Oblivious RAM: From Theory to Large-Scale Deployment

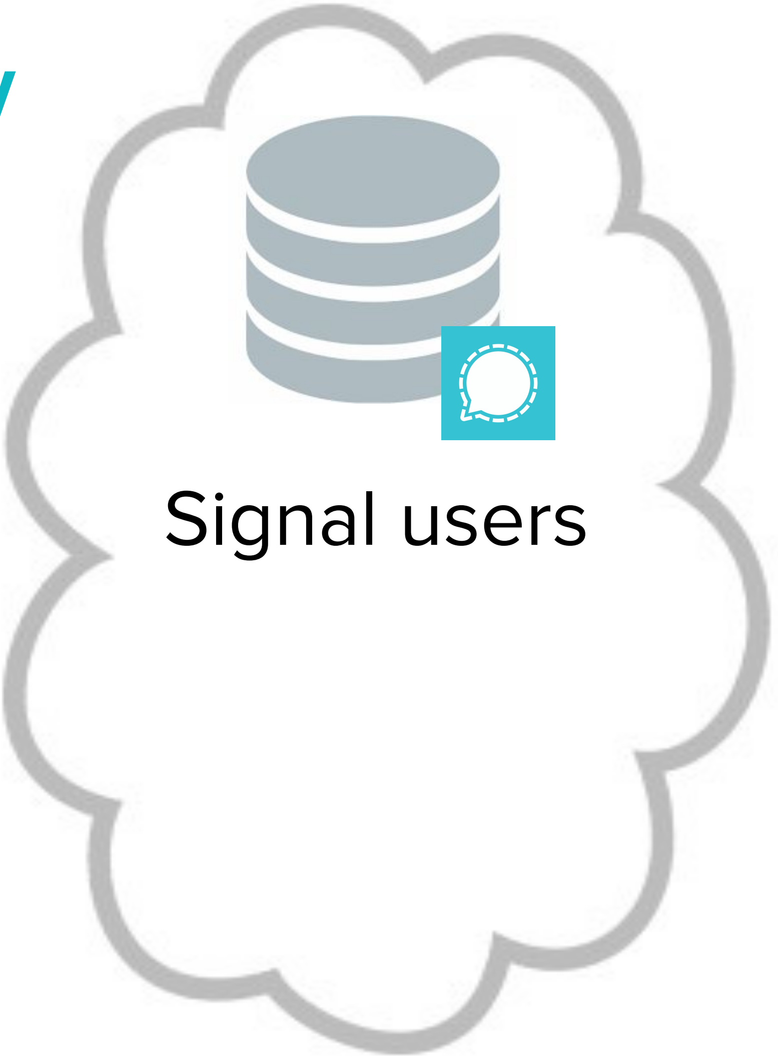
Elaine Shi
CMU

Applications and challenges

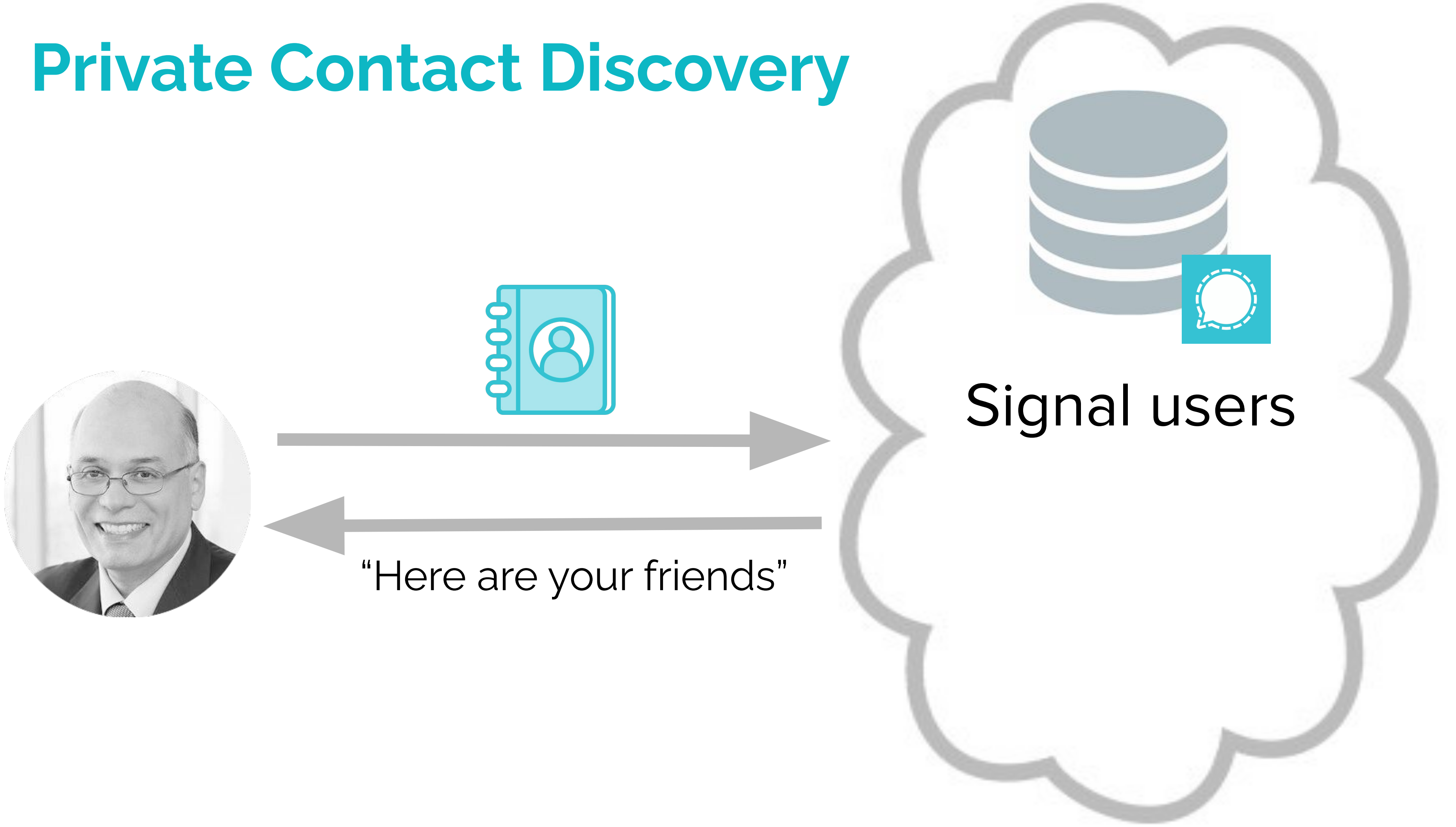
Oblivious RAM

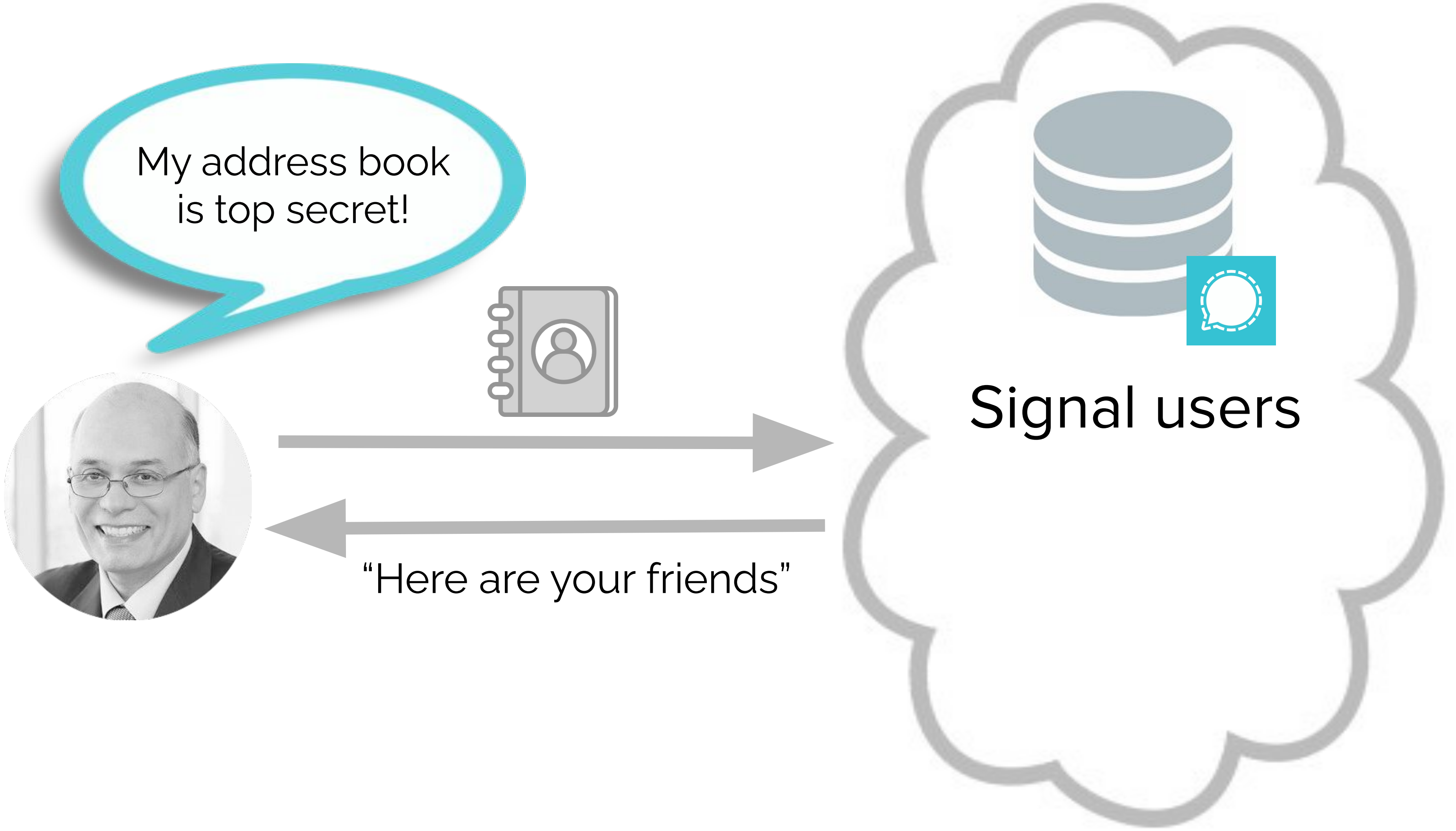
Signal's story

Private Contact Discovery

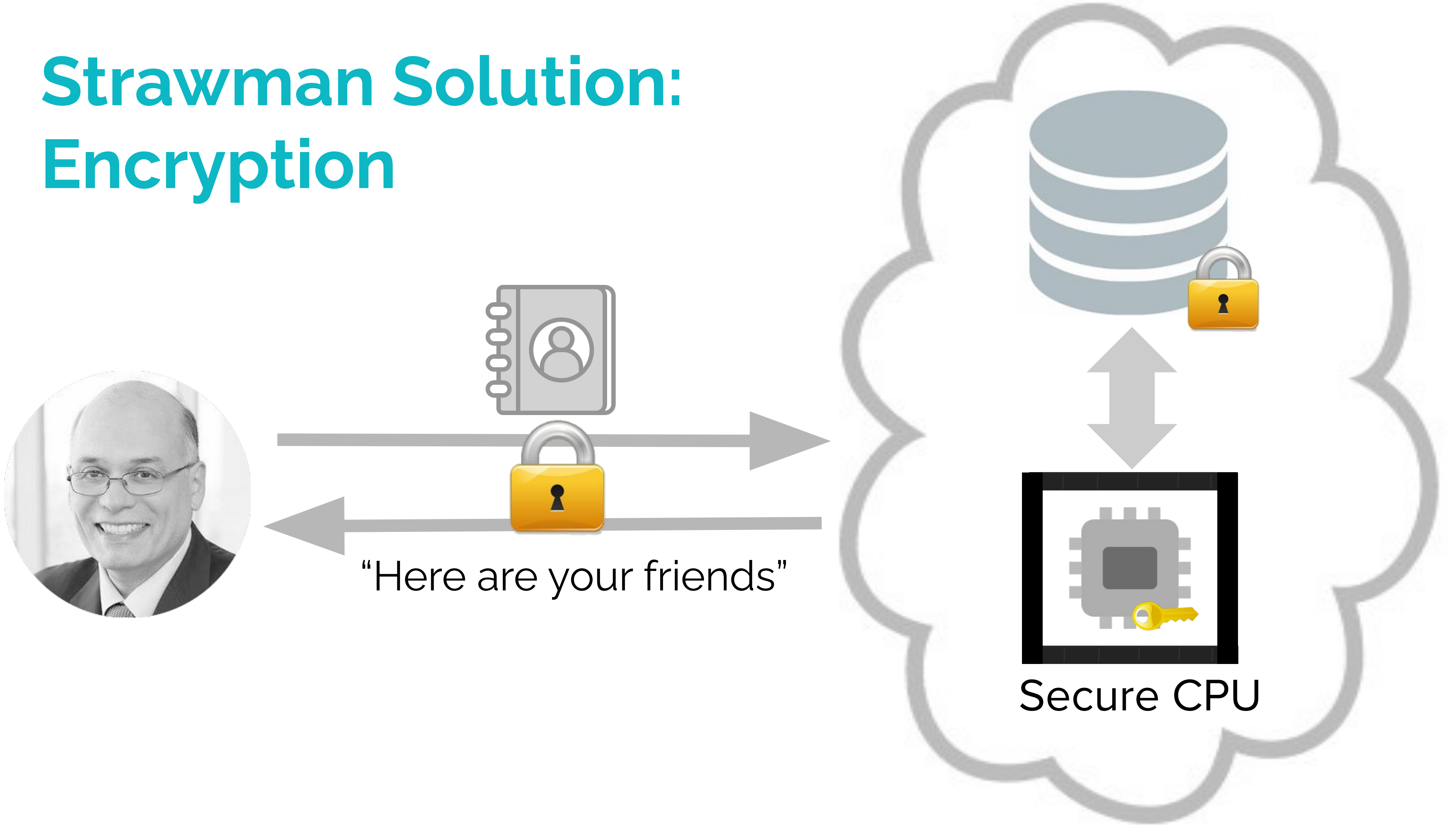


Private Contact Discovery

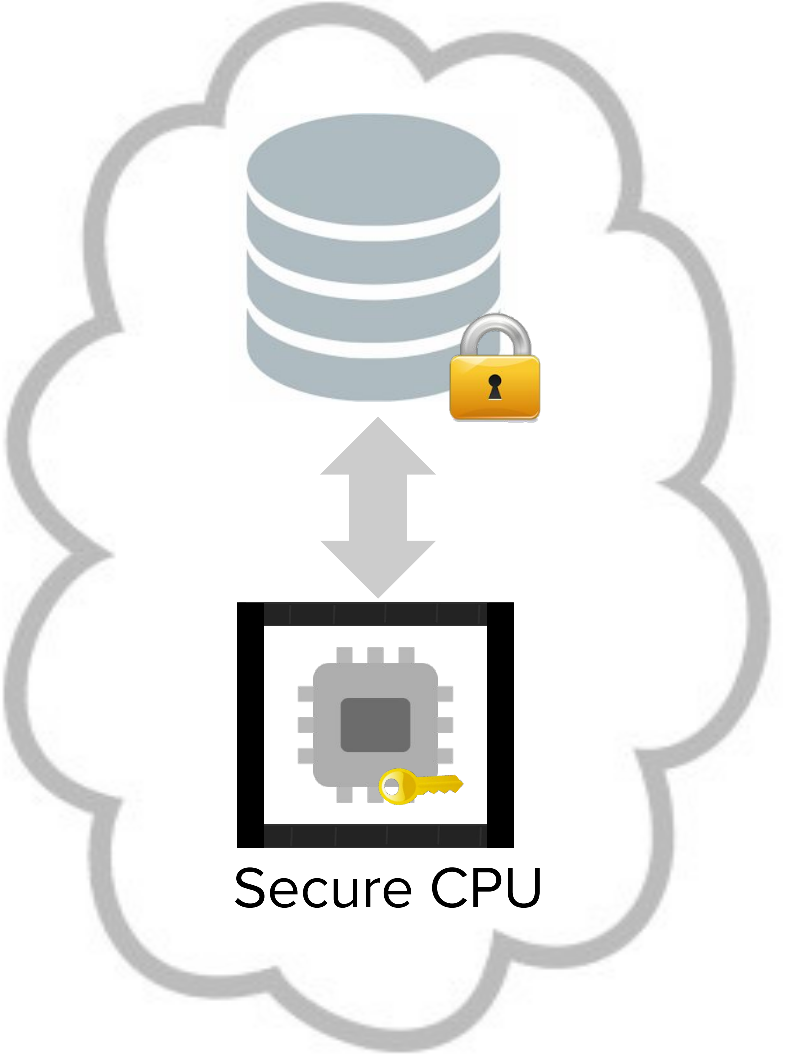



My address book is top secret!
Signal users
"Here are your friends"

Strawman Solution: Encryption



Access patterns to even encrypted data leak sensitive information.



