

V-ORAM: A Versatile and Adaptive ORAM Framework with Service Transformation for Dynamic Workloads

Bo Zhang[†], Helei Cui[†], Xingliang Yuan[◇], Zhiwen Yu^{†‡}, Bin Guo[†]

[†] *Northwestern Polytechnical University, China*

[◇] *The University of Melbourne, Australia*

[‡] *Harbin Engineering University, China*



Data encryption to achieve privacy

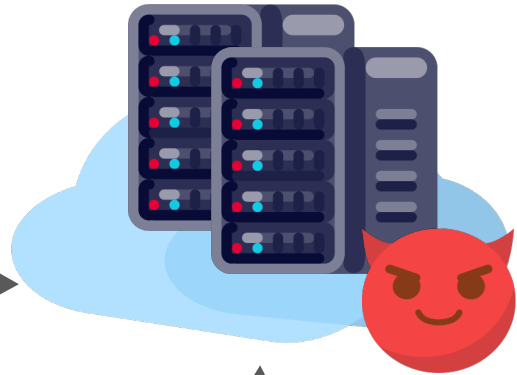


Key	Disease
1	COVID-19
2	Stroke
3	Alzheimer
4	Kidney diseases

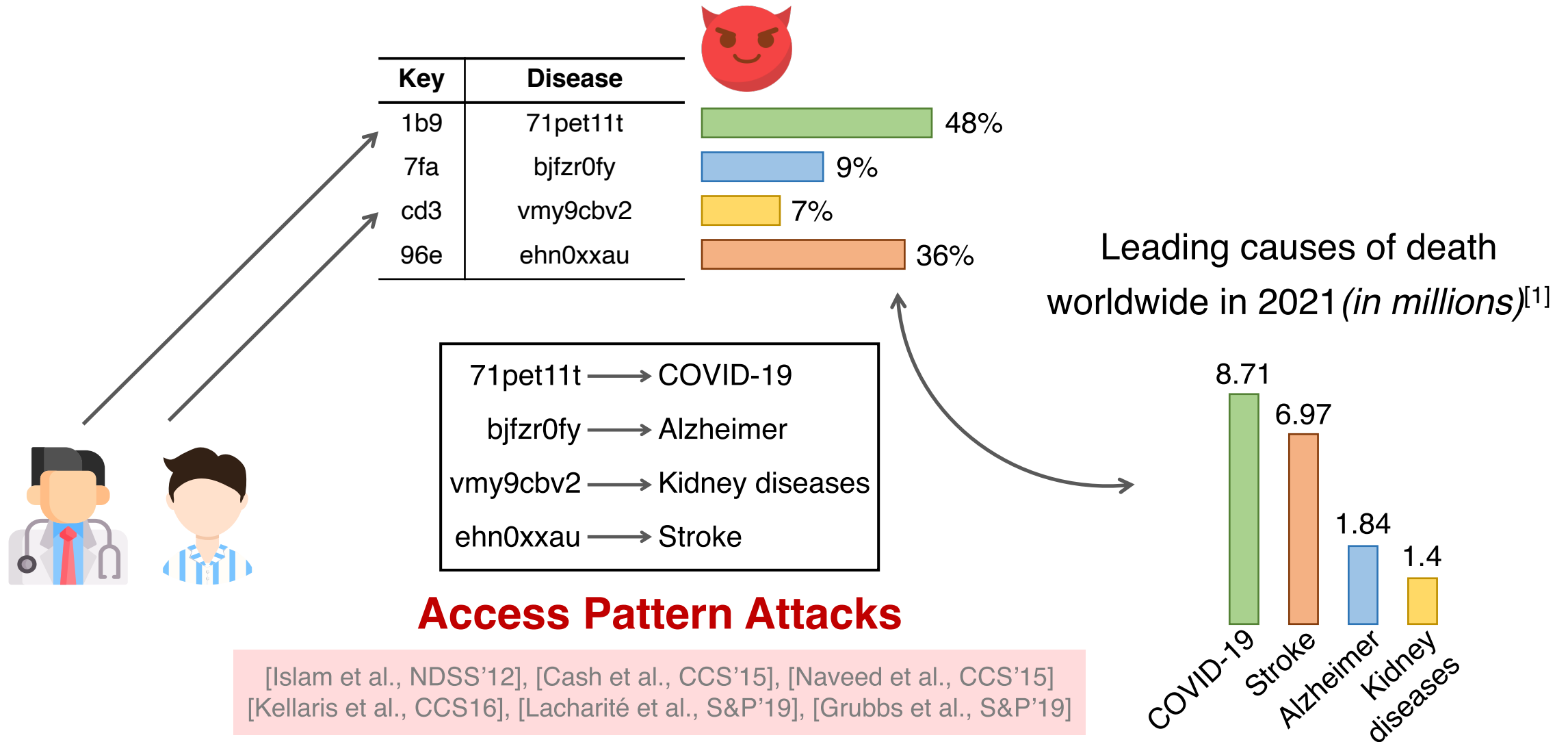


Key	Disease
1b9	71pet11t
7fa	bjfzr0fy
cd3	vmy9cbv2
96e	ehn0xxau

Honest-but-curious
adversary



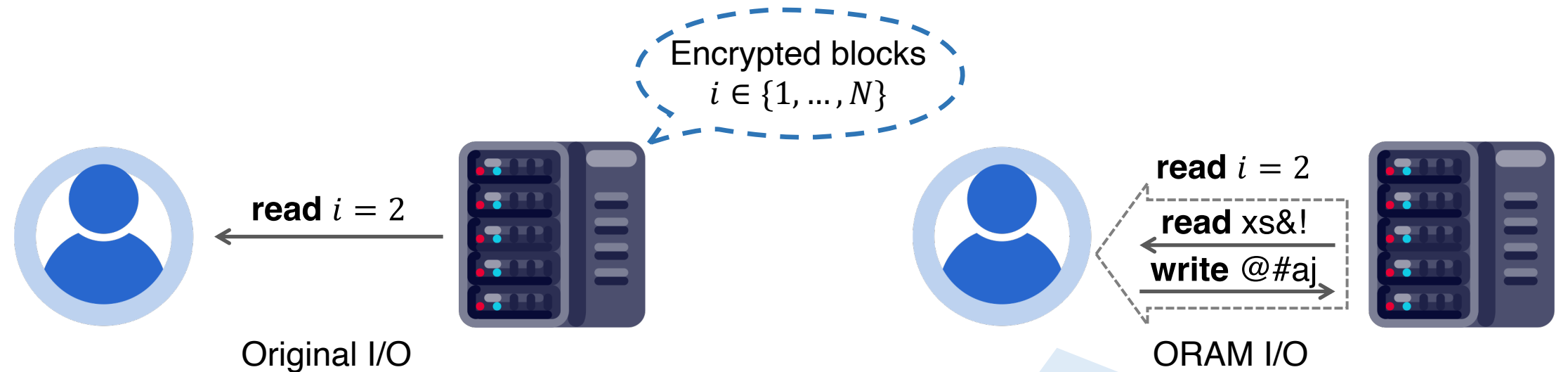
Encryption alone is **NOT** enough



[1] <https://www.statista.com/statistics/1488587/leading-causes-of-death-worldwide-2021/>

Oblivious RAM [GO, JACM'96]

- Goal: making the generated access pattern **random**



Applications to various scenarios

- Parallelization: [ConcurORAM, NDSS'19], [TaoStore, S&P16]
- Secure computation: [SCORAM, CCS'14], [Circuit ORAM, CCS'16]
- Searchable encryption: [OBI, NDSS'23], [AM, EUROCRYPT'23]
- ...

Remark

- **Stateful**, maintains auxiliary metadata
- $\Omega(\log N)$ costs lower bound
