Report on PIR with Low Storage Overhead

Ehsan Ebrahimi Targhi

University of Tartu

December 15, 2015

Abstract

Private information retrieval (PIR) protocol, introduced in 1995 by Chor, Goldreich, Kushilevitz and Sudan, allows a user to retrieve a data item from a database without revealing which item is retrieved. Two main parameters of a PIR protocol are communication complexity, which is the total number of bits communicated between the user and the servers, and storage overhead, which is the ratio between the total number of bits stored on all the servers and the number of bits in the database. This report is based on the paper by Fazeli, Vardy and Yaakobi that is focused on the trade-off between these parameters in the information-theoretic setting. They showed that the storage overhead of a k-server PIR protocol can be arbitrarily close to the optimal value of 1 without sacrificing its communication complexity. They reduced the storage overhead parameter using coding technique instead of replication.

1 Introduction

The notion of information-theoretic private information retrieval was first introduced in [CGKS95]. It is possible to retrieve a data item from a database without leaking any information about the item being retrieved. A naive solution for the user is to download the entire database. In the case of a single database stored in a single server, it is shown in [CKGS98] that this solution is the best possible answer. However, it has a very high communication cost, $\Omega(n)$, where n is the length of the database in bits. A sublinear communication complexity is possible by replicating the database on several servers that do not communicate with each other ([CKGS98], [AFK97],[IK02]). Using the replication, the communication cost has been reduced by a subpolynomial factor in recent works ([Yek08], [Efr12], [DG15]). However, replication of the database on several servers increases the storage overhead of a PIR protocol. Clearly, the storage overhead of the mentioned protocols is at least 2.

In order not to increase the storage overhead, a private information retrieval protocol in the computational setting was studied by Kushilevitz and Ostrovsky [KO97] that avoids the replication. In the computational setting, the identity of data item is only computationally hidden from the databases. Subsequently, such PIR protocols have been studied in several works and based on different computational assumptions ([Gas04]). This report is based on the paper by Fazeli, Vardy and Yaakobi [FVY15] that achieves a low storage overhead in the information-theoretic setting. It is shown that the storage overhead of a k-server PIR protocol can be arbitrarily close to the optimal value of 1 without sacrificing its communication complexity.

In what follows, we present an overview of their coding technique. The technique to reduce the storage overhead has two main ingredients. First, the existence of a k-server PIR protocol in which the servers responses are a linear function of the database bits. Second, a binary linear code with the property that for every message bit x_i there are k disjoint sets of coded bits from which the message x_i can be recovered. Suppose the generator matrix for this code is $G_{s\times m}$ and

there are $m := s + r$ servers. Next, they partition the database into s blocks, say $X_1, \cdots, X_s$. The j-th server stores n/s bits c_j that is the result of applying the generator matrix $G_{s\times m}$ to the s partitions $X_1, \cdots, X_s$ (or $(c_1, \cdots, c_m) = (X_1, \cdots, X_s) \cdot G_{s\times m}$). It is clear that the storage overhead is $(s + r)/s$ and it approaches 1 when s grows. Every k-server PIR protocol consists of three algorithms (Q, A, C) where algorithm Q determines k queries, algorithm A determines servers' responses to the user's queries and finally algorithm C that based on servers' answers output the bit x_i that user is interested in. Therefore, the existence of a k-server PIR protocol guarantees an algorithm Q to produce k queries for the user, called Alice, and algorithm A that determines the servers' answers to those queries. For simplicity, suppose Alice is interested in the i-th bit that belongs to the first block X_1. Alice invokes algorithm $Q(k, n; i)$ to produce $(q_1, \cdots, q_k)$ queries. She needs answers $A(k, 1, X_1, q_1), A(k, 2, X_1, q_2), \cdots, A(k, k, X_1, q_k)$ from k servers to recover the i-th bit. However, servers stores the coded information $c_1, \cdots, c_m$. Recall that there are k disjoint subset of coded bits $c_1, \ldots, c_m$ such that Alice can recover the block message X_1 from them. Lets call these sets $R_1, \cdots, R_k$. Next, Alice sends the same query q_i to every server that stores a codeword that belongs to the set R_i and arbitrary queries to the remaining servers. Linearity of servers' responses over database guarantees that Alice can obtain $A(k, j, X_1, q_j)$ for every $j \in [k]$ (Notation $[k]$ denotes set $\{1, 2, \ldots, k\}$). The same procedure follows when i-th bit belongs to the block X_j. We explain these ideas in the next example.