Access pattern leakage, more generally

Access patterns of binary search leaks the rank of the number being searched.

```
func search(val, s, t)
    mid = (s + t)/2
    if val < mem[mid]
            search (val, 0, mid)
    else search (val, mid+ 1, t)
```

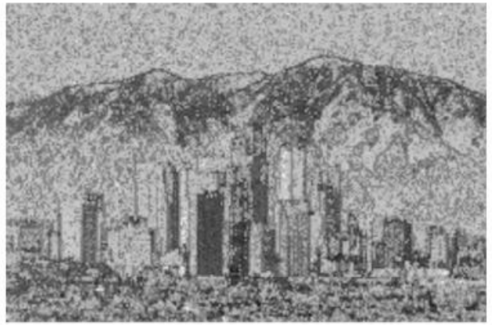
Access pattern leakage through a PL lens

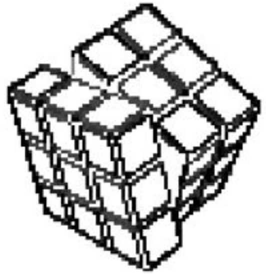
```
if (secret variable)
    read mem[x]
else
    read mem[y]
```

Recovering JPEG images through coarse-grained access patterns

Original

Recovered





[XCP'15 - IEEE S&P]

Can we **provably** defeat access pattern leakage

and **preserve efficiency**

Signal 2017:
batched linear scan

Signal 2017:
batched linear scan

$O(n/\beta)$ overhead
500 servers

n: total # memory blocks
β: batch size

Signal 2017: batched linear scan

$O(n/\beta)$ overhead
500 servers

Signal 2022: Path ORAM

[SDS+13]

$O(\log^2 n)$ overhead
6 servers

Applications and challenges

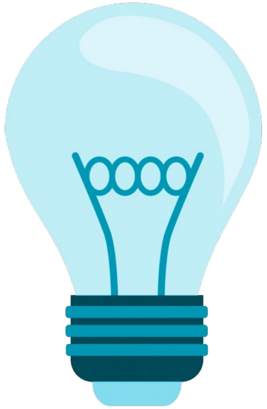
Oblivious RAM

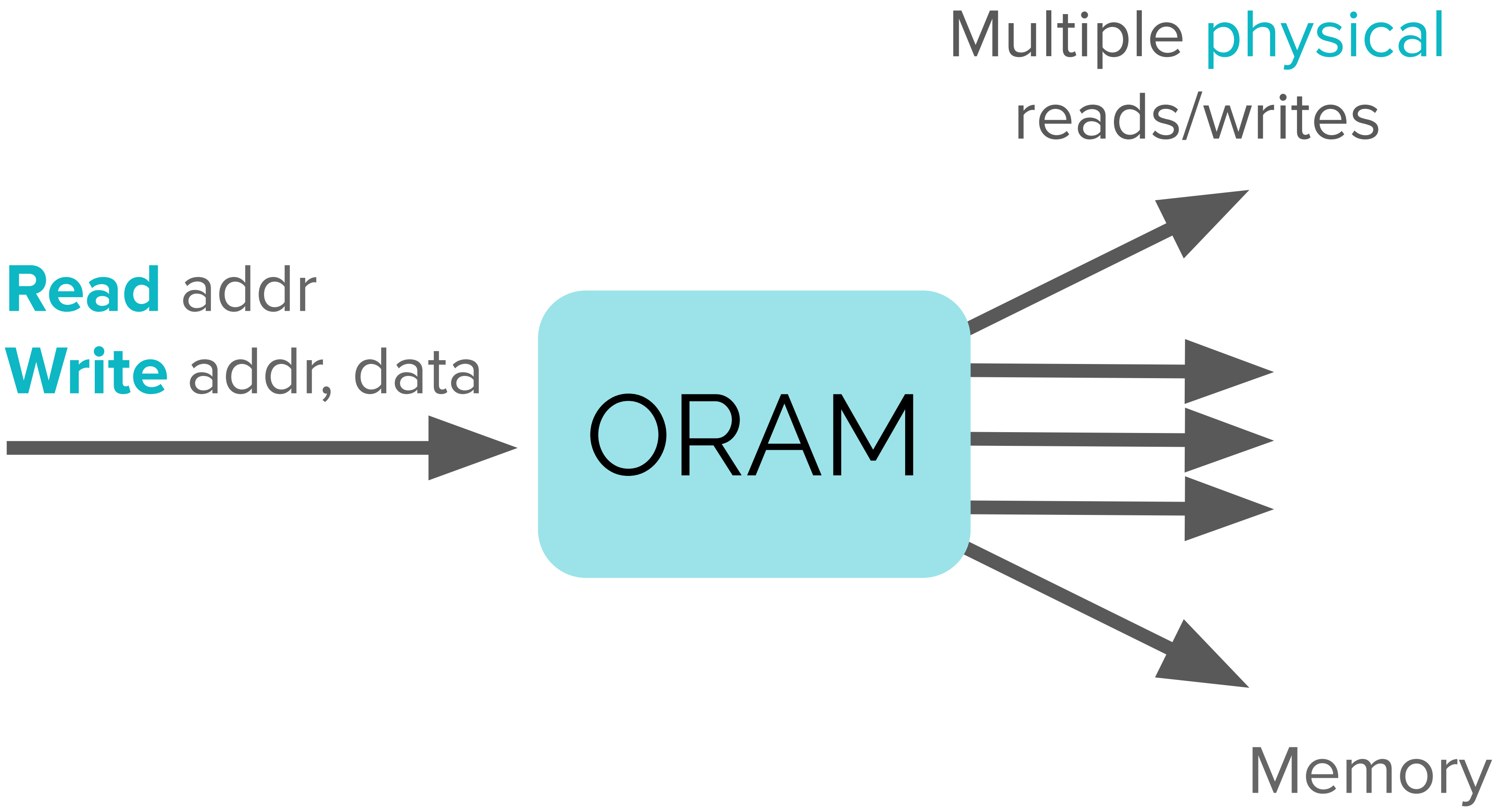
Signal’s story

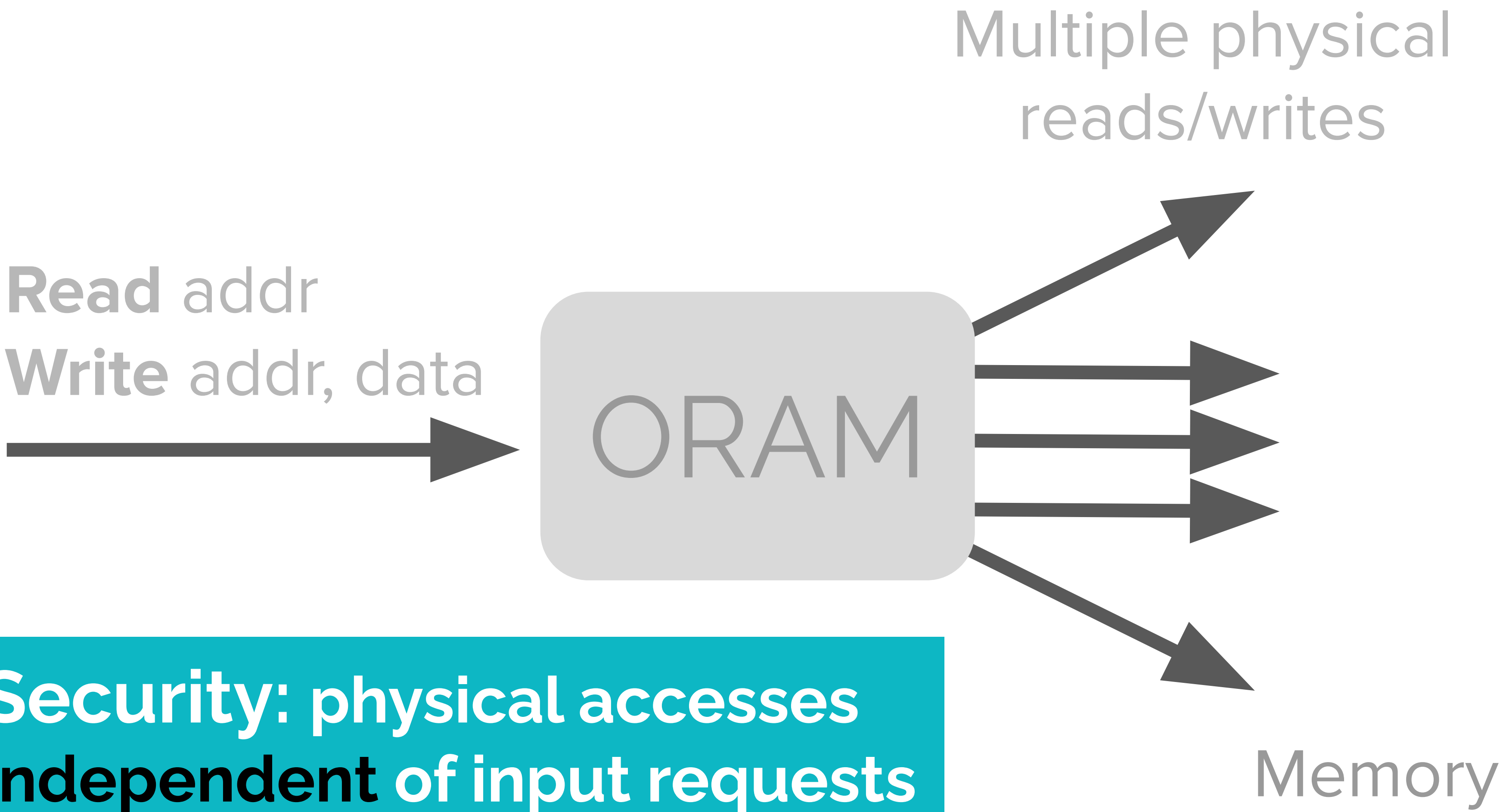
Oblivious RAM (ORAM)

is an algorithmic technique that provably "encrypts" access patterns

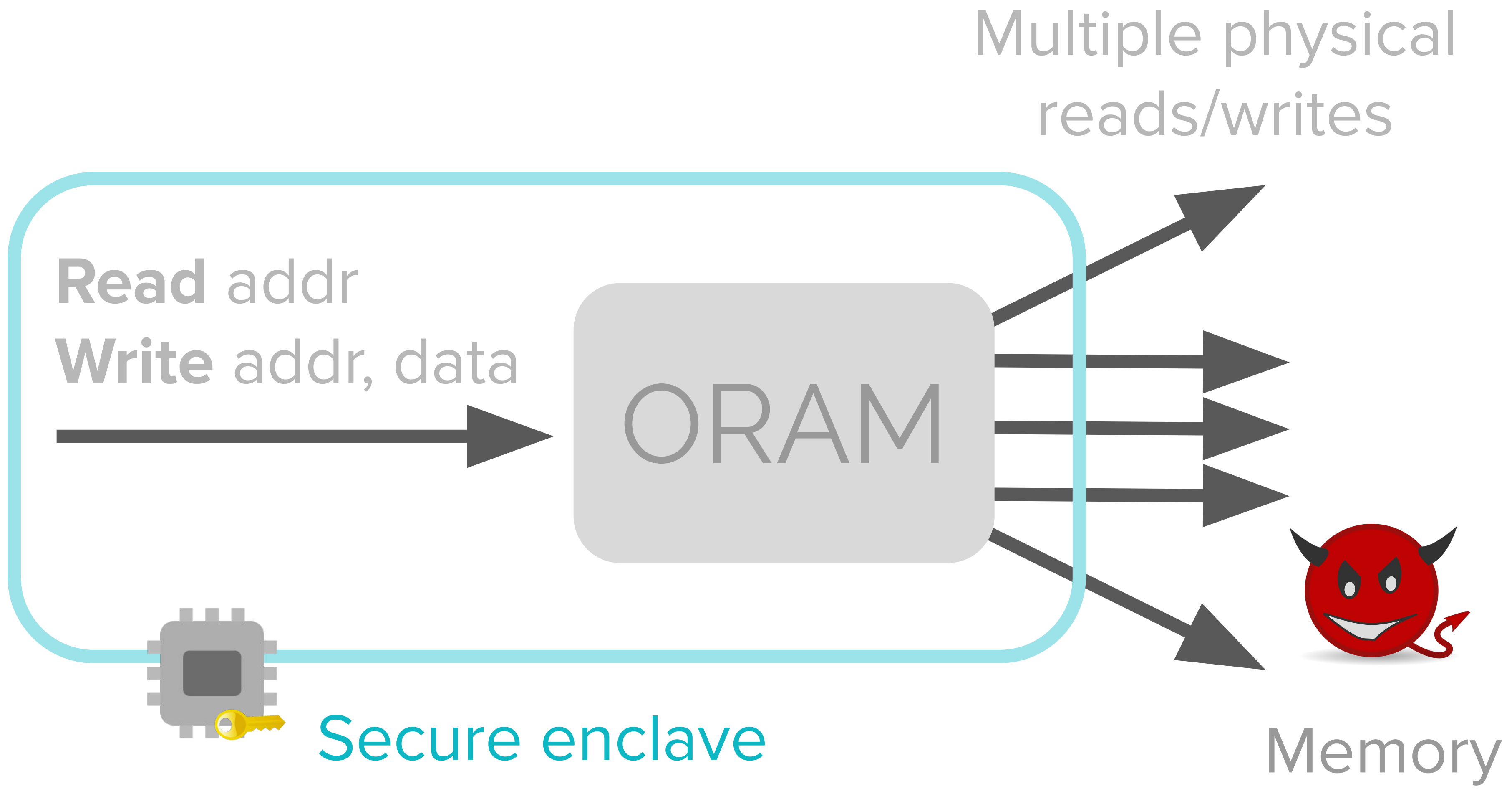
- Permutation
- Shuffling




Multiple physical reads/writes
Read addr
Write addr, data
ORAM
Memory



Security: physical accesses **independent** of input requests


Multiple physical reads/writes
Read addr
Write addr, data
ORAM
Secure enclave
Memory

