Doubly Efficient Private Information Retrieval

Model

\[
\begin{array}{c}
\text{Client} \\
q = \text{query}(i) \\
\end{array} \quad \xrightarrow{\text{q}} \quad \begin{array}{c}
\text{Server} \\
r \xleftarrow{} \text{dec}(r) \\
\end{array}
\]

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 achieve information theoretic 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 & \( T \) lower bound that
if no preprocessing is allowed work for server must be linear \[ \text{[Bim00]} \]

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

Goal: if we allow preprocessing can we achieve 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 DEPR: (secret-key)
\[(\text{KeyGen}, \text{Process}, \text{Query}, \text{Resp}, \text{Dec})\]
\[
- \text{KeyGen}(k^*), k^* \rightarrow \text{sk}
- \text{Process}(k, \text{DB}) \rightarrow \tilde{\text{DB}}
- \text{Query}(k, \text{i}) \rightarrow q, \text{state}
- \text{Resp}(\tilde{\text{DB}}, q) \rightarrow e
- \text{Dec}(k, \text{state}, c) \rightarrow \text{DB}[i] (y)
\]

- Correctness
\[
\Pr \left[ \exists \text{DB}, \text{i} \in \mathbb{N} \left( k^* \leftarrow \text{KeyGen}(\cdot) \right. \right.
\left. \left. \text{Dec}(k, \text{sk}, c) = \text{DB}[i] : \tilde{\text{DB}} \leftarrow \text{Proc}(k, \text{DB}) \quad (q, \text{state}) \leftarrow \text{Query}(k, \text{i}) \quad e \leftarrow \text{Resp}(\tilde{\text{DB}}, q) \right) \right) \right] = 1
\]

- Efficiency:
- KeyGen: \(O(n)\)
- Process: \(\text{poly}(N, z)\)
- Query, Dec: \(O(N) \cdot \text{poly}(z)\)

- Security:
\[
\frac{C}{A}
\]
Construction: The long winding road of code theory

- There is a thing called error correcting codes - they are used in many places.
- We are interested in a variant called locally decodable codes at a high level LDCs.

\[ b \in \{0, 1\} \]
\[ K = K_{6,1}^{(3)} \]
\[ \tilde{DB} = \text{Process}(K, DB) \]
\[ i_0, i_1 \in \{0\} \]
\[ q_b \leftarrow \text{Query}(K, i_0) \]
\[ i_0, i_1 \]
\[ \tilde{q_b} \]
\[ \text{repeat poly times} \]
\[ b' \]
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 from $\text{Gen} \{1^x\}$ $t$-smooth LDC parameters also choose $\mathcal{L}(k \text{-indices for query})$ $k$ permutations over encoded DB size $\mathcal{E}_M$
\[ Sk = (\pi_1, \ldots, \pi_k) \]

**Process** \((sk, DB)\):  
\[ \hat{DB} = \text{loc.enc}(DB) \]
\[ \hat{DB}_{i,j} = \pi_i(\hat{DB}) \]
output \((\hat{DB}, \ldots, \hat{DB}_k)\)

**Query** \((sk, i)\):  
\[ j_1, \ldots, j_k = \text{loc.query}(i) \]
output \((\pi_{j_1}(DB), \ldots, \pi_{j_k}(DB))\)

**Resp** \((j_1, \ldots, j_k, DB, \ldots, \hat{DB}_k)\):
output \(\hat{DB}_{j_1} \ldots \hat{DB}_{j_k}\)

**Dec** \((y_1, \ldots, y_k)\): output \(\text{loc.dec}(y_1, \ldots, y_k)\)

So this scheme is information-theoretic security to a bound \(B\) 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/Noises):**

\[ m < t < r < k < 2^k \]

\[ \mathbf{\epsilon} \in T \]

Note: only one permutation

---

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
- Rewindable Oram
- Two-server DORAM for secure computation