Example: Suppose two servers S_1 and S_2 store a database $X \in \{0, 1\}^n$ and Alice wants to retrieve bit x_i without revealing any information about index i. Alice chooses uniformly at random $a \in \{0, 1\}^n$ and sends a and $a + e_i$ to S_1 and S_2 respectively where e_i is a binary vector of length n with single 1 in position i. Note that the distribution of Alice's queries are the same and since servers do not communicate with each other, they are not able to obtain any information about the index i. Servers respond with inner product of Alice's query and the database X. In other words, S_1 and S_2 answer with bits $X \cdot a$ and $(a + e_i) \cdot X$ respectively. Next, Alice can recover bit x_i by adding (over a binary field) servers' answers:

$$x_i = a \cdot X + a \cdot X + e_i \cdot X = a \cdot X + (a + e_i) \cdot X$$

Now, assume that the database X is partitioned into two equal parts X_1 and X_2 and there are 3 servers S_1, S_2 and S_3. Servers S_1, S_2 and S_3 store X_1, X_2 and $X_1 + X_2$, respectively. Assume Alice wants to retrieve the i-th bit where $i \in [n/2]$. She chooses uniformly at random $a \in \{0, 1\}^{n/2}$ and sends vector a, $a + e_i$ and $a + e_i$ to servers S_1, S_2 and S_3 respectively. Since a is chosen uniformly at random and servers do not communicate, the privacy of the scheme is preserved. Each server responds with inner product of Alice's query and the coded information that the server stores. Alice can recover the bit x_i by adding the answers:

$$x_i = a \cdot X_1 + a \cdot X_1 + e_i \cdot X_1 = a \cdot X_1 + (a + e_i) \cdot X_1 = a \cdot X_1 + (a + e_i) \cdot X_2 + (a + e_i) \cdot (X_1 + X_2).$$

In the case where Alice wants to retrieve the bit x_i from the second half of database, she sends $a + e_i$, a and $a + e_i$ to S_1, S_2 and S_3 respectively. A similar argument shows that Alice can retrieve x_i without revealing any information about the index i.

It is clear that the communication complexity and the storage overhead of first protocol are $2n + 2$ and 2 respectively, while in coded scheme the communication complexity and the storage overhead are $3(n/2) + 3$ and $3/2$ respectively.

2 Coded PIR scheme

In this section, we formally define a k-server PIR protocol, an (m, s)-server coded PIR protocol and a k-server PIR code.

A k-server PIR scheme consists of k servers $S_1, \ldots, S_k$ in which every server stores a database

X of length n and a user U who wants to retrieve bit x_i from database without revealing the index i.

Definition 1. *A k-server PIR protocol is a triple of algorithms $P = (Q, A, C)$ such that:*

- *Algorithm Q is a probabilistic algorithm that on input $(k, n; i)$ outputs queries $(q_1, \ldots, q_k)$. Let $q_j := Q_j(k, n; i)$. User sends query q_j to the server S_j.*
- *Server S_k runs algorithm A on input (k, j, x, q_j) and outputs answer a_j.*
- *User invokes Algorithm C on input $(k, n; i, a_1, \ldots, a_k)$ to obtain an output bit.*

Properties:

- *Privacy: Every server can not learn any information about the index i. More formally, the distribution $Q_j(k, n; i_1)$ and $Q_j(k, n; i_2)$ are identical for any k, n, and $i_1, i_2 \in [n]$, and a server $j \in [n]$.*
- *Correctness: $C(k, n; i, a_1, \ldots, a_k) = x_i$ for every k, n and $i \in [n]$ and database $X \in \{0,1\}^n$.*

Definition 2. *A k-server PIR protocol $P = (Q, A, C)$ is called a **linear** PIR protocol if the output of Algorithm A is a linear function of the database bits. More formally, for every n, $j \in [k]$, database X_1, X_2 of length n, and query q_j:*

$$A(k, j, X_1 + X_2, q_j) = A(k, j, X_1, q_j) + A(k, j, X_2, q_j).$$

In what follows, we define an (m, s)-server coded PIR scheme. Note that the main two differences to the last definition are as follows. First, the database is partitioned into s parts $X_1, \ldots, X_s$. Second, a function of $X_1, \ldots, X_s$ is stored in the servers and consequently every server stores n/s bits.

