

Timelock Drive: Isolated Time-Based Defense for Storage Systems

Jonah Rosenblum · Juechu Dong · Peter M. Chen · Satish Narayanasamy

University of Michigan

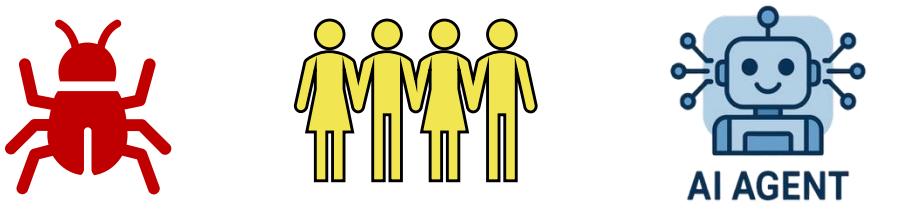
Data Integrity Threat: Credentials are not Enough

$265B annual ransomware cost by 2031

67% of hospitals attacked in 2024

66% of attacks against backup succeed

Vulnerabilities



12% **88%** Emerging

How do we protect data integrity when credential-based security fails?

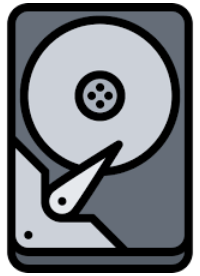
Use time as a defense

Standard Drive

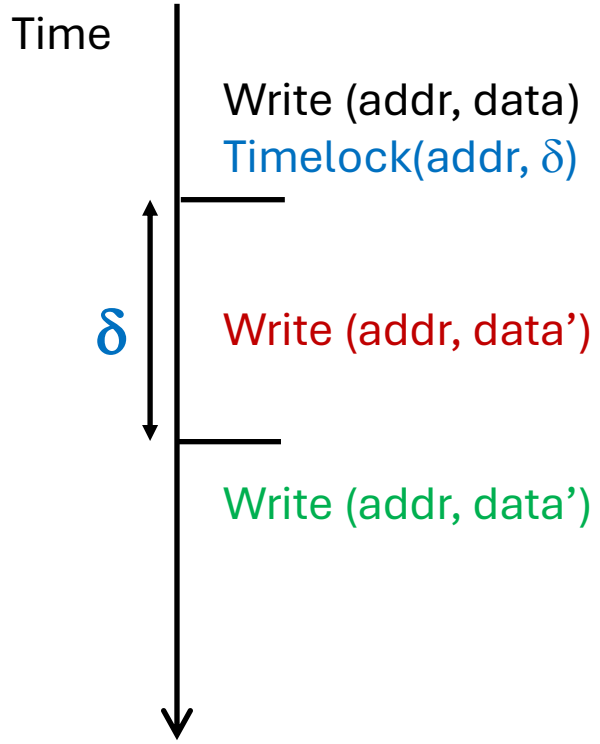
HDD/SSD

Read (addr)
Write (addr, data)

ATA/NVMe

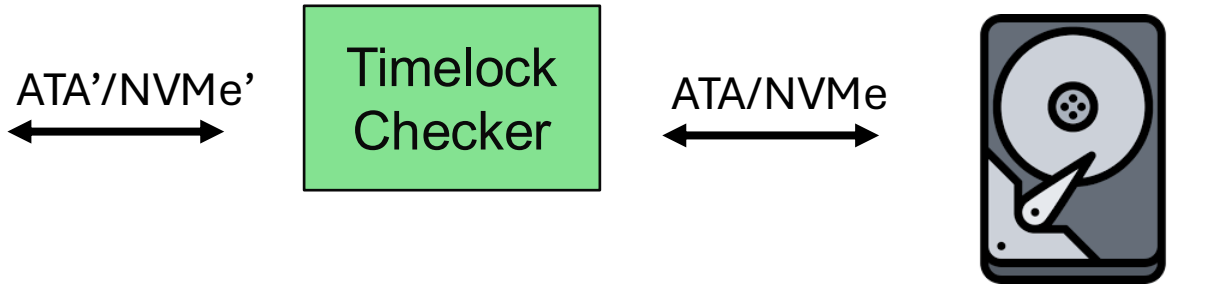


Timelock Drive



New interface command

Timelock(addr, δ)

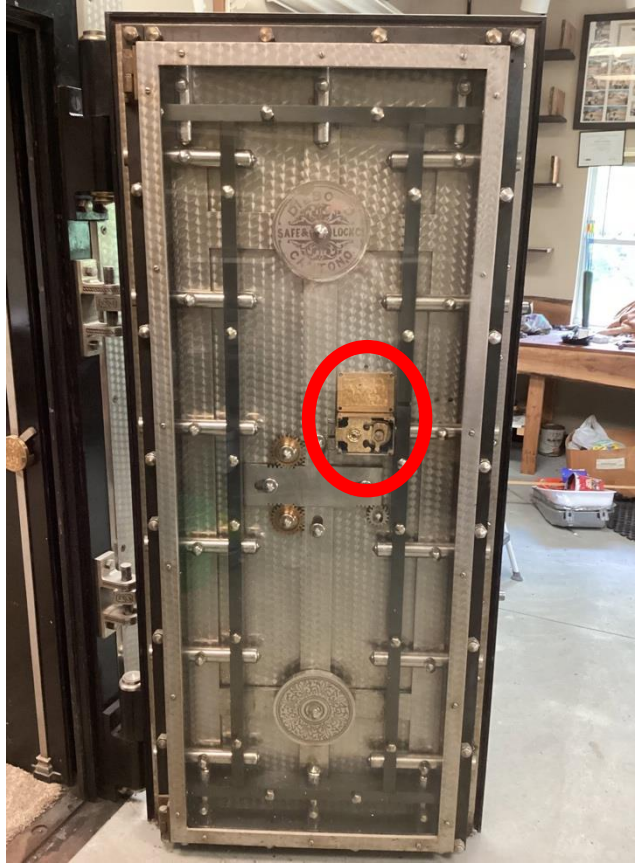


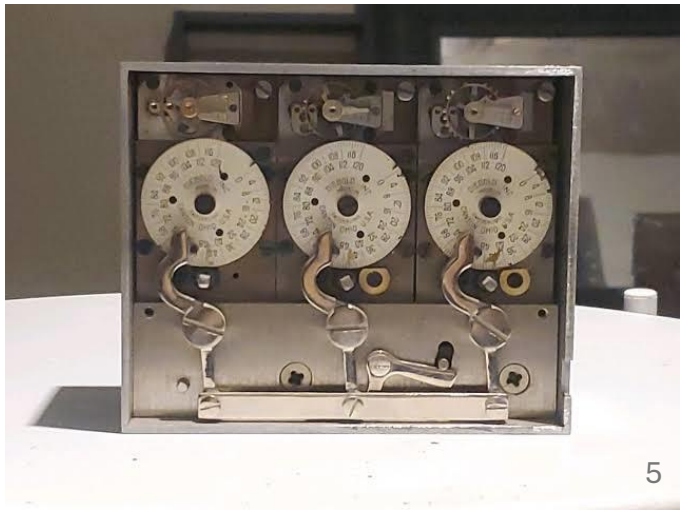
Timelock guarantees that data @ addr is immutable for the specified duration (δ)

