An Overview of Cryptographic Techniques for Memory Authentication

Reouven Elbaz and David Champagne
Department of Electrical Engineering
Princeton University, US

Princeton University

Introduction

- Computer systems contain sensitive information:
  - ✓ Private data (Photo, digital media)
  - ✓ Intellectual Property, Software…

  And execute sensitive applications:
  - ✓ Digital Right Management (DRM)
  - ✓ Distributed Computing Client and Attestation
  - ✓ Web-application (e-banking, e-commerce…)





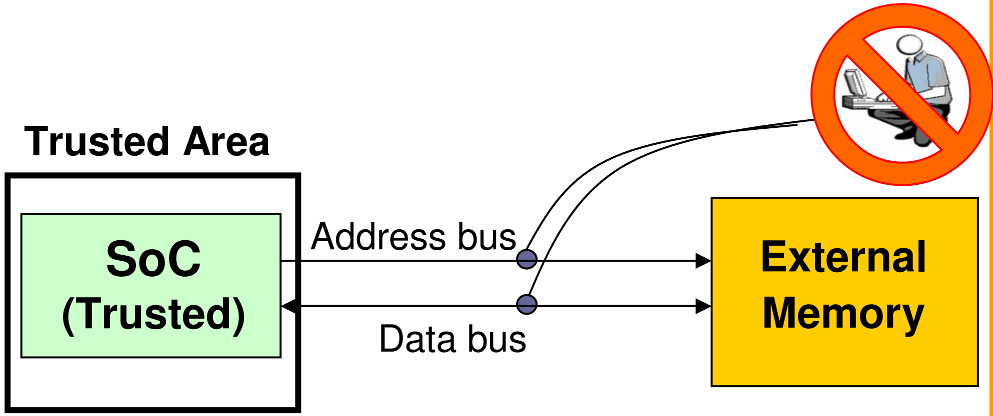
- Objectives of secure computing platforms (TPM, XOM, AEGIS, SP, SecureBlue) is to provide **trust in computations** performed by sensitive applications and to **protect private information**.

- An adversary **corrupting** the memory space through software or physical attacks can **affect the outcome** of its computations or **affect its trustworthiness**.

Motivations

- Most embedded systems and all high end systems use off-chip memories (RAM).

- Threats:
  - ✓ Code injection or data alteration
  - ✓ Memory tampering

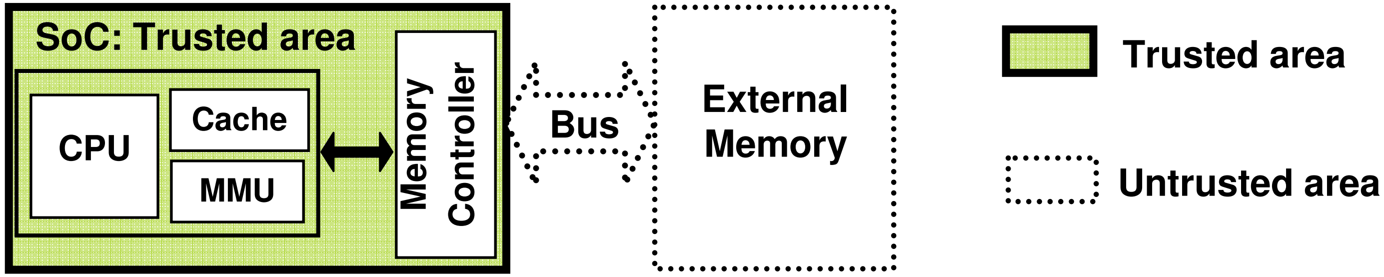


- Objectives: Provide ***integrity verification*** - i.e. ***tamper evidence*** – of data stored in off-chip memories and transferred on SoC memory interfaces

Outline

- Introduction
- Threat Model
- Authentication Primitives
- Integrity Trees
  - Generic Integrity Tree
  - Merkle Hash Trees
  - PAT: Parallelizable Authentication Tree
  - Tamper-Evident Counter Tree (TEC-Tree)
- Comparison
- Conclusion and Current Works

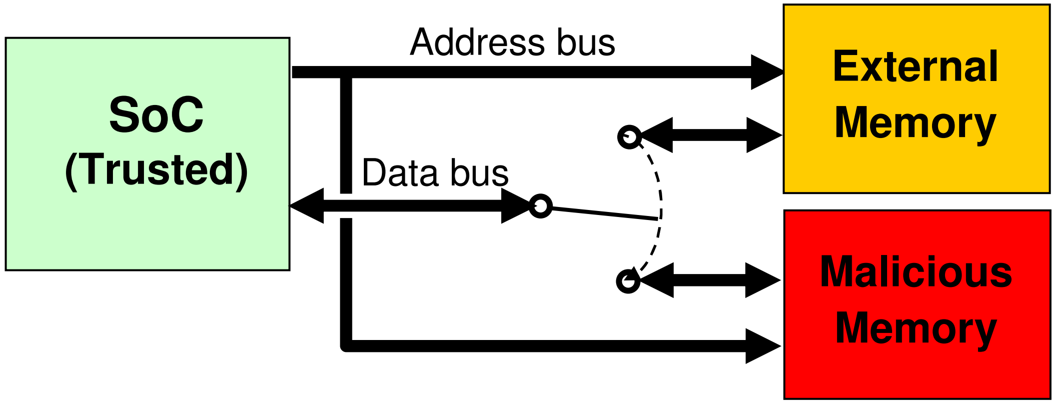
- Main hypothesis: SoC Trusted



- Attacks performed at the *board level* are considered
  - Bus probing
  - Memory tampering
- Attacks not considered:
  - Software attacks
  - Side-channel attacks
  - Invasive attacks

