Cryptography Meets Algorithms (15893) Lecture Notes

Lecture 1: Private Information Retrieval

Scribe: Justin Zhang

March 23, 2024

The Private Information Retrieval problem was first introduced by Chor, Kushilevitz, Goldreich and Sudan [**?**]. In this setting, we will have a client and server(s). The server(s) are connected to a public database indexed from 1 to n, which could represent DNS, webpage, leaked password database, etc.

A client wants to fetch an entry from this database $i \in [n]$ but does not want to leak their query to the server(s). More formally,

Definition 1 (PIR - Single Server Scheme)**.** *A protocol between a client c and server s where,*

- *Client has desired index $i \in [n]$, and server(s) have $\mathsf{DB} \in \{0,1\}^n$*
- *At the end of the protocol, client outputs bit $b \in \{0,1\}$*

with properties,

- ***Correctness**: $\forall n : \mathsf{DB} \in \{0,1\}^n, i \in [n]$, under honest execution,*

$$\Pr[b = \mathsf{DB}[i]] \geq 1 - \mu(n) \qquad \text{(negligible } \mu\text{)}$$

- ***Privacy**: $\forall i, j \in [n], \mathsf{DB} \in \{0,1\}^n$, and function $view_s$ representing the view of server s:*

$$view_s(n, \mathsf{DB}, i) \cong view_s(n, \mathsf{DB}, j)$$

where $\cong$ stands for identically distributed, statistically, or computationally indistinguishable.

Remark 1 (Honest-server vs. malicious-server privacy)**.** *The above privacy definition assumes an honest server. It is also possible to define privacy against a malicious server. In today's lecture, all the PIR constructions will only have a single round-trip — in this special case, honest-server privacy and malicious-server privacy are equivalent. So we will simply define honest-server privacy here.*

This definition is naturally extended to a two (and any $n \geq 2$) server definition, where the two servers do not communicate.

Definition 2 (PIR - Two Server Scheme)**.** *A protocol between a client c and servers s_1, s_2 where,*

- *Client has desired index $i \in [n]$, and server(s) have $\mathsf{DB} \in \{0,1\}^n$*
- *At the end of the protocol, client outputs bit $b \in \{0,1\}$*

with properties,

- ***Correctness:*** $\forall n : \mathsf{DB} \in \{0,1\}^n, i \in [n]$, *under honest execution,*

$$\Pr[b = \mathsf{DB}[i]] \geq 1 - \mu(n) \qquad \text{(negligible } \mu\text{)}$$

- ***Privacy:*** $\forall i, j \in [n], \mathsf{DB} \in \{0,1\}^n$, *and function* $view_s$ *representing the view of server* s*:*

$$view_{s_1}(n, \mathsf{DB}, i) \cong view_{s_1}(n, \mathsf{DB}, j)$$

$$view_{s_2}(n, \mathsf{DB}, i) \cong view_{s_2}(n, \mathsf{DB}, j)$$

Note that PIR schemes can additionally be extended to retrieve blocks of data by querying the database bit-by-bit.

1 Single Server PIR constructions

Naively, the client can just download the entire database, which is perfect correctness and privacy, but this has linear bandwidth, client computation, and server computation with respect to the database size.

1.1 FHE PIR

For a smarter and overall more efficient scheme, we assume we have an FHE scheme (Gen, Enc, Dec). An FHE scheme allows us to perform addition and multiplication operations on the ciphers which propagate to their underlying plain-texts. [1]

The scheme is as follows:

1. The client samples $(pk, sk) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$, and encrypts their query as $m \leftarrow \mathsf{FHE.Enc}(i)$. The client then sends m to the server.

2. The server homomorphically evaluates the selection circuit S and let $c = \mathsf{Eval}(S, m)$ ($S(i)$ selects the i-th bit from DB and the circuit is $\tilde{O}(n)$[2] in size), and sends it to the client.

3. The client decrypts $\mathsf{FHE.Dec}(sk, c)$.

The correctness of the scheme is clear. The scheme has $\tilde{O}(1)$ bandwidth and client computation and $\tilde{O}(n)$ server computation. Some notable PIR schemes include Spiral [**?**] and SimplePIR [**?**].

