Lecture 5: Offline/Online PIR

MIT - 6.893
Fall 2020
Henry Corrigan-Gibbs

Plan

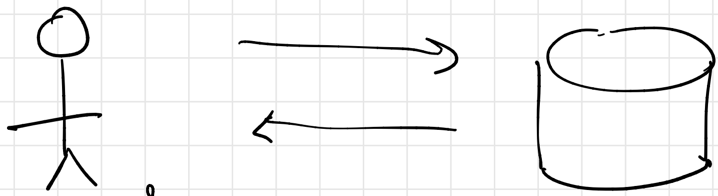
- Recap: Batch PIR
  (+ PIR by keywords)
- Computation in PIR
- Stretch break
- Offline/Online PIR

Logistics

- Last chance for HW1: today 5pm
- HW2 Posted
  ↳ Start early!
  I am not joking!
  Due: 10/2 5pm via Gradescope
- Breakout sessions: will aim for 1/week
- Guest lecture on W... please look at reading & bring one Q.
- Computer issues...

Recap:

PIR by Keywords.



$$DB = \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} \quad s_1 \in \{0,1\}^\ell$$

$\sigma \in \{0,1\}^\ell$

$$\downarrow$$

1 if $\sigma \in DB$

0 o.w.

Bottom Line: Cost of PIR by keywords $\approx$ Cost of normal PIR

Approach: Hash twice...

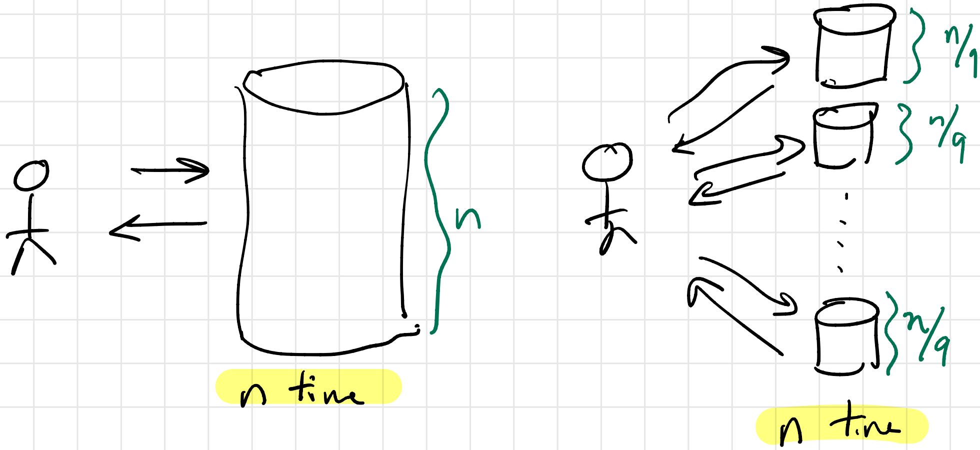
- Once into buckets (few collisions)
- Once within buckets (no collisions.)

Recap: Batch PIR

to same DB

Idea: Answer q queries at server-side cost of answering 1 query.

Observation:



1 query to DB of size n

q queries to DB of size n/q.

Strategy: If client wants to make q queries, partition DB into q chunks at random.

As long as clients' desired bits fall in diff chunks ⟹ can recover.

⟹ A little more work gets correctness w/ all but negl prob.

Server-Side Computation in PIR

BIM'04: If PIR servers store DB in original form, servers must probe every DB bit in responding to client's query.

Intuition: Servers don't touch bit i^*
$\Rightarrow$ Client is probably not reading bit i^*.

Takes some work to make precise.

How to get around?

1. Batching: Amortize cost of linear scan over many queries.

2. Preprocessing: Server does linear scan in a preprocessing phase.

↳ Per client (today) — offline/online
↳ Per database (HW) — "PIR w/ Preprocessing"

$\Rightarrow$ Active area of research. Lots to do still... $\Leftarrow$

Offline-Online PIR (with D. Kogan)

↳ Will discuss two-server setting... also makes sense in single-server setting

Idea: Push heavy work to an offline phase... takes place before client even knows which DB element it wants

→ Push heavy linear scan to a more convenient time (out of critical path)

