Constant Communication Oblivious RAM*

Tarik Moataz

June 2nd 2016

Aarhus MPC workshop 2016

Joint work with Travis Mayberry and Erik-Oliver Blass

Part I

ORAM Overview

Part II

C-ORAM*: Constant Communication ORAM with homomorphic Encryption

Part III

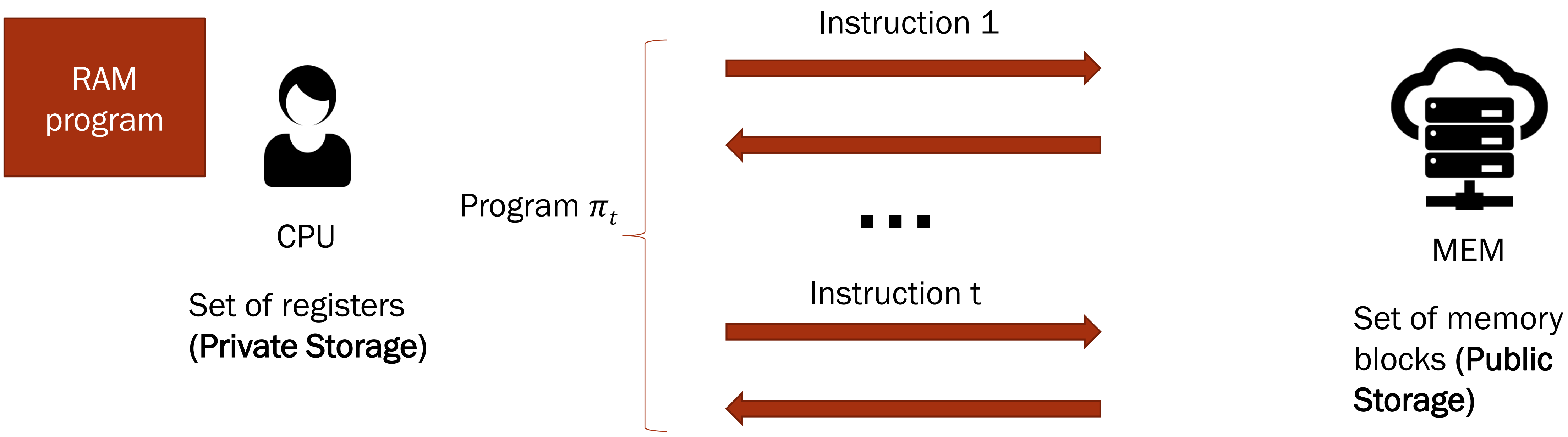
CH^f-ORAM**: Constant Communication ORAM without homomorphic Encryption

* *published at CCS'15*

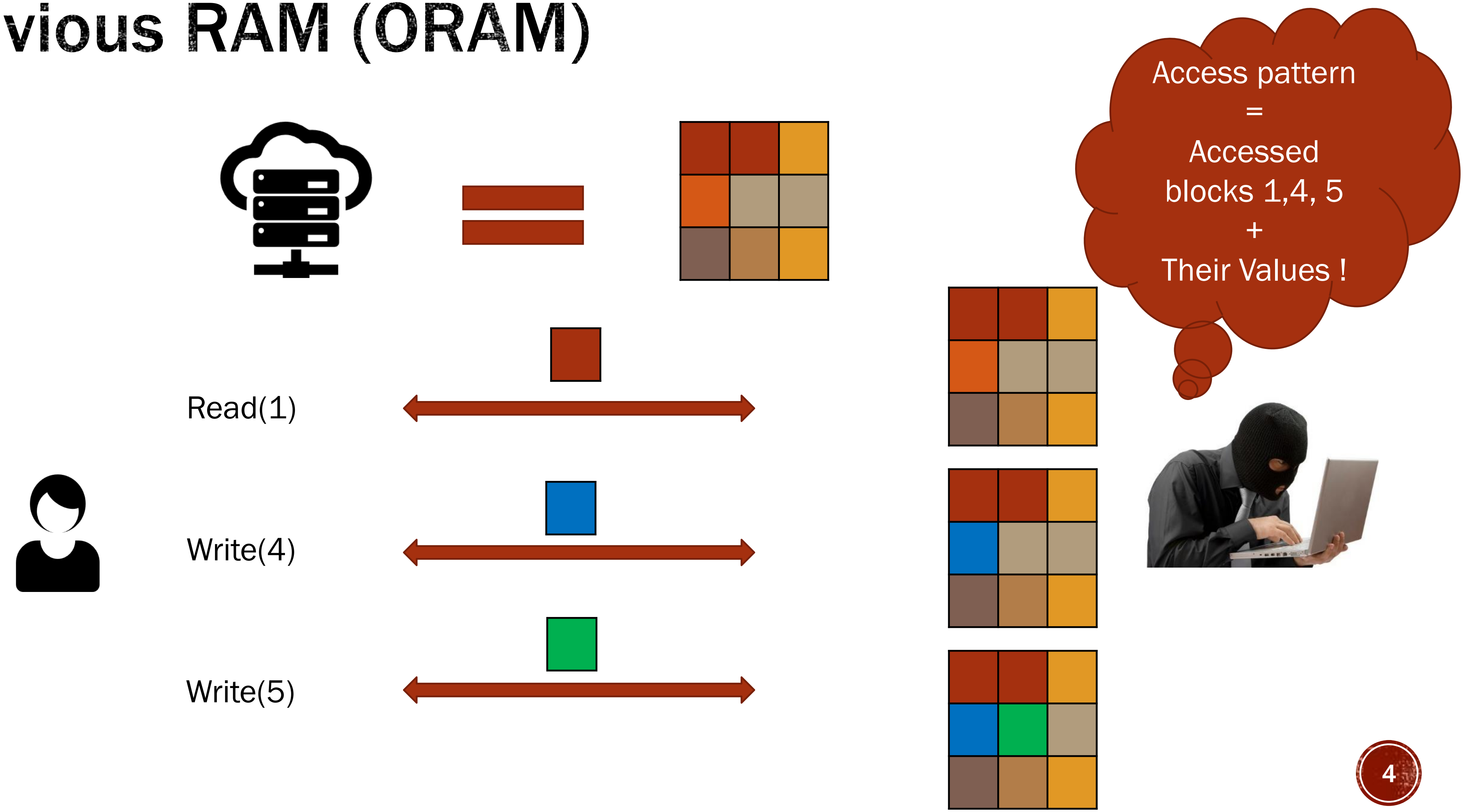
** *Work in progress*

Oblivious RAM (ORAM)

- ORAM first introduced by Goldreich in 87 further enhanced by Goldreich and Ostrovsky in 96



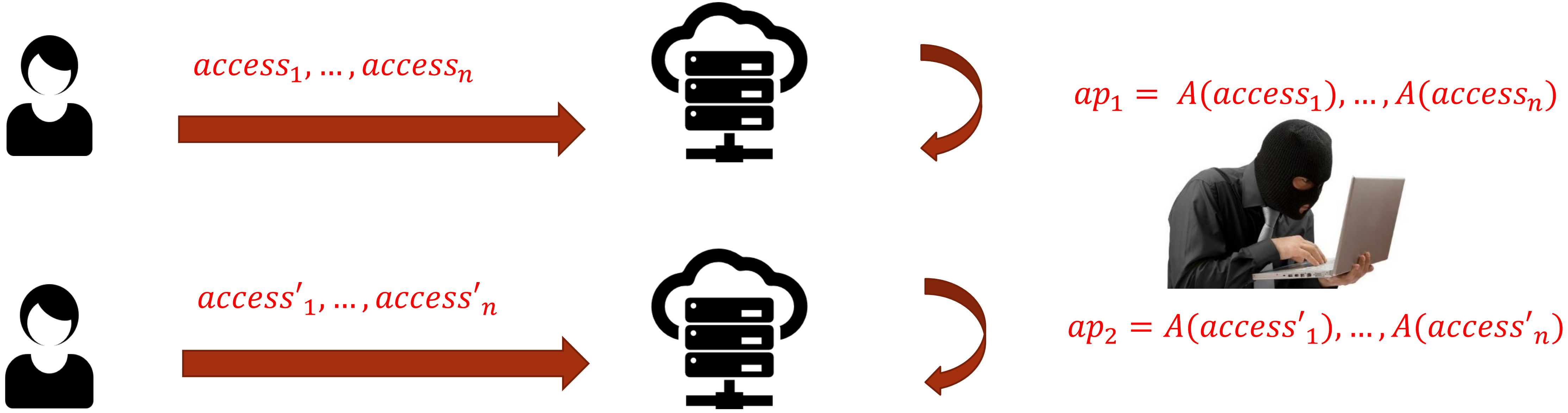
Oblivious RAM (ORAM)



What is an ORAM?

Picture from http://radix-communications.com/randomness/

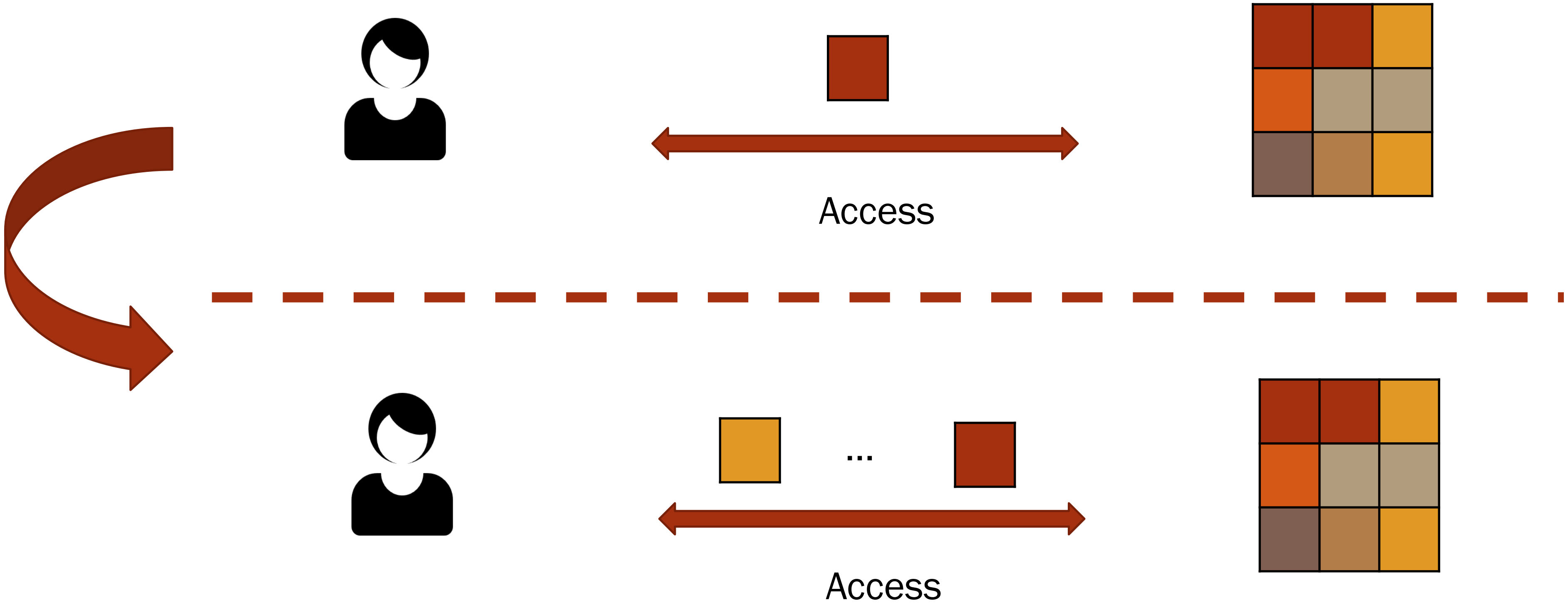
Security Definition of ORAM



- An access is either Read or Write
- For any probabilistic polynomial time adversary, the sequence ap_1 and ap_2 are indistinguishable
- We say that ORAM hides the access pattern