- I/O operations need encryption key

Tailored and optimized for specific workload

Serve for dynamic workloads

- ORAM faces dynamically changing workloads in practice

Electronic medical record system



Routine usage

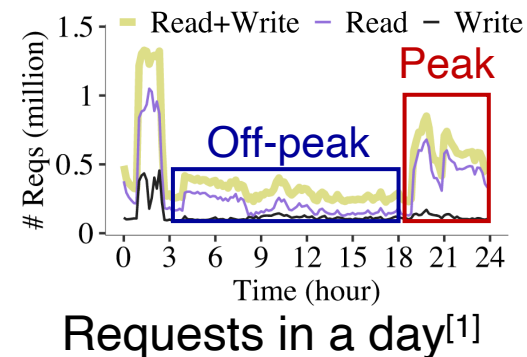
- No specific performance constraint
- Lower maintenance cost



Medical analysis

- Keyword search
- More complex query (join, aggregate)

Cloud storage system



Peak

- Higher throughput
- Lower latency

Off-peak

- Lower throughput
- Higher latency

- Or caused by system performance changes (e.g., bandwidth)

[1] Jinhong Li, Qiuping Wang, and Patrick P. C. Lee. "An In-depth Comparative Analysis of Cloud Block Storage Workloads: Findings and Implications", ACM TOS, 2023.

Serve for **dynamic workloads**

- Using **unmatched** ORAM scheme is not a good option
 - Sync. ORAM in async. settings → **Harm correctness**
 - Comm.-centric ORAM in low bandwidth → **Downgrade performance**
- Two strawman solutions:
 - Re-build the entire database → **$O(N)$ comm. costs**
 - Maintain multiple instances → **Heavy storage/access costs**

Can we achieve both security and efficiency?

