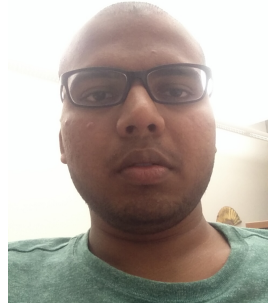
Private Information Retrieval: *Coding instead of Replication*

Alexander Vardy

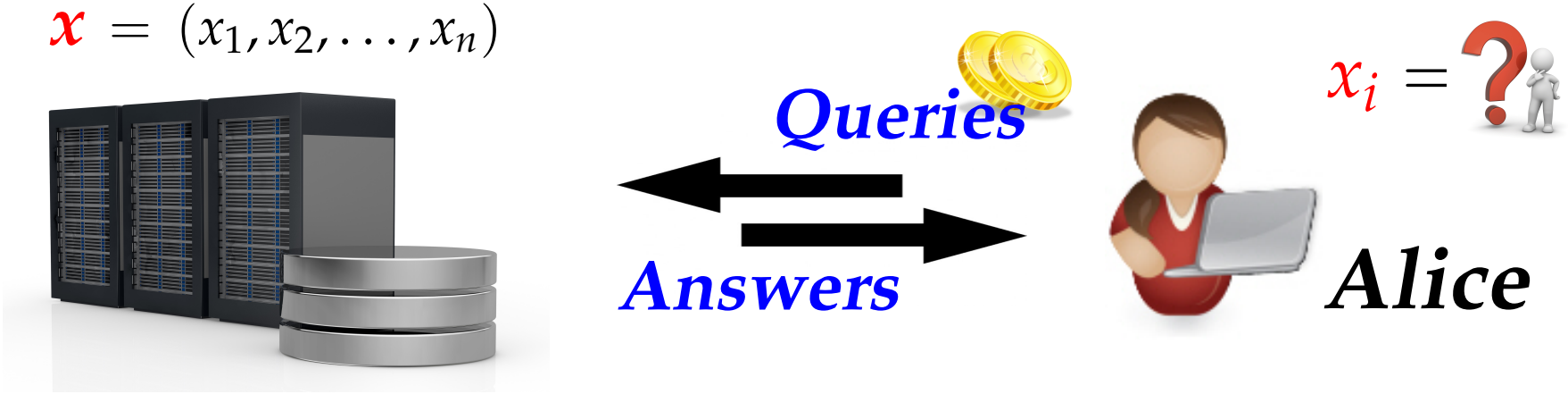
Information Theory & Applications Center
University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA

Jont work with:





What is private information retrieval?



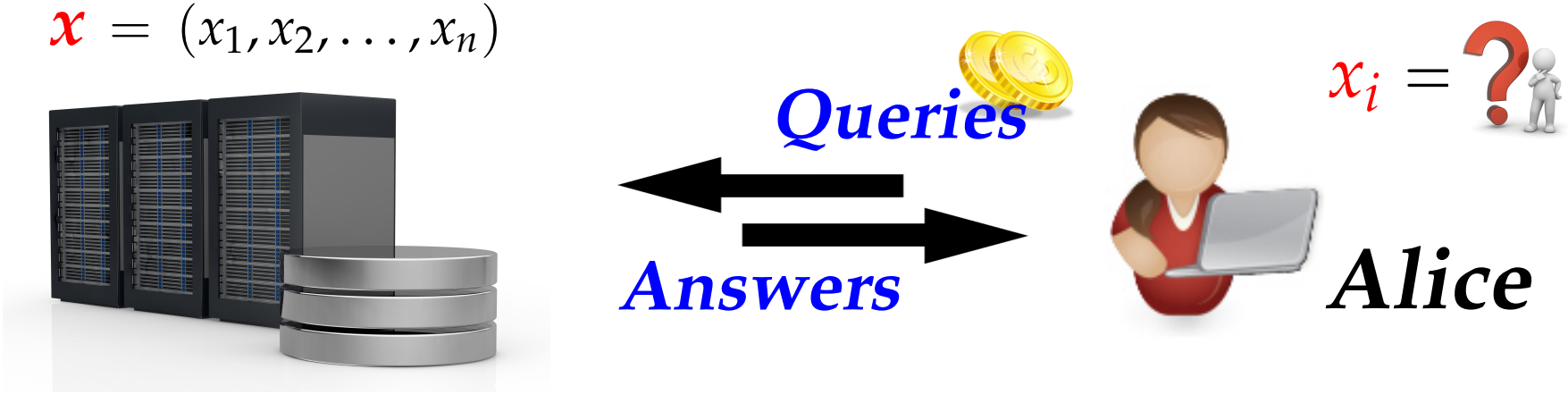
Private information retrieval (PIR)

Alice wishes to retrieve a data item x_i from the database $(x_1, x_2, \ldots, x_n)$ without revealing any information about i to the server.

Formal privacy condition: The distribution of randomized queries sent by the user to the server does not depend on i.

B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proceedings IEEE Symp. Foundations Computer Science*, pp. 41–50, October 1995.

What is private information retrieval?



Private information retrieval (PIR)

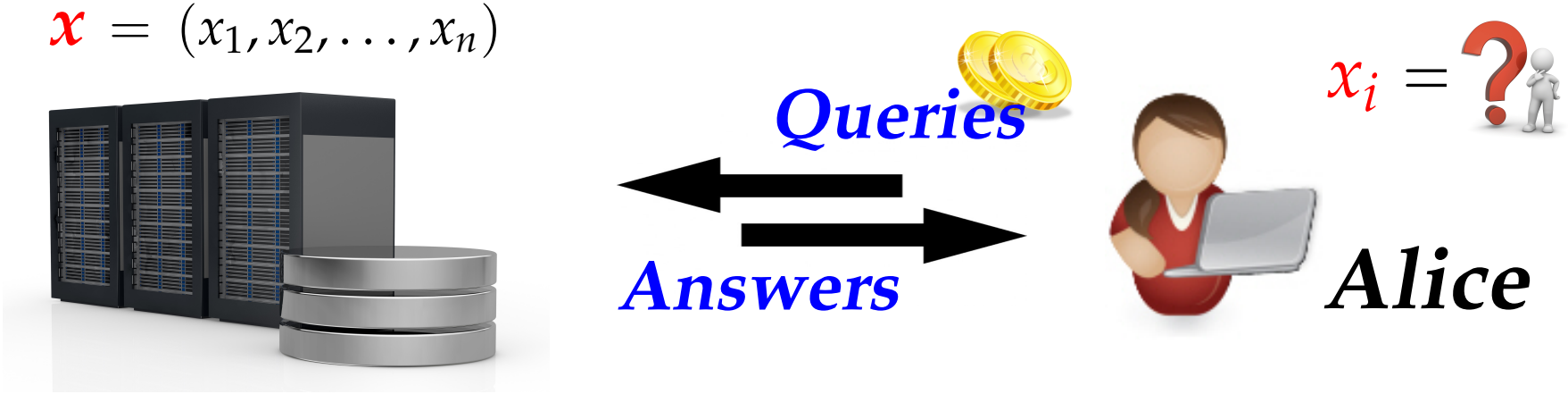
Alice wishes to retrieve a data item x_i from the database $(x_1, x_2, \ldots, x_n)$ without revealing any information about i to the server.

Formal privacy condition: The distribution of randomized queries sent by the user to the server does not depend on i.

Naive Solution: Ask the server to send the entire database!

B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proceedings IEEE Symp. Foundations Computer Science*, pp. 41–50, October 1995.

What is private information retrieval?



Private information retrieval (PIR)

Alice wishes to retrieve a data item x_i from the database $(x_1, x_2, \ldots, x_n)$ without revealing any information about i to the server.

Formal privacy condition: The distribution of randomized queries sent by the user to the server does not depend on i.

Solution: Ask the server to send the entire database!

This is the only solution possible! Communication cost $= \Omega(n)$.

B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proceedings IEEE Symp. Foundations Computer Science*, pp. 41–50, October 1995.

Two general classes of solutions

- ***Computational PIR***

  The server is computationally bounded + **standard cryptographic assumptions** (one-way functions, quadratic residuosity).

  E. Kushilevitz and R. Ostrovsky, Replication is not needed: Single database, computationally-private information retrieval, *Proc. 38-th IEEE Symp. Foundations Computer Science*, pp. 364–373, October 1997.

- ***Information-theoretic PIR***

  The **database is replicated** among $k \geqslant 2$ non-communicating servers, with guarantees of information-theoretic privacy.

B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proc. 36-th IEEE Symposium Foundations Computer Science*, pp. 41–50, October 1995.

Two general classes of solutions

- ***Computational PIR***

The server is computationally bounded + **standard cryptographic assumptions** (one-way functions, quadratic residuosity).

E. Kushilevitz and R. Ostrovsky, Replication is not needed: Single database, computationally-private information retrieval, *Proc. 38-th IEEE Symp. Foundations Computer Science*, pp. 364–373, October 1997.

- ***Information-theoretic PIR***

The **database is replicated** among $k \geqslant 2$ non-communicating servers, with guarantees of information-theoretic privacy.

B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proc. 36-th IEEE Symposium Foundations Computer Science*, pp. 41–50, October 1995.

This talk: We consider only information-theoretic PIR!

Information-theoretic PIR: Example

Replication among $k = 4$ servers $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ with communication cost of $8\sqrt{n} + 4$ bits. The database is represented as a square of side $\sqrt{n}$.

B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proceedings IEEE Symp. Foundations Computer Science*, pp. 41–50, October 1995.

Information-theoretic PIR: Example

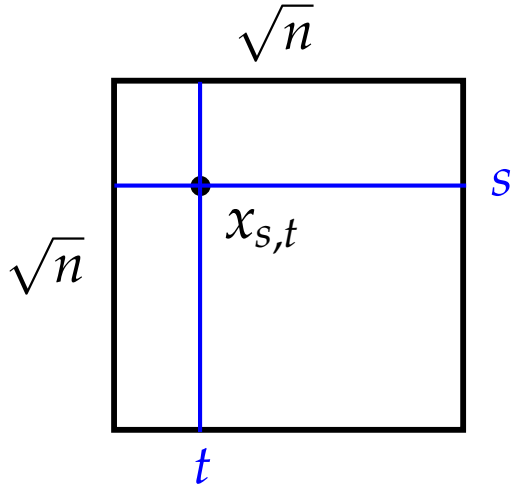
Replication among $k = 4$ servers $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ with communication cost of $8\sqrt{n} + 4$ bits. The database is represented as a square of side $\sqrt{n}$.

B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proceedings IEEE Symp. Foundations Computer Science*, pp. 41–50, October 1995.

Query generation:

Alice wishes to retrieve $x_{s,t}$. She generates the vectors $\boldsymbol{y}, \boldsymbol{z} \in \{0,1\}^{\sqrt{n}}$ ***uniformly at random,*** and sends

$$\mathcal{S}_1 \leftarrow (\boldsymbol{y}, \boldsymbol{z}), \quad \mathcal{S}_2 \leftarrow (\boldsymbol{y} + \boldsymbol{e}_s, \boldsymbol{z}),$$
$$\mathcal{S}_3 \leftarrow (\boldsymbol{y}, \boldsymbol{z} + \boldsymbol{e}_t), \quad \mathcal{S}_4 \leftarrow (\boldsymbol{y} + \boldsymbol{e}_s, \boldsymbol{z} + \boldsymbol{e}_t)$$



Information-theoretic PIR: Example

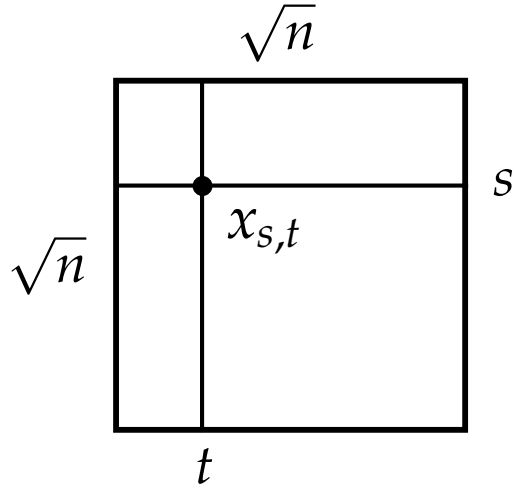
Replication among $k = 4$ servers $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$ with communication cost of $8\sqrt{n} + 4$ bits. The database is represented as a square of side $\sqrt{n}$.

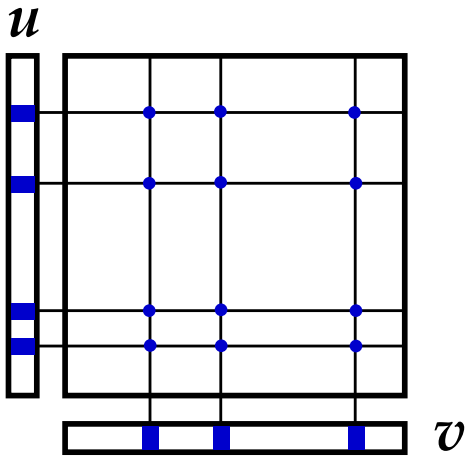
B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, Private information retrieval, *Proceedings IEEE Symp. Foundations Computer Science*, pp. 41–50, October 1995.