Oblivious RAM (ORAM)

Oblivious simulation of RAM



ORAM applications

Software Protection
G87

Cloud Storage
SS13a, SS13b

Garbled RAM
LO13

Secure RAM computation, MPC
OS97, GKKKMRV12,
GGHJRW13

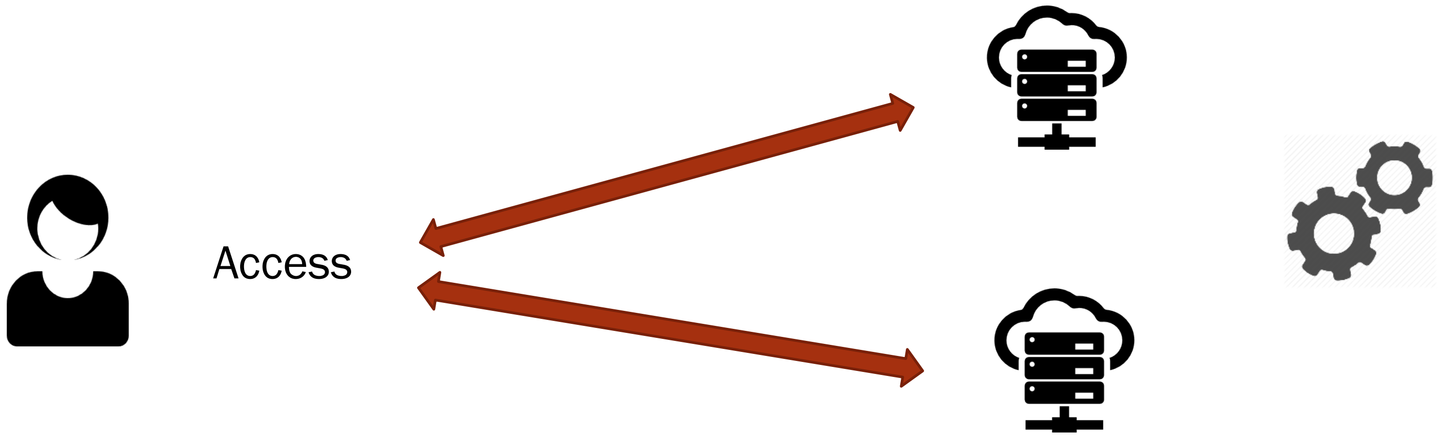
Privacy-preserving
WNLCSSH14, JMTS16*

* *Joint work with Shruti Tople, Yaoji Jia and Prateek Saxena to appear at USENIX'16*

ORAM settings

- Computational/non-computational (e.g., Onion ORAM, C-ORAM)

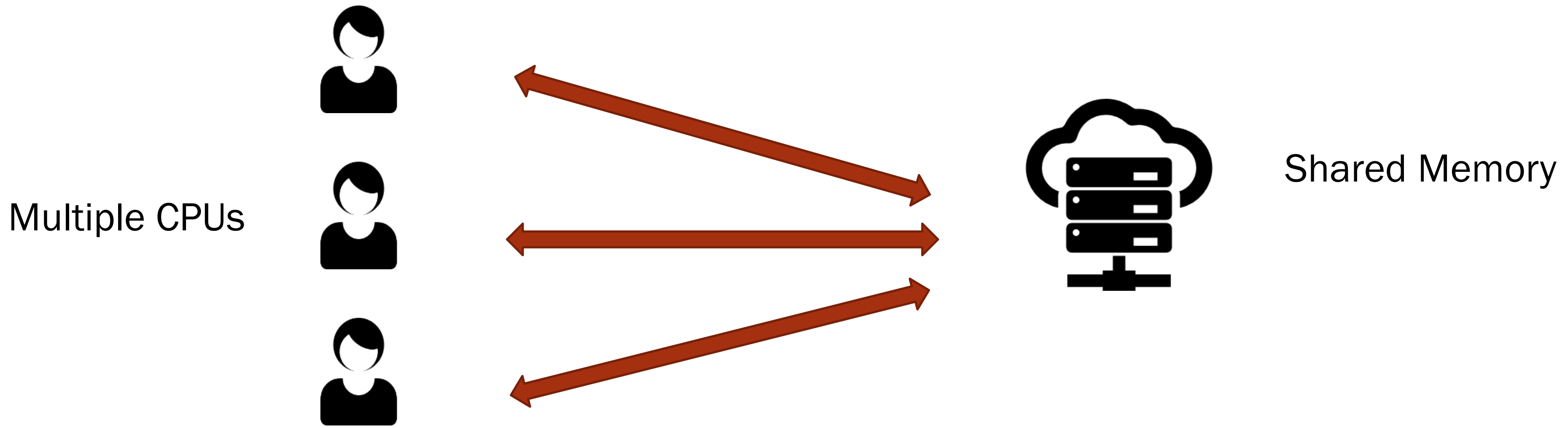
- One-server/Multi-servers (e.g., Multi Cloud SS13, Oblivious Network RAM DLPSV15, Private information Storage OS97)



(possible like in PIS)

ORAM settings

- One-CPU/Multiple CPUs (e.g., Oblivious Parallel RAM BCP16, CLT16)



- Computational HA / Information-theoretic secure (DMN11, A10)

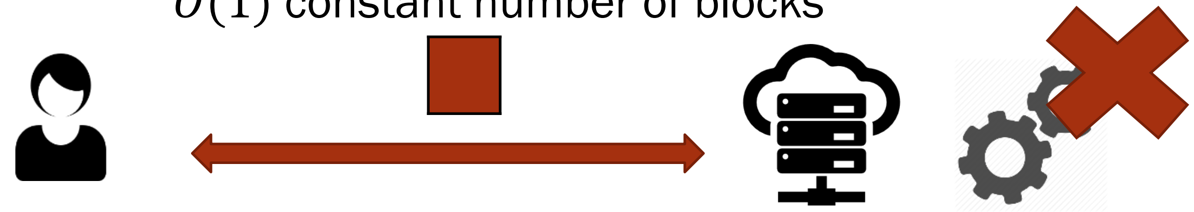
ORAM Main Metrics

- Worst-case communication overhead
- Private Storage
- Minimum Block Size
- Number of rounds
- MEM storage overhead
- Computational overhead

Ideally

- We want:
  - Constant Communication ORAM
  - Constant number of rounds
  - Very small Block Size
  - No Computation on the server Size
  - Constant Private Storage

$O(1)$ constant number of blocks



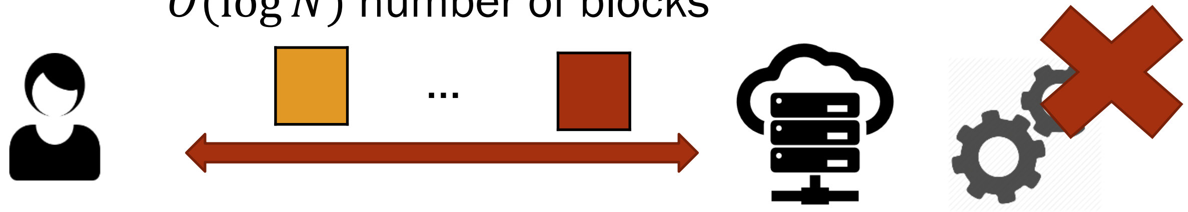
$O(1)$
private
storage

Unfortunately this is not possible

- Goldreich and Ostrovsky (GO96) lower bound of at least $\log N$ blocks
- In a one-server setting and without computation:

Ring/Path ORAM

$O(\log N)$ number of blocks



$O(\log N)$ private storage

Block size in $\Omega(\log^2 N)$

Fortunately

- GO lower bounds is based on Balls/bins and does not capture:
  - Encoding stored data and performing computation on outsourced data BN'15

$O(1)$ number of blocks

Onion ORAM

$O(1)$ private storage

Block size in $\Omega(\log^5 N)$

+

Very slow

Can we reduce computational overhead and block size?