Uses of Timelock

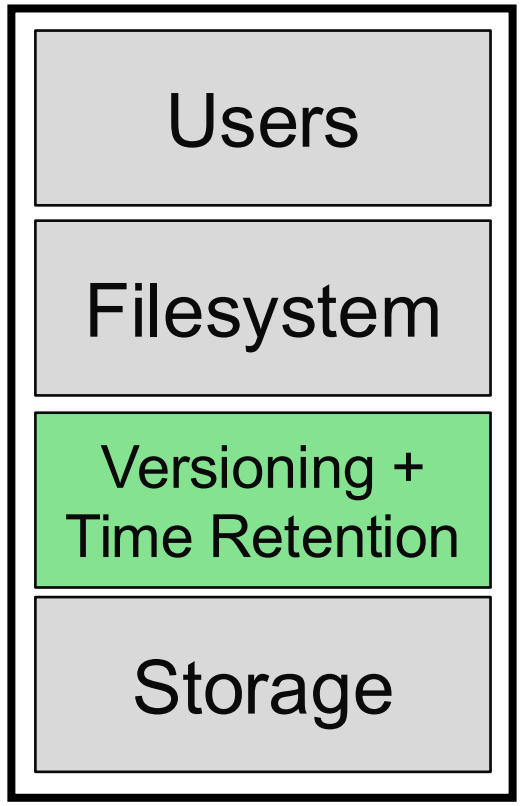
- Timelock forces attacker to wait
  - Waiting is expensive for attacker
  - Detect and recover during timelocked period

- Many uses
  - Ransomware defense
  - Tamper-proof logging
  - Regulatory data compliance (SOX, healthcare)





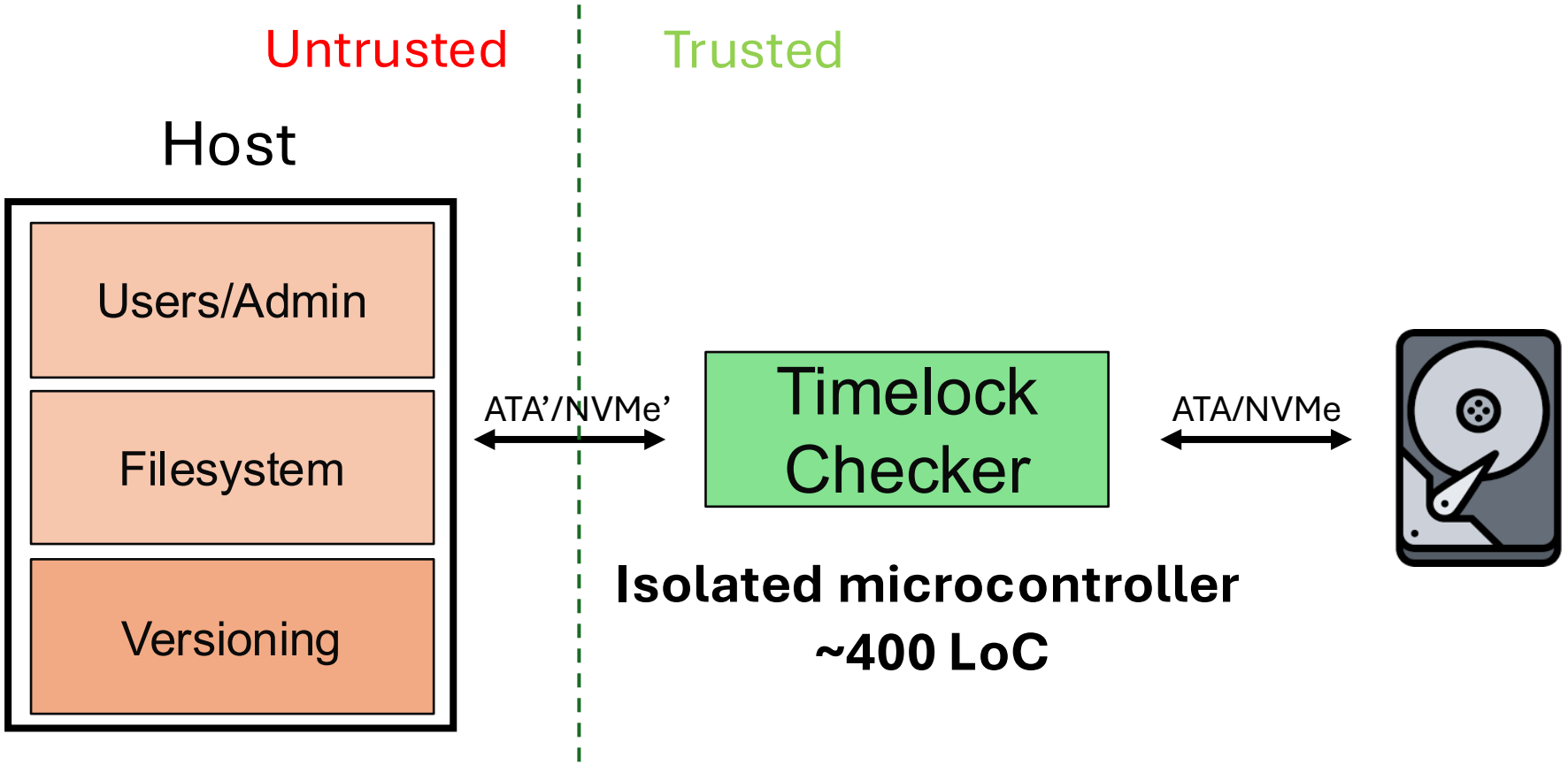
Time defense in versioning is not secure



Large TCB

Reidys, ASPLOS ‘22 demonstrated trimming attack

Goal: Separate Versioning and Timelock



- Enforced in small TCB
- Independent interface enables flexible policies

Contributions

How to build secure versioning using a timelock drive?

- **Challenge: can’t overwrite metadata**

How to realize a performant system?

