Recap:

We have seen a very strong general result:

Secure multiparty computation protocol for any efficiently computable function

$\Rightarrow$ parties learn nothing but the output of the function.

Last lecture:

- Differential Privacy: "When even learning only the output of the function might not capture our privacy requirement"

Solution: add some randomize noise s.t

$$\Pr[M(D, q) \in S] \leq e^{\varepsilon} \cdot \Pr[M(D', q) \in S]$$

mechanism

neighbouring DBs

privacy param.

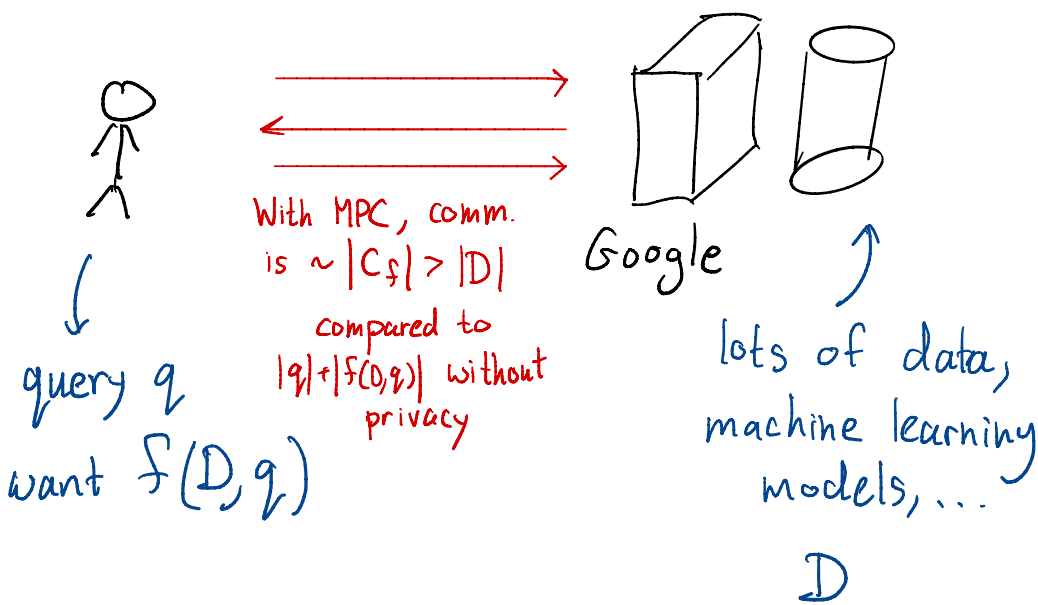
query

"bad" event

Another limitation of general purpose MPC:

The communication complexity is proportional to the size of the circuit computing the function.

E.g., Suppose we want protocol for

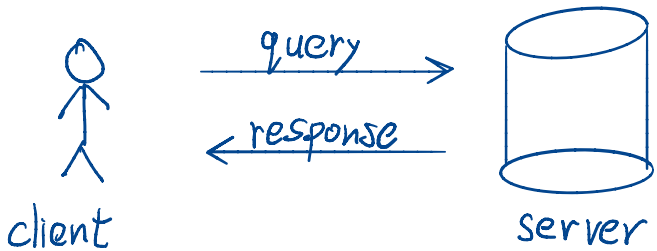


In the remaining part of the course, we will see what can we do about this

- Today: special case of DB lookups
- Next week: FHE - general purpose

Private Information Retrieval

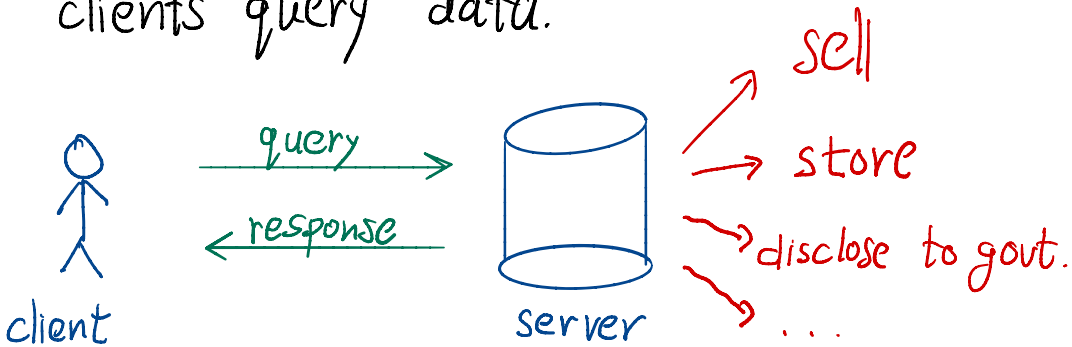
Every day on the internet:



Clients query often contains sensitive information (even when the data stored on the server is public). For example:

- WebMD - db of medical conditions
  ⇒ query contains information about disease
- Stock-Prices DB
  ⇒ query contains info about financial interests
- Patents DB, domain registry, ...