C-ORAM

C-ORAM

$O(1)$ number of blocks

$O(1)$ private storage

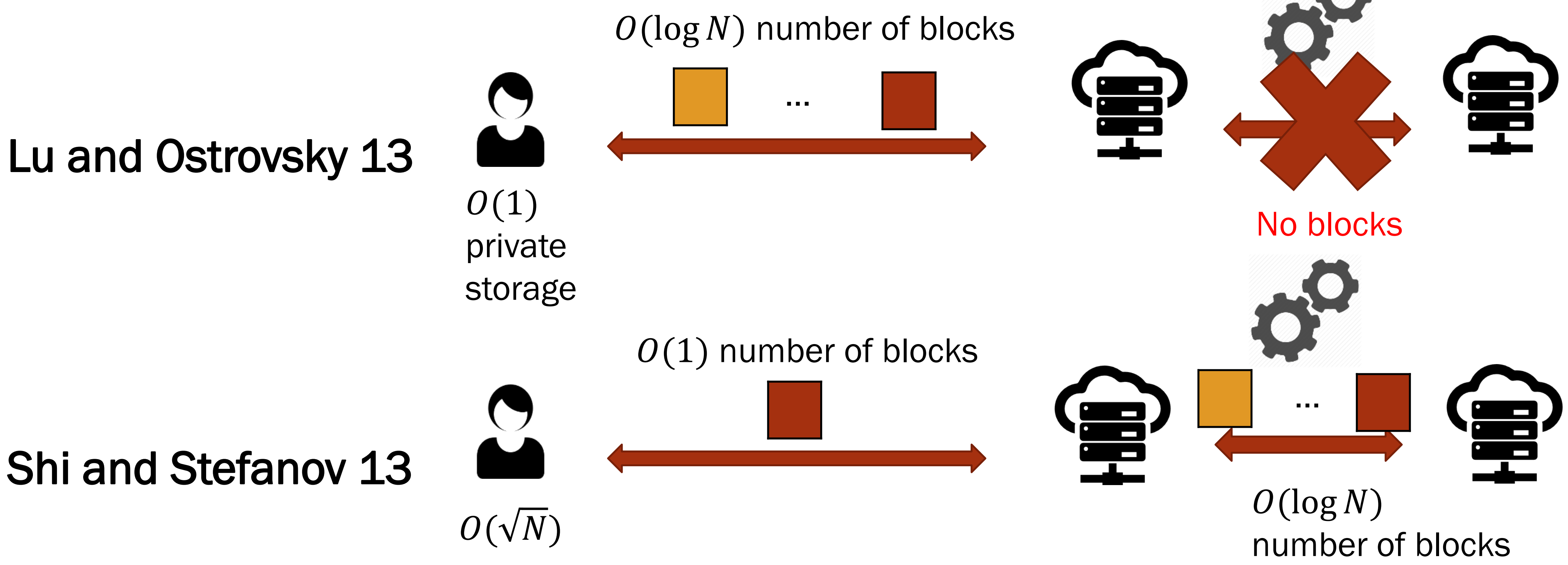
Block size in $\Omega(\log^4 N)$

+

10 times faster

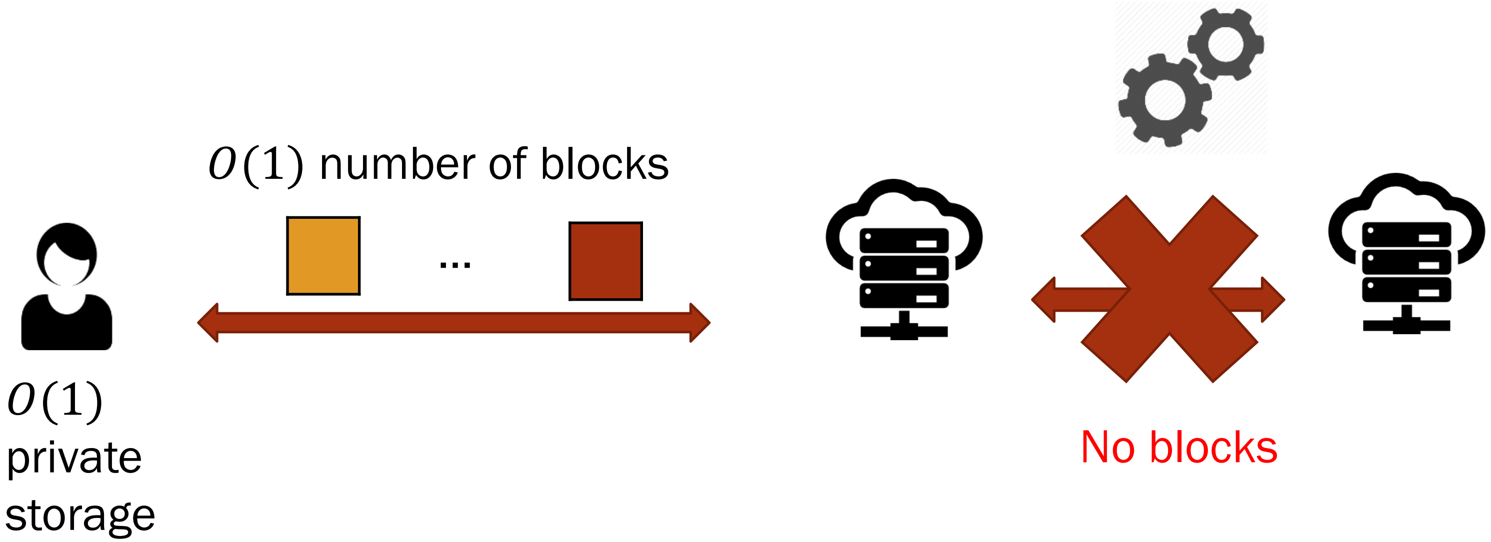
Impact of Multi-servers?

- GO lower bound does not capture multiple servers



CH^f-ORAM

- GO lower bounds does not capture multiple servers, Great!



Block size in
$\Omega(\log^3 N)$

What we can achieve so far

- We want:
  - Constant Communication ORAM ✓
  - Constant number of rounds Maybe, TWORAM, Bucket ORAM
  - Very small Block Size ✓
  - No Computation on the server Size ✗
  - Constant Private Storage ✓

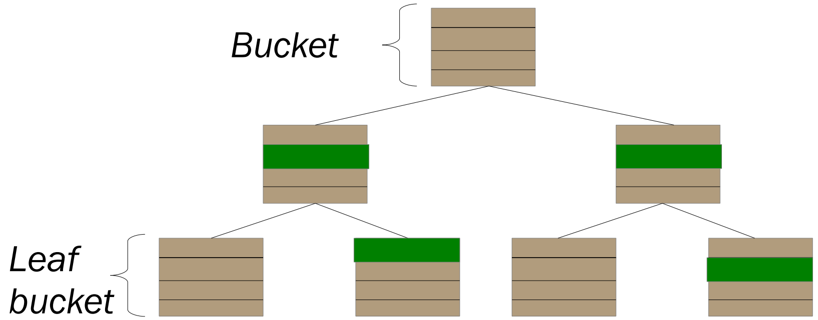
Computation should not annihilate constant communication

Tree-based ORAM

SCSL'11

Tree-based ORAM: SCSL'11

Structure and features



e_2	$leaf_1$
e_1	$leaf_2$
e_3	$leaf_4$
e_4	$leaf_3$

Position Map recursively stored

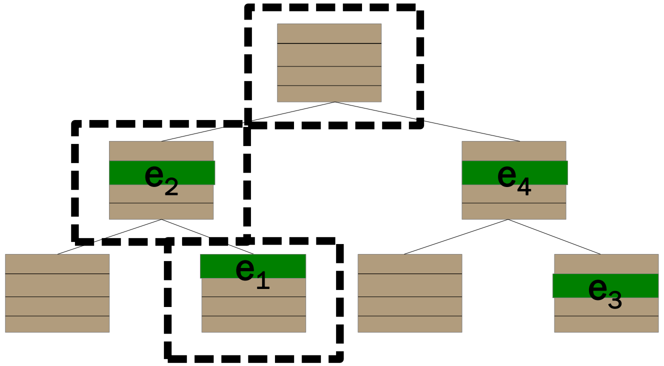
- Read and Write operations
  - Every element is defined by a leaf identifier
  - Every element read/updated is written in the root
- Eviction (Memory shuffle) process to percolate elements towards the leaves
- Recursive position Map

- Search complexity is **polylog**
- Bucket *size is a security parameter*

Tree-based ORAM: SCSL'11

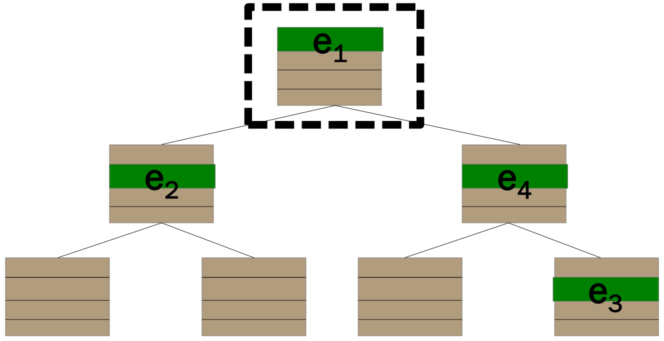
Read and Write operations

Step 1



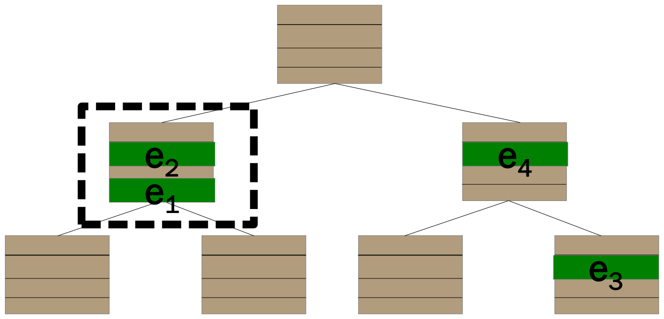
e_2	$leaf_1$
e_1	$leaf_2$
e_3	$leaf_4$
e_4	$leaf_3$

Step 2



e_2	$leaf_1$
e_1	$leaf_1$
e_3	$leaf_4$
e_4	$leaf_3$

Step 3



e_2	$leaf_1$
e_1	$leaf_1$
e_3	$leaf_4$
e_4	$leaf_3$

Part I

ORAM Overview

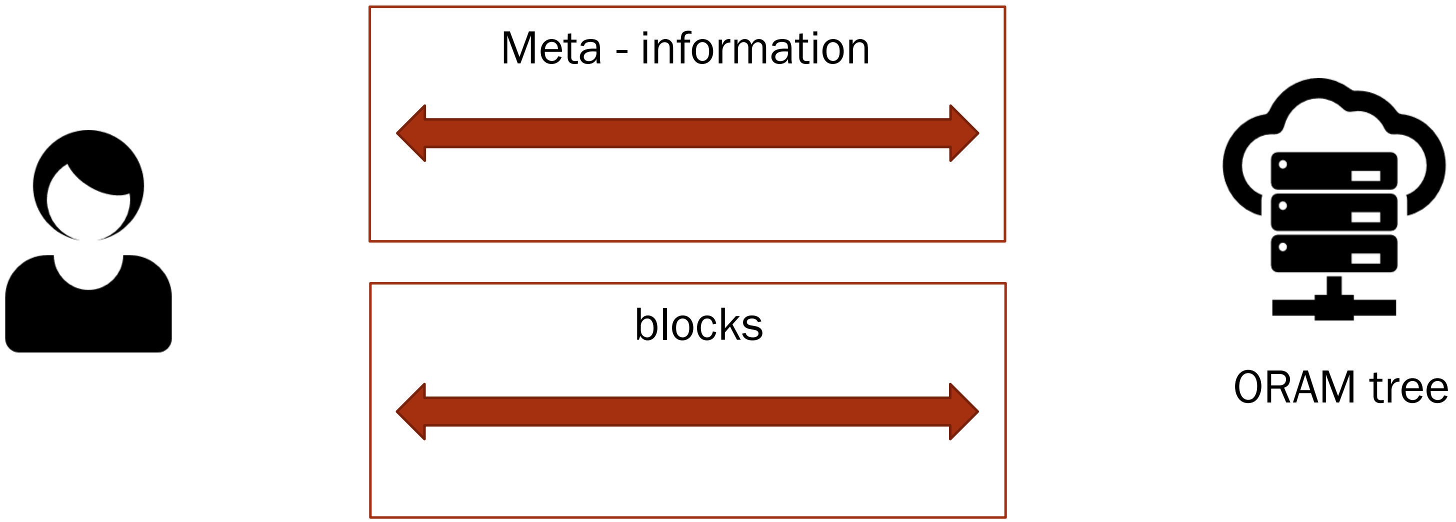
Part II

C-ORAM*: Constant Communication ORAM with homomorphic Encryption

Part III

CH^f-ORAM**: Constant Communication ORAM without homomorphic Encryption

What do we mean by “constant communication” ORAM?



We say that an ORAM is a constant communication ORAM if:

- Constant number of blocks
- Meta-information is dominated asymptotically by the size of constant number blocks

The server in this model is a computational server rather than a storage-only server

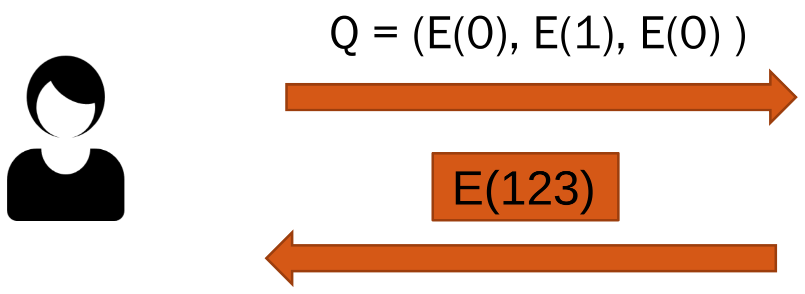
Why C-ORAM was needed?