Active Attacks

- Code and data injection

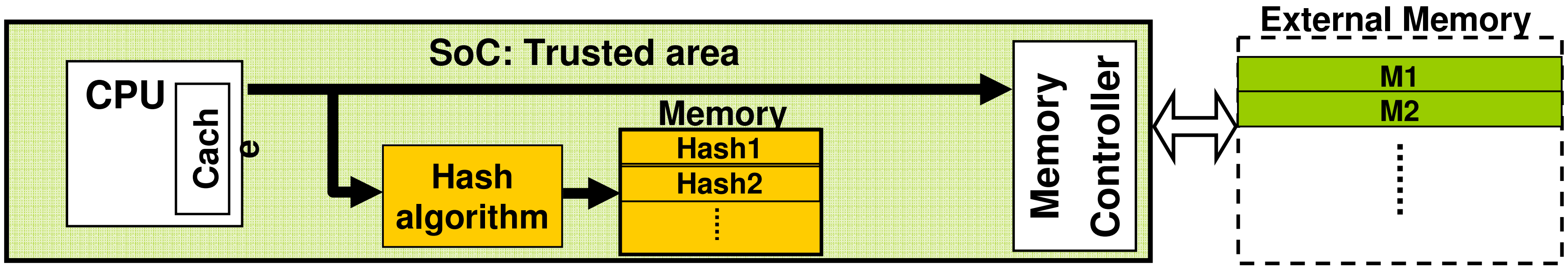


- Three kinds of active attacks are defined depending on the choice made by the adversary on the data to insert:
  - ✓ Spoofing: Random data injection
  - ✓ Splicing: Spatial permutation
  - ✓ Replay: Temporal permutation

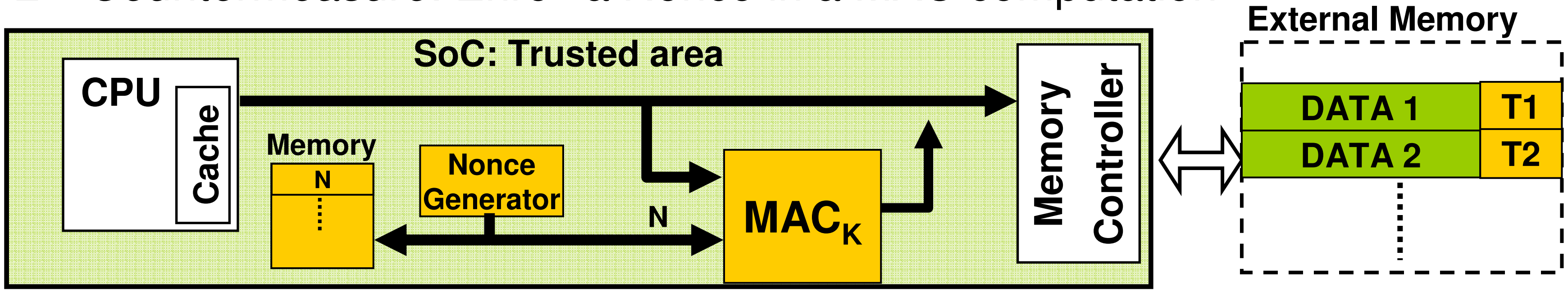
- Attacker motivation:
  - ✓ Hijack the software execution
  - ✓ Reduce the search space for key recovery or message recovery

Countermeasures

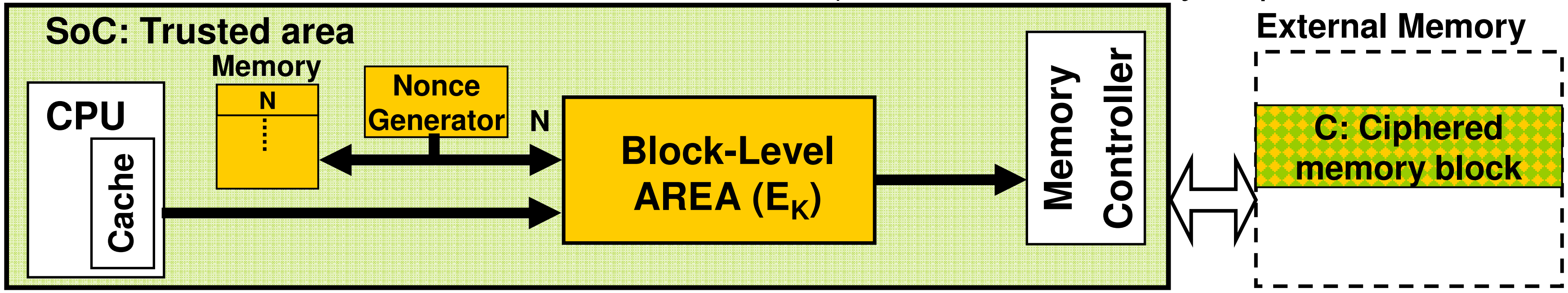
✓ *1st Countermeasure*: On-chip storage of hash values