Encryption (HTTPS) only protects the privacy of the client's query against a network adversary. The server still sees the query in the clear, and can sell/abuse/store/forced to disclose to government the client's query data.



Question: can you query DB without the DB operator learning your query?

Trivial answer: Yes! Just download the entire DB.

Better Question: can you do it efficiently?

with communication sublinear in DB size

Unconditionally, the answer is No.

Intuition: if client downloads $< n$ bits, then cannot get information-theoretic security.

Can prove this formally

But luckily, as often happens in crypto, can workaround such "impossibility":

Option 1 [CGKS'95]

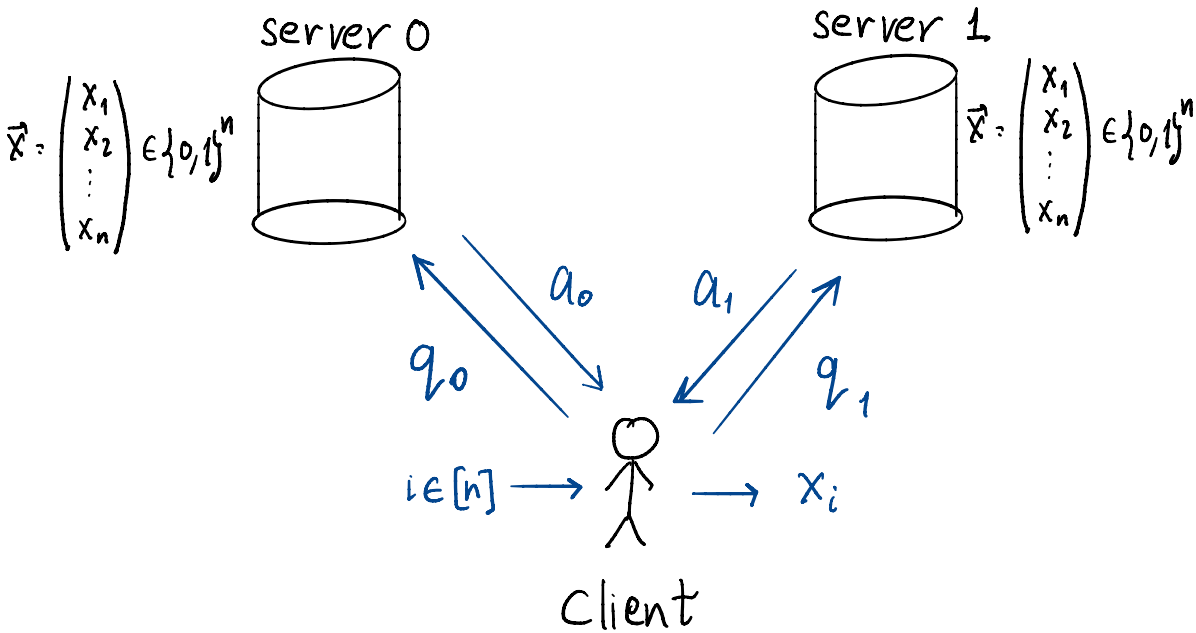
can get IT security by replicating DB on two or more non-colluding servers.

Option 2 [K'97]

can get computational security, under suitable assumptions

2-Server PIR

The model:



Important! security only against non-colluding servers.

Non-essential simplifications:

- DB is an array of bits (can be generalized to larger records)
- Lookup by index (can generalize to lookup by keywords)

More formally:

(2-Server) PIR consists of 3 algorithms:

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

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

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

<u>Properties:</u>

(1) <u>Correctness</u>: $\forall n \in \mathbb{N}, \forall i \in [n], \forall x \in \{0,1\}^n$

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

can relax to $1 - \text{negl}(\lambda)$

(2) <u>Security</u>: $\forall n \in \mathbb{N}, \forall i, i' \in [n], \forall \beta \in \{0,1\}$

$$\{q_\beta : q_0, q_1 \leftarrow \text{Query}(n, i)\} \approx \{q_\beta : q_0, q_1 \leftarrow \text{Query}(n, i')\}$$