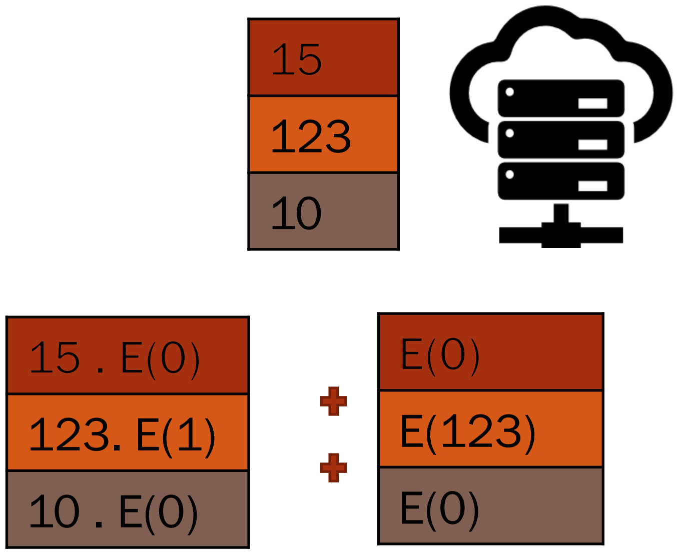
- Recent ORAM offers sublinear communication overhead
- Onion ORAM by Devadas et al. (TCC'16) first solution offering constant communication overhead, but
  - With a large block size and a high number of homomorphic multiplications
- Onion ORAM block size example:
  - For $N = 2^{20}$, the block size equals 33Mbit
  - Total data set size: 34 Tbit

Onion ORAM

High level

- Components and primitives:
  - Tree based ORAM
    - Additive homomorphic encryption such as Pailler or Damgard-Jurik
  - Private Information Retrieval (Kushilivitz et al.'97)

