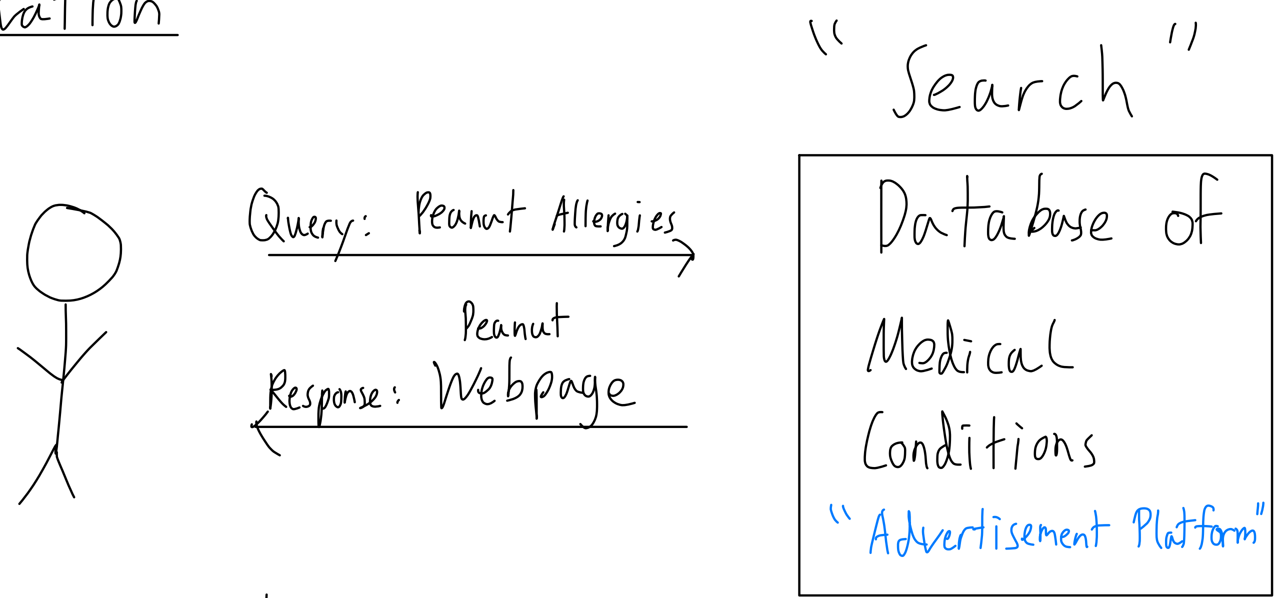
Private Information Retrieval

- Motivations
- 2-Server PIR
- Single Server PIR
- Modern Results

Motivation



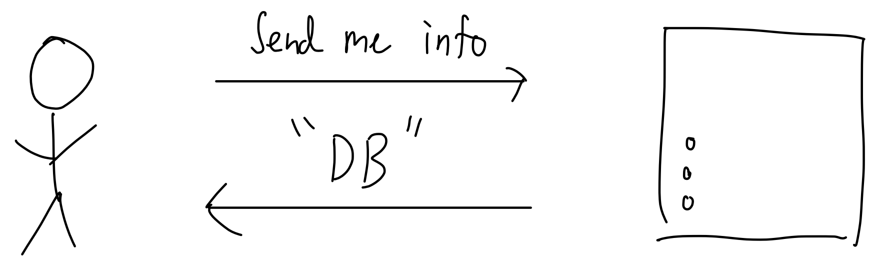
- Database server learns your query and can sell this information to Ad networks or health providers.
- Could lead to more targeted ads, higher premiums, targeted pricing

No privacy!!!

- If only we could query databases without the server learning any information about our query!

We can with Private Information Retrieval!

First (trivial Construction)



- The server could just send the whole database!
- <u>Issue:</u> Communication is linear!
- <u>Question:</u> Can we obtain sublinear communication without the server learning our query?

Not Quite! — if comm < |DB|, then we lose info about some entry j; an unbounded server could distinguish which entry!

Getting around the issue!

<u>Noncollusion [CGKS95]:</u>

- Replace single server with 2 or more non-colluding servers.

