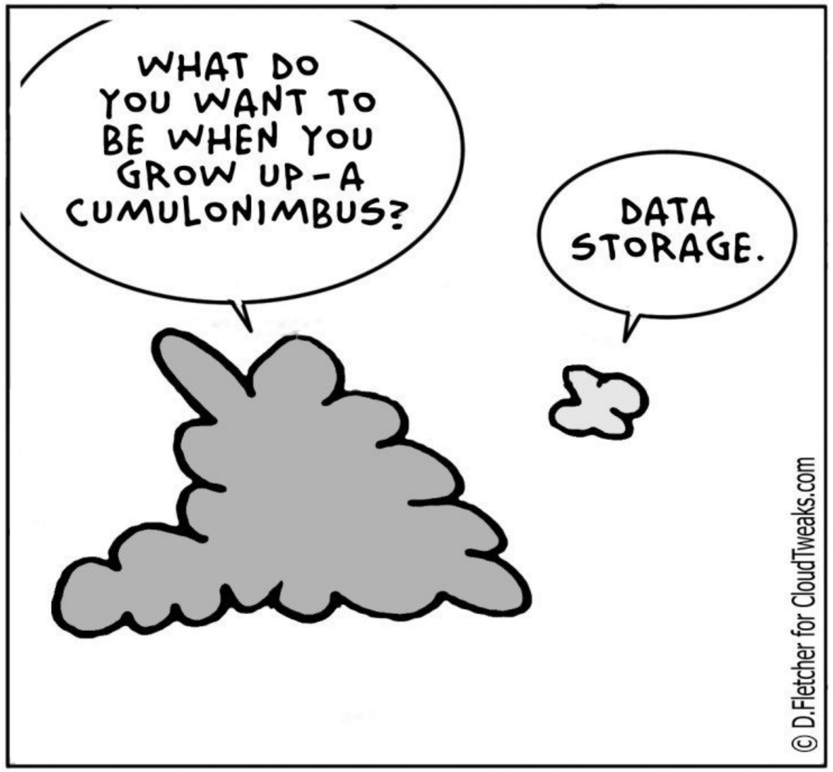
rORAM: Efficient Range ORAM with Locality

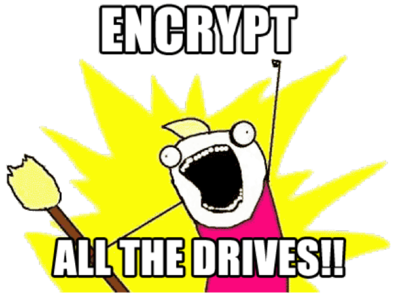
Anrin Chakraborti, Radu Sion
Stony Brook University

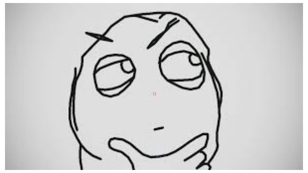
Adam Aviv**,** Seung Geol Choi, Travis Mayberry, Daniel Roche
United States Naval Academy

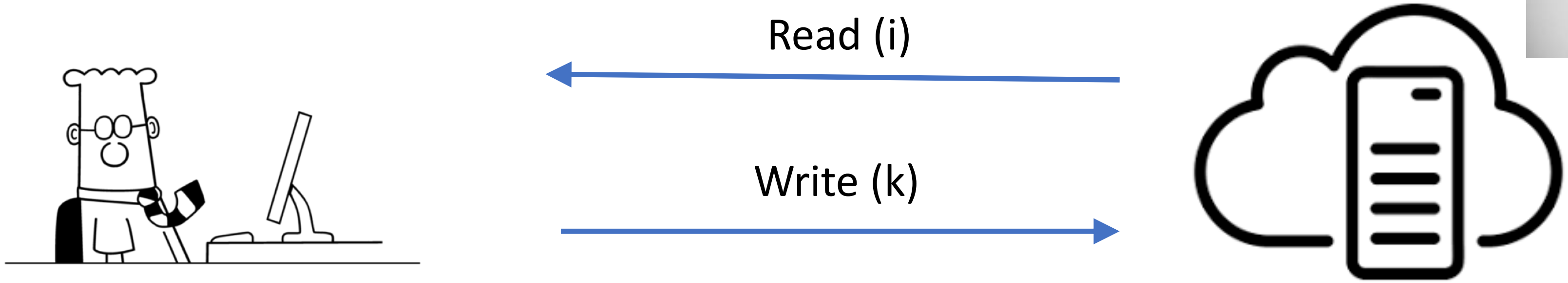
Its all about the Clouds!



