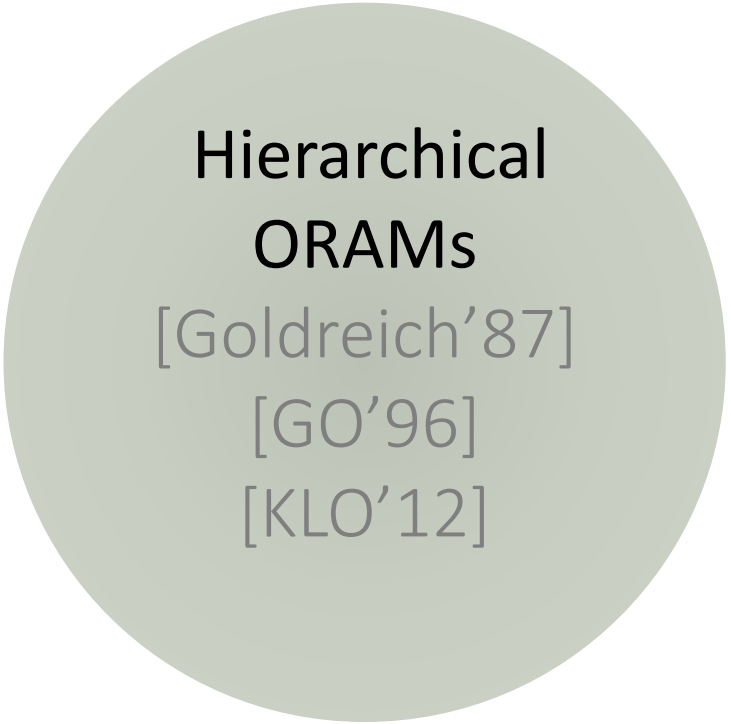
Oblivious Data Structures

Xiao Shaun Wang, Kartik Nayak, Chang Liu, T-H. Hubert Chan, Elaine Shi, Emil Stefanov, Yan Huang

Oblivious RAM is a cryptographic primitive for provably obfuscating access patterns to data.



$$\text{Bandwidth Overhead:} = \frac{\text{Data transferred in oblivious case}}{\text{Data transferred in non-oblivious case}}$$

Hierarchical ORAMs

[Goldreich'87]

[GO'96]

[KLO'12]

$$\text{Bandwidth Overhead:} = \frac{\text{Data transferred in oblivious case}}{\text{Data transferred in non-oblivious case}}$$

Hierarchical ORAMs

[Goldreich'87]

[GO'96]

[KLO'12]

Tree-based ORAM

[SCSL'11]

[SDSCFRYD'13]

Best known ORAM achieves
$O(\log^2 N/\log \log N)$ overhead [KLO'12]

Best known ORAM achieves
$O(\log^2 N/\log \log N)$ overhead [KLO'12]

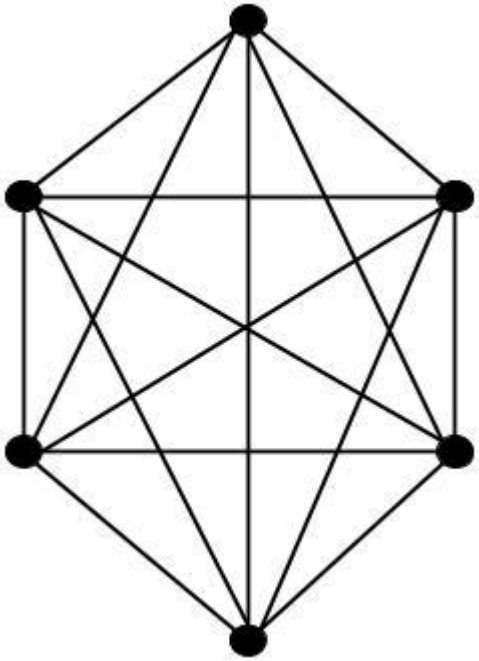
Path ORAM partially solves this problem

Best known ORAM achieves
$O(\log^2 N/\log \log N)$ overhead [KLO'12]

