Lecture 7: Oblivious RAM

MIT - 6.893
Fall 2020
Henry Corrigan-Gibbs

Plan

Breakout rooms
→ Applications of PIR
→ Gaps b/w theory + practice?

Recap: DPF construction

Oblivious RAM: Defn

Stretch Break

"Square-root construction"

Logistics

* HW 2 due Friday at 5pm via Gradescope
* OH Today
* Look for Piazza poll

Recap: DPF from PRG

Important things to remember

1. DPF = succinct secret sharing of a (possibly exponentially long) vector of all zeros w/ a single 1.

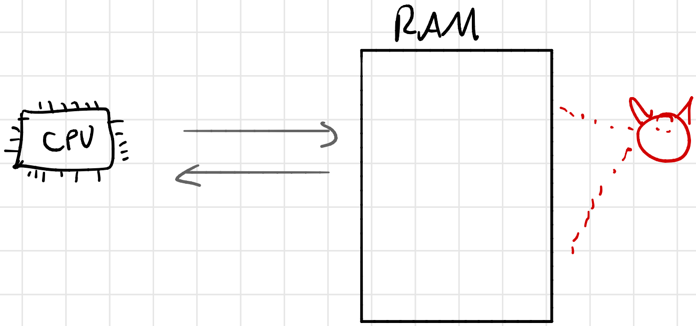
2. Simple + clever construction from PRG in two party setting.

   Key size $O(\lambda \cdot n)$ on sec param λ, vector length 2^n

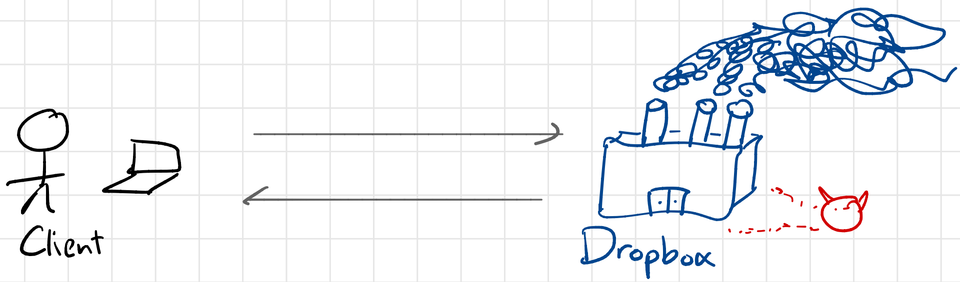
3. Many applications: PIR, private statistics, MPC, ...

Motivation

Hardware enclave



Store your files on Dropbox without leaking your file contents or access patterns to Dropbox.



In both cases, encryption can hide contents of RAM/storage, but access pattern leaks.

→ This is enough to leak all sorts of sensitive information.

Examples

Dropbox: Sizes of files leak type/content.

↳ Server can learn what programs you're running and when.

Enclave: for each patient w/ condition X {

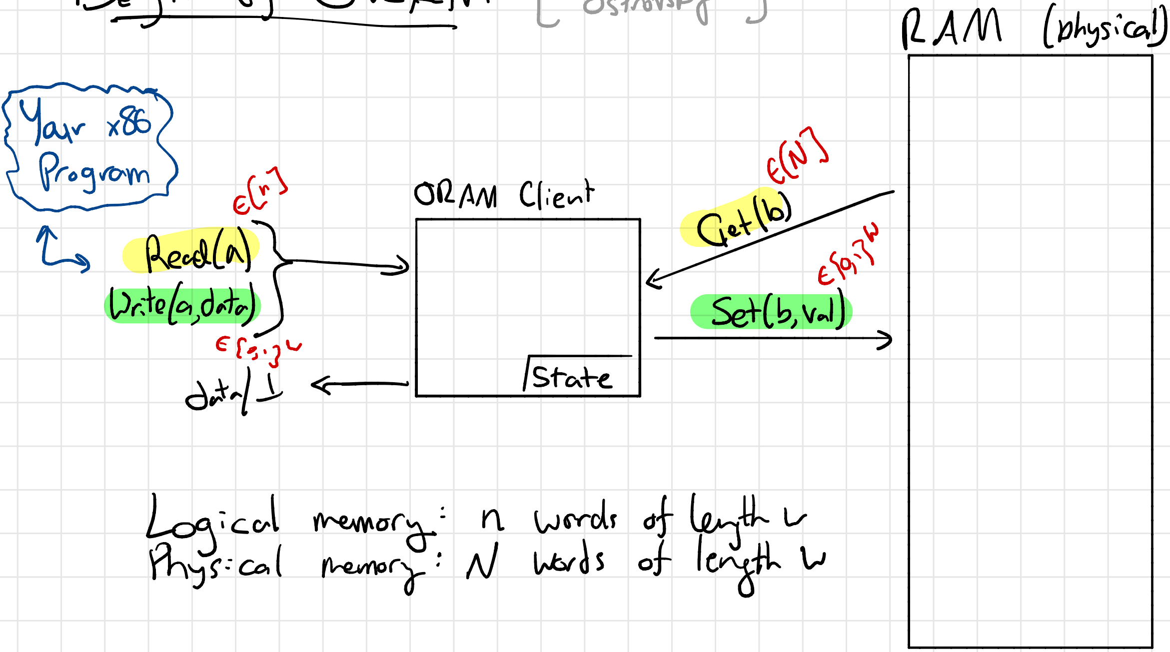
- look up patient's phone #
- add to output

