Cryptography Meets Algorithms (15893) Lecture Notes

Lecture 7: Preprocessing PIR Lower Bounds

Scribe: William Seo

February 27, 2024

In this lecture, we will prove a lower bound about the client space and server computation tradeoff for preprocessing PIR schemes. We will borrow techniques for proving time-space tradeoff from the complexity theory literature.

Specifically, we consider a 1-server preprocessing PIR scheme with the following syntax:

- *Preprocessing algorithm.* Suppose there is a (possibly randomized and unbounded) preprocessing function denoted $\mathsf{Prep} : \{0,1\}^n \to \{0,1\}^S$ that takes in an n-bit database $\mathsf{DB} \in \{0,1\}^n$ as input, and outputs an S-bit hint string denoted h.
- *A single query.* The client and the server perform a (possibly randomized) query protocol. The client takes in h and some index $i \in [n]$ as input, and the server takes the database $\mathsf{DB} \in \{0,1\}^n$ as input. To answer the query, the server is allowed to read at most T locations of the database.

In other words, we do not place any restriction on the amount of work performed during preprocessing. The only constraints are that at the end of the preprocessing: 1) the client is allowed to store only the hint h; and 2) the server is allowed to store only the original database DB and no extra information.

We assume perfect correctness, i.e., for any $\mathsf{DB} \in \{0,1\}^n$, any query $i \in [n]$, correctness holds with probability 1. Let $\mathsf{view}_S(\mathsf{DB}, i)$ denote the server's view during the query phase when we run the PIR scheme (i.e., preprocessing followed by the query protocol) over inputs DB and i. For security, we require that for any $\mathsf{DB} \in \{0,1\}^n$, any $i, j \in [n]$, $\mathsf{view}_S(\mathsf{DB}, i) \approx \mathsf{view}_S(\mathsf{DB}, j)$ where $\approx$ means computational indistinguishability.

For such a preprocessing PIR scheme, we will prove a tradeoff between S and T as stated in the following theorem:

Theorem 1 (Time space tradeoff for preprocessing PIR [CGK20])**.** *Given a 1-server preprocessing PIR, let S be the client space, and let T be the server compuation per query. Then, $(S+1)(T+1) \geq N$. [elaine: TODO: edit the theorem based on what we can prove later]*

Again, note that the lower bound holds even when we allow the preprocessing to be unbounded, even when there is only a single query after the preprocessing, and even when the query phase may have arbitrarily many rounds of interaction.

Piano. Recall that in an earlier lecture, we covered Piano [ZPSZ23], a preprocessing 1-server PIR scheme. Piano enjoys the following performance bounds:

- Client space: $\widetilde{O}(\sqrt{n})$ where $\widetilde{O}(\cdot)$ hides a(n arbitrarily small) superlogarithmic function.

- Communication per query: $O(\sqrt{n})$
- Server computation per query: $O(\sqrt{n})$

We can see that Piano achieves optimal client space and server computation tradeoff (up to polylogarithmic factors) in light of Theorem 1.

1 Warmup: Yao's Box Problem

Before proving Theorem 1, we first prove a time-space tradeoff for a classical problem called the Yao's box problem [Yao90], and we shall see why Yao's box problem is closely related to preprocessing PIR.

We have a server with n boxes, each covering a bit. Henceforth we use $\mathsf{DB} := (\mathsf{DB}[1], \ldots, \mathsf{DB}[n])$ to denote the n bits. Consider the following game:

- *Preprocessing.* During a preprocessing phase, the server and client can perform an unbounded amount of computation. At the end of preprocessing, the client obtains an S-bit hint h; the server does not store any extra information besides DB itself where each bit is covered by some box.
- *Query.* The client wants to find out $\mathsf{DB}[i]$ for some index $i \in [n]$. The client and server now engage in some protocol at the end of which the client outputs an answer β. The query protocol must satisfy two requirements: 1) the server is *not allowed to open box i* during the protocol; and 2) the server can open at most T boxes.

We allow the preprocessing algorithm and query protocol to be possibly randomized. We require that the protocol to have *perfect correctness*, i.e., for any $\mathsf{DB} \in \{0,1\}^n$, any index $i \in [n]$, after performing the preprocessing and the query protocol for index i, the client's output $\beta = \mathsf{DB}[i]$ with probability 1.

Note that unlike PIR, Yao's box problem does not have any privacy requirement. In particular, it is perfectly ok for the query index i to be leaked to the server.