Protecting Outsourced Data

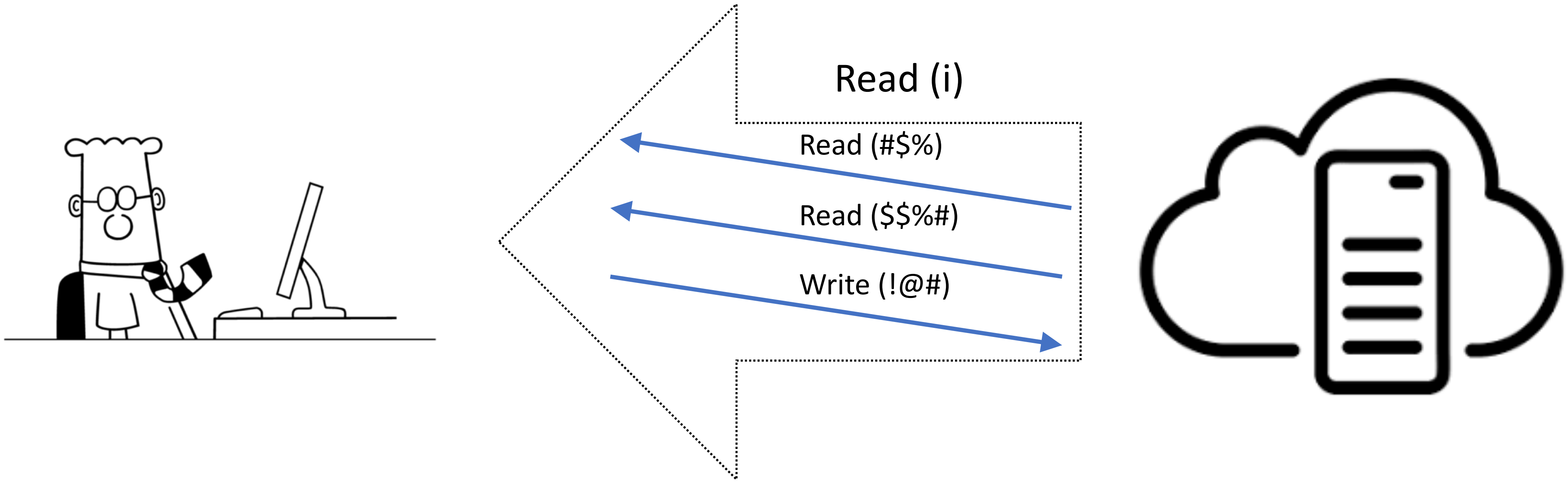






- Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. Islam et al. NDSS, ‘12
- Connecting the Dots: Privacy Leakage via Write-Access Patterns to the Main Memory. John et al. HOST, ’17
- …

Oblivious RAM (ORAM)



Observing the physical memory accesses, an adversary cannot learn

1. Which item has been accessed.
2. What operation has been performed.

Path ORAM [Stefanov et al. CCS '13]