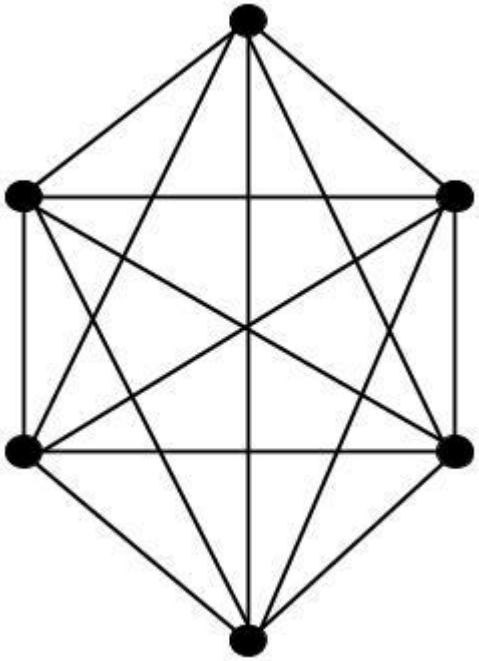
Best known ORAM achieves
$O(\log^2 N/\log\log N)$ overhead [KLO'12]

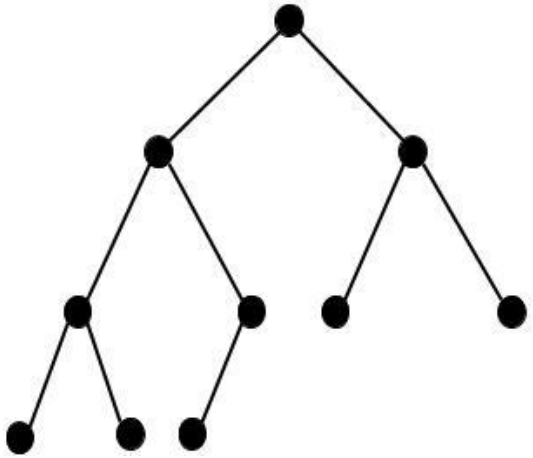
Can we do better for **restricted access patterns?**