✓ *2nd Countermeasure*: Enroll a Nonce in a MAC computation

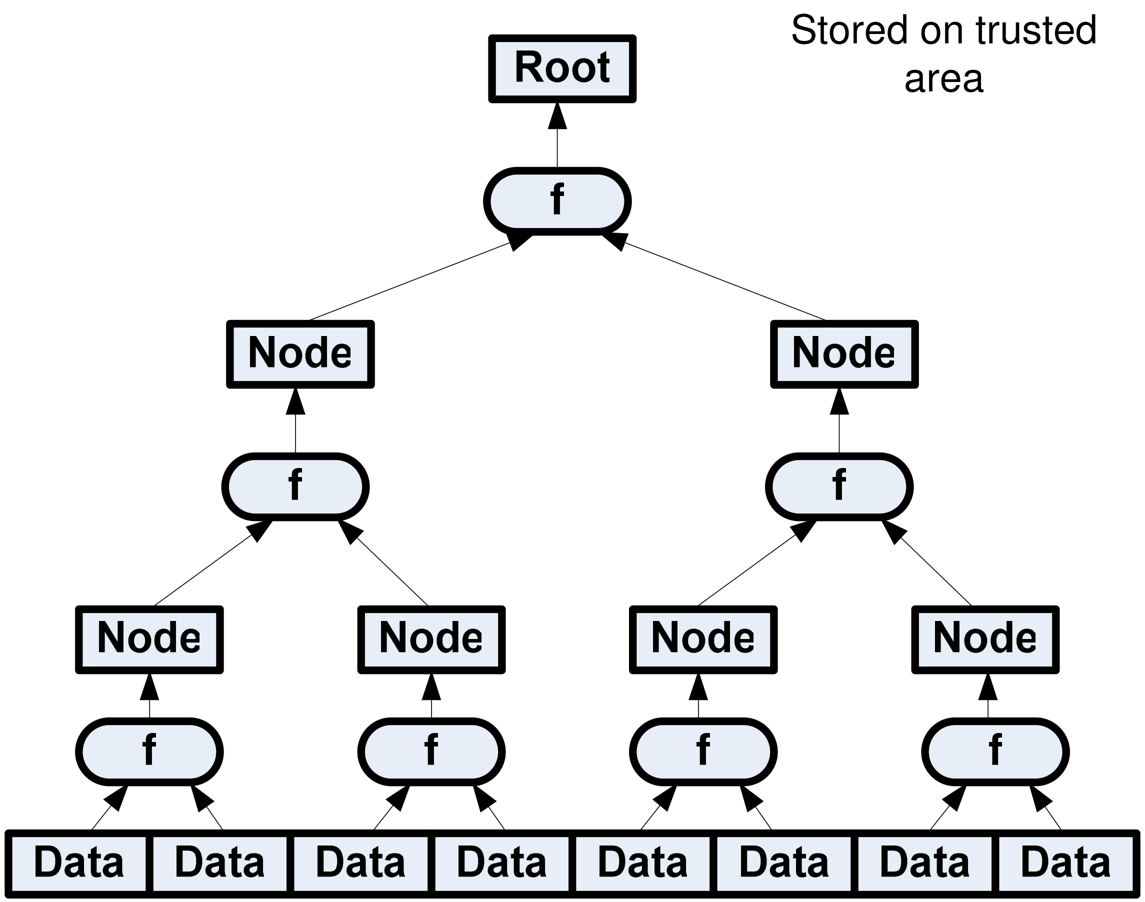


✓ *3rd Countermeasure*: Block-Level AREA (Added Redundancy Explicit Authentication)



Generic Integrity Tree

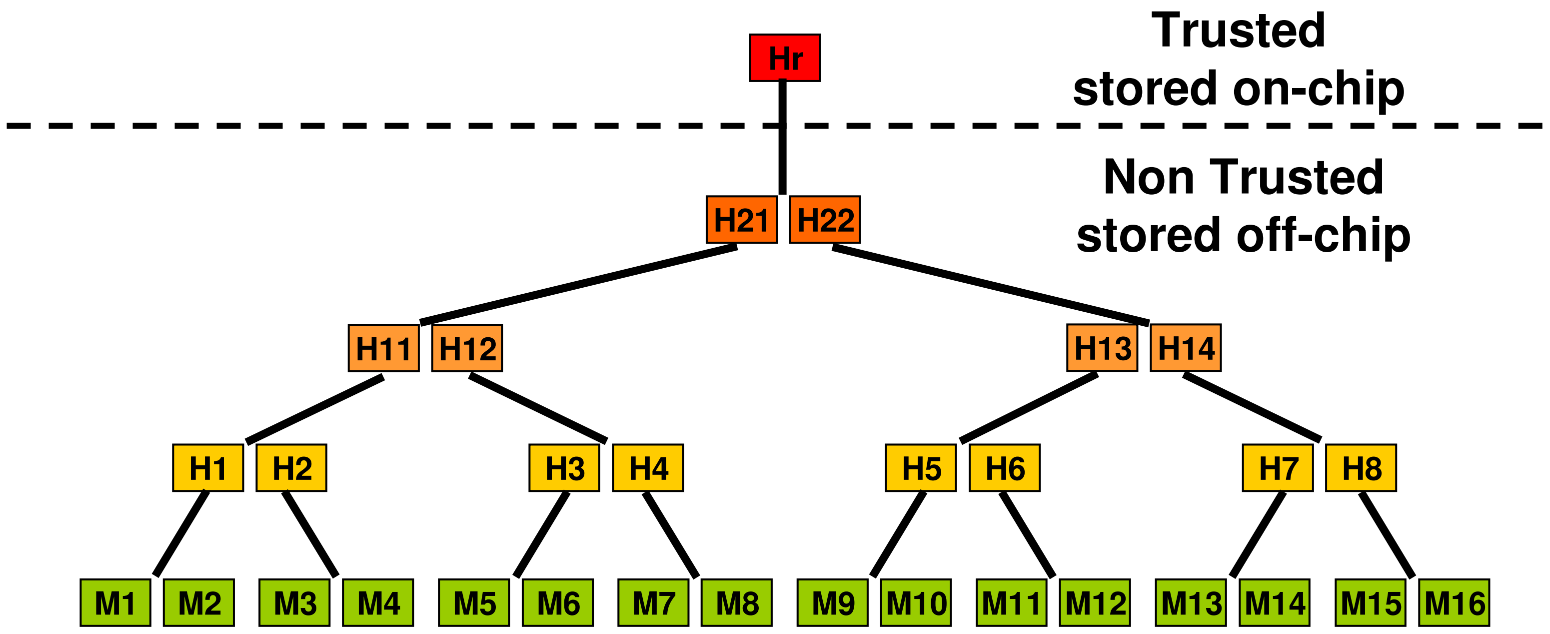
- Principle of Integrity Trees:



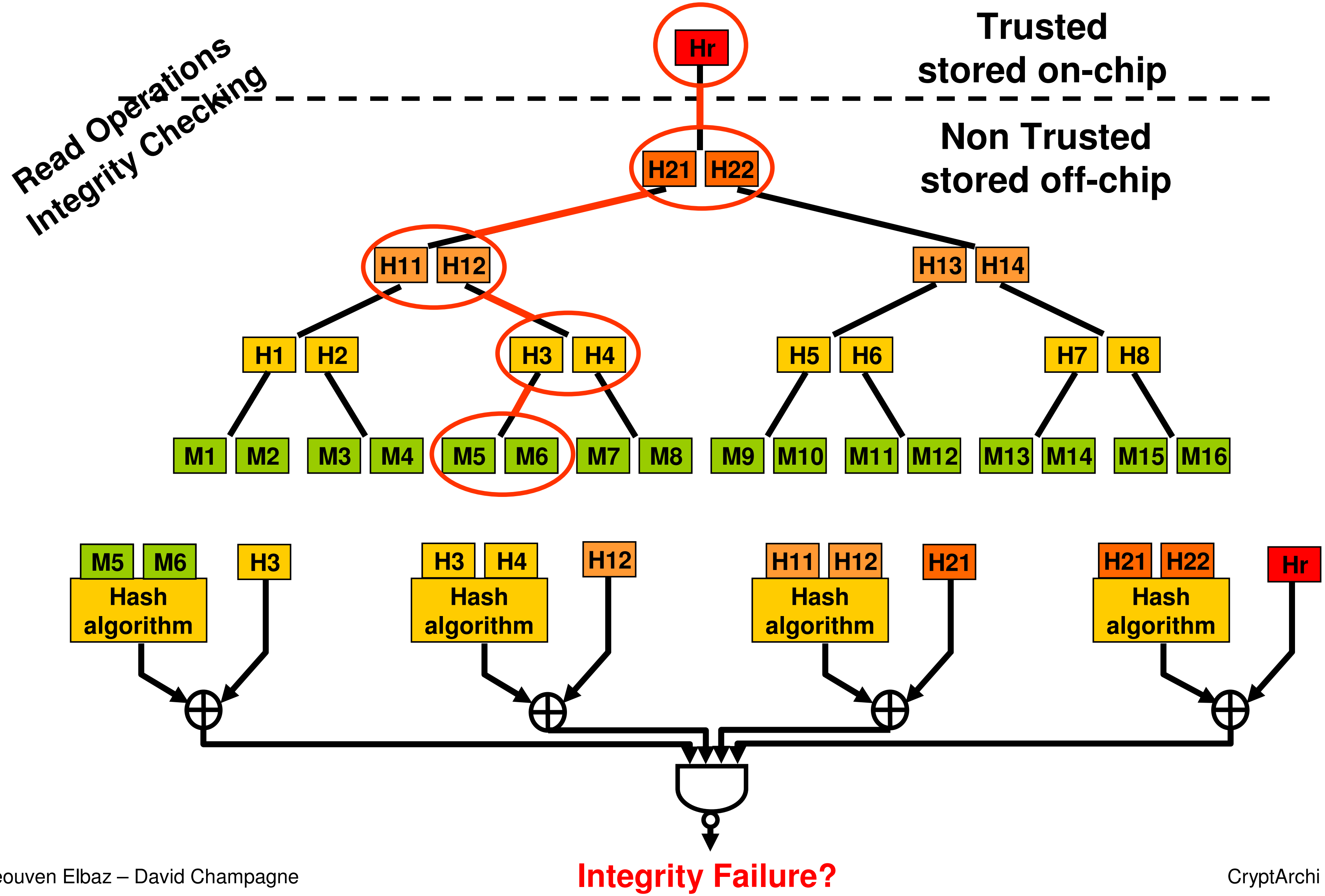
3 authentication primitives f

→ 3 Existing Trees

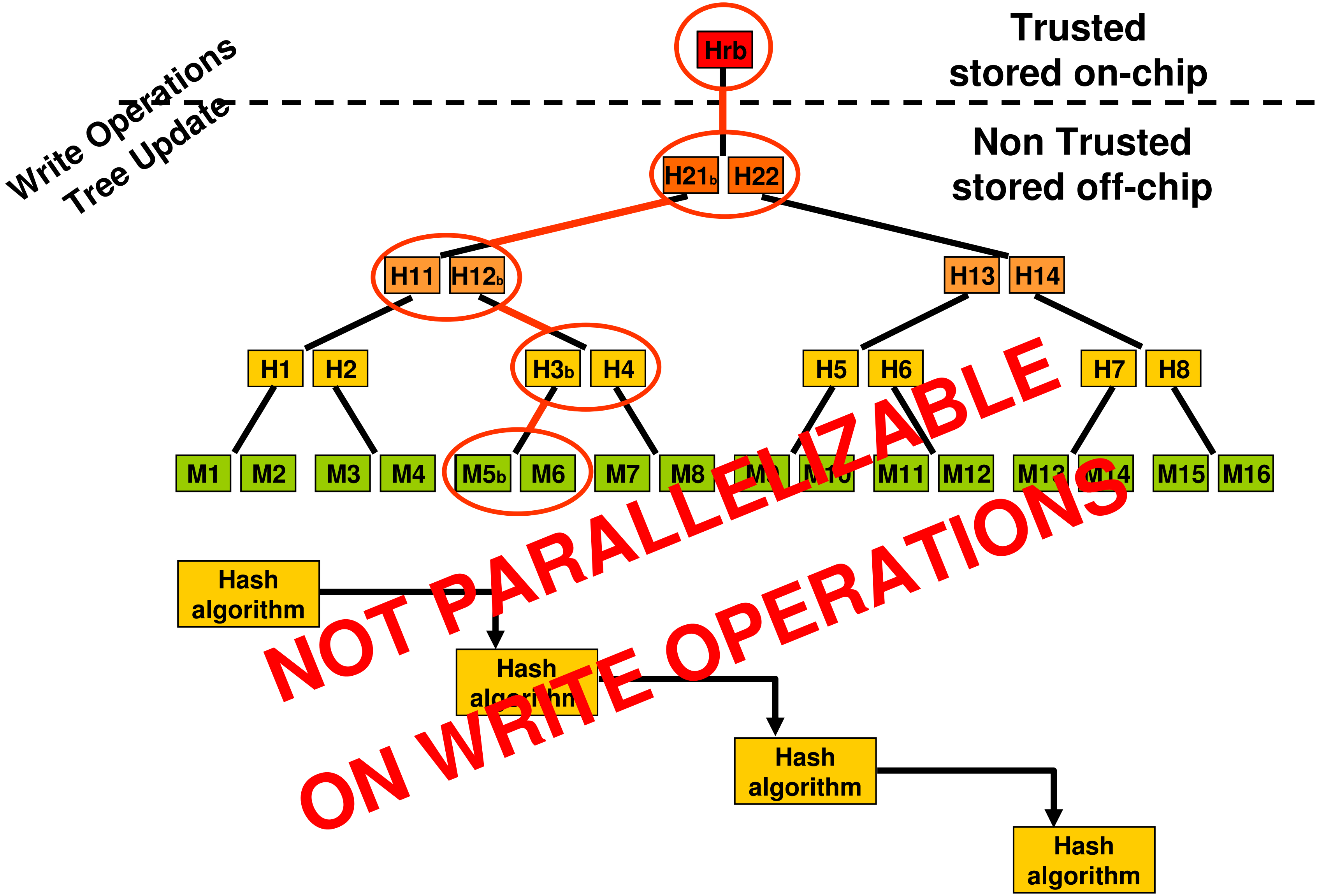
Hash Tree Structure - Initialisation



Read Operations – Integrity Checking



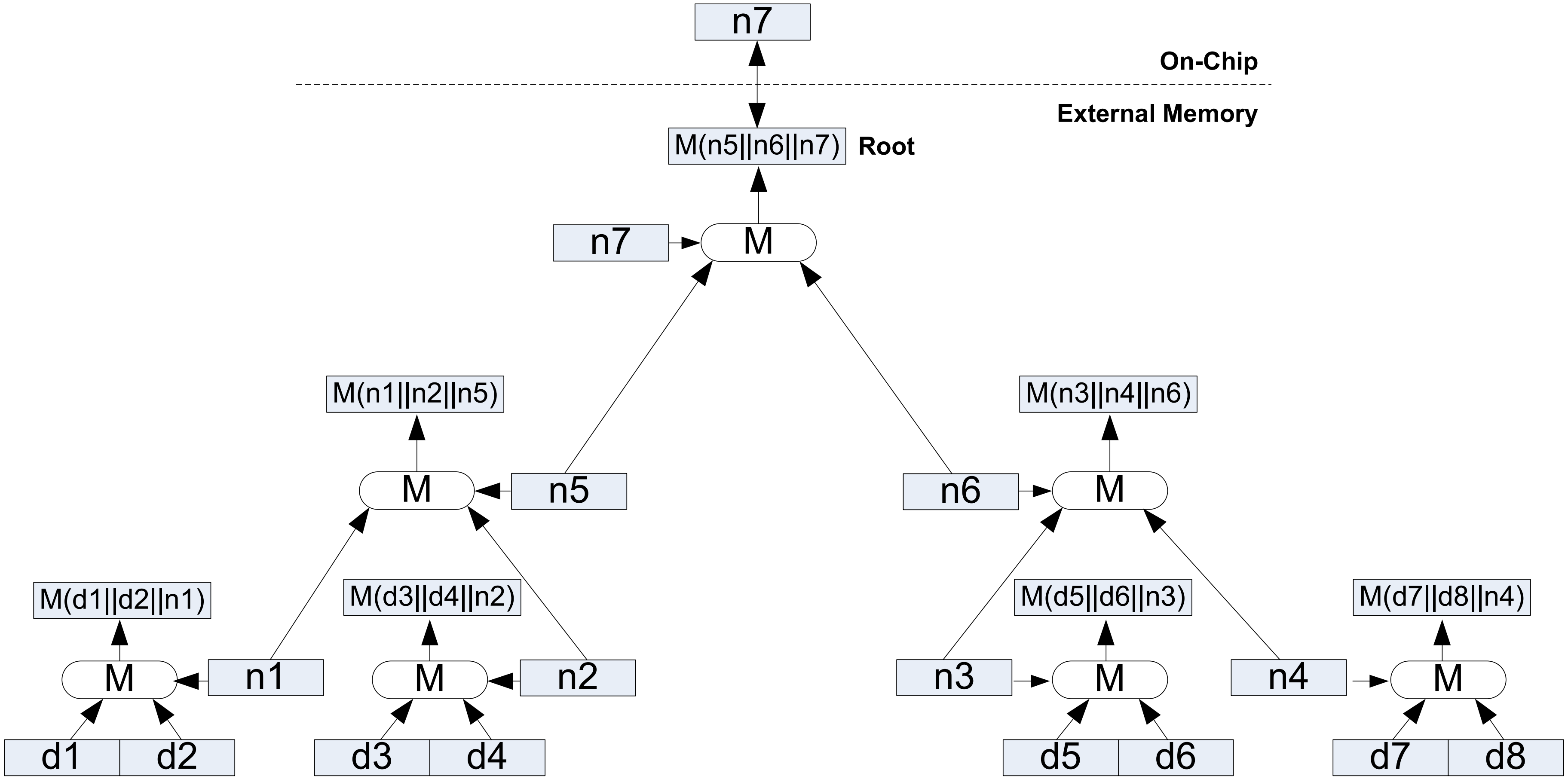
Write Operations – Tree Update



Merkle Trees

	Merkle Tree
Authentication Primitive / Reference Value	Hash Algorithms / Hash