There exist asymptotically “efficient” ORAMs

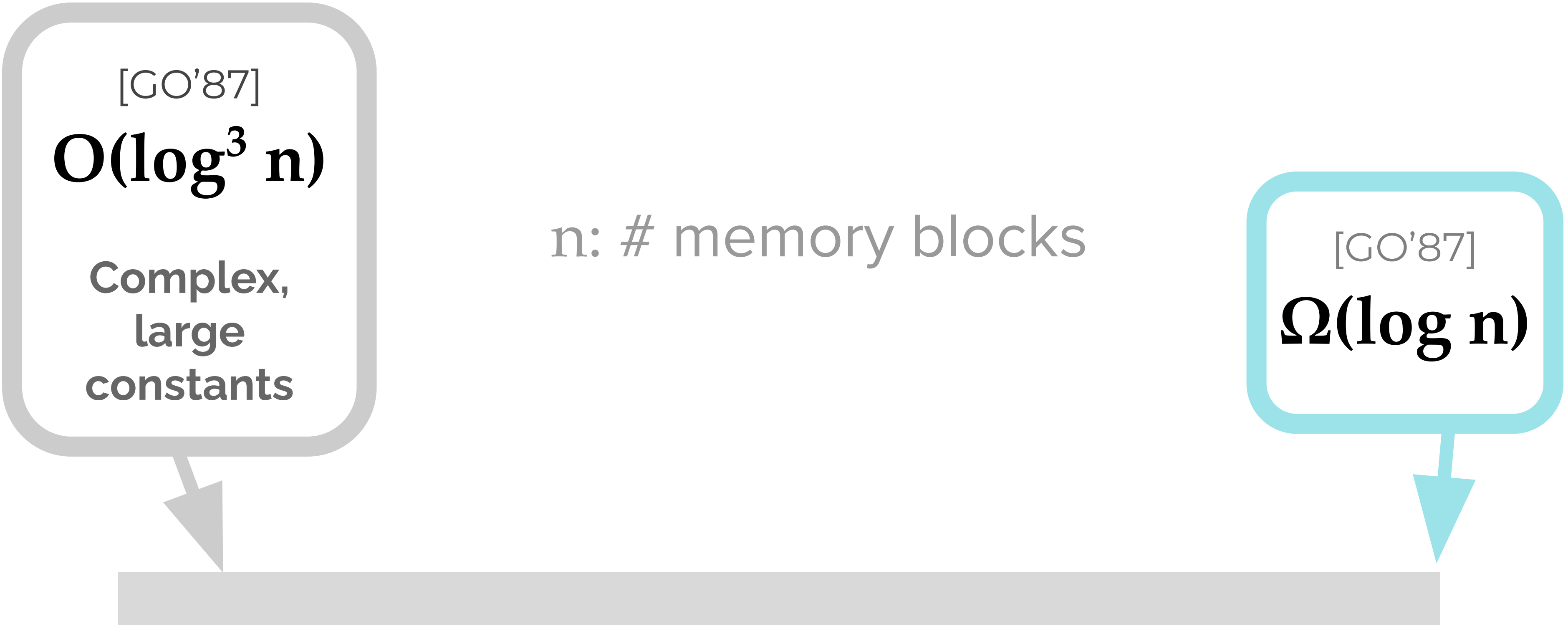
[GO’87]

$O(\log^3 n)$

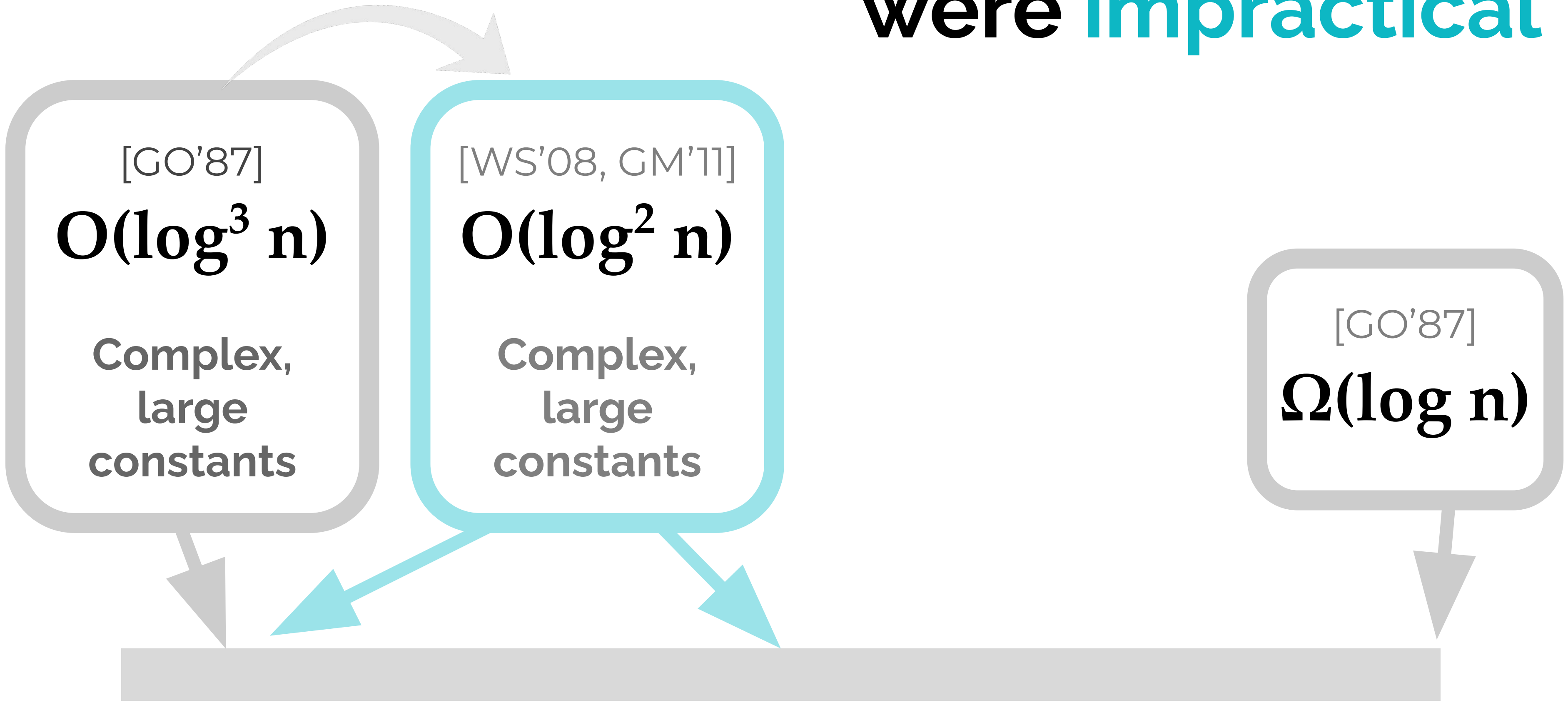
Complex, large constants

n: # memory blocks

ORAM must incur $\Omega(\log n)$ overhead



Back in 2011: known ORAM schemes were impractical



Dream questions for ORAM

[GO'87]

$O(\log^3 n)$

Complex, large constants

- Can ORAM ever be **practical**?
- Can we **bridge** the theoretical **gap**?

[GO'87]

$\Omega(\log n)$

YES and YES!

[GO'87]

$O(\log^3 n)$

Complex, large constants

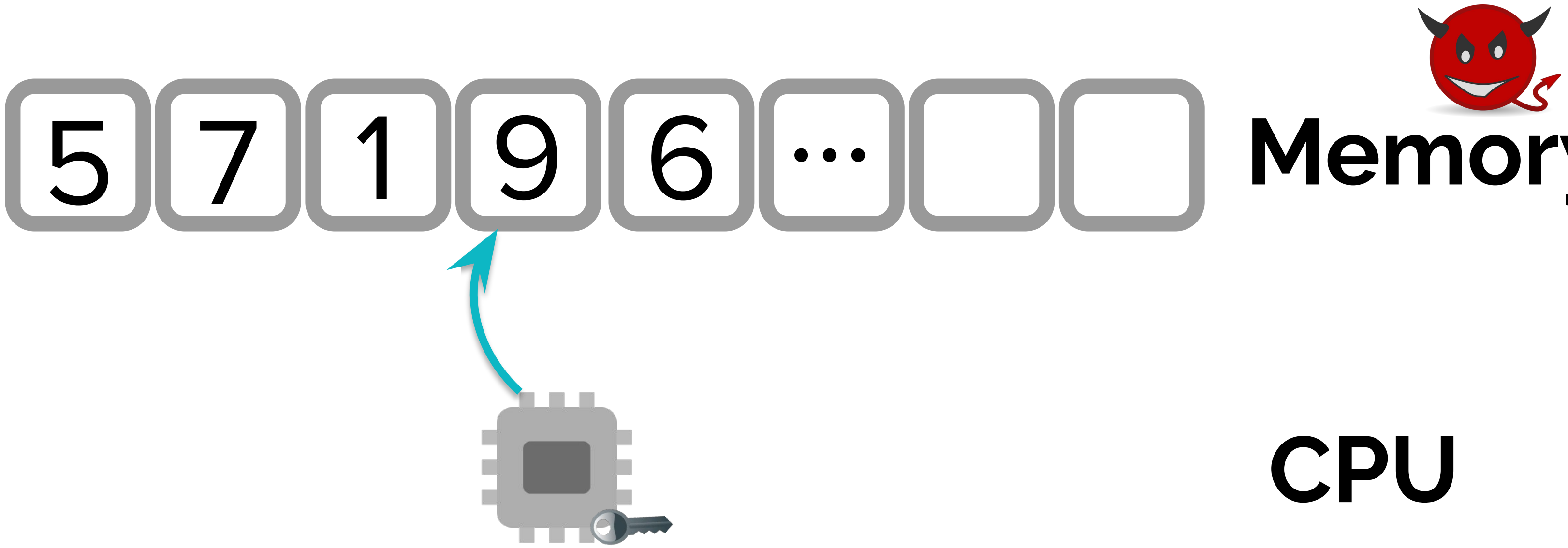
- Can ORAM ever be **practical**?
- Can we **bridge** the theoretical **gap**?

[GO'87]

$\Omega(\log n)$

Constructing an ORAM

Strawman: permute blocks in memory

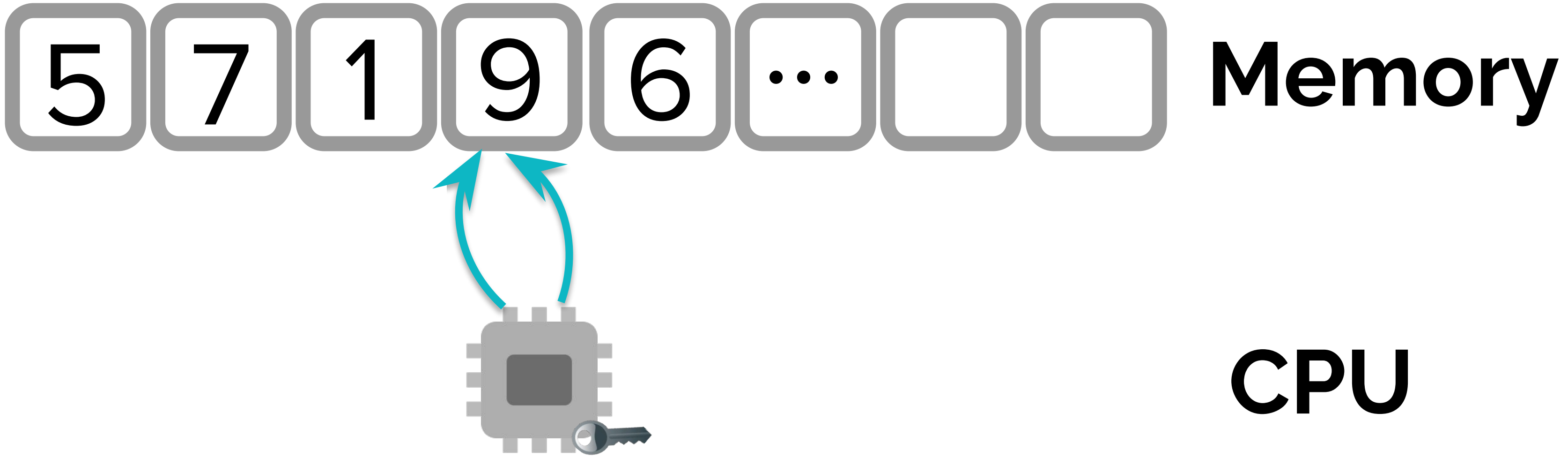


Memory

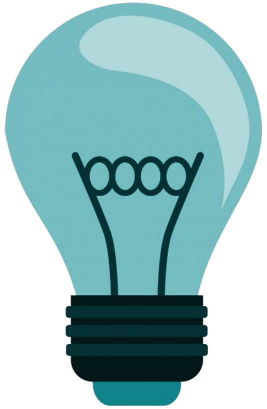
CPU

Strawman: provides one-time security!

e.g., leaks frequency, co-occurrence

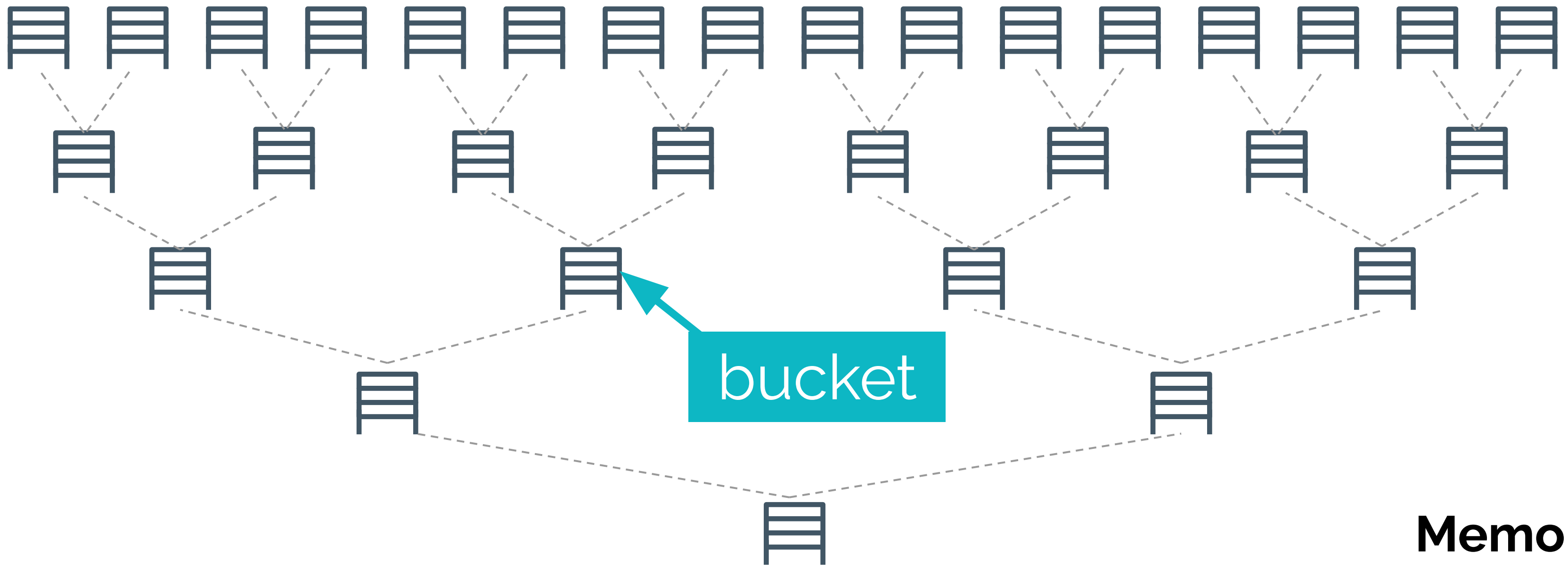


Blocks must move around in memory



A tree-based paradigm for ORAMs

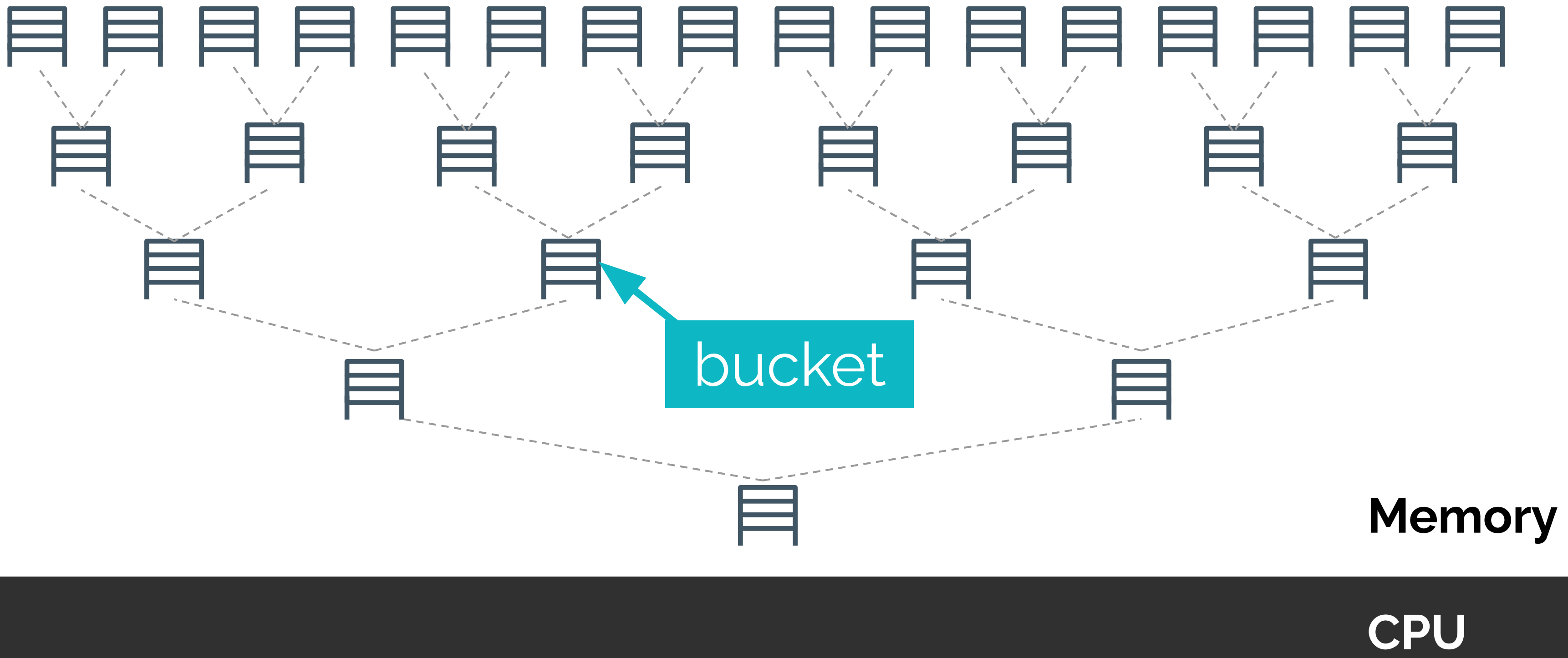
[SCSL'11]