RAM

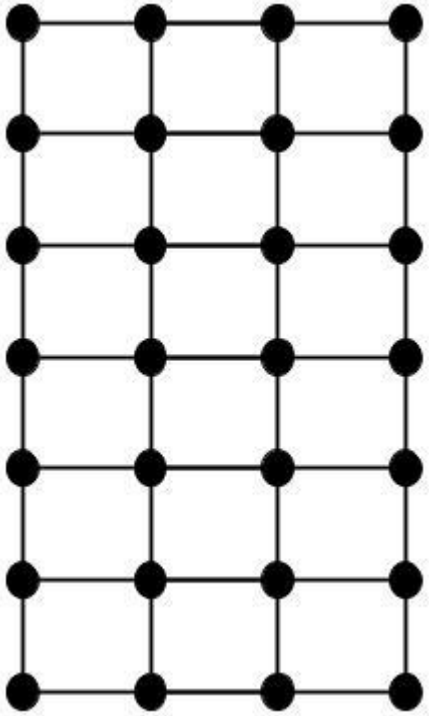


RAM



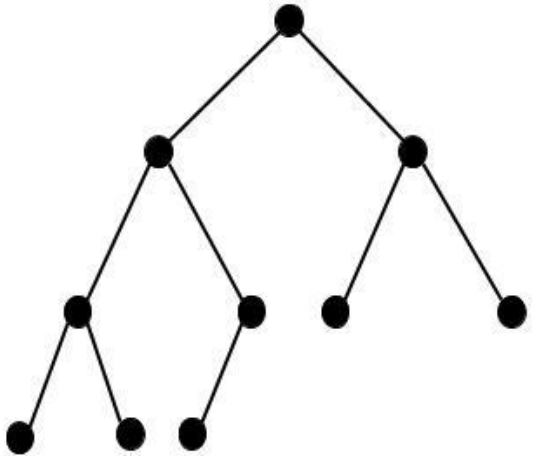


Bounded-degree trees

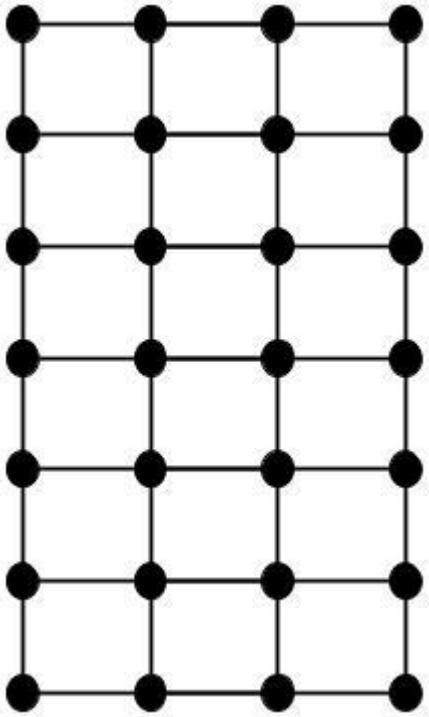


Access patterns with locality

Can we do better for these restricted access patterns?



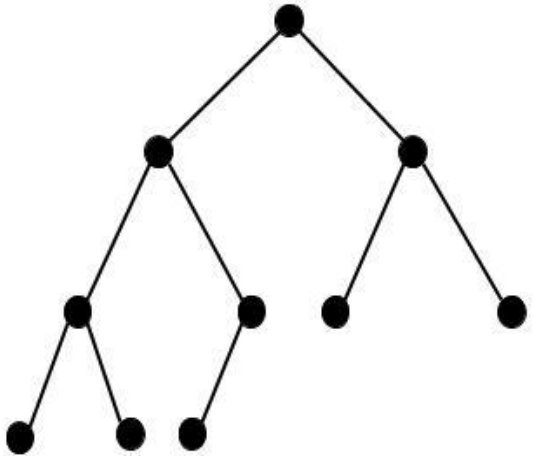
Bounded-degree trees



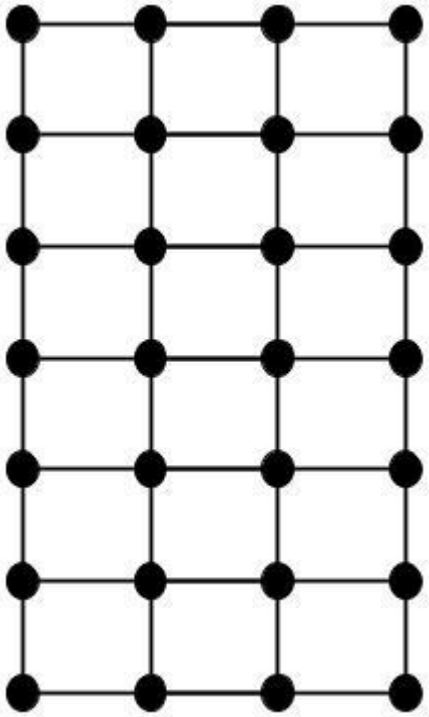
Access patterns with locality

Can we do better for these restricted access patterns?