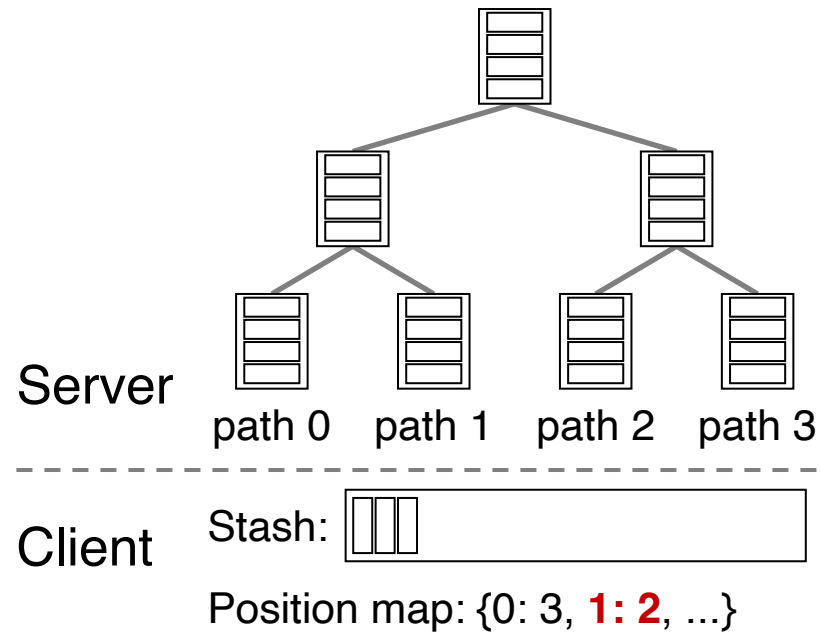
This paper & this talk

- **V-ORAM**, a versatile and adaptive ORAM framework
 - Securely transform between three tree-based ORAMs
[Path ORAM, CCS'13], [Ring ORAM, Security'15], [ConcurORAM, NDSS'19]
 - **Constant** transformation costs
- Secure **ORAM service transformation (OST)** protocol
 - Identify **two leakages** during transformations
 - Mitigation with **constant** communication costs
- Heuristic **planner** to help choose parameters
- Implementations & Evaluations

[See paper](#)

Workflow of considered ORAMs

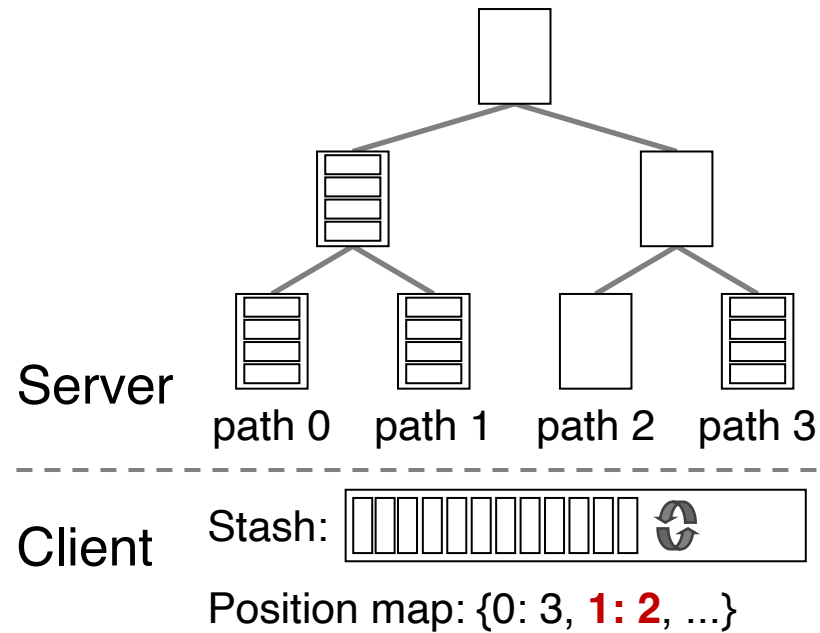
- Path ORAM, Ring ORAM, and ConcurORAM (built upon Ring ORAM)



Path ORAM

Workflow of considered ORAMs

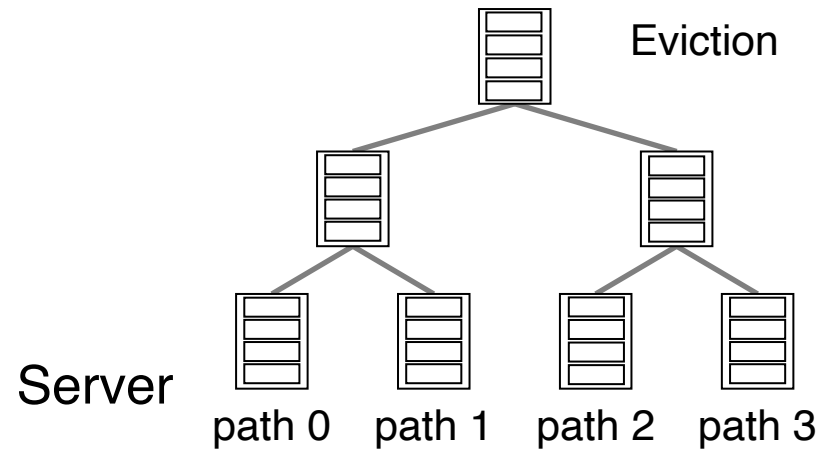
- Path ORAM, Ring ORAM, and ConcurORAM (built upon Ring ORAM)



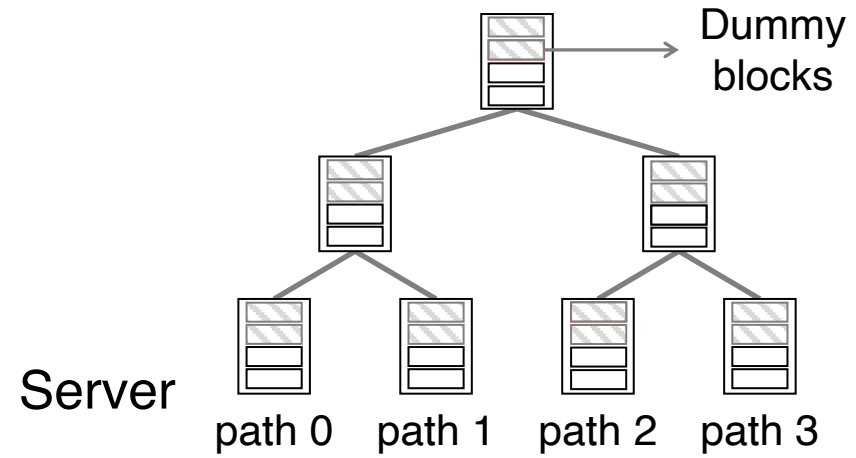
Path ORAM

Workflow of considered ORAMs

- Path ORAM, Ring ORAM, and ConcurORAM (built upon Ring ORAM)