}

→ Notice that # of accesses here can leak info too!

id_1	$phone_1$
id_2	$phone_2$
⋮	⋮

Defn of ORAM [Goldreich & Ostrovsky]



Logical memory: n words of length w
Physical memory: N words of length w

For $op \in \{\text{Read}(\cdot), \text{Write}(\cdot,\cdot)\}$, let $A(op)$ be the physical addresses that the ORAM client probes when servicing op.

ORAM Properties

1. Correctness. For every sequence $\mathcal{O} = \{op_1, op_2, \dots\}$, where each op is a R/W, client (talking to honest RAM) answers each op correctly. ... maybe up to correctness error.

2. Security. For any two poly-size op seqs of = length:

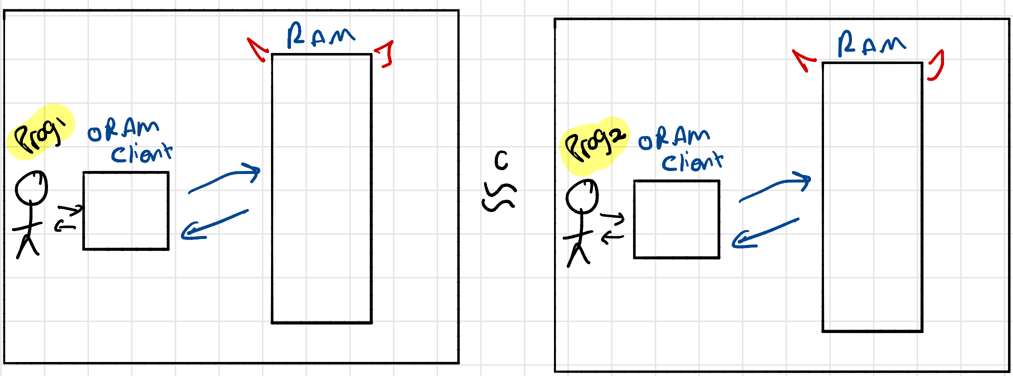
$$\mathcal{O} : (op_1, \dots, op_m)$$

$$\mathcal{O}' = (op_1', \dots, op_m')$$

it holds that

$$\{A(op_1), \dots, A(op_m)\} \approx_c \{A(op_1), \dots, A(op_m)\}.$$

$\Rightarrow$ ORAM leaks # of accesses to RAM.
(as sem. sec encryption leaks msg length)



Stretch Break

Sanity Checks (see Elaine Shi's notes on ORAM)

Simple solutions

- ORAM client stores all n words in its internal state... doesn't use RAM at all
  - ↳ n words of storage "stash"
  - ↳ 0 RAM accesses per op
- ORAM Client reads entire RAM on every op (uses encryption to hide contents)
  - ↳ $O(1)$ words of storage (e.g. AES key)
  - ↳ n RAM accesses per op.

Goal: Small storage + few RAM accesses per op.

("online")

Best possible: $O(\log n)$ RAM accesses per op., even if client stores n^{ε} bits for $\varepsilon > 0$ [Larsen & Nielsen 2018]

* Achieved by scheme we will see next class ("Path ORAM") w/ some restrictions.

* Achieved by a very subtle scheme this year 2020 ("OptORAMa") w/o restrictions.

ORAM vs PIR

Both primitives involve hiding client's access pattern from a potentially adversarial server...

Good to understand how they differ.

ORAM	PIR
- Memory contents on server changes w/ each query	- DB is public, static
- One client <-> one server	- Many clients talk to same server DB
- Supports reads & writes	- Supports only private reads
- Server can process R/W ops in poly log(n) time on memory of n words	- Linear server work per query *
- Can build from PRFs.	- In single-server setting, requires pub. key crypto
"Private access to private data"	"Private reads to public data"

The "Square-Root ORAM" (Goldreich & Ostrovsky '92)

Simple + clean. We will see a more efficient construction next class.