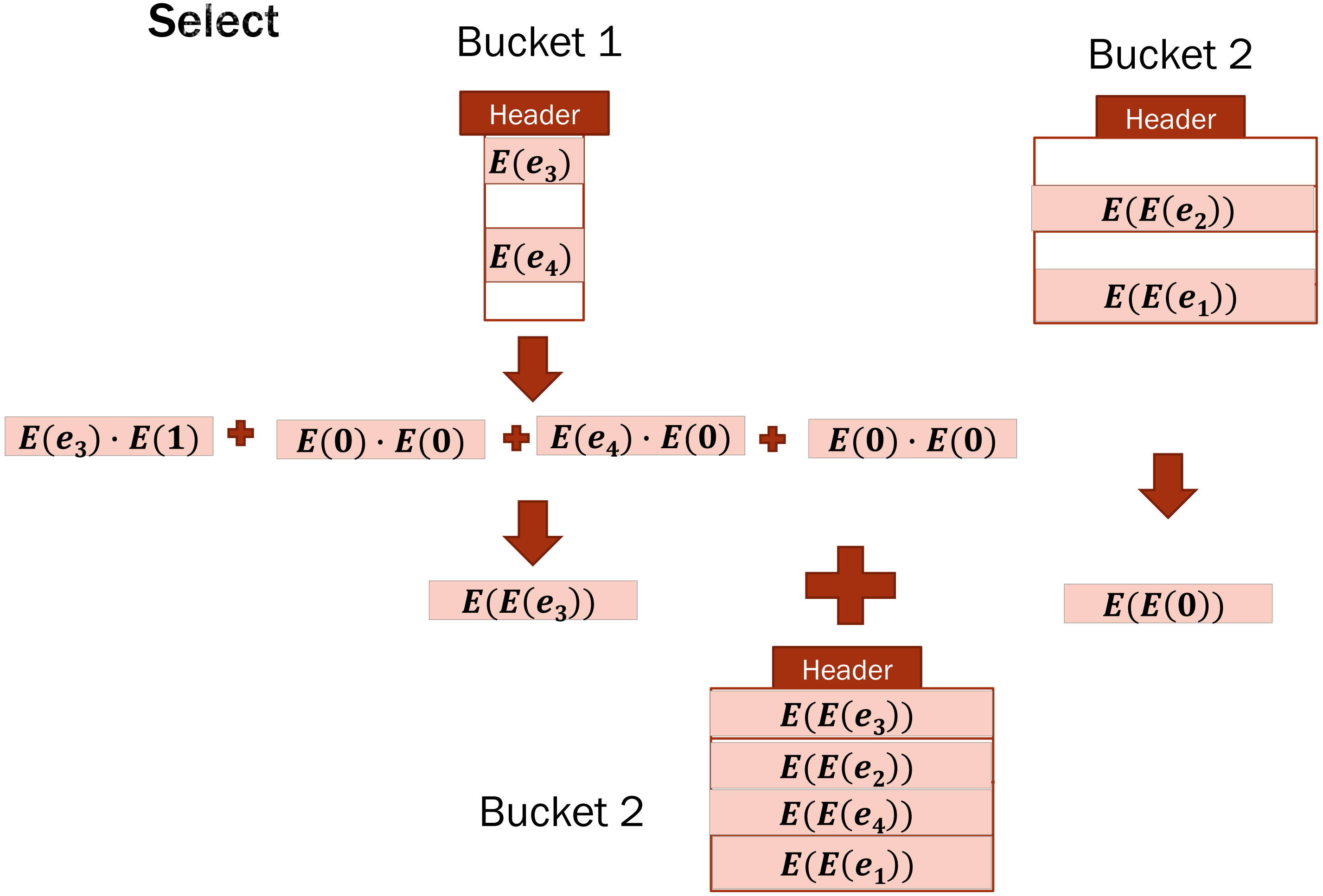


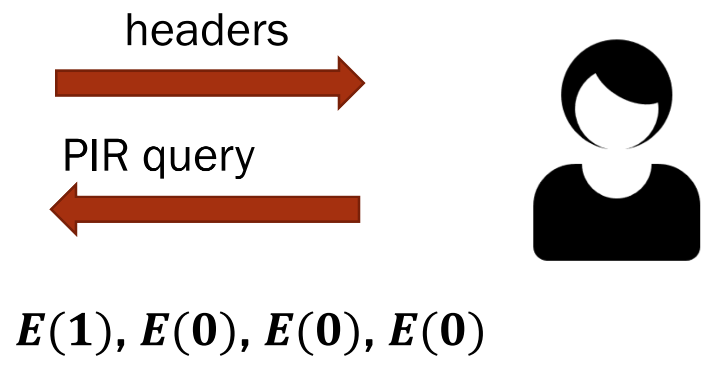


  - Select
    - Eviction without downloading the bucket

Onion ORAM

Select

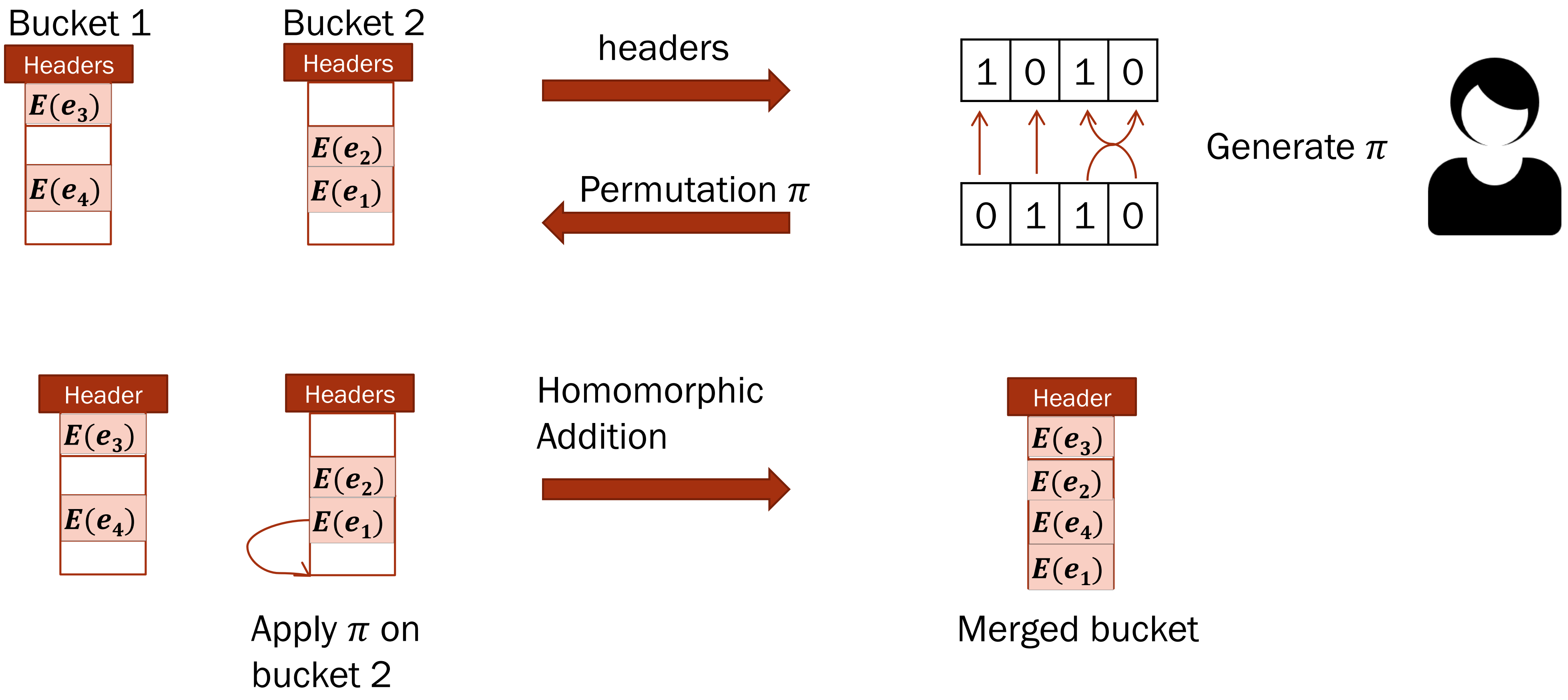




- Onion layers
- Select operation is the most expensive operation in Onion ORAM

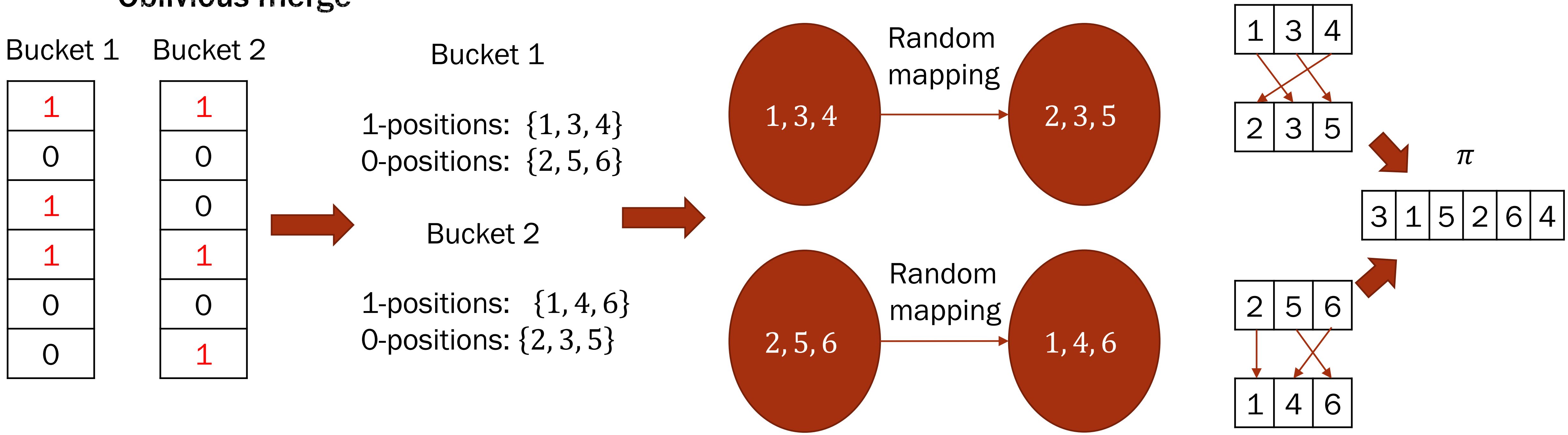
C-ORAM

Oblivious merge algorithm



C-ORAM

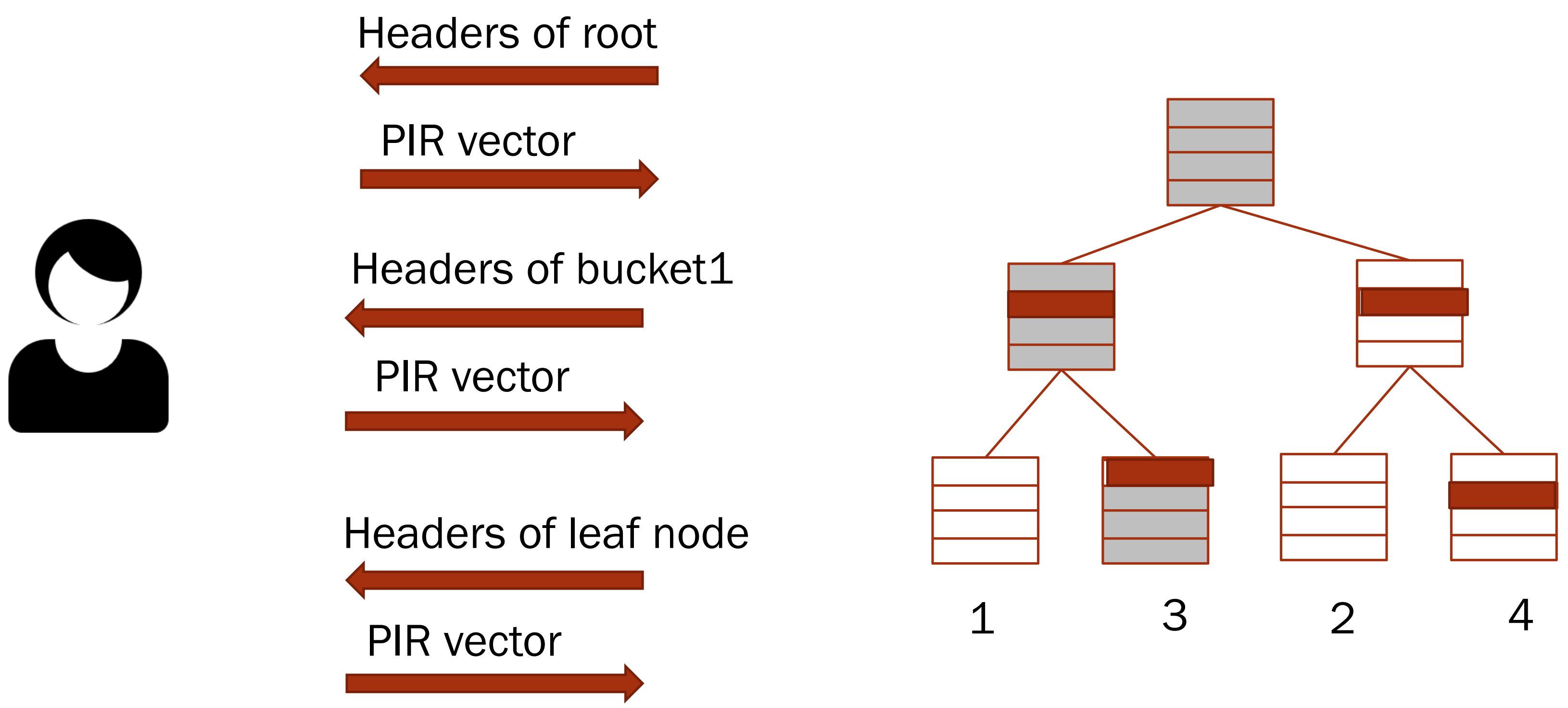
Oblivious merge



- Oblivious merge saves a $\log^2 N$ multiplicative factor over Onion ORAM's select permutation
- From $\log N$ PIR operation to 1 PIR operation
- Main challenges: Security and correctness