Replay – Splicing – Spoofing Detection	Yes
Confidentiality	No
Parallelizability	Authentication process only
Detection Speed for Splicing / Spoofing	After Root-check
Off-chip memory overhead	1/A-1

PAT: Parallelizable Authentication Tree



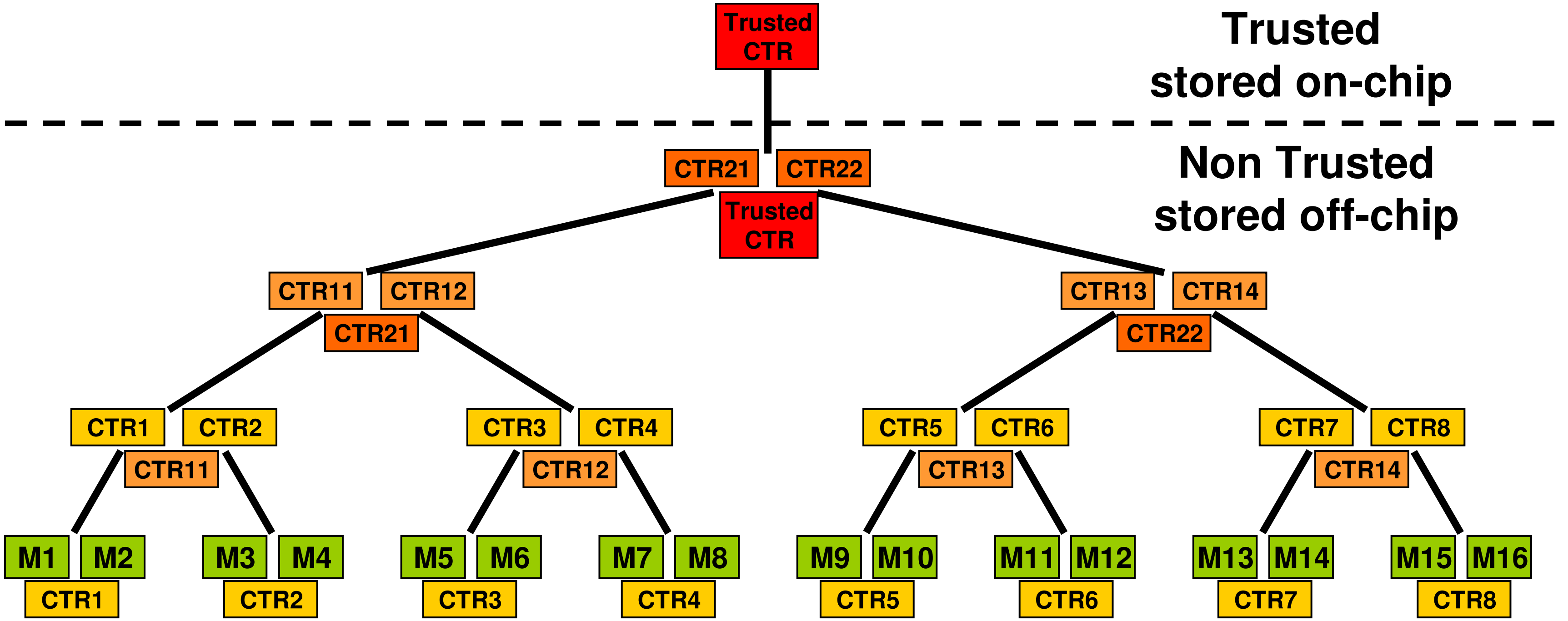
PAT: Parallelizable Authentication Tree

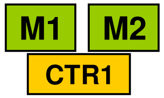
- Comparison of existing Integrity Tree

	Merkle Tree	PAT
Authentication Primitive / Reference Value	Hash Algorithms / Hash	MAC Algorithms / Nonce
Replay – Splicing – Spoofing Detection	Yes	Yes
Confidentiality	No	No
Parallelizability	Authentication process only	Authentication ***and*** Tree update
Detection Speed for Splicing / Spoofing	After Root-check	1st tree-level check / After Root-check*
Off-chip memory overhead	1/A-1	1.5/A-1