Query generation:

Alice wishes to retrieve $x_{s,t}$. She generates the vectors $\boldsymbol{y}, \boldsymbol{z} \in \{0,1\}^{\sqrt{n}}$ ***uniformly at random,*** and sends

$$\mathcal{S}_1 \leftarrow (\boldsymbol{y}, \boldsymbol{z}), \quad \mathcal{S}_2 \leftarrow (\boldsymbol{y} + \boldsymbol{e}_s, \boldsymbol{z}),$$
$$\mathcal{S}_3 \leftarrow (\boldsymbol{y}, \boldsymbol{z} + \boldsymbol{e}_t), \quad \mathcal{S}_4 \leftarrow (\boldsymbol{y} + \boldsymbol{e}_s, \boldsymbol{z} + \boldsymbol{e}_t)$$





Answer computation:

Given a query $(\boldsymbol{u}, \boldsymbol{v})$, each server $\mathcal{S}_i$ returns the following:

$$a = \sum_{i \in \mathrm{supp}(\boldsymbol{u})} \sum_{j \in \mathrm{supp}(\boldsymbol{v})} x_{i,j}$$

Information-theoretic PIR: Example

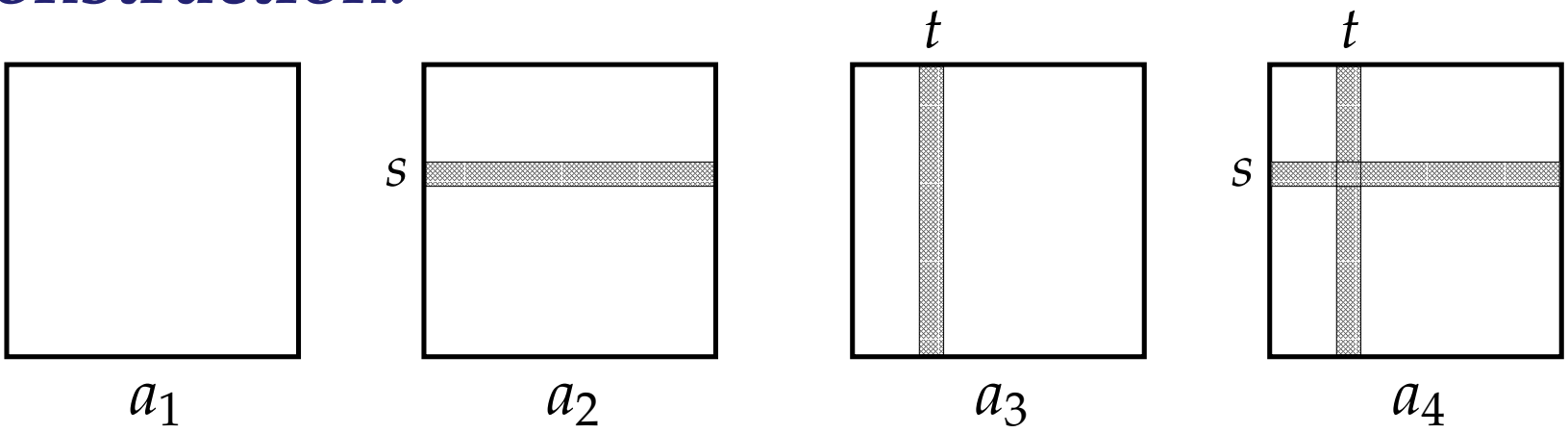
Query generation:

$$\mathcal{S}_1 \leftarrow (y, z),\ \mathcal{S}_2 \leftarrow (y + e_s, z),\ \mathcal{S}_3 \leftarrow (y, z + e_t),\ \mathcal{S}_4 \leftarrow (y + e_s, z + e_t)$$

Answer computation:

$$a = \sum_{i \in \mathrm{supp}(u)} \sum_{j \in \mathrm{supp}(v)} x_{i,j}$$

Reconstruction:



Information-theoretic PIR: Example

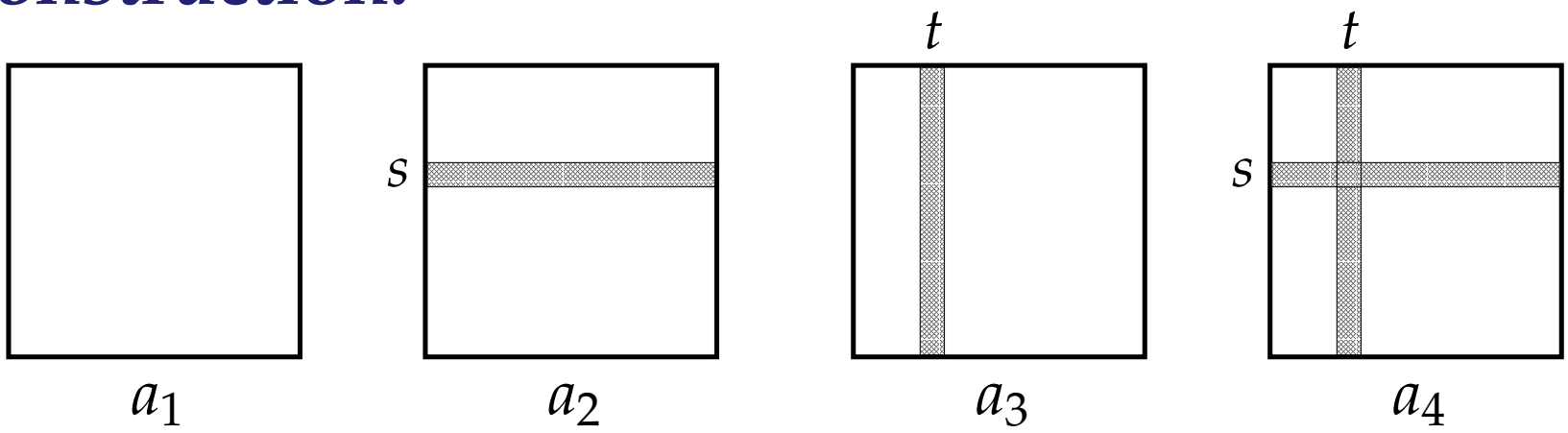
Query generation:

$$\mathcal{S}_1 \leftarrow (y, z),\ \mathcal{S}_2 \leftarrow (y + e_s, z),\ \mathcal{S}_3 \leftarrow (y, z + e_t),\ \mathcal{S}_4 \leftarrow (y + e_s, z + e_t)$$

Answer computation:

$$a = \sum_{i \in \mathrm{supp}(u)} \sum_{j \in \mathrm{supp}(v)} x_{i,j}$$

Reconstruction:



1. The bit $x_{s,t}$ contributes to **exactly one** of the answers a_1, a_2, a_3, a_4.

Information-theoretic PIR: Example

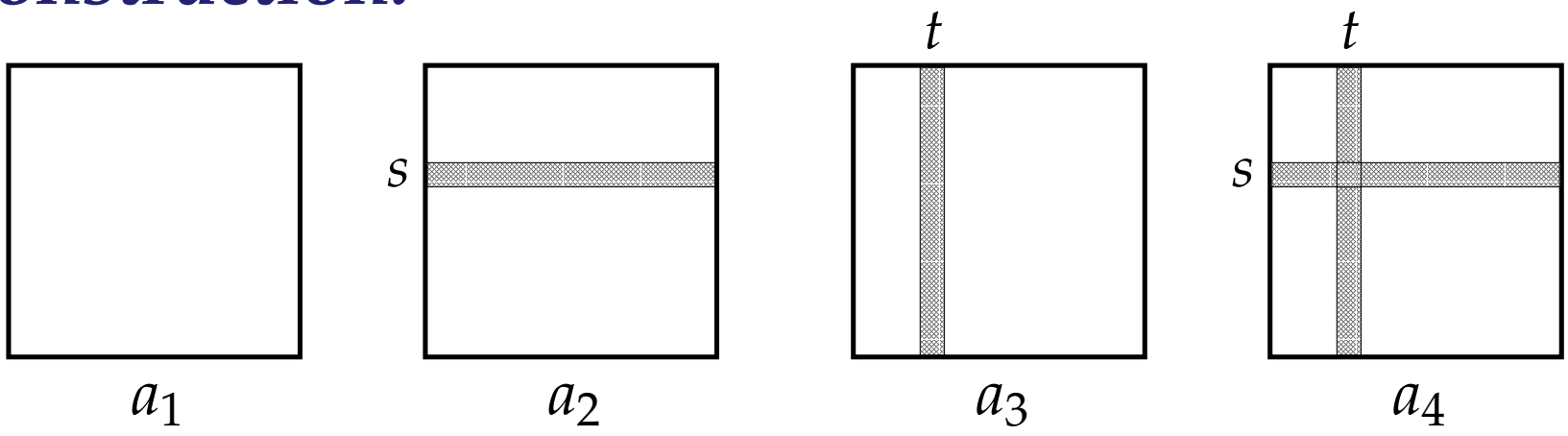
Query generation:

$$\mathcal{S}_1 \leftarrow (y, z),\ \mathcal{S}_2 \leftarrow (y + e_s, z),\ \mathcal{S}_3 \leftarrow (y, z + e_t),\ \mathcal{S}_4 \leftarrow (y + e_s, z + e_t)$$

Answer computation:

$$a = \sum_{i \in \mathrm{supp}(u)} \sum_{j \in \mathrm{supp}(v)} x_{i,j}$$

Reconstruction:



1. The bit $x_{s,t}$ contributes to **exactly one** of the answers a_1, a_2, a_3, a_4.
2. All other bits in the database contribute an **even number** of times.

Information-theoretic PIR: Example

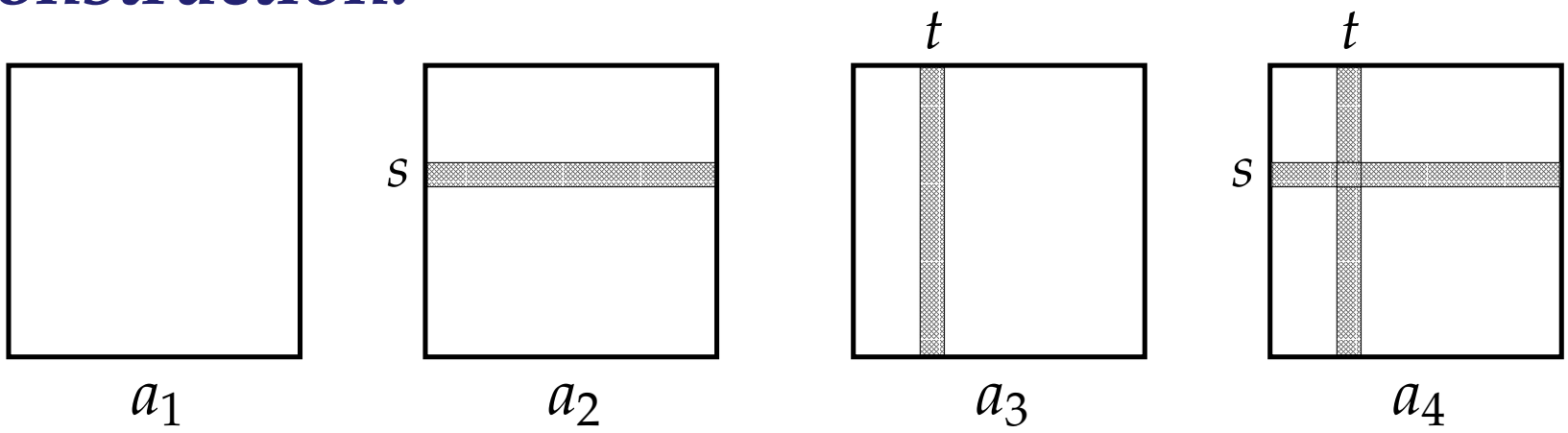
Query generation:

$$\mathcal{S}_1 \leftarrow (y, z),\ \mathcal{S}_2 \leftarrow (y + e_s, z),\ \mathcal{S}_3 \leftarrow (y, z + e_t),\ \mathcal{S}_4 \leftarrow (y + e_s, z + e_t)$$

Answer computation:

$$a = \sum_{i \in \mathrm{supp}(u)} \sum_{j \in \mathrm{supp}(v)} x_{i,j}$$

Reconstruction:



1. The bit $x_{s,t}$ contributes to **exactly one** of the answers a_1, a_2, a_3, a_4.
2. All other bits in the database contribute an **even number** of times.