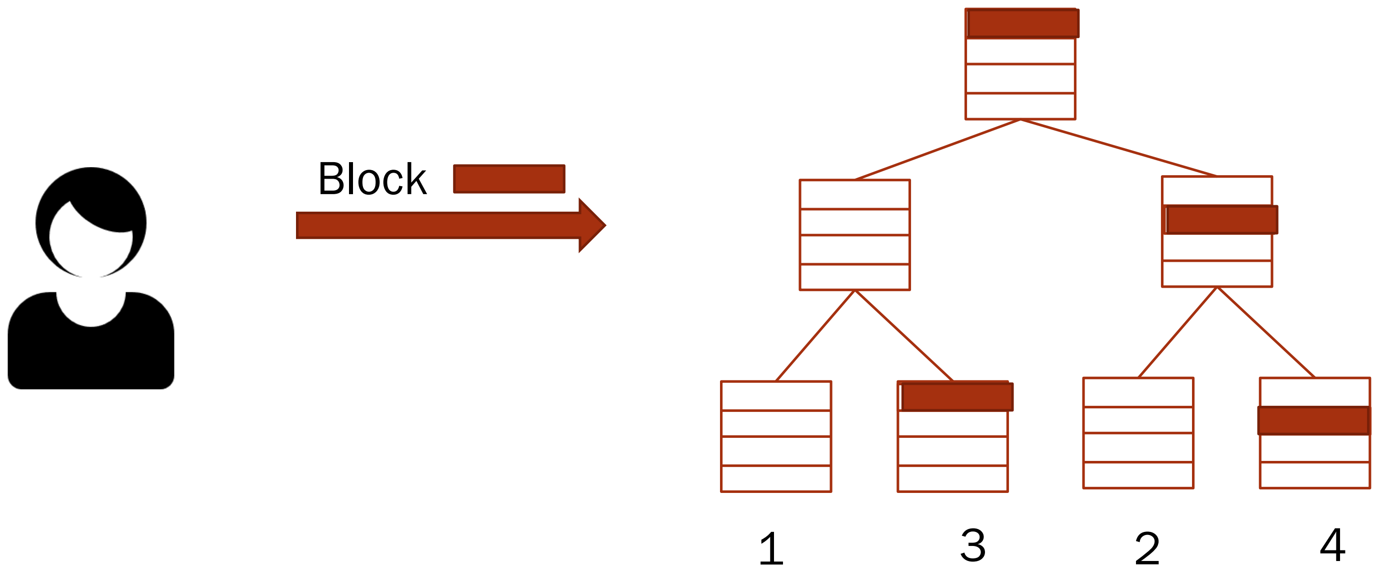
C-ORAM: Access

illustration



C-ORAM: Access

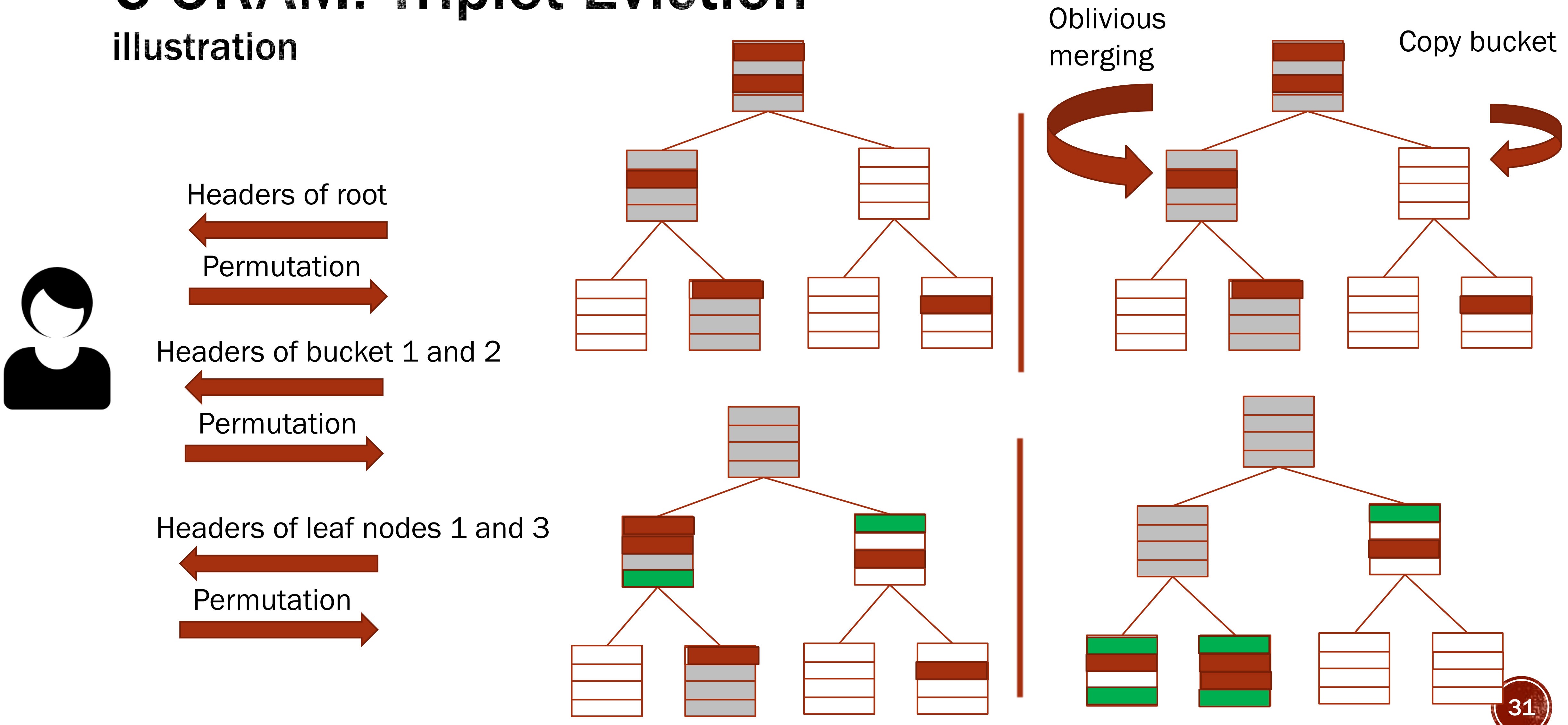
illustration



Adding the block to the root with **PIR-Write**

C-ORAM: Triplet-Eviction

illustration



Oblivious merging

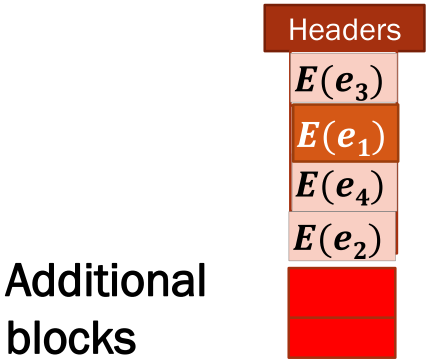
Security

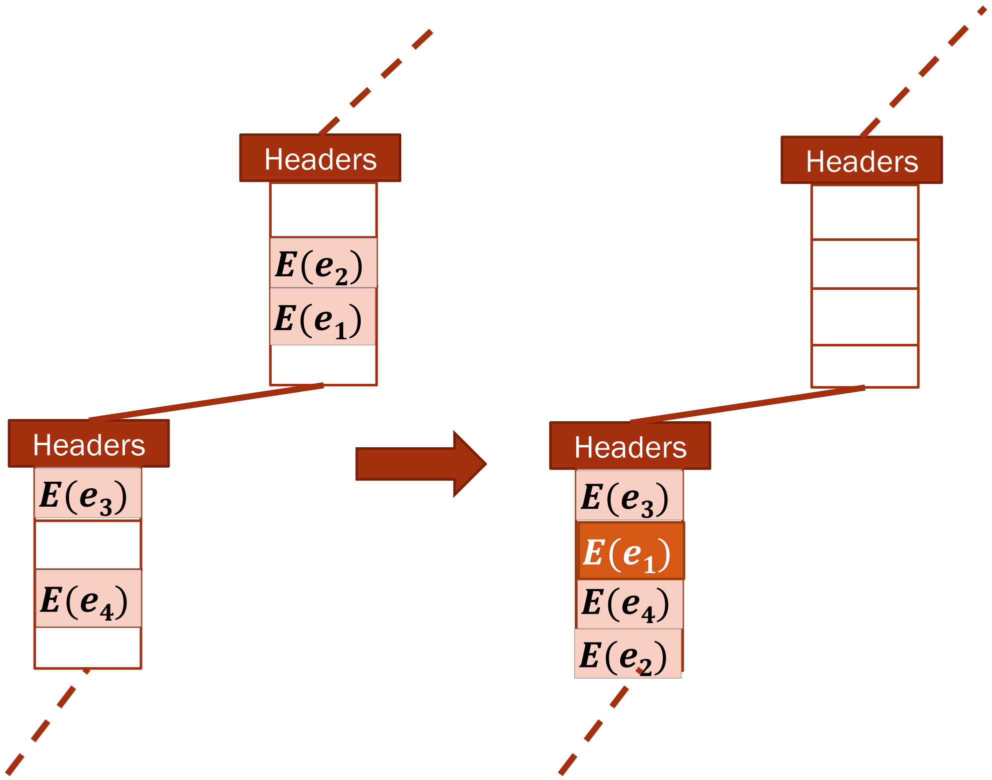
- Adversary, given π, does not get any additional knowledge over
  - load of a bucket
  - distribution of real, empty blocks
- Permutation outputted by oblivious merging is indistinguishable from a random permutation

C-ORAM

Correctness

- Noisy blocks
- Increasing bucket size by factor φ





- Oblivious merge fails if at a given level and eviction

#empty blocks of parent < #real blocks of child

#empty blocks of child < #real blocks of parent

φ is constant equal to 4 (empirically 2.2)

C-ORAM features

$$O(\underbrace{\log^4 N}_{\text{Meta-information: |PIR vectors| + |headers| + |Permutations|}} + B)$$

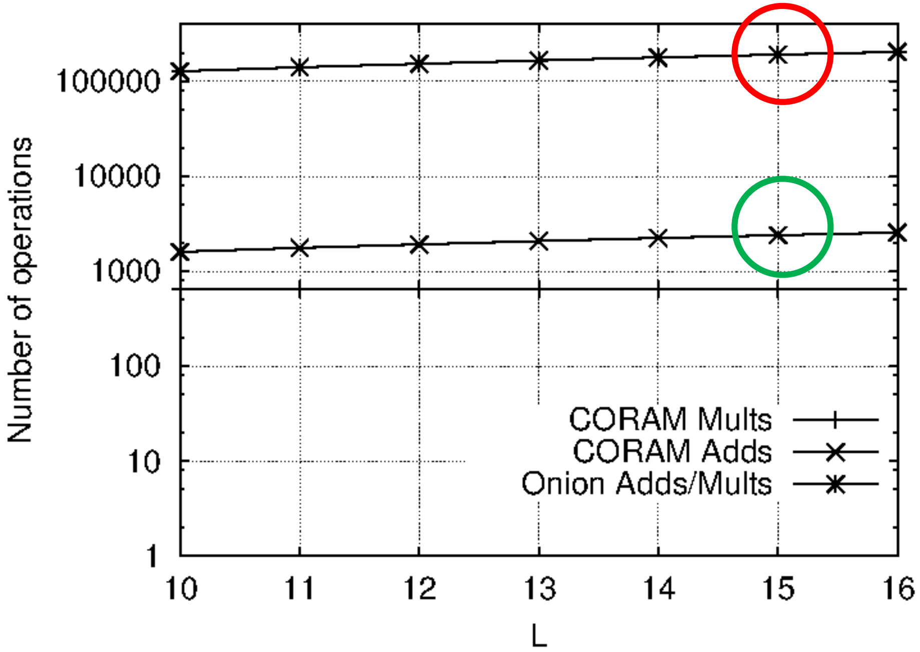
	Simplified block size	Homomorphic additions	Homomorphic scalar multiplications
Onion ORAM	$\Omega(\log^5 N)$	$\Theta(\log^8 N)$	$\Theta(\log^8 N)$
C-ORAM	$\Omega(\log^4 N)$	$\Theta(\log^6 N)$	$\Theta(\log^5 N)$

C-ORAM

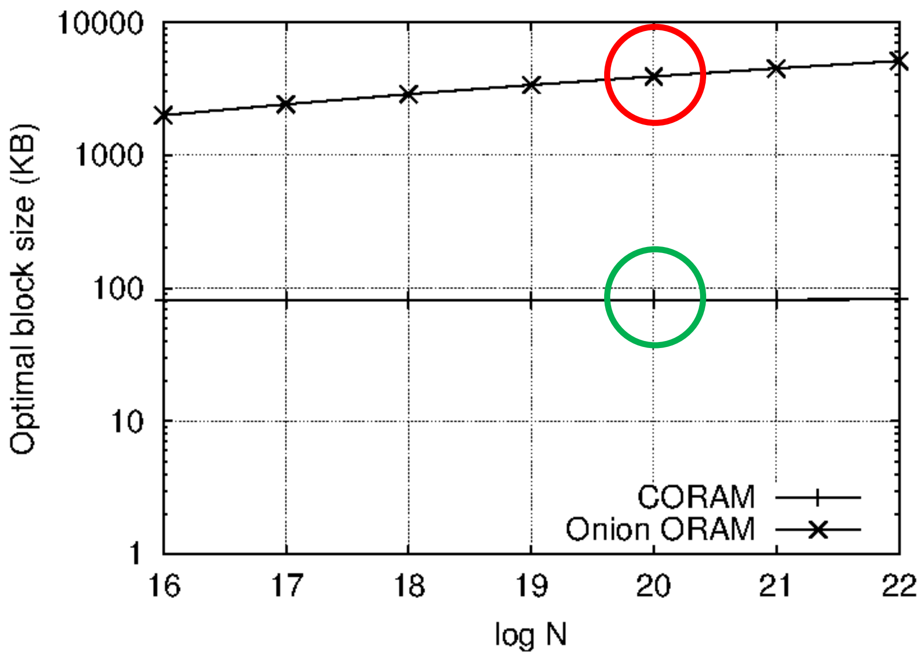
Evaluation

10 000 % fewer
homomorphic operations

4000 % smaller block
size for the same dataset



Computation



Storage

However C-ORAM still needs 5~10 minutes per access?

Part I

ORAM Overview

Part II

C-ORAM: Constant Communication ORAM with homomorphic Encryption

Part III

CH^f-ORAM: Constant Communication ORAM without homomorphic Encryption

CH^f-ORAM

Motivation

How can we get rid of the **very expensive** Homomorphic encryption?