YES



Bounded-degree trees

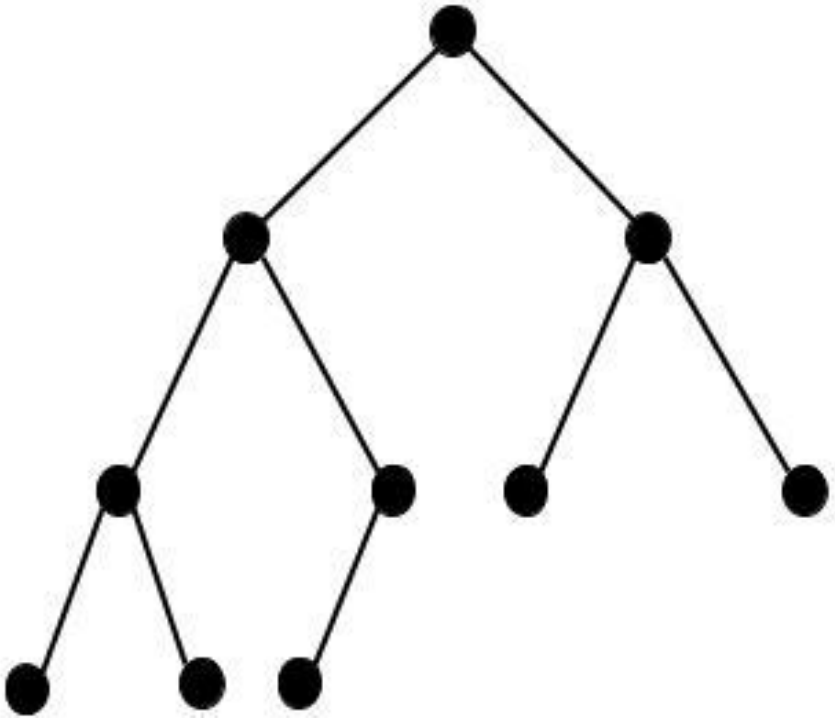


Access patterns with locality

Bounded-degree tree

Stack / Queue

Bounded-degree tree



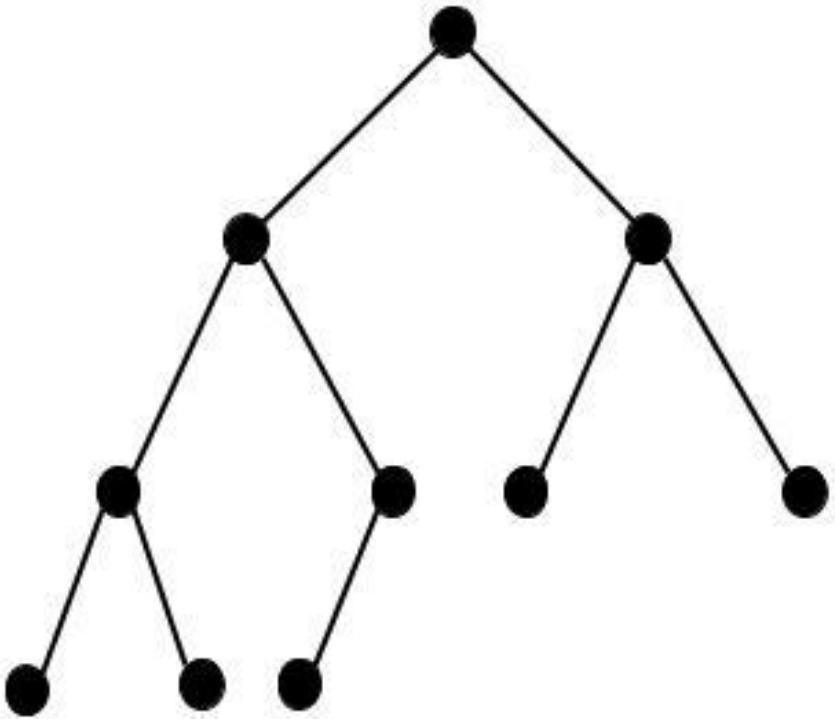
Stack / Queue

Map (AVL tree, B tree)

Heap

Bounded-degree tree

- ✓The effective overhead is O(log N)



