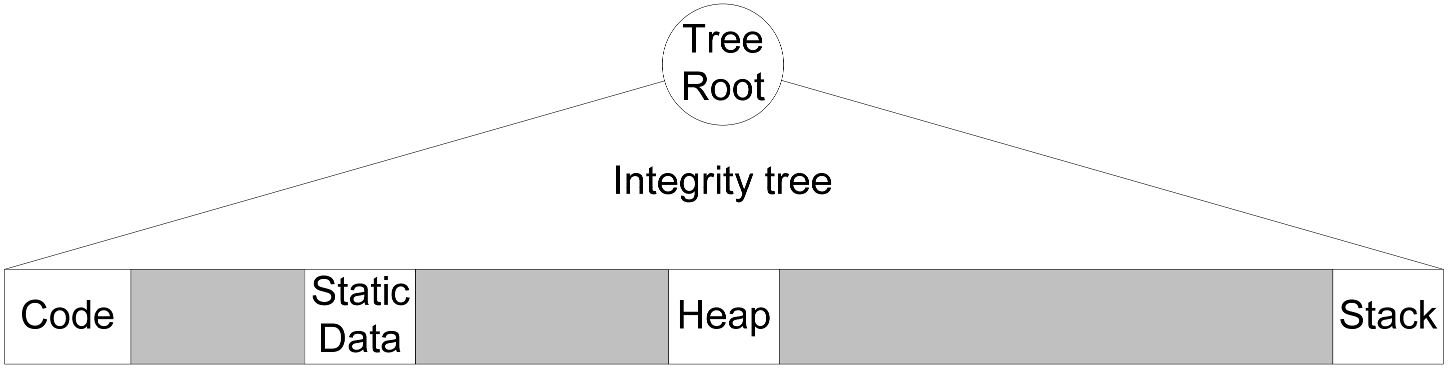
🟥 corrupted

Outline

- Introduction to memory integrity trees
- Past Work:
  - ✓ Building a tree over the physical address space (PAS Tree)
  - ✓ Building a tree over the virtual address space (VAS Tree)
- Proposed Approach
  - ✓ A novel Reduced Address Space (RAS)
  - ✓ Building a tree over the RAS (RAS Tree)
  - ✓ Managing the RAS Tree with the Tree Management Unit (TMU)
  - ✓ Performance evaluation
- Conclusion

VAS-Tree Traversal Issues

- When OS is untrusted, VAS trees are implemented:
  - Tree nodes form a contiguous memory region in the VAS
- Problem: Tree must cover huge segment of memory space



▨ = Unused memory region

This leads to huge integrity trees:

- Very large memory capacity overhead
- Very large initialization latencies

Tree span	32-bit	40-bit	48-bit	56-bit	64-bit
Total memory footprint (B)	5.7×10^{9}	1.5×10^{12}	3.8×10^{14}	9.6×10^{16}	2.5×10^{19}
Initialization latency (cycles)	7.4×10^{10}	1.4×10^{14}	3.5×10^{16}	8.9×10^{18}	2.3×10^{21}

4-ary hash tree @ 2GHz

Outline

- Introduction to memory integrity trees
- Past Work:
  - ✓ Building a tree over the physical address space (PAS Tree)
  - ✓ Building a tree over the virtual address space (VAS Tree)
- Proposed Approach
  - ✓ A novel Reduced Address Space (RAS)
  - ✓ Building a tree over the RAS (RAS Tree)
  - ✓ Managing the RAS Tree with the Tree Management Unit (TMU)
  - ✓ Performance evaluation
- Conclusion