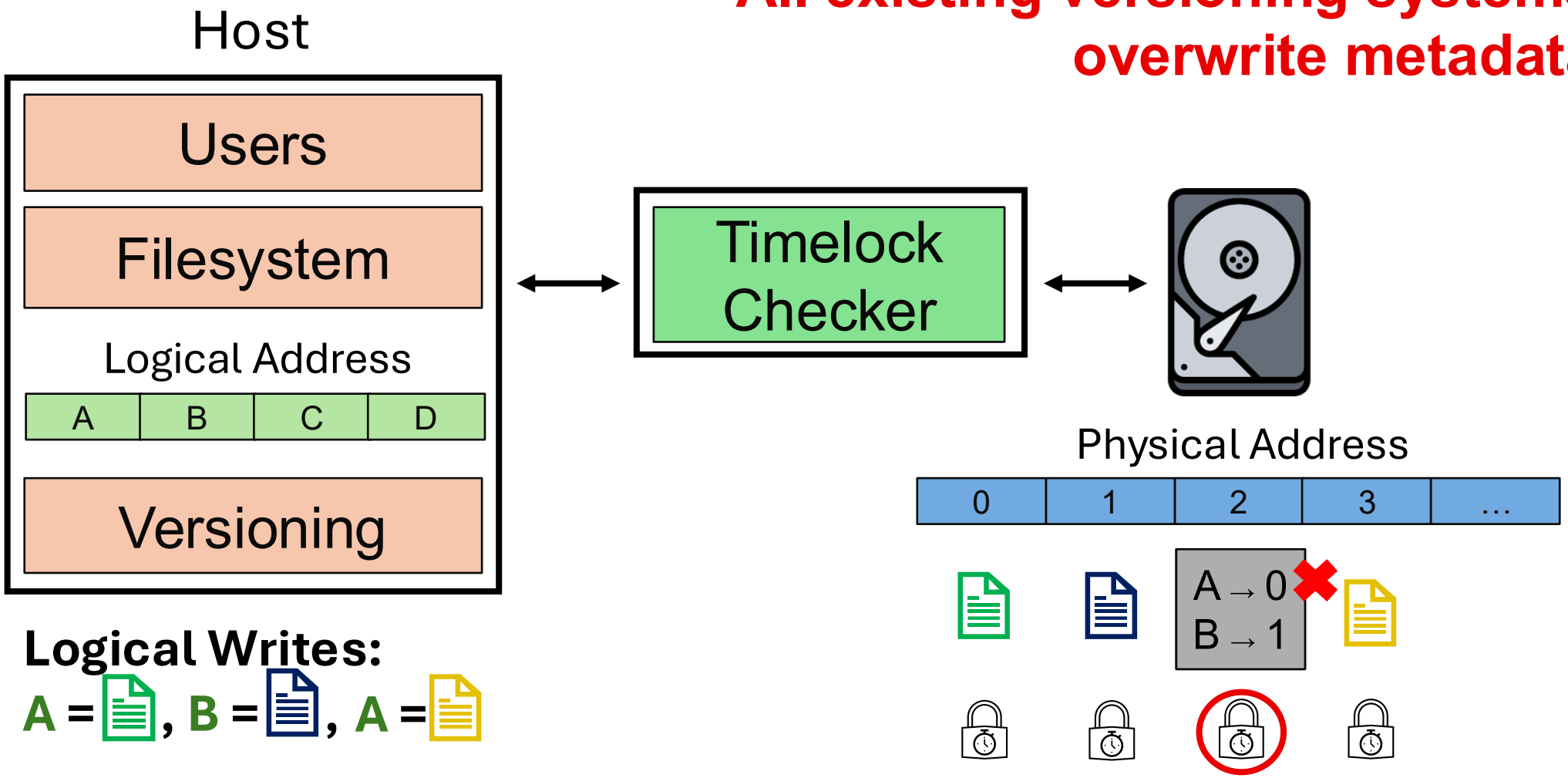
How to recover after an attack?

Evaluation

- Formally verified timelock checker
- Negligible overhead

Challenge: Version Metadata Updates

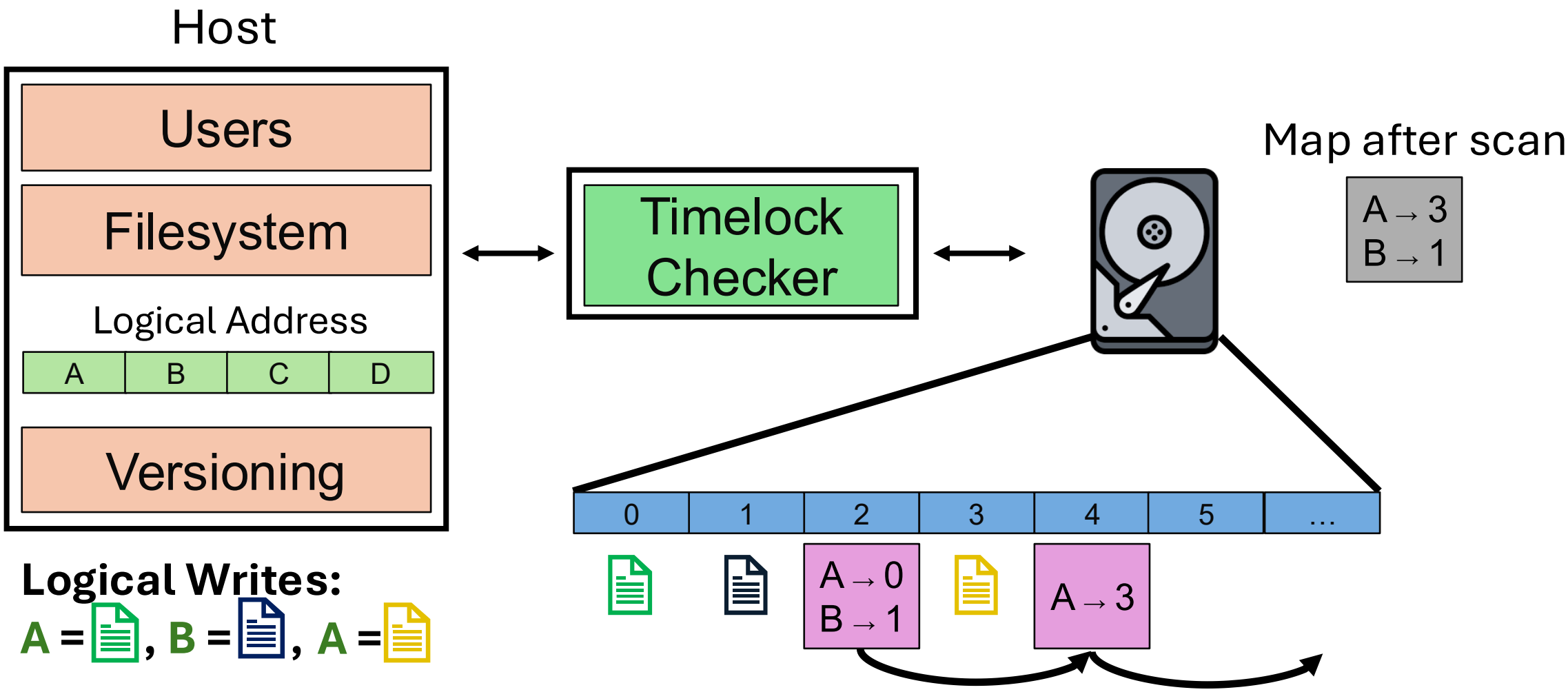
All existing versioning systems overwrite metadata



