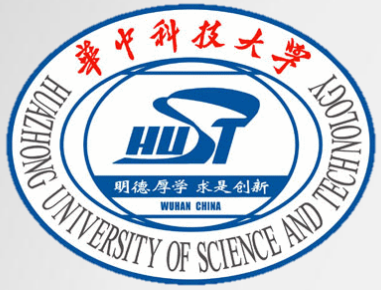


Root Crash Consistency of SGX-style Integrity Trees in Secure Non-Volatile Memory Systems

Jianming Huang, Yu Hua

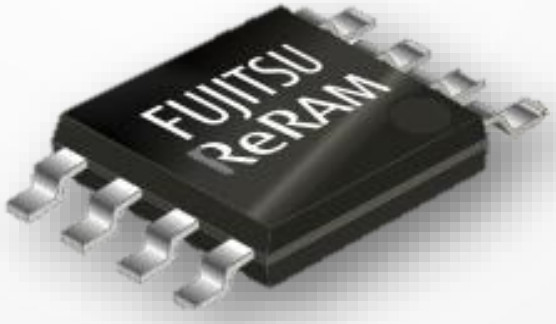
Huazhong University of Science and Technology, China

HPCA 2023

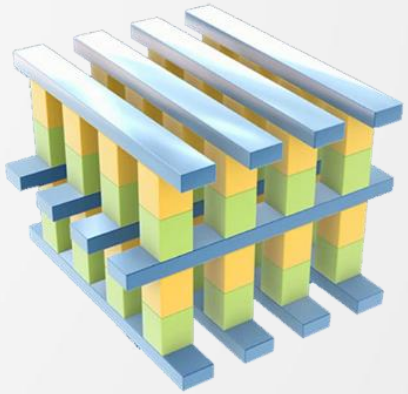
Non-volatile Memory (NVM)

- Non-volatile memory
  - ✓ Non-volatility
  - ✓ Low stand-by power consumption
  - ✓ Large capacity

PCM



ReRAM

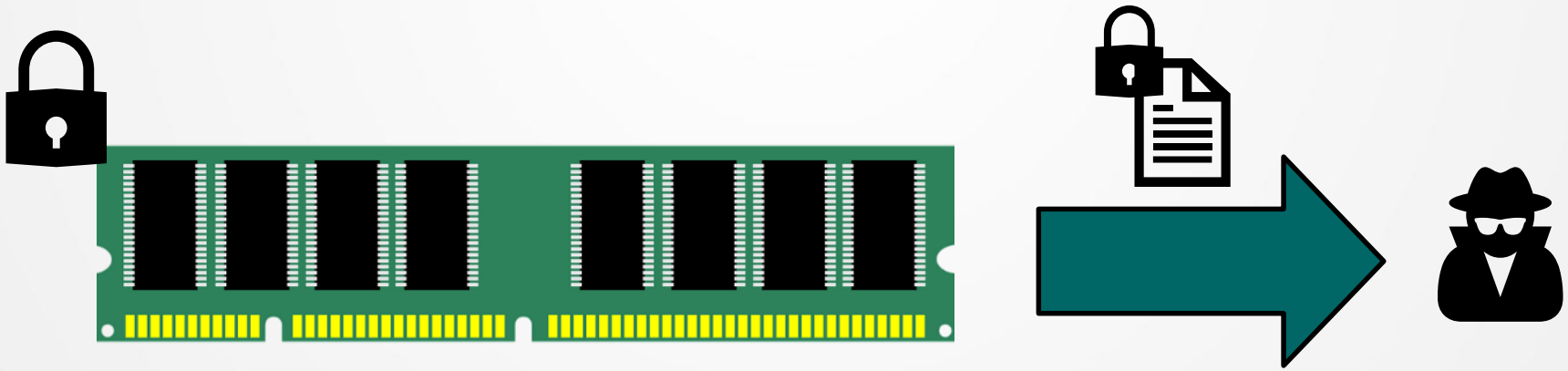


3D Xpoint

Data Security in NVM

Data security = Data confidentiality + Data integrity

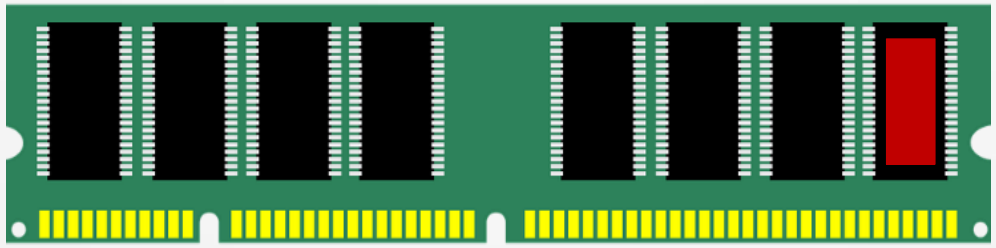
- Data confidentiality
  - Data leakage to attackers due to non-volatility
  - Encryption for data confidentiality



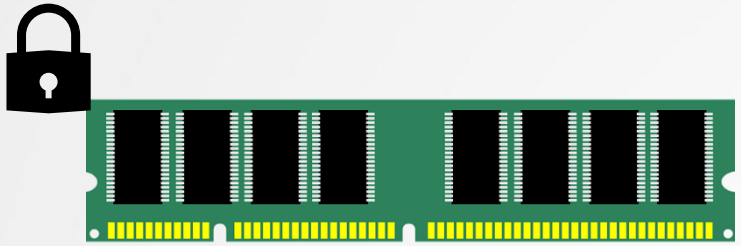
Data Security in NVM

Data security = Data confidentiality + Data integrity

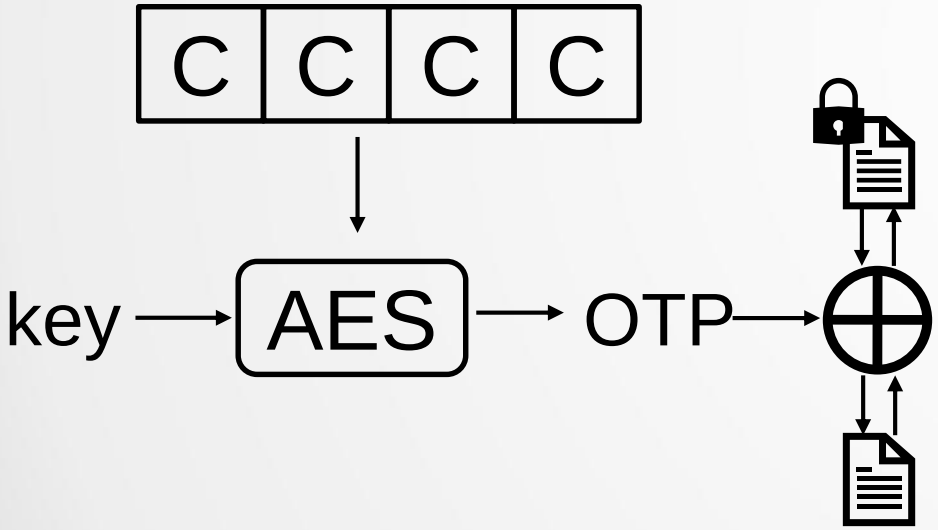
- Data confidentiality
- Data integrity
  - Data modification by attackers
  - Authentication for data integrity



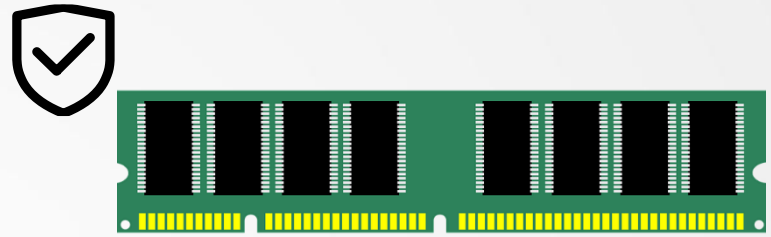
Data Security in NVM



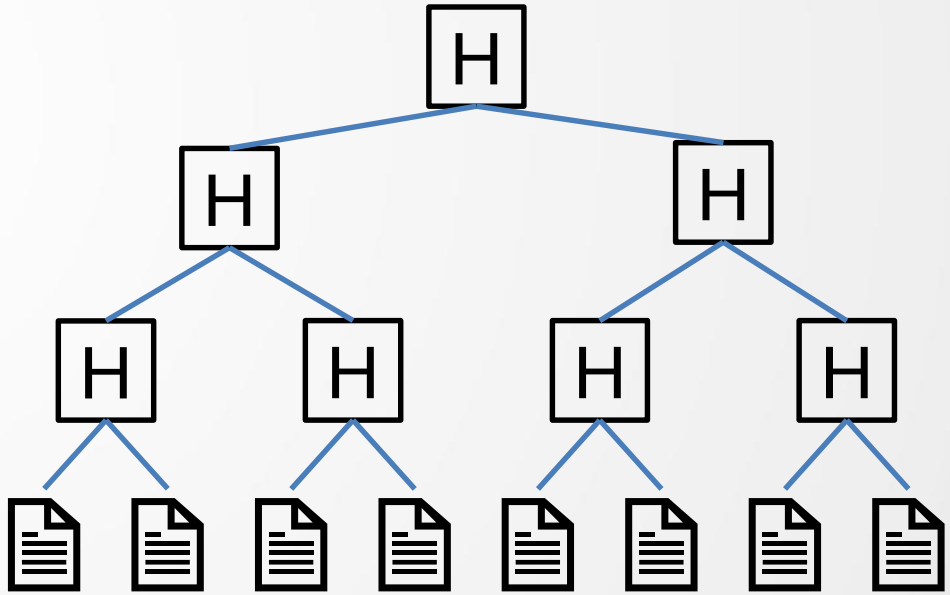
Confidentiality



Counter Mode Encryption (CME)



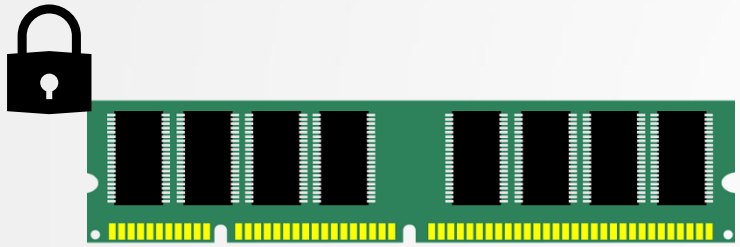
Integrity



Merkle Tree (MT)

