The 17th USENIX Symposium on Operating Systems Design and Implementation

CONFLUX

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

LVMT: An Efficient Authenticated Storage for Blockchain

Chenxing Li [1], *Sidi Mohamed Beillahi* [2], *Guang Yang* [1], *Ming Wu* [1], *Wei Xu* [3], *Fan Long* [1,2]

[1]Shanghai Tree-Graph Blockchain Research Institute

[2]University of Toronto [3]Tsinghua University

Evolution of Blockchain Performance

Early blockchain systems are slow

(< 30 transactions / second)

Reaches 20,000 transactions/second

Resolved bottlenecks

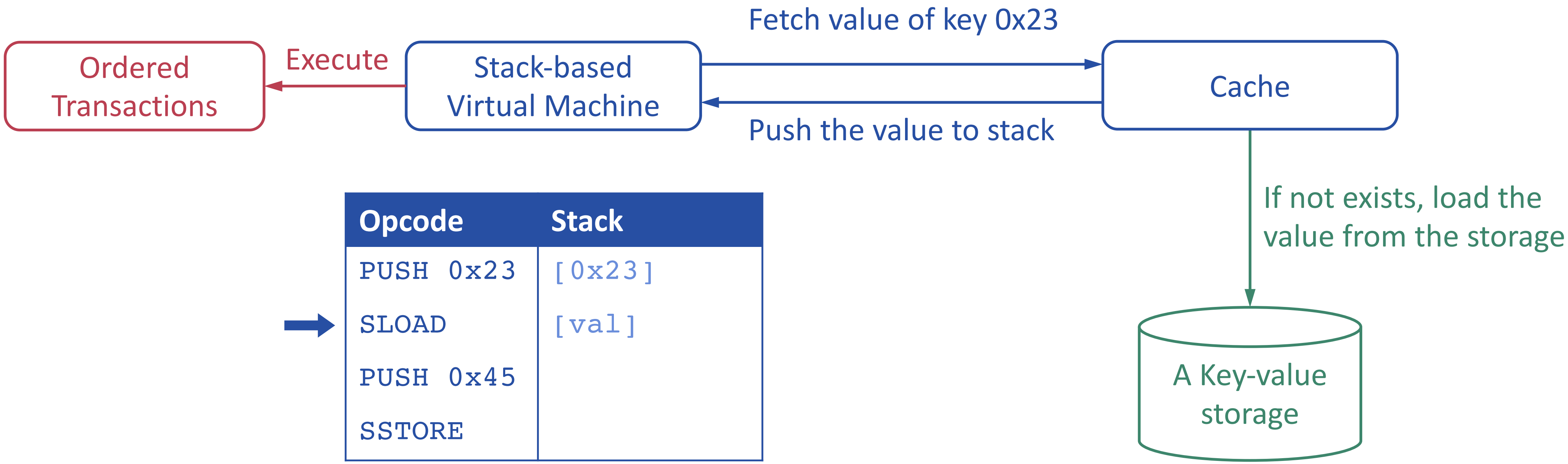
(reaches 20,000 transactions/second)

User transaction → **Transaction Broadcast** Resolved by bandwidth-efficient protocol → Blockchain nodes receive transactions → **Consensus for Transaction Order** Resolved by high performance consensus protocols → Ordered transaction with consensus

Next bottleneck

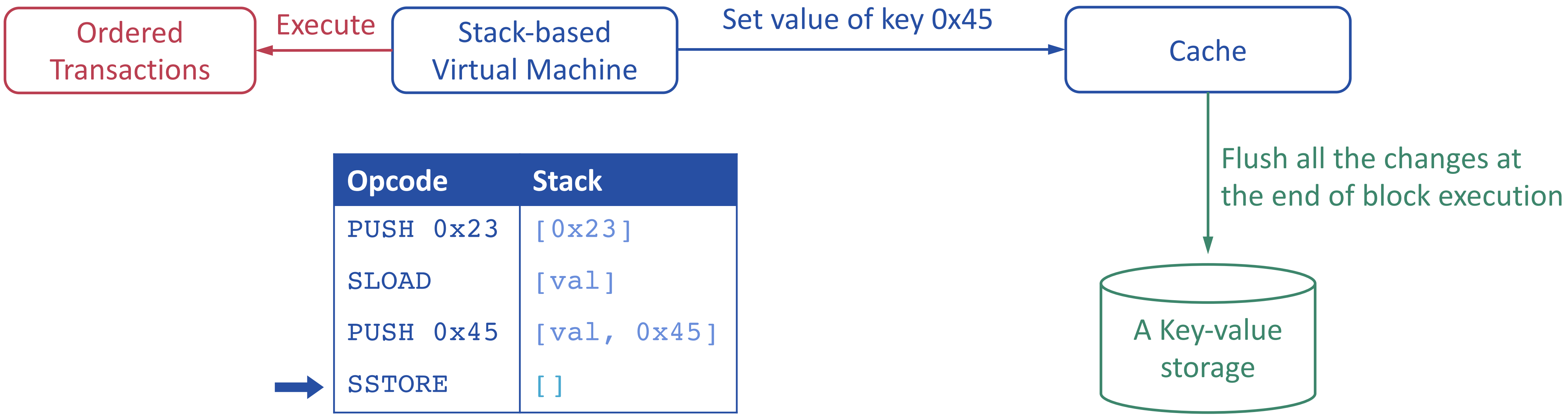
Ordered transaction with consensus → **Transaction Execution** Impedes by inefficient authenticated storage. → Execution Receipt