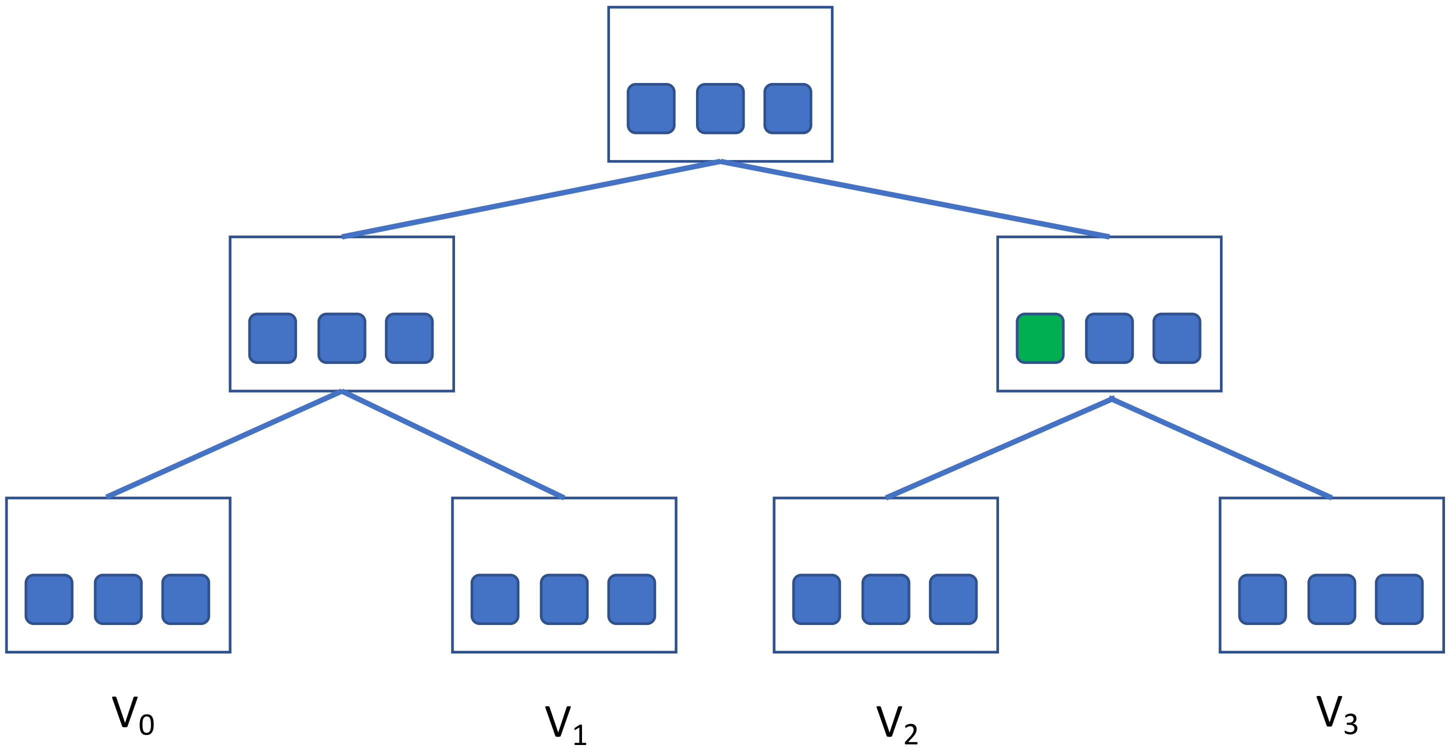
CLIENT

Position Map

LogicalBlockID	LeafLabel
0	v_2
1	v_0
2	v_3
…	…

STASH

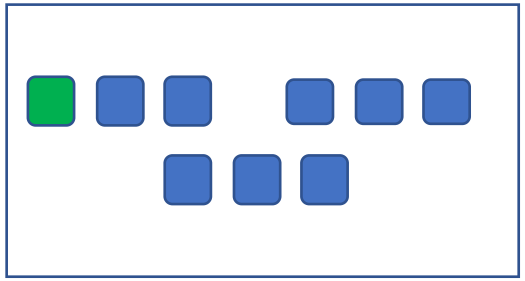
SERVER



Path ORAM Evictions

CLIENT

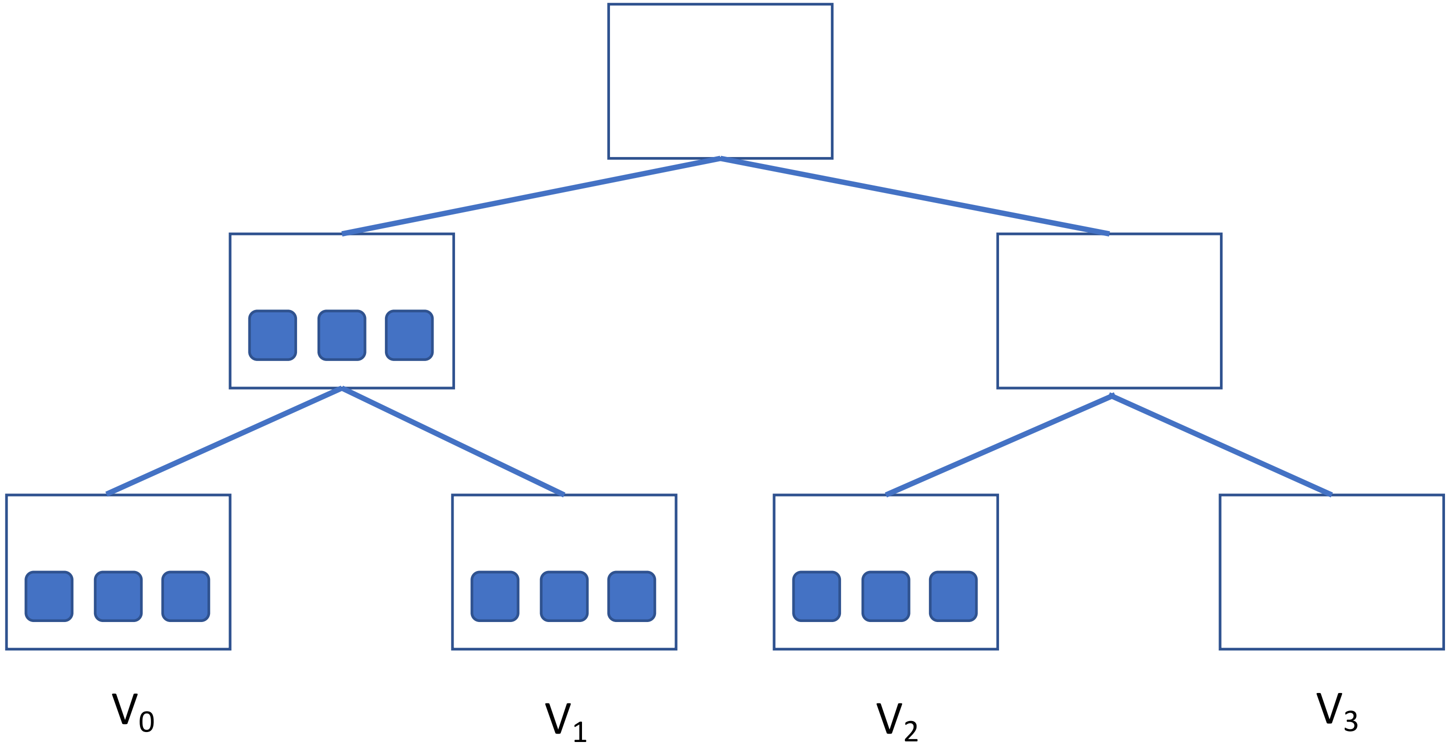
STASH



Position Map

LogicalBlockID	LeafLabel
0	v_2
1	v_0
2	v_3
…	…

SERVER



Can also evict along pre-determined paths

Path ORAM: Performance Metrics

Bandwidth: $O(\log n)$, worst-case

Round-trips: 1 RT per access

Computational complexity: trivial

Locality of Access:

- # of seeks: $O(\log n)$
- Access seq. chunk: $O(\boldsymbol{chunkSize} \, * \, logN)$

Why Locality of Access?

- HDD: 1 seek = 10,000x slower
- SSD: Random placement $\Longrightarrow$ Significant wear
- File systems
  - caching, prefetching require data locality
- Applications with range queries e.g., GIS

Locality-Privacy Tradeoff

Data locality for “free”?

No

What can we afford to leak?
Sequential access size?

[Asharov ‘17]

Why is this acceptable?