Definition 3. *An (m, s)-server coded PIR scheme consists of:*

- *A database $X \in \{0,1\}^n$ that partitions to s parts $X_1, \ldots, X_s$, each of length n/s.*
- *m servers $S_1, \ldots, S_m$ in which j-th server stores a coded information c_j that is a function of $X_1, \ldots, X_s$.*
- *user U who wants to retrieve the bit x_i from the database X without revealing index i.*

An (m, s)-server coded PIR protocol consists of three algorithms $P^* = (Q^*, A^*, C^*)$ such that:

- *Algorithm Q^* is a probabilistic algorithm that on input $(m, s, n; i)$ outputs queries $(q_1, \ldots, q_m)$. Let $q_j = Q_j^*(k, n; i)$. User sends query q_j to the server S_k.*
- *Server S_k runs algorithm A^* on input (m, s, j, c_j, q_j) and outputs answer a_j^*.*
- *User invokes Algorithm C on input $(m, s, n; i, a_1^*, \ldots, a_k^*)$.*

Properties:
Privacy and correctness as stated in Definition 1.

Definition 4. *we say that a binary matrix $\mathbf{G}_{s\times m}$ has property A_k if for every $i \in [s]$, there exist k disjoint subsets of columns of $\mathbf{G}$ whose sum is a binary vector of single 1 in position i. A binary linear $[m, s]$ code $\mathcal{C}$ will be called a k-server PIR code if there exist a generator matrix $\mathbf{G}$ for $\mathcal{C}$ with property A_k. In other words, if $\mathbf{c} = \mathbf{uG}$ is the encoding of message $\mathbf{u} \in \{0,1\}^s$ where $\mathbf{G}$ has property A_k, then there exist k disjoint sets $R_1, \ldots, R_k \subseteq [m]$ such that:*

$$u_i = \sum_{j\in R_1} c_j = \cdots = \sum_{j\in R_k} c_j.$$

Example: The following matrix has property A_3:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Now, assume there exists a 3-server linear PIR protocol $P = (Q, A, C)$ and a length-n database. Suppose that each server can store at most $n/4$ bits. If one wants to invoke the 3-server PIR protocol P, then the database has to be partitioned into 4 parts X_1, X_2, X_3, X_4 and each part has to be stored in three different servers. Therefore, the total number of servers needed is 12 servers and the storage overhead of the protocol is 3. In order to decrease the storage overhead, we use the matrix $\mathbf{G}$ that has A_3 property to construct (8,4)-server coded PIR protocol $P^* = (Q^*, A^*, C^*)$. We calculate the coded information $\mathbf{c} = (c_1, c_2, \ldots, c_8)$ using the generator matrix G:

$$(c_1, c_2, \ldots, c_8) = (X_1, X_2, X_3, X_4) \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Therefore,

$$\mathbf{c} = \big(X_1, \quad X_2, \quad X_3, \quad X_4, \quad X_1 + X_2, \quad X_2 + X_3, \quad X_3 + X_4, \quad X_1 + X_4\big)$$

We need 8 servers to store the coded information and j-th server stores the coded information c_j for $j \in [8]$. Suppose Alice wants to retrieve the i-th bit x_i where $i \in [n/4]$. It means this bit belongs to the first part of the database. She invokes the algorithm Q to get three queries:

$$Q(3, n/4; i) = (q_1, q_2, q_3).$$

Since bit x_i belongs to the database X_1, if Alice manages to receive answers $a_1 = A(3, 1, X_1, q_1)$, $a_2 = A(3, 2, X_1, q_2)$ and $a_3 = A(3, 3, X_1, q_3)$ from the servers, then she can invoke the algorithm $C(3, n/4; i, a_1, a_2, a_3)$ to retrieve the bit x_i. Therefore, Alice has to come up with the algorithm Q^* to produce 8 clever queries based on q_1, q_2, q_3. Since the first server stores X_1, if Alice submits q_1 to the first server, she would receive the answer $a_1 = A(3, 1, X_1, q_1)$. Note that the matrix $\mathbf{G}$ has property A_3 therefore $c_1 = X_1$, $c_2 + c_5 = X_1$ and $c_4 + c_8 = X_1$. Now, if Alice sends the query q_2 to the second and fifth server, she receives $a_2' = A(3, 2, c_2, q_2)$ and $a_5' = A(3, 2, c_5, q_2)$ and since the servers' responses are linear over database, she calculates $a_2 = A(3, 2, X_1, q_2)$ as follows:

$$A(3, 2, c_2, q_2) + A(3, 2, c_5, q_2) = A(3, 2, c_2 + c_5, q_2) = A(3, 2, X_1, q_2) = a_2.$$

In order to obtain answer a_3, Alice sends query q_3 to the 4-th server and 8-th server. As above, by linearity of servers' answers and A_3 property of matrix $\mathbf{G}$ Alice can compute $a_3 = A(3, 3, X_1, q_3)$. To sum up:

$$Q^*(8, 4, n; i) = (q_1,\ q_2,\ \text{anything},\ q_3,\ q_2,\ \text{anything},\ \text{anything},\ q_3).$$

Finally, Alice invokes algorithm C on input $(3, n/4; i, a_1, a_2, a_3)$ to retrieve the bit x_i.

The same procedure follows if the bit x_i that Alice wants to retrieve belongs to the other parts of database. For example suppose Alice wants to retrieve a bit from X_2. It is clear that the 3 disjoint set $\{c_2\}$, $\{c_1,\ c_5\}$ and $\{c_3,\ c_6\}$ can recover X_2 and consequently algorithm $Q^*(8, 4, n; i)$ can return queries as follows:

$$Q^*(8, 4, n; i) = (q_1,\ q_2,\ q_3,\ \text{anything},\ q_1,\ q_3,\ \text{anything},\ \text{anything}).$$

The queries that have not been specified can be assigned in a way to preserve the privacy of the protocol. Since the queries to the servers may differ depending on the part of database from which Alice wants to retrieve the bit, every server returns all possible outputs as follows:

$$A^*(8,4,1,X_1,q_1) = (A(3,1,X_1,q_1), A(3,2,X_1,q_1), A(3,3,X_1,q_1)).$$

Another solution to the privacy problem stated above which improves the download complexity is as follows. Alice chooses a random permutation and applies to the index of queries received from the algorithm Q and then does the same procedure as explained in the example. More formal analysis is given in the proof of Theorem 5 in [FVY15].

Theorem 1. *If there exist an $[m,s]$ k-server PIR code $\mathcal{C}$ and a k-server linear PIR protocol $\mathcal{P}$ then there exist an (m,s)-server coded PIR protocol $\mathcal{P}^*$. Furthermore,*

$$U^*(\mathcal{P}^*; n, m, s) = m \cdot U(\mathcal{P}; n/s, k),$$

$$D^*(\mathcal{P}^*; n, m, s) = m \cdot D(\mathcal{P}; n/s, k),$$

where U and D are the number of bits uploaded and downloaded respectively.

Proof. Refer to [FVY15, Theorem 5]. □

Since almost all known PIR schemes have linearity property as stated in Definition 2, we focus on the construction of PIR codes in the next section. For given s and k, let $A(s,k)$ be the optimal value of m such that an $[m,s]$ k-server PIR code exists.

3 The k-server PIR code

In this section, we present three constructions of k-server PIR codes. A k-server PIR code is one of the main ingredients of an (m,s)-server PIR code.

3.1 The Cubic Construction

We present a construction of coded PIR scheme based on geometry of multidimensional cubes. We construct an $[m,s]$ k-server PIR code where $s = \sigma^{k-1}$ and $m = \sigma^{k-1} + (k-1)\sigma^{k-2}$ for some positive integer σ. This code will be denoted by $\mathcal{C}_A(\sigma,k)$. We demonstrate the construction of the code $\mathcal{C}_A(4,3)$ in the next example.

Example: Assume $k = 3$ and $\sigma = 4$. The code $\mathcal{C}_A(4,3)$ is the following square array of size $(\sigma+1) \times (\sigma+1)$, without a bit in the bottom right corner.