Architecture of Blockchain Execution Layer



Opcode	Stack
PUSH 0x23	[0x23]
SLOAD	[val]
PUSH 0x45	
SSTORE	

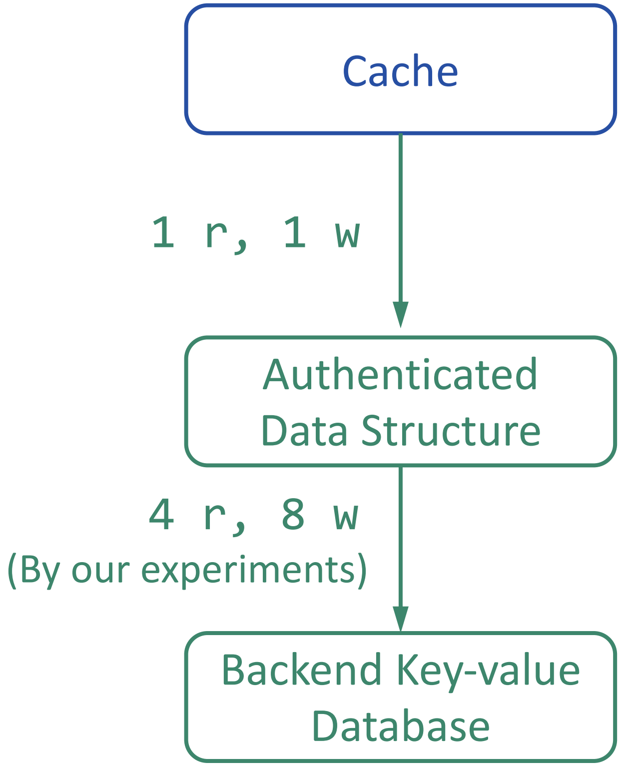
Architecture of Blockchain Execution Layer



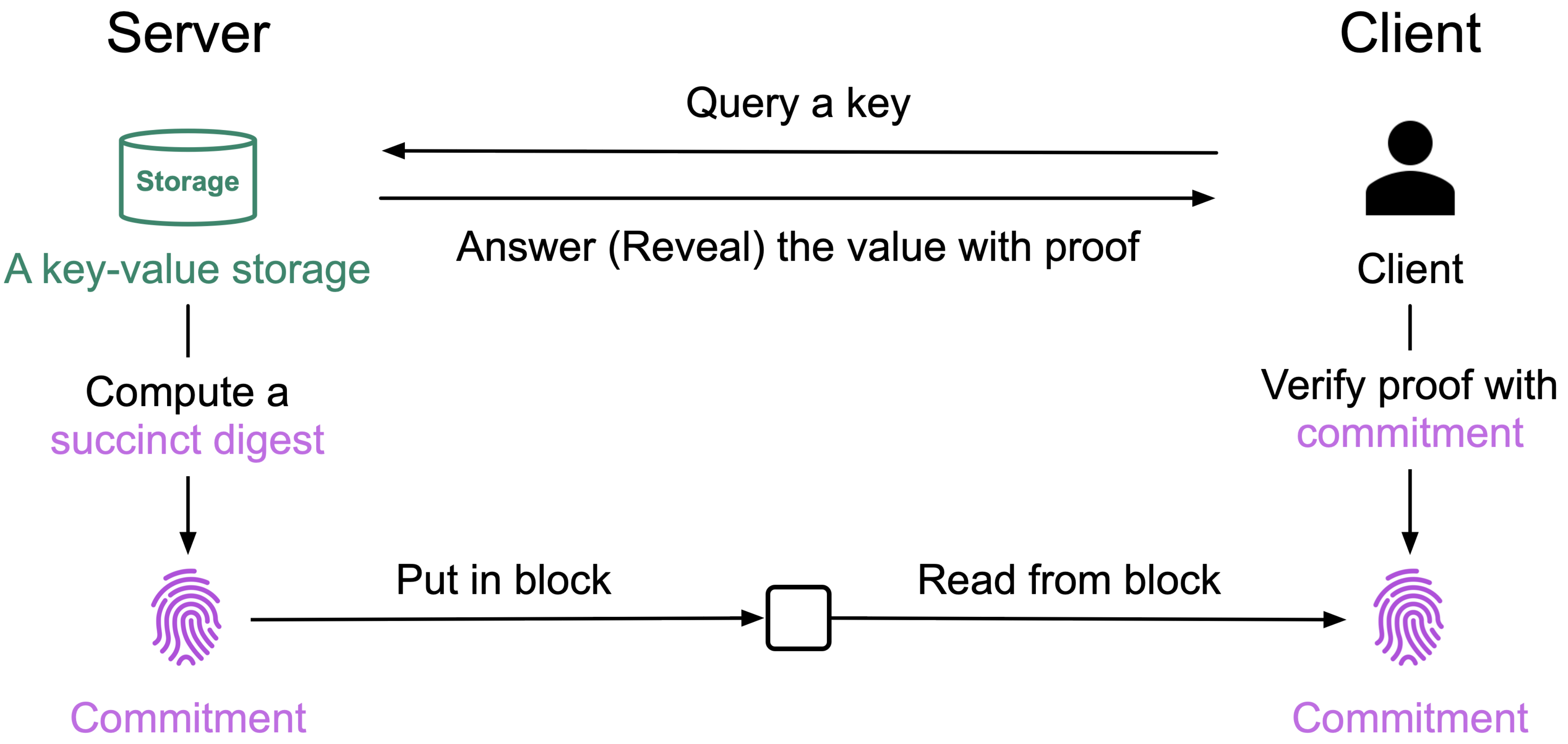
Opcode	Stack
PUSH 0x23	[0x23]
SLOAD	[val]
PUSH 0x45	[val, 0x45]
SSTORE	[]

Architecture of Blockchain Execution Layer

Opcode	Stack
PUSH 0x23	[0x23]
SLOAD	[val]
PUSH 0x45	[val, 0x45]
SSTORE	[ ]



