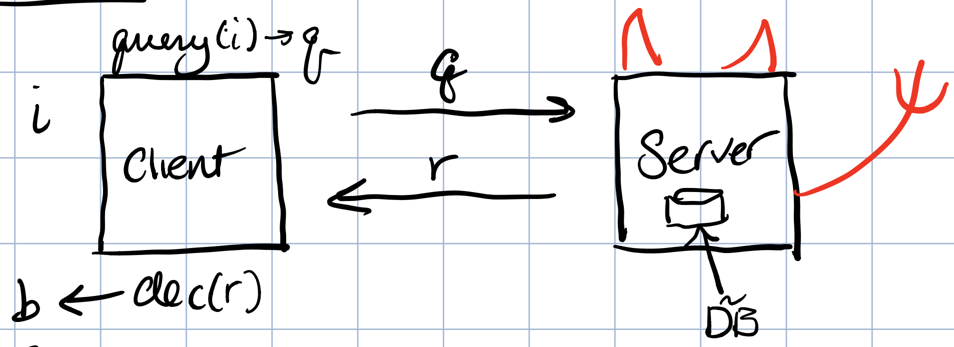
Doubly Efficient Private Information Retrieval

Model



Goal: • Not reveal the index being searched for (Note: not necessarily hide the contents of database)

Context: the larger world of PIR

- Multiserver: two or more non-colluding servers with clients talking to them
  - can acheive information theorectic security
- Single server schemes: single server
  - relies on computational assumptions
- Efficiency — lots of work focusing on communication complexity
  - computational complexity for server is often a real bottleneck
  - most schemes have linear server work & ∃ lower bound that

if no pre-processing is allowed work for server must be linear [Bim00]

- Relation to ORAM:
  - ORAM can get sublinear work so what is the difference?
  - ORAM schemes have clients maintain state ⇒ PIR schemes are assumed to natively be "Public Key" → shouldn't need secret state to make a query ⇒ don't want to have to have clients share state
  - ORAM requires the ability to write or update the database ⇒ PIR is read only

Goal: If we allow preprocessing can we acheive sub-linear server work?
slight variant where we are going to have two versions
- secret-key
- public-key

Going to call this Doubly Efficient PIR (DEPIR)

So what is DEPIR: (secret-key)

(KeyGen, Process, Query, Resp, Dec)

- → $\text{KeyGen}(1^\lambda) \to sk$
- → $\text{Process}(k, DB) \to \widetilde{DB}$
- → $\text{Query}(k; i) \to q, \text{state}$
- → $\text{Resp}(\widetilde{DB}, q) \to c$
- → $\text{Dec}(k, \text{state}, c) \to DB[i]$ (y)

- Correctness

$$\forall\ DB,\ i \in [N]$$

$$\Pr\left[\text{Dec}(k, st, c) = DB[i] : \begin{array}{l} k \leftarrow KGen(1^\lambda) \\ \widetilde{DB} \leftarrow Proc(k, DB) \\ (q, st) \leftarrow Query(k, i) \\ c \leftarrow Resp(\widetilde{DB}, q) \end{array}\right] = 1$$

- Efficiency: - KeyGen $O(\lambda)$
  - Process $\text{poly}(N, \lambda)$
  - query, dec $o(N) \cdot \text{poly}(\lambda)$

- Security:

C A

$b \in \{0,1\}$ ⟵ DB

$K \leftarrow KGen(1^\lambda)$

$\widetilde{DB} \leftarrow Process(K, DB)$

⟶ $\widetilde{DB}$ ⟶ $i_0, i_1 \in [N]$

$q_b \leftarrow Query(K, i_b)$ ⟵ i_0, i_1 ⟵

⟶ q_b ⟶

repeat poly times

↓

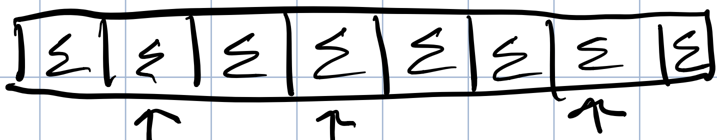
b'

Construction: The long winding road of code theory

- There is a thing called Error correcting codes - they are used in <u>many</u> places
- we are interested in a variant called <u>locally</u> decodable codes ⇒ at a high level LDCs ⟺ SK-DEPIR

$M \in \Sigma^N \longrightarrow$ Enc $\longrightarrow$ codeword

(M ← message, Σ ← alphabet; $c \in \Sigma^M$)



$i \to$ Query $\longrightarrow (x, x, \ldots, x_k)$

$\text{Dec}(x_1, \ldots, x_k) \to y$

Properties: • local decodability

$k \ll M$

• Smoothness:

$\underbrace{x_1 \ldots x_k}_{s}$ every subset of size s appears to just be random indexes in codeword