It follows from 1 and 2 that:

$$a_1 + a_2 + a_3 + a_4 = x_{s,t}$$

Progress in information-theoretic PIR

During the past 20 years, the communication cost of information-theoretic PIR has been reduced dramatically by many researchers:

# of Servers	Communication Complexity	Year	Reference
2	$O\left(n^{\frac{1}{3}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{k}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{2k-1}}\right)$	1997	A. Ambainis
k	$O\left(n^{\frac{\log\log k}{k\log k}}\right)$	2002	A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond
$k \geqslant 3$	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2008	S. Yekhanin; K. Efremenko
2	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2014	Z. Dvir and S. Gopi

Progress in information-theoretic PIR

During the past 20 years, the communication cost of information-theoretic PIR has been reduced dramatically by many researchers:

# of Servers	Communication Complexity	Year	Reference
2	$O\left(n^{\frac{1}{3}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{k}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{2k-1}}\right)$	1997	A. Ambainis
k	$O\left(n^{\frac{\log\log k}{k\log k}}\right)$	2002	A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond
$k \geqslant 3$	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2008	S. Yekhanin; K. Efremenko
2	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2014	Z. Dvir and S. Gopi

Progress in information-theoretic PIR

During the past 20 years, the communication cost of information-theoretic PIR has been reduced dramatically by many researchers:

# of Servers	Communication Complexity	Year	Reference
2	$O\left(n^{\frac{1}{3}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{k}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{2k-1}}\right)$	1997	A. Ambainis
k	$O\left(n^{\frac{\log\log k}{k\log k}}\right)$	2002	A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond
$k \geqslant 3$	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2008	S. Yekhanin; K. Efremenko
2	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2014	Z. Dvir and S. Gopi

Progress in information-theoretic PIR

During the past 20 years, the communication cost of information-theoretic PIR has been reduced dramatically by many researchers:

# of Servers	Communication Complexity	Year	Reference
2	$O\left(n^{\frac{1}{3}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{k}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{2k-1}}\right)$	1997	A. Ambainis
k	$O\left(n^{\frac{\log\log k}{k\log k}}\right)$	2002	A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond
$k \geqslant 3$	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2008	S. Yekhanin; K. Efremenko
2	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2014	Z. Dvir and S. Gopi

Progress in information-theoretic PIR

During the past 20 years, the communication cost of information-theoretic PIR has been reduced dramatically by many researchers:

# of Servers	Communication Complexity	Year	Reference
2	$O\left(n^{\frac{1}{3}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{k}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{2k-1}}\right)$	1997	A. Ambainis
k	$O\left(n^{\frac{\log\log k}{k\log k}}\right)$	2002	A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond
$k \geqslant 3$	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2008	S. Yekhanin; K. Efremenko
2	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2014	Z. Dvir and S. Gopi

Progress in information-theoretic PIR

During the past 20 years, the communication cost of information-theoretic PIR has been reduced dramatically by many researchers:

# of Servers	Communication Complexity	Year	Reference
2	$O\left(n^{\frac{1}{3}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{k}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{2k-1}}\right)$	1997	A. Ambainis
k	$O\left(n^{\frac{\log\log k}{k\log k}}\right)$	2002	A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond
$k \geqslant 3$	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2008	S. Yekhanin; K. Efremenko
2	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2014	Z. Dvir and S. Gopi

Progress in information-theoretic PIR

During the past 20 years, the communication cost of information-theoretic PIR has been reduced dramatically by many researchers:

# of Servers	Communication Complexity	Year	Reference
2	$O\left(n^{\frac{1}{3}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{k}}\right)$	1995	B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan
k	$O\left(n^{\frac{1}{2k-1}}\right)$	1997	A. Ambainis
k	$O\left(n^{\frac{\log\log k}{k\log k}}\right)$	2002	A. Beimel, Y. Ishai, E. Kushilevitz, and J.F. Raymond
$k \geqslant 3$	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2008	S. Yekhanin; K. Efremenko
2	$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)}$	2014	Z. Dvir and S. Gopi

Note: Dvir-Gopi protocol gives an even better communication cost for large k.

What about storage overhead?

In addition to the communication cost, another important cost metric is the storage overhead, defined as follows:

$$\textbf{storage overhead} \stackrel{\text{def}}{=} \frac{\text{total number of bits stored on all the servers}}{\text{number of bits in the database}}$$

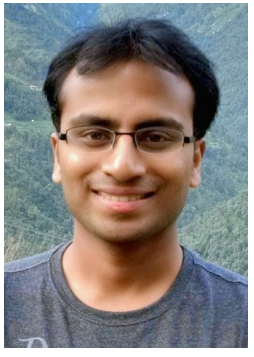
- The storage overhead of replicating the database k times is trivially k. The Dvir and Gopi paper is considered a breakthrough in part because it reduces the storage overhead from $k \geqslant 3$ to $k = 2$, for the same complexity.

What about storage overhead?

In addition to the communication cost, another important cost metric is the storage overhead, defined as follows:

$$\textbf{storage overhead} \stackrel{\text{def}}{=} \frac{\text{total number of bits stored on all the servers}}{\text{number of bits in the database}}$$

- The storage overhead of replicating the database k times is trivially k. The Dvir and Gopi paper is considered a breakthrough in part because it reduces the storage overhead from $k \geqslant 3$ to $k = 2$, for the same complexity.

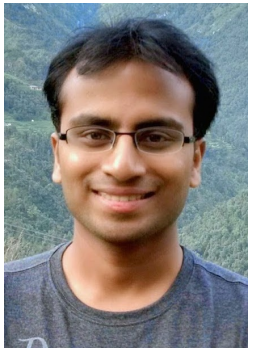


What about storage overhead?

In addition to the communication cost, another important cost metric is the storage overhead, defined as follows:

$$\textbf{storage overhead} \stackrel{\text{def}}{=} \frac{\text{total number of bits stored on all the servers}}{\text{number of bits in the database}}$$

- The storage overhead of replicating the database k times is trivially k. The Dvir and Gopi paper is considered a breakthrough in part because it reduces the storage overhead from $k \geqslant 3$ to $k = 2$, for the same complexity.



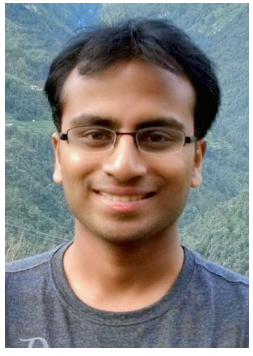
Should we be happy with $k = 2$? In coding theory, increasing the amount of stored data by a factor of two is ***often undesirable***.

What about storage overhead?

In addition to the communication cost, another important cost metric is the storage overhead, defined as follows:

$$\textbf{storage overhead} \stackrel{\text{def}}{=} \frac{\text{total number of bits stored on all the servers}}{\text{number of bits in the database}}$$

- The storage overhead of replicating the database k times is trivially k. The Dvir and Gopi paper is considered a breakthrough in part because it reduces the storage overhead from $k \geqslant 3$ to $k = 2$, for the same complexity.



Should we be happy with $k = 2$? In coding theory, increasing the amount of stored data by a factor of two is ***often undesirable***.

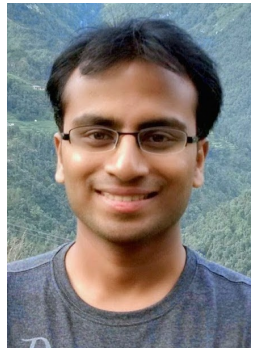
But doing better than $k = 2$ is impossible! It was shown back in 1995 that the communication cost is $\Omega(n)$ unless the database is replicated on at least two non-communicating servers.

What about storage overhead?

In addition to the communication cost, another important cost metric is the storage overhead, defined as follows:

$$\textbf{storage overhead} \stackrel{\text{def}}{=} \frac{\text{total number of bits stored on all the servers}}{\text{number of bits in the database}}$$

- The storage overhead of replicating the database k times is trivially k. The Dvir and Gopi paper is considered a breakthrough in part because it reduces the storage overhead from $k \geqslant 3$ to $k = 2$, for the same complexity.



Should we be happy with $k = 2$? In coding theory, increasing the amount of stored data by a factor of two is ***often undesirable***.

But doing better than $k = 2$ is impossible! It was shown back in 1995 that the communication cost is $\Omega(n)$ unless the database is replicated on at least two non-communicating servers.

Or is it?

This talk: The main theme

This is cryptography, people!
We do the impossible
for breakfast.

Open Problem: Can we achieve information-theoretic PIR with low communication cost but *without doubling (or worse if $k \geqslant 3$) the number of bits we need to store?*

This talk: The main theme

This is cryptography, people!
We do the impossible
for breakfast.

Open Problem: Can we achieve information-theoretic PIR with low communication cost but *without doubling (or worse if $k \geqslant 3$) the number of bits we need to store?*

Taking cue from distributed storage: In practice, the database may need to be stored in a distributed manner (e.g., for security or reliability purposes).

This talk: The main theme

This is cryptography, people!
We do the impossible
for breakfast.

Open Problem: Can we achieve information-theoretic PIR with low communication cost but *without doubling (or worse if $k \geqslant 3$) the number of bits we need to store?*

Taking cue from distributed storage: In practice, the database may need to be stored in a distributed manner (e.g., for security or reliability purposes).

Key idea: Partitioning the database

Partition the database string x into parts $x_1, x_2, \ldots, x_s$. We will use $m \geqslant k$ non-communicating servers. But **each server will store only part of the database,** so that the total number of bits stored is $(1+\varepsilon)n$.

Conventional k-server PIR

Definition: k-server PIR scheme

*A **k-server PIR scheme** consists of the following: a binary string $\mathbf{x}$ of length n, called the **database**, k non-communicating **servers** $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ each storing a replica of $\mathbf{x}$, a **user Alice** who wishes to retrieve x_i for some $i \in [n]$, without revealing i to any of the servers, and a **k-server PIR protocol**.*

Definition: k-server PIR protocol [CKGS95]

*The **k-server PIR protocol** $\mathcal{P}$ involves a triple of algorithms $\mathcal{Q}, \mathcal{A}, \mathcal{C}$ and consists of the following steps:*

1. *Alice flips coins and uses the random outcome to invoke the **query algorithm** $\mathcal{Q}(k, n; i)$ that generates a k-tuple of queries $q_1, q_2, \ldots, q_k$.*
2. *For all $j \in [k]$, Alice sends the query q_j to the j-th server $\mathcal{S}_j$.*
3. *For all $j \in [k]$, the server $\mathcal{S}_j$ invokes the **answer algorithm** $\mathcal{A}$ to respond with the answer $a_j = \mathcal{A}(k, j; \mathbf{x}, q_j)$.*
4. *Alice computes x_i using the **reconstruction algorithm** $\mathcal{C}(k, n; i, a_1, \ldots, a_k)$.*

*The three algorithms together satisfy the **correctness** ($\mathcal{C}(k, n; i, a_1, \ldots, a_k) = x_i$) and the **privacy** (distibution of q_j independent of i) conditions defined earlier.*

Conventional k-server PIR: Linearity

Our construction of distributed PIR schemes with low storage overhead uses **two main ingredients:**

1. A binary linear code $\mathbb{C}$ with a certain special property, to be defined shortly.
2. An existing k-server PIR protocol in which the answer algorithm is ***linear in the database.***

Conventional k-server PIR: Linearity

Our construction of distributed PIR schemes with low storage overhead uses **two main ingredients:**

1. A binary linear code $\mathbb{C}$ with a certain special property, to be defined shortly.
2. An existing k-server PIR protocol in which the answer algorithm is ***linear in the database.***

Definition: Linear k-server PIR protocol

*A k-server PIR protocol $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ is **linear** if for all $x_1, x_2 \in \{0,1\}^n$ and for all possible queries q, the following holds:*

$$\mathcal{A}(k,j;x_1+x_2,q) \;=\; \mathcal{A}(k,j;x_1,q) \;+\; \mathcal{A}(k,j;x_2,q) \qquad \text{for all } j \in [k]$$

Conventional k-server PIR: Linearity

Our construction of distributed PIR schemes with low storage overhead uses **two main ingredients:**

1. A binary linear code $\mathbb{C}$ with a certain special property, to be defined shortly.
2. An existing k-server PIR protocol in which the answer algorithm is ***linear in the database.***

Definition: Linear k-server PIR protocol

A k-server PIR protocol $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ is ***linear*** *if for all $x_1, x_2 \in \{0,1\}^n$ and for all possible queries q, the following holds:*

$$\mathcal{A}(k,j;x_1+x_2,q) \;=\; \mathcal{A}(k,j;x_1,q) \;+\; \mathcal{A}(k,j;x_2,q) \qquad \text{for all } j \in [k]$$

Good news: All known PIR protocols are linear!

Note: We also assume that the answer algorithm $\mathcal{A}$ is public knowledge. This means that any server can simulate the answers of any other server.

Example: Coded 3-server PIR

Example: Reducing the storage overhead of 3-server PIR

Consider ***any*** *existing 3-server PIR protocol* $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$*, and assume it is linear. We will reduce its storage overhead from* $k = 3$ *to* $m/s = 2$*.*

Example: Coded 3-server PIR

Example: Reducing the storage overhead of 3-server PIR

*Consider **any** existing 3-server PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$, and assume it is linear. We will reduce its storage overhead from $k = 3$ to $m/s = 2$.*

We partition the database x of length n into 4 parts x_1, x_2, x_3, x_4, each of length $n/4$. These parts are distributed among 8 servers as follows:

$$
\begin{array}{ll}
\mathcal{S}_1\colon\ c_1 = x_1 & \mathcal{S}_5\colon\ c_5 = x_1 + x_2 \\
\mathcal{S}_2\colon\ c_2 = x_2 & \mathcal{S}_6\colon\ c_6 = x_2 + x_3 \\
\mathcal{S}_3\colon\ c_3 = x_3 & \mathcal{S}_7\colon\ c_7 = x_3 + x_4 \\
\mathcal{S}_4\colon\ c_4 = x_4 & \mathcal{S}_8\colon\ c_8 = x_4 + x_1
\end{array}
\qquad (\star)
$$

The result is a coded PIR scheme with $s = 4$ ***parts*** x_1, x_2, x_3, x_4 and $m = 8$ ***coded shares*** $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$.