Authenticated Storage

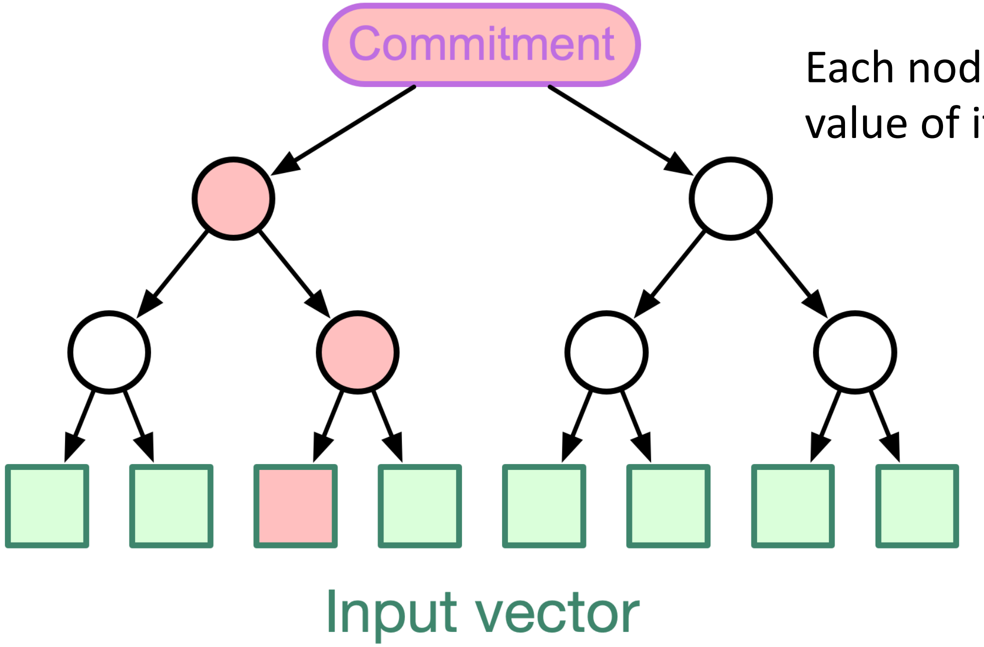


The Merkle Tree

Vector Commitment protocol: Merkle Tree

Variants for blockchain system: MPT, RainBlocks, LMPTs

- When an input element changes, the nodes along the path also changes.
- Each node is a key-value pair in backend $\rightarrow O(\log n)$ read-write amplification

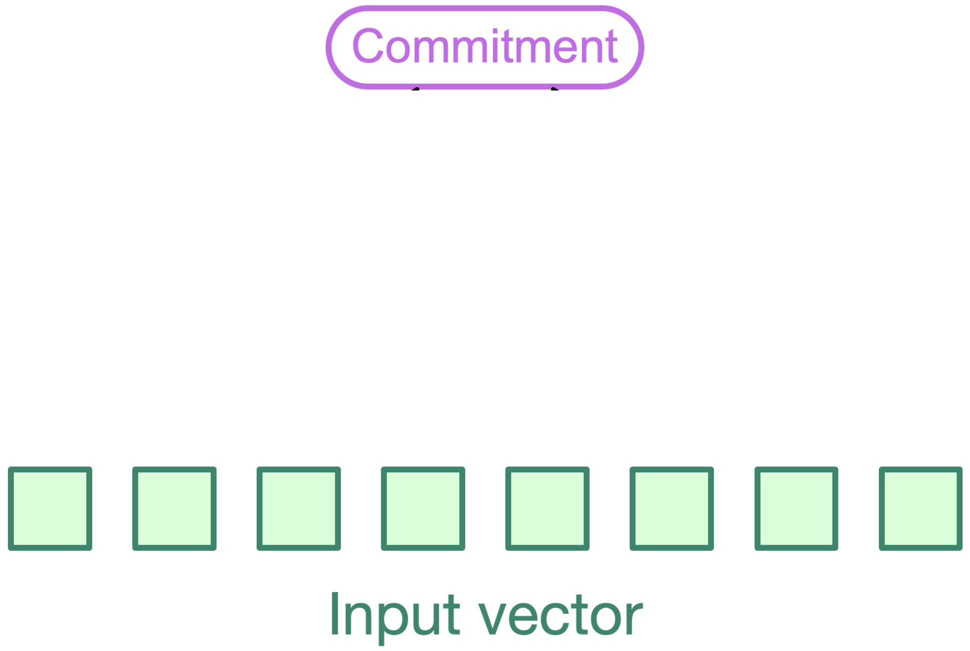


Each node is the hash value of its children.

The Merkle Tree

Vector Commitment protocol: AMT

Variants for blockchain system: LVMT (our work)



- AMT removes the inner nodes and achieves $O(1)$ cost in maintaining commitment.

Challenges in using AMT

- Fast in complexity $\neq$ fast in practice
  - AMT has slow cryptographic operations.
- AMT is not scalable.
  - Max capacity of AMT $=$ Size of public parameters.
- Proof generation is Expensive
  - Maintaining data for generating proofs is also $O(\log n)$

Challenge 1: Costly cryptographic operations

In AMT, each time a value changes as,

$$a_i \rightarrow a_i{}'$$

the commitment adjusts accordingly

$$C \rightarrow C + (a_i{}' - a_i) \cdot G_i.$$

Precomputed Parameter (200 byte)

Elliptic Curve Multiplication **(92 μs)**

Big Integer Subtraction (<0.01 μs)

Elliptic Curve Addition (0.34 μs)