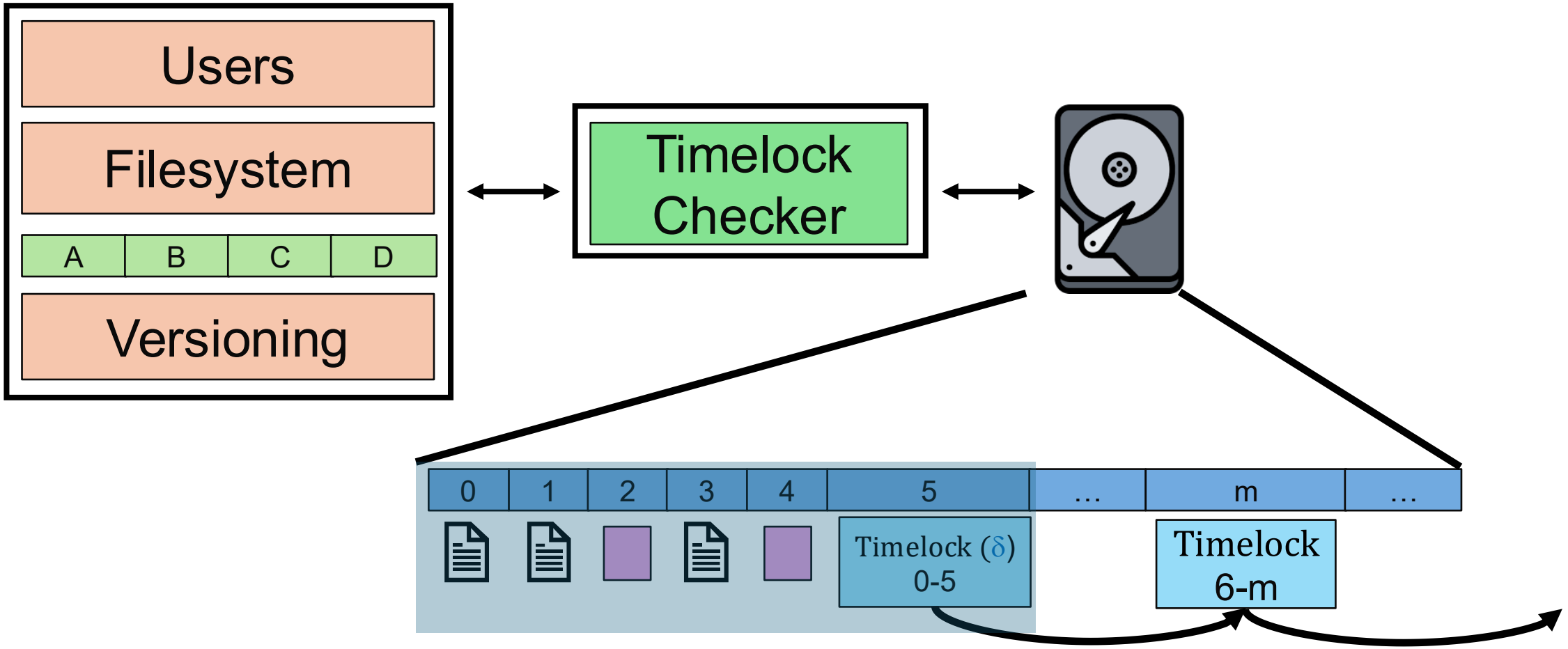
Logical Writes:

A = , **B =** , **A =**

Solution: Append-only Version Log



Solution: Append-only Timelock Log



See paper for details on garbage collecting logs!

Contributions

How to build a secure versioning using timelock drive?

- Challenge: can’t overwrite metadata

How to realize a performant system?

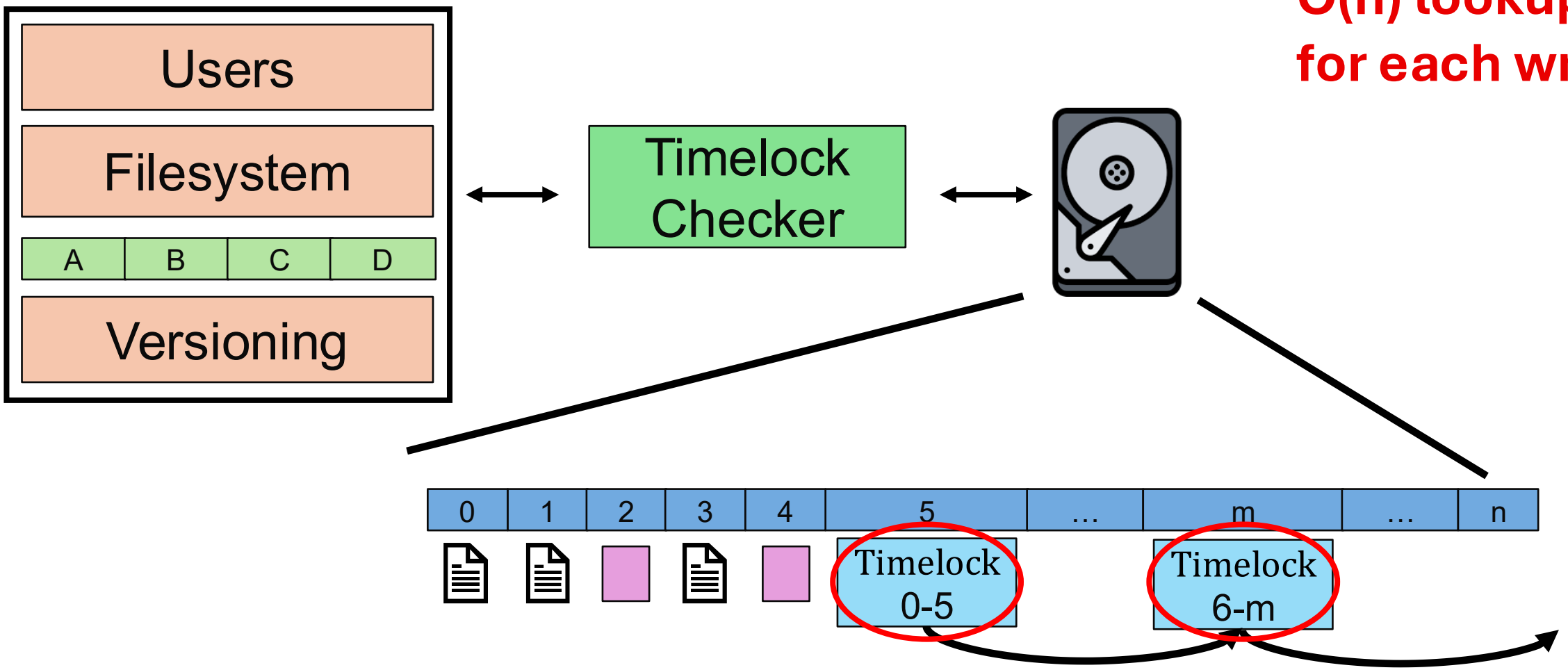
How to recover after an attack?