  Not allowed to communicate!
- info-theoretic

<u>Computational Bound [KO97]:</u>

- Assume the server is a computationally bounded adversary
- rely on cryptographic primitives

Two Server PIR

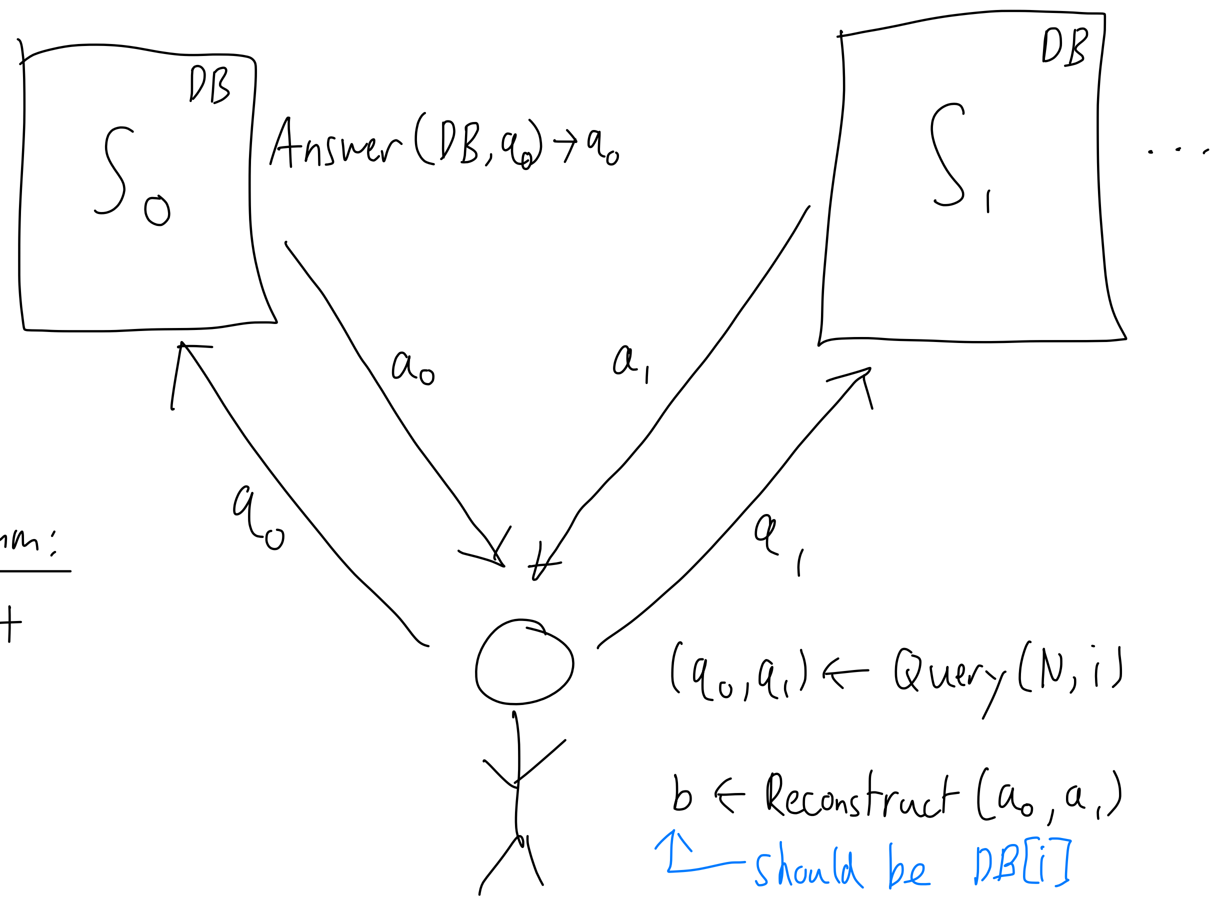
- We'll refer to the servers as S_0, S_1 and treat the database $DB \in \{0,1\}^N$ (i.e. Entry $j \in [N] = DB[j] \in \{0,1\}$)

A *two server PIR* scheme is a tuple of eff algs:

- $\text{Query}(N, i) \to (q_0, q_1)$: $N = |DB|$ and i is the query index. q_0, q_1 are query strings sent to S_0, S_1 respectively.

- $\text{Answer}(DB, q) \to a$: $DB \in \{0,1\}^N$, q is the query string from the client and a is the server response.