Proposed Approach

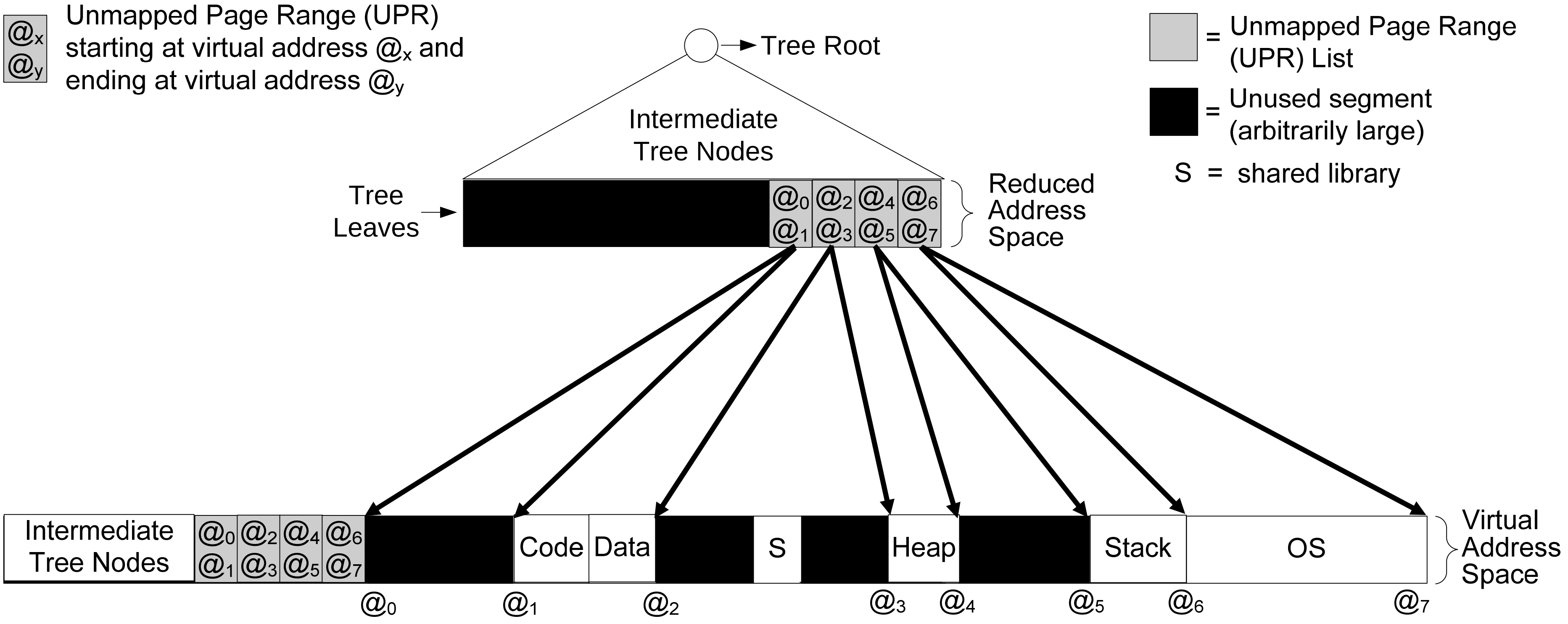
- The Reduced Address Space (RAS): a novel address space containing only pages necessary to the application's execution
  - RAS expands dynamically to fit the application's memory needs
  - RAS contains compact descriptions of memory regions not mapped in RAS, the *Unmapped Page Ranges (UPRs)*

- Compute integrity tree over RAS (a RAS tree) for dramatic reduction of memory and initialization overheads

- When application touches a previously unused page, on-chip logic expands RAS and adds branch to RAS tree

The Reduced Address Space (RAS)

- RAS initially contains the application image authenticated at load-time.
- Page mapped into RAS when application touches it for the first time.
- Tree is built over RAS so span follows the execution of the application.
- This selective tree coverage allows dramatic reduction of all overheads.



$@_x$ $@_y$ Unmapped Page Range (UPR) starting at virtual address $@_x$ and ending at virtual address $@_y$

Tree Root

Intermediate Tree Nodes

Tree Leaves

$@_0$ $@_2$ $@_4$ $@_6$
$@_1$ $@_3$ $@_5$ $@_7$

Reduced Address Space

= Unmapped Page Range (UPR) List

= Unused segment (arbitrarily large)