Evaluation

- Formally verified timelock checker
- Negligible overhead

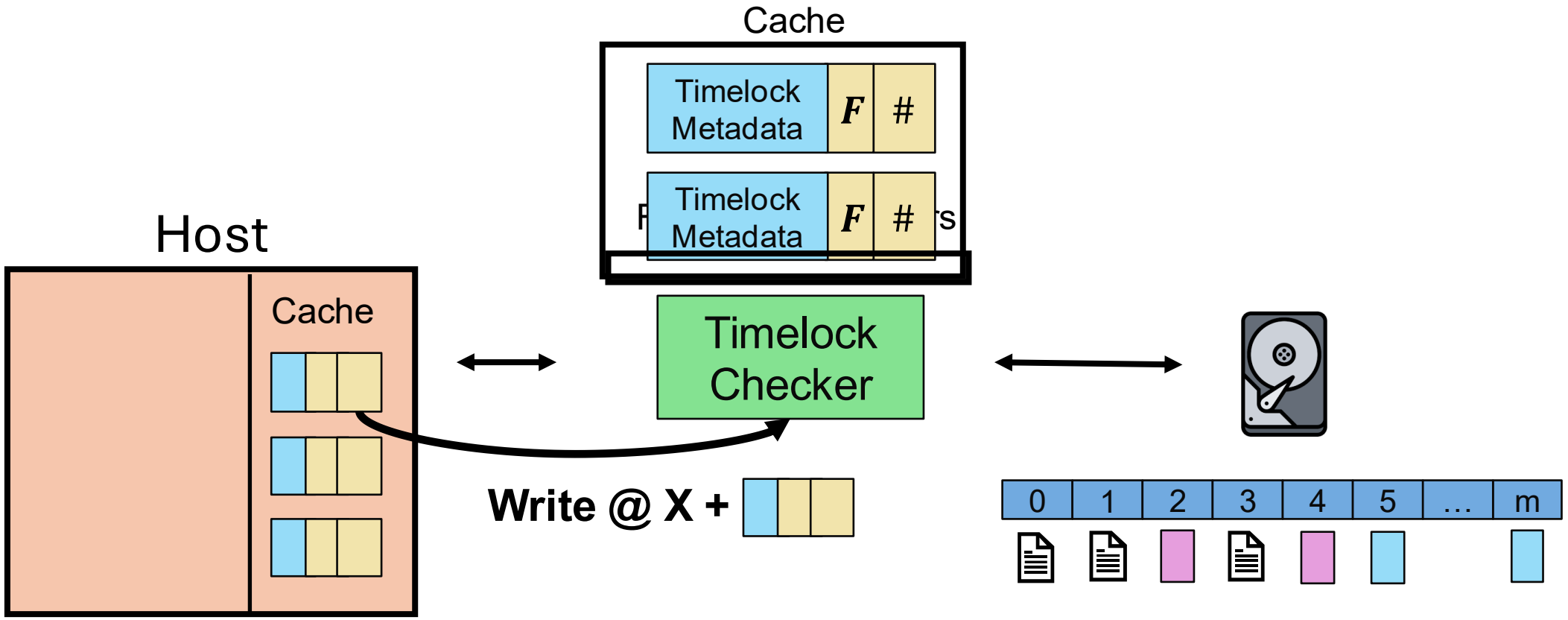
Challenge: Overhead of Scanning Metadata Log

O(n) lookup for each write!



Solution: Secure Delegated Host Caching

- Caching in checker is not feasible
  - 2 GB of cache per TB of storage
- Checker resource requirements: → **1 MB**



Contributions

How to build a secure versioning using timelock drive?

- Challenge: can’t overwrite metadata

How to realize a performant system?

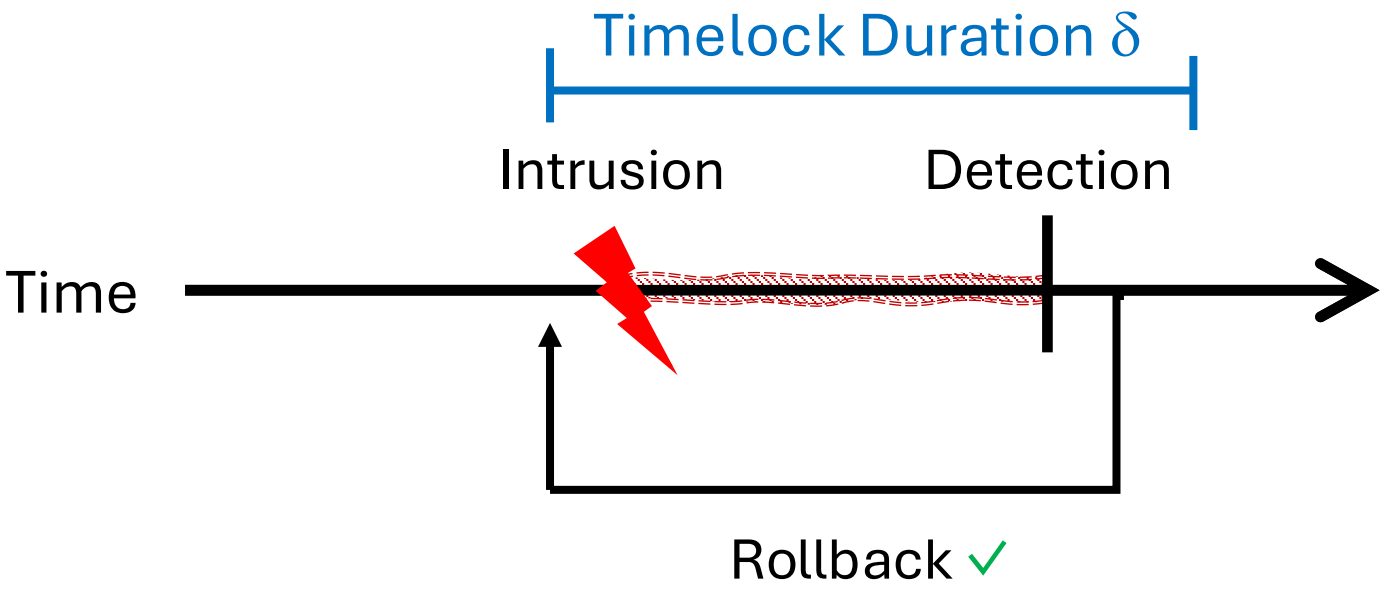
- Challenge: additional I/O for metadata access

How to recover after an attack?