- $\text{Reconstruct}(a_0, a_1) \to b$: a_0, a_1 are responses from S_0, S_1 and b is the reconstructed entry.



Total Comm:
$|a_0| + |a_1| + |q_0| + |q_1|$

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

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

should be $DB[i]$

Correctness:

$\forall N \in \mathbb{N}, i \in [N], DB \in \{0,1\}^N,$

$$\Pr\left[b = DB[i] : \begin{array}{l} (q_0, q_1) \leftarrow \text{Query}(N, i) \\ a_0 \leftarrow \text{Answer}(DB, q_0) \\ a_1 \leftarrow \text{Answer}(DB, q_1) \\ b \leftarrow \text{Reconstruct}(a_0, a_1) \end{array} \right] = 1$$

Security:

$\forall$ polybounded $N \in \mathbb{N}$, $DB \in \{0,1\}^N$, pairs $(i, j) \in [N]^2$, $b \in \{0,1\}$, the following distributions are indistinguishable,

$$\{ \text{Query}(DB, i)[b] \} \approx \{ \text{Query}(DB, j)[b] \}$$

could be perfect or computational

Note: b selects only one of the server queries.

Warmup: A trivial construction!

ith basis vector

Query(N, i): Sample $q_0 \xleftarrow{r} \{0,1\}^N$ and $q_1 \leftarrow q_0 \oplus e_i$. Output (q_0, q_1)

Answer(DB, q): Output $a \leftarrow \langle DB, q \rangle$ ← inner product

Reconstruct(a_0, a_1): Output $b \leftarrow a_0 \oplus a_1$

Correctness: $a_0 \oplus a_1 = \langle DB, q_0 \rangle \oplus \langle DB, q_1 \rangle$

$$= \langle DB, q_0 \oplus q_1 \rangle$$

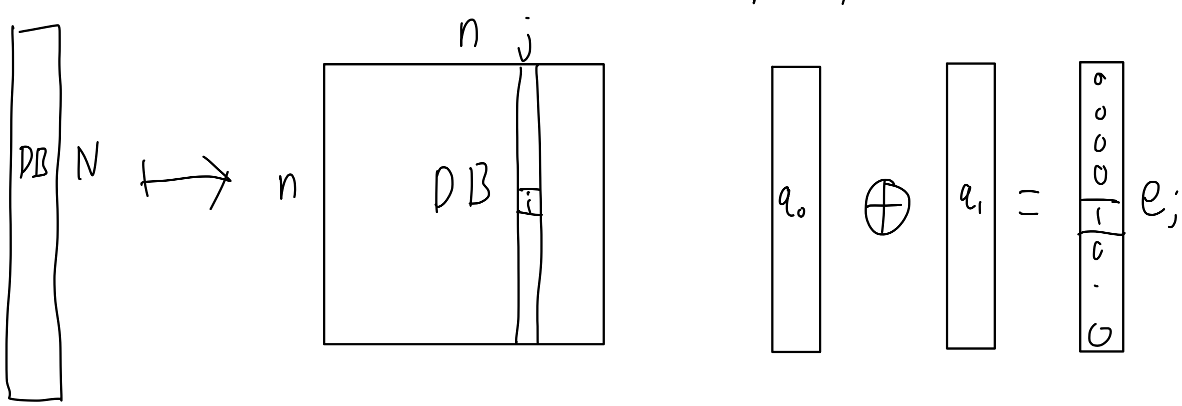
$$= \langle DB, e_i \rangle = DB[i]$$

Security: Separately, q_0 and q_1 are uniformly random N bit vectors. Thus, the distributions will be identical!

Issue: Communication is still linear! (swapped direction upload vs download with just sending DB)

CGKS Construction

- Assume $N = n^2$ is a perfect square for simplicity.



Query$(N, (i,j))$ Sample $q_0 \xleftarrow{r} \{0,1\}^n$ and $q_1 \leftarrow q_0 \oplus e_j$. Output (q_0, q_1)

Answer(DB, q): Compute vector $a \leftarrow DB \cdot q$. Output a.