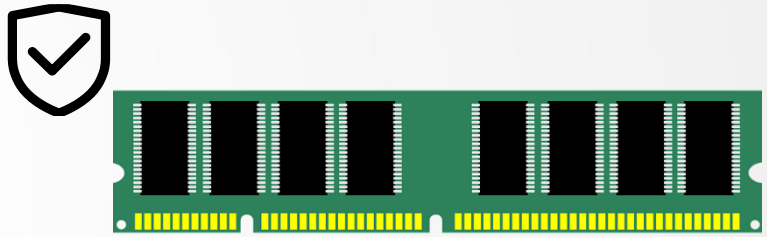
Data Security in NVM

Data Security



Confidentiality

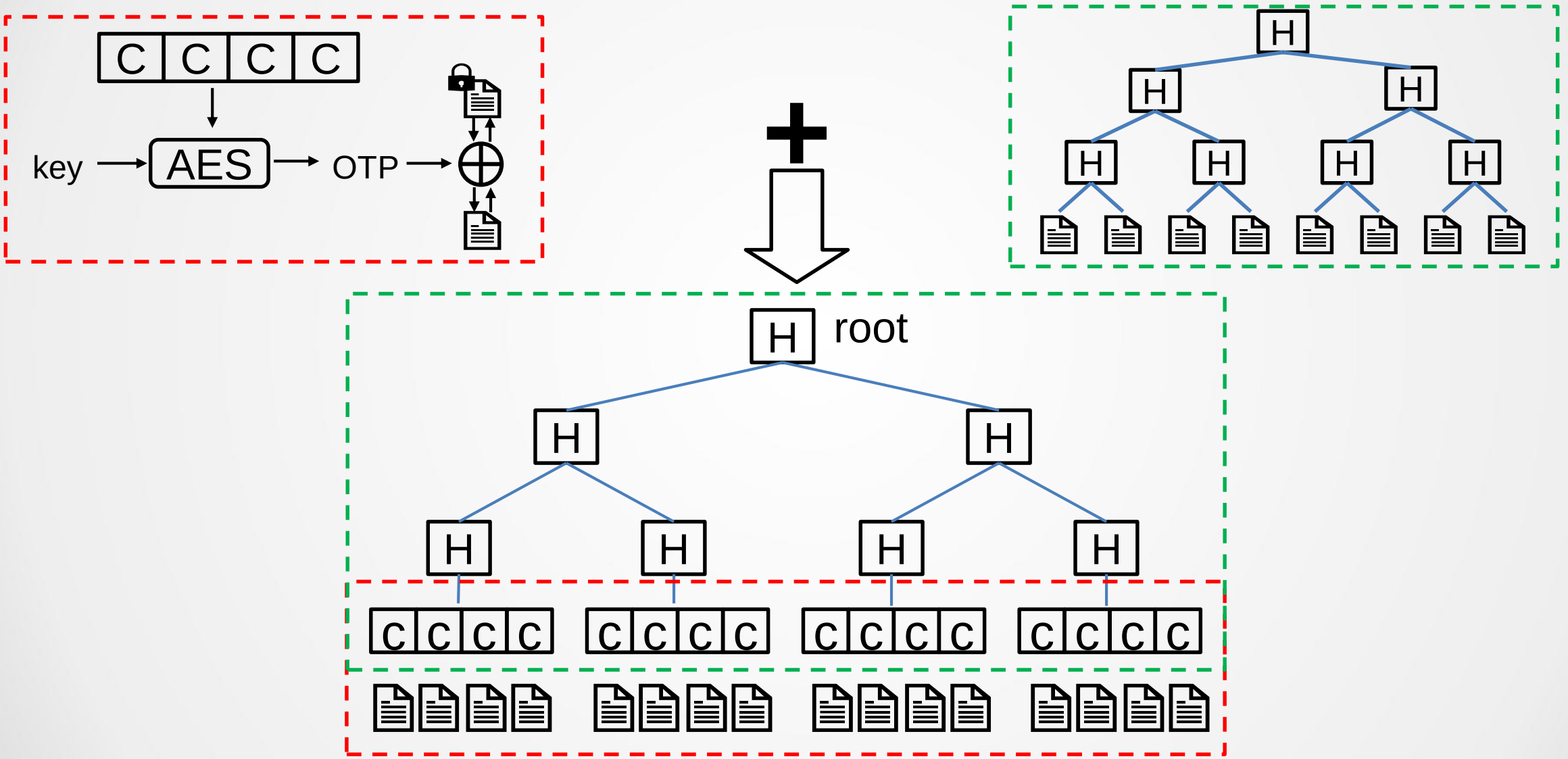
+



Integrity

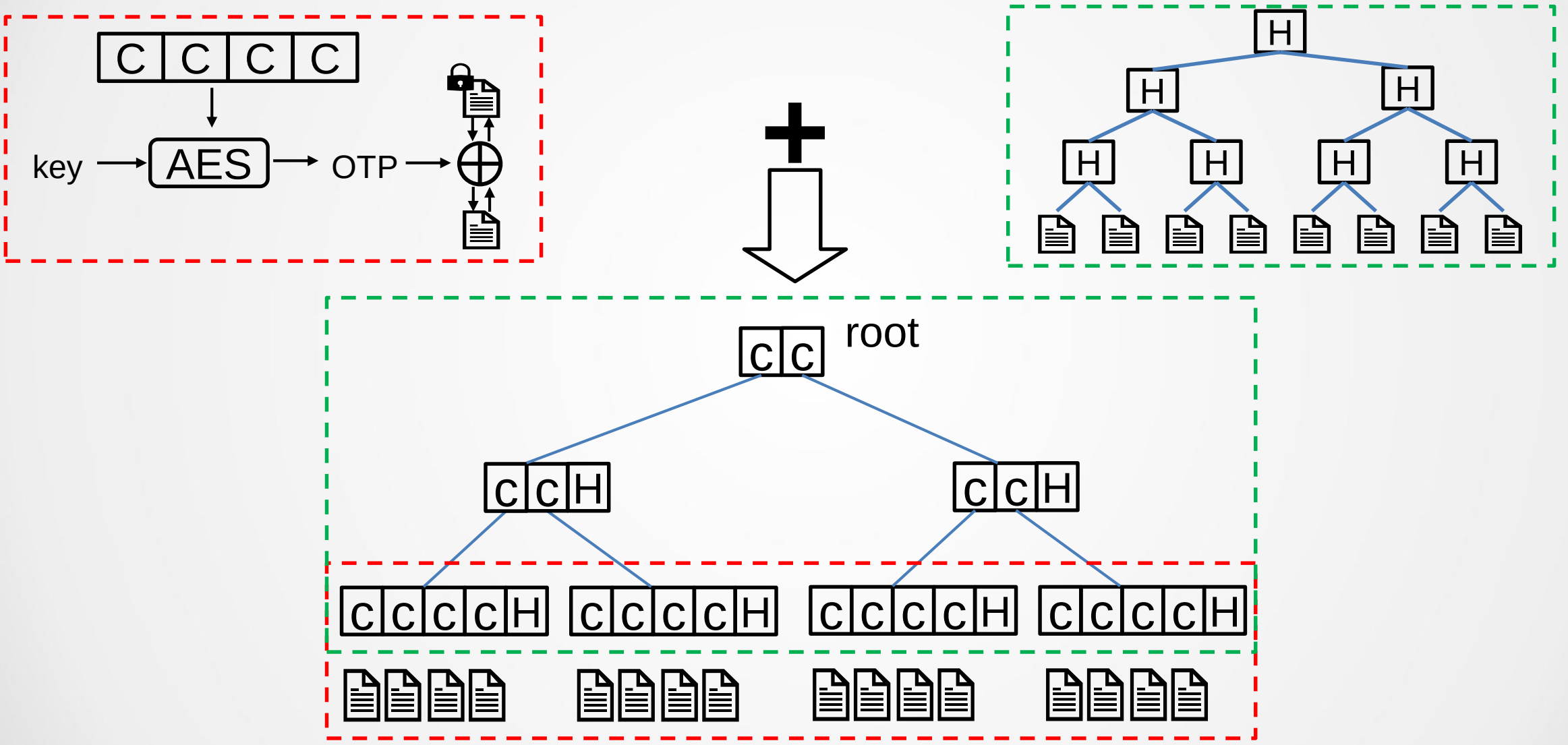
2019	2020	2021	2022
Anubis@ISCA Janus@ISCA Triad-NVM@ISCA ……	PLP@MICRO Phoenix@TDSC ……	BMF@MICRO STAR@HPCA ProMT@ICS ……	Horus@MICRO ……

Integrity Tree



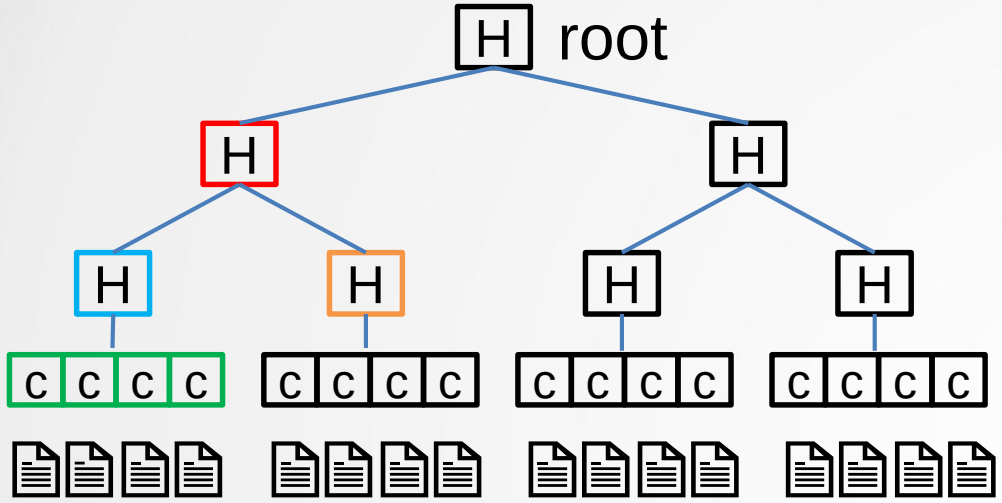
Bonsai Merkle tree (BMT)