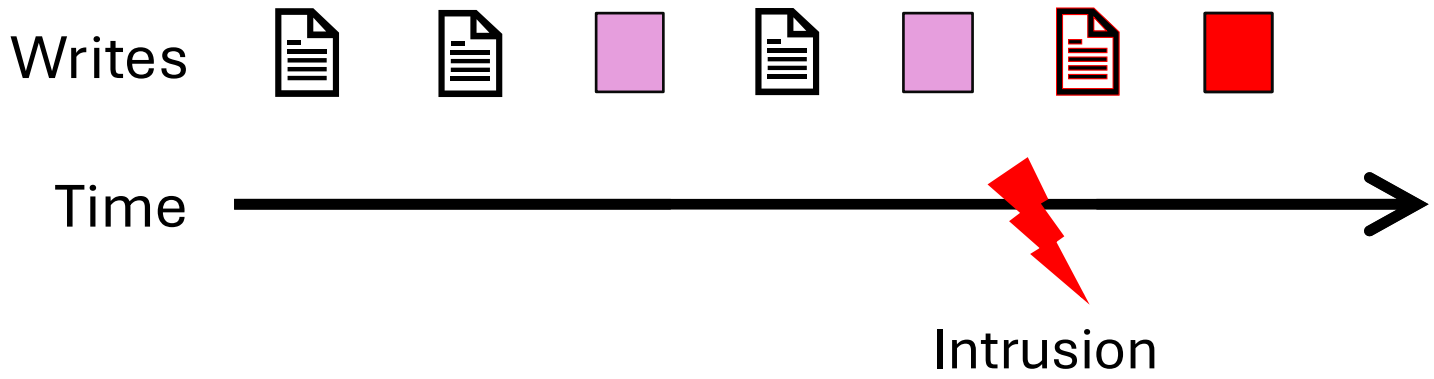
Evaluation

- Formally verified timelock checker
- Negligible overhead

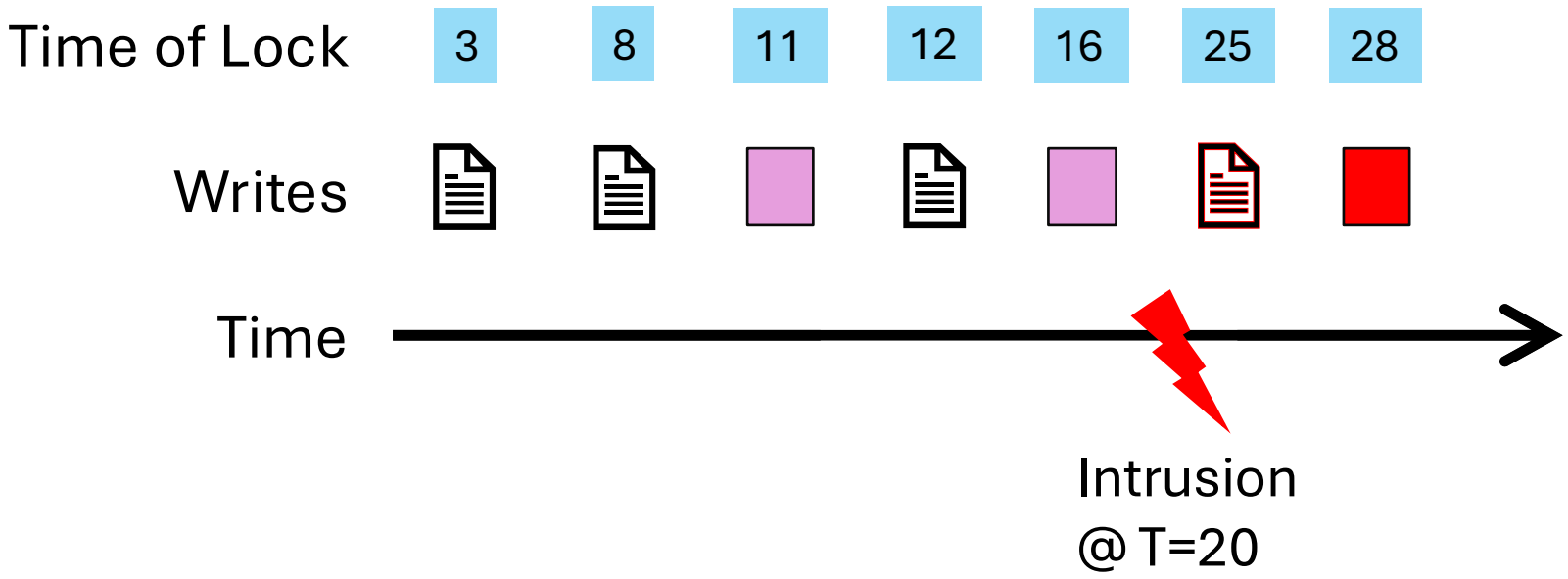
Recovery Process



Problem: Avoid Spoofed Metadata



Solution: Time of Lock



Contributions

How to build a secure versioning using timelock drive?

- Challenge: can’t overwrite metadata

How to realize a performant system?

- Challenge: additional I/O for metadata access

How to recover after an attack?