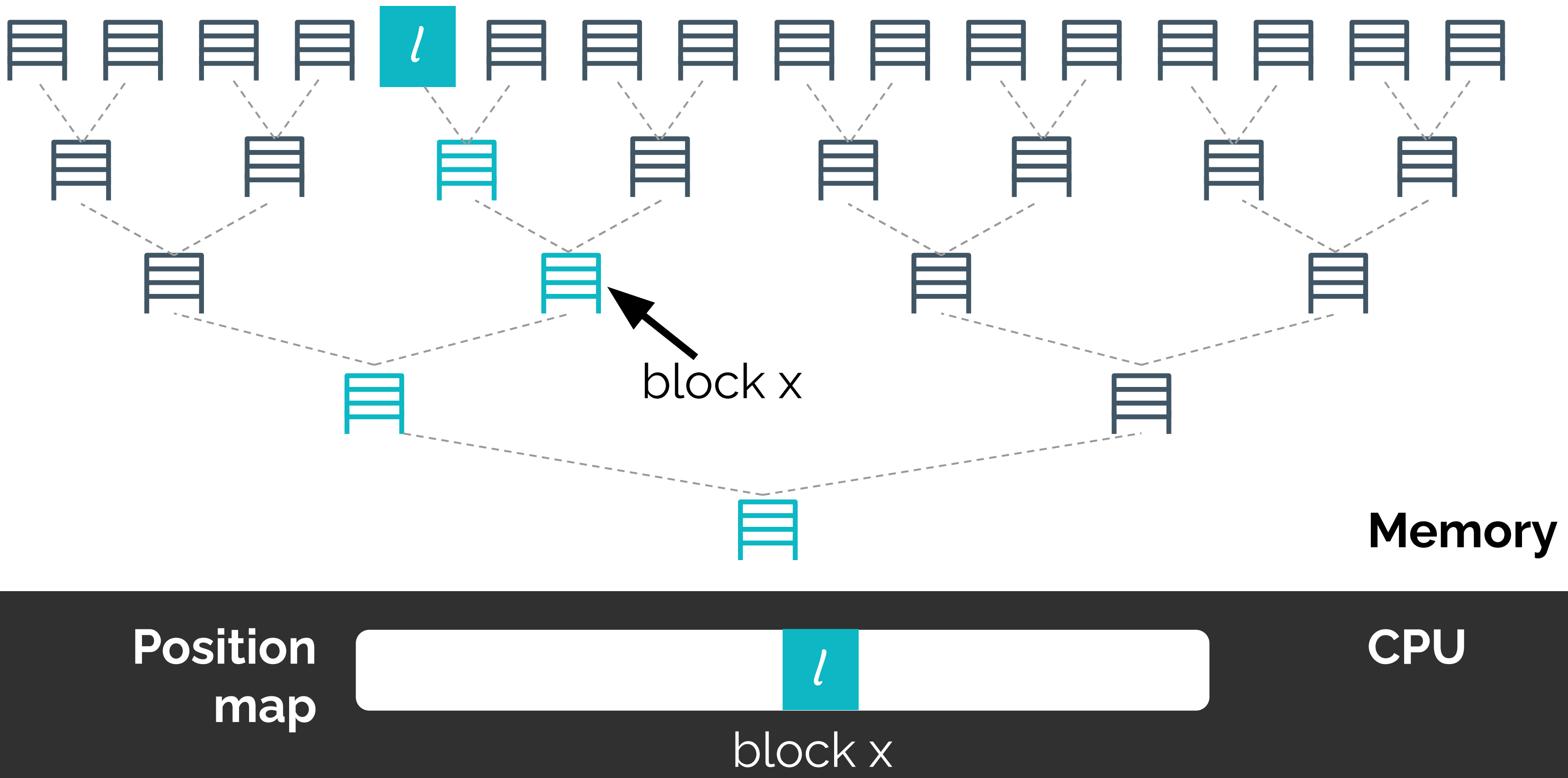
Memory

CPU

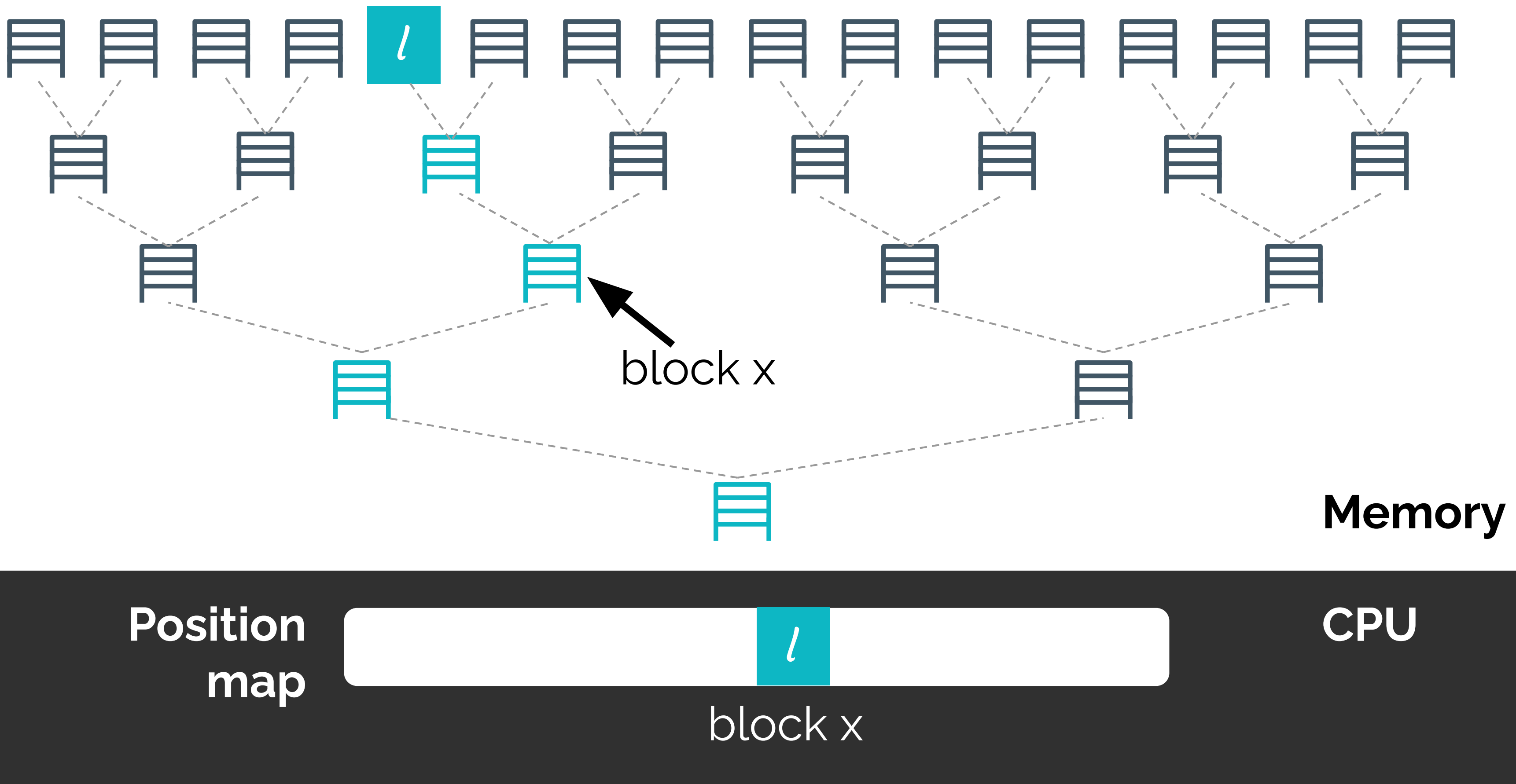
Each bucket stores real and filler blocks



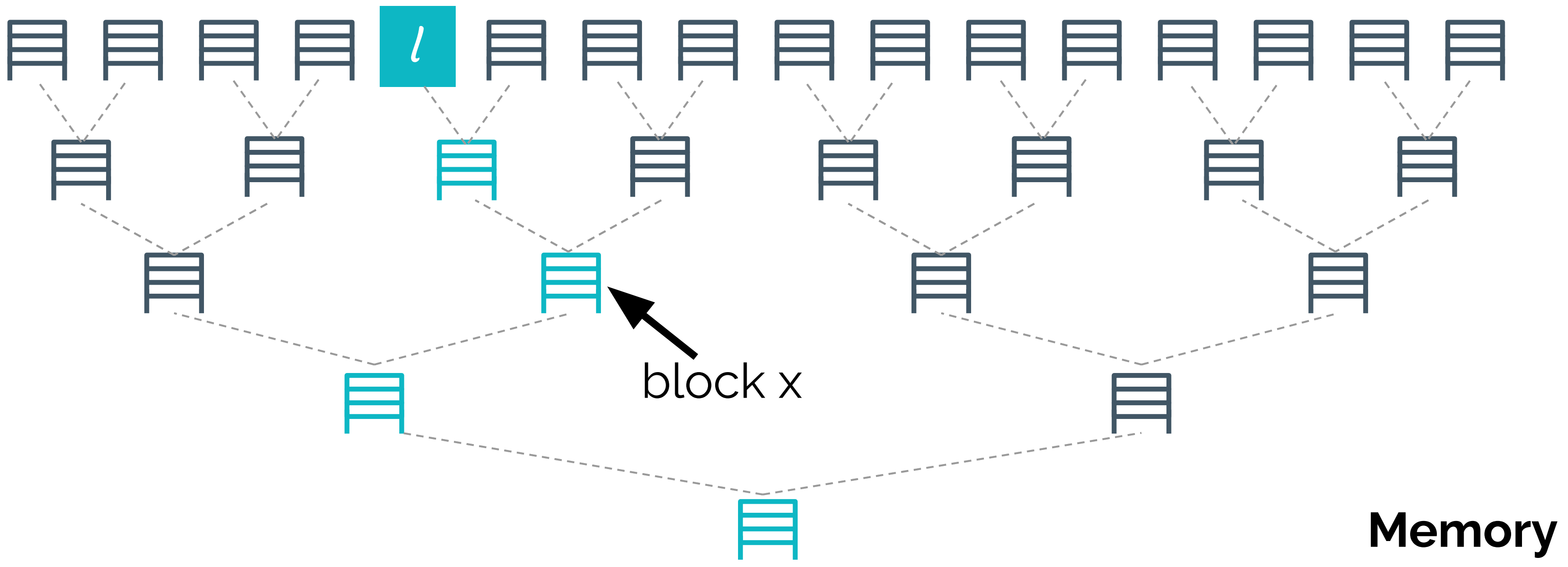
Path invariant: every block mapped to a random path

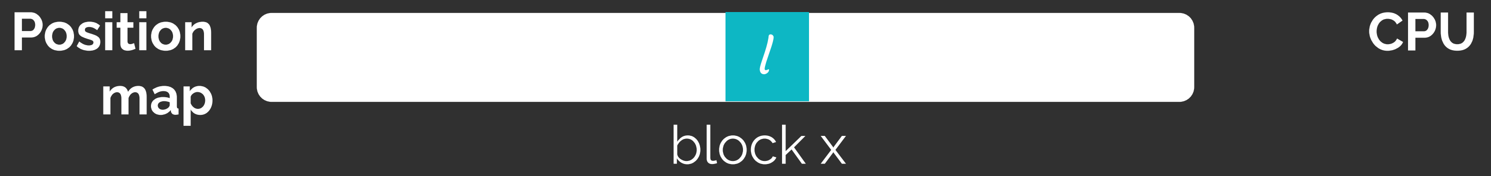


Reading a block is simple!

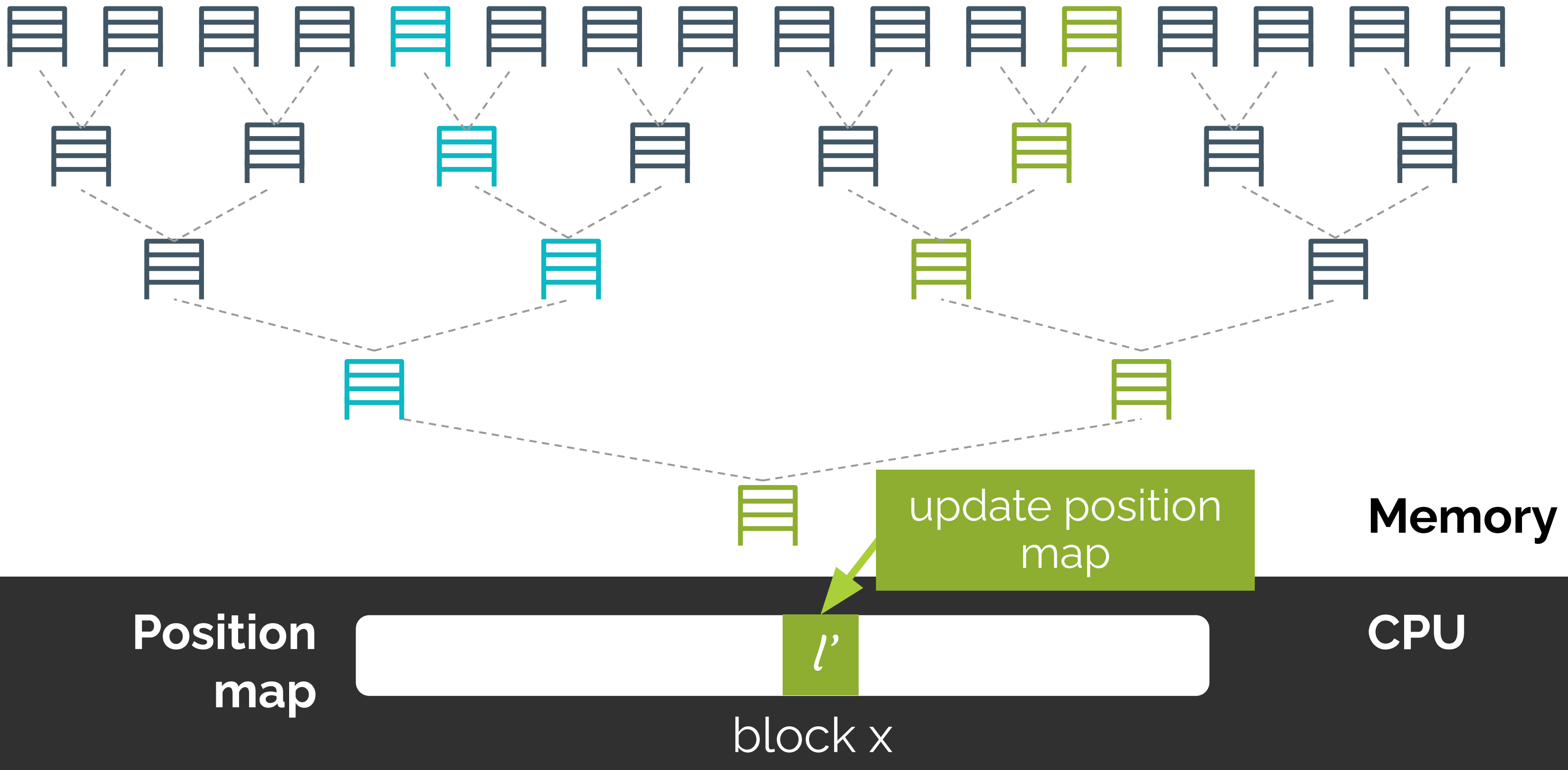


After being read, block x must relocate!