S = shared library

Intermediate Tree Nodes

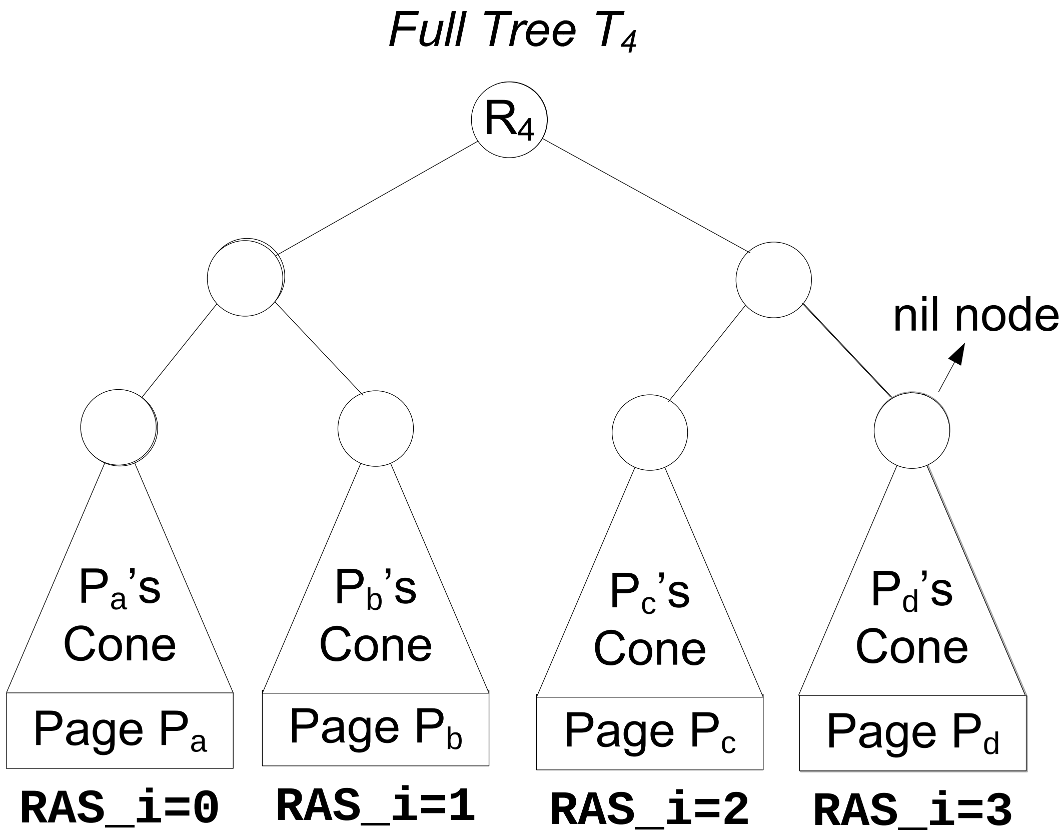
$@_0$ $@_2$ $@_4$ $@_6$
$@_1$ $@_3$ $@_5$ $@_7$