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$	$p_1^{(1)}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$	$p_2^{(1)}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$	$p_3^{(1)}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$	$p_4^{(1)}$
$p_1^{(2)}$	$p_2^{(2)}$	$p_3^{(2)}$	$p_4^{(2)}$	

The information bits are denoted by $x_{i,j}$ for $i,j \in [4]$. The remaining bits are redundancy bits and for $i \in [4]$ they are defined as follows:

$$p_i^{(1)} = \sum_{j=1}^{\sigma} x_{i,j},$$

$$p_i^{(2)} = \sum_{j=1}^{\sigma} x_{j,i}.$$

For every information bit $x_{i,j}$, there are three mutually disjoint sets such that $x_{i,j}$ is a linear function of the bits in each set. For example for information bit $x_{1,1}$, three mutually disjoint sets that are used to recover information bit $x_{1,1}$ are $\{x_{1,1}\}$, $\{x_{1,2},\ x_{1,3},\ x_{1,4},\ p_1^{(1)}\}$ and $\{x_{2,1},\ x_{3,1},\ x_{4,1},\ p_1^{(2)}\}$.

Theorem 2. *For two positive integer σ and k, the coded $\mathcal{C}(\sigma, k)$ is a k-server PIR code. In particular, we get for any positive s*

$$A(s,k) \leq s + (k-1)\lceil s^{\frac{1}{k-1}} \rceil^{k-2}.$$

Proof. Refer to [FVY15, Theorem 6]. □

Therefore, the asymptotic behavior of the storage overhead in the cubic construction approaches 1, that is,

$$\lim_{s\to\infty} \frac{A(s,k)}{s} = 1.$$

3.2 One-step Majority Logic Codes

Let $\mathbf{G}_{s\times m} = [I_s | P_{s\times(m-s)}]$ be a systematic generator matrix for the code $\mathcal{C}$. The parity-check matrix for the code $\mathcal{C}$ is $[P^\top | I_{m-s}]$ that generates the dual code $\mathcal{C}^\perp$. Now, assume that for every $i \in [n]$, there exist J codewords $h_{i_1}, \ldots, h_{i_J}$ in the dual code that (mutually) intersect only on the i-th bit. The code with this property is called one-step majority code with J orthogonal vectors. This property guarantees that for every bit in the codeword $\mathbf{c} = (c_1,\ c_2, \ldots, c_m)$, there exist J mutually disjoint set $R_1, \ldots, R_j \subseteq [m]$ such that $c_i = \sum_{k\in R_1} c_k = \cdots = \sum_{k\in R_J} c_k$. These equation can be obtained since $h_{i_1} \cdot c = \cdots = h_{i_J} \cdot c = 0$ and the sets $R_1, \ldots, R_j \subseteq [m]$ are mutually disjoint since codewords $h_{i_1}, \ldots, h_{i_J}$ mutually intersect only on the i-th bit. Next example explain how one can obtain those mutually disjoint sets. For the reason that every information bit u_i for $i \in [s]$ is i-th bit in the codeword, any one-step majority logic code fits to the definition of $(J+1)$-server PIR code.

Example: Consider a (15, 7) cyclic code generated by polynomial

$$g(x) = 1 + x^4 + x^6 + x^7 + x^8.$$

The following codewords in $C^\perp$

$$h_1 = (1,0,0,0,1,0,0,0,0,0,0,0,1,1,0),$$
$$h_2 = (1,0,1,0,0,0,1,0,0,0,0,0,0,0,1),$$
$$h_3 = (1,1,0,1,1,0,0,0,1,0,0,0,0,0,0),$$
$$h_4 = (1,0,0,0,1,0,0,0,0,1,1,0,1,0,0),$$

mutually intersect only on the first bit. We use these codewords to obtain 5 mutually disjoint sets such that information bit u_1 can recover from them. Let $\mathbf{c} = (u_1, \ldots, u_8, c_9, \ldots, c_{15}) \in \mathcal{C}$. Using equation $h_1 \cdot \mathbf{c}^\top = 0$ we would get $u_1 = u_4 + c_{13} + c_{14}$. This gives us the set $\{4, 13, 14\}$ and similarly we can obtain sets $\{3, 7, 15\}$, $\{2, 4, 8\}$ and $\{9, 10, 12\}$ using other codewords. In addition, the set $\{1\}$ is used to recover information bit u_1.