Example: Coded 3-server PIR

Example: Reducing the storage overhead of 3-server PIR

*Consider **any** existing 3-server PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$, and assume it is linear. We will reduce its storage overhead from $k = 3$ to $m/s = 2$.*

We partition the database x of length n into 4 parts x_1, x_2, x_3, x_4, each of length $n/4$. These parts are distributed among 8 servers as follows:

$$\begin{array}{ll} \mathcal{S}_1\colon\ c_1 = x_1 & \mathcal{S}_5\colon\ c_5 = x_1 + x_2 \\ \mathcal{S}_2\colon\ c_2 = x_2 & \mathcal{S}_6\colon\ c_6 = x_2 + x_3 \\ \mathcal{S}_3\colon\ c_3 = x_3 & \mathcal{S}_7\colon\ c_7 = x_3 + x_4 \\ \mathcal{S}_4\colon\ c_4 = x_4 & \mathcal{S}_8\colon\ c_8 = x_4 + x_1 \end{array} \qquad (\star)$$

The result is a coded PIR scheme with $s = 4$ ***parts*** x_1, x_2, x_3, x_4 and $m = 8$ ***coded shares*** $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$.

$$\textbf{storage overhead} = \frac{n/s \text{ bits stored on } m \text{ servers}}{n \text{ bits in the database}} = \frac{m}{s}$$

Example: How to retrieve x_i?

Assume, for now, that Alice wishes to read the ***i*-th bit from the first part x_1.** That is, she wants the bit $x_{1,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3,1; c_1, q_1) = \mathcal{A}(3,1; x_1, q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3,2; c_2, q_2) = \mathcal{A}(3,2; x_2, q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3,3; c_4, q_3) = \mathcal{A}(3,3; x_4, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2; c_5, q_2) = \mathcal{A}(3,2; x_1 + x_2, q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3,3; c_5, q_3) = \mathcal{A}(3,3; x_4 + x_1, q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a_2' = a_2 + a_5 = \mathcal{A}(3,2; x_2, q_2) + \mathcal{A}(3,2; x_1 + x_2, q_2) = \mathcal{A}(3,2; x_1, q_2)$$

Example: How to retrieve x_i?

Assume, for now, that Alice wishes to read the ***i*-th bit from the first part x_1.** That is, she wants the bit $x_{1,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3,1; c_1, q_1) = \mathcal{A}(3,1; x_1, q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3,2; c_2, q_2) = \mathcal{A}(3,2; x_2, q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3,3; c_4, q_3) = \mathcal{A}(3,3; x_4, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2; c_5, q_2) = \mathcal{A}(3,2; x_1 + x_2, q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3,3; c_5, q_3) = \mathcal{A}(3,3; x_4 + x_1, q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a_2' = a_2 + a_5 = \mathcal{A}(3,2; x_2, q_2) + \mathcal{A}(3,2; x_1 + x_2, q_2) = \mathcal{A}(3,2; x_1, q_2)$$

Example: How to retrieve x_i?

Assume, for now, that Alice wishes to read the ***i*-th bit from the first part x_1.** That is, she wants the bit $x_{1,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3,1; c_1, q_1) = \mathcal{A}(3,1; x_1, q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3,2; c_2, q_2) = \mathcal{A}(3,2; x_2, q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3,3; c_4, q_3) = \mathcal{A}(3,3; x_4, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2; c_5, q_2) = \mathcal{A}(3,2; x_1 + x_2, q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3,3; c_5, q_3) = \mathcal{A}(3,3; x_4 + x_1, q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_2 + a_5 = \mathcal{A}(3,2; x_2, q_2) + \mathcal{A}(3,2; x_1 + x_2, q_2) = \mathcal{A}(3,2; x_1, q_2)$$

Example: How to retrieve x_i?

Assume, for now, that Alice wishes to read the ***i*-th bit from the first part x_1.** That is, she wants the bit $x_{1,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3,1;\boldsymbol{c}_1, q_1) = \mathcal{A}(3,1;\boldsymbol{x}_1, q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3,2;\boldsymbol{c}_2, q_2) = \mathcal{A}(3,2;\boldsymbol{x}_2, q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3,3;\boldsymbol{c}_4, q_3) = \mathcal{A}(3,3;\boldsymbol{x}_4, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;\boldsymbol{c}_5, q_2) = \mathcal{A}(3,2;\boldsymbol{x}_1 + \boldsymbol{x}_2, q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3,3;\boldsymbol{c}_5, q_3) = \mathcal{A}(3,3;\boldsymbol{x}_4 + \boldsymbol{x}_1, q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_2 + a_5 = \mathcal{A}(3,2;\boldsymbol{x}_2, q_2) + \mathcal{A}(3,2;\boldsymbol{x}_1 + \boldsymbol{x}_2, q_2) = \mathcal{A}(3,2;\boldsymbol{x}_1, q_2)$$

Example: How to retrieve x_i?

Assume, for now, that Alice wishes to read the ***i*-th bit from the first part x_1.** That is, she wants the bit $x_{1,i}$ for some $i \in [n/4]$. She proceeds as follows:

2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3,1;\boldsymbol{c}_1,q_1) = \mathcal{A}(3,1;\boldsymbol{x}_1,q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3,2;\boldsymbol{c}_2,q_2) = \mathcal{A}(3,2;\boldsymbol{x}_2,q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3,3;\boldsymbol{c}_4,q_3) = \mathcal{A}(3,3;\boldsymbol{x}_4,q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;\boldsymbol{c}_5,q_2) = \mathcal{A}(3,2;\boldsymbol{x}_1+\boldsymbol{x}_2,q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3,3;\boldsymbol{c}_5,q_3) = \mathcal{A}(3,3;\boldsymbol{x}_4+\boldsymbol{x}_1,q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_2 + a_5 = \mathcal{A}(3,2;\boldsymbol{x}_2,q_2) + \mathcal{A}(3,2;\boldsymbol{x}_1+\boldsymbol{x}_2,q_2) = \mathcal{A}(3,2;\boldsymbol{x}_1,q_2)$$
$$a'_3 = a_4 + a_8 = \mathcal{A}(3,3;\boldsymbol{x}_4,q_3) + \mathcal{A}(3,3;\boldsymbol{x}_4+\boldsymbol{x}_1,q_3) = \mathcal{A}(3,3;\boldsymbol{x}_1,q_3)$$

Example: How to retrieve x_i?

Assume, for now, that Alice wishes to read the ***i*-th bit from the first part** x_1. That is, she wants the bit $x_{1,i}$ for some $i \in [n/4]$. She proceeds as follows:

2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3,1;c_1,q_1) = \mathcal{A}(3,1;x_1,q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3,2;c_2,q_2) = \mathcal{A}(3,2;x_2,q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3,3;c_4,q_3) = \mathcal{A}(3,3;x_4,q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;c_5,q_2) = \mathcal{A}(3,2;x_1+x_2,q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3,3;c_5,q_3) = \mathcal{A}(3,3;x_4+x_1,q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_2 + a_5 = \mathcal{A}(3,2;x_2,q_2) + \mathcal{A}(3,2;x_1+x_2,q_2) = \mathcal{A}(3,2;x_1,q_2)$$
$$a'_3 = a_4 + a_8 = \mathcal{A}(3,3;x_4,q_3) + \mathcal{A}(3,3;x_4+x_1,q_3) = \mathcal{A}(3,3;x_1,q_3)$$

Example: How to retrieve x_i?

Assume, for now, that Alice wishes to read the ***i*-th bit from the first part x_1.** That is, she wants the bit $x_{1,i}$ for some $i \in [n/4]$. She proceeds as follows:

3. Alice ignores the answers from $\mathcal{S}_3, \mathcal{S}_6, \mathcal{S}_7$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_1	$a_1 = \mathcal{A}(3,1;c_1,q_1) = \mathcal{A}(3,1;x_1,q_1)$
$\mathcal{S}_2$	q_2	$a_2 = \mathcal{A}(3,2;c_2,q_2) = \mathcal{A}(3,2;x_2,q_2)$
$\mathcal{S}_4$	q_3	$a_4 = \mathcal{A}(3,3;c_4,q_3) = \mathcal{A}(3,3;x_4,q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;c_5,q_2) = \mathcal{A}(3,2;x_1+x_2,q_2)$
$\mathcal{S}_8$	q_3	$a_8 = \mathcal{A}(3,3;c_5,q_3) = \mathcal{A}(3,3;x_4+x_1,q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_2 + a_5 = \mathcal{A}(3,2;x_2,q_2) + \mathcal{A}(3,2;x_1+x_2,q_2) = \mathcal{A}(3,2;x_1,q_2)$$
$$a'_3 = a_4 + a_8 = \mathcal{A}(3,3;x_4,q_3) + \mathcal{A}(3,3;x_4+x_1,q_3) = \mathcal{A}(3,3;x_1,q_3)$$

5. Using the ***reconstruction algorithm of*** $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$, Alice now computes $\mathcal{C}(3,n/4;i,a_1,a'_2,a'_3)$, which is given by:

$$\mathcal{C}(3,n/4;i,\mathcal{A}(3,1;x_1,q_1),\mathcal{A}(3,2;x_1,q_2),\mathcal{A}(3,3;x_1,q_3)) = x_{1,i}$$

Example: How to retrieve x_i?

Now assume that Alice wishes to read the ***i*-th bit from the second part x_2.** That is, she wants the bit $x_{2,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$, exactly as before.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_2, q_1, q_3, q_3, q_2, q_3, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_4, \mathcal{S}_7, \mathcal{S}_8$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_2	$a_1 = \mathcal{A}(3, 2; c_1, q_2) = \mathcal{A}(3, 2; x_1, q_2)$
$\mathcal{S}_2$	q_1	$a_2 = \mathcal{A}(3, 1; c_2, q_1) = \mathcal{A}(3, 1; x_2, q_1)$
$\mathcal{S}_3$	q_3	$a_3 = \mathcal{A}(3, 3; c_3, q_3) = \mathcal{A}(3, 3; x_3, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3, 2; c_5, q_2) = \mathcal{A}(3, 2; x_1 + x_2, q_2)$
$\mathcal{S}_6$	q_3	$a_6 = \mathcal{A}(3, 3; c_6, q_3) = \mathcal{A}(3, 3; x_2 + x_3, q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_1 + a_5 = \mathcal{A}(3, 2; x_1, q_2) + \mathcal{A}(3, 2; x_1 + x_2, q_2) = \mathcal{A}(3, 2; x_2, q_2)$$

Example: How to retrieve x_i?

Now assume that Alice wishes to read the ***i*-th bit from the second part x_2.** That is, she wants the bit $x_{2,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$, exactly as before.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_2, q_1, q_3, q_3, q_2, q_3, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_4, \mathcal{S}_7, \mathcal{S}_8$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_2	$a_1 = \mathcal{A}(3,2; c_1, q_2) = \mathcal{A}(3,2; x_1, q_2)$
$\mathcal{S}_2$	q_1	$a_2 = \mathcal{A}(3,1; c_2, q_1) = \mathcal{A}(3,1; x_2, q_1)$
$\mathcal{S}_3$	q_3	$a_3 = \mathcal{A}(3,3; c_3, q_3) = \mathcal{A}(3,3; x_3, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2; c_5, q_2) = \mathcal{A}(3,2; x_1 + x_2, q_2)$
$\mathcal{S}_6$	q_3	$a_6 = \mathcal{A}(3,3; c_6, q_3) = \mathcal{A}(3,3; x_2 + x_3, q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_1 + a_5 = \mathcal{A}(3,2; x_1, q_2) + \mathcal{A}(3,2; x_1 + x_2, q_2) = \mathcal{A}(3,2; x_2, q_2)$$

Example: How to retrieve x_i?

Now assume that Alice wishes to read the ***i*-th bit from the second part x_2.** That is, she wants the bit $x_{2,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$, exactly as before.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_2, q_1, q_3, q_3, q_2, q_3, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_4, \mathcal{S}_7, \mathcal{S}_8$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_2	$a_1 = \mathcal{A}(3,2;c_1,q_2) = \mathcal{A}(3,2;x_1,q_2)$
$\mathcal{S}_2$	q_1	$a_2 = \mathcal{A}(3,1;c_2,q_1) = \mathcal{A}(3,1;x_2,q_1)$
$\mathcal{S}_3$	q_3	$a_3 = \mathcal{A}(3,3;c_3,q_3) = \mathcal{A}(3,3;x_3,q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;c_5,q_2) = \mathcal{A}(3,2;x_1+x_2,q_2)$
$\mathcal{S}_6$	q_3	$a_6 = \mathcal{A}(3,3;c_6,q_3) = \mathcal{A}(3,3;x_2+x_3,q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_1 + a_5 = \mathcal{A}(3,2;x_1,q_2) + \mathcal{A}(3,2;x_1+x_2,q_2) = \mathcal{A}(3,2;x_2,q_2)$$

Example: How to retrieve x_i?

Now assume that Alice wishes to read the ***i*-th bit from the second part x_2.** That is, she wants the bit $x_{2,i}$ for some $i \in [n/4]$. She proceeds as follows:

1. Alice flips coins and invokes the ***query algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ to generate three queries $q_1, q_2, q_3 := \mathcal{Q}(3, n/4; i)$, exactly as before.
2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_2, q_1, q_3, q_3, q_2, q_3, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_4, \mathcal{S}_7, \mathcal{S}_8$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_2	$a_1 = \mathcal{A}(3,2;\boldsymbol{c}_1, q_2) = \mathcal{A}(3,2;\boldsymbol{x}_1, q_2)$
$\mathcal{S}_2$	q_1	$a_2 = \mathcal{A}(3,1;\boldsymbol{c}_2, q_1) = \mathcal{A}(3,1;\boldsymbol{x}_2, q_1)$
$\mathcal{S}_3$	q_3	$a_3 = \mathcal{A}(3,3;\boldsymbol{c}_3, q_3) = \mathcal{A}(3,3;\boldsymbol{x}_3, q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;\boldsymbol{c}_5, q_2) = \mathcal{A}(3,2;\boldsymbol{x}_1 + \boldsymbol{x}_2, q_2)$
$\mathcal{S}_6$	q_3	$a_6 = \mathcal{A}(3,3;\boldsymbol{c}_6, q_3) = \mathcal{A}(3,3;\boldsymbol{x}_2 + \boldsymbol{x}_3, q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_1 + a_5 = \mathcal{A}(3,2;\boldsymbol{x}_1, q_2) + \mathcal{A}(3,2;\boldsymbol{x}_1 + \boldsymbol{x}_2, q_2) = \mathcal{A}(3,2;\boldsymbol{x}_2, q_2)$$

Example: How to retrieve x_i?