In the above formulation, S denotes the client space at the end of preprocessing, and T can be viewed as a lower bound on the server's running time during the query phase. We care about characterizing the tradeoff between S and T, that is, the client-space and server-time tradeoff.

Yao [Yao90] proved the following theorem:

Theorem 2 (Yao's box problem)**.** $S \cdot T \geq \Omega(n)$. *[elaine: TODO: modify the statement based on proof]*

1.1 Upper Bound

Before proving Theorem 2, let us first see a simple upper bound that can match the $S \cdot T = \Theta(n)$ tradeoff. For simplicity, assume n is a perfect square.

1. Divide the n boxes into $\sqrt{n}$ segments each of size $\sqrt{n}$.
2. Preprocessing: the client stores the parity of each segment, denoted $p_1, \ldots, p_{\sqrt{n}}$ respectively.
3. Query for index $i \in [n]$: suppose i belongs to the j-th segment. The server opens every other box except i in the j-th segment, and responds with the parity (denoted p^*) of all opened bits. The client reconstructs the answer as $p^* \oplus p_j$.

In this construction, $S = \sqrt{n}$ and $T = \sqrt{n} - 1$.

You might have observed that this upper bound for Yao's box problem is reminiscient of the Piano PIR scheme [ZPSZ23].

1.2 Lower Bound

We now prove Theorem 2. The intuition of the proof is if we can obtain a too-good-to-be-true tradeoff between S and T, then we can construct an encoding of $\mathsf{DB} \in \{0,1\}^n$ whose length is less than n, which violates Shannon's fundamental theorem of information theory. Specifically, we can obtain some compression in the encoding by leveraging the following fact: every time we open T boxes not including i, we learn not just the values under the T opened boxes but also an additional bit, namely, the i-th bit — this effectively gains us one bit of advantage.

The formal proof works as below.

Proof of Theorem 2. Suppose we have some protocol for solving Yao's box problem with parameters S and T. We fix all the coins in the protocol denoted coins, and we will give coins as input to both the encoder and decoder. As long as DB is randomly sampled independently of coins, we can use Shannon's theorem to argue that the encoding length must be at least n.

Encoding. The encoding algorithm is given coins and DB as input, and constructs an encoding of DB as follows.

1. Seed the protocol (for solving Yao's box problem) with coins, and perform preprocessing such that the client obtains an S-bit hint h.
2. Initially, let $\mathsf{known} = \{\}$. Repeat the following: in each time step i, the client
   - finds the smallest index $q_i \notin \mathsf{known}$, and runs the query protocol for index q_i;
   - let $\mathsf{known} \leftarrow \mathsf{known} \cup \{q_i\} \cup \{\text{all boxes opened during this query}\}$
   - if $\mathsf{known} = [n]$, break.

Define the *newly opened* boxes in time step i as follows: all boxes opened in time step i that are not in the known set yet. Output an encoding containing the following terms:

1. The client's hint h;
2. Let Q_i be the newly opened values (in the order they are opened) for time step i. Include $Q_1, Q_2, \ldots, Q_k$ in the encoding, where k is the total number of time steps.

Decoding. The decoding algorithm is given coins and an encoding C as input, and outputs a decoded string as follows.

1. Seed the protocol (for solving Yao's box problem) with coins. Treat the first S bits of C as the client's hint, and in the steps below, consume the rest of C bit by bit in a streaming manner.
2. Initially, let $\mathsf{known} = \emptyset$. Every time step i,
   - Let q_i be the smallest index not in known;

- The client performs a query for q_i with the server. Whenever the server needs to open some box: if the box was already opened in some earlier time step, use the same opened value as previously learned; else if the box has not been opened, treat the next bit in C as the opened value.
- For all newly opened boxes, record the opened values. Additionally, reconstruct the value at queried index q_i.
- Let $\mathsf{known} \leftarrow \mathsf{known} \cup \{q_i\} \cup \{\text{all boxes opened during this query}\}$; if $\mathsf{known} = [n]$, break.

3. At the end of the protocol, $\mathsf{known} = [n]$, i.e., n indices have been reconstructed. Output this reconstructed string.

□

Theorem 3. $S(T+1) \geq N$

Proof. We will be using an encoding type argument. Consider the following experiment.

1. Run preprocessing
2. Define an empty set called $\mathsf{known} = \{\}$. In each step i, the client finds the smallest $q_i \notin \mathsf{known}$, queries q_i.