Reconstruct(a_0, a_1): Output $b \leftarrow (a_0 \oplus a_1)[i]$

Matrix $n \times n$ vector $n \times 1$ product

Correctness: $a_0 \oplus a_1 = DB \cdot q_0 \oplus DB \cdot q_1$

$$= DB \cdot (q_0 \oplus q_1)$$

$$= DB \cdot e_j = DB_j \quad \text{"jth col of DB"}$$

$DB_j[i] = DB[i,j]$.

Security: Similarly, both q_0 and q_1 are uniformly random vectors.

Communication: $O(\sqrt{N})$ bits! (upload = download)

Can we do better in the two server setting?

- When considering unbounded servers (info-theoretic), [DG15] obtain

$$O\left(N^{\sqrt{\log\log N / \log N}}\right) = O\left(N^{o(1)}\right) \text{ comm from}$$

little o

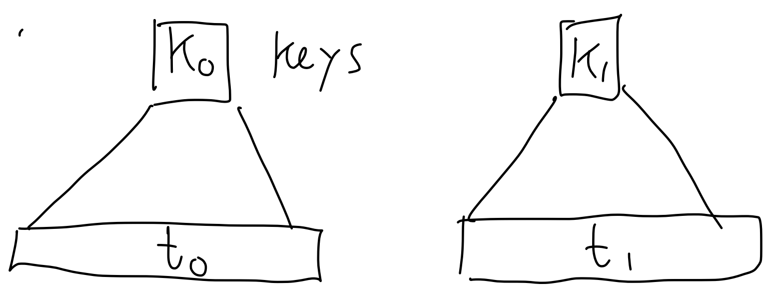
locally-decodeable codes.

$DB \mapsto C$

[BGIL15]

- In computationally bounded setting, Distributed Point functions (constructed from PRGs) gives us a 2-server PIR with $O_\lambda(\log N)$ communication.

Expand



$t_0 \oplus t_1 = e_i$

Computational Single Server PIR

- [KO97] leverages linearly-homomorphic encryption (El-gamal, etc)

$$\forall k \in \mathcal{K},\ \forall m_0, m_1 \in \mathcal{M},$$
$$\mathrm{Enc}(k, m_0) + \mathrm{Enc}(k, m_1) = \mathrm{Enc}(k, m_0 + m_1)$$

Query$(N, (i,j))$: $k \xleftarrow{r} \mathcal{K}$. Output query

$$q \leftarrow (\mathrm{Enc}(k, e_j[1]), \mathrm{Enc}(k, e_j[2]), \ldots, \mathrm{Enc}(k, e_j[n]))$$

vector of ciphertexts

Answer(DB, q): Output $a \leftarrow DB \cdot q$.

$$\boxed{DB} \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} = \underset{DB_1}{\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}} c_1 + \underset{DB_2}{\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}} c_2 + \cdots$$

Linear Homomorphism

$$= \underbrace{\begin{bmatrix} c_1 + c_2 + \cdots \\ c_1 + \cdots \\ 0 + c_2 + \cdots \\ \vdots \end{bmatrix}}_{\leq n \text{ additions}}$$

Reconstruct(k, a): Decrypt ciphertexts,

$$u \leftarrow \left(\mathrm{Dec}(k, a_1), \mathrm{Dec}(k, a_2), \ldots, \mathrm{Dec}(k, a_n) \right)$$

Output $u[i]$.

<u>Correctness:</u> Follows from linear homomorphism of Enc,

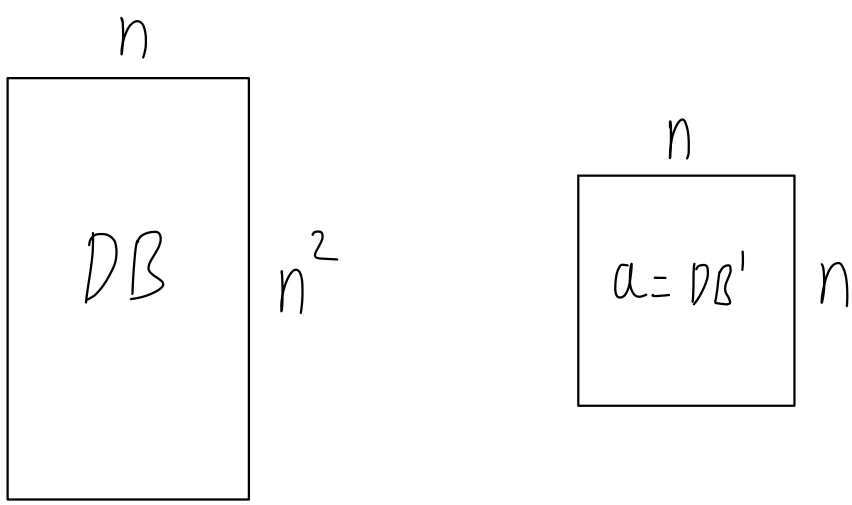
$$a = DB \cdot q = \begin{bmatrix} \sum_t DB_{1,t} \cdot q_t \\ \vdots \\ \sum_t DB_{j,t} \cdot q_t \end{bmatrix} = \begin{bmatrix} \mathrm{Enc}(k, DB_{1,j}) \\ \mathrm{Enc}(k, DB_{2,j}) \\ \vdots \\ \mathrm{Enc}(k, DB_{n,j}) \end{bmatrix}$$