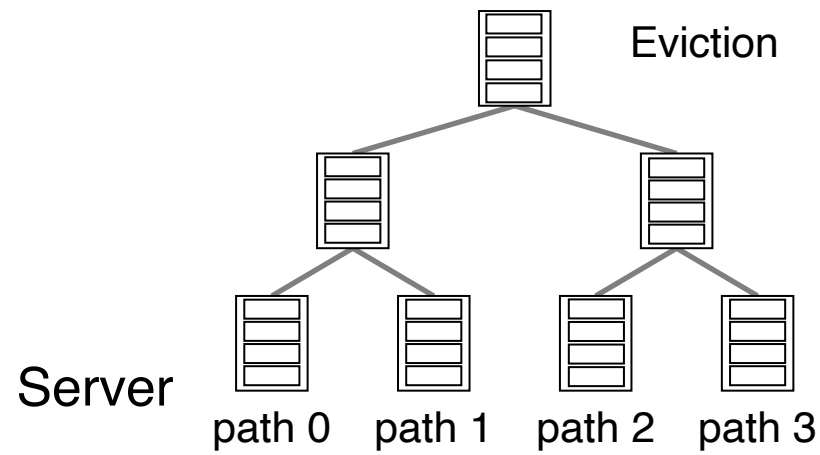
Path ORAM



Ring ORAM

Workflow of considered ORAMs

- Path ORAM, Ring ORAM, and ConcurORAM (built upon Ring ORAM)



Server

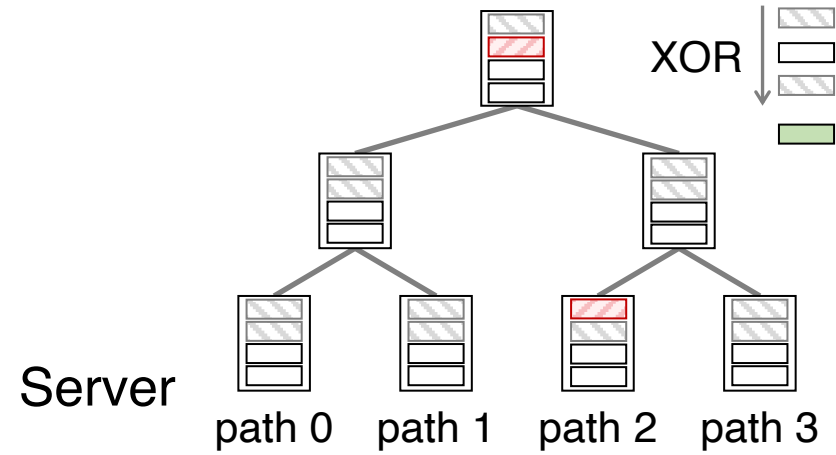
path 0 path 1 path 2 path 3

Client

Stash:

Position map: {0: 3, **1: 0**, ...}

Path ORAM



Server

path 0 path 1 path 2 path 3

Client

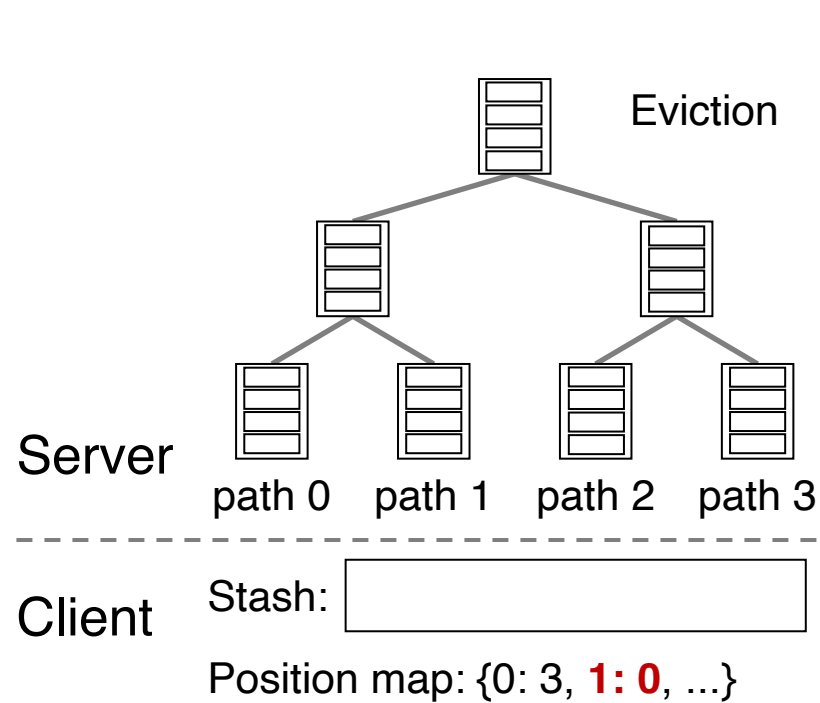
Stash:

Position map: {0: 3, **1: 2**, ...}

Ring ORAM

Workflow of considered ORAMs

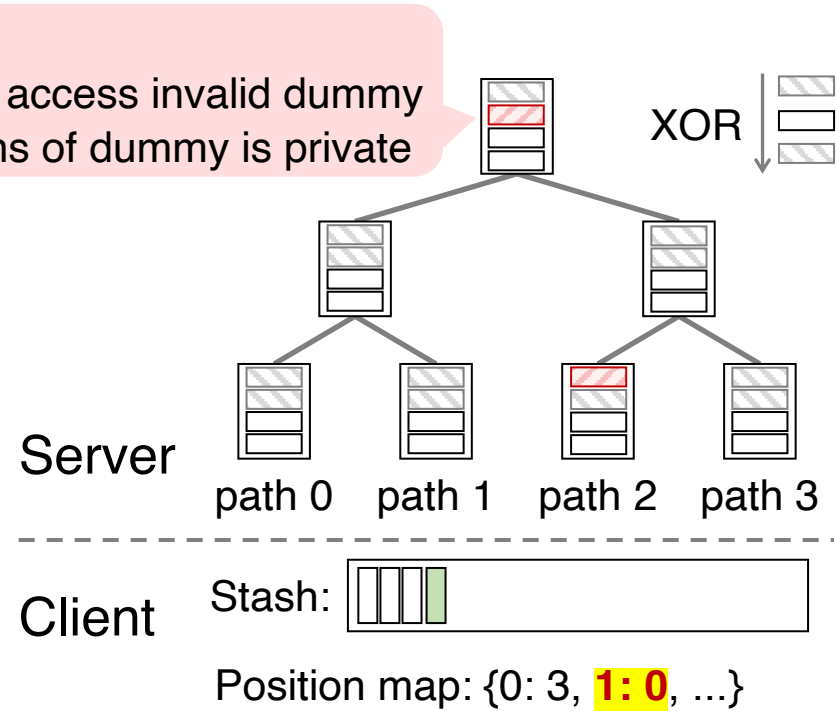
- Path ORAM, Ring ORAM, and ConcurORAM (built upon Ring ORAM)



Path ORAM
Instant eviction

Remark

- Cannot access invalid dummy
- Positions of dummy is private



Ring ORAM
Periodic eviction

EvictPath for every A queries

Why we choose these ORAMs?

- Workloads in practice
 - General storage: Path ORAM, [rORAM, NDSS19]
 - Real-time updates: Ring ORAM, ConcurORAM
 - Parallel access: [Snoopy, SOSP21], ConcurORAM, [TaoStore, S&P16]
 - Resource constrained: [FutORAMa, CCS23], [CSCL11], Path ORAM (recursive)
 - Specialized functionality: [OBI, NDSS23], [DUORAM, Security23], ConcurORAM
 - ... (see our paper for detailed taxonomy)

Why we choose these ORAMs?

- Workloads in practice
 - General storage: **Path ORAM**, [rORAM, NDSS19]
 - Real-time updates: **Ring ORAM**, **ConcurORAM**
 - Parallel access: [Snoopy, SOSP21], **ConcurORAM**, [TaoStore, S&P16]
 - Resource constrained: [FutORAMa, CCS23], [CSCL11], **Path ORAM** (recursive)
 - Specialized functionality: [OBI, NDSS23], [DUORAM, Security23], **ConcurORAM**
 - ... (see our paper for detailed taxonomy)

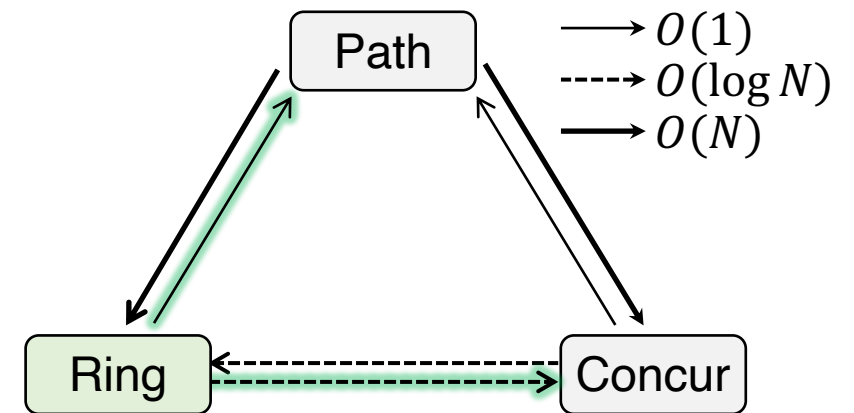
Our considered ORAMs cover the general workloads

System model & treat assumptions

- **Honest** client and **semi-honest** server
- During transformations, the adversary can learn:
 - Historical accesses
 - Access status of blocks
 - API calls of ORAMs
 - Encrypted data, metadata, and communication channel
- Accesses within each ORAM are secure and correct
- Attacks that can not be defended by original ORAMs

Design: select a base ORAM

- Transformation protocol for arbitrary ORAMs? $\rightarrow O(k^2)$ protocols
- Find a transfer **inter-media**
 - Compatible to other schemes
 - Affordable, no heavy comp./comm.
 - Transformation-efficient, avoid massive comp./comm.
- Ring ORAM as **base ORAM** ✓
 - Compatible access protocol
 - Light-weighted XOR
 - Re-use the dummy blocks