Now assume that Alice wishes to read the ***i*-th bit from the second part x_2.** That is, she wants the bit $x_{2,i}$ for some $i \in [n/4]$. She proceeds as follows:

2. She sends queries to the 8 servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5, \mathcal{S}_6, \mathcal{S}_7, \mathcal{S}_8) \leftarrow (q_2, q_1, q_3, q_3, q_2, q_3, q_3, q_3)$$

3. Alice ignores the answers from $\mathcal{S}_4, \mathcal{S}_7, \mathcal{S}_8$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_2	$a_1 = \mathcal{A}(3,2;c_1,q_2) = \mathcal{A}(3,2;x_1,q_2)$
$\mathcal{S}_2$	q_1	$a_2 = \mathcal{A}(3,1;c_2,q_1) = \mathcal{A}(3,1;x_2,q_1)$
$\mathcal{S}_3$	q_3	$a_3 = \mathcal{A}(3,3;c_3,q_3) = \mathcal{A}(3,3;x_3,q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;c_5,q_2) = \mathcal{A}(3,2;x_1+x_2,q_2)$
$\mathcal{S}_6$	q_3	$a_6 = \mathcal{A}(3,3;c_6,q_3) = \mathcal{A}(3,3;x_2+x_3,q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_1 + a_5 = \mathcal{A}(3,2;x_1,q_2) + \mathcal{A}(3,2;x_1+x_2,q_2) = \mathcal{A}(3,2;x_2,q_2)$$
$$a'_3 = a_3 + a_6 = \mathcal{A}(3,3;x_3,q_3) + \mathcal{A}(3,3;x_2+x_3,q_3) = \mathcal{A}(3,3;x_2,q_3)$$

Example: How to retrieve x_i?

Now assume that Alice wishes to read the ***i*-th bit from the second part x_2.** That is, she wants the bit $x_{2,i}$ for some $i \in [n/4]$. She proceeds as follows:

3. Alice ignores the answers from $\mathcal{S}_4, \mathcal{S}_7, \mathcal{S}_8$ but collects the other five answers as follows:

Server	Query	Response
$\mathcal{S}_1$	q_2	$a_1 = \mathcal{A}(3,2;c_1,q_2) = \mathcal{A}(3,2;x_1,q_2)$
$\mathcal{S}_2$	q_1	$a_2 = \mathcal{A}(3,1;c_2,q_1) = \mathcal{A}(3,1;x_2,q_1)$
$\mathcal{S}_3$	q_3	$a_3 = \mathcal{A}(3,3;c_3,q_3) = \mathcal{A}(3,3;x_3,q_3)$
$\mathcal{S}_5$	q_2	$a_5 = \mathcal{A}(3,2;c_5,q_2) = \mathcal{A}(3,2;x_1+x_2,q_2)$
$\mathcal{S}_6$	q_3	$a_6 = \mathcal{A}(3,3;c_6,q_3) = \mathcal{A}(3,3;x_2+x_3,q_3)$

4. Since the ***answer algorithm of*** $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ ***is linear*** in the database, Alice can compute:

$$a'_2 = a_1 + a_5 = \mathcal{A}(3,2;x_1,q_2) + \mathcal{A}(3,2;x_1+x_2,q_2) = \mathcal{A}(3,2;x_2,q_2)$$
$$a'_3 = a_3 + a_6 = \mathcal{A}(3,3;x_3,q_3) + \mathcal{A}(3,3;x_2+x_3,q_3) = \mathcal{A}(3,3;x_2,q_3)$$

5. Using the ***reconstruction algorithm of*** $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$, Alice now computes $\mathcal{C}(3,n/4;i,a_2,a'_2,a'_3)$, which is given by:

$$\mathcal{C}(3,n/4;i,\mathcal{A}(3,1;x_2,q_1),\mathcal{A}(3,2;x_2,q_2),\mathcal{A}(3,3;x_2,q_3)) = x_{2,i}$$

Coded k-server PIR: Definition

Definition: Coded k-server PIR scheme

*A **coded k-server PIR scheme with s parts and m shares** consists of the following ingredients:*

- *A binary string $\mathbf{x}$ of length n, called the **database**, that is partitioned into s **parts** $x_1, x_2, \ldots, x_s$, each of length n/s.*
- *Coded **shares** $c_1, c_2, \ldots, c_m$ of length n/s, where $\mathbf{c}_j$ is a linear function of $x_1, x_2, \ldots, x_s$ for all $j \in [m]$, stored in m **non-communicating servers** $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$.*
- *A **user Alice** who wishes to retrieve x_i for some $i \in [n]$, without revealing i to any of the servers.*
- *A **coded k-server PIR protocol** $\mathcal{P}^*(\mathcal{Q}^*, \mathcal{A}^*, \mathcal{C}^*)$ that emulates a conventional k-server PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$.*

Note: The emulation property of $\mathcal{P}^*(\mathcal{Q}^*, \mathcal{A}^*, \mathcal{C}^*)$ can be formally defined.

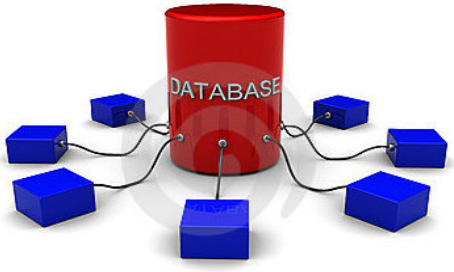
Coded k-server PIR: Definition

Definition: Coded k-server PIR scheme

*A **coded k-server PIR scheme with s parts and m shares** consists of the following ingredients:*

- *A binary string $\mathbf{x}$ of length n, called the **database**, that is partitioned into s **parts** $x_1, x_2, \ldots, x_s$, each of length n/s.*
- *Coded **shares** $c_1, c_2, \ldots, c_m$ of length n/s, where $\mathbf{c}_j$ is a linear function of $x_1, x_2, \ldots, x_s$ for all $j \in [m]$, stored in m **non-communicating servers** $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$.*
- *A **user Alice** who wishes to retrieve x_i for some $i \in [n]$, without revealing i to any of the servers.*
- *A **coded k-server PIR protocol** $\mathcal{P}^*(\mathcal{Q}^*, \mathcal{A}^*, \mathcal{C}^*)$ that emulates a conventional k-server PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$.*

Note: The emulation property of $\mathcal{P}^*(\mathcal{Q}^*, \mathcal{A}^*, \mathcal{C}^*)$ can be formally defined.



Theorem 1: Storage overhead of coded PIR

The storage overhead of a coded k-server PIR scheme with s parts and m coded shares is m/s.

General coded PIR schemes?

So far, we have seen a general definition, and a single example of a coded PIR scheme with 4 parts and 8 shares that conforms to this defintion.

General coded PIR schemes?

So far, we have seen a general definition, and a single example of a coded PIR scheme with 4 parts and 8 shares that conforms to this defintion.

General coded PIR schemes?

So far, we have seen a general definition, and a single example of a coded PIR scheme with 4 parts and 8 shares that conforms to this defintion.

- Why does the bit retrieval in the example work? Why does everything nicely cancel out?
- For which values of m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about their communication complexity?
- How small can we make the storage overhead ratio m/s?

General coded PIR schemes?

So far, we have seen a general definition, and a single example of a coded PIR scheme with 4 parts and 8 shares that conforms to this defintion.

- Why does the bit retrieval in the example work? Why does everything nicely cancel out?
- For which values of m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about their communication complexity?
- How small can we make the storage overhead ratio m/s?

General coded PIR schemes?

So far, we have seen a general definition, and a single example of a coded PIR scheme with 4 parts and 8 shares that conforms to this defintion.

- Why does the bit retrieval in the example work? Why does everything nicely cancel out?
- For which values of m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about their communication complexity?
- How small can we make the storage overhead ratio m/s?

General coded PIR schemes?

So far, we have seen a general definition, and a single example of a coded PIR scheme with 4 parts and 8 shares that conforms to this defintion.

- Why does the bit retrieval in the example work? Why does everything nicely cancel out?
- For which values of m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about their communication complexity?
- How small can we make the storage overhead ratio m/s?

General coded PIR schemes?

So far, we have seen a general definition, and a single example of a coded PIR scheme with 4 parts and 8 shares that conforms to this defintion.

- Why does the bit retrieval in the example work? Why does everything nicely cancel out?
- For which values of m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about their communication complexity?
- How small can we make the storage overhead ratio m/s?

To answer these questions, let us begin by revisiting the example.

The example revisited

In the encoding equations ($\star$) of the example, the 8 coded shares are computed from the four database parts x_1, x_2, x_3, x_4 as follows:

$$\begin{array}{llll} c_1 = x_1, & c_3 = x_3, & c_5 = x_1 + x_2, & c_7 = x_3 + x_4 \\ c_2 = x_2, & c_4 = x_4, & c_6 = x_2 + x_3, & c_8 = x_4 + x_1 \end{array}$$

Rewrite these equations in matrix form:

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8) = (x_1, x_2, x_3, x_4) \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Observe that each part x_1, x_2, x_3, x_4 of the database can be recovered from the coded shares in $k = 3$ different ways. Explicitly:

$$\begin{array}{llll} x_1 = c_1 = c_5 + c_2 = c_8 + c_4 \\ x_2 = c_2 = c_5 + c_1 = c_6 + c_3 \\ x_3 = c_3 = c_6 + c_3 = c_7 + c_4 \\ x_4 = c_4 = c_7 + c_3 = c_8 + c_1 \end{array}$$

Moreover, each coded share $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$ appears in each of the four ***recovery equations*** above no more than once.

The example revisited

In the encoding equations ($\star$) of the example, the 8 coded shares are computed from the four database parts x_1, x_2, x_3, x_4 as follows:

$$\begin{array}{llll} c_1 = x_1, & c_3 = x_3, & c_5 = x_1 + x_2, & c_7 = x_3 + x_4 \\ c_2 = x_2, & c_4 = x_4, & c_6 = x_2 + x_3, & c_8 = x_4 + x_1 \end{array}$$

Rewrite these equations in matrix form:

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8) = (x_1, x_2, x_3, x_4) \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Observe that each part x_1, x_2, x_3, x_4 of the database can be recovered from the coded shares in $k = 3$ different ways. Explicitly:

$$\begin{array}{llll} x_1 = c_1 = c_5 + c_2 = c_8 + c_4 \\ x_2 = c_2 = c_5 + c_1 = c_6 + c_3 \\ x_3 = c_3 = c_6 + c_3 = c_7 + c_4 \\ x_4 = c_4 = c_7 + c_3 = c_8 + c_1 \end{array}$$

Moreover, each coded share $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$ appears in each of the four ***recovery equations*** above no more than once.

The example revisited

In the encoding equations ($\star$) of the example, the 8 coded shares are computed from the four database parts x_1, x_2, x_3, x_4 as follows:

$$\begin{array}{llll} c_1 = x_1, & c_3 = x_3, & c_5 = x_1 + x_2, & c_7 = x_3 + x_4 \\ c_2 = x_2, & c_4 = x_4, & c_6 = x_2 + x_3, & c_8 = x_4 + x_1 \end{array}$$

Rewrite these equations in matrix form:

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8) = (x_1, x_2, x_3, x_4) \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Observe that each part x_1, x_2, x_3, x_4 of the database can be recovered from the coded shares in $k = 3$ different ways. Explicitly:

$$\begin{array}{lllll} x_1 = c_1 = c_5 + c_2 = c_8 + c_4 \\ x_2 = c_2 = c_5 + c_1 = c_6 + c_3 \\ x_3 = c_3 = c_6 + c_3 = c_7 + c_4 \\ x_4 = c_4 = c_7 + c_3 = c_8 + c_1 \end{array}$$

Moreover, each coded share $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$ appears in each of the four ***recovery equations*** above no more than once.

The example revisited

In the encoding equations ($\star$) of the example, the 8 coded shares are computed from the four database parts x_1, x_2, x_3, x_4 as follows:

$$\begin{array}{llll} c_1 = x_1, & c_3 = x_3, & c_5 = x_1 + x_2, & c_7 = x_3 + x_4 \\ c_2 = x_2, & c_4 = x_4, & c_6 = x_2 + x_3, & c_8 = x_4 + x_1 \end{array}$$

Rewrite these equations in matrix form:

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8) = (x_1, x_2, x_3, x_4) \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Observe that each part x_1, x_2, x_3, x_4 of the database can be recovered from the coded shares in $k = 3$ different ways. Explicitly:

$$\begin{array}{rcccccc} x_1 & = & c_1 & = & c_5 + c_2 & = & c_8 + c_4 \\ x_2 & = & c_2 & = & c_5 + c_1 & = & c_6 + c_3 \\ x_3 & = & c_3 & = & c_6 + c_3 & = & c_7 + c_4 \\ x_4 & = & c_4 & = & c_7 + c_3 & = & c_8 + c_1 \end{array}$$

Moreover, each coded share $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$ appears in each of the four ***recovery equations*** above no more than once.