→ Servers can still store DB in unmodified form

↳ Other approaches that you'll see on HW blow up server storage

n-bit DB ⟶ n^3-bit preprocessed data structure

Nice open Qs here on whether it's possible to avoid these blow ups

<u>Notation</u>: $\tilde{O}(f(n)) = O(f(n) \cdot \text{polylog}(n))$

Stretch Break

Will present a two-server scheme with computational security (PRG)

comm $\tilde{O}(\sqrt{n})$
online time $\tilde{O}(\sqrt{n})$

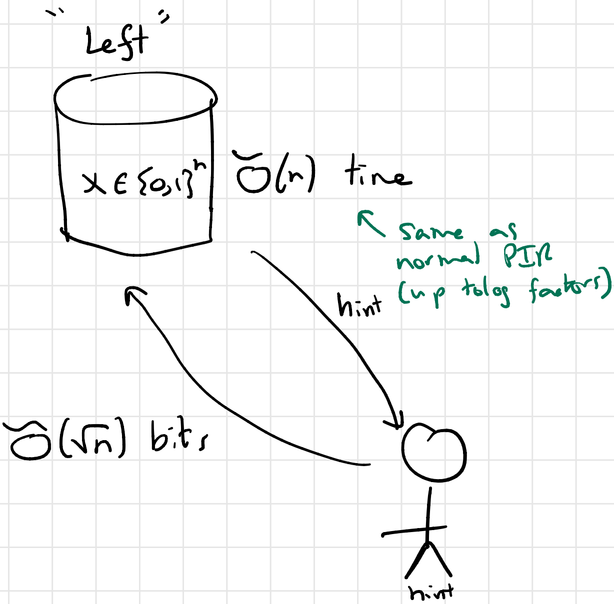
$$\boxed{\begin{array}{l}\text{We show that for PIR schemes in which servers store DB in unmodified form} \\ \quad \text{comm. } S \\ \quad \text{online time } T \quad \Rightarrow \quad S \cdot T \geq \tilde{\Omega}(n).\end{array}}$$

$\Rightarrow$ This $\sqrt{n}$ scheme is optimal w.r.t. these parameters.

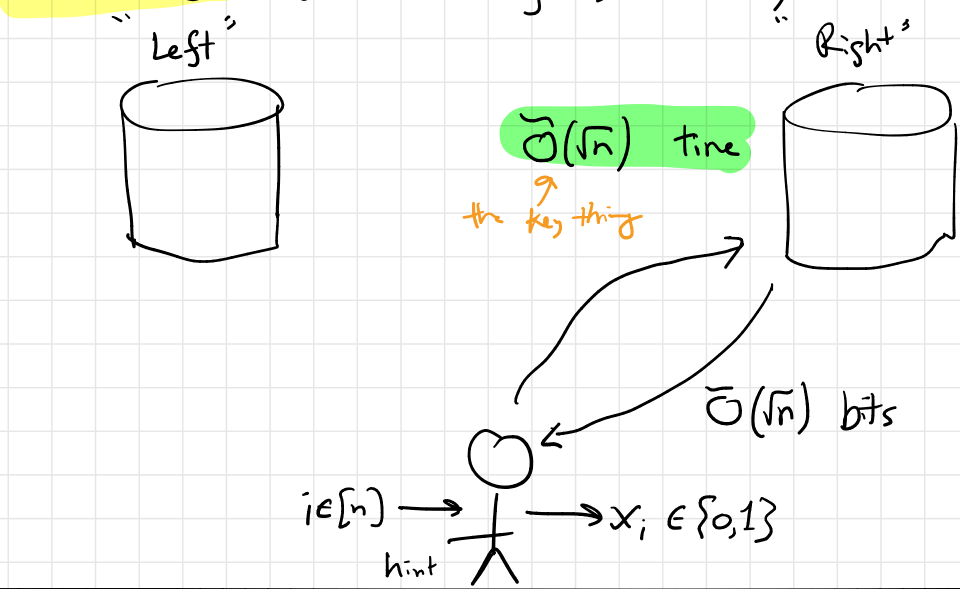
$$\left.\begin{array}{l}\text{* Can get info-theoretic security.} \\ \text{* Can get perfect correctness} \\ \text{* Can reduce client running time.} \\ \text{* Can reduce online communication.}\end{array}\right\} \text{See paper for many of these extensions.}$$

⋮

OFFLINE (think: overnight batch job)

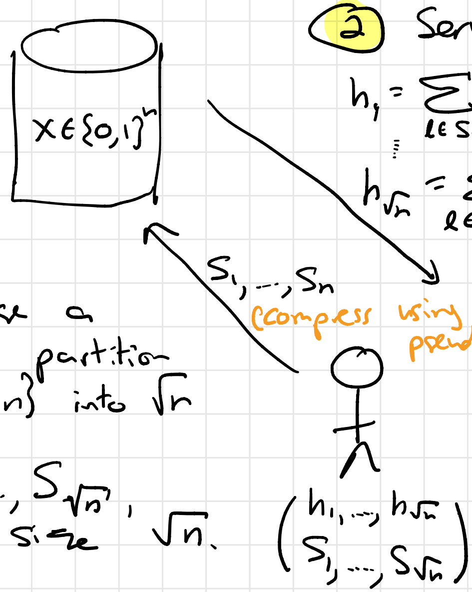


ONLINE (think: when you browse web)



PIR Security & correctness properties are as before.