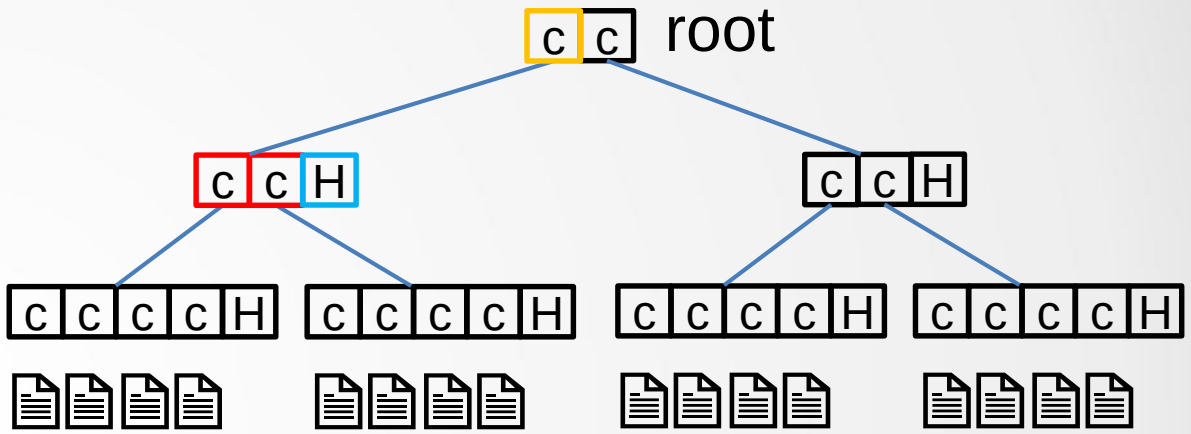
Integrity Tree



SGX-style Integrity Tree (SIT)

Integrity Tree





$H = Hash_{K}(c\,c\,c\,c)$

$H = Hash_{K}(H\;\;H)$

$H = Hash_{K}(c\,c\,|\,c)$

Bonsai Merkle tree (BMT)

SGX-style Integrity Tree* (SIT)

BMT vs SIT

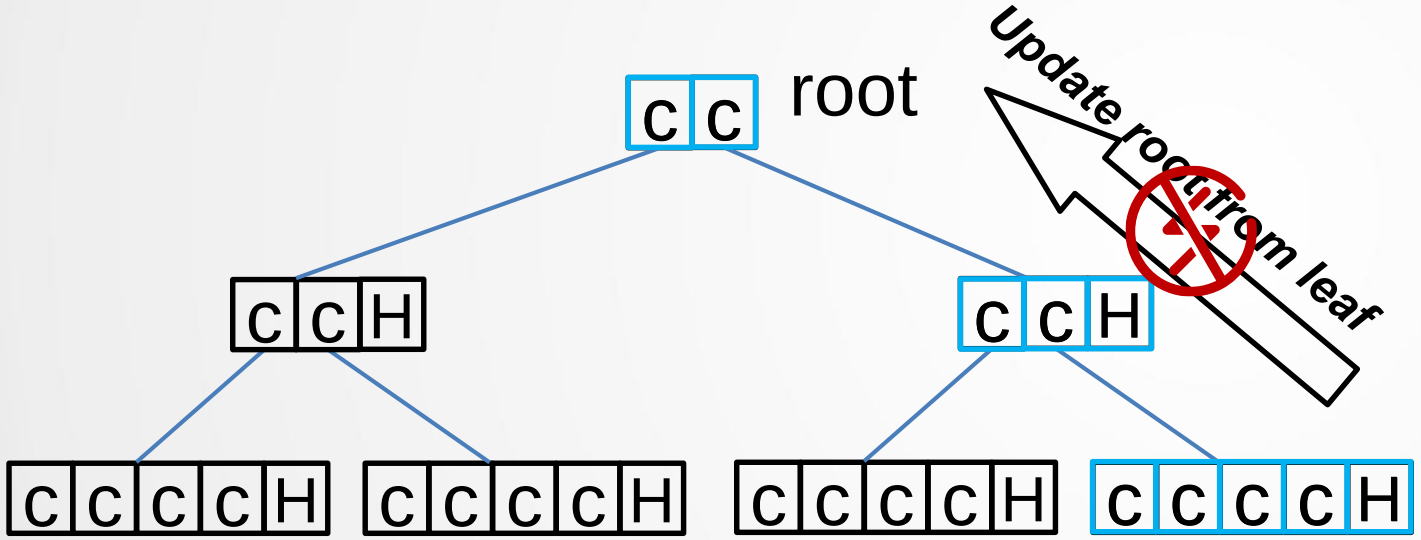
- BMT
  - ✗ Large space overhead
  - ✗ High height of tree
  - ✗ Sequential update
- SIT
  - ✓ Small space overhead[1]
  - ✓ Low height of tree[2]
  - ✓ Parallel update[3]

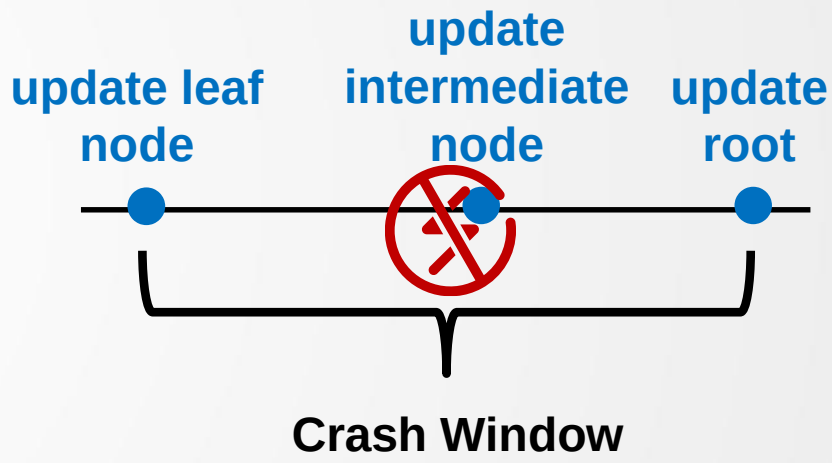
SIT provides higher security[3] than BMT, and we focus on SIT in our work

[1] G. Saileshwar, P. Nair, P. Ramrakhyani, W. Elsasser, J. Joao, and M. Qureshi, “Morphable counters: Enabling compact integrity trees for low-overhead secure memories”, MICRO18.
[2] J. Huang and Y. Hua, “A write-friendly and fast-recovery scheme for security metadata in non-volatile memories”, HPCA21
[3] M. Alwadi, K. Zubair, D. Mohaisen, and A. Awad, “Phoenix: Towards ultra-low overhead, recoverable, and persistently secure nvm”, TDSC20
.

Challenges of leveraging SIT in NVM

- **Crash inconsistency between root and leaf nodes**

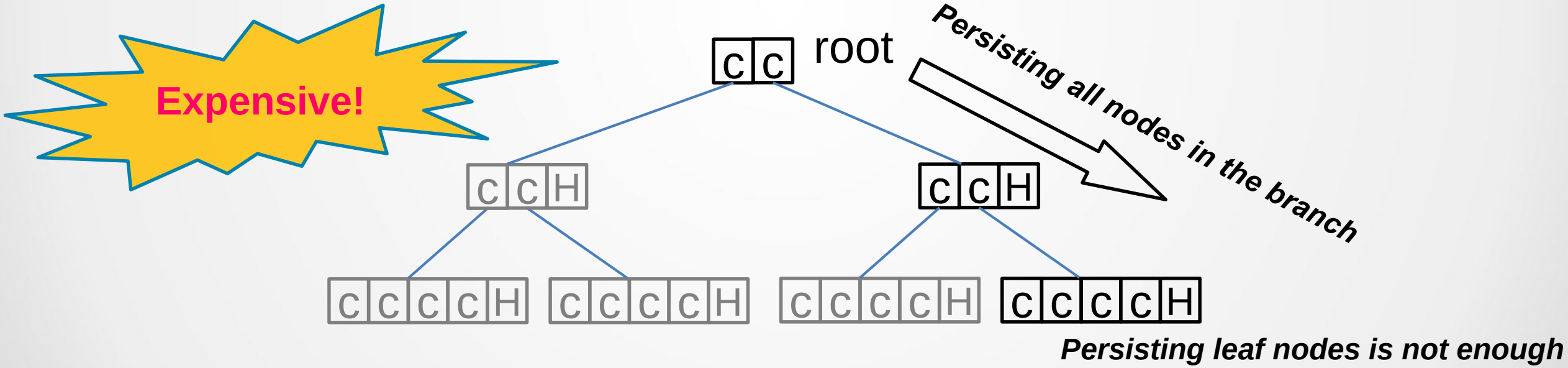




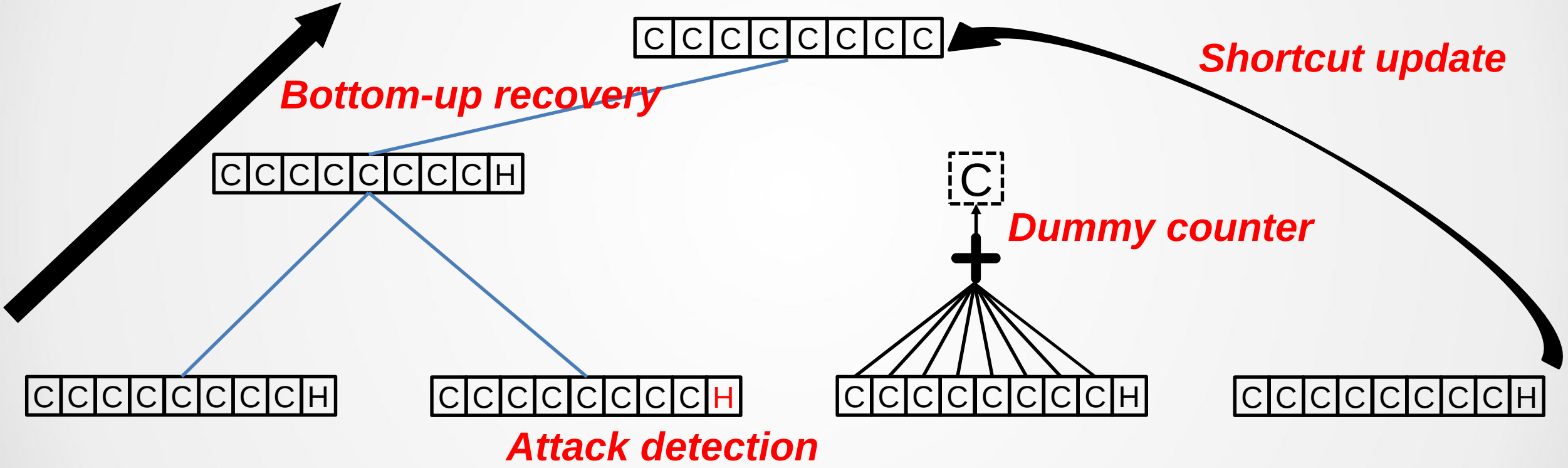
Inconsistency!