Challenge 1: Costly cryptographic operations

In AMT, each time a value changes as,

$$a_i \rightarrow a_i{}' \qquad \text{(Assumes } a_i' - a_i = 1\text{)}$$

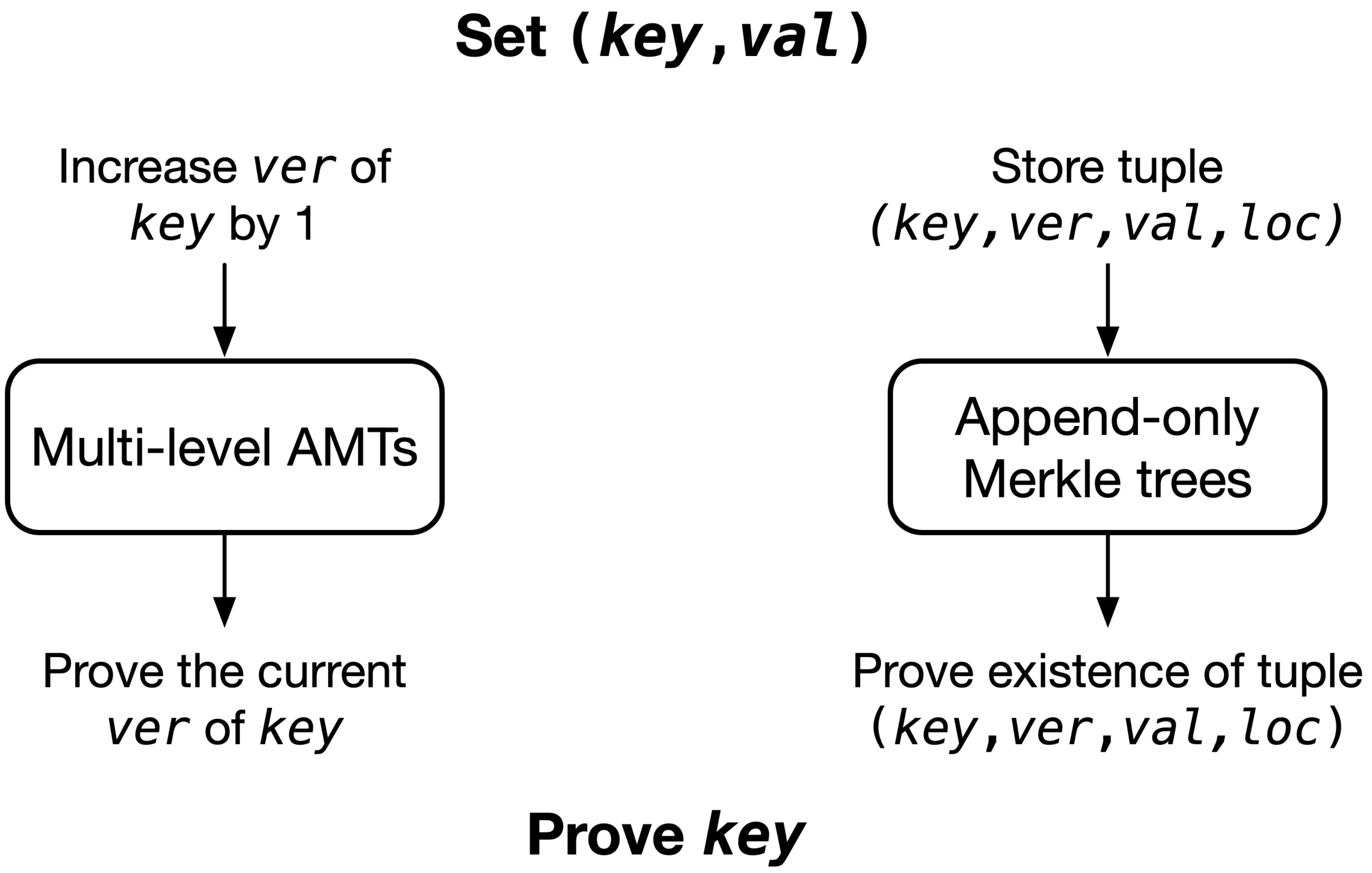
the commitment adjusts accordingly

$$C \rightarrow C + \quad 1 \quad \cdot G_i.$$

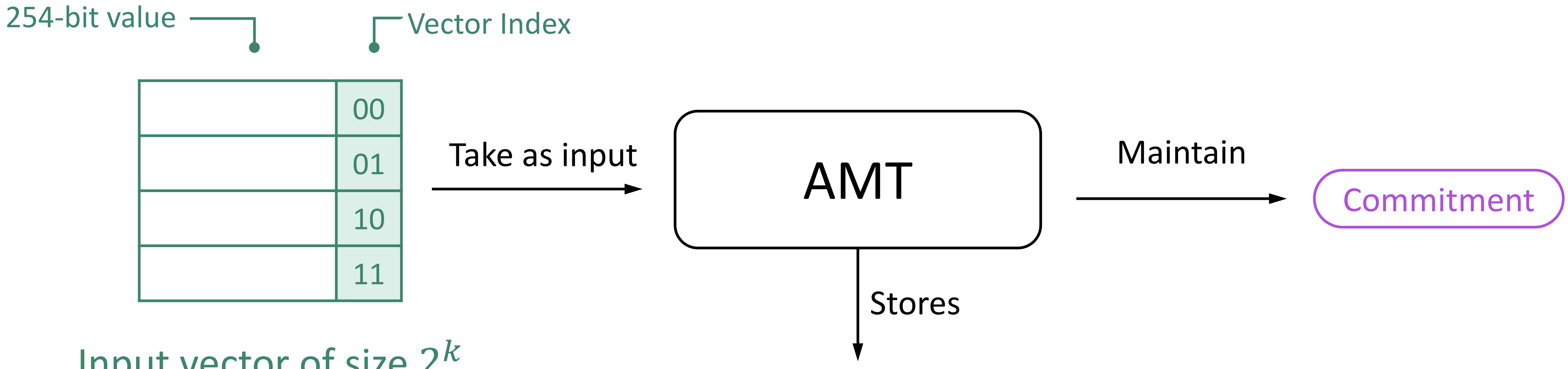
Precomputed Parameter (200 byte)

Elliptic Curve Addition (0.34 μs)

Solution 1: Version-based database



Challenge 2: AMT is not scalable



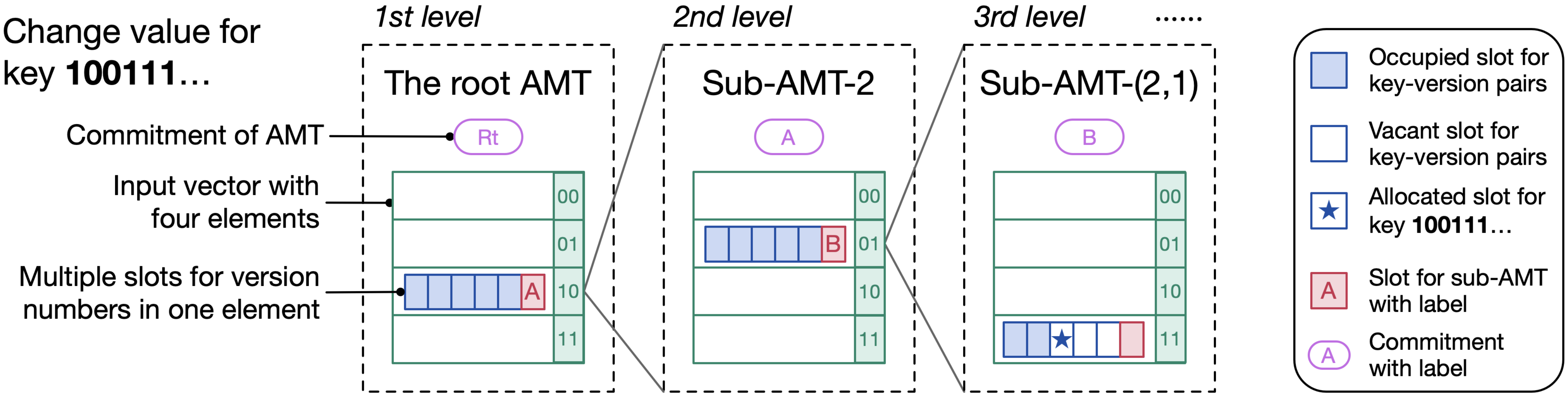
Input vector of size 2^k
(for any k)

