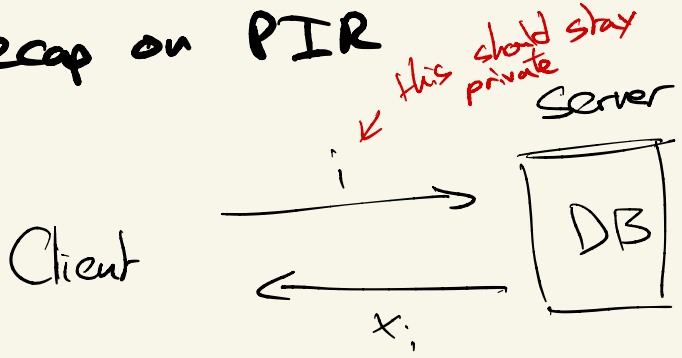
Privacy Enhancing Technologies

Private Information Retrieval

Florian Tramèr

Recap on PIR



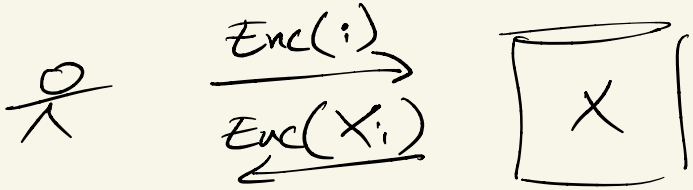
last time: 2 non-colluding servers

more generally k out of n servers

(k = # that need to collude to break privacy; n = # of servers)

2 ideas:
* "encrypt" the query (secret sharing)
* load balancing

Single Server PIR



We'll need some form of homomorphic encryption.

How:

* FHE: see HW #?

* linearly homomorphic encryption

$$Enc(k, m_1) \text{ "+" } Enc(k, m_2) = Enc(k, m_1 + m_2)$$

Recall: $Commit(m, r) = g^m h^r$

linearly homomorphic encryption can be built from Diffie Hellman, QR, RSA, CWE, etc.

PIR from linear-homomorphic encryption

Strawman:

Client → Server: $\vec{q}_{st} = [0, \ldots, 0, \underbrace{1}_{\text{index } i}, 0, \ldots 0]$

Server → Client: $\vec{x} \circ \vec{q}_{st} = \vec{x}_i$

Scheme:

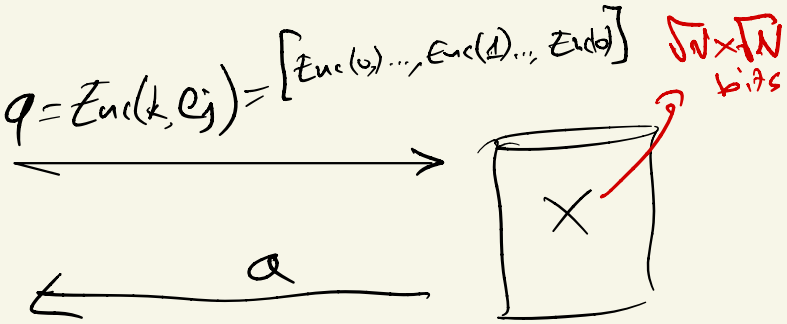
$$\vec{q} = Enc(\vec{q}_{st}) = [Enc(0), \ldots, Enc(1), \ldots Enc(0)]$$

response: $\vec{x} \circ \vec{q} = Enc(0) \cdot x_1 + \ldots + Enc(1) \cdot x_i + \ldots$
$= Enc(x_i)$

Issue: Communication is $O(\lambda \cdot |DB|)$

Now: load balancing

1) $\sqrt{N}$ scheme



$Dec(k,a)_i = \vec{X}_{ij}$

$a = X \cdot q$

$= X \cdot Enc(k, e_j)$

$= Enc(k, X \cdot e_j)$

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

Communication: $|q| + |a|$

$O(\sqrt{N} \cdot \lambda)$ $O(\sqrt{N} \cdot \lambda)$ bits

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

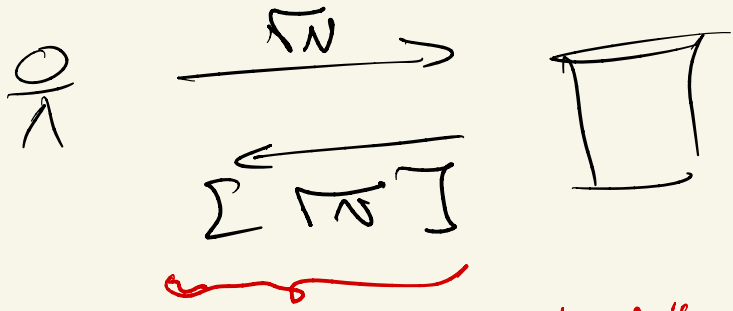
$Enc(k, m_1) \cdot c = Enc(k, m_1 \cdot c)$