- Challenge: attacker may spoof version metadata

Evaluation

- Formally verified timelock checker
- Negligible overhead

Methodology



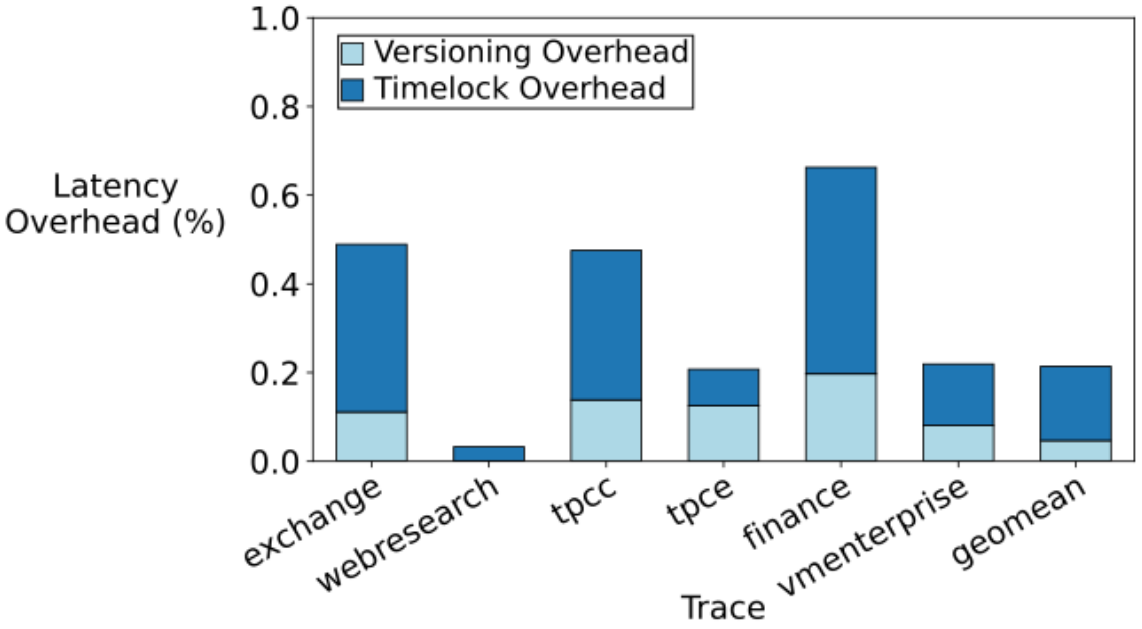
- Timelock checker written in ~400 Dafny LoC
  - Transpiled to Rust
- Custom versioning driver → checker → HDD/SSD
- Ransomware traces (Reidys, ASPLOS ‘22)
- Storage traces (SNIA IOTTA traces)
- Filesystem benchmarks (filebench)

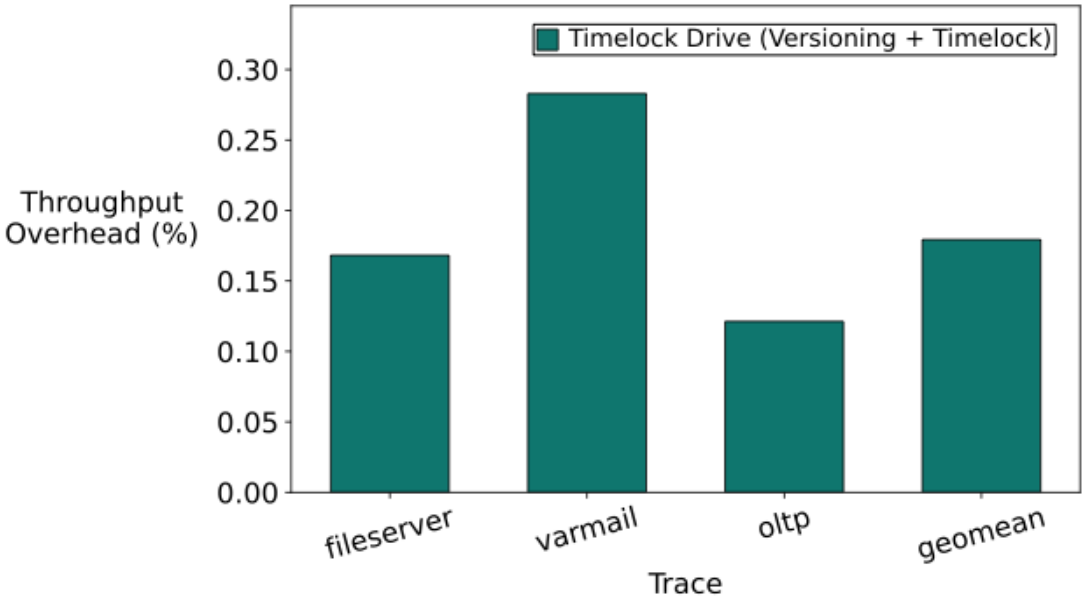
Security Evaluation

- Formally verified code obeys timelock behavior
- 100% success rate recovering from ransomware samples

Performance Evaluation

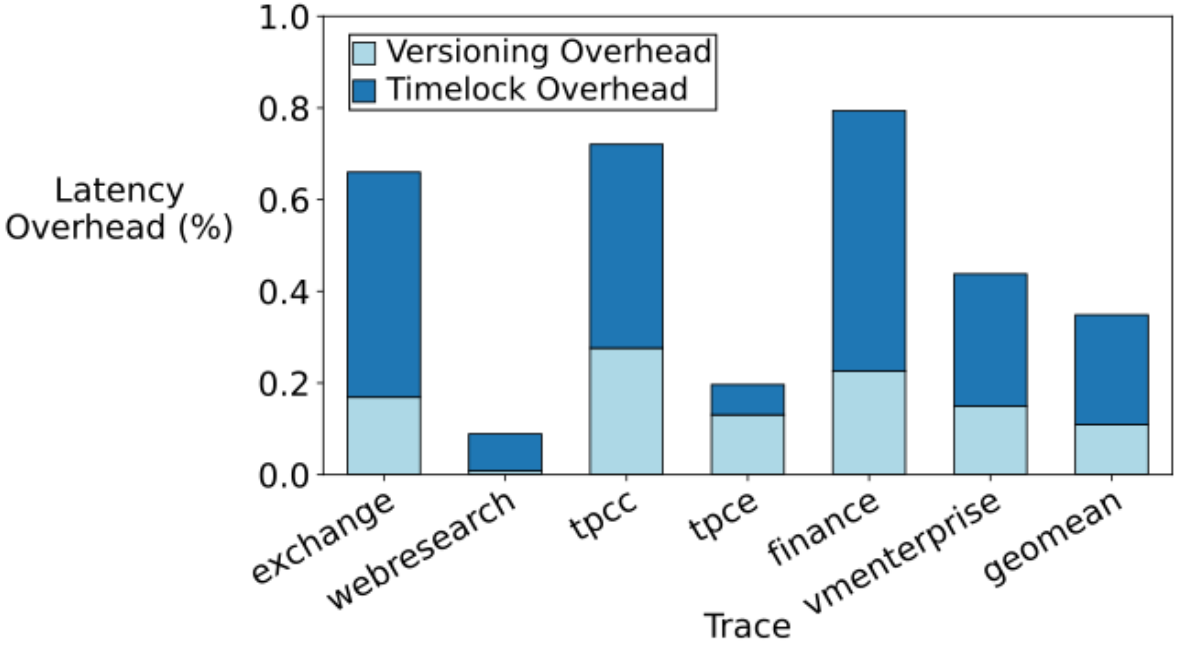
- Negligible overhead on trace and filesystem benchmarks