Range ORAM: Locality-Optimized Range Queries

Range ORAM [Asharov et al. '17]:

For range query of size *r*

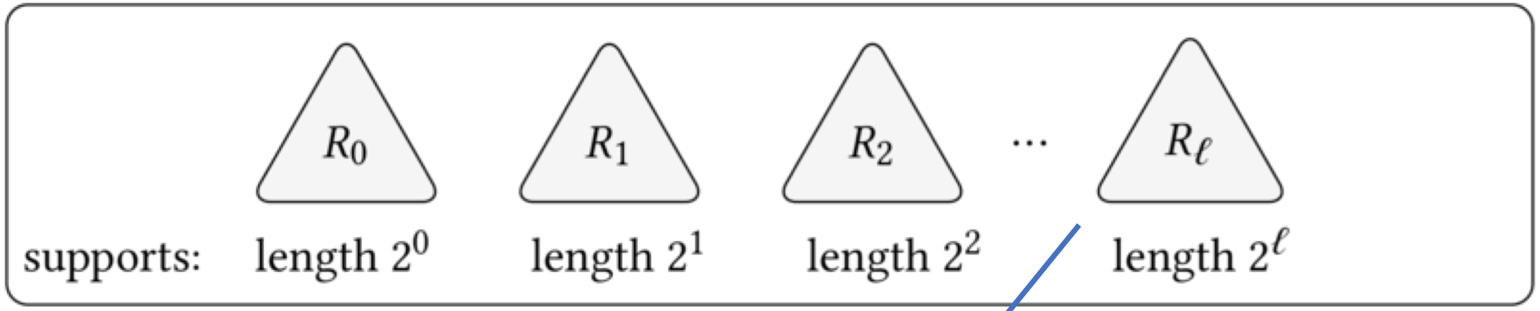
- ✓ $O\left(\log^3 N\right)$ seeks
- ✓ $O\left(\mathrm{r.}\ \log^3 N\right)$ bandwidth required

rORAM:

- ✓ $O\left(\log N\right)$x fewer seeks
- ✓ $O\left(\log N\right)$x lower bandwidth required

rORAM

- $l \in O(\log N)$ *independent* ORAMs
- Data is duplicated



Seek-optimized for querying ranges of size 2^l

- For R_l:
  - # of seeks for reading ($r = 2^l$) blocks in range: O(logN) ind. of r
  - # of seeks for evicting ($r=2^l$) blocks: O(logN) ind. of r

Insight 1: Locality-Optimized Layout

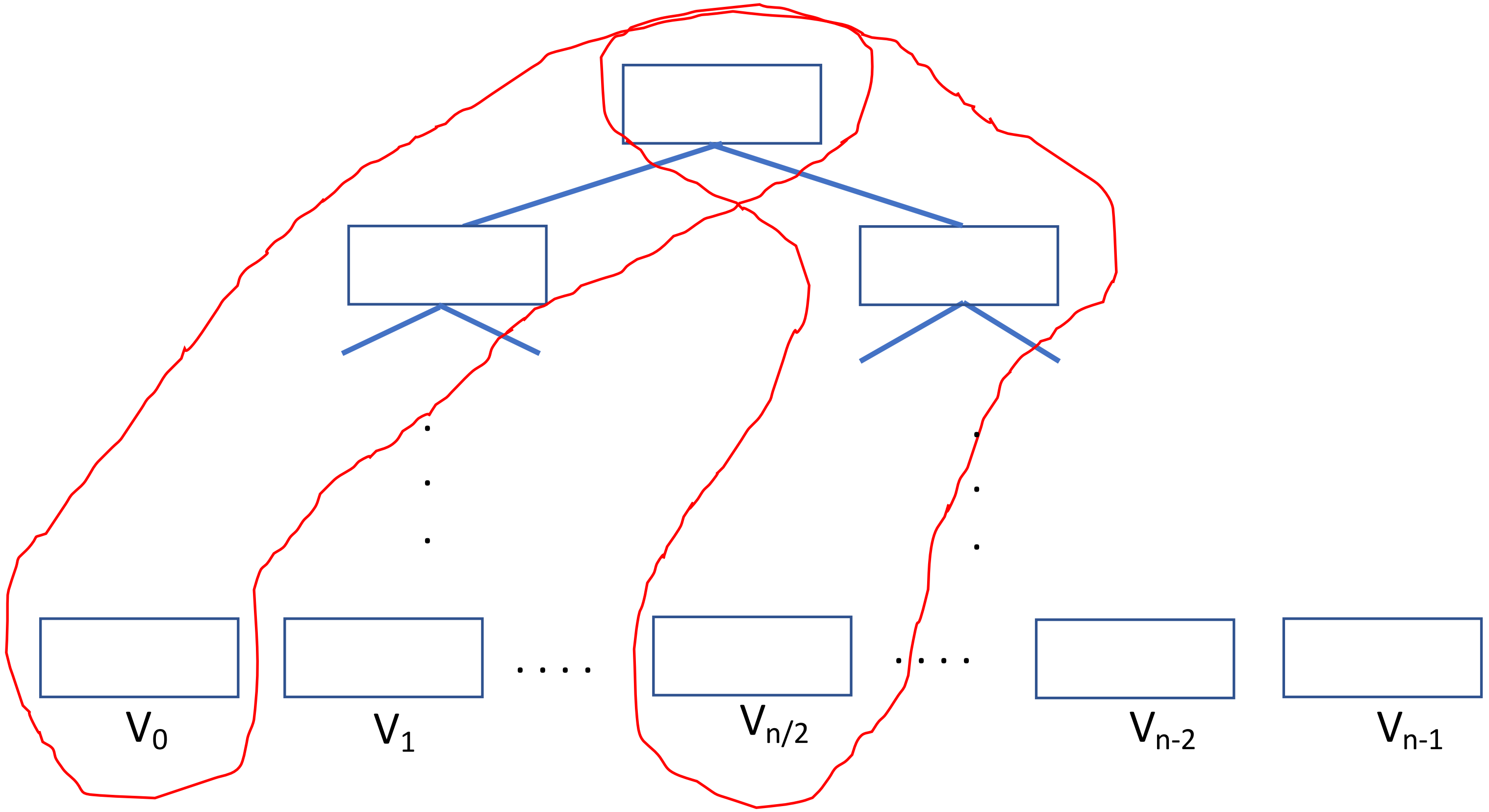
Problem: Evicting *r* blocks requires O(r*logN) seeks