$Enc(k, m_1) \cdot c = \underbrace{Enc(k, m_1) + \cdots + Enc(k, m_1)}_{c \text{ times}}$

2) Recursion ∇_0

We'll show how to get a $N^{1/3}$ scheme

Insight:



Here we "download" $\sqrt{N}$ elements even though we only want one of them (index :)

This is essentially a trivial PIR

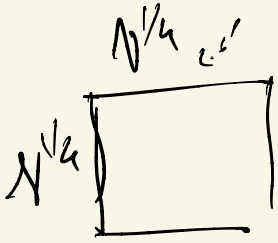
Why don't we do PIR again?

Naive approach

Client (i,j) $\xrightarrow{Enc(e_j)}$ 

PIR for element i of a

$a = [Enc(x_{1,j}) \ldots Enc(x_{N,j})]$ (column j)



$$\text{Comm} : O(\lambda \cdot \lceil N \rceil) + O(\lambda \cdot N^{1/4})$$

$$= O(\lambda \cdot \lceil N \rceil)$$

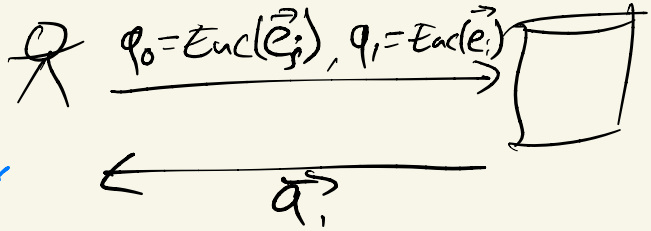
The $N^{1/3}$ approach:

$$X = \left[\; a \;\right]^{N^{2/3}}$$

a : $N^{1/3}$

$$a = N^{1/3}\left[\; \cdot \;\right]^{N^{1/3}}$$

$(i,j,k) \in [N^{1/3}] \times [N^{1/3}] \times [N^{1/3}]$



$X \cdot q_0 = a_0$ ($N^{2/3} \times N^{1/3}$ · $N^{1/3}$ = $N^{2/3}$)

$A \cdot q_1 = a_1$ ($N^{1/3} \times N^{1/3}$ · $N^{1/3}$ = $N^{1/3}$)

$Dec(\vec{a}_i)_k = X_{ijk}$

recurse again!!!

Recurse again $\Rightarrow O(N^{1/4})$ scheme

$\Rightarrow O(N^{1/5})$ scheme

$\vdots$

$\log$ recursions $\quad O(\underbrace{\text{poly}(\log(N))}_{\log(N)^k \text{ for some } k} \text{ comm.}$

Ciphertext blowup:

$$a_0 = X \cdot Enc(k, \vec{e}_j) = Enc(k, X \cdot \vec{e}_j) = Enc(k, \vec{X}_j)$$

$$a_1 = \overbrace{A_0}^{\text{view as ctxt}} \cdot Enc(k, \vec{e}_i) = Enc(k, A_0 \cdot \vec{e}_i) = \underbrace{Enc(k, Enc(k, \vec{X}_{ij}))}_{\text{double layer of encryption}}$$

Naively: $Enc(1\,bit) \Rightarrow \lambda$ bit ciphertext

$Enc(Enc(1\,bit)) \Rightarrow \lambda^2$ bits

$\log n \Rightarrow$ Too large $\forall$

What you need: encryption scheme with "low expansion"

$(|ctxt| \approx |plaintext|)$

Computational Complexity

For now, we only cared about communication

Server work is always $O(\lambda \cdot N)$

$\Rightarrow$ isn't this unavoidable?

You can get around this with:

- Batching: answer K PIR queries at once in $O((\lambda \log K) \cdot N)$ server work

- Preprocessing of the DB
  - 2-server setting with a one-time offline preprocessing, server work $O(\sqrt{N}\,\mathrm{polylog} N)$
  - Single server: special DB encoding with $O(\mathrm{polylog} N)$ online server time

Last notes on PIR:

Q: Why does the recursive PIR idea not work in the 2-server setting?

State-of-the-art in 2-server PIR:

⇒ see homework on PIR from Distributed Point Functions

→ DPF ≈ a way to secret share $\vec{e}_j$ with $o(N)$ communication

↳ $poly(\log(N))$ communication using only symmetric crypto