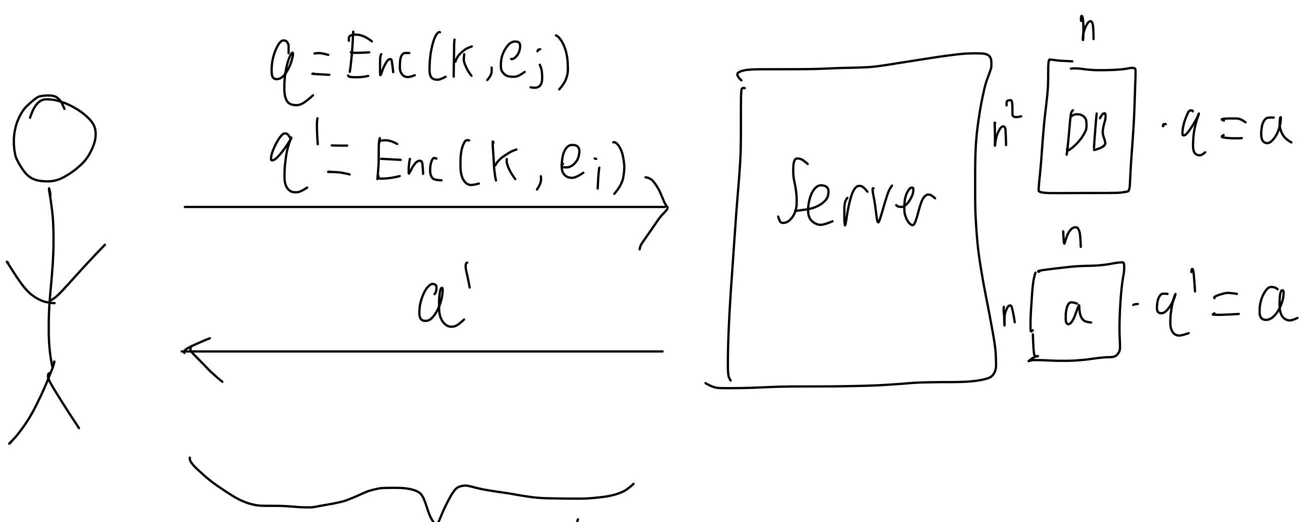
<u>Security:</u> Follows from semantic security of Enc, notably that the Enc of an e_j should be computationally indistinguishable from $e_{j' \neq j}$.

<u>Comm:</u> $|\text{Ciphertext}| = O(\lambda)$ so overall comm is $O(\lambda n) = O(\lambda \sqrt{N})$

Can we do better?

Idea: Recursion! Notice the client only needs 1 ciphertext, so why not treat the server response as another database DB'. $N = n^3$





$O_\lambda(n) = O_\lambda(N^{1/3})$

Can we recurse further?

Issue: $|\text{Entry}| = 1$ bit

$$\Downarrow$$

$|\text{ciphertext}| = \lambda$ bit

So, everytime we recurse we blow up communication by a factor of λ. $|a| = \lambda^2 n^2$, $|a'| = \lambda^2 n$

Solution: Leverage a lin-hom scheme which

- good rate: $|m| / |c|$ is closer to 1
- large message spaces

[Damgard-Jurik]

Best Comp Single Server: Polylog(n) comm [CMS99, Lip05] from QR, DDH, LWE assumptions