We have constructions of LDCs using Reed-Mueller codes with "reasonable" parameters ← will get into reasonable later, basically DB is a low degree extension and query are random polynomials

KeyGen(1^λ): t-smooth LDC parameters

also choose [(k-indices for query)

k permutations over encoded ($\uparrow i$)

DB size [M]

$sk = (\pi_1, \ldots \pi_k)$

<u>Process(sk, DB)</u>: $\widetilde{DB} \leftarrow LDC.Enc(DB)$

$\widetilde{DB}_{(i)} \Leftarrow \pi_i(\widetilde{DB})$

output $(\widetilde{DB}_1, \ldots, \widetilde{DB}_k)$

<u>Query(sk, i)</u>: $j_1, \ldots, j_k \leftarrow LDC.query(i)$

output $(\pi_1(j_1), \ldots, \pi_k(j_k))$

<u>Resp$(\tilde{j}_1, \ldots \tilde{j}_k, \widetilde{DB}_1, \ldots, \widetilde{DB}_k)$</u>:

output $\widetilde{DB}_1[\tilde{j}_1] \ldots \widetilde{DB}_k[\tilde{j}_k]$

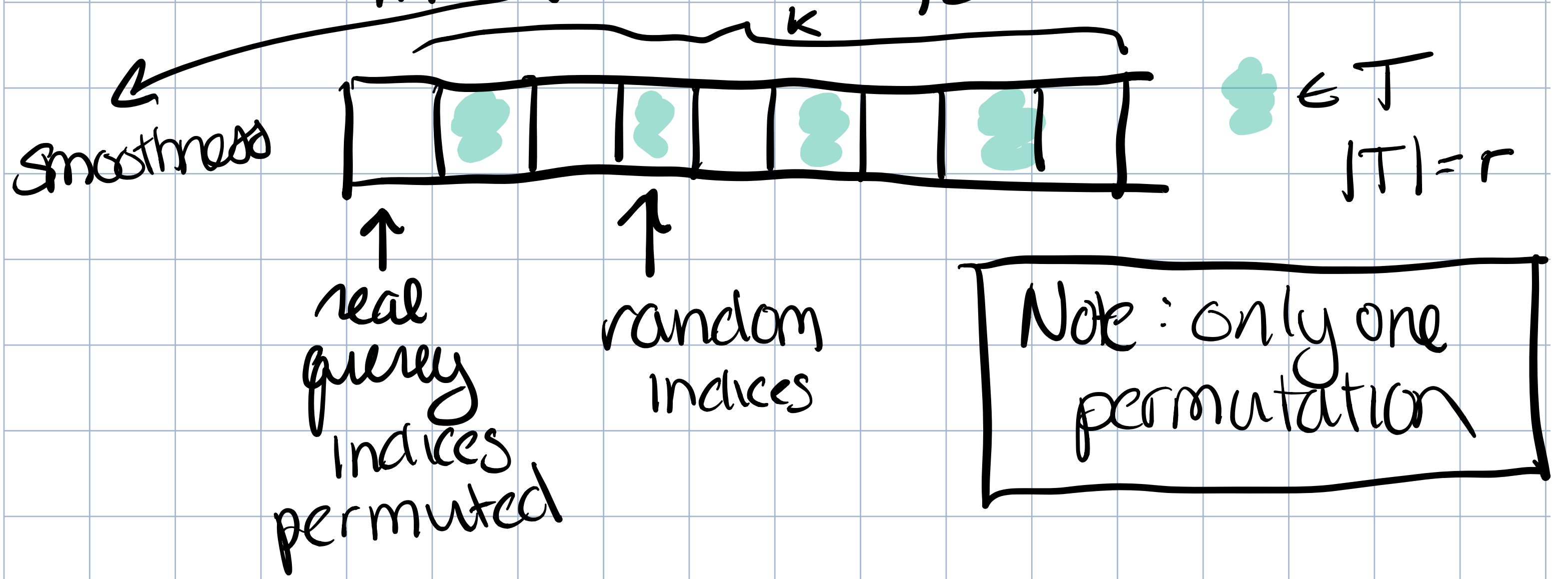
<u>Dec$(y_1 \ldots y_k)$</u>: output $LDC.Dec(y_1 \ldots y_k)$

So this scheme is info-theorectic security to a <u>bound B</u>, smoothness and the permutation. The sk size is linear in the bound (# of permutations)

We can make unlimited queries if we introduce a computational assumption

HPN (Hidden Permutation w/ Noise):

$m < t < r < k < |\Sigma|$



HPN is a new type of 'permuted puzzle' assumptions, some work in this field where simpler assumptions have been shown to be broken but nothing directly saying HPN is broken

There is a Public Key version of this construction but it uses obfuscation and we don't like it.

What can we do with SK-DEPIR?

- Private Anonymous Data Access

(PANDA)

- Rewindable Oram*
- Two-server DORAM for secure computation