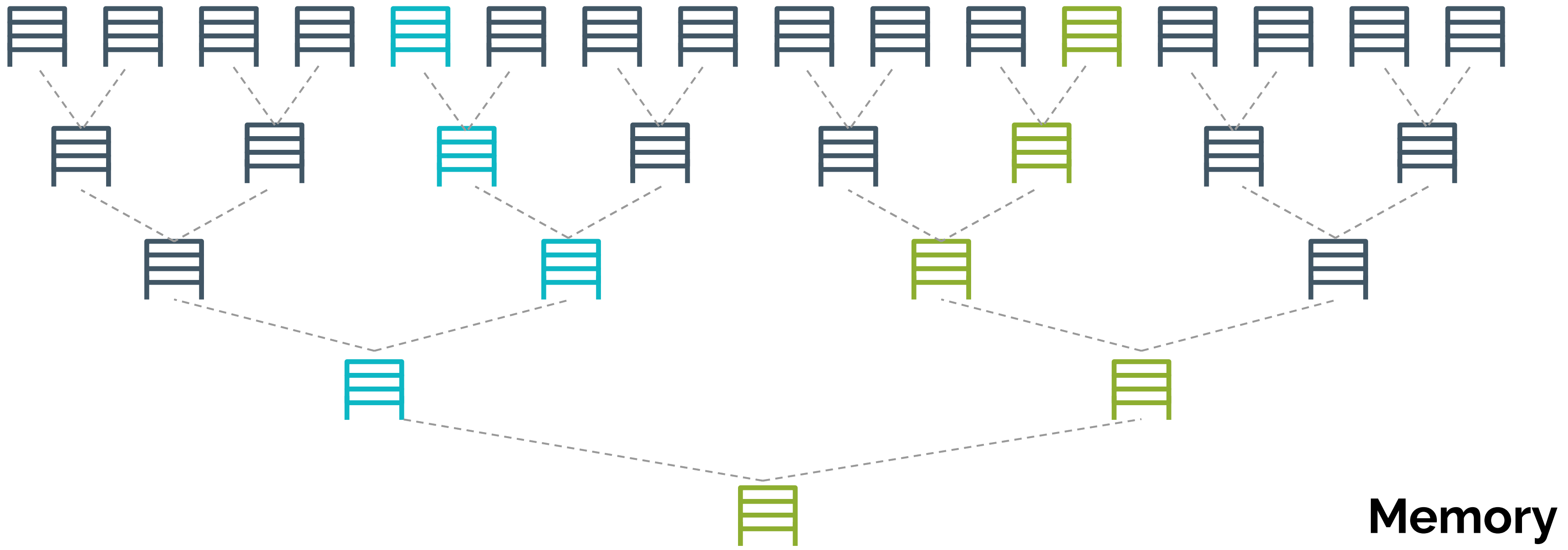


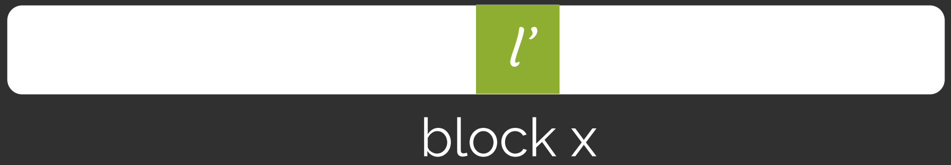


Pick a new random path and move x there



Where on the new path can we write block x ?

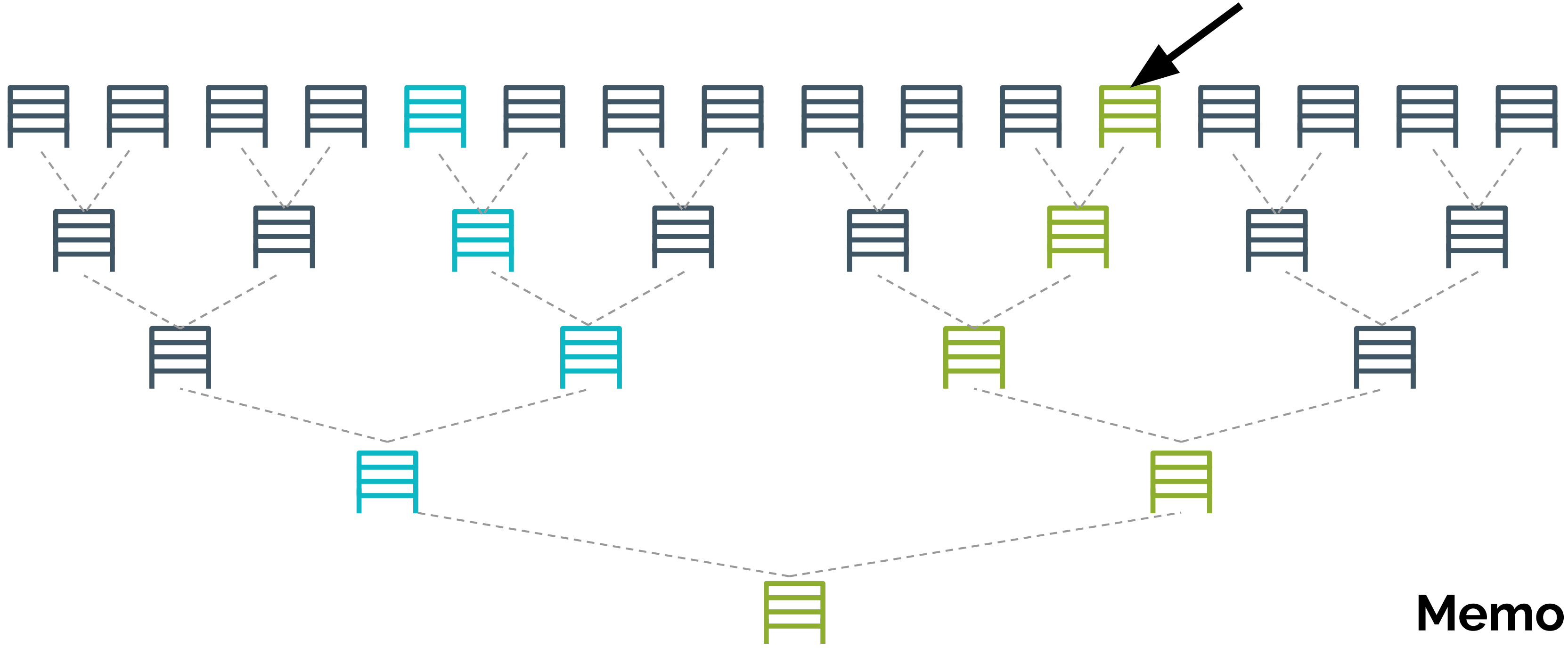


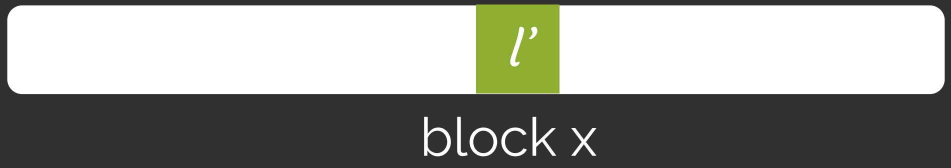


Position map

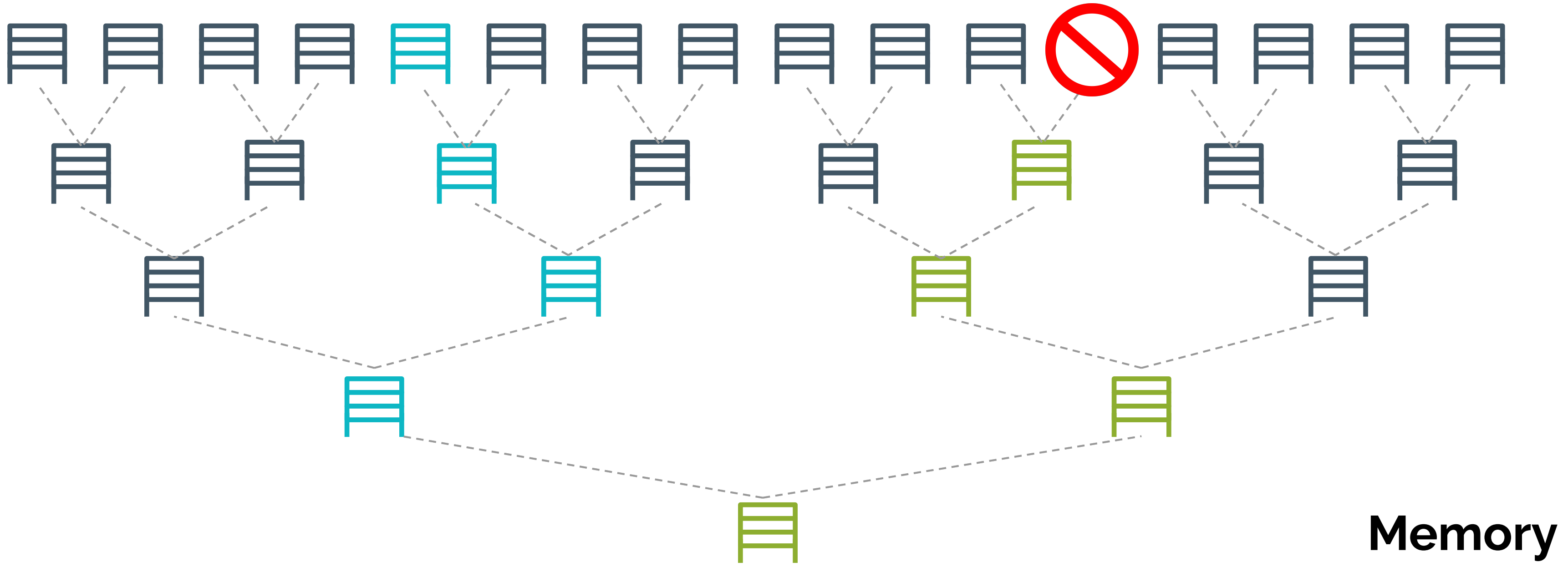
CPU

Can we write it to the leaf?





Can we write it to the leaf?



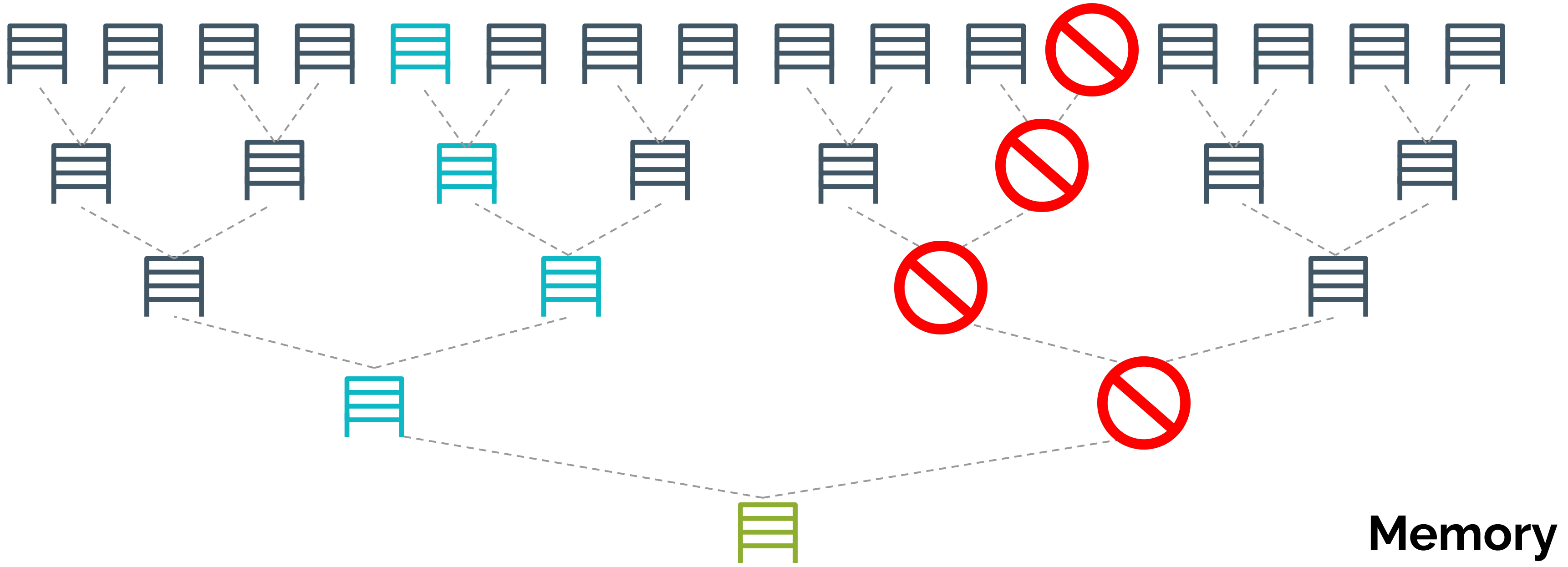
Position map

l'