CH^f-ORAM

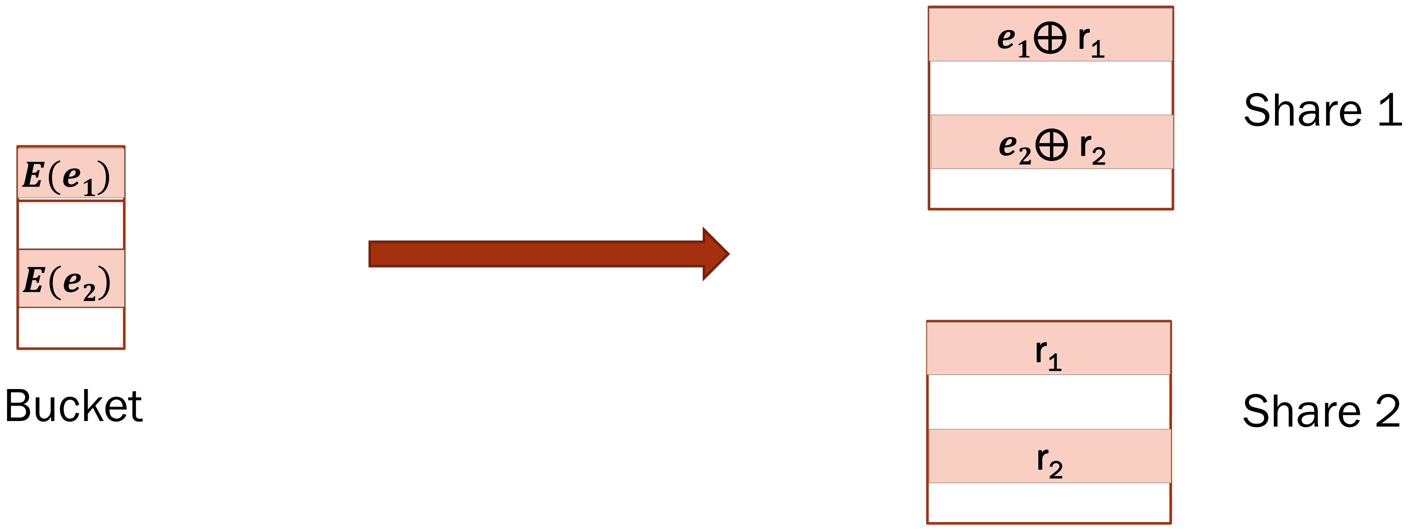
Intuition

1. Replace Homomorphic encryption with secret shared block
2. Replace computational PIR with Information-theoretic PIR

CH^f-ORAM

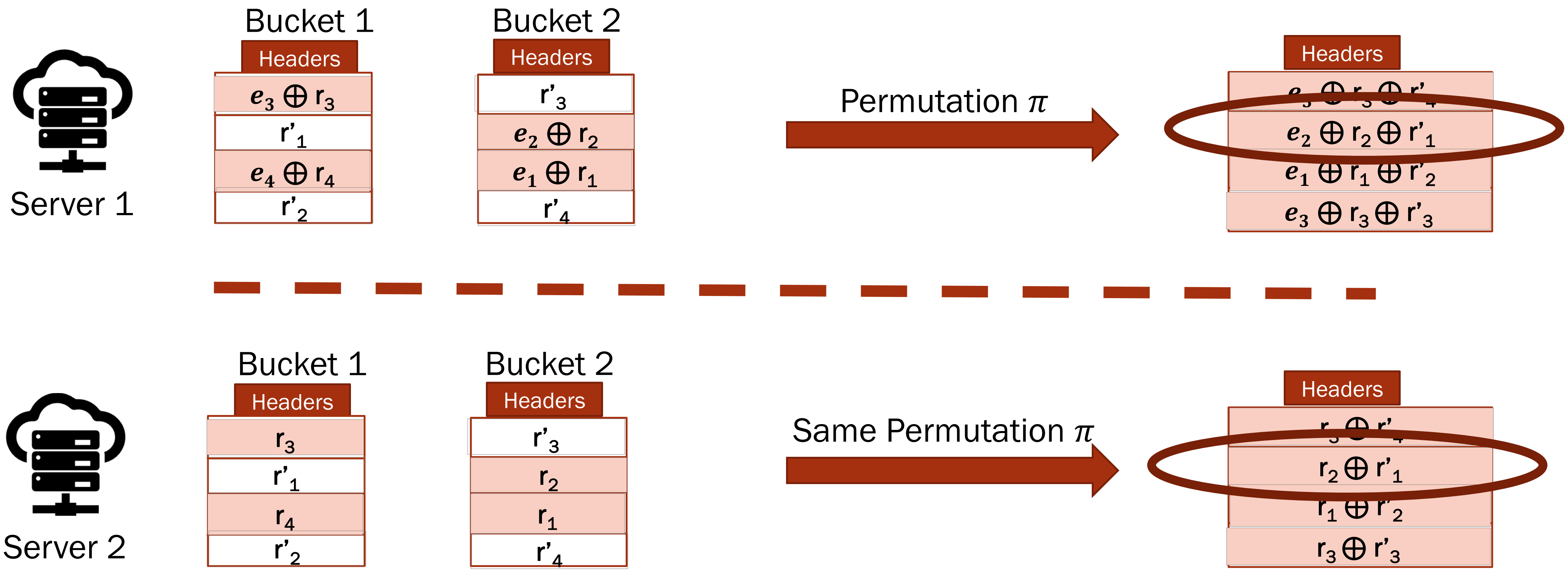
Multi-servers: 1st Step

- We use secret sharing and replace a homomorphically encrypted block by two shares:



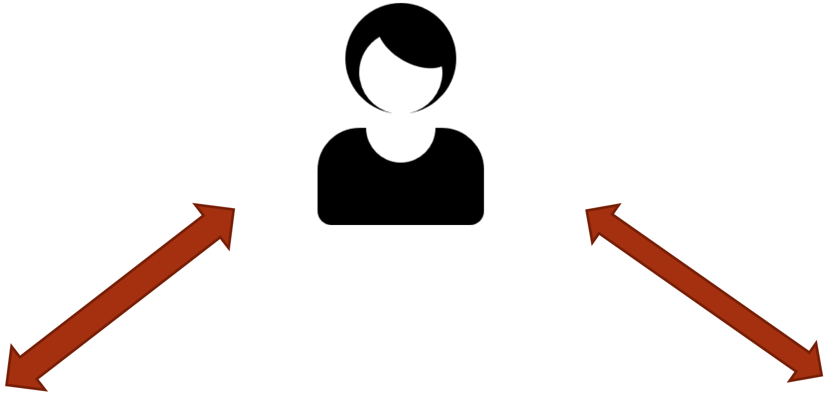
CH^f-ORAM

Oblivious merge algorithm



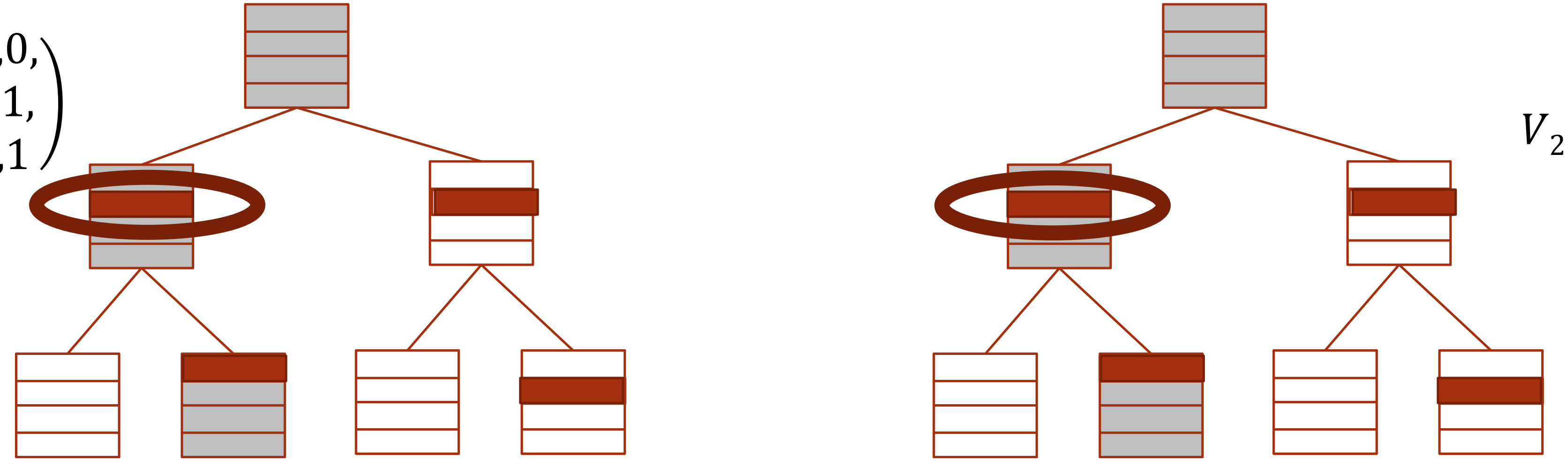
CH^f-ORAM

Read operation



Download all headers of the selected path

Determine the exact position of the block

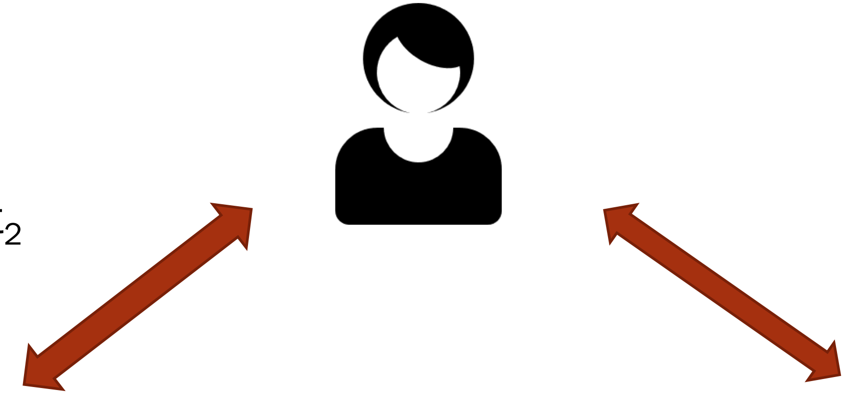


$$V_1 = \begin{pmatrix} 0,1,0,0, \\ 1,{\color{red}0},1,1, \\ 0,1,1,1 \end{pmatrix}$$

$$V_2 = \begin{pmatrix} 0,1,0,0, \\ 1,{\color{red}1},1,1, \\ 0,1,1,1 \end{pmatrix}$$

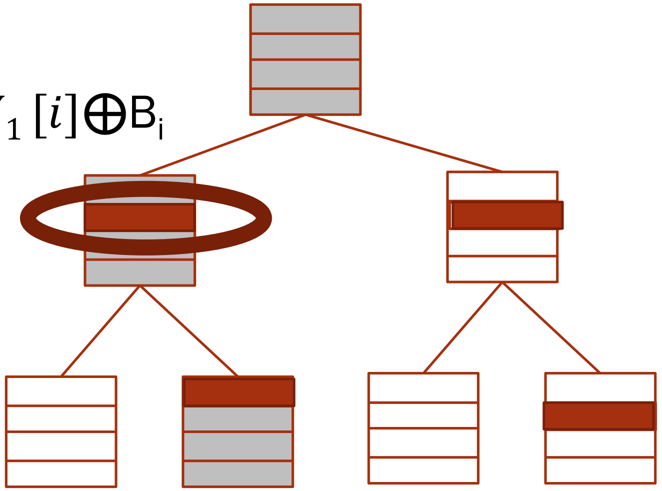
CH^f-ORAM

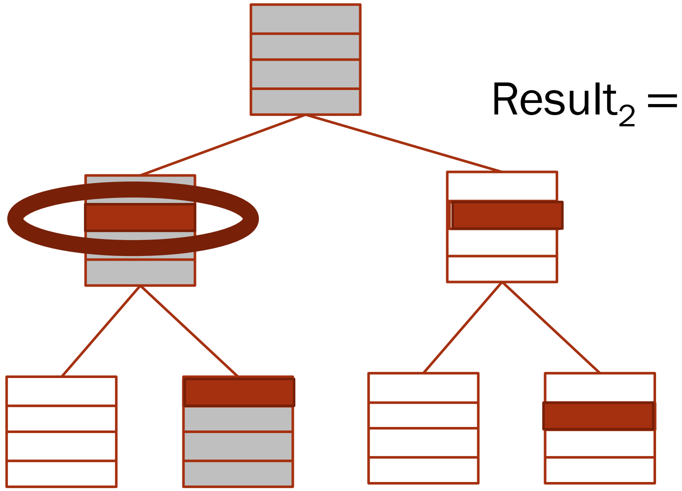
Read operation



Compute $\text{Result}_1 \oplus \text{Result}_2$

$\text{Result}_1 = \sum_{i=1}^{\log N} V_1[i] \oplus \text{B}_i$





$\text{Result}_2 = \sum_{i=1}^{\log N} V_2[i] \oplus \text{B}_i$

CH^{f}-ORAM

Multi-servers: 2nd Step

For any constant $\#\boldsymbol{Server} \geq \mathbf{2}$ and for any $\boldsymbol{B} \geq \boldsymbol{k} \cdot \boldsymbol{N}$, there exists an IT-PIR construction with communication complexity O(B) bit.