Challenges of leveraging SIT in NVM

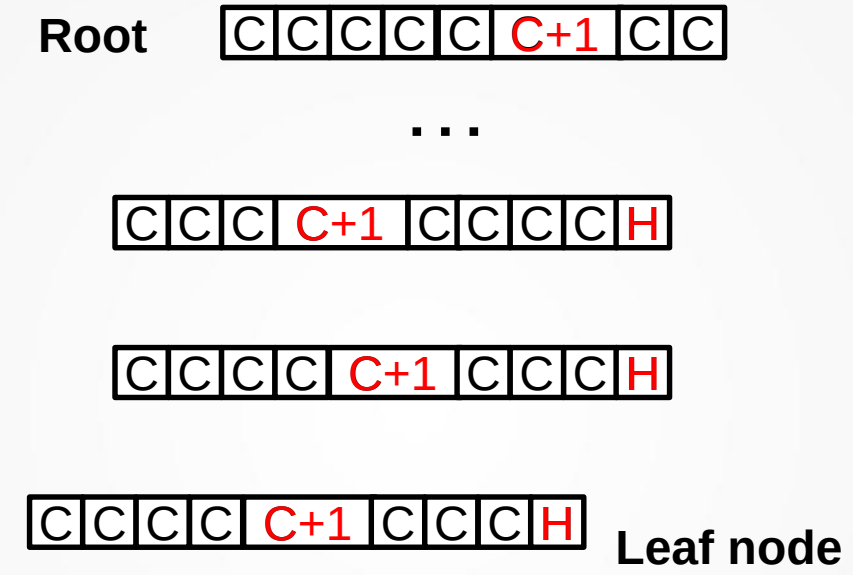
- **Crash inconsistency between root and leaf nodes**
- **High persistence overhead due to complex node-to-node dependency**



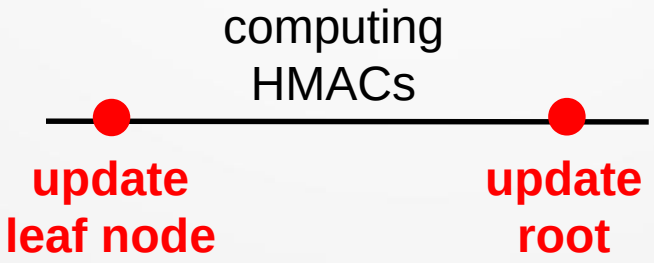
SCUE


C C C C C C C C
Bottom-up recovery
Shortcut update
C C C C C C C C H
C
Dummy counter
C C C C C C C C H
C C C C C C C C H
C C C C C C C C H
C C C C C C C C H
Attack detection

