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 [CKGS98]. In this setting, we will have a client and one or more server(s). The servers each have a public database indexed from 1 to n (e.g., the DNS repository, a repository of webpages, a leaked password database, etc).

A client wants to fetch an entry indexed $i \in [n]$ from this database but does not want to leak its query to the server(s). More formally, we define a single-server PIR scheme as follows.

Definition 1 (Single-server PIR). A single-server PIR, parametrized by a security parameter $\lambda \in \mathbb{N}$, is a protocol between a client and a server with the following syntax:

- The client's input is a desired index $i \in [n]$, and the server's input is a database $\mathsf{DB} \in \{0,1\}^n$. Both the client and server also obtain 1^λ as input.

- At the end of the protocol, the client outputs a bit $b \in \{0,1\}$.

We want the scheme to satisfy the following properties.

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

$$\Pr[b = \mathsf{DB}[i]] = 1$$

- **Privacy**: For any λ, any n polynomially bounded in λ, any $i, j \in [n], \mathsf{DB} \in \{0,1\}^n$, it holds that

$$\mathsf{view}_S(1^\lambda, \mathsf{DB}, i) \approx \mathsf{view}_S(1^\lambda, \mathsf{DB}, j)$$

where $\mathsf{view}_S(1^\lambda, \mathsf{DB}, i)$ is a random variable representing the view of the server if we execute the PIR protocol over client input $(1^\lambda, i)$ and server input $(1^\lambda, \mathsf{DB})$, and $\approx$ stands for statistical or computational indistinguishability.

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 setting with two or more servers that do not communicate, where privacy should hold for any individual server's view. In a setting with more than two servers, it also makes sense to define t-out-of-n security, where we want privacy to hold for the union of any combination of t servers' views. For example, a 2-server PIR scheme is defined as follows.

Definition 2 (Two-server PIR). A two-server PIR, parametrized with some security parameter λ, is a protocol between a client and two servers with the following syntax:

- The client's input is 1^λ and a desired index $i \in [n]$, and each server's input is a database $\mathsf{DB} \in \{0,1\}^n$.

- At the end of the protocol, the client outputs a bit $b \in \{0,1\}$.

with properties,

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

$$\Pr[b = \mathsf{DB}[i]] = 1$$

- **Privacy**: for any λ, any n that is polynomially bounded in λ, any $i, j \in [n]$, any $\mathsf{DB} \in \{0,1\}^n$, it holds that

$$\mathsf{view}_1(1^\lambda, \mathsf{DB}, i) \approx \mathsf{view}_1(1^\lambda, \mathsf{DB}, j)$$

$$\mathsf{view}_2(1^\lambda, \mathsf{DB}, i) \approx \mathsf{view}_2(1^\lambda, \mathsf{DB}, j)$$

where $\mathsf{view}_1(1^\lambda, \mathsf{DB}, i)$ and $\mathsf{view}_2(1^\lambda, \mathsf{DB}, i)$ are random variables representing the view of the first and the second server, respectively, in a protocol execution with client input $(1^\lambda, i)$ and server input $(1^\lambda, \mathsf{DB})$.

Note that PIR schemes can be extended for retrieving records containing multiple bits, rather than just 1-bit records.

Naïve approach. The naïve approach is for the client to download the entire database. However, this approach suffers from linear bandwidth, and linear server/client computation.

We will now show some PIR constructions with sublinear bandwidth.

1   Single-Server PIR Based on Fully Homorphic Encryption

It is easy to obtain a bandwidth-efficient PIR scheme if we assume a Fully Homomorphic Encryption (FHE) scheme. An FHE scheme allows us to perform addition and multiplication operations in the ciphertext space. An FHE scheme supports the following operations:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$: samples a public key pk and a secret key sk;

- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$: encrypts a message m from some message space using the public key pk, and outputs the ciphertext c;

- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$: decrypts a ciphertext c using the secret key sk, and outputs a plaintext message m;

- $c' \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{Circ}, c)$: given the public key pk, some circuit Circ, and a ciphertext c, output a transformed ciphertext c'. Correctness requires that c' be a valid FHE encryption of $\mathsf{Circ}(c)$.

PIR from FHE. We can construct a single-server PIR scheme from an FHE scheme as follows.

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

2. The server homomorphically evaluates the selection circuit S_{DB} by calling $c \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, S_{\mathsf{DB}}, q)$, where $S_{\mathsf{DB}}(i)$ is the circuit that selects the i-th bit from the input DB, and sends the resulting ciphertext c back to the client.