Construction: Offline Phase



(compress using pseudorandomness)

1. Choose a random partition of $\{1, \ldots, n\}$ into $\sqrt{n}$ sets $S_1, \ldots, S_{\sqrt{n}}$, each of size $\sqrt{n}$.

2. Server sends parity of DB bits in each set

$$h_1 = \sum_{\ell \in S_1} x_\ell \mod 2$$

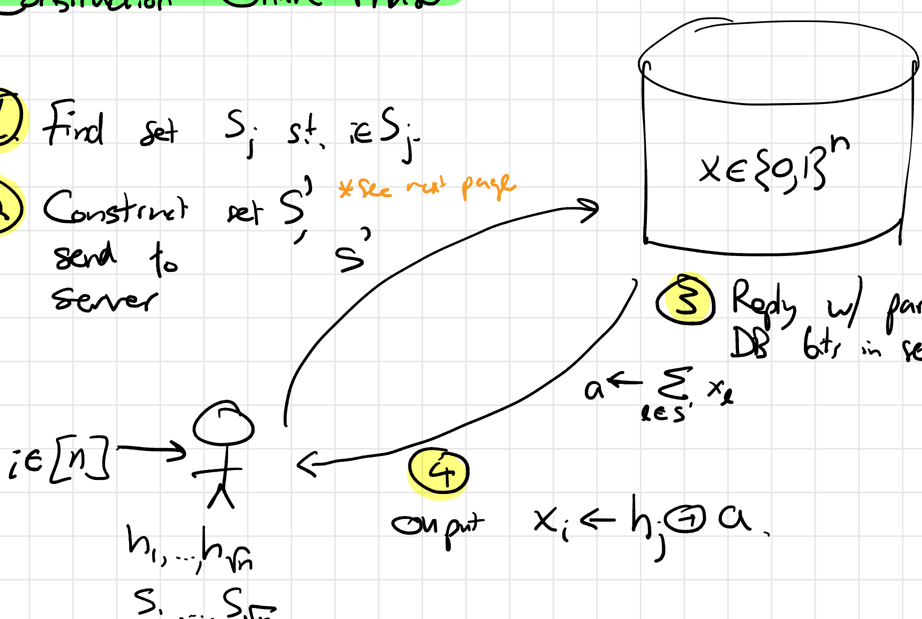
$$\vdots$$

$$h_{\sqrt{n}} = \sum_{\ell \in S_{\sqrt{n}}} x_\ell \mod 2$$

$$\begin{pmatrix} h_1, \ldots, h_{\sqrt{n}} \\ S_1, \ldots, S_{\sqrt{n}} \end{pmatrix}$$

3. Client stores h's and S's as hint.

Construction: Online Phase



1. Find set S_j s.t. $i \in S_j$.
2. Construct set S', send to server *see next page
3. Reply w/ parity of DB bits in set.

$$a \leftarrow \sum_{\ell \in S'} x_\ell$$

4. output $x_i \leftarrow h_j \oplus a$.

How client constructs S'...

Flip a coin w/ $p_{heads} = \frac{\sqrt{n}-1}{n}$ $\left[\approx \frac{1}{\sqrt{n}}\right]$

↳ If heads, choose random $i' \xleftarrow{R} S_j \setminus \{i\}$
Set $S' \leftarrow S_j \setminus \{i'\}$
Output "fail".

↳ If tails, set $S' \leftarrow S_j \setminus \{i\}$.

Why this works...

Correctness: with probability $1 - p_{heads} \approx 1 - \frac{1}{\sqrt{n}}$ ...

$S' = S_j \setminus \{i\}$, so client outputs

$$h_j \oplus a = \left(\sum_{\ell \in S_j}' x_\ell\right) + \left(\sum_{\ell \in S'}' x_\ell\right) \pmod 2$$

$$= \left(\sum_{\ell \in S_j}' x_\ell\right) + \left(\sum_{\ell \in S_j}' x_\ell\right) + x_i \pmod 2$$

$$= x_i.$$

$\Rightarrow$ Scheme fails w.p. $\approx \frac{1}{\sqrt{n}}$.

Repeat the whole thing λ times in parallel to ensure that 1 execution succeed w.p. $\geq 1 - \left(\frac{\sqrt{n}-1}{n}\right)^\lambda \geq 1 - 2^{-\lambda}$.