PIR matrix and PIR codes

Definition: k-server PIR matrix

Let $\boldsymbol{e}_i$ denote the binary unit vector with 1 in position i and zeros elsewhere. An $s \times m$ binary matrix G is said to have ***property*** *P_k if for all $i \in [s]$ there exist k disjoint sets of columns of G that add to $\boldsymbol{e}_i$. A matrix that has property P_k is also said to be a k-**server PIR matrix**.*

PIR matrix and PIR codes

Definition: k-server PIR matrix

Let $\boldsymbol{e}_i$ denote the binary unit vector with 1 in position i and zeros elsewhere. An $s \times m$ binary matrix G is said to have ***property*** *P_k if for all $i \in [s]$ there exist k disjoint sets of columns of G that add to $\boldsymbol{e}_i$. A matrix that has property P_k is also said to be a* ***k-server PIR matrix****.*

Example: 4×8 matrix with property P_3

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Note: This is the encoding matrix for the PIR scheme in our example.

PIR matrix and PIR codes

Definition: k-server PIR matrix

Let $\boldsymbol{e}_i$ denote the binary unit vector with 1 in position i and zeros elsewhere. An $s \times m$ binary matrix G is said to have ***property*** *P_k if for all $i \in [s]$ there exist k disjoint sets of columns of G that add to $\boldsymbol{e}_i$. A matrix that has property P_k is also said to be a* ***k-server PIR matrix****.*

Example: 4×8 matrix with property P_3

$$\begin{bmatrix} \boxed{1} & \boxed{0} & 0 & \boxed{0} & \boxed{1} & 0 & 0 & \boxed{1} \\ \boxed{0} & \boxed{1} & 0 & \boxed{0} & \boxed{1} & 1 & 0 & \boxed{0} \\ \boxed{0} & \boxed{0} & 1 & \boxed{0} & \boxed{0} & 1 & 1 & \boxed{0} \\ \boxed{0} & \boxed{0} & 0 & \boxed{1} & \boxed{0} & 0 & 1 & \boxed{1} \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Note: This is the encoding matrix for the PIR scheme in our example.

PIR matrix and PIR codes

Definition: k-server PIR matrix

Let $\boldsymbol{e}_i$ denote the binary unit vector with 1 in position i and zeros elsewhere. An $s \times m$ binary matrix G is said to have ***property*** *P_k if for all $i \in [s]$ there exist k disjoint sets of columns of G that add to $\boldsymbol{e}_i$. A matrix that has property P_k is also said to be a* ***k-server PIR matrix****.*

Example: 4×8 matrix with property P_3

$$\begin{bmatrix} \boxed{1}\boxed{0}\boxed{0} & 0 & \boxed{1}\boxed{0} & 0 & 1 \\ \boxed{0}\boxed{1}\boxed{0} & 0 & \boxed{1}\boxed{1} & 0 & 0 \\ \boxed{0}\boxed{0}\boxed{1} & 0 & \boxed{0}\boxed{1} & 1 & 0 \\ \boxed{0}\boxed{0}\boxed{0} & 1 & \boxed{0}\boxed{0} & 1 & 1 \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Note: This is the encoding matrix for the PIR scheme in our example.

PIR matrix and PIR codes

Definition: k-server PIR matrix

Let $\boldsymbol{e}_i$ denote the binary unit vector with 1 in position i and zeros elsewhere. An $s \times m$ binary matrix G is said to have ***property*** *P_k if for all $i \in [s]$ there exist k disjoint sets of columns of G that add to $\boldsymbol{e}_i$. A matrix that has property P_k is also said to be a* ***k-server PIR matrix****.*

Example: 4×8 matrix with property P_3

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \Rightarrow \boldsymbol{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Note: This is the encoding matrix for the PIR scheme in our example.

PIR matrix and PIR codes

Definition: k-server PIR matrix

Let $\boldsymbol{e}_i$ denote the binary unit vector with 1 in position i and zeros elsewhere. An $s \times m$ binary matrix G is said to have ***property*** *P_k if for all $i \in [s]$ there exist k disjoint sets of columns of G that add to $\boldsymbol{e}_i$. A matrix that has property P_k is also said to be a* ***k-server PIR matrix****.*

Example: 4×8 matrix with property P_3

$$\begin{bmatrix} \boxed{1} & 0 & \boxed{0} & \boxed{0} & 1 & 0 & \boxed{0} & \boxed{1} \\ \boxed{0} & 1 & \boxed{0} & \boxed{0} & 1 & 1 & \boxed{0} & \boxed{0} \\ \boxed{0} & 0 & \boxed{1} & \boxed{0} & 0 & 1 & \boxed{1} & \boxed{0} \\ \boxed{0} & 0 & \boxed{0} & \boxed{1} & 0 & 0 & \boxed{1} & \boxed{1} \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{e}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Note: This is the encoding matrix for the PIR scheme in our example.

PIR matrix and PIR codes

Definition: k-server PIR matrix

*Let $\boldsymbol{e}_i$ denote the binary unit vector with 1 in position i and zeros elsewhere. An $s \times m$ binary matrix G is said to have **property** P_k if for all $i \in [s]$ there exist k disjoint sets of columns of G that add to $\boldsymbol{e}_i$. A matrix that has property P_k is also said to be a **k-server PIR matrix**.*

Example: 4×8 matrix with property P_3

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{e}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Note: This is the encoding matrix for the PIR scheme in our example.

Definition: k-server PIR code

*A binary linear code $\mathbb{C}$ of length m and dimension s will be called a **k-server PIR code** if there exists a generator matrix G for $\mathbb{C}$ with property P_k.*

Recovery equations from PIR codes

Lemma 2: Disjoint recovery sets

Let $\mathbb{C}$ be a k-server PIR code and let G be an $s \times m$ generator matrix for $\mathbb{C}$ with property P_k. Let $\boldsymbol{c} = \boldsymbol{x}G$ be the encoding of a message $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$. Then for all $i \in [s]$, there exist k disjoint recovery sets $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k$ such that

$$x_i = \sum_{j \in \mathcal{R}_1} c_j = \sum_{j \in \mathcal{R}_2} c_j = \cdots = \sum_{j \in \mathcal{R}_k} c_j$$

Proof. Let $g_1, g_2, \ldots, g_m$ denote the columns of G. Then $\boldsymbol{c} = \boldsymbol{x}G$ can be written in terms of the inner products of these columns with $\boldsymbol{x}$, as follows:

$$\boldsymbol{c} = (c_1, c_2, \ldots, c_m) = (\langle \boldsymbol{x}, g_1 \rangle, \langle \boldsymbol{x}, g_2 \rangle, \ldots, \langle \boldsymbol{x}, g_m \rangle)$$

Now suppose that for some set of indices $\mathcal{R} = \{j_1, j_2, \ldots, j_r\} \subseteq [m]$, the corresponding columns of G add to the unit vector $\boldsymbol{e}_i$. Then

$$c_{j_1} + \cdots + c_{j_r} = \langle \boldsymbol{x}, g_{j_1} \rangle + \cdots + \langle \boldsymbol{x}, g_{j_r} \rangle = \langle \boldsymbol{x}, g_{j_1} + \cdots + g_{j_r} \rangle = \langle \boldsymbol{x}, \boldsymbol{e}_i \rangle = x_i$$

It follows from the above that the recovery sets $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k \subseteq [m]$, are simply the indices of the disjoint sets of columns of G that add up to $\boldsymbol{e}_i$. □

Recovery equations from PIR codes

Lemma 2: Disjoint recovery sets

Let $\mathbb{C}$ be a k-server PIR code and let G be an $s \times m$ generator matrix for $\mathbb{C}$ with property P_k. Let $\boldsymbol{c} = \boldsymbol{x}G$ be the encoding of a message $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$. Then for all $i \in [s]$, there exist k disjoint recovery sets $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k$ such that

$$x_i = \sum_{j \in \mathcal{R}_1} c_j = \sum_{j \in \mathcal{R}_2} c_j = \cdots = \sum_{j \in \mathcal{R}_k} c_j$$

Proof. Let $g_1, g_2, \ldots, g_m$ denote the columns of G. Then $\boldsymbol{c} = \boldsymbol{x}G$ can be written in terms of the inner products of these columns with $\boldsymbol{x}$, as follows:

$$\boldsymbol{c} = (c_1, c_2, \ldots, c_m) = (\langle \boldsymbol{x}, g_1 \rangle, \langle \boldsymbol{x}, g_2 \rangle, \ldots, \langle \boldsymbol{x}, g_m \rangle)$$

Now suppose that for some set of indices $\mathcal{R} = \{j_1, j_2, \ldots, j_r\} \subseteq [m]$, the corresponding columns of G add to the unit vector $\boldsymbol{e}_i$. Then

$$c_{j_1} + \cdots + c_{j_r} = \langle \boldsymbol{x}, g_{j_1} \rangle + \cdots + \langle \boldsymbol{x}, g_{j_r} \rangle = \langle \boldsymbol{x}, g_{j_1} + \cdots + g_{j_r} \rangle = \langle \boldsymbol{x}, \boldsymbol{e}_i \rangle = x_i$$

It follows from the above that the recovery sets $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k \subseteq [m]$, are simply the indices of the disjoint sets of columns of G that add up to $\boldsymbol{e}_i$. □

Construction of coded PIR schemes

Theorem 3: Coded PIR schemes from PIR codes

Suppose there exists a ***k-server PIR code $\mathbb{C}$ of length m and dimension s*** *and a k-server linear PIR protocol $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$. Then there exists a* ***coded PIR scheme with s parts and m shares*** *along with the corresponding coded PIR protocol.*

Proof. Let G be a generator matrix for $\mathbb{C}$ with property P_k. Then the coded shares are computed from the database parts $x_1, x_2, \ldots, x_s$ as follows:

$$(c_1, c_2, \ldots, c_m) = (x_1, x_2, \ldots, x_s)\, G$$

Assume Alice wishes to read the i-th bit from the ℓ-th part of the database, namely the bit $x_{\ell,i}$ for some $i \in [n/s]$. She will proceed as follows.

1. Alice invokes the query algorithm of $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ to generate k randomized queries $q_1, q_2, \ldots, q_k := \mathcal{Q}(k, n/s; i)$.
2. She next finds k disjoint recovery sets $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k \subseteq [m]$ such that

$$x_\ell = \sum_{j\in\mathcal{R}_1} c_j = \sum_{j\in\mathcal{R}_2} c_j = \cdots = \sum_{j\in\mathcal{R}_k} c_j$$

Such sets exist by Lemma 2. They are used to determine how to assign the queries $q_1, q_2, \ldots, q_k$ to the servers $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$.

Construction of coded PIR schemes

Theorem 3: Coded PIR schemes from PIR codes

Suppose there exists a ***k-server PIR code $\mathbb{C}$ of length m and dimension s*** *and a k-server linear PIR protocol $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$. Then there exists a* ***coded PIR scheme with s parts and m shares*** *along with the corresponding coded PIR protocol.*

Proof. Let G be a generator matrix for $\mathbb{C}$ with property P_k. Then the coded shares are computed from the database parts $x_1, x_2, \ldots, x_s$ as follows:

$$(c_1, c_2, \ldots, c_m) = (x_1, x_2, \ldots, x_s)\, G$$

Assume Alice wishes to read the i-th bit from the ℓ-th part of the database, namely the bit $x_{\ell,i}$ for some $i \in [n/s]$. She will proceed as follows.

1. Alice invokes the query algorithm of $\mathcal{P}(\mathcal{Q},\mathcal{A},\mathcal{C})$ to generate k randomized queries $q_1, q_2, \ldots, q_k := \mathcal{Q}(k, n/s; i)$.
2. She next finds k disjoint recovery sets $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k \subseteq [m]$ such that

$$x_\ell = \sum_{j\in\mathcal{R}_1} c_j = \sum_{j\in\mathcal{R}_2} c_j = \cdots = \sum_{j\in\mathcal{R}_k} c_j$$

Such sets exist by Lemma 2. They are used to determine how to assign the queries $q_1, q_2, \ldots, q_k$ to the servers $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$.

Proof of main theorem ...continued

3. Let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cdots \cup \mathcal{R}_k$ be the union of the k recovery sets. For each $j \in \mathcal{R}$, Alice finds the unique $t \in [k]$ such that $j \in \mathcal{R}_t$ and sets $q_j^* = q_t$. For $j \notin \mathcal{R}$, the query q_j^* can be set arbitrarily (say $q_j^* = q_1$), since the response from $\mathcal{S}_j$ will be ignored. Alice sends the queries to servers as follows:

$$(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m) \longleftarrow (q_1^*, q_2^*, \ldots, q_m^*)$$

Note: *The privacy of the queries $q_1^*, q_2^*, \ldots, q_m^*$ is inherited from the original PIR protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ being emulated.*

4. Alice collects the answers $a_j = \mathcal{A}(k, j; \boldsymbol{c}_j, q_j^*) = \mathcal{A}(k, t; \boldsymbol{c}_j, q_t)$ from the servers, for all $j \in \mathcal{R}$, and computes:

$$a'_t \stackrel{\text{def}}{=} \sum_{j \in \mathcal{R}_t} \mathcal{A}(k, t; \boldsymbol{c}_j, q_t) = \mathcal{A}\left(k, t; \textstyle\sum_{j \in \mathcal{R}_t} \boldsymbol{c}_j, q_t\right) = \mathcal{A}(k, t; \boldsymbol{x}_\ell, q_t)$$

for $t = 1, 2, \ldots, k$, where the first equality follows from the linearity of the answer algorithm $\mathcal{A}$ and the second from the recovery equations for $\boldsymbol{x}_\ell$.