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

The correctness of the PIR scheme is easy to see given correctness of the FHE scheme. The scheme has $\widetilde{O}(1)$ bandwidth and client computation, and $\widetilde{O}(n)$ server computation[1] where $\widetilde{O}(\cdot)$ hides $\mathsf{poly}(\lambda, \log n)$ factors. Note that the server computation is at least linear because the selection circuit $S_{\mathsf{DB}}(\cdot)$ must encode the entire database.

Many recent works optimized FHE-based PIR schemes, e.g., Spiral [MW22] and SimplePIR [HHCG+22].

Question: Can we get sub-linear bandwidth PIR without any cryptographic assumptions?

In fact, this is possible in the two-server setting (we will see this next), and later we will prove that it is impossible in the single-server setting.

2 Two-Server PIR Constructions

2.1 $\sqrt{n}$-Bandwidth 2-Server PIR

We now introduce a 2-server $\sqrt{n}$-bandwidth PIR scheme with information theoretic security, i.e., the scheme does not rely on any cryptographic assumptions.

The key idea is to view the database $\mathsf{DB} \in \{0,1\}^n$ as a $\sqrt{n}$ by $\sqrt{n}$ matrix[2] henceforth denoted $M \in \{0,1\}^{\sqrt{n} \times \sqrt{n}}$. Say the client wants to query the bit $M[i, j]$. To achieve this, the client will perform a 2-server PIR protocol at the end of which it retrieves the entire column $M[:, j]$. The scheme is as follows:

1. The client samples random column vectors $v_1, v_2 \in \{0,1\}^{\sqrt{n}}$ such that $v_1 \oplus v_2 = e_j$ where e_j is the vector that is 1 at position j, and 0 everywhere else, and $\oplus$ denotes bitwise XOR (i.e., addition mod 2). The client sends v_1 and v_2 to the two servers, respectively.

2. For $i \in \{1, 2\}$, server i computes the following matrix-vector product on receiving v_i:

$$r_i \leftarrow M v_i \bmod 2,$$

and sends the $\sqrt{n}$-sized 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

$$r_1 \oplus r_2 = M v_1 + M v_2 \bmod 2 = M(v_1 + v_2) \bmod 2 = M \cdot e_j$$

Lastly, this scheme is private, since each individual vector v_1 or v_2 is uniformly random.

2.2 $n^{\frac{1}{3}}$-Bandwith 2-Server PIR

Chor et al. [CKGS98] showed how to get an information-theoretic 2-Server PIR scheme with $O(n^{1/3})$ bandwidth in expectation. To get this scheme, we will go through a couple stepping stones. Specfically, we first describe a 2-server scheme with expected $\frac{n}{2}$ bandwidth, and an 8-server scheme with $O(n^{\frac{1}{3}})$ bandwidth. Eventually, we will get a 2-server scheme with $O(n^{\frac{1}{3}})$ bandwidth by coalescing the eight servers down to two.

---

[1] We assume a compact FHE scheme with the following performance bounds: the ciphertext size is $\widetilde{O}(1)$ for encrypting a plaintext of $\widetilde{O}(1)$ bits, and the encryption and decryption times are also $\widetilde{O}(1)$, and the homomorphic evaluation time is $\widetilde{O}(|\mathsf{Circ}|)$ for a circuit Circ.

[2] Without loss of generality, we may assume that n is a perfect square — if not, we can always round it up to the nearest perfect square incurring only constant blowup.

Warmup 1: $\mathbb{E}[\frac{n}{2}]$-bandwidth 2-server scheme. Below is a simple 2-server PIR scheme with $\frac{n}{2}$ expected bandwidth:

1. The client samples a subset $S_1 \subseteq [n]$ 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. The client sends S_1, S_2 to each server respectively.

2. For $i \in \{1, 2\}$, server i computes and sends back

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

3. On receiving r_1, r_2, the client outputs $r_1 \oplus r_2$.

Correctness follows from the fact that i is the only database index that appears once in S_1 and S_2, and every other index $j \neq i$ appears twice and thus the j-th index XORs away. For privacy, observe that S_1 is a uniformly random set. Further, for any $i \in [n]$, $S_2 = S_1 \Delta \{i\}$ is also a uniform random set. To see this, just consider the following distribution: toss n random coins, and then flip the i-th coin. This distribution is uniformly random.

Remark 2. *Note that if we run the above $n/2$-bandwidth 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 $\sqrt{n}$-bandwidth scheme in Section 2.1.*

Warmup 2: $n^{\frac{1}{3}}$–bandwidth 8-server scheme. We assume that $n = k^3$ for some integer k — if not, we can always round it up to the nearest cubic number, incurring only constant blowup. For convenience, we will number the databases indices from 0 to $n - 1$.