Design: handle the transformations

- Switch with ConcurORAM

- **Inherit** the dummy blocks \longrightarrow **Potential leakage**
- Download the server-side metadata

- Switch with Path ORAM

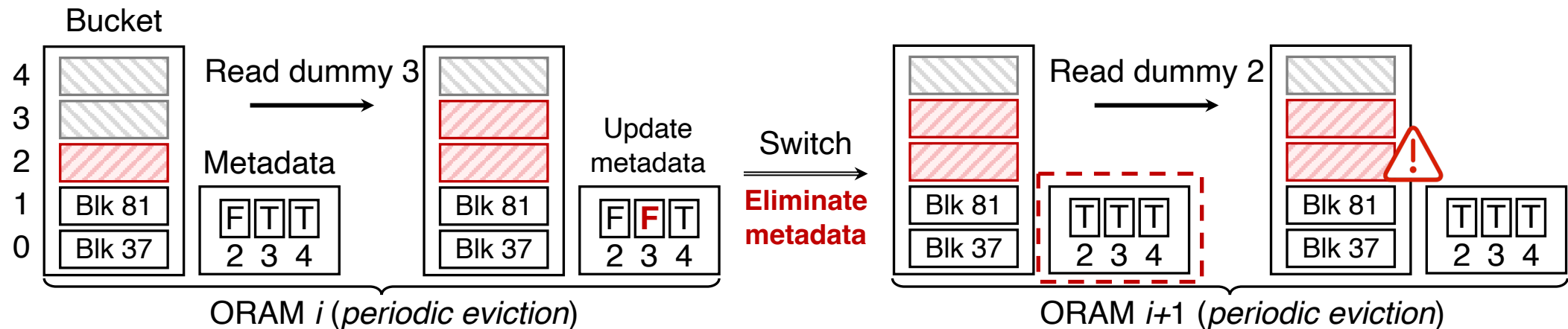
- Directly inherit stash and position map

- **What about the dummy blocks?**

- Option 1: treat dummies as real blocks \longrightarrow **✗** \longrightarrow **Increase the costs**
- Option 2: only access real blocks \longrightarrow **Potential leakage**

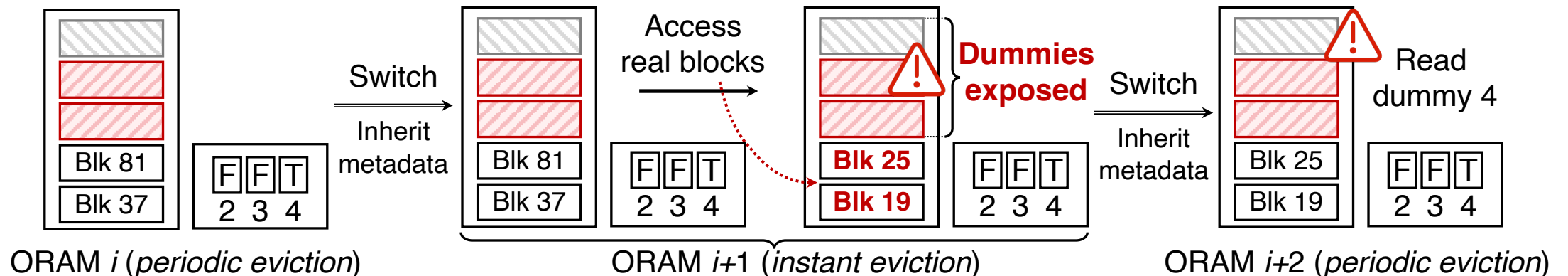
Design: identify the leakages

- Re-accessing invalid dummies
 - During transformation between **periodic eviction**



Design: identify the leakages

- Re-accessing invalid dummies
 - Between ORAMs with **periodic eviction**
- Exposing dummies' positions
 - Between ORAMs with **instant and periodic eviction**

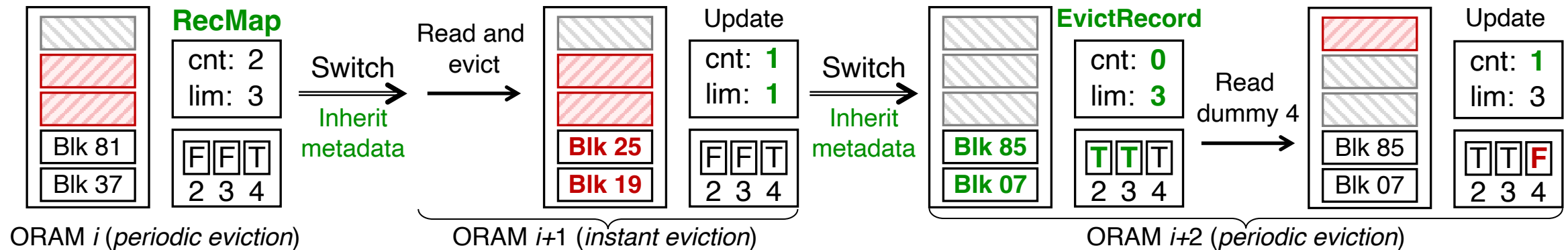


Design: mitigate the leakages

- Intuition: evict the accessed buckets before using dummies
- Access limit: number of secure accesses before eviction
 - **1** for Path ORAM, **S** (number of dummies) for Ring ORAM
 - Evict the bucket before exceed the limit
- Record map: $\{bucket\ ID \rightarrow (cnt, lim)\}$
 - *cnt*: access count, *lim*: current limit
- EvictRecord: evict the bucket when $cnt = lim$
 - Triggered **before access** and **after eviction**