1. Determined at the setup phase

2. Precomputed parameters in size of 2^k

3. Blockchain has 256-bit key space, but $k = 256$ is infeasible.

Solution 2: Use multiple-level AMT.



Set *(key, val)*

1. Incease version numbers in ★ , B and A by 1 and update commitments.
2. Add the following tuples to the Merkle trees.

(*key* , ★ , *val*, (3,2))

((2,1), B , B)

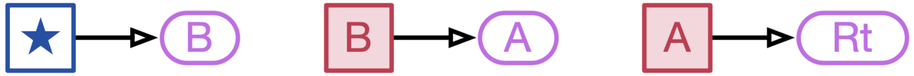
(2 , A , A)

The slot index

The sub-AMT level allocating this key

Prove *key*

1. Prove the version numbers with respect to the AMT commitment:

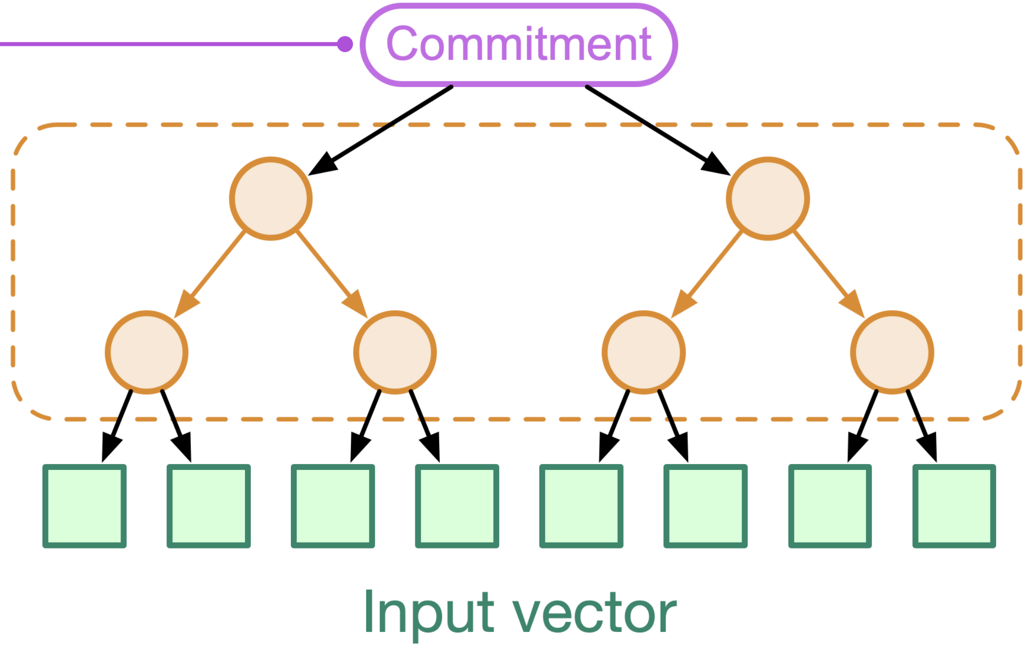


2. Prove the existence of the left three tuples in Merkle trees to demonstrate the commitments at specifined version numbers.

Challenge 3: Maintaining proof data incurs significant costs

Nodes *not serving clients* only need to maintain commitment in $O(1)$ time

Nodes *serving clients* maintain auxiliary information for generating proof in $O(\log n)$ time.