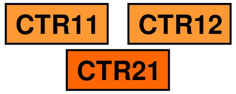
**Adding the address in the MAC computation allows for detection after first tree-level*

TEC-Tree Structure & Initialization



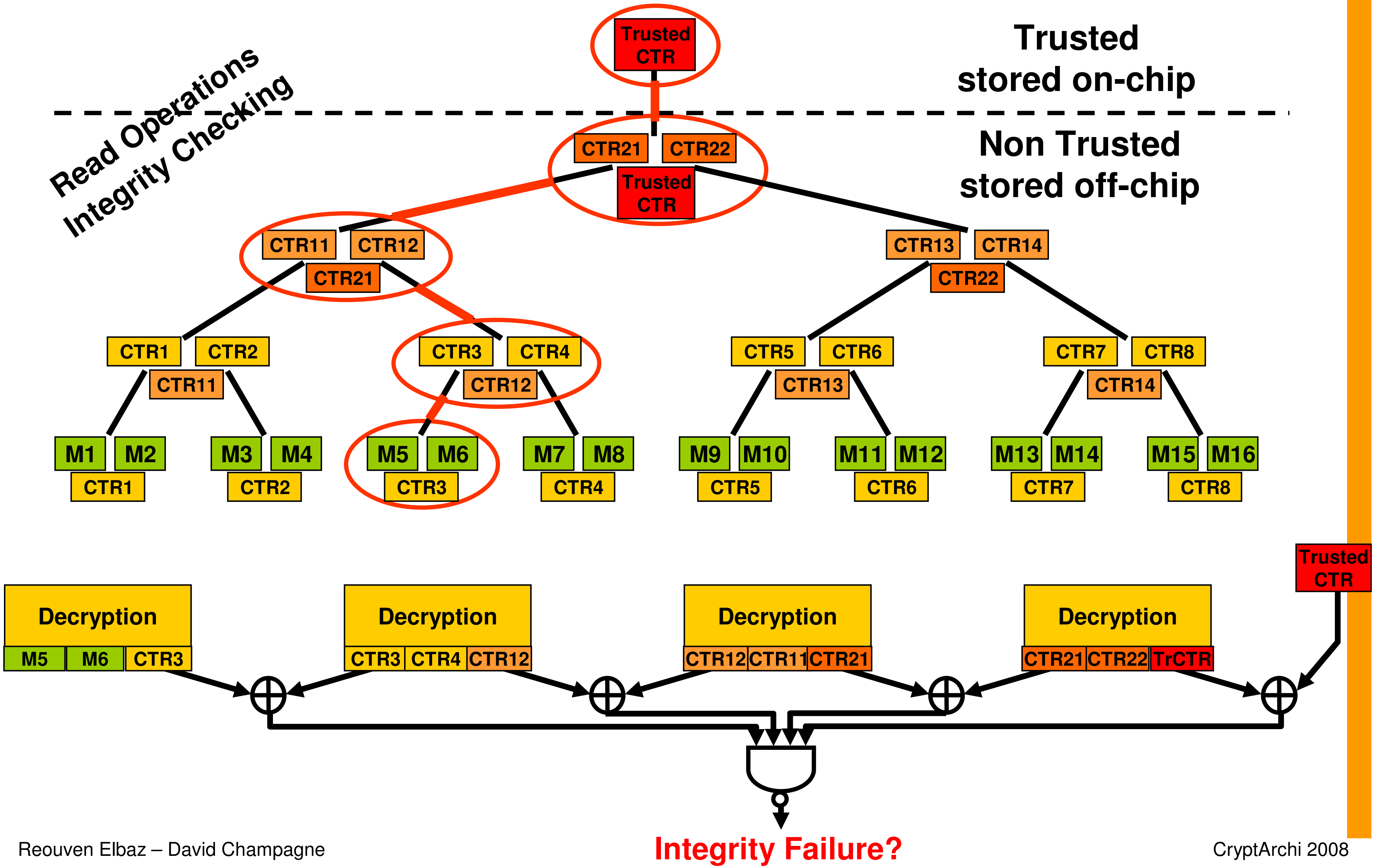


: a leaf node = encrypted 3-tuple (2 memory blocks + 1 counter in a single encrypted block)

