Privacy Enhancing Technologies

5. PIR

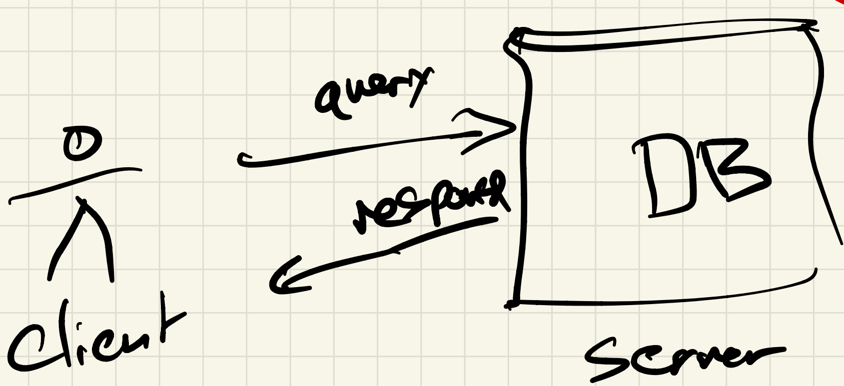
What's a perfect research problem?

1. A beautiful theory ✓
2. It works in practice ~
3. It solves a problem people care about ✓

What's PIR?

"A private Google"

server learns query



What we want: query a DB
without leaking the query

Sol 1: Client $\xleftrightarrow{\text{MPC}}$ Server

↪ Communication: $O(|DB|)$

trivial **Sol 2:**

Client $\xleftarrow{\text{DB}}$ Server

look up DB

What we *really* want: sublinear search

Client $\xrightarrow{q}$ Server

Client $\xleftarrow{\text{res}}$ Server

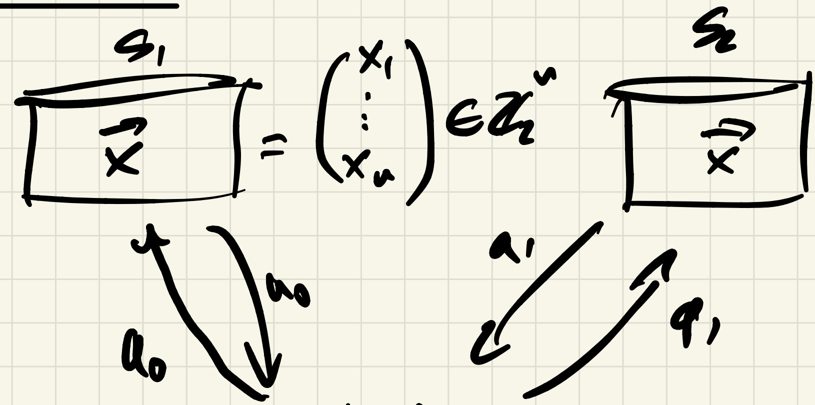
DB_{idx}

$\underbrace{\hphantom{xxxxxxxx}}_{o(|DB|)}$

2 Ways to get PIR:

- public-key crypto
- with non-colluding servers

2-Server PIR



Client

↳ learn x_i

Def: 2-Server PIR

$q_0, q_1 \leftarrow \text{Query}(j)$

$a_i \leftarrow \text{Answer}(\vec{x}, q_i)$

$x_j \leftarrow \text{Reconstruct}(a_0, a_1)$

- Correctness

$$\Pr\left[\text{Recons}(a_0, a_1) = x_j \;\middle|\; \begin{array}{l} q_0, q_1 \leftarrow \text{Query}(j) \\ a_0 \leftarrow \text{Answer}(\vec{x}, q_0) \\ a_1 \leftarrow \quad \text{"} \quad (\text{"}, q_1) \end{array} \right] = 1$$

- Security/privacy:

$\forall\, j, j' \in [n],\ b \in \{0,1\}$

view of server b

$$\{ q_b : q_0, q_1 \leftarrow \text{Query}(j) \}$$

$$\overset{c}{\approx} \{ q_b : q_0, q_1 \leftarrow \text{Query}(j') \}$$

- Sublinear comm: $\forall\, j \in [N],\ \vec{x} \in \mathbb{Z}_2^n,\ \forall b$

$$|\text{Query}(j)| + |\text{Answer}(\vec{x}, q_b)| = o(n)$$

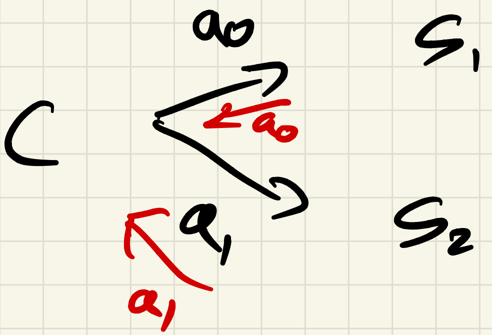
Client wants $x_i = \vec{x} \circ e_i$

ith pos

$[0, \dots, 0, 1, 0, \dots, 0]$

secret-share $e_i = q_0 \oplus q_1$

$\hookrightarrow$ random in $\mathbb{Z}_2^n$



$a_0 = \vec{x} \circ q_0$

$a_1 = \vec{x} \circ q_1$

$q_0 \oplus a_1 = \vec{x}(q_0 \oplus q_1) = x_i$

This isn't sublinear: $q_0, q_1 \in \mathbb{Z}_2^n$

this is non-trivial if each element in the DB has $\gg 1$ bit

Idea: load balancing

Protocol 1 : $\xrightarrow{\text{upload } O(1)}$ $\xleftarrow[O(n) \text{ download}]{}$

Protocol 2 : $\xrightarrow{O(n) \text{ upload}}$ $\xleftarrow[O(1) \text{ download}]{}$

Protocol 3 : $\xrightarrow{\sqrt{n} \text{ upload}}$ $\xleftarrow[\sqrt{n} \text{ download}]{}$

How? view DB as matrix $X \in \mathbb{Z}_2^{\sqrt{n} \times \sqrt{n}}$

$$X = \begin{bmatrix} x_0 & \cdots & x_{\sqrt{n}} \\ x_{\sqrt{n}+1} & \cdots & \\ & \ddots & x_{n-1} \end{bmatrix}$$

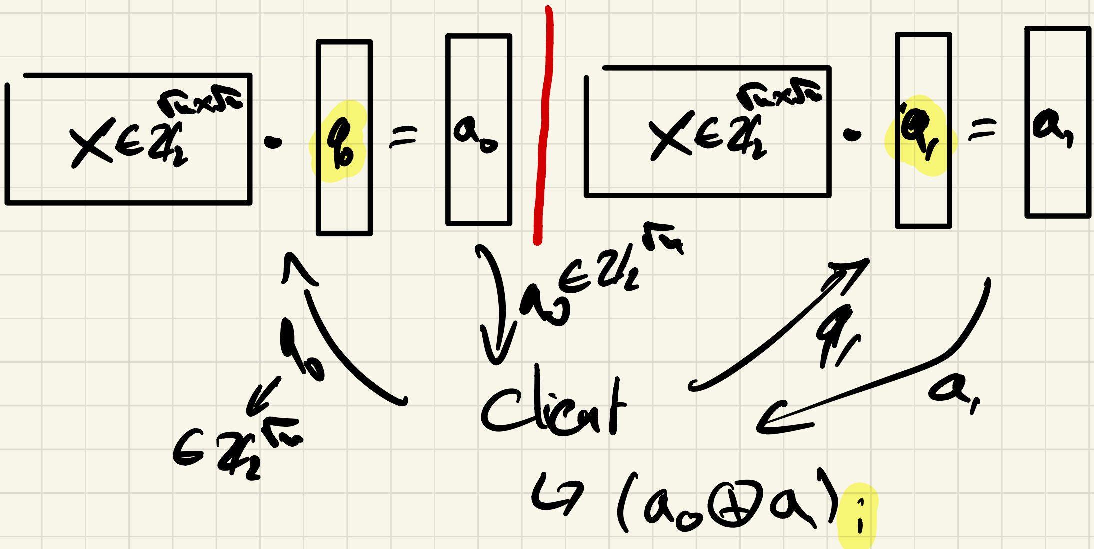
1) do PIR to learn one of $\sqrt{n}$ cols of X

2) do "trivial PIR" on that column

$\mathrm{Query}((\overset{\in\sqrt{n}}{i}, \overset{\in\sqrt{n}}{j}))$: $\quad e_j = \overset{\in \mathbb{Z}_2^{\sqrt{n}}}{q_0} \oplus q_1$