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 triple $(x^\star, y^\star, z^\star) \in \{0, \ldots, n^{\frac{1}{3}} - 1\}^3$. Note that $(x^\star, y^\star, z^\star)$ is also the base-$n^{\frac{1}{3}}$ representation of n.

The client samples subsets $X, Y, Z \subseteq \{0, \ldots, n^{\frac{1}{3}} - 1\}$ 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, the client computes 8 sets as follows, where $\times$ denotes Cartesian product:

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

Although each set's size is linear in n with high probability, observe that each of these sets $S_{000}, \ldots, S_{111}$ has a succinct representation of size $O(n^{1/3})$ — for example, S_{100} can be represented by the three sets $X\Delta\{x^*\}$, Y, and Z.

Now, we can construct an 8-server PIR protocol works as follows:

1. The client samples random X, Y, Z, and using its query (x^*, y^*, z^*), it computes the eight sets $S_{000}, \ldots, S_{111} \subseteq \{0, 1, \ldots, n^{\frac{1}{3}} - 1\}$ as mentioned above.

   The client sends a succinct representation of $S_{000}, \ldots, S_{111}$ to each of the eight servers, respectively.

2. For $i \in \{0, 1, \ldots, 7\}$, server i receives X', Y', Z' and computes

$$p_i = \bigoplus_{j \in X' \times Y' \times Z'} \mathsf{DB}[j]$$

   It sends back p_i to the client.

3. The client computes $p_0 \oplus p_1 \ldots \oplus p_7$.

Claim 1. $p_0 \oplus \cdots \oplus p_7 = \mathsf{DB}[i]$ *where* $i = (x^\star, y^\star, z^\star)$.

Proof. For every (x, y, z) not equal to the query (x^*, y^*, z^*), $\mathsf{DB}[x, y, z]$ appears an even number times in the XOR summation. We can pair up the sets to see this. On the other hand, the queried index $\mathsf{DB}[x^\star, y^\star, z^\star]$ appears only once in the XOR summation. $\square$

Privacy follows the argument as before, by observing that each of $S_{000}, \ldots, S_{111}$ has the same distribution as: sample X, Y, Z at random by including each of $\{0, \ldots, n^{\frac{1}{3}} - 1\}$ with probability $1/2$, and output the Cartesian product $X \times Y \times Z$.

$n^{\frac{1}{3}}$-bandwidth 2-server scheme. 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.

Henceforth, the parity for some set S means $\oplus_{j \in S} \mathsf{DB}[j]$. Each server, on receiving X', Y', Z', returns not only the parity of the set $X' \times Y' \times Z'$, but also the parities of the following $3n^{\frac{1}{3}}$ sets:

1. For each $x' \in \{0, \ldots, n^{\frac{1}{3}} - 1\}$, the server returns the parity of the set $(X' \Delta \{x'\}) \times Y' \times Z'$.

2. For each $y' \in \{0, \ldots, n^{\frac{1}{3}} - 1\}$, the server returns the parity of the set $X' \times (Y' \Delta \{y'\}) \times Z'$.

3. For each $z' \in \{0, \ldots, n^{\frac{1}{3}} - 1\}$, the server returns the parity of the set $X' \times Y' \times (Z' \Delta \{z'\})$.

Now, observe that the client can find the parities of S_{000} and $S_{100}, S_{010}, S_{001}$ from the first server's response, and it can find the parities of S_{111} and $S_{011}, S_{101}, S_{110}$ from the second server's response. Thus, it can still reconstruct the answer $\mathsf{DB}[i]$ as in the eight-server scheme.

Thus, this scheme still has $O(n^{\frac{1}{3}})$ bandwidth, and correctness and privacy follow as before.

2.3 Status Quo and Open Problems

It is known any 2-server information-theoretic PIR must incur at least $5 - o(1) \log n$ bandwidth [WdW05], while the best known upper bound requires $n^{O\left(\sqrt{\lg \lg n / \lg n}\right)} = n^{o(1)}$ bandwith (i.e., sub-polynomial bandwidth) [DG16]. Closing this gap is a long-standing open problem.

References

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[DG16] Zeev Dvir and Sivakanth Gopi. 2-server pir with subpolynomial communication. *Journal of the ACM (JACM)*, 63(4):1–15, 2016.

[HHCG+22] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meikle-john, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. Cryptology ePrint Archive, Paper 2022/949, 2022. `https://eprint.iacr.org/2022/949`.

[MW22] Samir Jordan Menon and David J. Wu. SPIRAL: Fast, high-rate single-server PIR via FHE composition. In *IEEE S&P*, 2022.

[WdW05] Stephanie Wehner and Ronald de Wolf. Improved lower bounds for locally decod-able codes and private information retrieval. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Lan-guages and Programming*, pages 1424–1436, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.