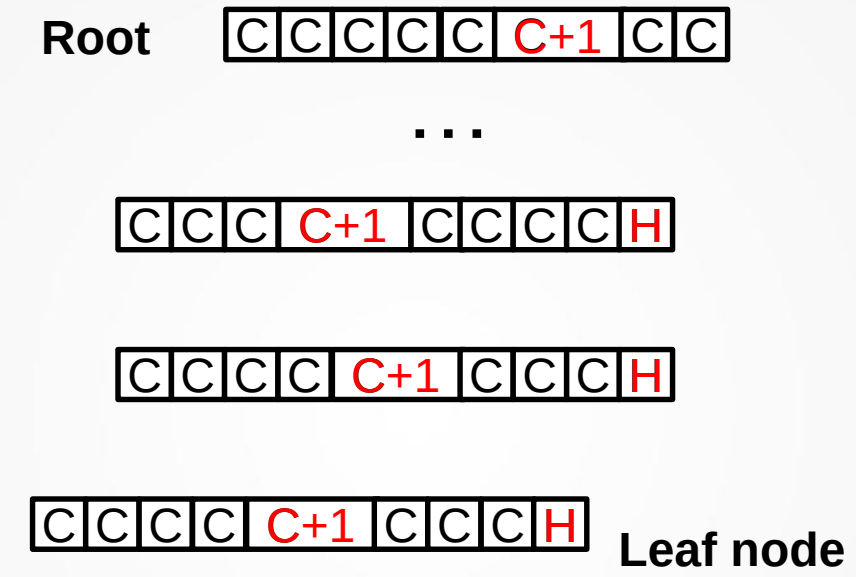
Shortcut Update



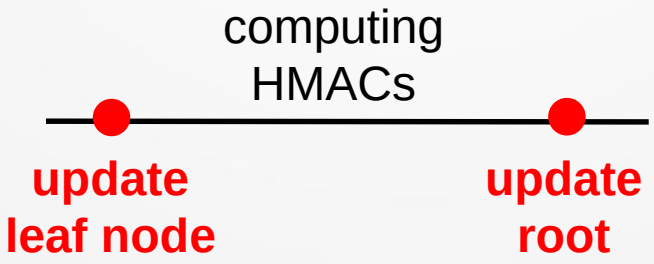
Eager update



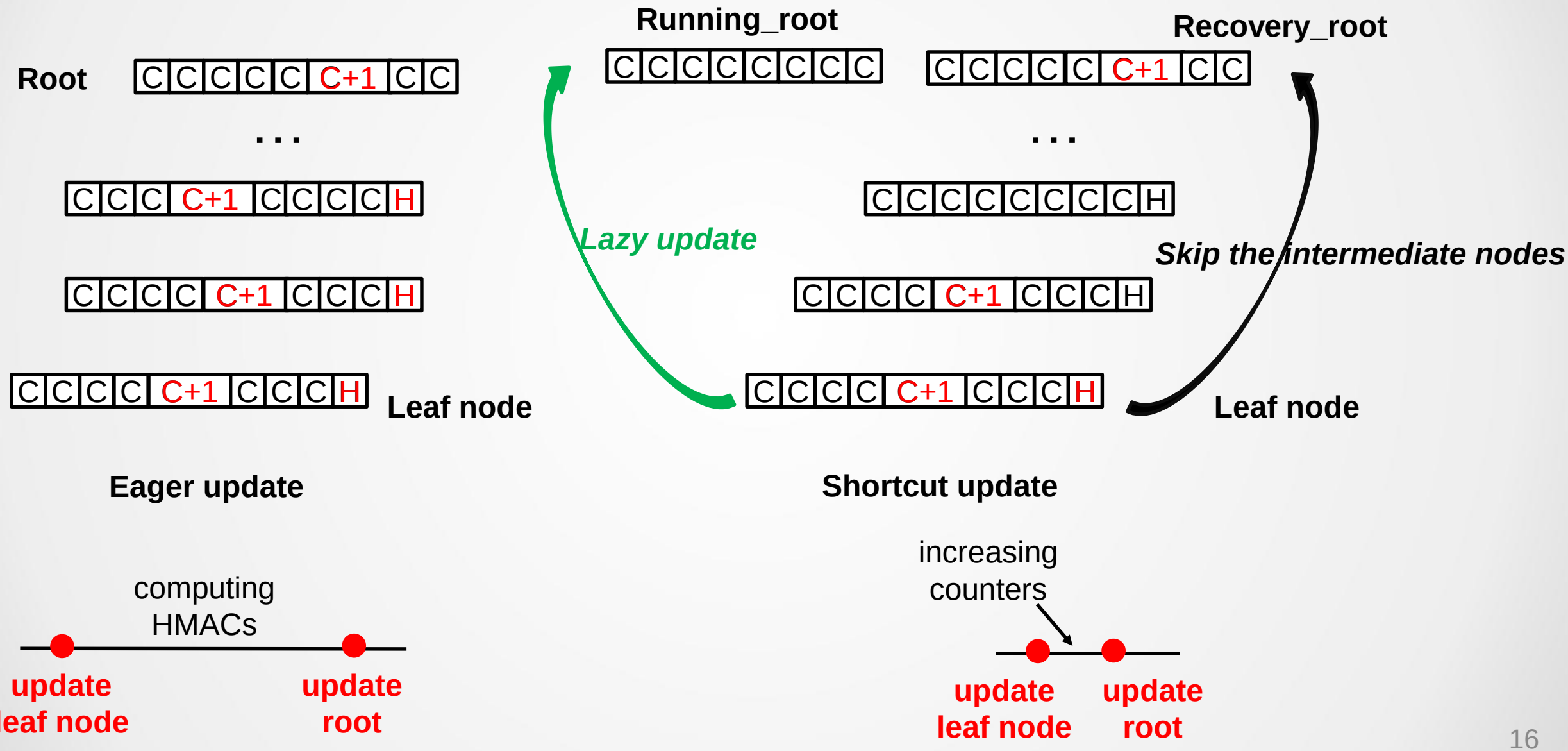
Shortcut Update



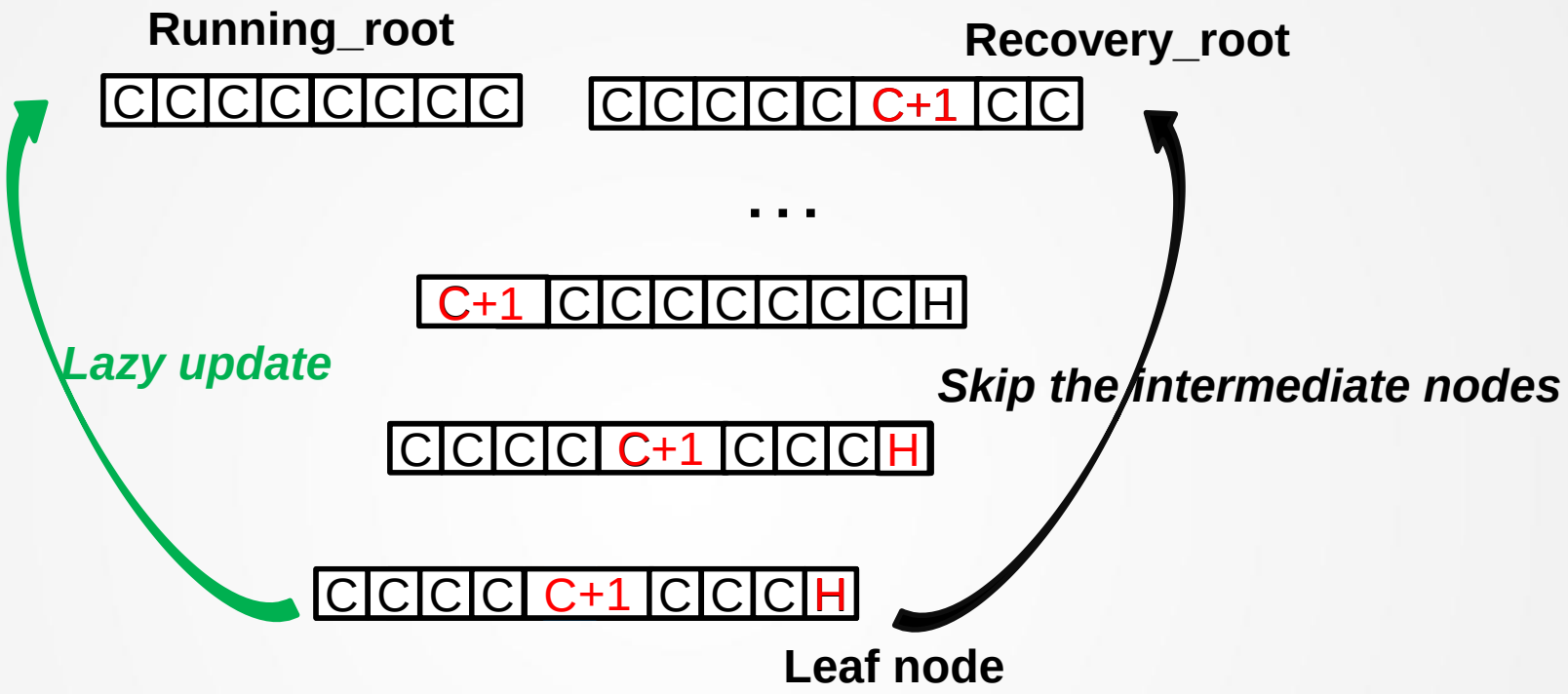
Eager update



Shortcut Update



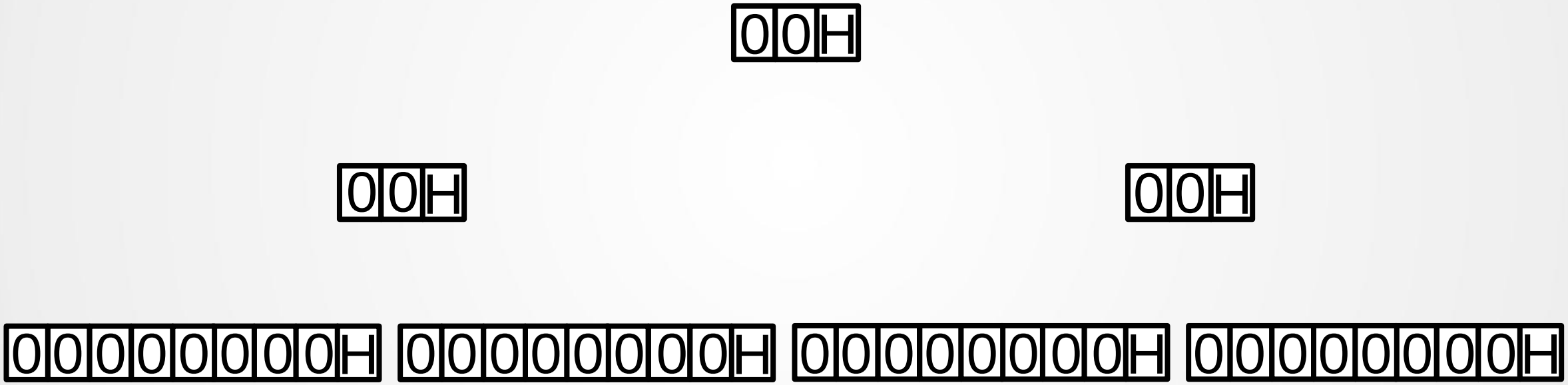
Shortcut Update



Leaf node: protected by **HMAC** and **updated parent counter**

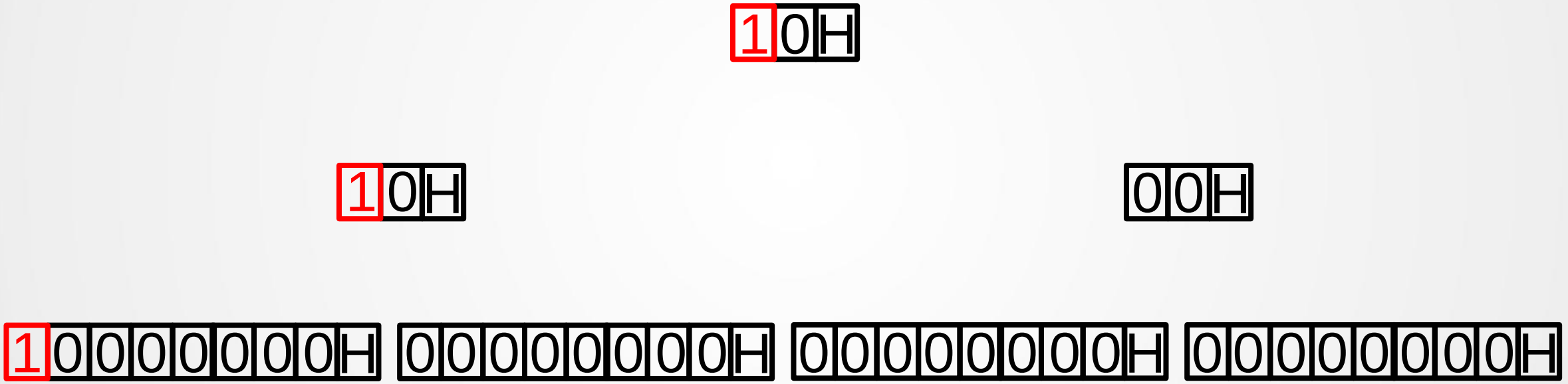
Intermediate node: protected by **HMAC** and **updated parent counter**, which is lazily updated when this node is persisted into NVM

Dummy Counter



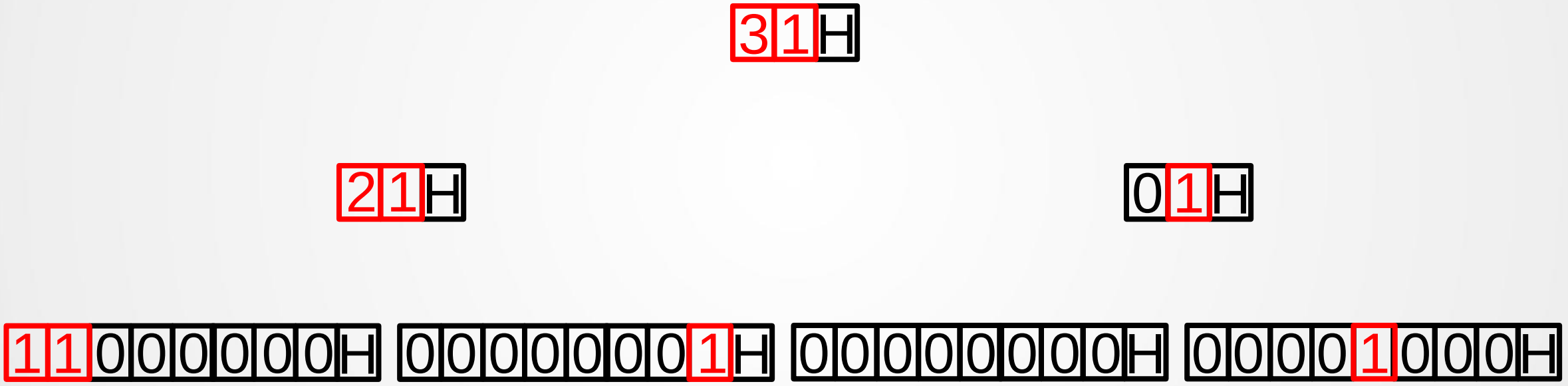
Initial counter: 0

Dummy Counter



When value of leaf counter increases

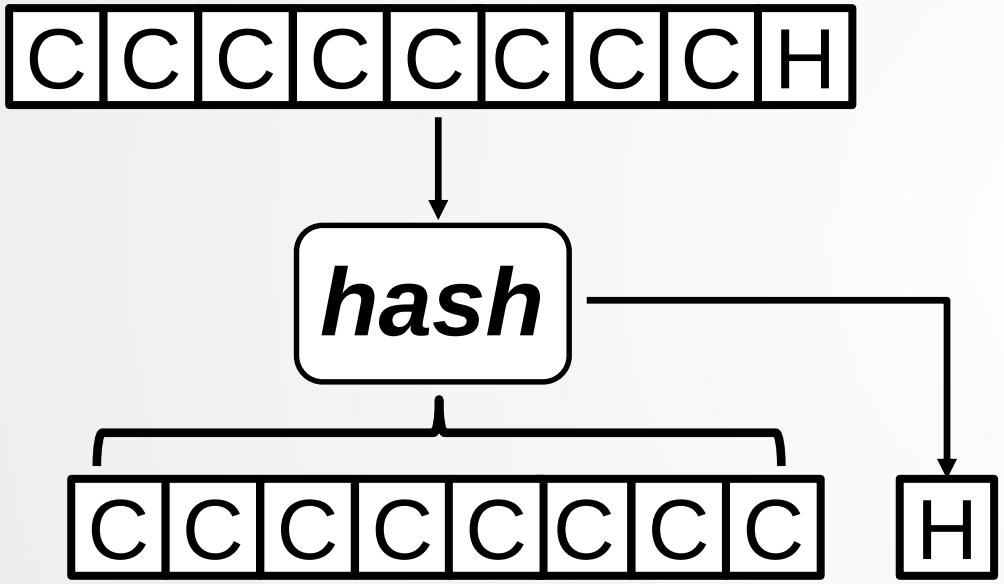
Dummy Counter



Parent_Counter = Sum(Child_Counters)
Root_Counter = Sum(Leaf_Counters)