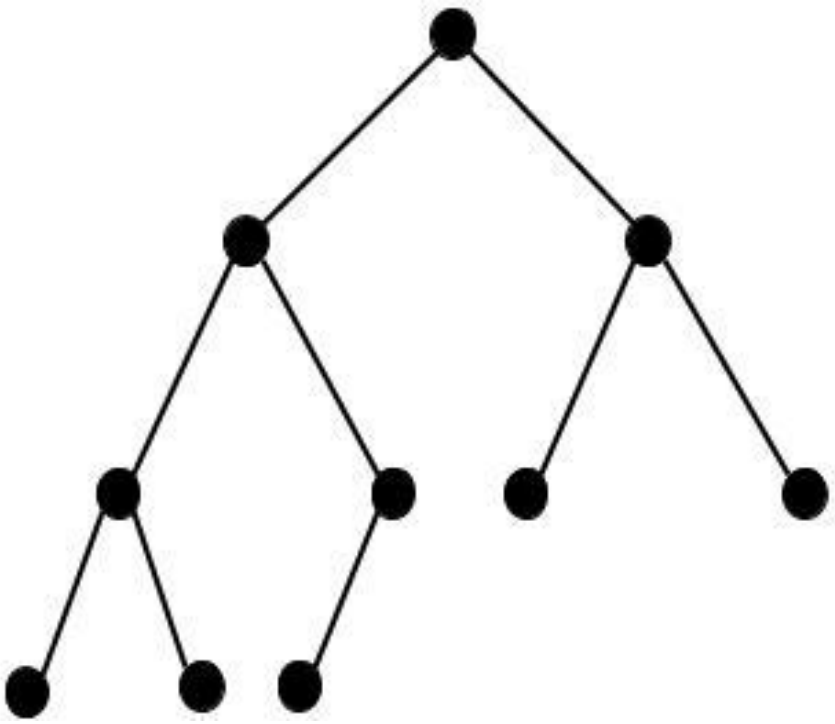
Stack / Queue

Map (AVL tree, B tree)

Heap

Bounded-degree tree

- ✓The effective overhead is O(log N)



ORAM: $O(\log^2 N/\log \log N)$

Stack / Queue

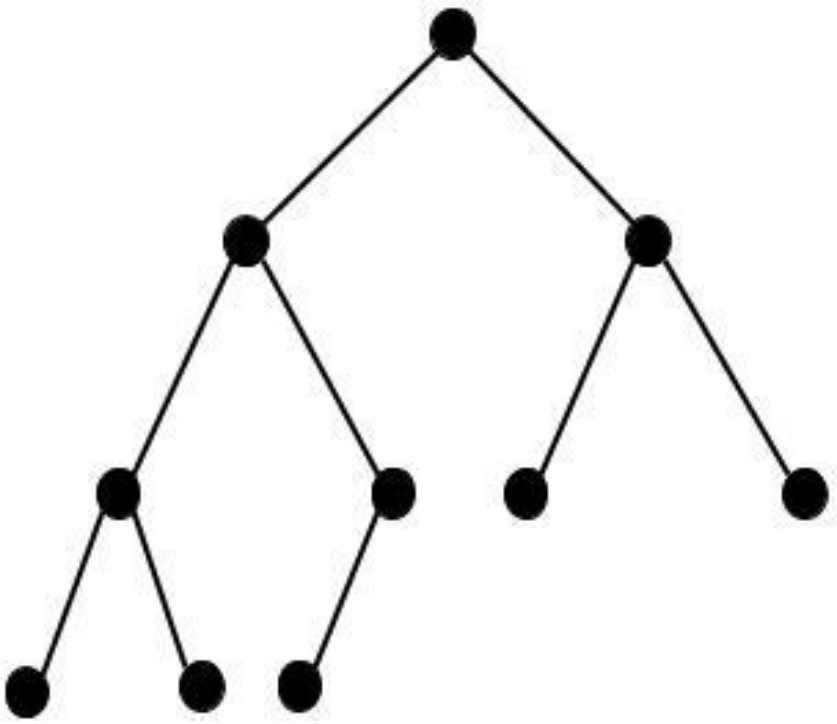
Map (AVL tree, B tree)