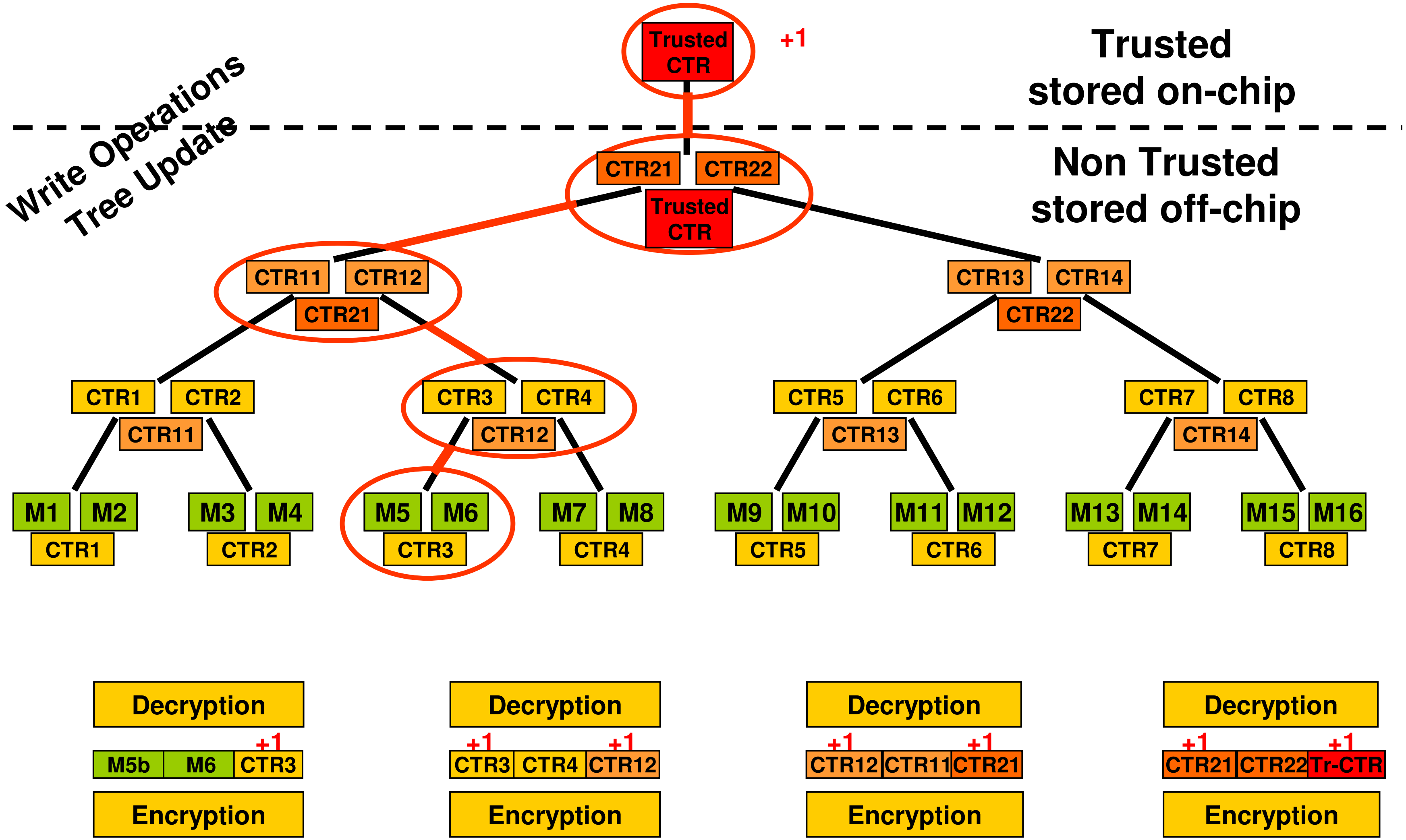


: a intermediate node = encrypted 3-tuple (3 counters in a single encrypted block)

Read Operations – Integrity Checking



Write Operations – Tree Update



Conclusion & Perspectives

Comparison of existing Integrity Tree

	Merkle Tree	**PAT**	**TEC-Tree**
Authentication Primitive / Reference Value	Hash Algorithms / Hash	MAC Algorithms / Nonce	Block Level AREA / Nonce
Replay – Splicing – Spoofing Detection	Yes	Yes	Yes
Confidentiality	No	No	Yes
Parallelizability	Authentication process only	Authentication ***and*** Tree update	Authentication ***and*** Tree update
Detection Speed for Splicing / Spoofing	After Root-check	1st tree-level check / After Root-check*	After First Tree-level check
Off-chip memory overhead	1/A-1	1.5/A-1	2/A-1

**Adding the address in the MAC computation allows for detection after first tree-level*

Conclusion & Perspectives

- Integrity trees do provide memory authentication
- The three existing schemes can be viewed as recursive applications of an authentication primitive
- The schemes have their different advantages and shortcomings (Parallelizability, Confidentiality…)

- *Related Work:* Architectural support has been proposed to enhance performance of Integrity Trees during the steady state execution of an application (Cached Hash Tree).

- *Current work:* Managing trees efficiently with an untrusted operating system

Thank You

REFERENCES

[Merkle Tree] R. C. Merkle, “Protocols for Public Key Cryptography”, IEEE Symp. on Security and Privacy, pages 122–134, 1980.

[Merkle Tree 2] M. Blum, W. Evans, P Gemmell, S. Kannan, and M. Naor, Checking the correctness of memories,Proc. 32nd IEEE Symposium on Foundations of Computer Science, pages 90–99, 1991

[PAT] W. E. Hall and C. S. Jutla. Parallelizable authentication trees. In Selected Areas in Cryptography SAC 2005: 95-109

[TEC-Tree] R. Elbaz, D. Champagne, R. B. Lee, L. Torres, G. Sassatelli and P. Guillemin, “TEC-Tree: A Low Cost and Parallelizable Tree for Efficient Defense against Memory Replay Attacks,” In Proc of the Workshop on Cryptographic Hardware and embedded systems (CHES), pp. 289-302, 2007.

[Related Work] B. Gassend, G. E. Suh, D. Clarke, M. van Dijk, and S. Devadas, “Caches and Merkle Trees for Efficient Memory Integrity Verification”, Proceedings of Ninth International Symposium on High Performance Computer Architecture, February 2003

[Current Work] D. Champagne, R. Elbaz, and R. Lee, “The Reduced Address Space (RAS) for Application Memory Authentication”, in Proceedings of the 11th Information Security Conference (ISC’08), Sept 2008.