See paper

Design: mitigate the leakages



Theoretical bounds

- Costs of EvictRecord
 - For an ORAM with N blocks, bucket size of Z , and block size of B
 - Comm. and comp. costs of $O(ZB)$, storage costs of $N \log Z$ bit
- Stash size
 - The stash size of V-ORAM is bounded by $O(\log N)$ blocks
- Transformation costs
 - c : batch size of Ring ORAM, MS : max stash size

	Switch to base ORAM		Switch to base ORAM	
	Comm.	Comp.	Comm.	Comp.
Path ORAM	$O(1)$	$O(1)$	$O(1)$	$O(1)$
ConcurORAM	$B(2c^2 + c + MS)$	$B(2c^2 + c + MS)$	$B(c + MS)$	$B(c + MS)$

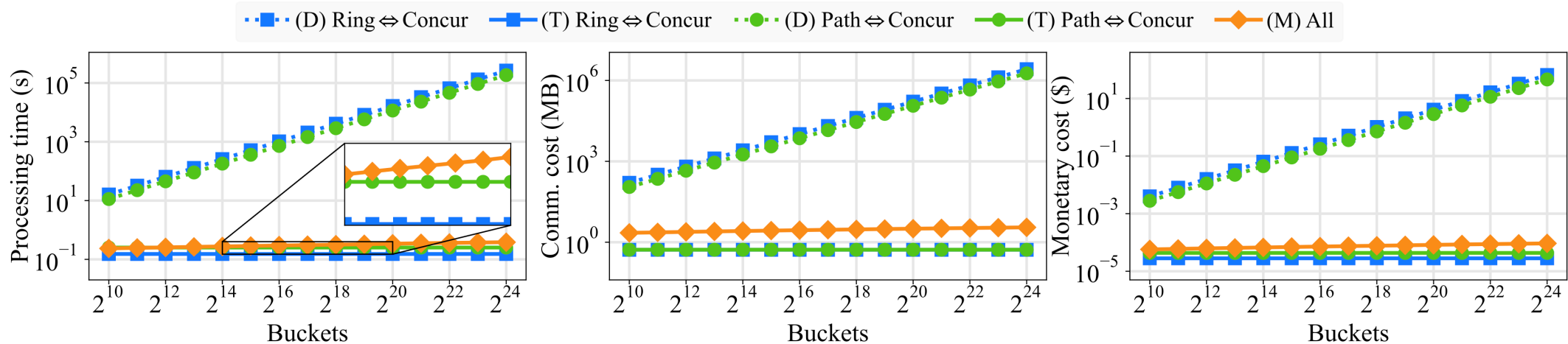
Implementations

- Open-source code at github.com/BoZhangCS/V-ORAM
 - V-ORAM with Path ORAM, Ring ORAM and ConcurORAM
 - “pycryptodemo” for data encryption
- Experimental setup
 - Mac mini with 512GB storage and an M2 chip with 16GB RAM
- Real-world datasets
 - Microsoft Research Cambridge (MSRC)
 - Alibaba cloud trace (AliCloud)
 - Twitter workload, ChestX-ray8, COVIDx



Constant transformation costs

- **(D)**: denotes downloading and re-building the ORAMs $\longrightarrow O(N)$
- **(M)**: denotes maintaining multiple ORAMs $\longrightarrow O(\log N)$
- **(T)**: denotes transformation via OST protocol $\longrightarrow O(1)$