Heap

Bounded-degree tree

- ✓The effective overhead is O(log N)
- ✓Inspired by [GGHJRW'13]
- ✓Speedup
  - ✓Bandwidth overhead 12x – 16x
  - ✓Circuit size 10x – 14x

    Compared with Path ORAM; data size 2^{30}

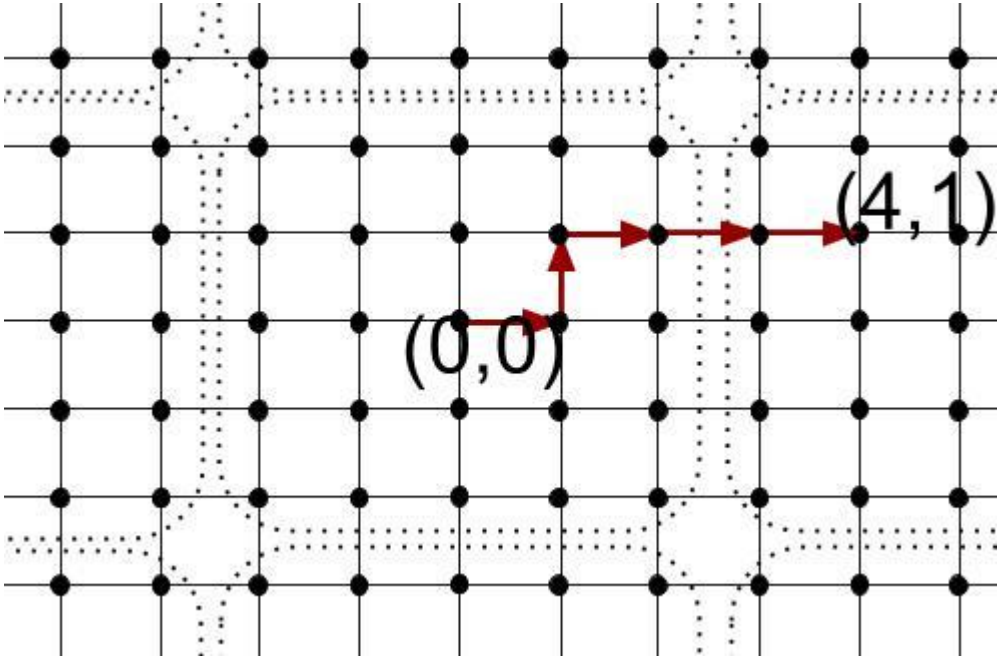


Stack / Queue

Map (AVL tree, B tree)

Heap

Access patterns with locality



Access patterns with locality

✓Overhead:

Deque, doubly linked list

O(log N)