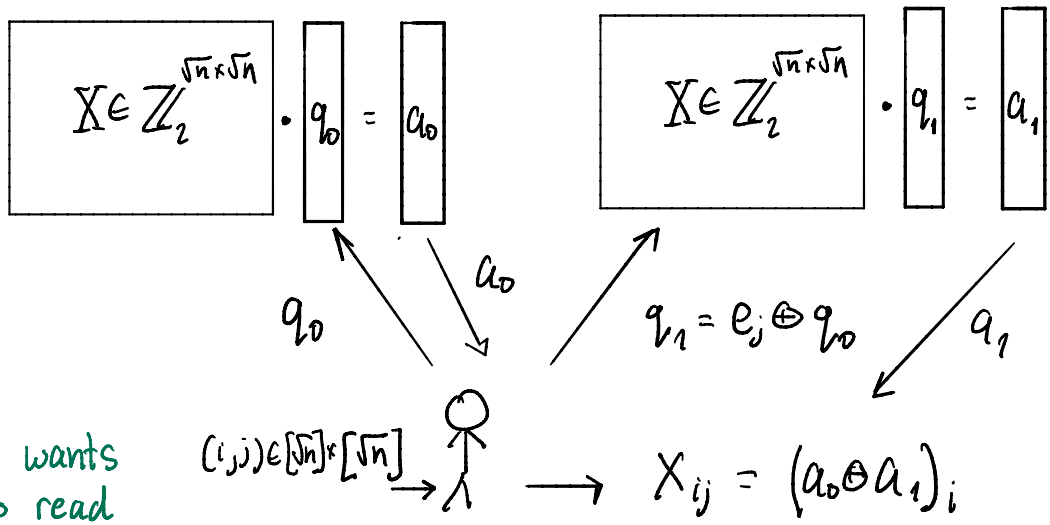
How is non-collusion assumption captured?

$\Rightarrow$ We only require the <u>marginal distribution</u> of q_β to be indist.

(q_0, q_1) together may leak i

Example: 2-Server PIR with $O(\sqrt{n})$ communication

Idea: view DB as a matrix $X \in \mathbb{Z}_2^{\sqrt{n} \times \sqrt{n}}$



$\text{Query}(n, (i,j)) \to (q_0, q_1)$

sample $q_0 \xleftarrow{\$} \mathbb{Z}_2^{\sqrt{n}}$

set $q_1 \leftarrow e_j \oplus q_0$

$\text{Answer}(x, q) \to X \cdot q \in \mathbb{Z}_2^{\sqrt{n}}$

$\text{Reconstruct}(a_0, a_1) \to (a_0 \oplus a_0)_i$

Correctness:

$$(a_0 \oplus a_1)_i = (X \cdot q_0 \oplus X \cdot q_1)_i$$

$$= (X \cdot q_0 \oplus X \cdot (e_j \oplus q_0))_i$$

$$= (X \cdot (q_0 \oplus e_j \oplus q_0))_i =$$

$$= (X \cdot e_j)_i = X_{ij}$$

Security:

q_0 is uniformly random independent of (i,j)

$q_1 = e_i \oplus q_0$ is also uniformly random.

Efficiency:

upload: $|q_0| + |q_1| = 2\sqrt{n}$

download: $|a_0| + |a_1| = 2\sqrt{n}$

Total communication $O(\sqrt{n})$ bits.

Single-server PIR:

Idea: instead of secret sharing $e_j = q_1 \oplus q_2$,

encrypt e_j using linearly homomorphic encryption

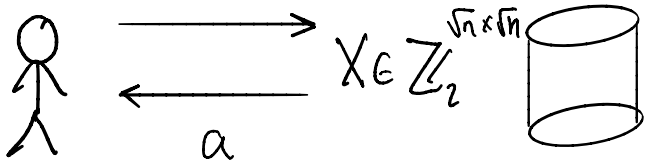
$$E(k, m_1) + E(k, m_2) = E(k, m_1 + m_2)$$

can build from DDH, Quadratic Residuosity, ...

element-wise encryption

$$q = \mathrm{Enc}(k, e_j) = \big(E(k,0), \dots, E(k,1), \dots, E(k,0)\big)$$

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



$$a = X \cdot q = X \cdot \mathrm{Enc}(k, e_j)$$
$$= \mathrm{Enc}(k, X \cdot e_j)$$
$$= \mathrm{Enc}(k, \vec{x}_j)$$

requires only adding ciphertexts

output

$$x_{ij} = \big(\mathrm{Dec}(k, a)\big)_i$$
$$= \big(\mathrm{Dec}(k, \mathrm{Enc}(k, x_j))\big)_i$$
$$= x_{ij}$$