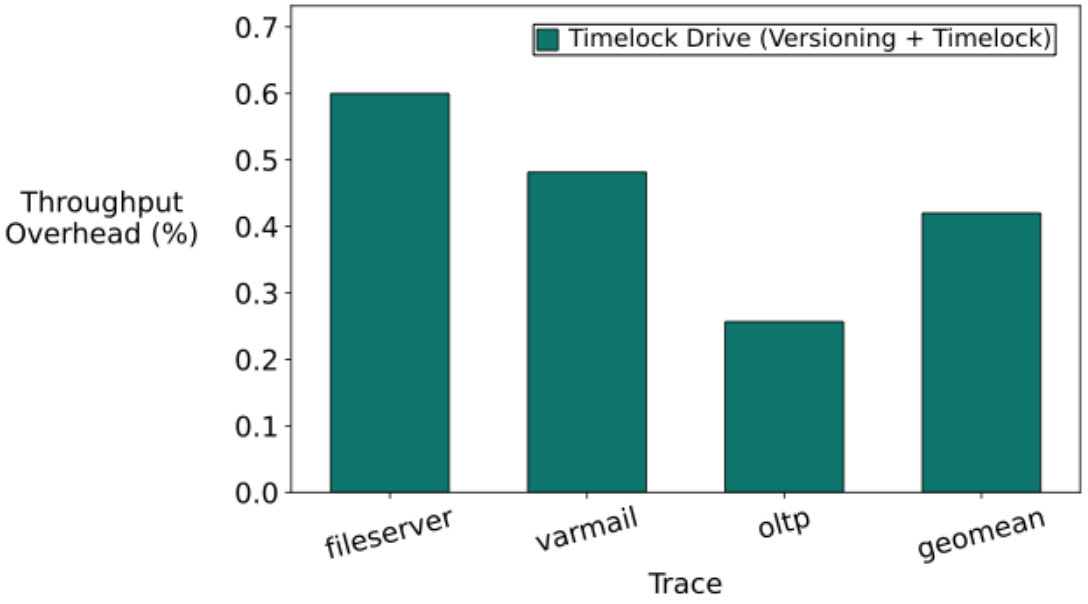




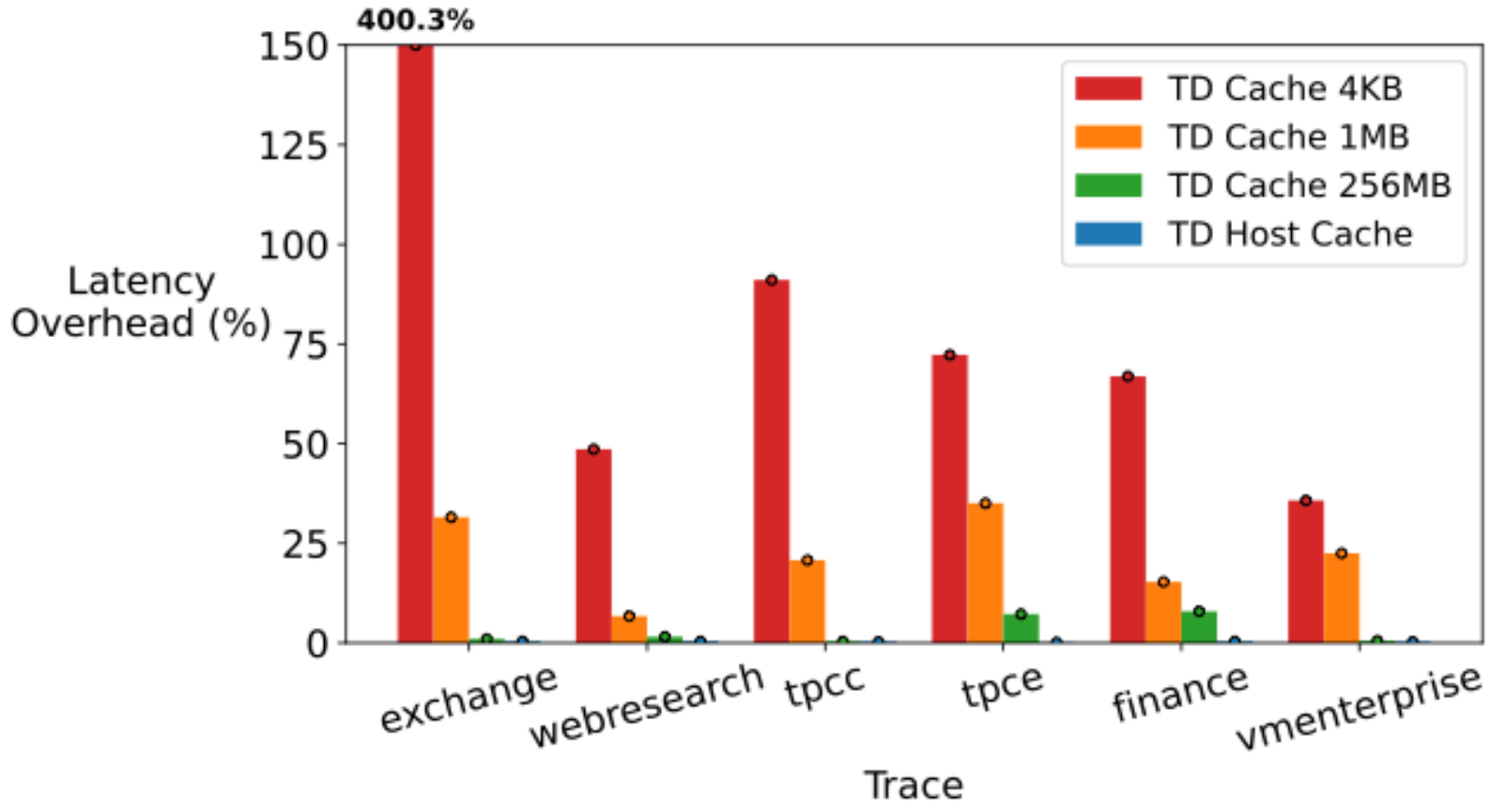
Performance Evaluation

- SSD results, slightly higher overhead still negligible





Secure Delegated Host Caching is Crucial



Summary

Objective: last-line for defense if credentials are compromised

Approach: timelock interface for storage

Deliverable: verified timelock checker + timelock-aware versioning driver

Broader Impact: A step toward security without identity



Thank You

Questions?



Artifact

Contact me:
Jonah Rosenblum
jonaher@umich.edu

Timelock Drive: Isolated Time-Based Defense for Storage Systems

University of Michigan