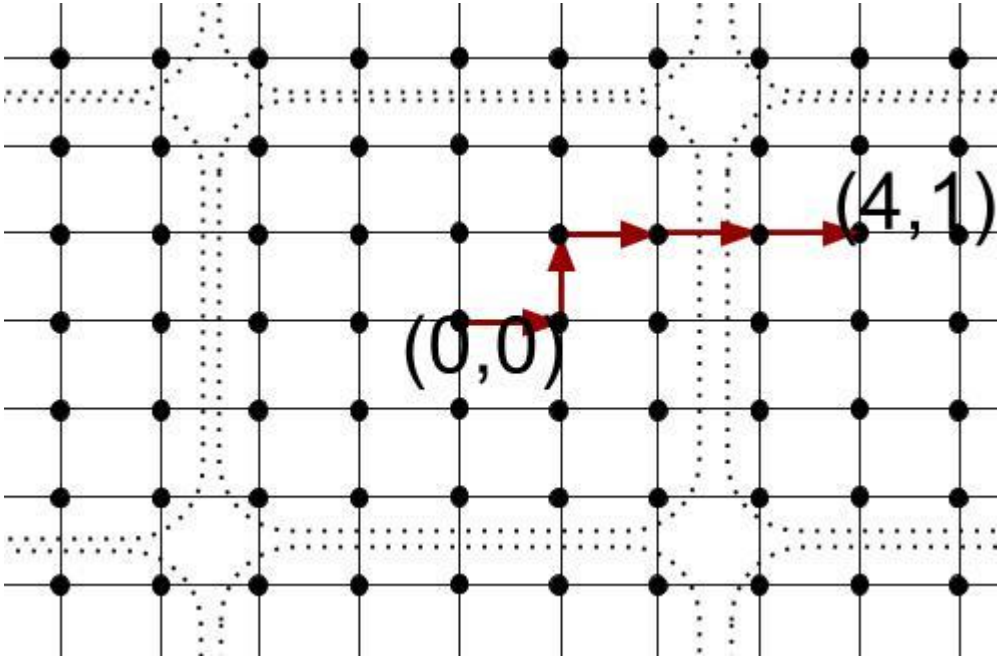
2-dimensional grid

$O(\log^{1.5} N)$

General graphs

$O(12^d \log^{2-1/d} N)$

d: doubling dimension of the graph



Access patterns with locality

✓Overhead:

Deque, doubly linked list

$O(\log N)$

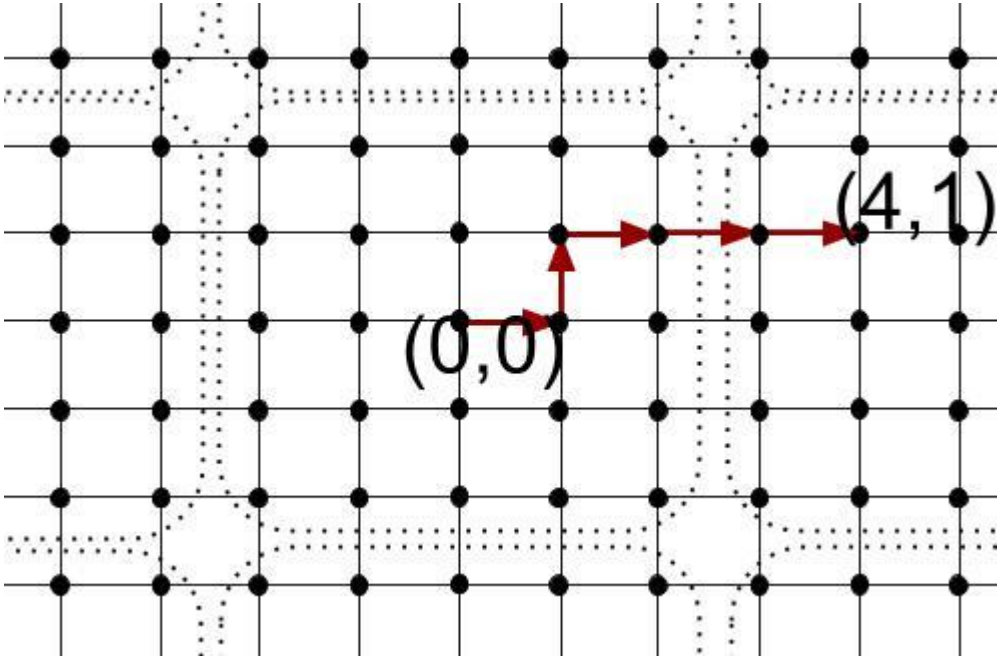
2-dimensional grid

$O(\log^{1.5} N)$

General graphs

$O(12^d \log^{2-1/d} N)$

d: doubling dimension of the graph



✓Bandwidth overhead speedup for deque, doubly linked list - 9x

Compared with Path ORAM; data size 2^{30}

Stack / Queue

Map (AVL tree)

Heap

Deque

Doubly linked list

Open source implementation on a garbled circuit backend coming soon

Oblivious Data Structures:
[WNLCSSH'14]

Thank You!

kartik@cs.umd.edu