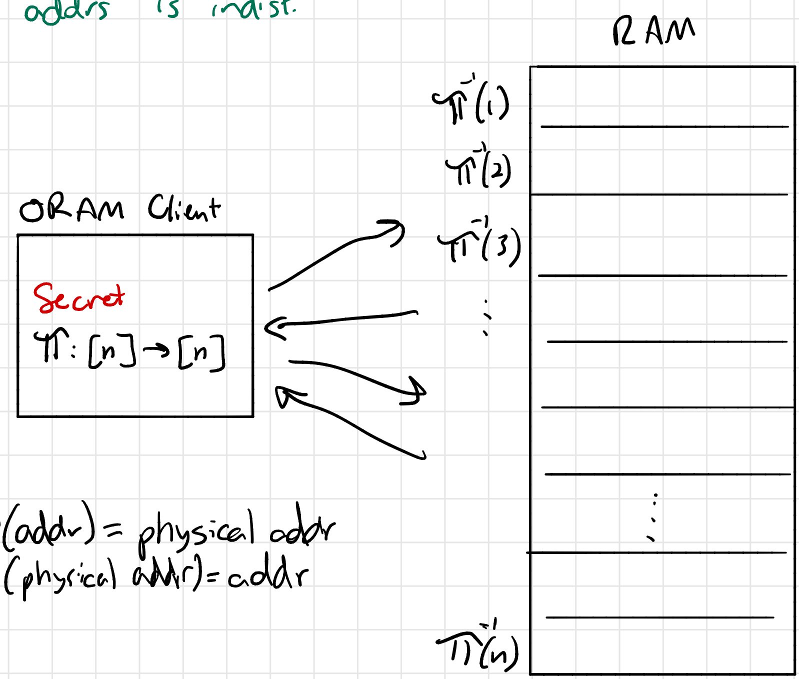
Client storage: $O(1)$ words (PRF key)

Server storage: $n + O(\sqrt{n})$ words

$O(\sqrt{n})$ RAM accesses per op., amortized

Key idea: Suppose RAM holds logical mem contents permuted according to some π that only client knows.

$\Rightarrow$ "Read-once ORAM"... any sequence of ops to distinct addrs is indist.



$\pi(\text{addr}) = \text{physical addr}$

$\pi^{-1}(\text{physical addr}) = \text{addr}$

Construction

* Initialize memory contents with encryption of O's (using sem sec enc scheme)
  n data blocks, $\sqrt{n}$ dummy blocks, $\sqrt{n}$ stash blocks

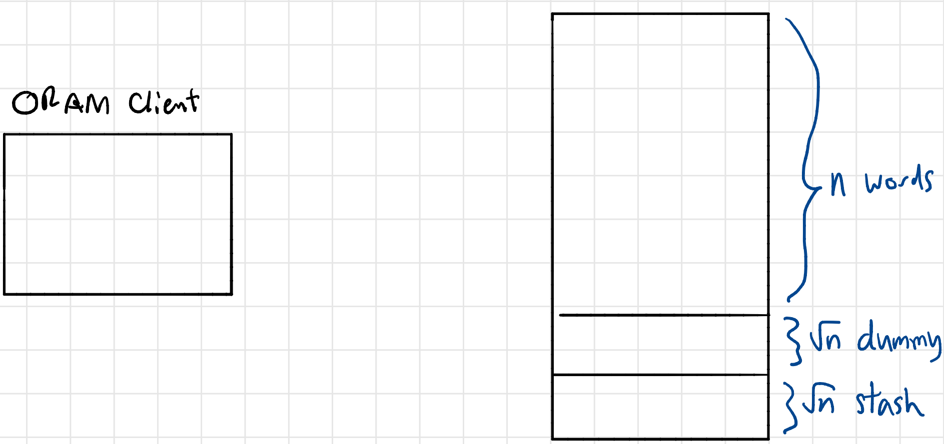
* While true:

  1. Shuffle $n+\sqrt{n}$ data + dummy blocks using fresh random perm $\pi : [n+\sqrt{n}] \to [n+\sqrt{n}]$.

  2. Process $\sqrt{n}$ ops:

     - Read + write back entire stash
     - If desired element is in stash
       → read one dummy block
     - Else
       → read data block
     - Read + write back entire stash

  3. Return all words to their starting location.



Details

Step 1: Sorting.

* Use PRP to assing tag $\pi(i)$ to addr i.
* Run a sorting network to sort by tags in $O(n \log^2 n)$ RAM access

Step 2: Access

* Read stash: $\sqrt{n}$ RAM accesses
* Read data/dummy elem: 1 "
* Read stash: $\sqrt{n}$ "

Step 3: Unsort, again using Batcher

$O(n \log^2 n)$ RAM accesses

Total cost: $O(n \log^2 n)$ RAM accesses

per $\sqrt{n}$ logical ops

$\Longrightarrow$ Amortized $O(\sqrt{n} \log^2 n)$ cost per access.

So, we saw that ORAM is possible w/ sym-key tools w/ $O(\sqrt{n}\log^2 n)$ overhead per access.

Next time: a more efficient scheme.