Code | Data | S | Heap | Stack | OS

$@_0$ $@_1$ $@_2$ $@_3$ $@_4$ $@_5$ $@_6$ $@_7$

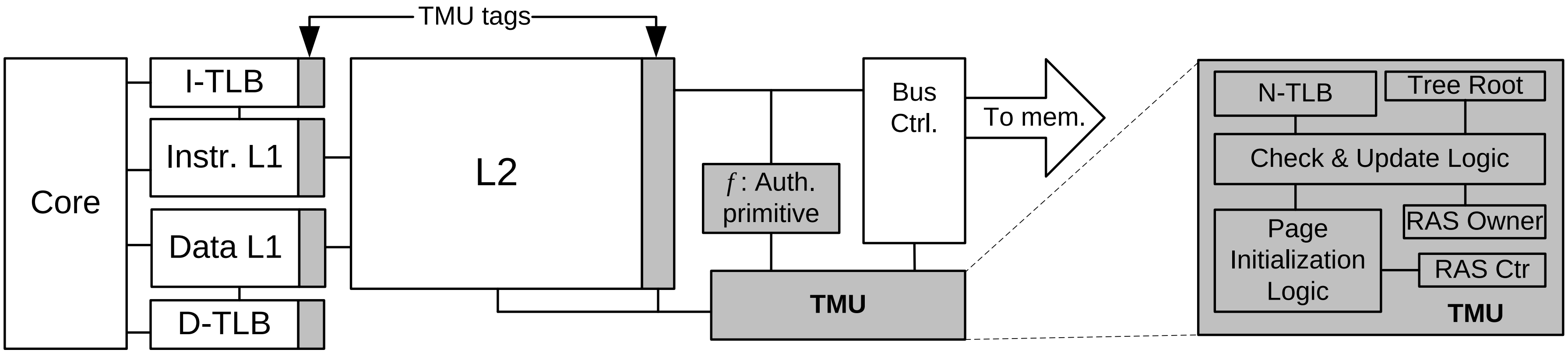
Virtual Address Space

Tree Expansion



TMU: Tree Management Unit

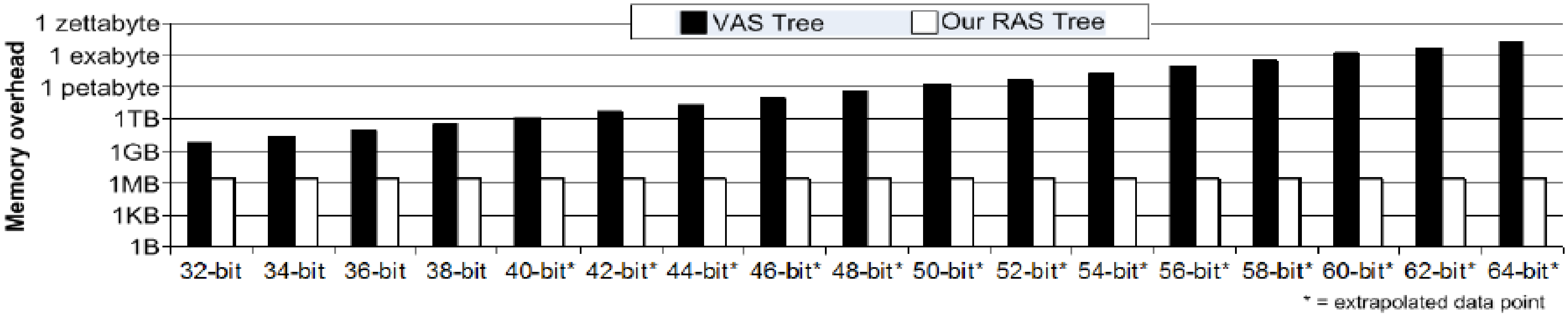
- TMU: Architectural Support for building a Tree over a RAS:
  - New TLB fields or TMU fields: RAS index (20 bits), Excluded bit (E) and the Mapped bit (M).
  - A TLB for tree nodes
- Authentication primitive and Check/Update logic



addr = *RAS_index* || *offset*,

Performance Evaluation

IPC hit less than 5% over no integrity tree and 2.5% over cached Merkle Tree.



* = extrapolated data point

Fig. 10. Comparisons of tree overheads for different memory spans for `gcc`

Conclusions

- We can provide application memory authentication despite an untrusted OS
- We reduced **memory capacity** overhead by 3 orders of magnitude on average
- We reduced CPU time overhead for **initialization** by 3 orders of magnitude on average

References

[D. Champagne, R. Elbaz et al.] “The Reduced Address Space (RAS) for Application Memory Authentication” In Proceedings of the 11th Information Security Conference (ISC’08), September 2008.

[R. Elbaz, D. Champagne et al.] “TEC-Tree: A Low Cost and Parallelizable Tree for Efficient Defense against Memory Replay Attacks,” Cryptographic Hardware and embedded systems (CHES), September 2007.

[B. Gassend et al.] “Caches and Merkle Trees for Efficient Memory Authentication,” High Performance Computer Architecture (HPCA-9), February 2003.

[R. Merkle] “Protocols for Public Key Cryptosystems,” IEEE Symposium on Security and Privacy, 1980.

[G. E. Suh et al.] “AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing,” Proc. of the 17th Int’l Conf. on Supercomputing (ICS), 2003.

[C. Yan et al.] “Improving Cost, Performance, and Security of Memory Encryption and Authentication”, Int’l Symposium on Computer Architecture (ISCA’06), June 2006.

BACKUP SLIDES

Root Recomputation Equations

Leaf Position to Verify	Root Recomputation Performed by On-chip Authentication Engine
1	$COMP1 = h\{\ h[\ h(\boxed{D1} \Vert D2) \Vert H4\ ] \Vert H2\ \}$
2	$COMP2 = h\{\ h[\ h(D1 \Vert \boxed{D2}) \Vert H4\ ] \Vert H2\ \}$
3	$COMP3 = h\{\ h[\ H3 \Vert h(\boxed{D3} \Vert D4)\ ] \Vert H2\ \}$
4	$COMP4 = h\{\ h[\ H3 \Vert h(D3 \Vert \boxed{D4})\ ] \Vert H2\ \}$
5	$COMP5 = h\{\ H1 \Vert h[\ h(\boxed{D5} \Vert D6) \Vert H6\ ]\ \}$
6	$COMP6 = h\{\ H1 \Vert h[\ h(D5 \Vert \boxed{D6}) \Vert H6\ ]\ \}$
7	$COMP7 = h\{\ H1 \Vert h[\ H5 \Vert h(\boxed{D7} \Vert D8)\ ]\ \}$
8	$COMP8 = h\{\ H1 \Vert h[\ H5 \Vert h(D7 \Vert \boxed{D8})\ ]\ \}$