Question: Can we get sub-linear bandwidth without any cryptographic (hardness) assumptions? In fact, this is possible in the two server setting (we will see this next), and we can prove that this is impossible in the one server case.

2 Two Server PIR Constructions

2.1 $\sqrt{n}$-BW 2-Server-Scheme with Information-theoretic (IT) security

Note that IT security implies no harness assumptions, so this is perfect and statistically private.

The key idea is the view of database $DB \in \{0,1\}^n$ as a $S \in \{0,1\}^{\sqrt{n} \times \sqrt{n}}$ matrix. Say the client wants to query the (i, j)'th entry. To do this, the server can simply return the entire column j, which we can afford to do via the square-root bandwidth. To provide security, the client also supplies a one-hot-vector q_j, split into two random shares v_1, v_2 via a 2-share secret sharing scheme (that is, sample random v_1, v_2 conditioned on $v_1 \oplus v_2 = q_j$).

The scheme is as follows:

[1] TA: Arithmetic circuit (including addition and multiplication gates) is Turing-complete. Thus, FHE allows the server to evaluate arbitrary computation given the encrypted input. However, we have to be careful about its efficiency – we usually discuss the time complexity of an algorithm in the RAM model, whereas FHE only works in the circuit model (in general).

[2] $\tilde{O}$ hides the polylogarithmic factors.

1. The client creates v_1, v_2 as described above. The client sends these shares to servers s_1, s_2 respectively.

2. Each server s_i, on receiving v_i computes the matrix-vector product,

$$r_i \leftarrow Sv_i \bmod 2$$

and sends the $\sqrt{n}$-size vector r_i to the client.

3. The client computes and outputs $r_1 \oplus r_2$.

It is straightforward to verify correctness by seeing that

$$r_1 \oplus r_2 = Sr_1 \oplus Sr_2 \bmod 2 = S(r_1 \oplus r_2) \bmod 2$$

Lastly, this scheme is private, since v_1, v_2 are uniformly random in each server's view, respectively.

2.2 $n^{\frac{1}{3}}$-BW 2-server Scheme[?]

We will first motivate this construction with a two server scheme with expected $\frac{n}{2}$ bandwidth and a eight server scheme with $n^{\frac{1}{3}}$ bandwidth.

1. **$\mathbb{E}[\frac{n}{2}]$-BW 2-server Scheme**

   The client samples $S_1 \subseteq [n]$ uniformly as follows: for each $i \in [n]$, add i to S_1 with probability $\frac{1}{2}$. Then, the client computes

$$S_2 = S_1 \Delta \{i\}$$

   where Δ is the symmetric difference operator. That is, if i is included in S_1, we remove it from the set, otherwise we add it to the set. S_1, S_2 are sent to each server respectively, where server i computes and sends back,

$$r_i \leftarrow \bigoplus_{j \in S_i} \mathsf{DB}[j]$$

   The client on receiving r_1, r_2 outputs $r_1 \oplus r_2$.

   Correctness follows from the fact that i is the only database index that appears once, with every other index appearing twice (hence XOR'ing to 0).

   Privacy: First, S_1 is uniformly random. Second, to see why S_2 is uniformly random to server 2, just consider the following distribution – "tossing n random coins, and flip the i-th coin afterwards, regardless of the original result". This distribution is uniformly random even when i is known.

   Remark 2. *Note that if we run the above $n/2$-BW scheme not on bits, but on blocks of $\sqrt{n}$ size (i.e., treat the n-bit database as $\sqrt{n}$ blocks each of size $\sqrt{n}$), the scheme is equivalent to the earlier $\sqrt{n}$-BW scheme.*

2. **$n^{\frac{1}{3}}$−BW 8-server Scheme**

   The idea is to view the database as a $n^{\frac{1}{3}} \times n^{\frac{1}{3}} \times n^{\frac{1}{3}}$ cube. Then, each index $i \in \{0, 1, \ldots, n-1\}$ can be expressed as a thruple $(x^\star, y^\star, z^\star)$ (note we start at 0 for natural base 3 representation.