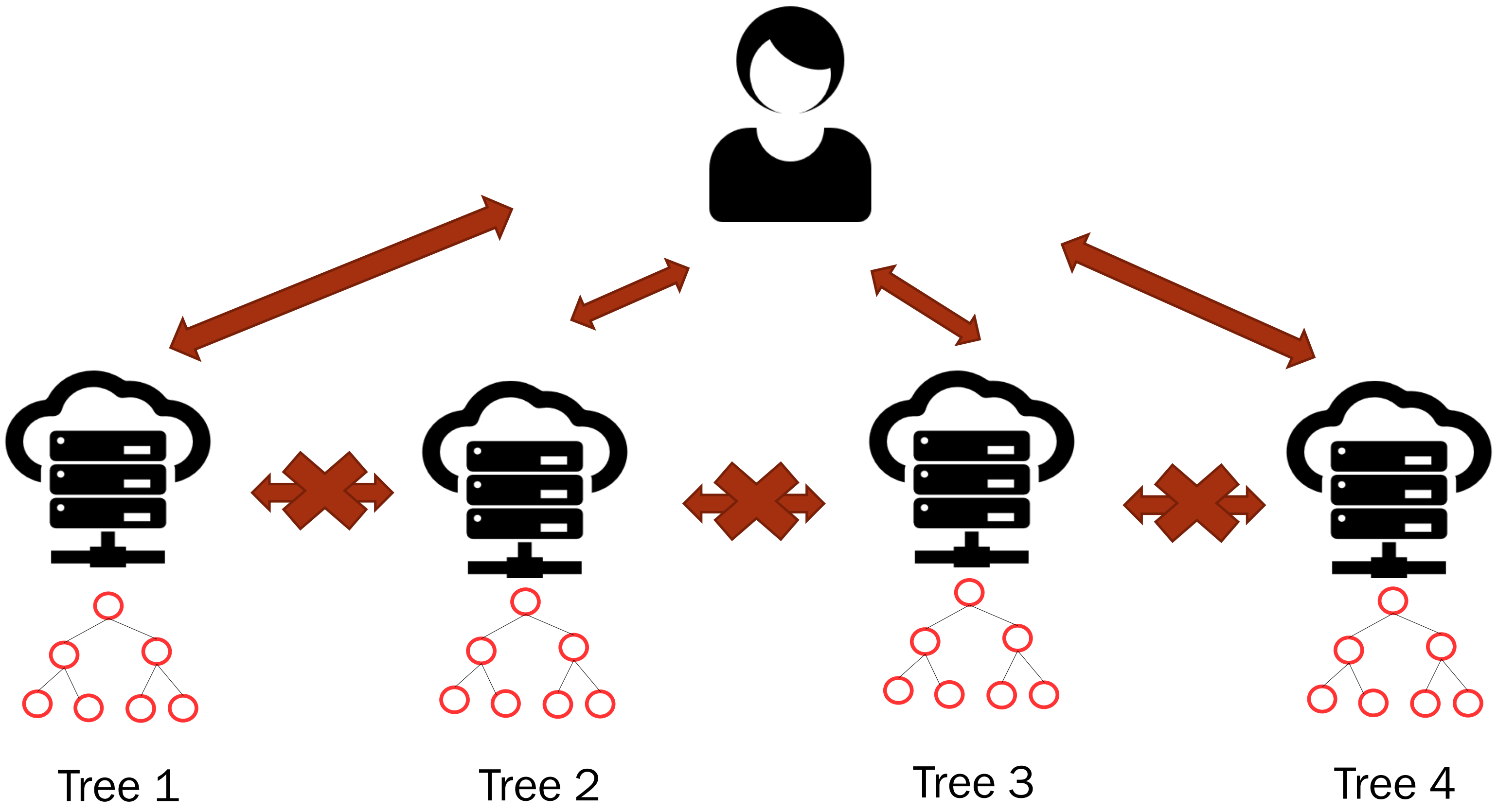
- Replace C-PIR with IT-PIR while taking advantage of the obliviousness of tree-based ORAM

For any constant $\#\boldsymbol{Server} \geq \mathbf{2}$ and for any $\boldsymbol{B} \geq \boldsymbol{k} \cdot \boldsymbol{\log N}$, there exists an IT-PIR construction with communication complexity O(B) bit.

CH^f-ORAM

Multi-servers

- Tree 1 and Tree 2 are secret shared (block per block)
- Tree 3 is a replica of Tree 1
- Tree 4 is a replica of Tree 2



CHf-ORAM

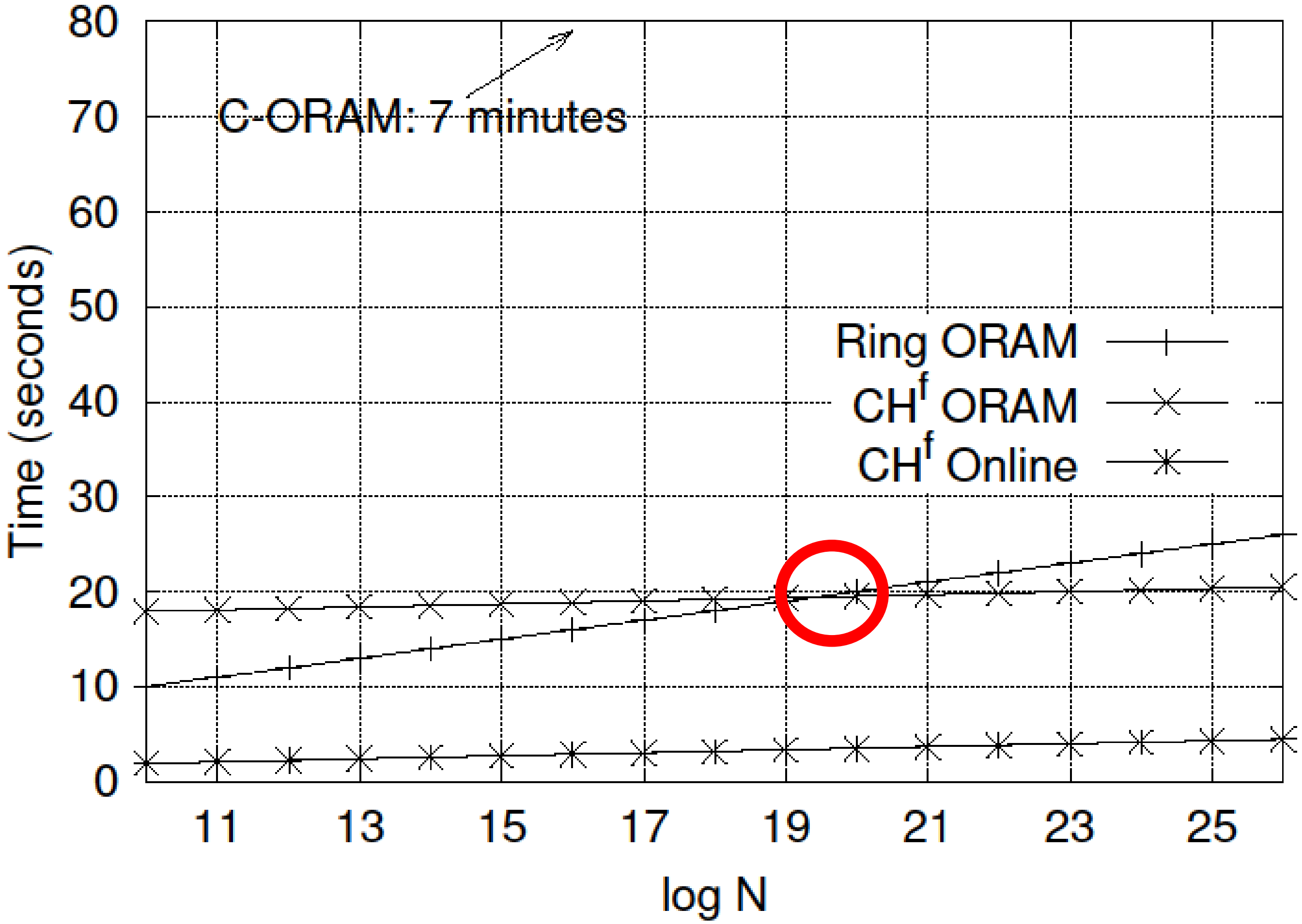
Gain over C-ORAM

C-ORAM		CH^f-ORAM
$O(\log^2 N)$ homomorphic multiplications	→	$O(\log N)$ XOR operations
$O(\log N)$ C-PIR query generation	→	$O(\log N)$ Random bit generations
Encrypt the block homomorphically	→	Secret share the block
Computational HA	→	IT-secure

CHf-ORAM is as good as PIS in communication enjoying a polylog in computation (rather than linear)

CH^f-ORAM

Evaluation



1. block size of 1 MB.
2. network speed of 20 Mbps.
3. XOR of two 1 MB blocks in 1 ms (2012 Macbook Pro with 2.4 Ghz Intel i7)

CH^f-ORAM

Eviction Circuit

- In SCORAM, eviction circuit size in tree-based ORAM is a bottleneck for secure RAM computation
- Best ORAM for secure RAM computation are those with constant private storage
- Tree-based ORAM with stash are not good for secure RAM computation due to the oblivious sorting

CHf-ORAM has **constant circuit size, with constant private storage with no need for OS**

CH^f-ORAM

Eviction Circuit (more details)

Scheme	Circuit Size
SCSL'11	$O(\log^4 N + B \cdot \log^2 N)$
CLP'14	$O(\log^4 N + B \cdot \log^2 N)$
Path SC ORAM	$O(\log\log \mathrm{N}\ (\log^3 N + B \cdot \log N))$
LO'13	$O(\log N\ \cdot C_{PRF} + B \cdot \log N)$
Circuit ORAM	$O(\log^3 N + B \cdot \log N)$
CH^f-ORAM	$O(\log^4 N + B)$

If B is larger than $\log^4 N$, then circuit size is constant in B

Conclusion

	Simplified block size in bits	Private Storage in block	Communication in block	Homomorphic additions	Homomorphic scalar multiplications	#Servers
C-ORAM	$\Omega(\log^4 N)$	$O(1)$	$O(1)$	$\Theta(\log^6 N)$	$\Theta(\log^5 N)$	1
CH^f-ORAM	$\Omega(\log^3 N)$	$O(1)$	$O(1)$	–	–	4

To do

- We have:
  - Constant Communication ORAM
  - Constant number of rounds
  - Very small Block Size
  - No Computation on the server Size
  - Constant Private Storage
  - One-server

Reduce the block size to be in $O(\log^2 N)$

(No heavy computation)

To do

	Simplified block size in bits	Private Storage in block	Communication in block	Homomorphic additions	Homomorphic scalar multiplications	#Servers
C-ORAM	$\Omega(\log^4 N)$	$O(1)$	$O(1)$	$\Theta(\log^6 N)$	$\Theta(\log^5 N)$	1
CH^f-ORAM	$\Omega(\log^3 N)$	$O(1)$	$O(1)$	–	–	4
	$\Omega(\log N)$ or $\Omega(\log^2 N)$	$O(1)$	$O(1)$	–	–	1

Picture from http://www.deviantart.com/browse/all/fanart/?q=super-sheep&order=9

Thanks!