   With this step, $\mathsf{known} \Longleftarrow \mathsf{known} \cup \{q_i\} \cup$ all boxes opened during query
3. If $\mathsf{known} \neq [N]$, break. Otherwise, repeat step 2 again.

Let the "client's hint" denote whatever information that the client stores after the preprocessing phase. The hint is at most S bits long.

Define the encoding enc for this process as follows:

$$\mathsf{enc} = \text{client's hint} + \text{all "newly opened" boxes in all queries}$$

We write "newly opened" because we do not want to include values that have already been recorded in the encoding.

By Shannon's theorem, we will show that $|\mathsf{enc}| \geq N$.

Let $t_1, t_2, \ldots t_k$ be the number of newly opened boxes at each step $i \in [k]$.

Note here that we have the power of **plus one**; if I open t boxes, I end up learning $t+1$ new bits. This is because I also learn the value of the query without opening its box.

Thus, the known set increments with the following pattern.

- We newly open t_1 boxes: known increments by $t_1 + 1$
- We newly open t_2 boxes: known increments by $t_2 + 1$
- and so on...

Thus, we have that $\sum_{i=1}^{k}(t_i+1) \geq N$. Let $t = \frac{\sum_{i=1}^{k} t_i}{k}$. Note that t is the average number of boxes opened on each iteration so it must be upper bounded by T, the maximum number of boxes that can be opened in an iteration. Since we repeat the steps until $\mathsf{known} = [N]$, we have that

$$\sum_{i=1}^{k}(t_i+1) \geq N \implies (t+1)k \geq N \implies k \geq \frac{N}{t+1} \tag{1}$$

The encoding size is $S + \sum_{i=1}^{k} t_k = S + tk$. By Shannon's we have that $S + tk \geq N$. By (1), we have that

$$\begin{aligned} & S + tk \geq N \\ & \implies S + t\frac{N}{t+1} \geq N && \text{(By (1))} \\ & \implies S(t+1) \geq N && \text{(Simplification)} \\ & \implies S(T+1) \geq N && \text{(As } t \leq T\text{)} \end{aligned}$$

□

2 PIR Lower Bound

Using our knowledge of Yao's Box Problem, let's know try to prove the PIR lower bound.

Theorem 4. *Suppose we have a 1 server prepossessing PIR with perfect correctness and negl(n) privacy loss with client space S and server computation T per query. Then,*

$$(S+1)(T+1) \geq N \quad [CGK20]$$

This lower bound holds even for computationally private schemes. It holds even for a single query, regardless of bandwidth, server space, even when preprocessing can be unbounded. However, we have a restriction that the server stores the original database and nothing else. That is, it does not store any encoding of the database.

For the proof, we will show that a solution to the PIR problem can be used to construct an algorithm to solve a probabilistic version of Yao's Box Problem.

Probabilistic Yao's Box Problem Suppose we have a working PIR scheme. Now, we will construct a solution to probabilistic Yao's Box Problem as follows:

- Client's Hint: PIR's hint
- Query for $i \in [N]$: Run PIR for query i. If server looks at $\mathsf{DB}[i]$, then output "error".

We want to show the following. Given that PIRExpt: $i \xleftarrow{\$} [N]$, PIR preprocessing, PIR query on i,

$$p = \mathbb{P}[\text{PIRExpt opens } i] \leq \frac{T}{N} + \mathsf{negl}(N)$$

That is, we want to show that if we run a PIR query with the a random index i, the probability we open that index is small.

For a fixed i, define the following probability $p_i = \mathbb{P}[\text{PIR on } i \text{ looks at } i]$. Then,

$$p = \frac{1}{N}\sum_{i} p^i$$

Assume for the sake of contradiction that $p > \frac{T}{N} + \mu$, where μ is non-negligible.

Let $p_{ji} = \mathbb{P}[\text{PIR on } j \text{ opens } i]$. Since our scheme is private, the different indices should be computationally indistinguishable. Thus, this probability should be equally distributed. As a result,

$$p_{ji} = \mathbb{P}[\text{PIR on } j \text{ opens } i] \geq p_i - \mathsf{negl}(N)$$

$$\begin{aligned} E[\text{server work for PIR on } j] &\geq \sum_{i=1}^{N} p_{ji} \\ &\geq \sum_{i=1}^{N} (p_i - \mathsf{negl}(N)) \\ &= Np - \mathsf{negl}(N) && \text{(By def. of } p\text{)} \\ &> T + \mu N - \mathsf{negl}(N) && \text{(As } p > \tfrac{T}{N} + \mu\text{)} \end{aligned}$$