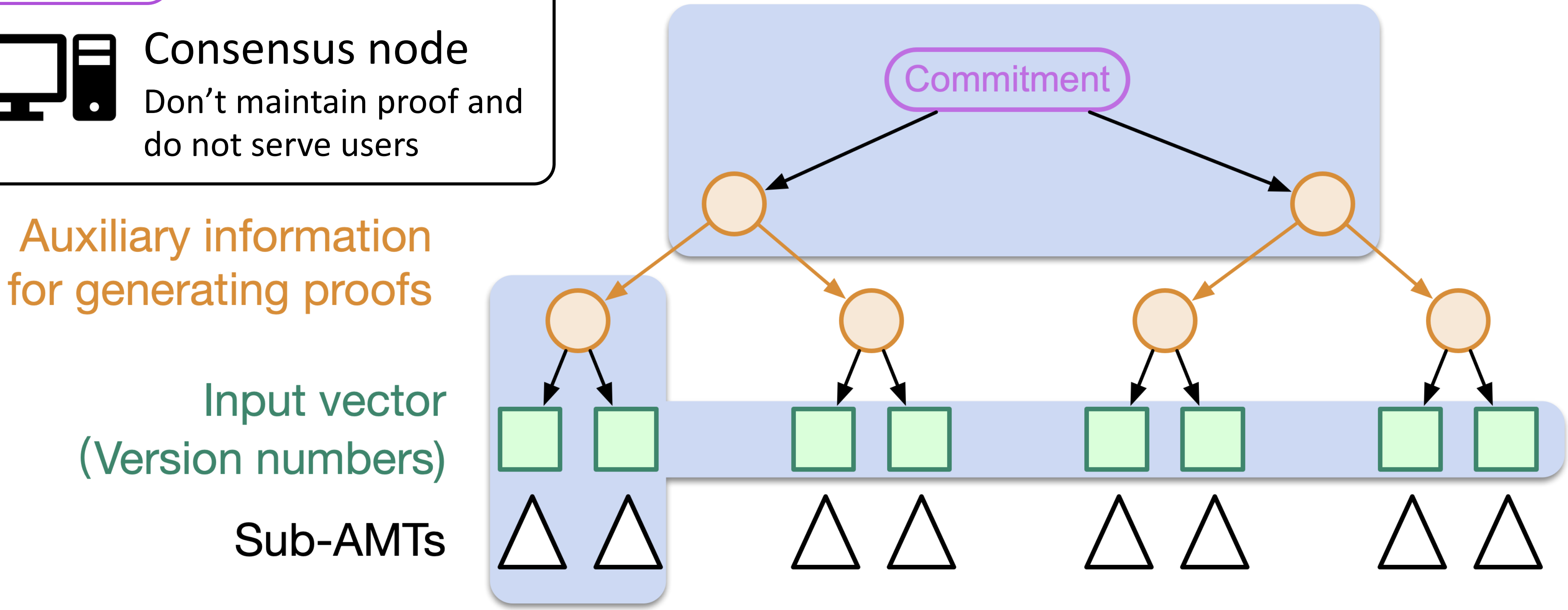


Solution 3: Proof Sharding

Commitment

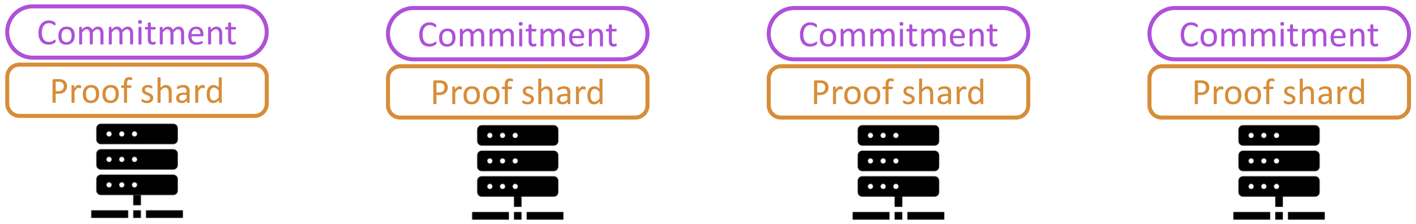
Consensus node

Don't maintain proof and do not serve users



RPC provider

Maintain proofs with a cluster to serve users



Modular Authenticated Storage Benchmark Tool

Execution Task

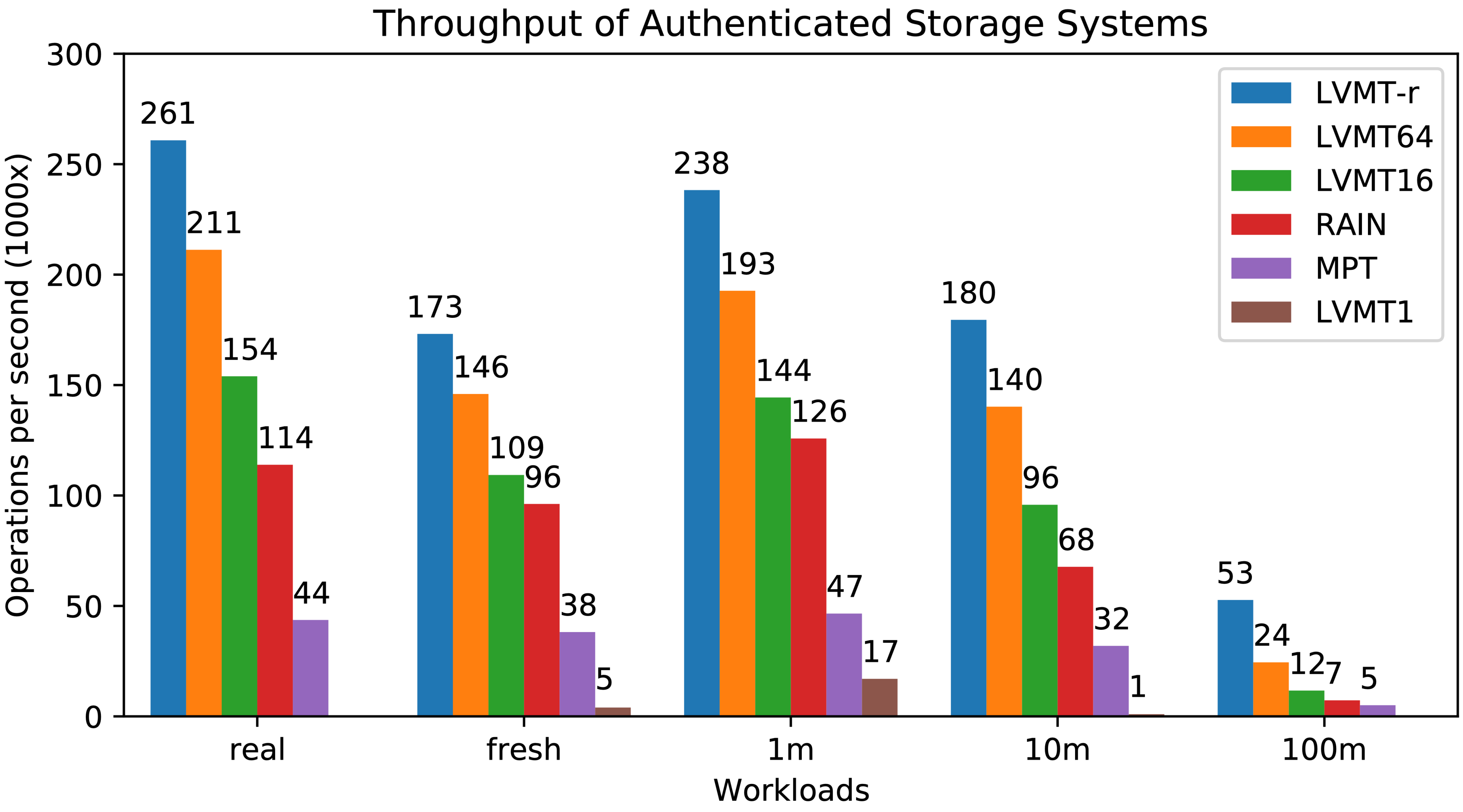
- 1m 10m 100m Random task on various ledger size (in million)
- fresh Randomly access new keys only
- real Real Ethereum trace