Observation: Eviction Path Selection is Deterministic

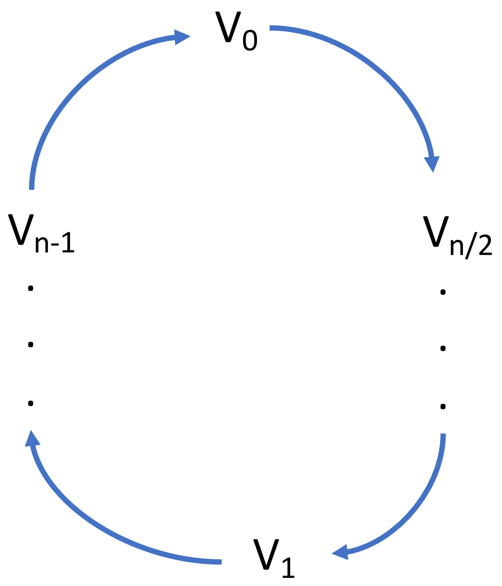
$\Longrightarrow$ Paths for consecutive evictions known apriori

$\Longrightarrow$ Order in which nodes are accessed per level known apriori

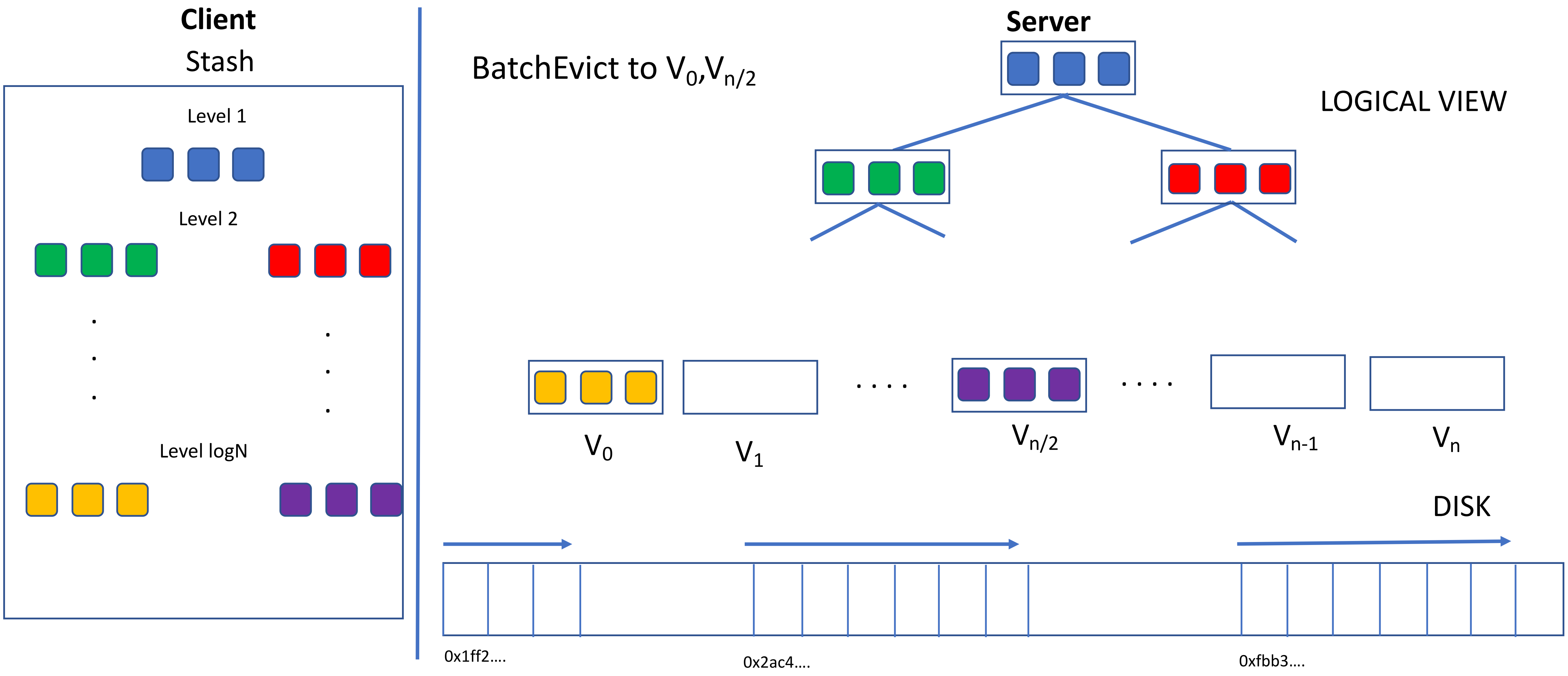
$\Longrightarrow$ Perform evictions level-wise