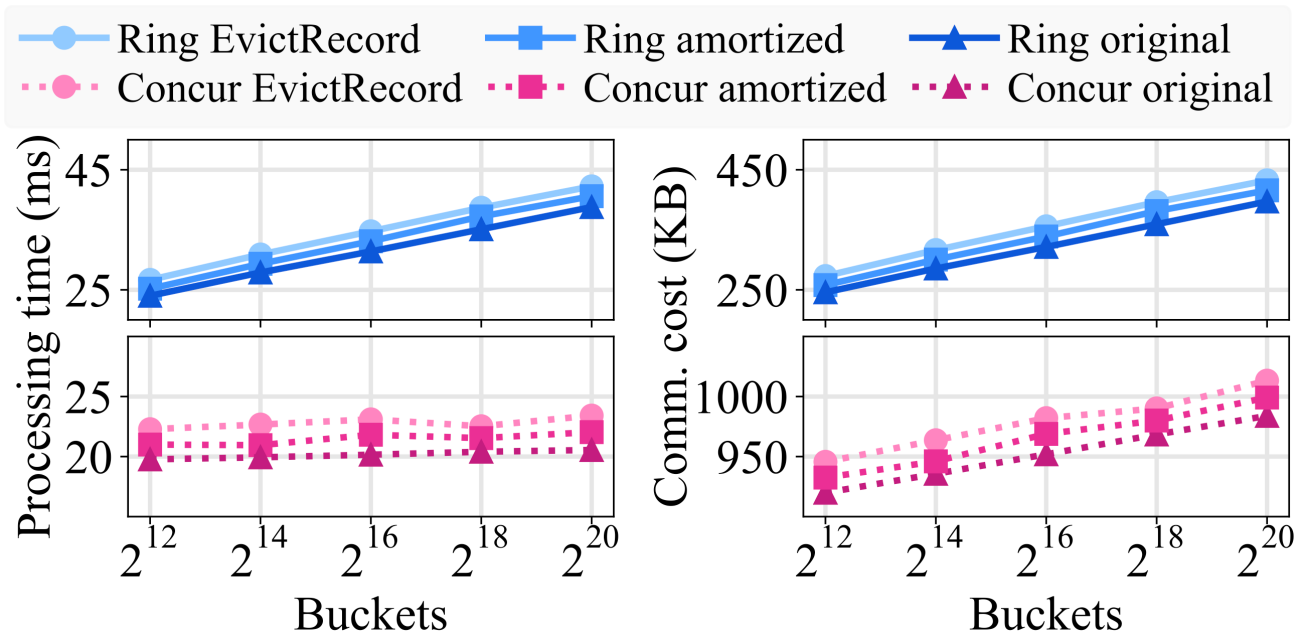
\uparrow 13,000x faster

The more data we store, the more benefit we gain

Constant EvictRecord costs

- **EvictRecord**: Eviction costs of each EvictRecord
- **Amortized**: Average costs in V-ORAM
- **Original**: Costs of ORAM without OST

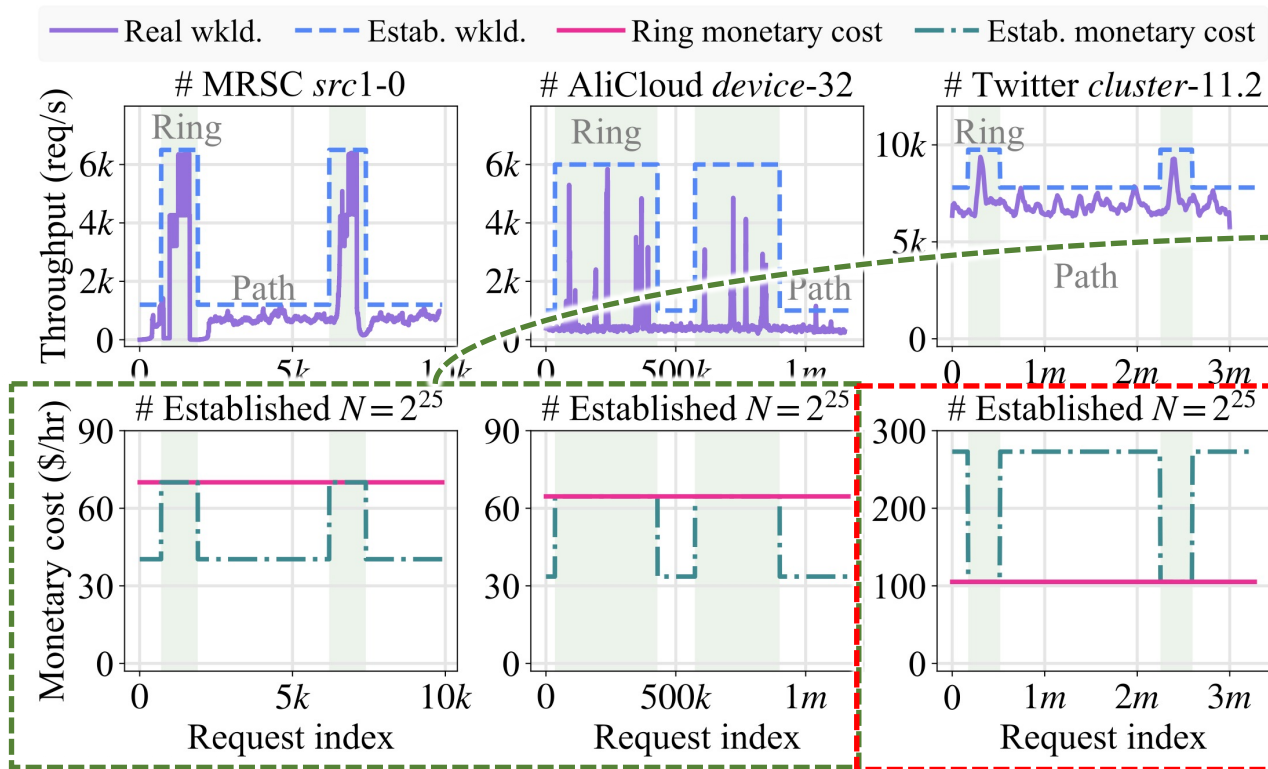
EvictRecord > Amortized > Original



**Additional
processing time < 5ms
comm. costs < 50kB**

Real-world case study

- Stripped workload from MRSC, AliCloud and Twitter
 - Divide workload into peaks and off-peaks
 - Ring ORAM for peaks, Path ORAM for off-peaks



Saving monetary costs of 33.1% and 24.5%

Increased costs

Suitable for workloads with higher variation

Conclusion

A versatile and adaptive ORAM framework, V-ORAM

- Secure and efficient ORAM transformation

Performs well

- 13,000x faster than strawman solutions
- Constant additional costs of $< 5\text{ms}$ and $< 50\text{kB}$
- Saves up to 33.1% costs in real-world datasets



bo.zhang@mail.nwpu.edu.cn