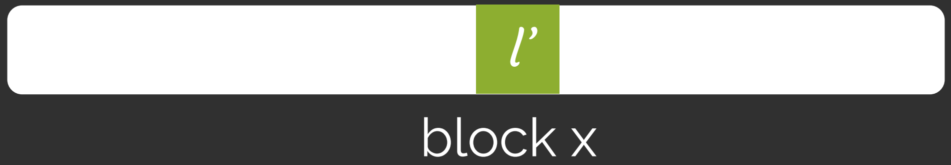
block x

CPU

Writing to any non-root bucket leaks information

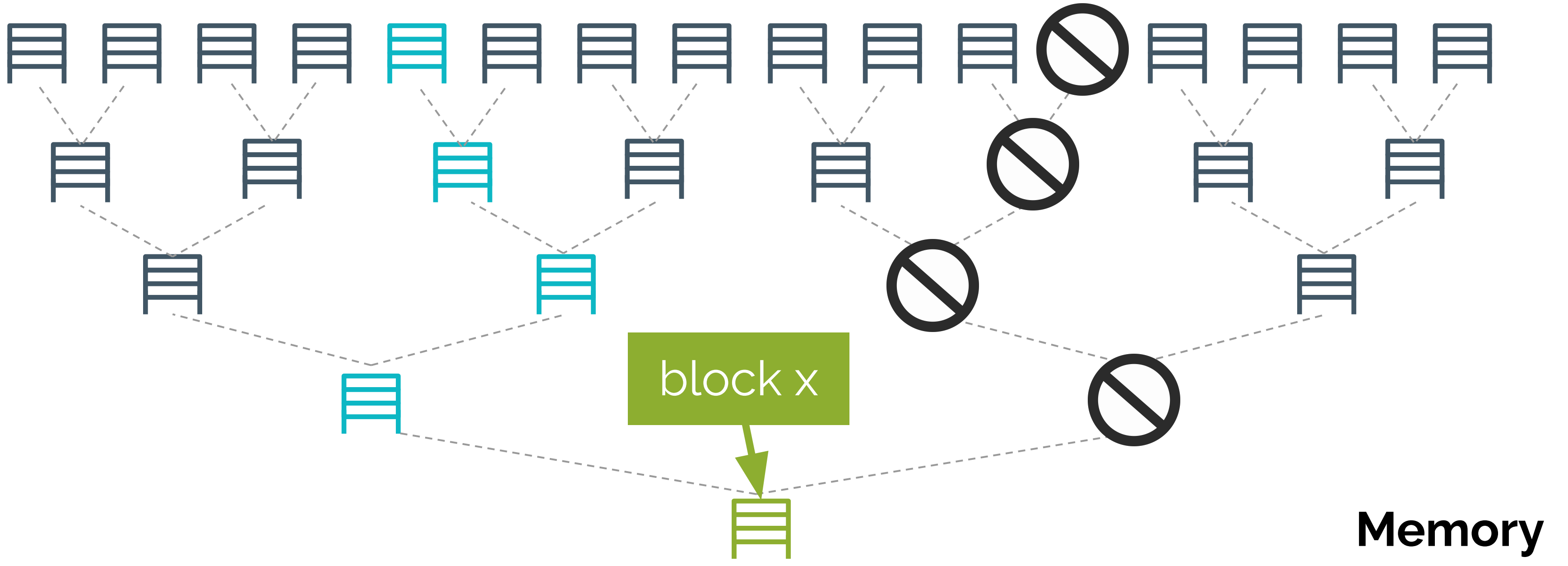


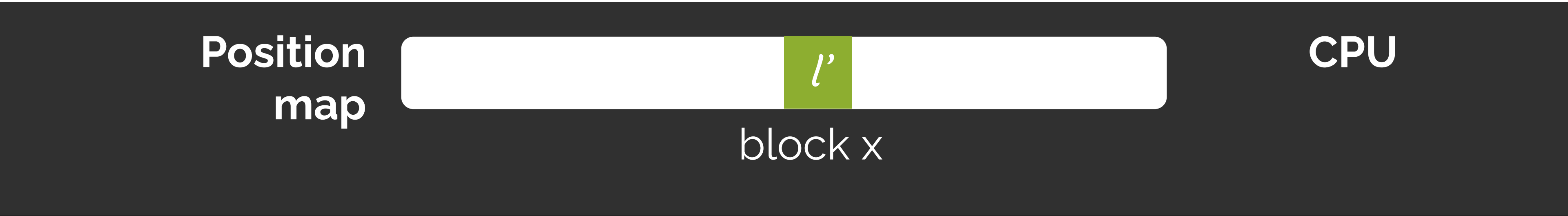
Position map



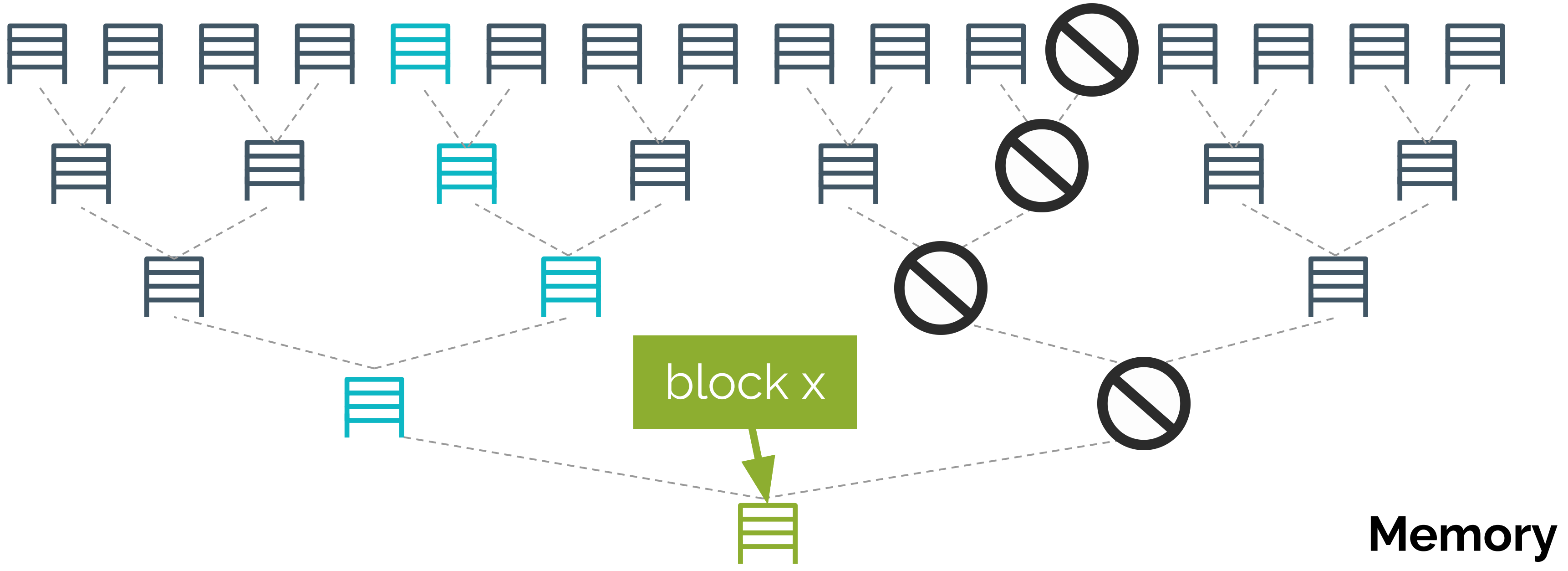
CPU

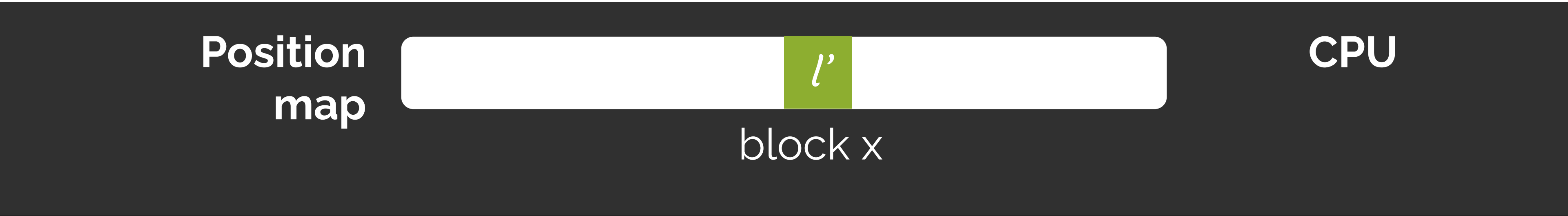
Write it to the root!



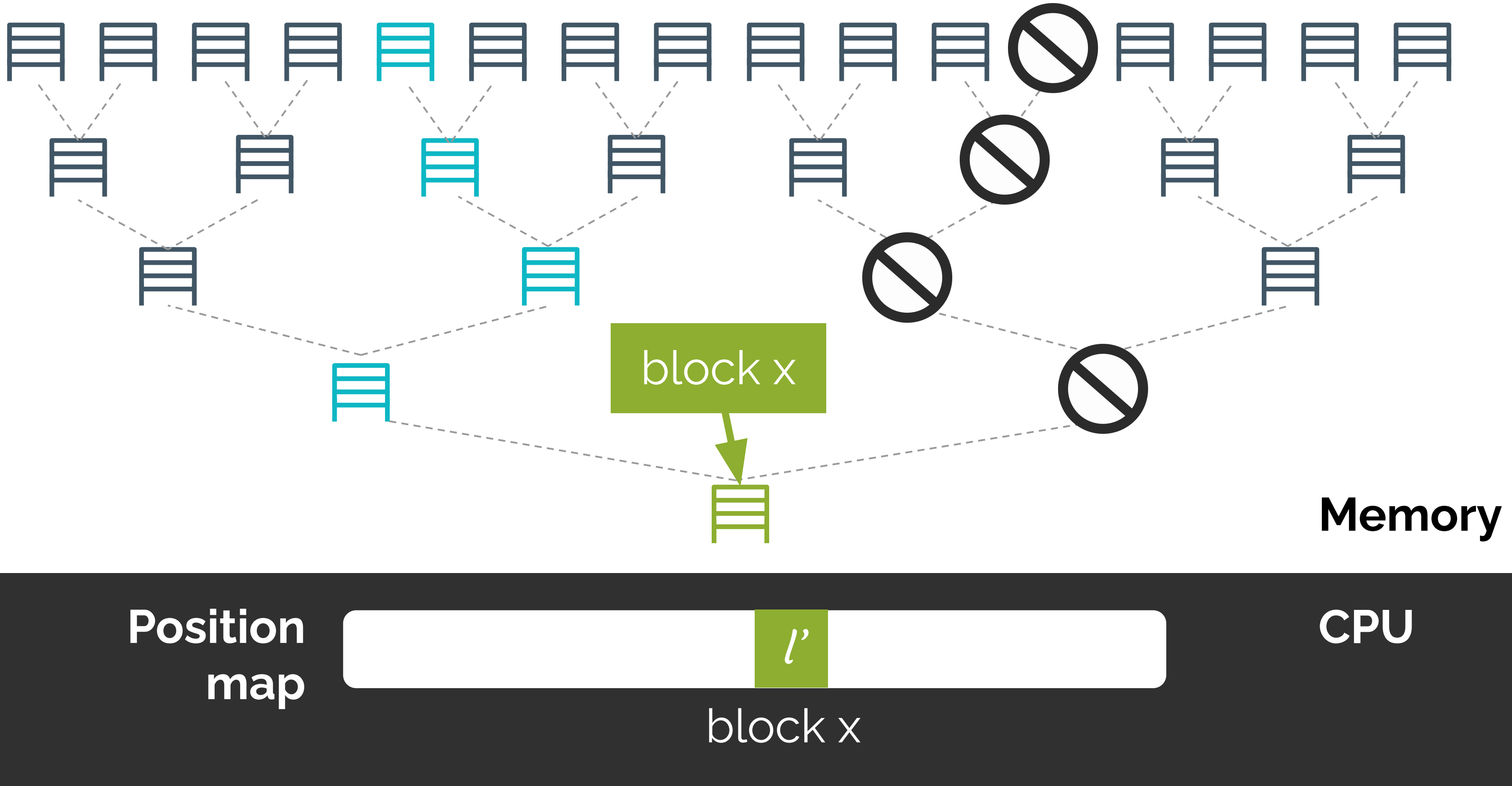


Security: every request, visit a **random** path that has **not** been revealed

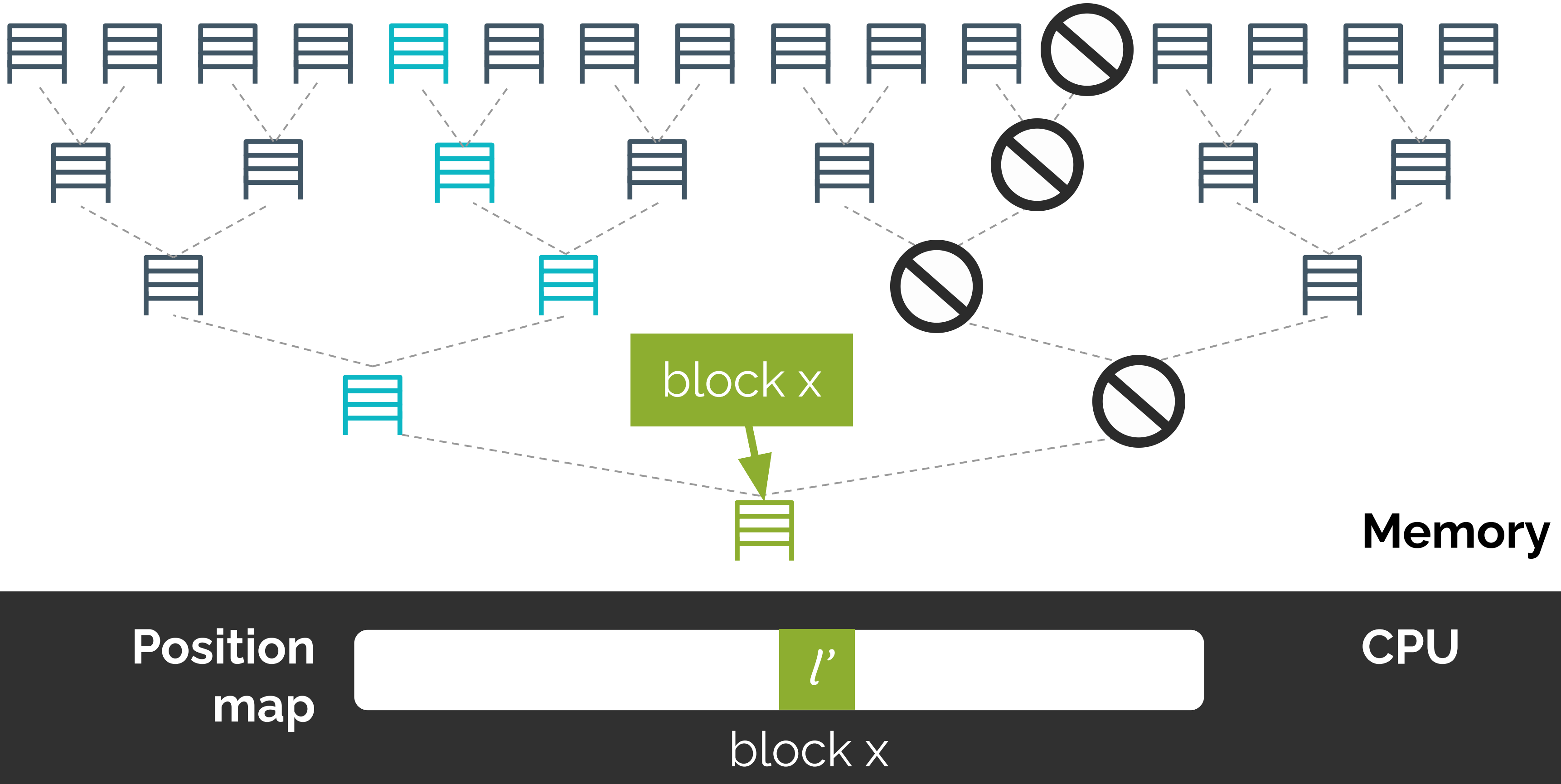




Problem?