Eviction Path Selection Order



Batching Evictions Example



Batch *r* evictions: O(log N) seeks

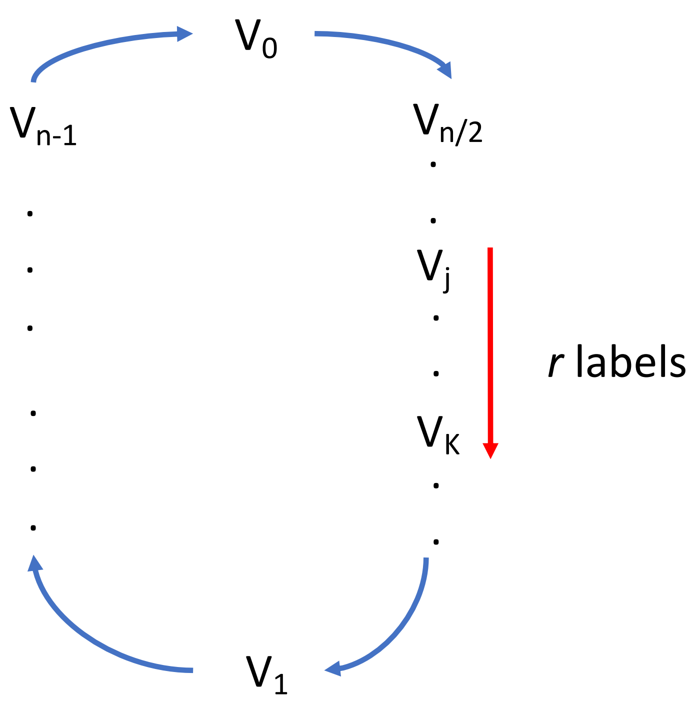
Insight 2: Locality-Optimized Re-Mapping

Problem: Reading *r* blocks in range requires O(r*logN) seeks

Idea: Any *r* consecutive eviction paths can be read with O(logN) seeks

Map Blocks in Range to Consecutive Eviction Paths

Eviction Path Selection Order



Remap: [a, a+1, …, a+r-1]