An interested reader can refer to [FVY15, Theorem 9] to read about a detailed example of one-step majority codes.

3.3 Constant-Weight Code

If we look at the systematic PIR code from graph theory point of view, it will motivate us to look at constant-weight codes. Let $[I_s|M_{s\times r}]$ be generator matrix of a k-server PIR code $\mathcal{C}$ of length $m = s + r$ and dimension s. Suppose $\mathbf{M}$ is an incident matrix for bipartite graph $\mathcal{G}$ with the vertex sets $\mathcal{X} = \{x_1, \ldots, x_s\}$ and $\mathcal{P} = \{p_1, \ldots, p_r\}$, and edge set $\mathcal{E} = \{\{x_i, p_j\}|M_{i,j} = 1\}$. The following lemma shows connection between a bipartite graph and a k-server PIR code.

Lemma 3. *Let $\mathcal{G}$ be a bipartite graph with parties set $\mathcal{X} = \{x_1, \ldots, x_s\}$ and $\mathcal{P} = \{p_1, \ldots, p_r\}$ and incidence matrix $\mathbf{M}$. Assume $\min_{x\in\mathcal{X}} deg(x) = k - 1$ and $\mathcal{G}$ has no cycles of length 4. If $\mathcal{C}$ is a systematic code with generator matrix $[I_s|M_{s\times r}]$, then $\mathcal{C}$ is a k-server PIR code of length $m = s + r$ and dimension s.*

Proof. Let $\mathbf{x} = (x_1, \ldots, x_s)$ be an information vector. For every information bit x_i, we prove that there exist k mutually disjoint subsets of $[m]$ such that one can recover information bit x_i using each set and codeword $\mathbf{xG} = (x_1, \ldots, x_s, p_1, \ldots, p_r)$ where $p_1, \ldots, p_r$ are redundancy bits. Let $\{p_{i_1}, p_{i_2}, \cdots, p_{i_{k-1}}\}$ be set of $k-1$ neighbors of x_i and $\mathcal{N}(p_{i_j})$ be set of all neighbors of p_{i_j}. Since $\mathcal{G}$ is 4-cycle free, then $\mathcal{N}(p_{i_j}) \bigcap \mathcal{N}(p_{i_{j'}}) = \{x_i\}$ for every $j \neq j' \in [k-1]$. Therefore the sets $\mathcal{N}(p_{i_j})\backslash\{x_i\}$ (for $j \in [k-1]$) are mutually disjoint and we can recover information bit x_i using each set as follows:

$$x_i = p_{i_j} + \sum_{x_\alpha \in \mathcal{N}(p_{i_j})\backslash\{x_i\}} x_\alpha.$$

As a result, the sets $\{x_i\}$, and $\mathcal{N}(p_{i_j})\backslash\{x_i\}$ for $j \in [k-1]$ forms k disjoint recovery sets for information bit x_i.

□

Now, if we suppose the requirement $deg(x) = k - 1$ for every $x \in \mathcal{X}$, then we can look at constant-weight codes where the codewords are all rows in the matrix M. An interested reader can refer to [FVY15, Theorem 11] to analyse an example of constant-weight code and its parameters.

References

[AFK97] Andris Ambainis, Rusins Freivalds, and Marek Karpinski. Weak and strong recognition by 2-way randomized automata. In *Randomization and Approximation Techniques in Computer Science, International Workshop, RANDOM'97, Bolognna, Italy, July 11-12. 1997, Proceedings*, pages 175–185, 1997.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 41–50, 1995.

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

[DG15] Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 577–584, 2015.

[Efr12] Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012.

[FVY15] Arman Fazeli, Alexander Vardy, and Eitan Yaakobi. PIR with low storage overhead: Coding instead of replication. *CoRR*, abs/1505.06241, 2015.

[Gas04] William I. Gasarch. A survey on private information retrieval (column: Computational complexity). *Bulletin of the EATCS*, 82:72–107, 2004.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 364–373, 1997.

[Yek08] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1), 2008.