Authenticated Data Structure

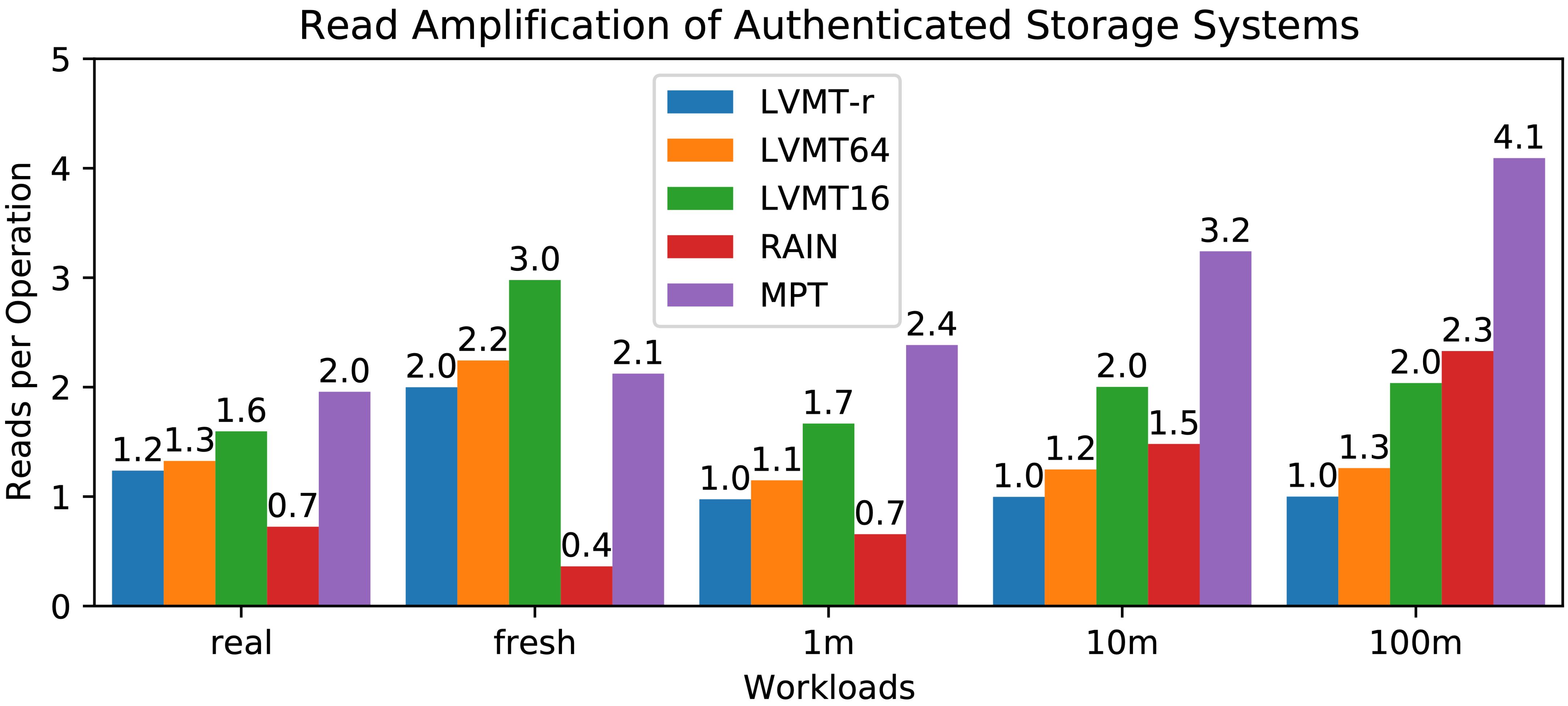
- LVMT-r Only maintains commitment
- LVMT# #: the fraction of proof shards
- MPT RAIN LMPTs Baselines