5. Alice completes the retrieval by invoking the reconstruction algorithm of the emulated protocol $\mathcal{P}(\mathcal{Q}, \mathcal{A}, \mathcal{C})$ as follows:

$$\mathcal{C}(k, n/s; i, a'_1, \ldots, a'_k) = \mathcal{C}(k, n/s; i, \mathcal{A}(k, 1; \boldsymbol{x}_\ell, q_1), \ldots, \mathcal{A}(k, k; \boldsymbol{x}_\ell, q_k)) = x_{\ell,i}$$

□

What about communication cost?

In order to reduce storage overhead, we emulate a conventional PIR protocol $\mathcal{P}$ by a coded PIR protocol $\mathcal{P}^*$. How much do we pay in communication complexity?

$$U(\mathcal{P};n) \stackrel{\text{def}}{=} \textit{Worst-case total number of bits uploaded by a protocol } \mathcal{P} \textit{ for a database of length } n$$

$$D(\mathcal{P};n) \stackrel{\text{def}}{=} \textit{Worst-case total number of bits downloaded by a protocol } \mathcal{P} \textit{ for a database of length } n$$

Theorem 4: Communication complexity of coded PIR

Suppose there exists a k-server PIR code $\mathbb{C}$ of length m and dimension s. Then any linear k-server PIR protocol $\mathcal{P}$ can be emulated by a coded PIR protocol $\mathcal{P}^*$ with s parts and m shares, having communication complexity:

$$U(\mathcal{P}^*;n) \leqslant \frac{m}{k}\,U(\mathcal{P};n/s) + m\log k \quad \text{and} \quad D(\mathcal{P}^*;n) \leqslant \frac{m}{k}\,D(\mathcal{P};n/s)$$

Proof. On the upload side, the number of queries increases from k to m, but each query is shorter as it is generated by $\mathcal{Q}(k,n/s;i)$ rather than $\mathcal{Q}(k,n;i)$. On the download side, the number of answers also increases from k to m. □

What about communication cost?

In order to reduce storage overhead, we emulate a conventional PIR protocol $\mathcal{P}$ by a coded PIR protocol $\mathcal{P}^*$. How much do we pay in communication complexity?

$U(\mathcal{P};n) \stackrel{\text{def}}{=}$ *Worst-case total number of bits uploaded by a protocol $\mathcal{P}$ for a database of length n*

$D(\mathcal{P};n) \stackrel{\text{def}}{=}$ *Worst-case total number of bits downloaded by a protocol $\mathcal{P}$ for a database of length n*

Theorem 4: Communication complexity of coded PIR

Suppose there exists a k-server PIR code $\mathbb{C}$ of length m and dimension s. Then any linear k-server PIR protocol $\mathcal{P}$ can be emulated by a coded PIR protocol $\mathcal{P}^*$ with s parts and m shares, having communication complexity:

$$U(\mathcal{P}^*;n) \leqslant \frac{m}{k}\, U(\mathcal{P};n/s) + m\log k \quad \text{and} \quad D(\mathcal{P}^*;n) \leqslant \frac{m}{k}\, D(\mathcal{P};n/s)$$

Proof. On the upload side, the number of queries increases from k to m, but each query is shorter as it is generated by $\mathcal{Q}(k,n/s;i)$ rather than $\mathcal{Q}(k,n;i)$. On the download side, the number of answers also increases from k to m. □

Summary of our results so far

We have shown that:

existing k-server linear PIR protocol $\mathcal{P}$

+

k-server PIR code $\mathbb{C}$ of length m and dimension s

=

coded k-server PIR protocol $\mathcal{P}^*$ with storage overhead m/s

Summary of our results so far

We have shown that:

- exis… protocol $\mathcal{P}$
- k-server… on s
- coded k-se… rhead m/s

Summary of our results so far

- Why does the bit retrieval in the example work? Why does everything cancel out?
- For which m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about the communication complexity of coded PIR schemes?

We have shown that:

existing k-server linear PIR protocol $\mathcal{P}$

+

k-server PIR code $\mathbb{C}$ of length m and dimension s

=

coded k-server PIR protocol $\mathcal{P}^*$ with storage overhead m/s

Summary of our results so far

- Why does the bit retrieval in the example work? Why does everything cancel out? ✔
- For which m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about the communication complexity of coded PIR schemes? ✔

We have shown that:

existing k-server linear PIR protocol $\mathcal{P}$

+

k-server PIR code $\mathbb{C}$ of length m and dimension s

=

coded k-server PIR protocol $\mathcal{P}^*$ with storage overhead m/s

Summary of our results so far

- Why does the bit retrieval in the example work? Why does everything cancel out? ✔
- For which m, s, and k do coded k-server PIR schemes with s parts and m shares exist?
- What about the communication complexity of coded PIR schemes? ✔

How small can we make the storage overhead ratio m/s?

We have shown that:

existing k-server linear PIR protocol $\mathcal{P}$

+

k-server PIR code $\mathbb{C}$ of length m and dimension s

=

coded k-server PIR protocol $\mathcal{P}^*$ with storage overhead m/s

New problem: High-rate PIR codes

According to our construction, coded PIR schemes exist whenever PIR codes exist. The storage overhead of such coded PIR schemes is completely determined by the rate of the underlying PIR code.

Open Problem: Given positive integers s and k, determine the smallest m such that there exists a k-server PIR code of length m and dimension s.

$$M(s,k) \stackrel{\text{def}}{=} \textit{Shortest possible length } m \textit{ of a } k\textit{-server PIR code of dimension } s$$

$$\rho(s,k) \stackrel{\text{def}}{=} \textit{Smallest possible redundancy of a } k\textit{-server PIR code of dimension } s$$

With this notation:

$$\textbf{storage overhead} = \frac{M(s,k)}{s} = 1 + \frac{\rho(s,k)}{s}$$

New problem: High-rate PIR codes

According to our construction, coded PIR schemes exist whenever PIR codes exist. The storage overhead of such coded PIR schemes is completely determined by the rate of the underlying PIR code.

Open Problem: Given positive integers s and k, determine the smallest m such that there exists a k-server PIR code of length m and dimension s.

$$M(s,k) \stackrel{\text{def}}{=} \textit{Shortest possible length } m \textit{ of a } k\textit{-server PIR code of dimension } s$$

$$\rho(s,k) \stackrel{\text{def}}{=} \textit{Smallest possible redundancy of a } k\textit{-server PIR code of dimension } s$$

With this notation:

$$\textbf{storage overhead} = \frac{M(s,k)}{s} = 1 + \frac{\rho(s,k)}{s}$$

New problem: High-rate PIR codes

According to our construction, coded PIR schemes exist whenever PIR codes exist. The storage overhead of such coded PIR schemes is completely determined by the rate of the underlying PIR code.

Open Problem: Given positive integers s and k, determine the smallest m such that there exists a k-server PIR code of length m and dimension s.

$$M(s,k) \stackrel{\text{def}}{=} \textit{Shortest possible length } m \textit{ of a } k\textit{-server PIR code of dimension } s$$

$$\rho(s,k) \stackrel{\text{def}}{=} \textit{Smallest possible redundancy of a } k\textit{-server PIR code of dimension } s$$

With this notation:

$$\textbf{storage overhead} = \frac{M(s,k)}{s} = 1 + \frac{\rho(s,k)}{s}$$

We have converted a PIR problem to a coding theory problem!

Optimal solution for two servers

For $k = 2$, the coding-theory problem is trivial. The single parity-check code of dimension s is a 2-server PIR code, and therefore:

$$M(s,2) = s+1 \qquad \rho(s,2) = 1$$

Why is this true? The encoding of each message $x = (x_1, x_2, \ldots, x_s)$ consists of appending an overall parity bit

$$c = x_1 + x_2 + \cdots + x_s$$

Thus for all $i \in [s]$, we have $x_i = x_1 + \cdots + x_{i-1} + c + x_{i+1} + \cdots + x_s$. This corresponds to two disjoint recovery sets $\mathcal{R}_1 = \{i\}$ and $\mathcal{R}_2 = [s+1] \setminus \{i\}$.

Optimal solution for two servers

For $k = 2$, the coding-theory problem is trivial. The single parity-check code of dimension s is a 2-server PIR code, and therefore:

$$M(s,2) = s+1 \qquad \rho(s,2) = 1$$

Why is this true? The encoding of each message $x = (x_1, x_2, \ldots, x_s)$ consists of appending an overall parity bit

$$c = x_1 + x_2 + \cdots + x_s$$

Thus for all $i \in [s]$, we have $x_i = x_1 + \cdots + x_{i-1} + c + x_{i+1} + \cdots + x_s$. This corresponds to two disjoint recovery sets $\mathcal{R}_1 = \{i\}$ and $\mathcal{R}_2 = [s+1] \setminus \{i\}$.

Optimal solution for two servers

For $k = 2$, the coding-theory problem is trivial. The single parity-check code of dimension s is a 2-server PIR code, and therefore:

$$M(s,2) = s+1 \qquad \rho(s,2) = 1$$

Why is this true? The encoding of each message $x = (x_1, x_2, \ldots, x_s)$ consists of appending an overall parity bit

$$c = x_1 + x_2 + \cdots + x_s$$

Thus for all $i \in [s]$, we have $x_i = x_1 + \cdots + x_{i-1} + c + x_{i+1} + \cdots + x_s$. This corresponds to two disjoint recovery sets $\mathcal{R}_1 = \{i\}$ and $\mathcal{R}_2 = [s+1] \setminus \{i\}$.

Theorem 5: PIR without storage overhead

For all $\varepsilon > 0$ it is possible to achieve information-theoretic PIR with communication complexity $n^{o(1)}$ by storing at most $(1+\varepsilon)n$ bits.

Proof. Take $s = 1/\varepsilon$, and combine our results for $k = 2$ with the results of Dvir-Gopi on 2-server PIR with subpolynomial communication. □

PIR codes for multiple servers

Open Problem: Can we achieve information-theoretic PIR with low communication cost *without doubling the number of bits we need to store?*

PIR codes for multiple servers

Open Problem: Can we achieve information-theoretic PIR with low communication cost *without doubling the number of bits we need to store?*

Any reason to go on... Why not stop here?

PIR codes for multiple servers

Open Problem: Can we achieve information-theoretic PIR with low communication cost *without doubling the number of bits we need to store?*

Any reason to go on... Why not stop here?

- As the number of servers k grows, the **communication complexity improves** dramatically:

$$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)} \xrightarrow{k \text{ large}} \text{polylog}(n)$$

PIR codes for multiple servers

Open Problem: Can we achieve information-theoretic PIR with low communication cost *without doubling the number of bits we need to store?*

Any reason to go on... Why not stop here?

- As the number of servers k grows, the **communication complexity improves** dramatically:

$$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)} \xrightarrow{k \text{ large}} \operatorname{polylog}(n)$$

- The coding-theory problem of determining $M(s,k)$ becomes **much more interesting for** $k \geqslant 3$. It has strong connections with:
  - Steiner systems and t-designs
  - majority-logic decodable codes
  - local codes with availability
  - multiset batch codes
  - bipartite graphs of girth 6
  - constant-weight codes

PIR codes for multiple servers

Open Problem: Can we achieve information-theoretic PIR with low communication cost *without doubling the number of bits we need to store?*

Any reason to go on... Why not stop here?

- As the number of servers k grows, the **communication complexity improves** dramatically:

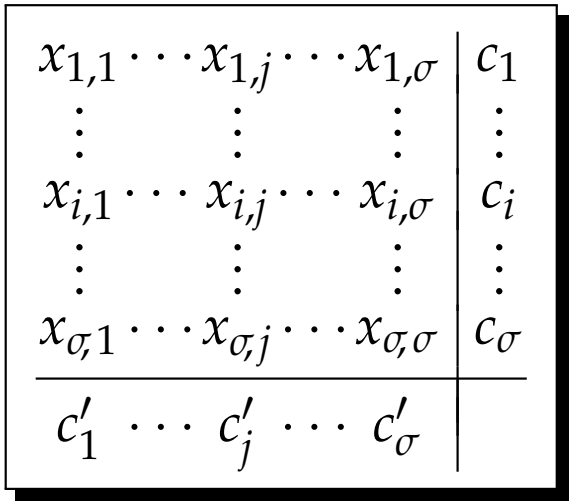
$$n^{O\left(\sqrt{\frac{\log\log n}{\log n}}\right)} \xrightarrow{k \text{ large}} \operatorname{polylog}(n)$$

- The coding-theory problem of determining $M(s,k)$ becomes **much more interesting for** $k \geqslant 3$. It has strong connections with:
  - Steiner systems and t-designs
  - majority-logic decodable codes
  - local codes with availability
  - multiset batch codes
  - bipartite graphs of girth 6
  - constant-weight codes

The hypercube construction

Suppose that $k = 3$ and $s = \sigma^2$ for some $\sigma \in \mathbb{Z}$. Arrange the σ^2 message bits in the form of a $\sigma \times \sigma$ ***square***. To every message, we append 2σ parity bits given by:

$$c_i = x_{i,1} + x_{i,2} + \cdots + x_{i,\sigma} \qquad \text{for } i \in [\sigma]$$
$$c'_j = x_{1,j} + x_{2,j} + \cdots + x_{\sigma,j} \qquad \text{for } j \in [\sigma]$$