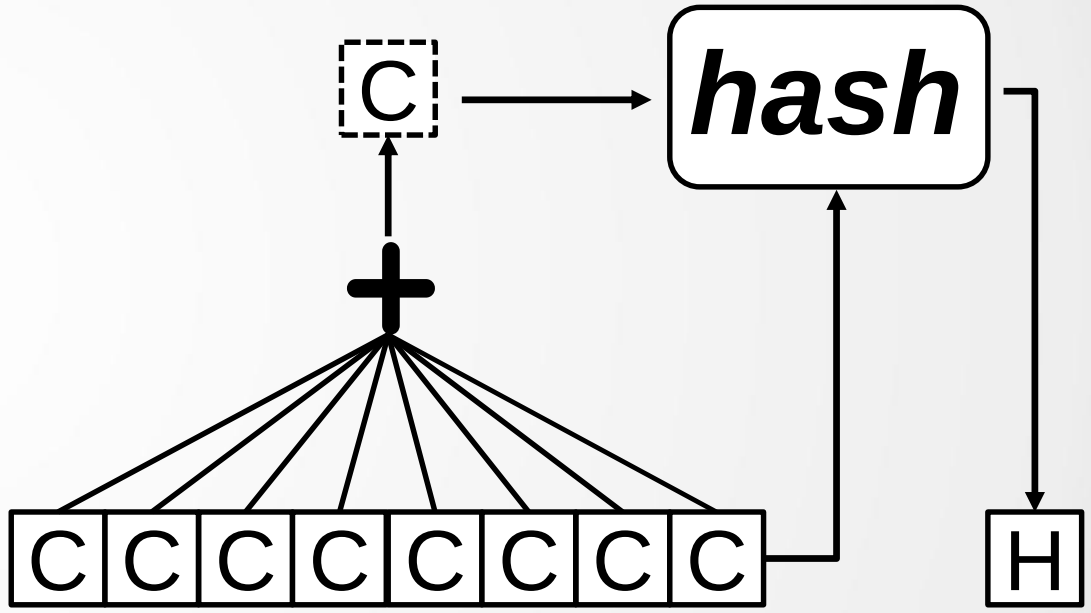
Dummy Counter

Need the parent node



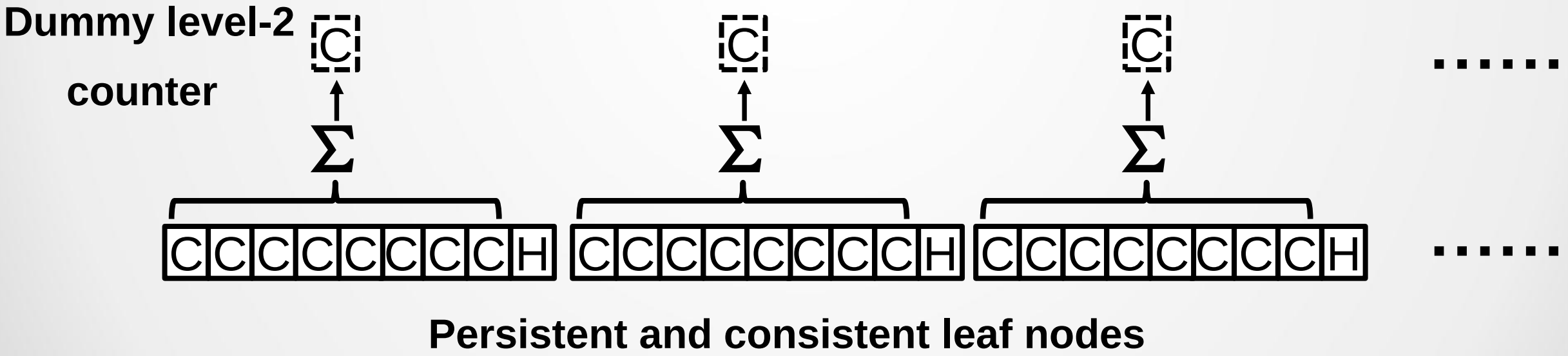
Original SIT

Don’t need the parent node



Dummy counter

Bottom-Up Recovery



Persistent and consistent leaf nodes

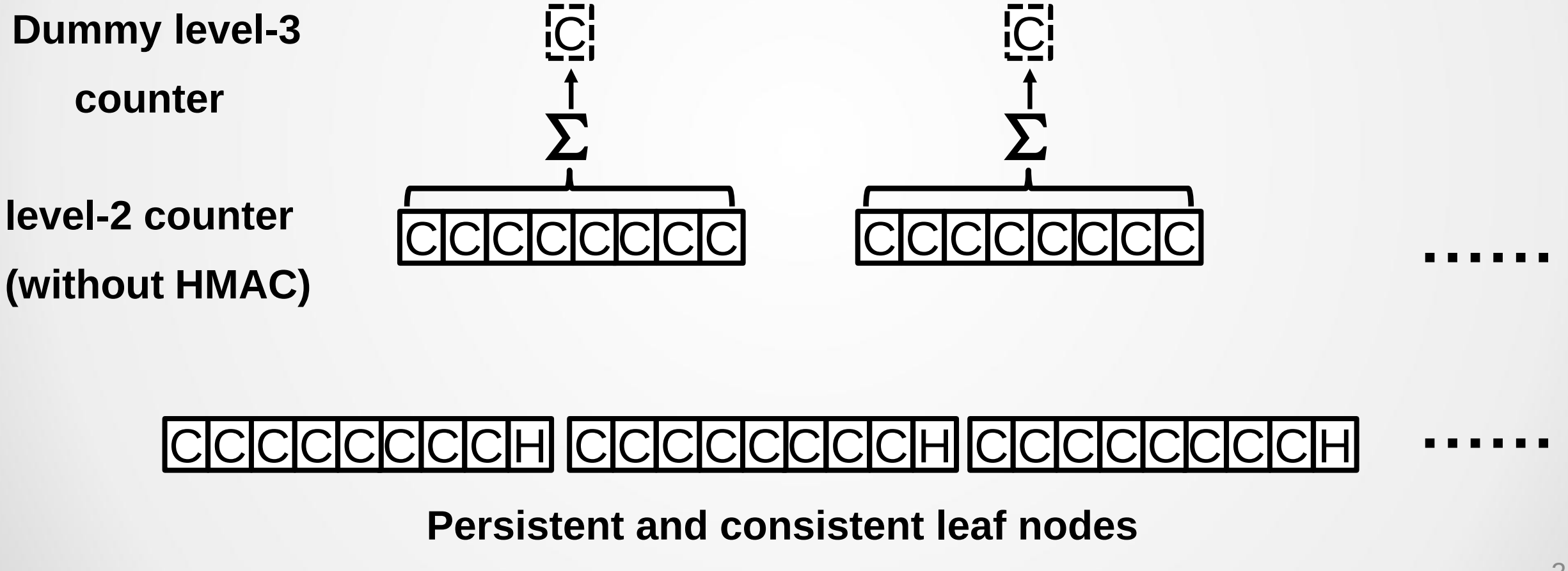
Bottom-Up Recovery

level-2 counter

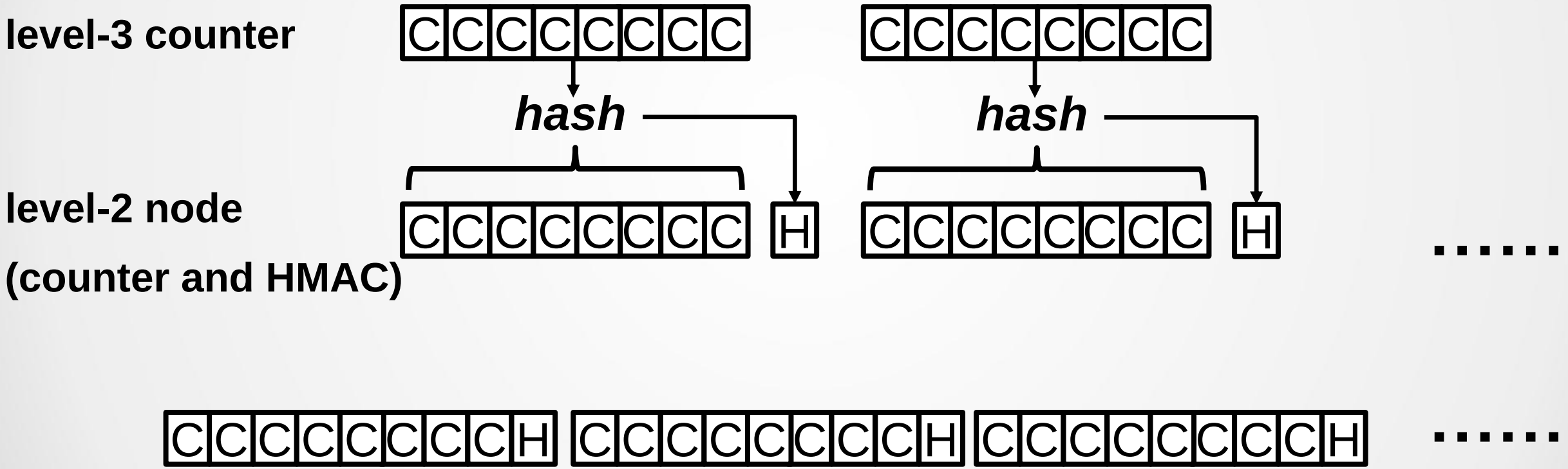
(without HMAC)