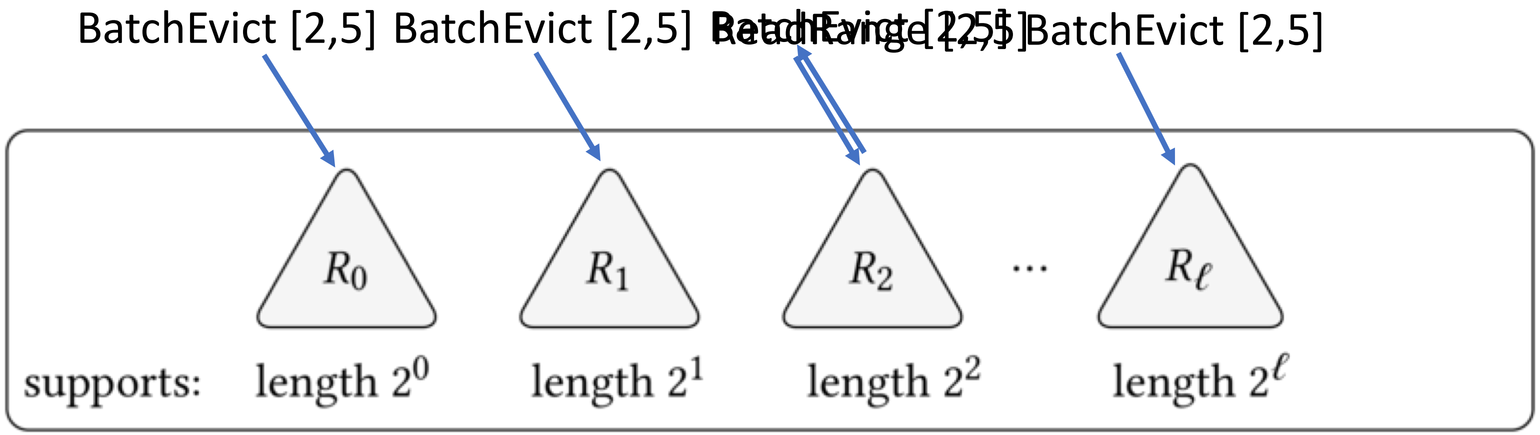
Position Map

LogicalBlockID	LeafLabel
a	v_j
a+1	…
…	…
a+r-1	v_k

ReadRange [a, b], b=a+r-1: O(logN) seeks

Access Protocol

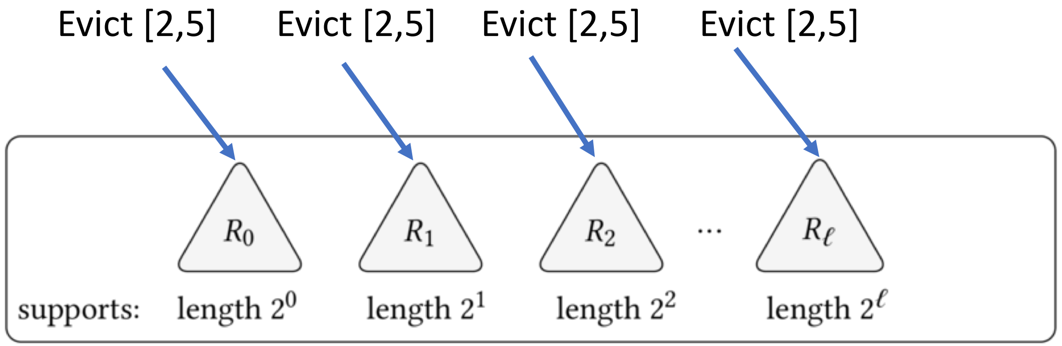
***Access*[2, 5]**



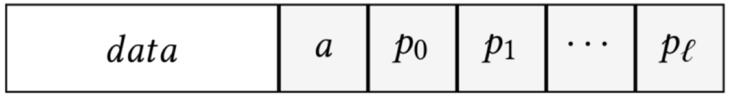
# of Seeks :

- $O(logN)$ disk seeks for ReadRange from R_2
- $O(logN)$ disk seeks for BatchEvict to Ri - $O(log^2N)$ seeks in total

Insight 3: Distributed Position Map



Insight: Reuse paths in ORAMs R_0, R_1, .. R_l



How do we know where block 2 is in R_0, R_1, …?

- $O(\log N)$ position map accesses

Pointer-based Oblivious Data Structure

- With each block, store pointers to its location in other ORAMs
- Locate position for "free" with reads

Asymptotic Performance

$\boldsymbol{Access[j, j+r-1]}$:

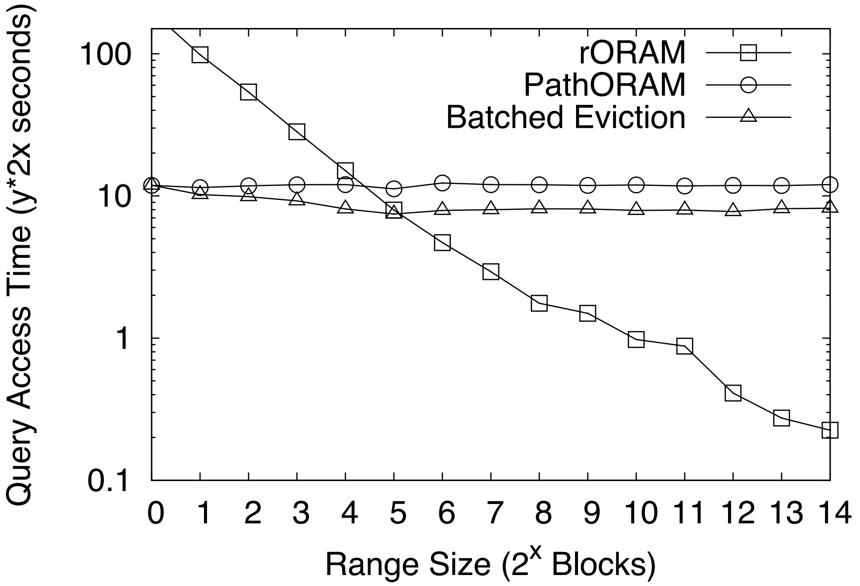
- ✓ $O(logN)$x fewer seeks
- ✓ $O(logN)$x lower bandwidth required

	Seeks	Bandwidth	Server Space	Leakage
PathORAM	$O(r.\log^2 N)$	$O(r.\log^2 N)$	$O(N)$	none
rORAM	$O(\log^2 N)$	$O(r.\log^2 N)$	$O(N\log N)$	Range size
Asharov et al.	$O(\log^3 N)$	$O(r.\log^3 N)$	$O(N\log N)$	Range size
Demertzis et al.	$O(r)$	$O(r.N^{1/3}.\log^2 N)$	$O(N)$	none

Query Access Time

Local HDD

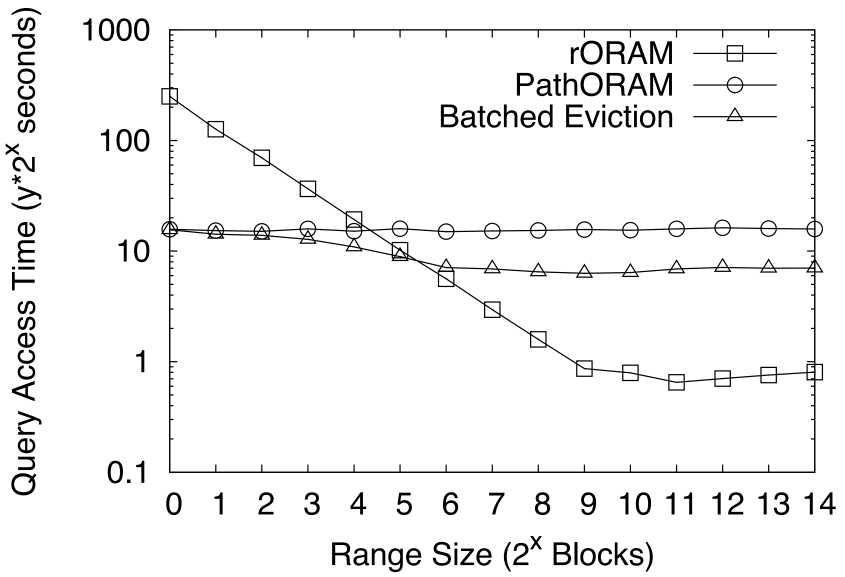
(logscale, higher is better)