$\mathrm{Answer}(X, q_b)$: $\quad a_b \leftarrow X \cdot q_b$

$\mathrm{Recons}(a_0, a_1)$: $\quad (a_0 \oplus a_1)_i$



total comm : $\underline{\underline{4 \cdot \sqrt{n} \text{ bits}}}$

Single Server PIR

$$\text{Client} \underset{\text{ans}}{\overset{q}{\rightleftarrows}} \boxed{\text{Server}}$$

$$q, sk \leftarrow \text{Query}(i)$$

$$a \leftarrow \text{Answer}(\vec{x}, q)$$

$$x_i \leftarrow \text{Recons}(a, sk)$$

Our strawman scheme

↳ $x_i = \vec{x} \circ e_i$

↳ $(q_0 \oplus q_1)$

Idea: use encryption instead of secret sharing

We need <u>linearly homomorphic encryption</u>

$$E(k, m_1) + E(k, m_2) = E(k, m_1 + m_2)$$

back to strawman scheme:

$$\vec{q} = E(k, \vec{e}_j) = [\overbrace{E(k,0), \dots, E(k,1), \dots, E(k,0)}^{O(n \cdot \lambda) \text{ bits}}]$$

$$\vec{X} \cdot \vec{q} = x_1 \cdot E(k,0) + \dots + x_j \cdot E(k,1) + \dots E(k,0) \cdot x_n$$

$$= E(k, x_j)$$

Load balancing again

$O(\lambda\sqrt{n})$ bits

$[E(k,0), \dots, E(k,1), \dots, E(k,0)]$

$\in [\sqrt{n}]$

(i,j) $\quad q = E(k, \vec{e}_j) \longrightarrow \quad [\,X\,] \in \mathbb{Z}_2^{\sqrt{n}\times\sqrt{n}}$

$\longleftarrow a$

$\mathrm{Dec}(k,a)_i$

$= x_{ij}$

$a = Xq = X \cdot E(k, \vec{e}_j)$

$= E(k, X\vec{e}_j)$

$= E(k, x_j)$

<u>Comm</u>: $|q| + |a| = 2\sqrt{n}$ ciphertexts

$= O(\lambda\sqrt{n})$ bits !

Recursive PIR

Naive

$$C \xrightarrow[\;a \,\circlearrowleft\;]{q \;\leftarrow\; \sqrt{n}\text{ ctxts}} S$$

$$C \xrightarrow{a' \;\leftarrow\; \sqrt{\sqrt{n}}\text{ ctxt}} S \quad a$$

$$C \xleftarrow{\;a'\;} S$$

Comm: $\sqrt{n} + 2\sqrt{\sqrt{n}}$ ctxts

Better:

$$X = \underbrace{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}_{}\!\!\left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} n^{2/3}, \quad \text{width } n^{1/3}$$

(i, j, k), $k \in [\sqrt[3]{n}]$

$j \in [\sqrt[3]{n}]$

$$C \xrightarrow[q_1 = E(k, \vec{e}_i)]{q_0 = E(k, \vec{e}_j)} S$$

$i \in [\sqrt[3]{n}]$

$n^{2/3} \times n^{1/3}$ $\qquad$ $n^{2/3}$

$$X \vec{q}_0 = \vec{a}_0$$

$$[a_0]\, \vec{q}_1 = \vec{a}_1$$

$n^{1/3} \times n^{1/3}$ $\qquad$ $n^{1/3}$

$$C \xleftarrow{\;\vec{a}_1\;} S$$

Comm: $O(\lambda \sqrt[3]{n})$!

More than 1 recursion is possible

view X as d-dimensional cube $\leftarrow O(\log n)$
of side $\sqrt[d]{n}$

limit : $O(\underline{\text{poly}}\log n)$ comm!

One issue : ciphertext blowup

$$a_0 = X \cdot E(k, \vec{e_j}) = E(k, X_j)$$

$$a_1 = a_0 \cdot E(k, \vec{e_i}) = E(k, a_0 \cdot \vec{e_i})$$

$$= \underline{E(k,} \ \underline{E(k,} \ X_{ij}))$$

Why does this matter?

Suppose your Encryption scheme
encrypts t bits into $O(\lambda + 2t)$ bits

after $\log(n)$ recursions, you get a ciphertext of $O(n)$ bits

We need encryption schemes with low (constant) expansion

Q: Does this recursive PIR idea work for 2-server PIR?

Modern PIR

- 2 server: - info theoretic: $n^{o(1)}$
  - computational: $O(\log n)$
- Single server: $O(\text{poly}\log n)$

Computational complexity

All our schemes have server complexity $O(n)$

Maybe inherent: if server doesn't "touch" x_i, then the query can't be i

Solns: 1) Batching: q queries in $O((\lambda \log q) n)$ work

2) pre-processing

$$C \xleftarrow{\text{"hint"}} S \quad \text{takes } O(n) \text{ work}$$

$$\left.\begin{array}{l} C \xrightarrow{q} S \\ C \xleftarrow{a} S \end{array}\right\} \text{online phase takes } o(n) \text{ work}$$

LMW23:

$$C \xrightarrow{q} S \qquad \underbrace{\text{encode } X \text{ as } X'}_{O(n) \text{ work}}$$

$$\underbrace{C \xleftarrow{a} S}_{O(\text{poly} \log n) \text{ work}}$$