Persistent and consistent leaf nodes

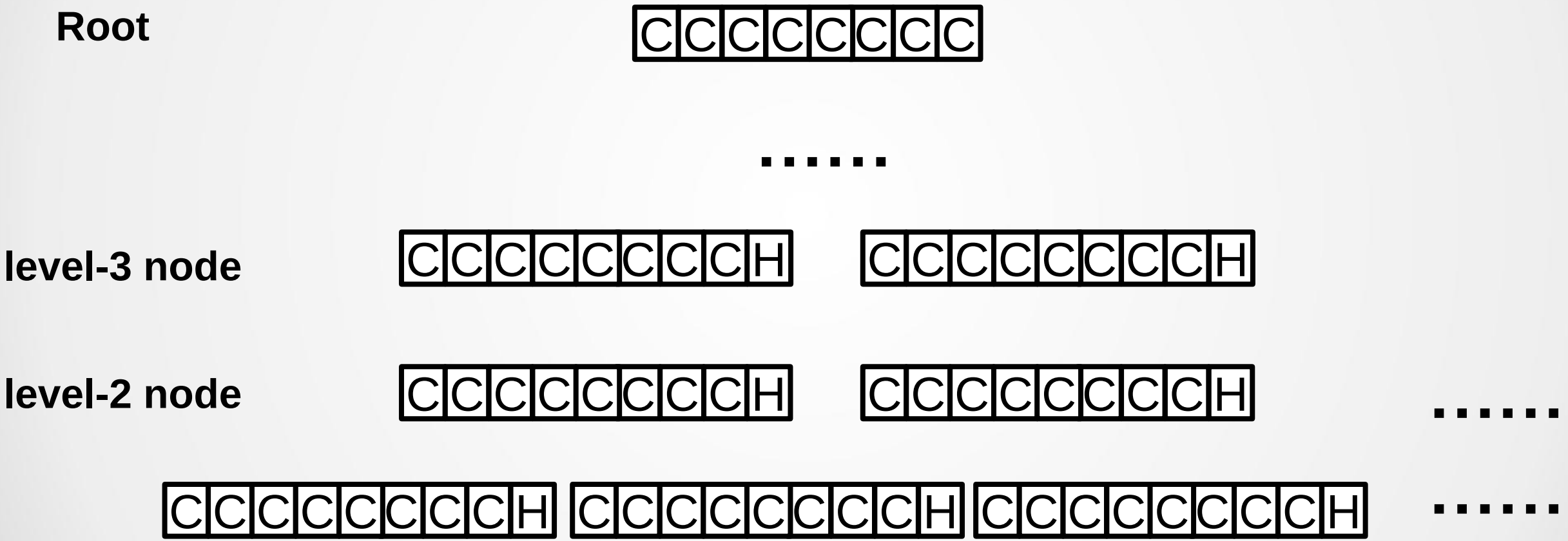
Bottom-Up Recovery



Bottom-Up Recovery



Bottom-Up Recovery



Persistent and consistent leaf nodes

Bottom-Up Recovery

Recovery Completed

Recovery Failed

Attack Detection

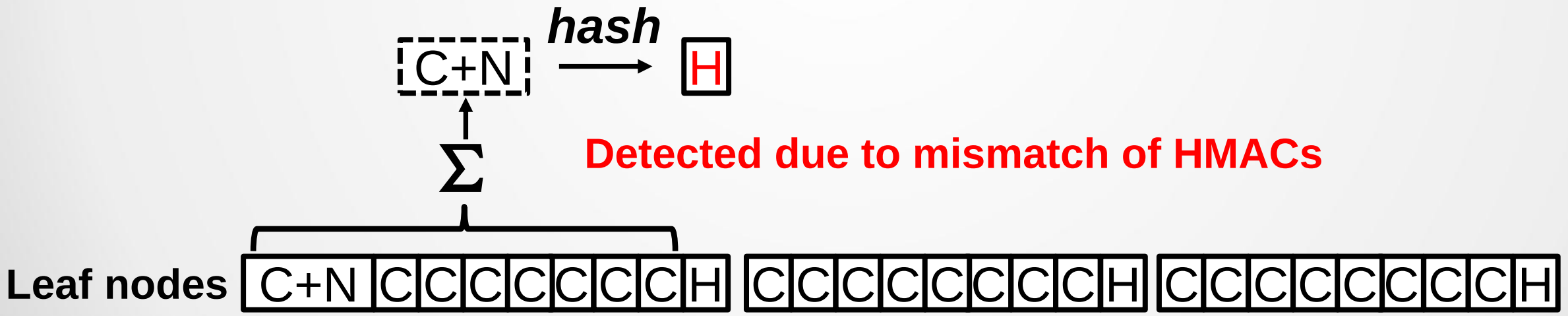
HMACs in leaf nodes			
Recovery_Root			

Recovery _Root

Leaf nodes

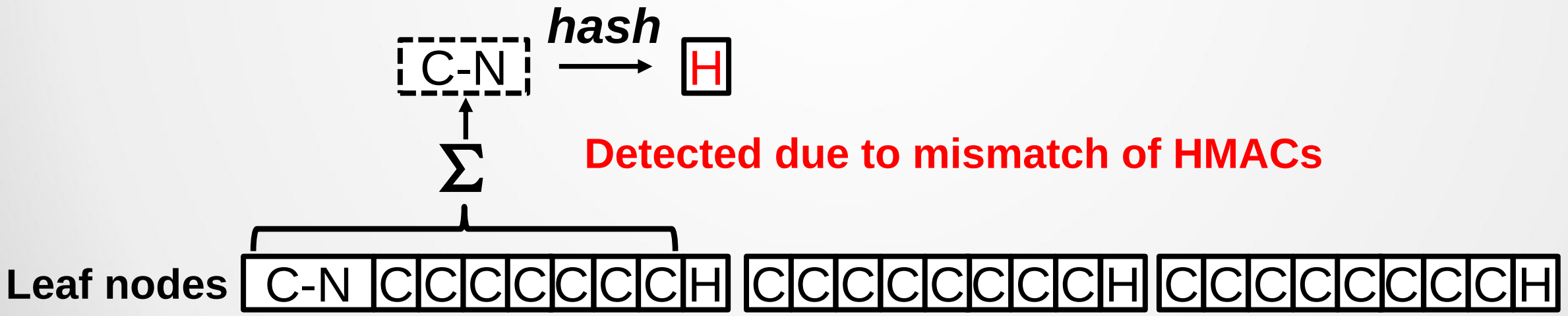
Attack Detection

	Roll-Forward Attacks		
HMACs in leaf nodes	**Detected**		
Recovery_Root	**/**		



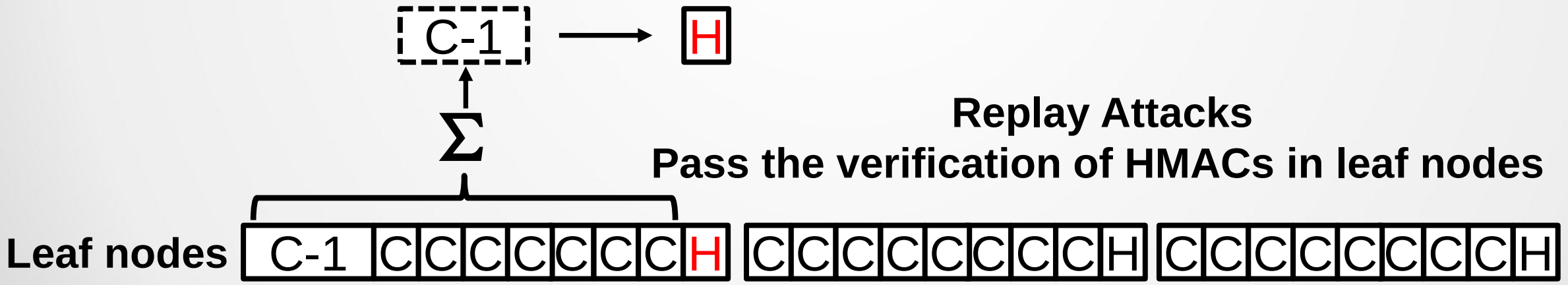
Attack Detection

	Roll-Forward Attacks	Roll-Back Attacks	
HMACs in leaf nodes	**Detected**	**Detected**	
Recovery_Root	**/**		



Attack Detection

	Roll-Forward Attacks	Roll-Back Attacks	
HMACs in leaf nodes	Detected	Detected	
Recovery_Root	/		



Replay Attacks

Pass the verification of HMACs in leaf nodes

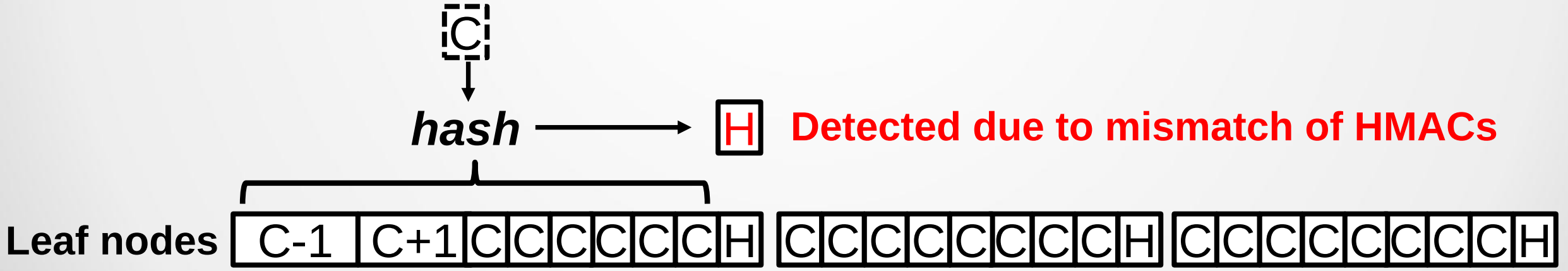
Attack Detection

	Roll-Forward Attacks	Roll-Back Attacks	
HMACs in leaf nodes	Detected	Detected	
Recovery_Root	/	Detected	

Detected due to mismatch of roots

Attack Detection

	Roll-Forward Attacks	Roll-Back Attacks	Roll-Forward+Roll-Back Attacks
HMACs in leaf nodes	Detected	Detected	Detected
Recovery_Root	/	Detected	/