Security

(Left server)

Sees only the random partition $S_1, \ldots, S_n$

↳ independent of client's index i

(Right server)

Claim: Set S' is a set of $\sqrt{n}-1$ random elements chosen w.o. replacement from $[n]$.

Here is one (funny) way to sample a set S' of $\sqrt{n}-1$ random elements from $[n]$.

Is $i \in S'$?

No w.p. $1 - \frac{\sqrt{n}-1}{n}$ → Output random set of $\sqrt{n}-1$ elements not containing i.

Yes w.p. $\frac{\sqrt{n}-1}{n}$ → Output random set of $\sqrt{n}-1$ elements containing i.

This is exactly how client constructs S'

$\Rightarrow S'$ is a random set, indep of i.

$\Rightarrow$ Info theoretic security

Efficiency

* Need to repeat λ times to drive failure prob $\to 0$.

* Offline cost
  - Server time : λn random reads
  - Comm : upload : 1 PRG seed
    - dnload : $\lambda\sqrt{n}$ parity bits
  - Storage : $\lambda\sqrt{n}$ parity bits + seed

* Online time
  - Server : $\lambda\sqrt{n}$ random reads
  - Comm : $\lambda\sqrt{n}$ bits ← can reduce to poly(λ, log n)
  - Client time : ???

Not clear how to reduce.

Standard PIR server: $n/2$ XORs over DB in linear scan

Off/on PIR offline server: $\lambda \cdot n$ XORs in random order

→ ~128x cost b/c of λ
→ ~10x cost b/c of random reads

Extensions

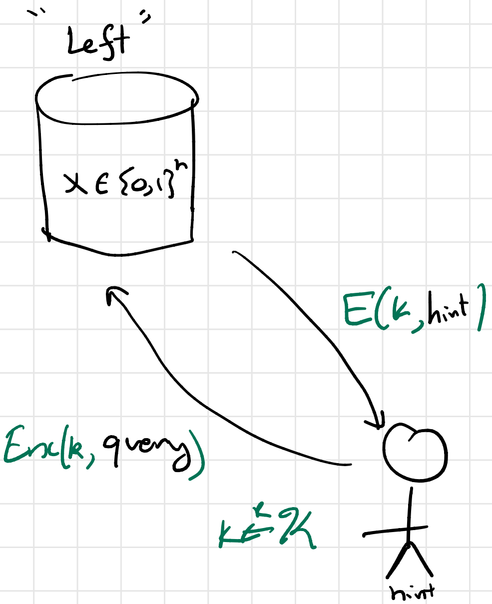
- Can reuse one hint for many subsequent online queries (requires tweaks)

  $\Rightarrow$ Amortized server cost after q queries

  $$\frac{\tilde{O}(n)}{q} + \tilde{O}(\sqrt{n}) \rightarrow \tilde{O}(\sqrt{n}) \text{ for } q \text{ large.}$$

- Works in single-server setting

OFFLINE



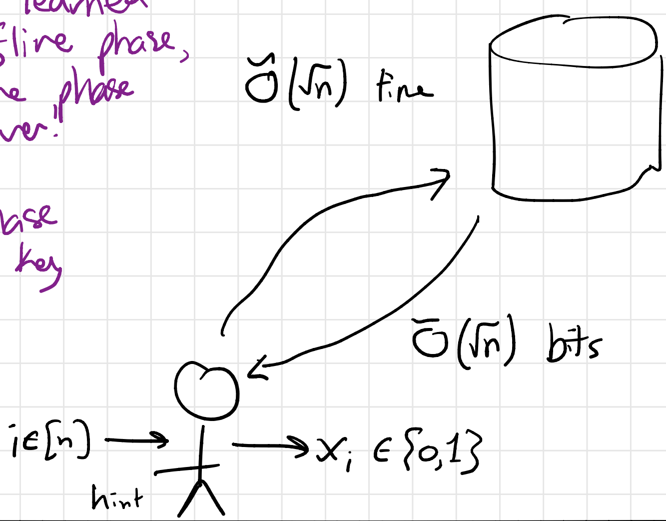
Idea: Offline phase happens "under encryption."

Can do w/ FHE or (worse parameters) lin. hom. enc.

ONLINE

Since server learned nothing in offline phase, can run online phase with same server!

Also, online phase requires no pub key ops.



Moral of the story:

1. Can reduce online server time in PIR using preprocessing.
   ↳ Still new things to say about an old problem.

2. Still not clear whether these techniques will work in practice at scale.

3. Competition is no privacy.
   ↳ How can we ever outperform that?

This is the important question to think about as you are working on crypto + privacy tech.