Problem: root will overflow



Remaining issues

Resolve overflow

Remove position map

Resolve overflow

Eviction moves blocks towards leaves

Remove position map

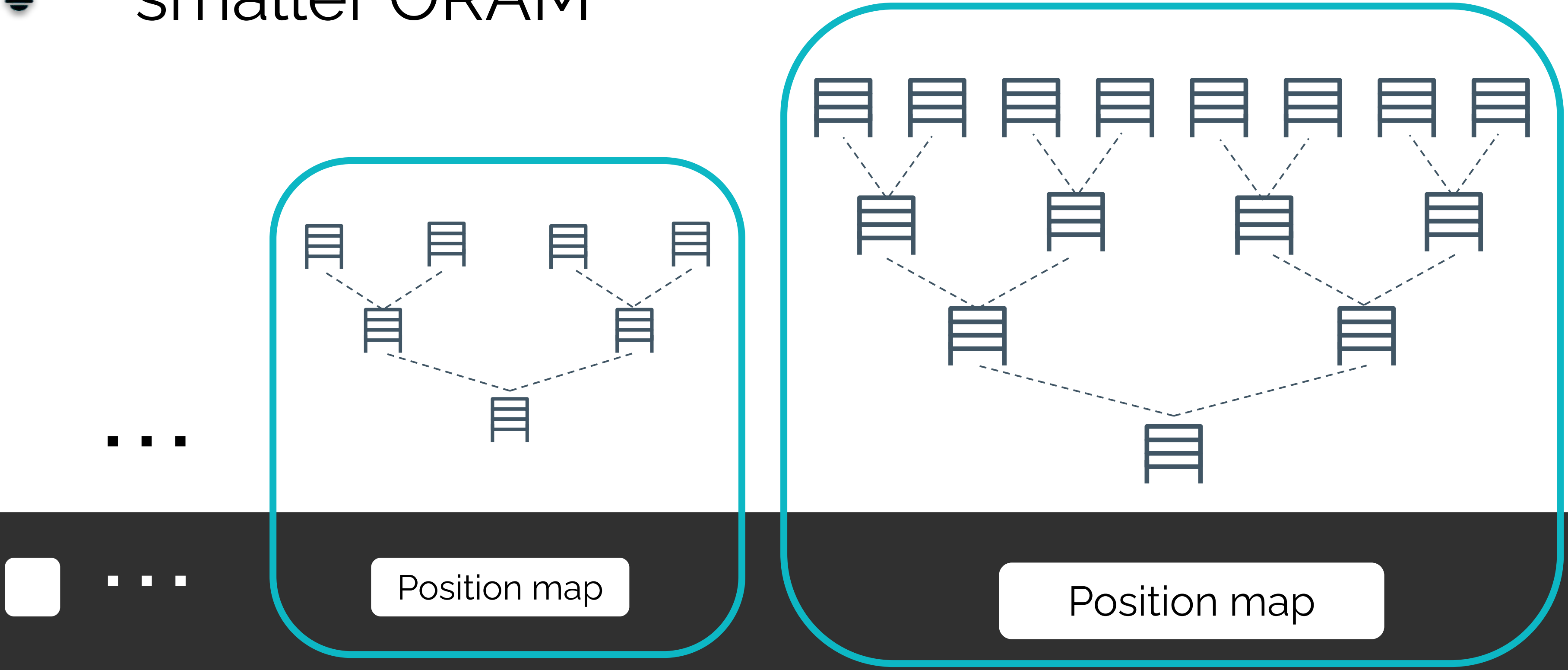
Resolve overflow

Eviction moves blocks towards leaves

Remove position map

Recursion

Store position map **recursively** in a smaller ORAM



Path ORAM

[SDS+'13]

```
$x \leftarrow$ position[a]
position[a] $\leftarrow$ UniformRandom$(0 \ldots 2^L - 1)$
for $\ell \in \{0, 1, \ldots, L\}$ do
    $S \leftarrow S \cup$ ReadBucket$(\mathcal{P}(x, \ell))$
end for
data $\leftarrow$ Read block a from $S$
if op = write then
    $S \leftarrow (S - \{(\mathsf{a}, \mathsf{data})\}) \cup \{(\mathsf{a}, \mathsf{data}^*)\}$
end if
for $\ell \in \{L, L-1, \ldots, 0\}$ do
    $S' \leftarrow \{(\mathsf{a}', \mathsf{data}') \in S : \mathcal{P}(x, \ell) = \mathcal{P}(\mathsf{position}[\mathsf{a}'], \ell)\}$
    $S' \leftarrow$ Select $\min(|S'|, Z)$ blocks from $S'$.
    $S \leftarrow S - S'$
    WriteBucket$(\mathcal{P}(x, \ell), S')$
end for
return data
```

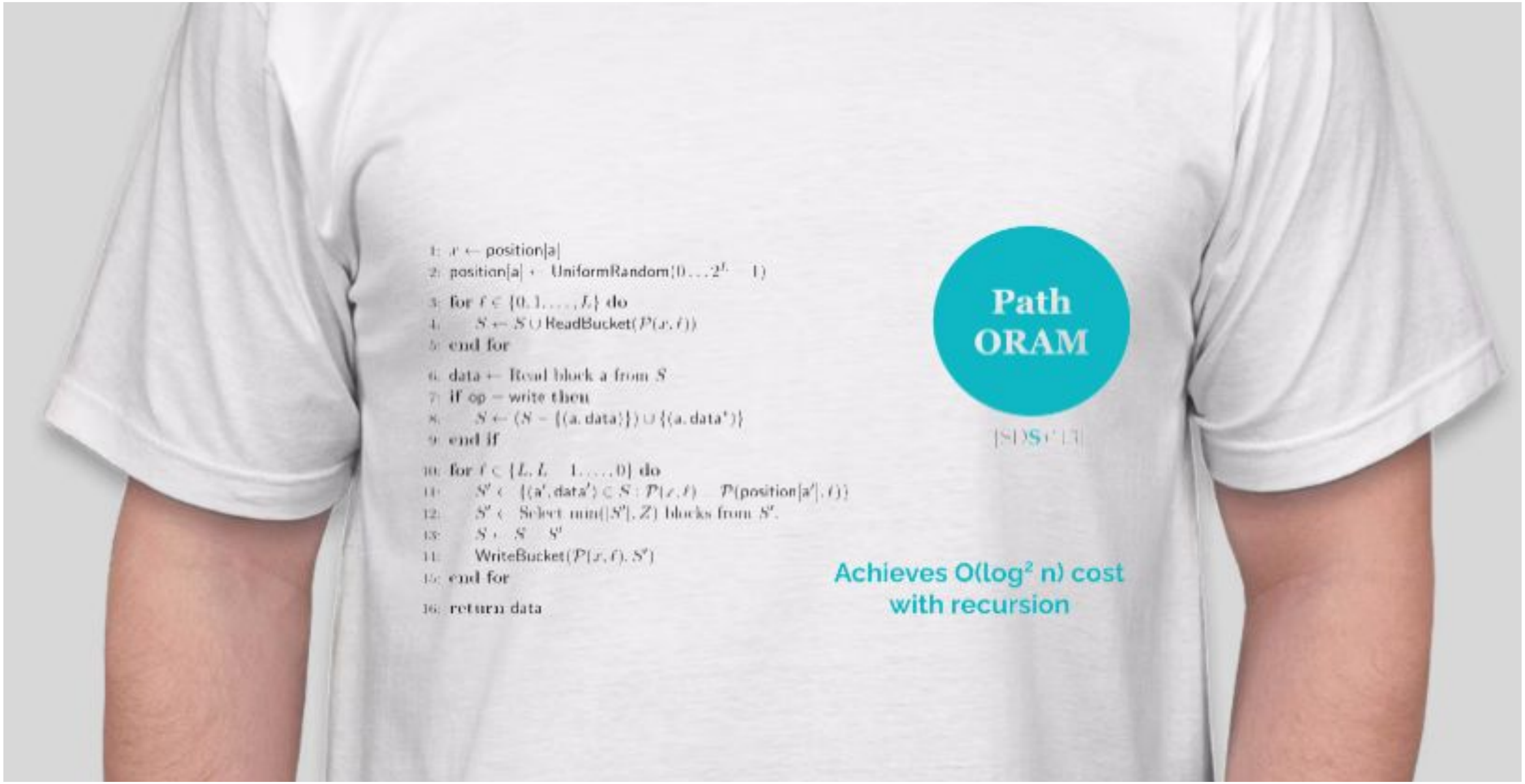
Achieves $O(\log^2 n)$ cost with recursion

```
1: $x \leftarrow$ position[a]
2: position[a] $\leftarrow$ UniformRandom$(0 \ldots 2^L - 1)$
3: for $\ell \in \{0, 1, \ldots, L\}$ do
4:     $S \leftarrow S \cup$ ReadBucket$(\mathcal{P}(x, \ell))$
5: end for
6: data $\leftarrow$ Read block a from $S$
7: if op = write then
8:     $S \leftarrow (S - \{(\mathsf{a}, \mathsf{data})\}) \cup \{(\mathsf{a}, \mathsf{data}^*)\}$
9: end if
10: for $\ell \in \{L, L-1, \ldots, 0\}$ do
11:     $S' \leftarrow \{(\mathsf{a}', \mathsf{data}') \in S : \mathcal{P}(x, \ell) = \mathcal{P}(\mathsf{position}[\mathsf{a}'], \ell)\}$
12:     $S' \leftarrow$ Select $\min(|S'|, Z)$ blocks from $S'$.
13:     $S \leftarrow S - S'$
14:     WriteBucket$(\mathcal{P}(x, \ell), S')$
15: end for
16: return data
```

Path ORAM

[SDS+'13]

Achieves O(log² n) cost with recursion



```
$x \leftarrow$ position[a]
position[a] $\leftarrow$ UniformRandom$(0 \ldots 2^L - 1)$
for $\ell \in \{0, 1, \ldots, L\}$ do
    $S \leftarrow S \cup$ ReadBucket$(\mathcal{P}(x, \ell))$
end for
data $\leftarrow$ Read block a from $S$
if op = write then
    $S \leftarrow (S - \{(\text{a}, \text{data})\}) \cup \{(\text{a}, \text{data}^*)\}$
end if
for $\ell \in \{L, L-1, \ldots, 0\}$ do
    $S' \leftarrow \{(\text{a}', \text{data}') \in S : \mathcal{P}(x, \ell) = \mathcal{P}(\text{position[a}'], \ell)\}$
    $S' \leftarrow$ Select $\min(|S'|, Z)$ blocks from $S'$.
    $S \leftarrow S - S'$
    WriteBucket$(\mathcal{P}(x, \ell), S')$
end for
return data
```

Summary: tree-based ORAMs

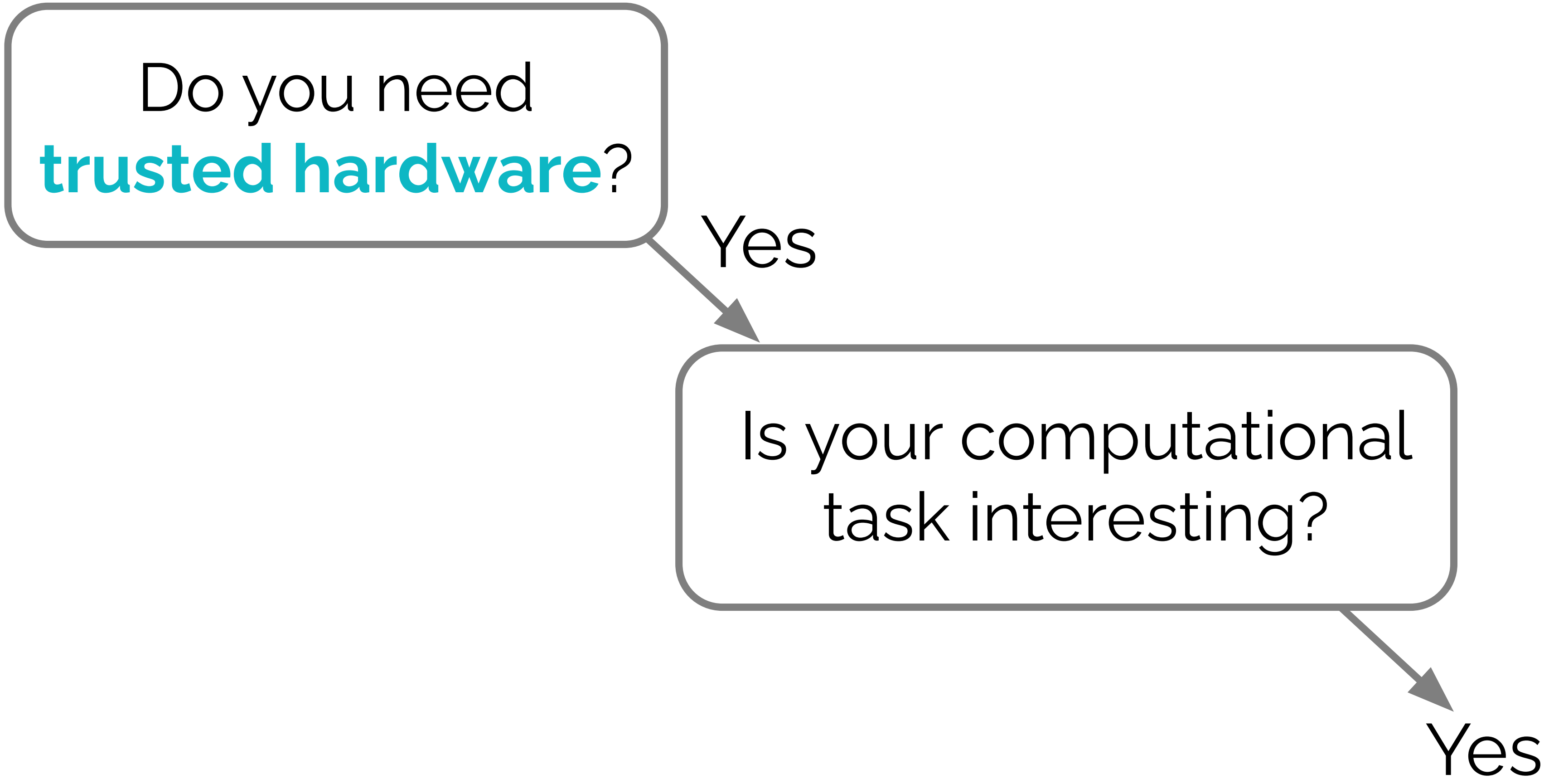
- A block is re-mapped to a new random path upon being read.
- The block must be relocated to the new path without revealing the new path
- Key challenge: design eviction process and prove no overflow.

Applications and challenges

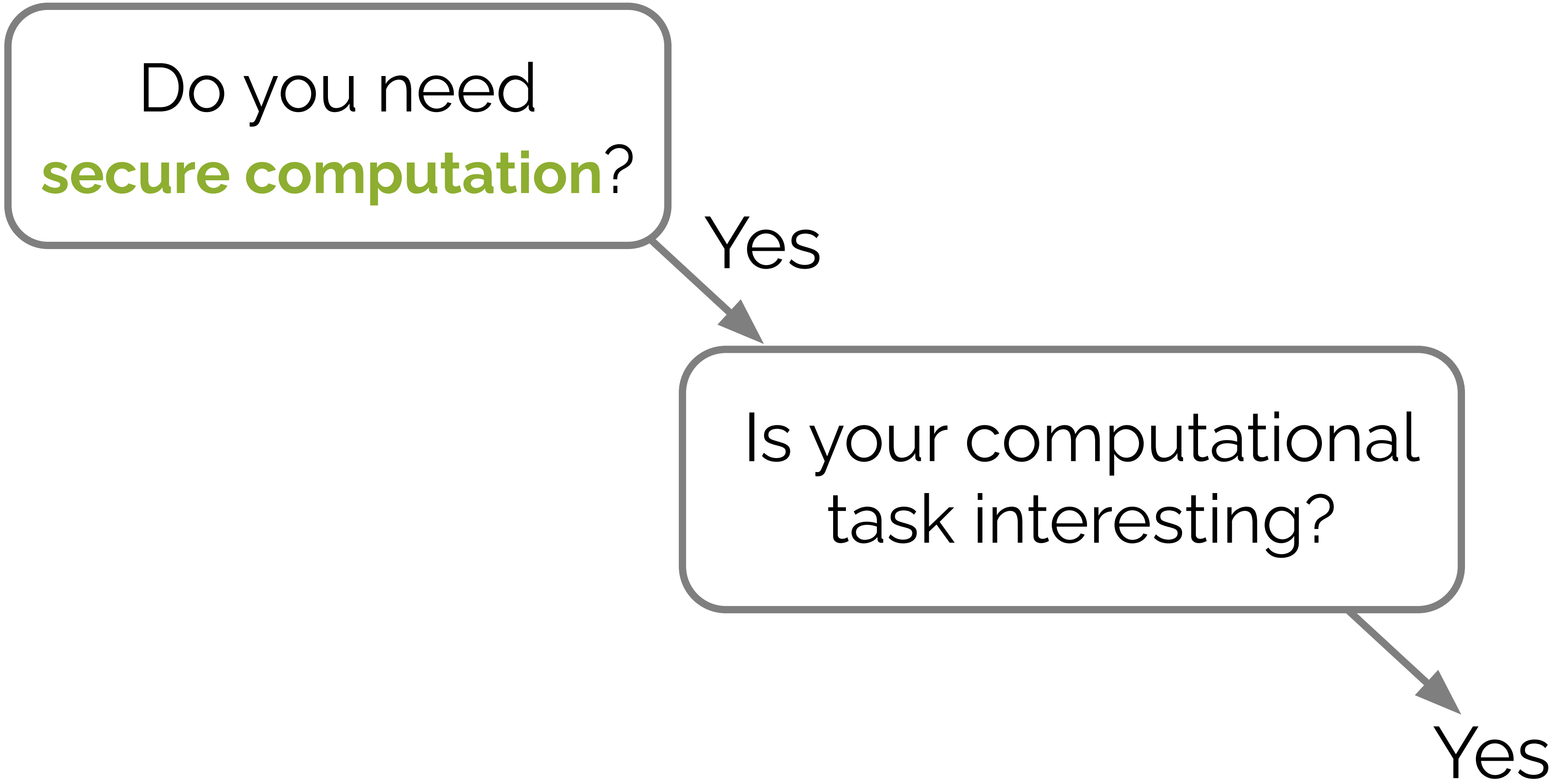
Oblivious RAM

Signal's story

Do you need an ORAM?