🟧 X recomputed on-chip

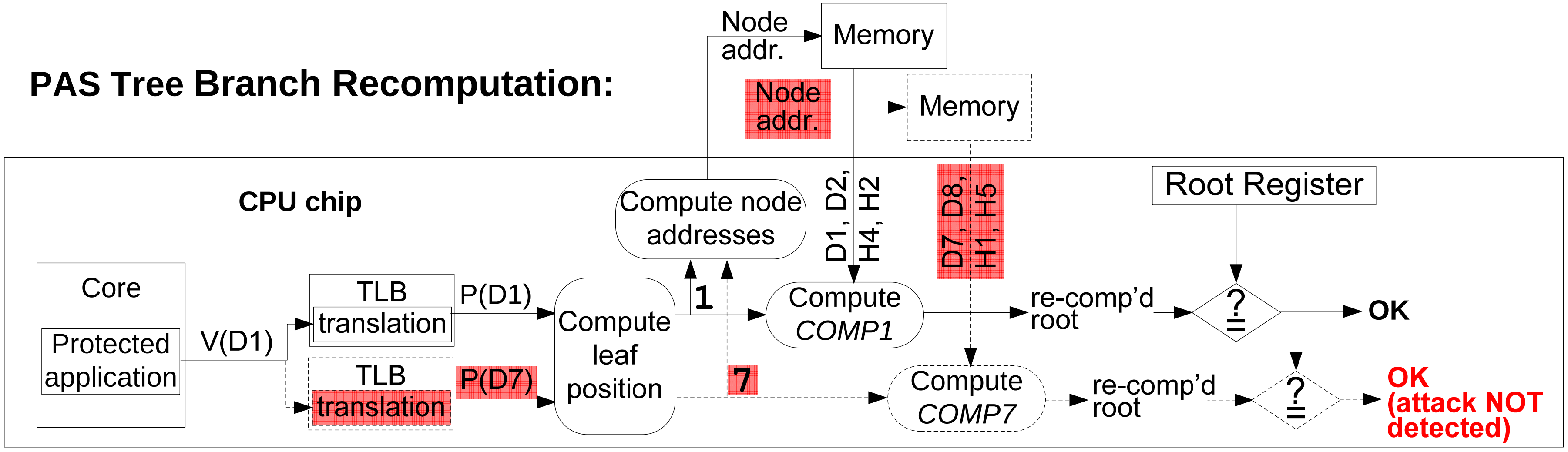
🟩 X fetched from memory

$\boxed{X}$ leaf to verify (fetched from memory)

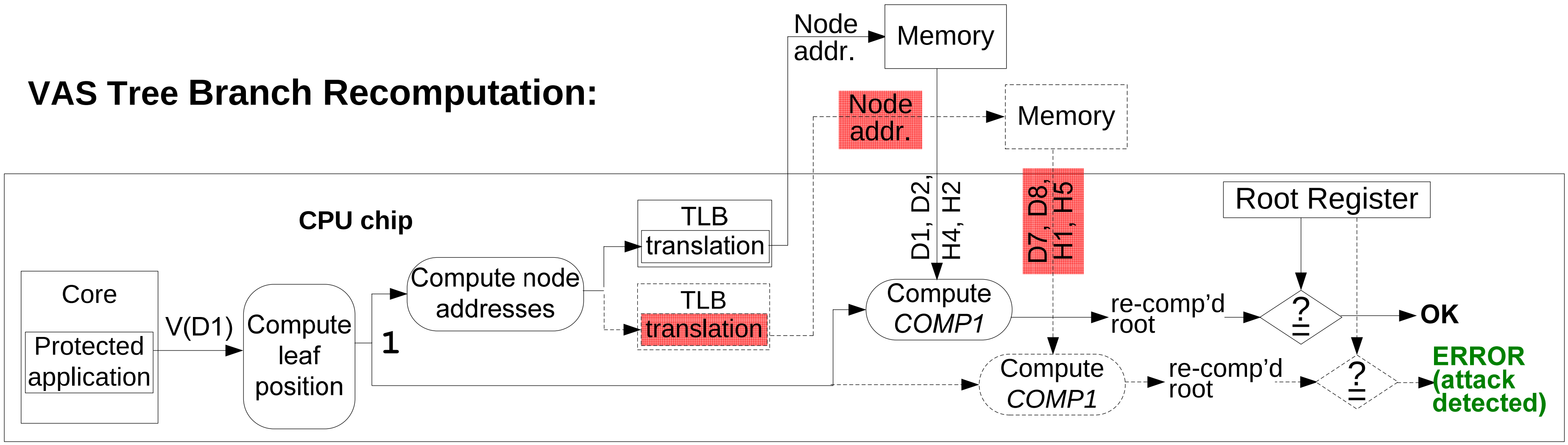
COMPi = root re**COMP**utation equation for leaf position i

Branch Splicing Attack

PAS Tree Branch Recomputation:



VAS Tree Branch Recomputation:



──► Normal verification flow

- - -► Verification flow under attack from malicious OS

🟥 XXX Data controlled by malicious OS

P(X) = physical address of X
V(X) = virtual address of X