Thus, we have a contradiction because the expected number of locations the server needs to look at is strictly greater than T. Thus, we have shown that

$$\mathbb{P}[\text{PIRExpt opens } i] \leq \frac{T}{N} + \mathsf{negl}(N)$$

Shifting our focus back to Yao's Box problem with probabilistic correctness on random index, the probabilistic correctness is

$$\mathbb{P}[i \xleftarrow{\$} \text{correct for } i] \geq 1 - \frac{T}{N} - \mathsf{negl}(N)$$

Encoding Argument Randomness comes from two parts: the preprocessing part (the client's hint), and the query part. With this in mind, we will be using an augmented version of the encoding type argument in **??**.

Consider the following experiment.

1. Run preprocessing, and choose a "reasonably good hint." Initially, let the encoding be just the hint. We will add to this encoding as we go forward.
2. Define an empty set called $\mathsf{known} = \{\}$
3. In each step i, find the smallest $q_i \notin \mathsf{known}$.
   - If $\exists$ online coins such that query q_i will give the correct answer, choose the lexicographically smallest coin. Execute query.

     $$\mathsf{known} \impliedby \cup\{q_i\} \cup \{\text{all newly opened}\}$$

     Add "newly opened" to encoding.
   - Else add q_i-th bit to the encoding.
4. Repeat until $\mathsf{known} = [N]$.

A hint is bad for $i \in [N]$ if $\mathbb{P}[\text{query } i \text{ correct}|\text{hint}] = 0$. That is, there does not exist an online coin such that the query is correct.

Claim 5. $\exists$ *hint that's bad for at most* $T+1$ *location.*

Proof. If all hints are bad for more than $T+1$ locations, then

$$\mathbb{P}[i \in [N], \text{correct on } i] < 1 - \frac{T+1}{N}$$

We have a contradiction, because it disagrees with our probabilistic correctness result above. □

Finally, we can reason about the encoding length.

Suppose the worst case where the hint is bad for exactly $T+1$ locations. Let $\mathsf{b}_{\mathsf{bad}}$ be the encoding of the "newly opened" boxes in the bad queries, and $\mathsf{b}_{\mathsf{good}}$ be the encoding of the "newly opened boxes in the good queries.

- $|\mathsf{b}_{\mathsf{bad}}| = T+1$, as each bad iteration adds one bit, and we have $T+1$ iterations.
- To find $|\mathsf{b}_{\mathsf{good}}|$, we repeat the argument from **??**. Let $t_1, t_2, \ldots t_k$ be the number of newly opened boxes at each good step $i \in [k]$.

  By the "plus one" argument, we have that $\sum_{i=1}^{k}(t_i+1) \geq N - (T+1)$. We subtract $T+1$ here because those indices have been handled by the bad iterations.

  Let $t = \frac{\sum_{i=1}^{k} t_i}{k}$. Then as before, we have $k \geq \frac{N}{t+1}$

$$\begin{aligned} \sum_{i=1}^{k}(t_i+1) &\geq N - (T+1) \\ \implies tk + k &\geq N - T - 1 \\ \implies k &\geq \frac{N-T-1}{t+1} \end{aligned}$$

  Thus, $|\mathsf{b}_{\mathsf{good}}| = tk \geq t\frac{N-T-1}{t+1}$

With the information above, we can mek the following conclusion.

$$\begin{aligned} |\mathsf{enc}| = S + |\mathsf{b}_{\mathsf{good}}| + |\mathsf{b}_{\mathsf{bad}}| &\geq N \\ \implies S + t\frac{N-T-1}{t+1} + T + 1 &\geq N \\ \implies S(t+1) + T + 1 &\geq N && \text{(By simplification)} \\ \implies (S+1)(T+1) &\geq N && \text{(As } t \text{ upper bounded by } T\text{)} \end{aligned}$$

□

References

[CGK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In *Advances in Cryptology – EUROCRYPT 2020*, pages 44–75, Cham, 2020. Springer International Publishing.

[Yao90] A. C.-C. Yao. Coherent functions and program checkers. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 84–94, New York, NY, USA, 1990. Association for Computing Machinery.

[ZPSZ23] Mingxun Zhou, Andrew Park, Elaine Shi, and Wenting Zheng. Piano: Extremely simple, single-server pir with sublinear server computation. Cryptology ePrint Archive, Paper 2023/452, 2023. `https://eprint.iacr.org/2023/452`.