Communication: $|q| + |a| = O(\lambda \sqrt{n})$

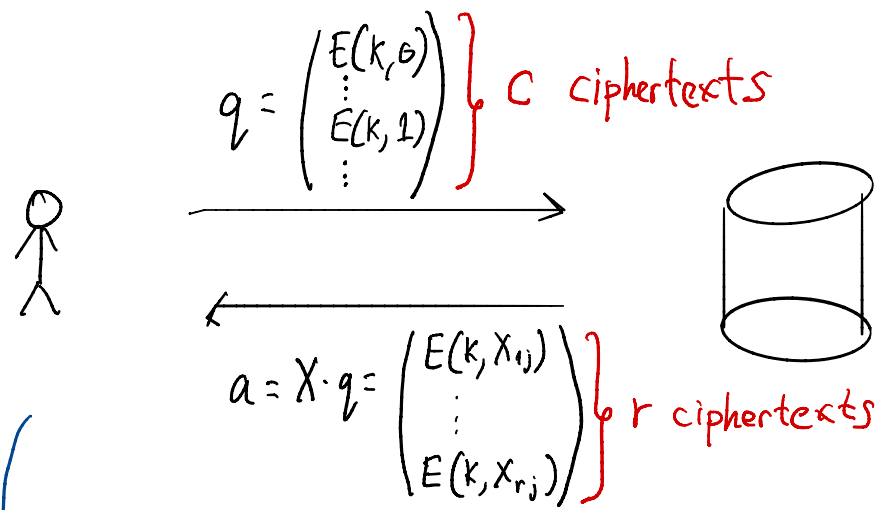
$\sqrt{n}$ ciphertexts each

Further reducing communication

Viewing the database as a $r \times c$ matrix

$$X = \begin{array}{c} \\ r \end{array}\overset{c}{\begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}}$$

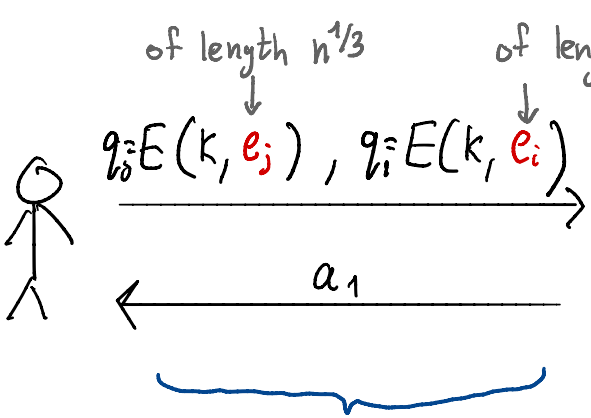
In the single-server scheme



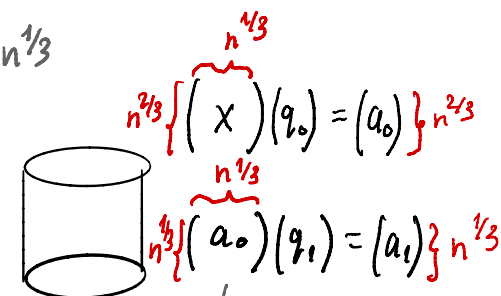
but client only needs one of the r ciphertexts in a (i.e., a_i)

<u>Idea:</u> view a as a database with r records (albeit larger than 1-bit) use PIR to read record i in that DB.

$$X = \underbrace{\left(\quad\right)}_{n^{1/3}} \Big\} n^{2/3}$$



total communication $O(n^{1/3})$ ciphertexts



express a_0 as a $n^{1/3} \times n^{1/3}$ matrix

Can then apply the same idea recursively to shrink communication even further.

However, there is a subtlety:

Usually the size of the ciphertext is larger than the size of the plaintext (e.g., when plaintext is 1 bit, ciphertext is $O(\lambda)$ bits, and e.g., in ElGamal $|m| = \log q$, $|ct| = 2\log p$)

This means that there is a blowup each step of the recursion:

$$n \text{ bits} \Rightarrow \sqrt{n} \text{ ciphertexts})$$

So we need an encryption scheme in which said blowup is small

$\Rightarrow$ e.g., Damgard-Jurik (generalization of Paillier Enc.)

State-of-the art in PIR (communication complexity)

- Two-server PIR
  - Information theoretic - $O\left(n^{\sqrt{\log\log n / \log n}}\right) = n^{o(1)}$

    [Dvir-Gopi 15]
  - Computational - $O(\log n)$ [G114, BG115]

    ↳ concretely very efficient

    ↳ requires only PRGs

- Single-server PIR
  - $\mathrm{polylog}(n)$ - from QR, DDH, LWE, ...

    [CMS99, Lip05, ...]

Computational Efficiency in PIR

In all the above schemes, the server's running time is linear in DB size

$\Rightarrow$ Very expensive in practice to have to scan entire DB on each query

Unfortunately, this is required (in some sense)

[BIM04] Lower Bound: server has to do $\Omega(n)$ work to respond to query.

Workaround approaches

- Preprocessing the DB
- Batching queries
- ...

active area of research (including in our group)