Key-Value Database

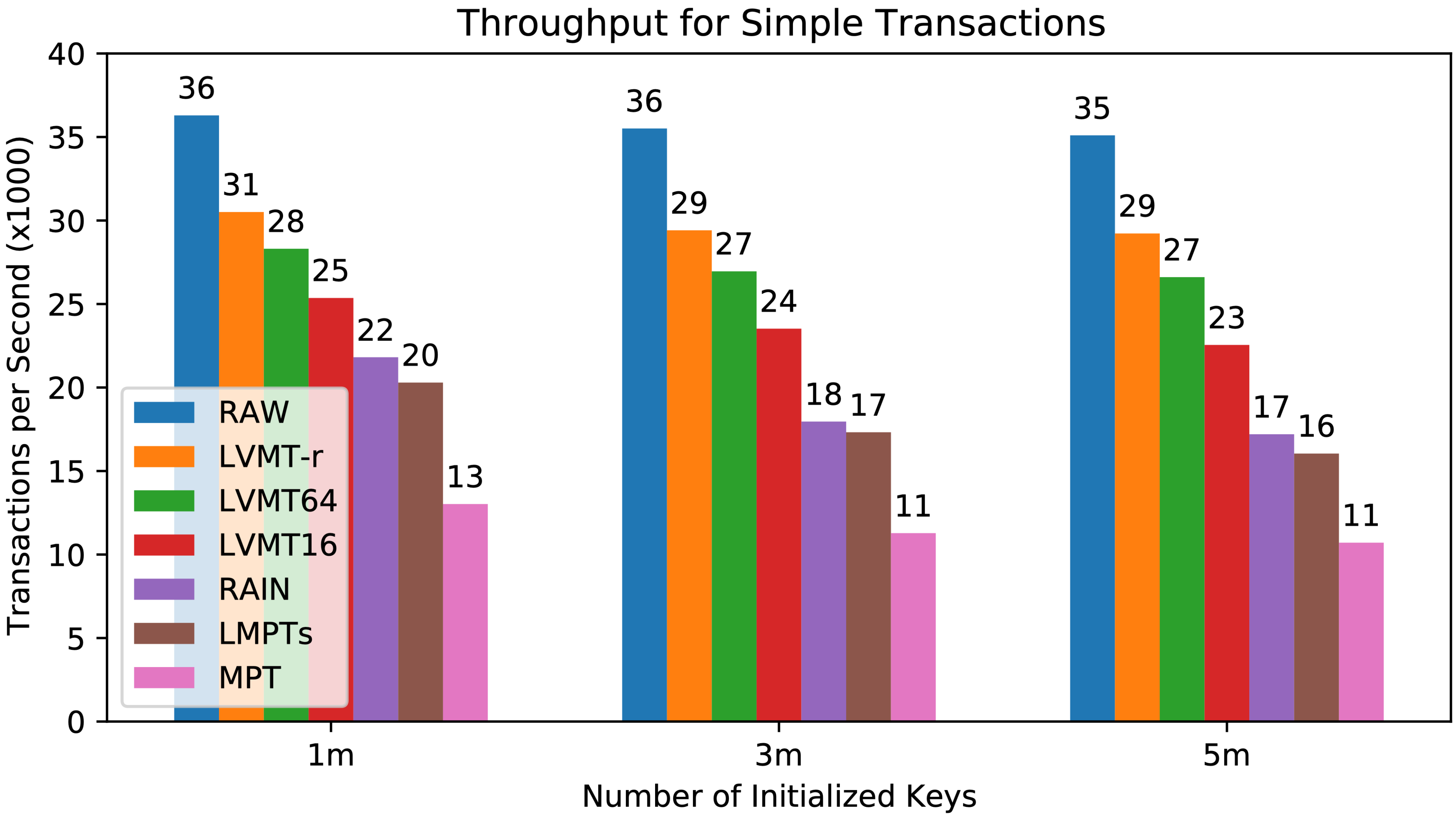
Throughput on micro-benchmarks



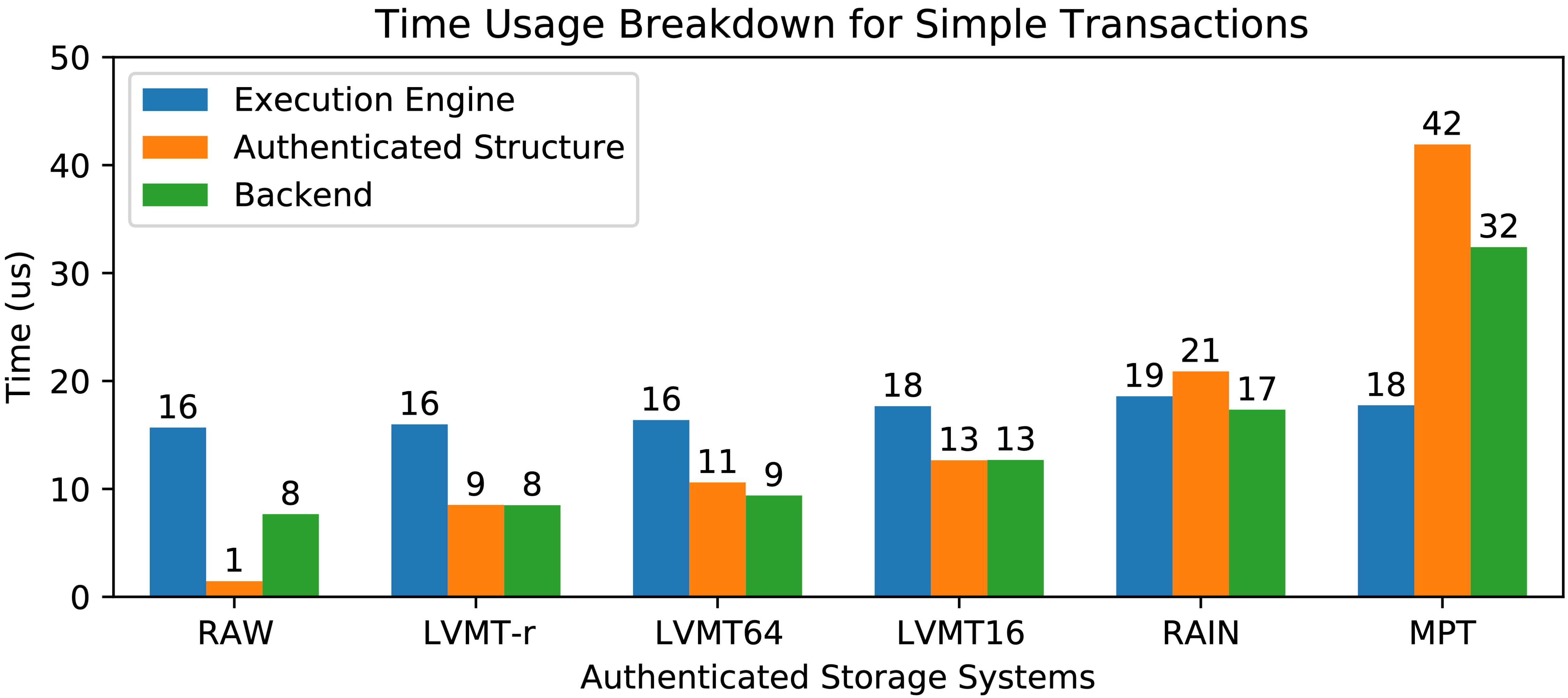
Read Amplification



Throughput on a Blockchain Node



Time Usage Breakdown



Conclusion

- LVMT utilizes the superior vector commitment protocol AMT, offering higher optimization potential.

- Through the version-based design, multi-level AMT, and proof sharding, LVMT addresses challenges effectively.

- LVMT enhances the execution throughput of a blockchain system by up to 2.7x.

Thank you and see you in Q&A

Email: lylcx2007@gmail.com

Github: https://github.com/ChenxingLi/authenticated-storage-benchmarks

https://github.com/Conflux-Chain/conflux-rust/tree/asb-e2e