Performance Evaluation

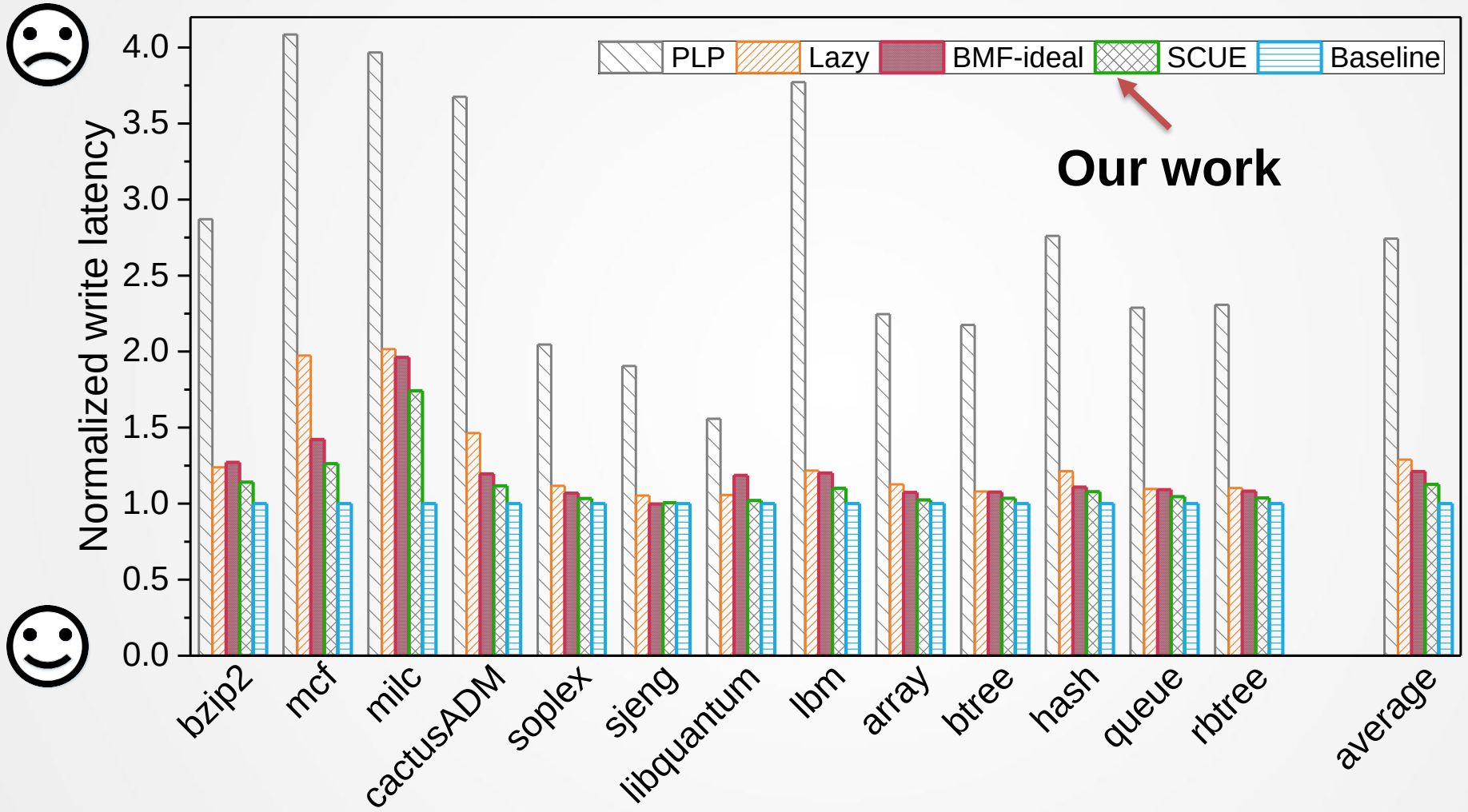
Gem5 + NVMain

Processor	8 cores(**2 GHz**); L1(**64 KB**), L2(**512 KB**), L3(**4 MB**) Caches
Memory Controller	Security Metadata Cache(**256 KB**)
NVM	16 GB
SIT	9 levels
Hash latency	{20, 40, 80, 160} cycles (default 40)

Comparisons

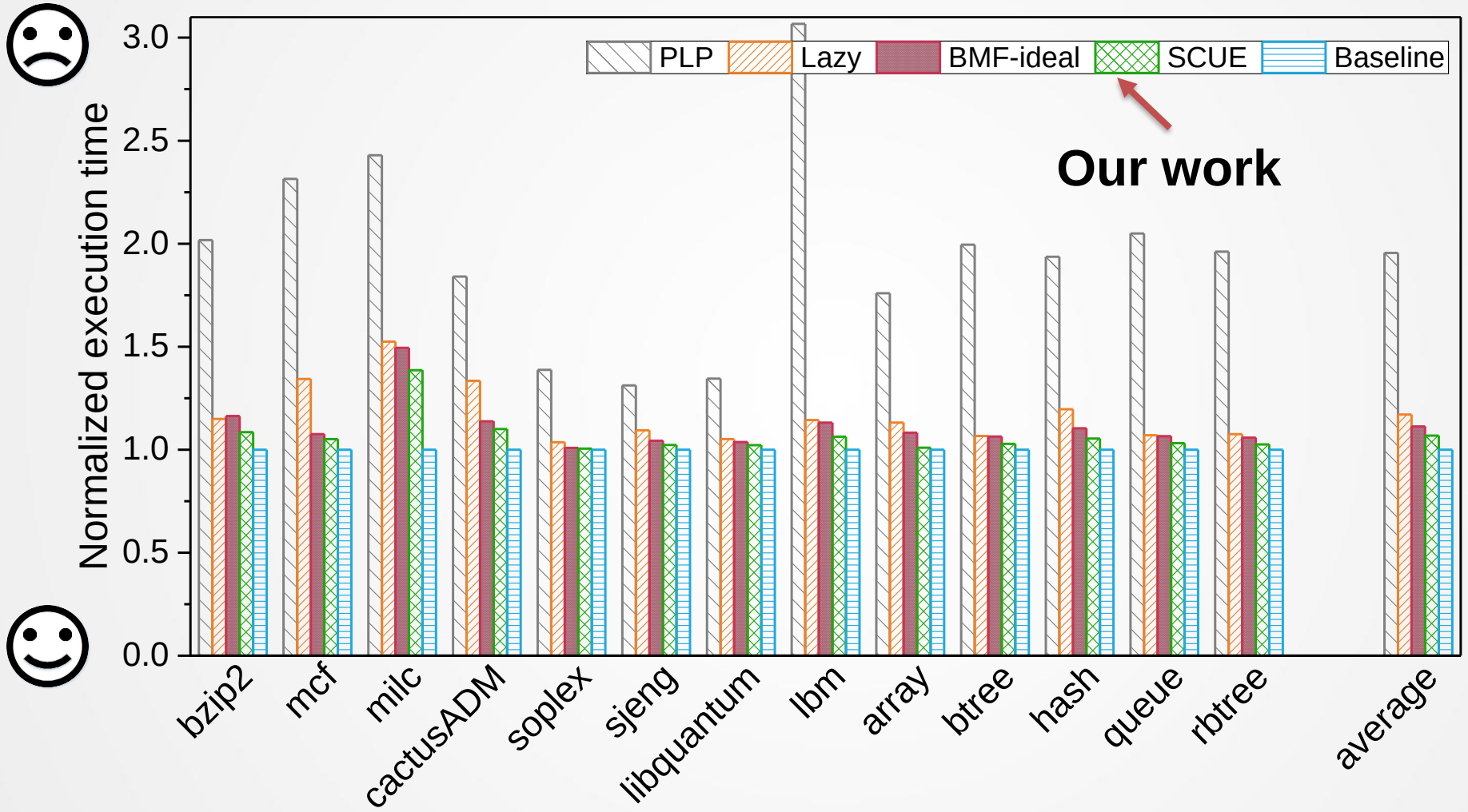
- Persist Level Parallelism (PLP) [MICRO'20]
- Ideal case of BMF (BMF-ideal) [MICRO'21]
- Lazy scheme (Lazy)
- **Our SCUE**
- Unsecure Baseline (Baseline)

Write latency



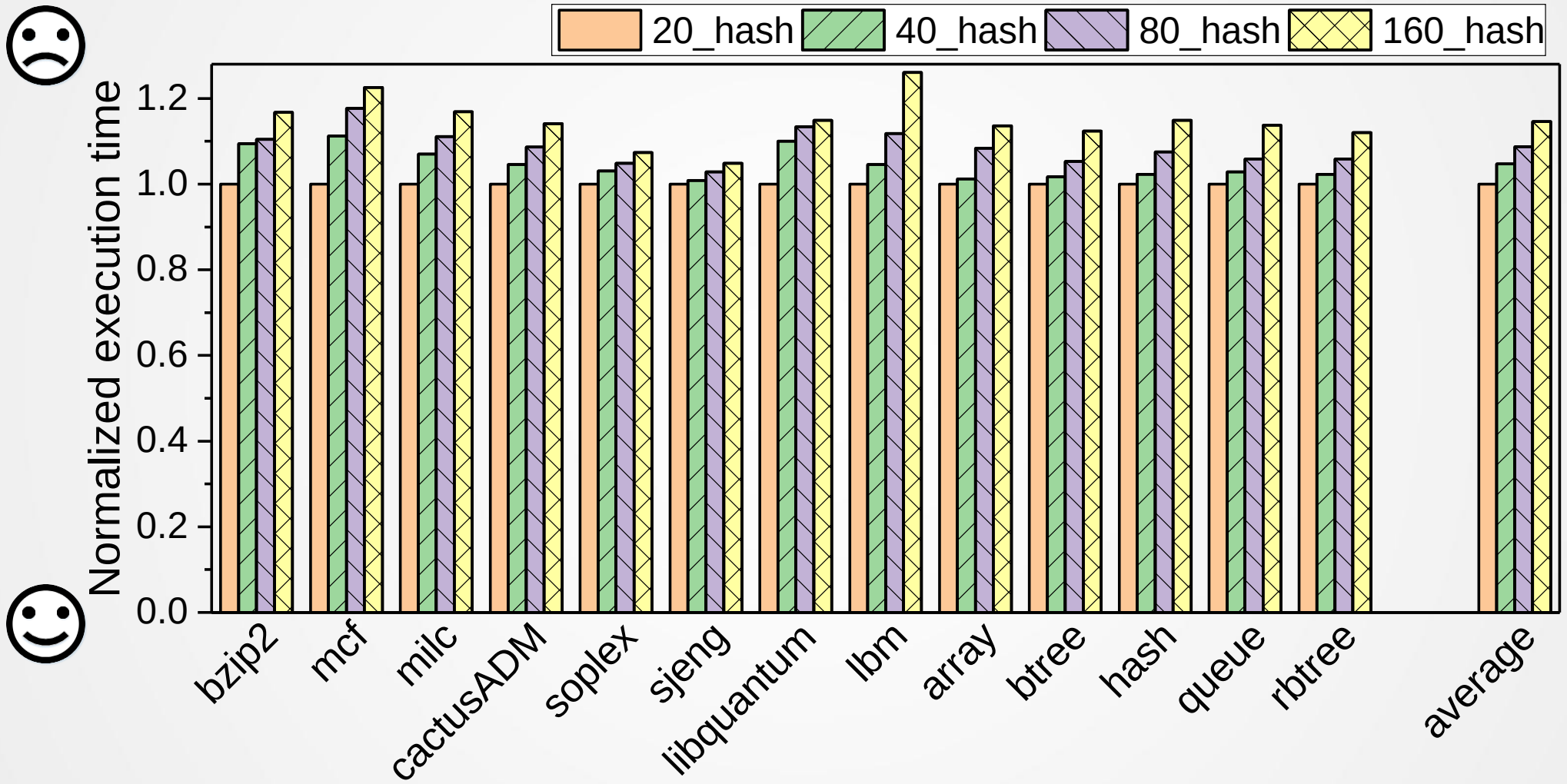
- SCUE reduces the write latency from 2.74x to 1.21x

Execution time



- SCUE reduces the execution time from 1.96x to 1.11x

Sensitive study



- Execution time increases by 1.14x when the hash latency increases from 20 cycles to 160 cycles

Conclusion

- Traditional SIT suffers from root crash inconsistency and complex node-to-node dependency
- We propose SCUE, a high-performance SIT update scheme in NVM
  - Shortcut update for root crash consistency in SIT
  - Dummy counter for decoupling node-to-node dependency
- SIT can completely replace BMT in NVM with high performance, low space overhead, and high security

Thanks!

Q&A

jmhuang@hust.edu.cn