30 – 50x speedup, range size >= 32 blocks

Network Block Device

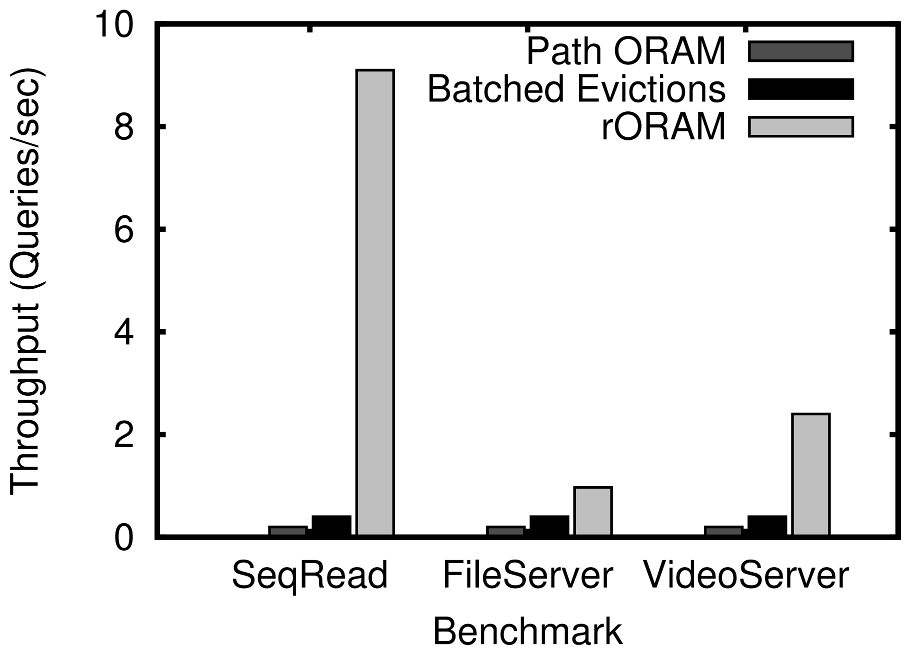
(logscale, higher is better)



10x speedup, range size >= 64 blocks

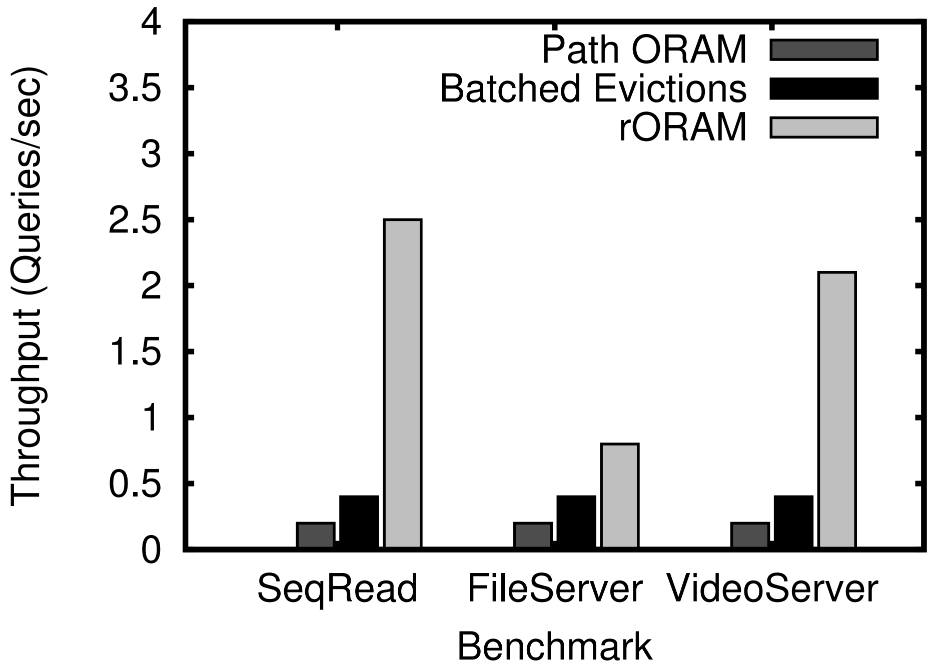
Throughput

Local HDD
(higher is better)



File Server = 5x, Video Server = 11x

Network Block Device
(higher is better)



File Server = 2x, Video Server = 4x

Summary

Practical Range ORAM

- ✓ $O\ (\log N)$x fewer seeks
- ✓ $O\ (\log N)$x lower bandwidth required

Optimized for Real World Applications

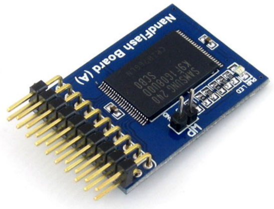
Can we do better?

app-specific optimizations

What I am working on

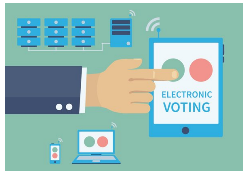
I am on the job market!

Oblivious RAM [NDSS ’19, ‘19]



Plausible Deniability [PETS ‘17, ‘19]

Integrity-Preserving Block Storage [ApSys ‘17]



History Independence [TIFS ’15]

Secure CPU Architecture & Secure Virtualization

Query Authentication [TKDE]

Thank you!!