Do you need **trusted hardware**?
Yes
Is your computational task interesting?
Yes

Do you need an ORAM?


Do you need secure computation?
Yes
Is your computational task interesting?
Yes

ORAM for **blockchains**

- Privacy-preserving transactions and smart contracts secret Oasis PHALA Obscuro
- Flashbots use case
- Privacy-preserving light-weight clients

ORAM for **AI** (**ORAIM**)

- Retrieval augmentation for LLM?

(Ongoing work)

Oblivious STL: oblivious counterpart of the STL library

- data structures
  e.g. map, set, priority queue, range query
- sorting, shuffling
- common algorithms
  e.g. graph algorithms

Challenges for practical deployment

- Lack of awareness
- Generic ORAM vs efficient oblivious alg
- Mismatch of performance metrics
- Security of implementation

Challenges for practical deployment

- Lack of awareness

ZKP	ORAM
More awareness	Less awareness
More complicated	Simple algorithms
No one-size-fits-all scheme	Unified solution
Higher barrier of entry	Lower barrier

Challenges for practical deployment

- Lack of awareness
- Generic ORAM vs efficient oblivious alg

  e.g., data structures, sorting, shuffling, graph algorithms

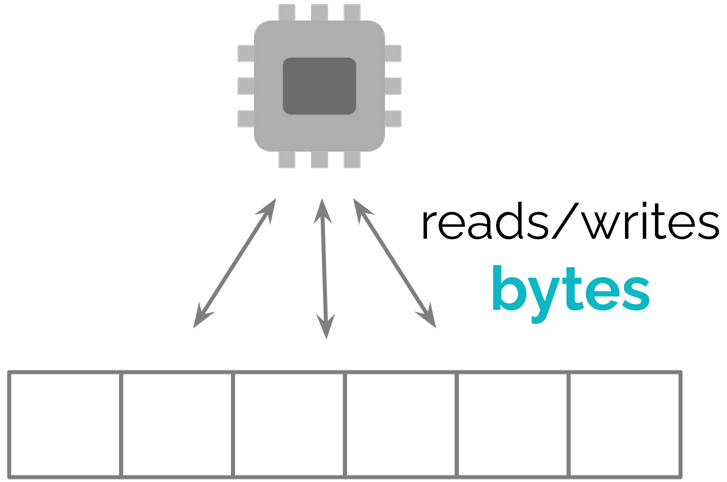
  [ZE'13, WNLCS+14, LWHNS'14, RS'21 ...]

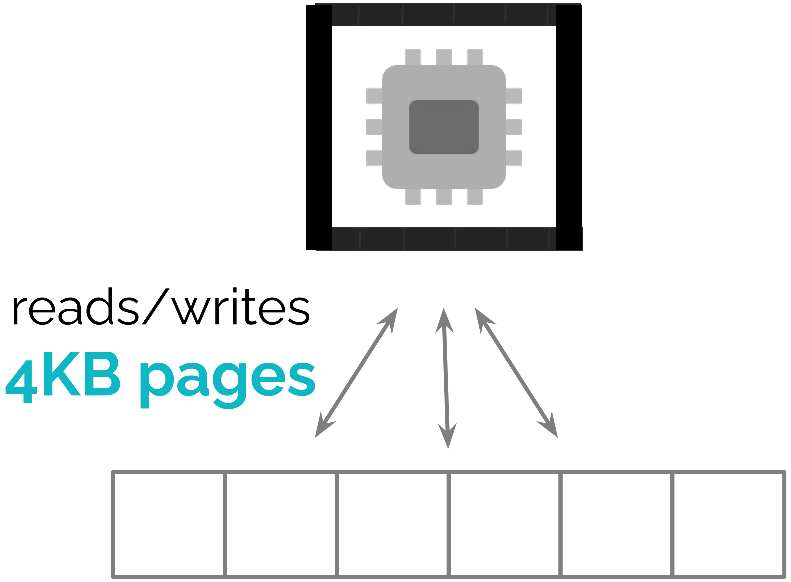
Challenges for practical deployment

- Lack of awareness
- Generic ORAM vs efficient oblivious alg
- Mismatch of performance metrics

ORAM/algorithms literature: **word RAM**

Secure enclaves: **external-memory**

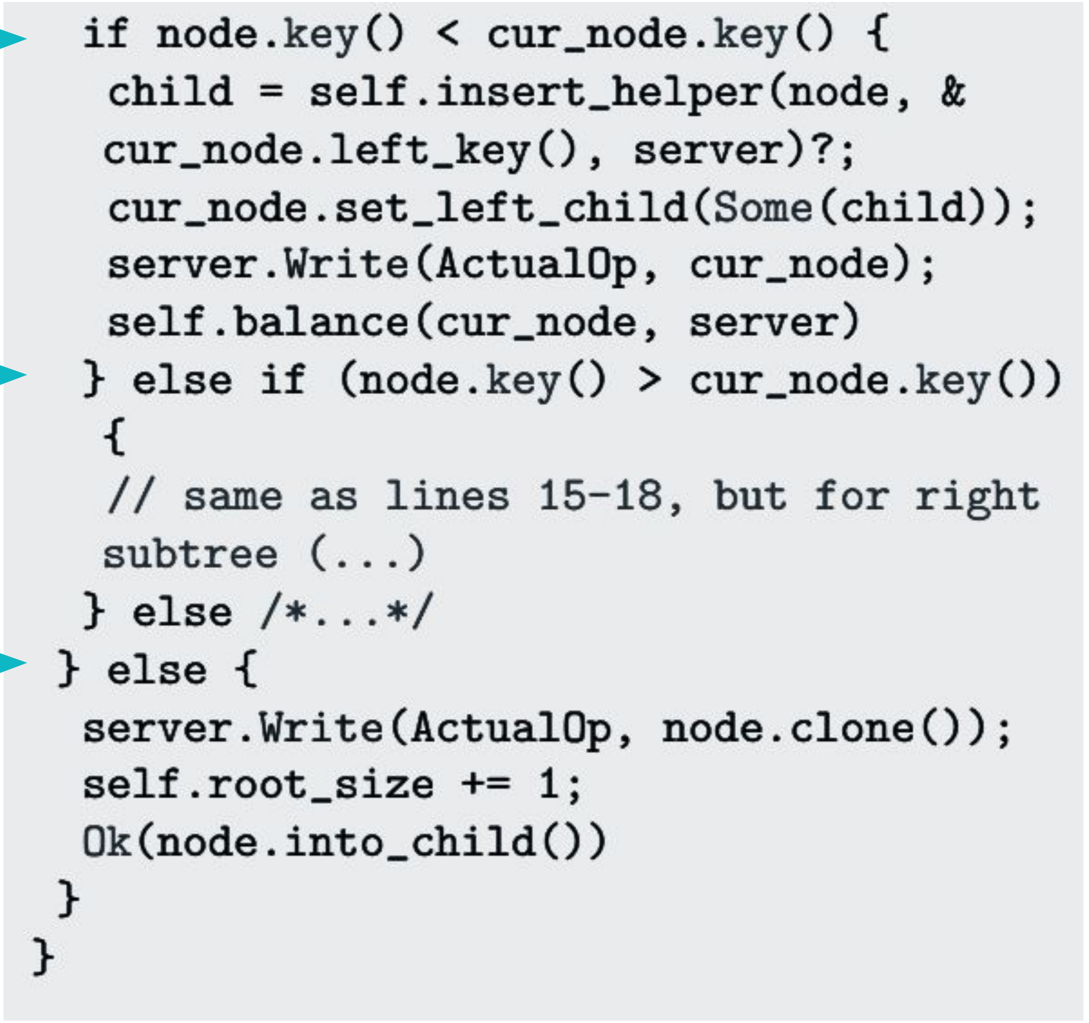




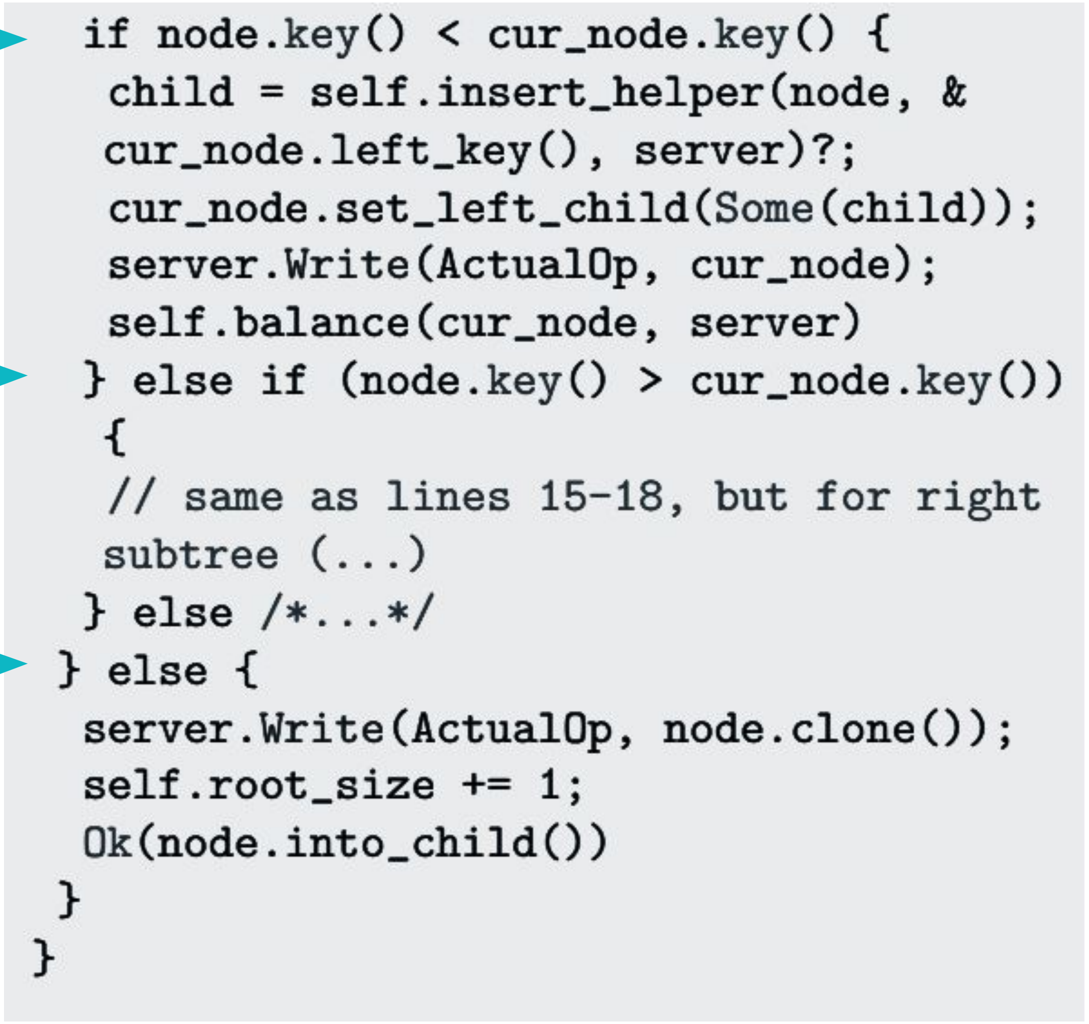
Challenges for practical deployment

- Lack of awareness
- Generic ORAM vs efficient oblivious alg
- Mismatch of performance metrics
- Security of implementation

Security flaw in Oblix



```
  if node.key() < cur_node.key() {
   child = self.insert_helper(node, &
   cur_node.left_key(), server)?;
   cur_node.set_left_child(Some(child));
   server.Write(ActualOp, cur_node);
   self.balance(cur_node, server)
  } else if (node.key() > cur_node.key())
   {
   // same as lines 15-18, but for right
   subtree (...)
  } else /*...*/
 } else {
  server.Write(ActualOp, node.clone());
  self.root_size += 1;
  Ok(node.into_child())
 }
}
```



```
    if node.key() < cur_node.key() {
     child = self.insert_helper(node, &
     cur_node.left_key(), server)?;
     cur_node.set_left_child(Some(child));
     server.Write(ActualOp, cur_node);
     self.balance(cur_node, server)
    } else if (node.key() > cur_node.key())
     {
     // same as lines 15-18, but for right
     subtree (...)
    } else /*...*/
   } else {
    server.Write(ActualOp, node.clone());
    self.root_size += 1;
    Ok(node.into_child())
   }
  }
```

[LHS13, LHHTMS15, DSLH'20]

Memory-trace oblivious type system

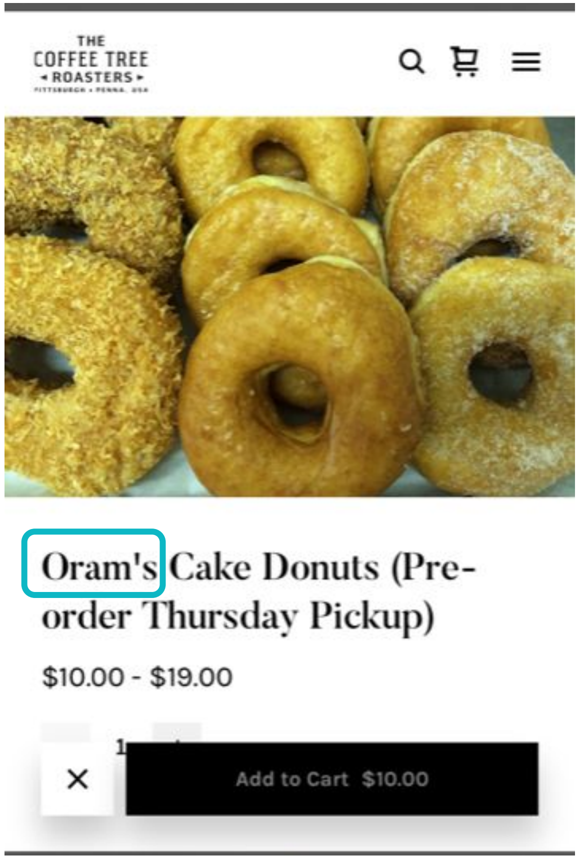
Oblivious STL: preliminary open-source

https://github.com/odslib/

https://github.com/obliviousram

Do you need an ORAM?

YES



Oram's Cake Donuts (Pre-order Thursday Pickup)

$10.00 - $19.00

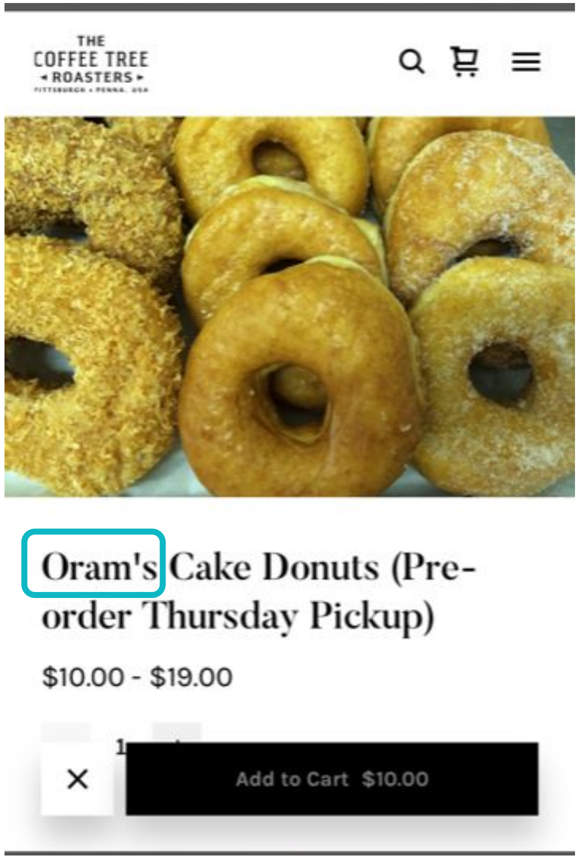
Do you need
an ORAM?

↓

YES

Thank you!

runting@cs.cmu.edu



Oram's Cake Donuts (Pre-order Thursday Pickup)

$10.00 - $19.00