   The client samples $X, Y, Z \subseteq \{0, \ldots, n^{\frac{1}{3}} - 1\}$ independently as follows: for each $x \in \{0, \ldots, n^{\frac{1}{3}}\}$ add it to X with probability $\frac{1}{2}$. Do the same for Y, Z.

Then, we compute 8 different sets by a 3-wise cartesian products with symmetric differences, enumerated as

$$\begin{aligned}
S_{000} &= X \times Y \times Z \\
S_{001} &= X \times Y \times (Z \Delta \{z^\star\}) \\
S_{010} &= X \times (Y \Delta \{y^\star\}) \times Z \\
S_{011} &= X \times (Y \Delta \{y^\star\}) \times (Z \Delta \{z^\star\}) \\
S_{100} &= (X \Delta \{x^\star\}) \times Y \times Z \\
S_{101} &= (X \Delta \{x^\star\}) \times Y \times (Z \Delta \{z^\star\}) \\
S_{110} &= (X \Delta \{x^\star\}) \times (Y \Delta \{y^\star\}) \times Z \\
S_{111} &= (X \Delta \{x^\star\}) \times (Y \Delta \{y^\star\}) \times (Z \Delta \{z^\star\})
\end{aligned}$$

See that each of these sets have their sizes concentrated around $\left(\frac{n^{\frac{1}{3}}}{2}\right)^3 = \frac{n}{8}$. Then, the client sends a succinct description of each set (i.e., sends the three "marginal" vectors instead of the full set) of size $O(n^{\frac{1}{3}})$ to each server.

Server i on receiving $S = X' \times Y' \times Z'$ computes

$$p_i = \bigoplus_{j \in S} \mathsf{DB}[j]$$

Then the client computes $p_0 \oplus \cdots \oplus p_7$.

We claim that $p_0 \oplus \cdots \oplus p_7 = \mathsf{DB}[i], i = (x^\star, y^\star, z^\star)$.

Proof. For every (x, y, z) not equal to the query, it will appear an even number times in the summation. We can pair up the sets to see this.

On the other hand, $(x^\star, y^\star, z^\star)$ only appears once in the summation so we are done. □

Privacy follows the argument as the previous case.

Now, we finally compress this scheme from eight servers to two servers. To do this, the client sends S_{000} to server 1 and S_{111} to server 2.

Each server on receiving $X' \times Y' \times Z'$ calculates $3n^{\frac{1}{3}}$ parities to send to the client as follows:

1. For each $x' \in \{0, \ldots, n^{\frac{1}{3}} - 1\}$ we calculate the parity for $(X' \Delta \{x'\}) \times Y' \times Z'$ as in the previous scheme.
2. For each $y' \in \{0, \ldots, n^{\frac{1}{3}} - 1\}$ we calculate the parity for $X' \times (Y' \Delta \{y'\}) \times Z'$ as in the previous scheme.
3. For each $z' \in \{0, \ldots, n^{\frac{1}{3}} - 1\}$ we calculate the parity for $X' \times Y' \times (Z' \Delta \{z'\})$ as in the previous scheme.

Now each server returns $1 + 3n^{1/3}$ parities to the client. That is, the server 1 will actually compute S_{000} and $S_{100}, S_{010}, S_{001}$ will be in those $3n^{1/3}$ parities. Similarly, the server 2 will compute S_{111}, and $S_{011}, S_{101}, S_{110}$ will be in those $3n^{1/3}$ parities. The client will be able to pick out the correct parities corresponding to its actual query.

Thus, this scheme still has $n^{\frac{1}{3}}$ bandwidth, and correctness and security still follow.

3 State of the Art and Open Problem

For the 2-server setting, the best known lower bound states that $5 - o(1) \log n$ bandwidth is necessary ([**?**]), while the best known upper bound requires $n^{O\left(\sqrt{\lg \lg n / \lg n}\right)} = n^{o(1)}$, i.e., sub-polynomial bandwidth ([**?**]). Closing this gap is a long-standing open problem.