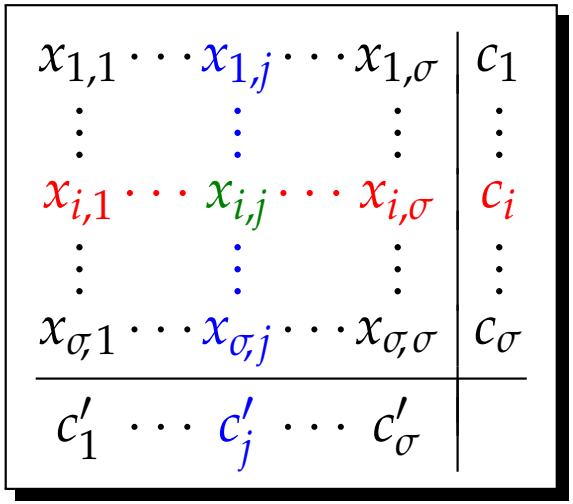
The hypercube construction

Suppose that $k = 3$ and $s = \sigma^2$ for some $\sigma \in \mathbb{Z}$. Arrange the σ^2 message bits in the form of a $\sigma \times \sigma$ ***square***. To every message, we append 2σ parity bits given by:

$$c_i = x_{i,1} + x_{i,2} + \cdots + x_{i,\sigma} \qquad \text{for } i \in [\sigma]$$
$$c'_j = x_{1,j} + x_{2,j} + \cdots + x_{\sigma,j} \qquad \text{for } j \in [\sigma]$$

Then for each message bit $x_{i,j}$ we have ***three disjoint recovery equations*** given by $x_{i,j}$ itself and:

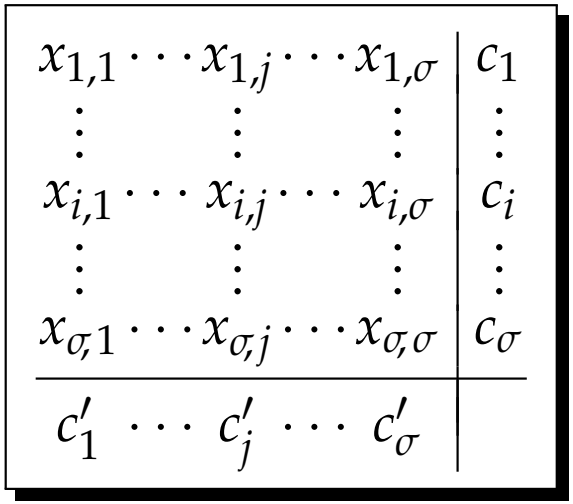
$$x_{i,1} + \cdots + x_{i,j-1} + c_i + x_{i,j+1} + \cdots + x_{i,\sigma} = x_{1,j} + \cdots + x_{i-1,j} + c'_j + x_{i+1,j} + \cdots + x_{\sigma,j}$$



The hypercube construction

Suppose that $k = 3$ and $s = \sigma^2$ for some $\sigma \in \mathbb{Z}$. Arrange the σ^2 message bits in the form of a $\sigma \times \sigma$ ***square***. To every message, we append 2σ parity bits given by:

$$c_i = x_{i,1} + x_{i,2} + \cdots + x_{i,\sigma} \qquad \text{for } i \in [\sigma]$$
$$c'_j = x_{1,j} + x_{2,j} + \cdots + x_{\sigma,j} \qquad \text{for } j \in [\sigma]$$



Then for each message bit $x_{i,j}$ we have ***three disjoint recovery equations*** given by $x_{i,j}$ itself and:

$$x_{i,1} + \cdots + x_{i,j-1} + c_i + x_{i,j+1} + \cdots + x_{i,\sigma} = x_{1,j} + \cdots + x_{i-1,j} + c'_j + x_{i+1,j} + \cdots + x_{\sigma,j}$$

More generally, we arrange σ^{k-1} message bits in the form of a $(k-1)$***-dimensional hypercube*** and append a parity bit to each of its $(k-1)\sigma^{k-2}$ columns. This proves:

$$M(s,k) = s + (k-1)\left\lceil \sqrt[k-1]{s} \right\rceil^{k-2}$$

$$\rho(s,k) \leqslant (k-1)\left\lceil \sqrt[k-1]{s} \right\rceil^{k-2}$$

The hypercube construction

Suppose that $k = 3$ and $s = \sigma^2$ for some $\sigma \in \mathbb{Z}$. Arrange the σ^2 message bits in the form of a $\sigma \times \sigma$ ***square***. To every message, we append 2σ parity bits given by:

$$c_i = x_{i,1} + x_{i,2} + \cdots + x_{i,\sigma} \qquad \text{for } i \in [\sigma]$$
$$c'_j = x_{1,j} + x_{2,j} + \cdots + x_{\sigma,j} \qquad \text{for } j \in [\sigma]$$

$$\begin{array}{c|c} \begin{matrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,\sigma} \\ \vdots & & \vdots & & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,\sigma} \\ \vdots & & \vdots & & \vdots \\ x_{\sigma,1} & \cdots & x_{\sigma,j} & \cdots & x_{\sigma,\sigma} \end{matrix} & \begin{matrix} c_1 \\ \vdots \\ c_i \\ \vdots \\ c_\sigma \end{matrix} \\ \hline \begin{matrix} c'_1 & \cdots & c'_j & \cdots & c'_\sigma \end{matrix} & \end{array}$$

Then for each message bit $x_{i,j}$ we have ***three disjoint recovery equations*** given by $x_{i,j}$ itself and:

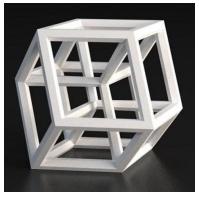
$$x_{i,1} + \cdots + x_{i,j-1} + c_i + x_{i,j+1} + \cdots + x_{i,\sigma} = x_{1,j} + \cdots + x_{i-1,j} + c'_j + x_{i+1,j} + \cdots + x_{\sigma,j}$$

More generally, we arrange σ^{k-1} message bits in the form of a $(k-1)$***-dimensional hypercube*** and append a parity bit to each of its $(k-1)\sigma^{k-2}$ columns. This proves:

$$M(s,k) = s + (k-1)\left\lceil \sqrt[k-1]{s} \right\rceil^{k-2}$$

$$\rho(s,k) \leqslant (k-1)\left\lceil \sqrt[k-1]{s} \right\rceil^{k-2}$$

It follows that $\lim_{s\to\infty} M(s,k)/s = 1$ for all fixed $k \geqslant 2$. Therefore, we have proved:



Corollary 6: Multiple-server PIR without storage overhead

For ***all fixed*** $k \geqslant 2$ *and all* $\varepsilon > 0$*, there exist* k***-server*** *coded PIR schemes that store at most* $(1+\varepsilon)n$ *bits.*

Majority-logic decodable codes

Majority-logic decoding originated with the work of Reed and Massey over 50 years ago. 100s of papers in the 1960s and 1970s... now forgotten.

Majority-logic decodable codes

Majority-logic decoding originated with the work of Reed and Massey over 50 years ago. 100s of papers in the 1960s and 1970s... now forgotten.

Definition: Majority-logic decodable codes

*A linear code $\mathbb{C}$ of length n is **majority-logic decodable** with parameter J iff for each position $i \in [n]$, there exist J **parity-checks orthogonal on this position:***

$$
\begin{array}{c}
\begin{array}{c} i \end{array} \\
\hline
\left.\begin{array}{l}
11\ldots11 \\
1\,11\ldots1 \\
\vdots \quad \ddots \\
1 \quad\quad 11\ldots1 \\
1 \quad\quad\quad\quad 11\ldots1
\end{array}\right\} J \text{ codewords of } \mathbb{C}^{\perp}
\end{array}
$$

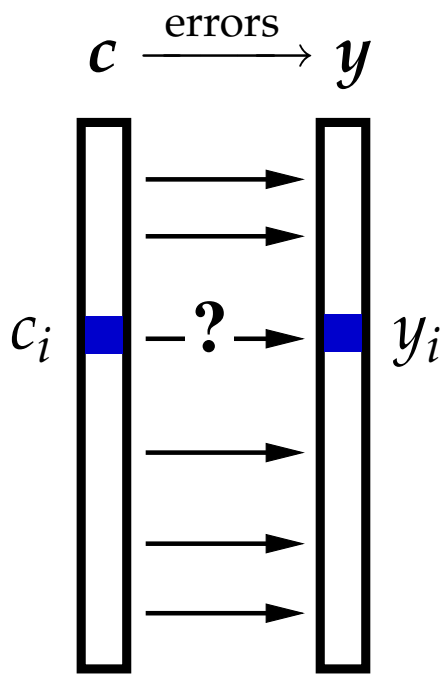
Majority-logic decodable codes

Majority-logic decoding originated with the work of Reed and Massey over 50 years ago. 100s of papers in the 1960s and 1970s... now forgotten.

Definition: Majority-logic decodable codes

*A linear code $\mathbb{C}$ of length n is **majority-logic decodable** with parameter J iff for each position $i \in [n]$, there exist J **parity-checks orthogonal on this position:***

$$\begin{array}{c} \overset{\overset{i}{\downarrow}}{} \\ \left.\begin{array}{l} 11\ldots11 \\ 1\,11\ldots1 \\ \vdots \\ 1 \\ 1 \end{array}\right\} \begin{array}{l} \\ \\ \ddots \\ 11\ldots1 \\ 11\ldots1 \end{array} \end{array} \quad J \text{ codewords of } \mathbb{C}^{\perp}$$



Majority-logic decoding: Given y, evaluate the J orthogonal parity-checks for each position i.

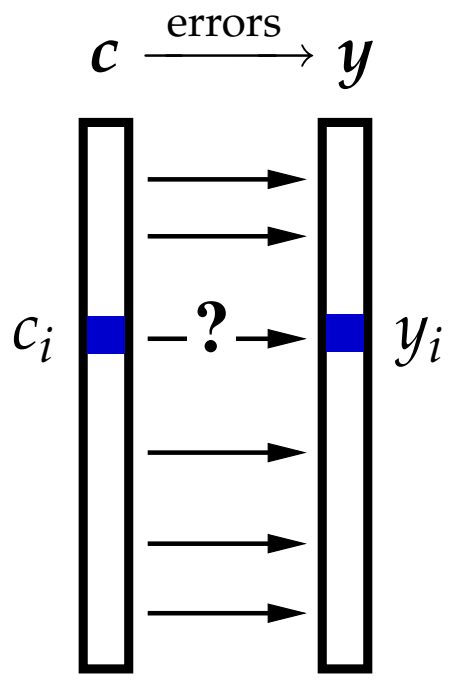
Majority-logic decodable codes

Majority-logic decoding originated with the work of Reed and Massey over 50 years ago. 100s of papers in the 1960s and 1970s... now forgotten.

Definition: Majority-logic decodable codes

*A linear code $\mathbb{C}$ of length n is **majority-logic decodable** with parameter J iff for each position $i \in [n]$, there exist J **parity-checks orthogonal on this position:***

$$\begin{array}{c|ccccc} & i & & & & \\ \hline 11\ldots11 & & & & & \\ & 1 & 11\ldots1 & & & \\ & \vdots & & \ddots & & \\ & 1 & & & 11\ldots1 & \\ & 1 & & & & 11\ldots1 \end{array} \Bigg\} \; J \text{ codewords of } \mathbb{C}^{\perp}$$



Majority-logic decoding: Given y, evaluate the J orthogonal parity-checks for each position i. Then:

- $y_i = c_i$ and $\leqslant t$ other errors $\Longrightarrow$ at least $J - t$ **checks evaluate to** 0

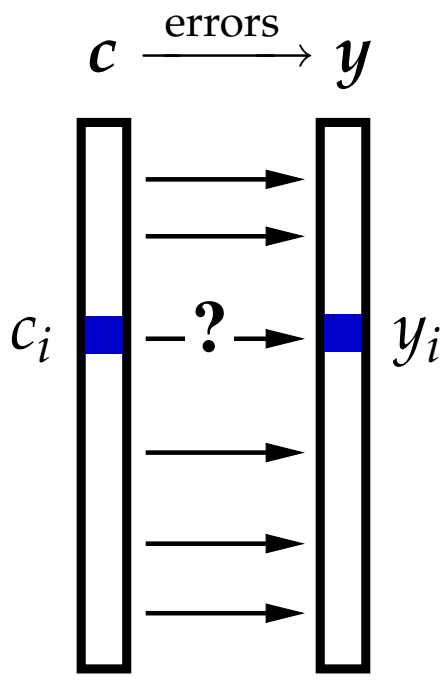
Majority-logic decodable codes

Majority-logic decoding originated with the work of Reed and Massey over 50 years ago. 100s of papers in the 1960s and 1970s... now forgotten.

Definition: Majority-logic decodable codes

*A linear code $\mathbb{C}$ of length n is **majority-logic decodable** with parameter J iff for each position $i \in [n]$, there exist J **parity-checks orthogonal on this position:***

$$\begin{array}{c} \overset{\overset{i}{\downarrow}}{\begin{array}{l} 11\ldots11 \\ \quad 1\,11\ldots1 \\ \quad \vdots \quad \ddots \\ \quad 1 \quad\quad 11\ldots1 \\ \quad 1 \quad\quad\quad\quad 11\ldots1 \end{array}} \end{array} \Bigg\} \; J \text{ codewords of } \mathbb{C}^{\perp}$$



Majority-logic decoding: Given y, evaluate the J orthogonal parity-checks for each position i. Then:

- $y_i = c_i$ and $\leqslant t$ other errors $\Longrightarrow$ at least $J - t$ **checks evaluate to** 0
- $y_i \neq c_i$ and $\leqslant t$ other errors $\Longrightarrow$ at least $J - t$ **checks evaluate to** 1

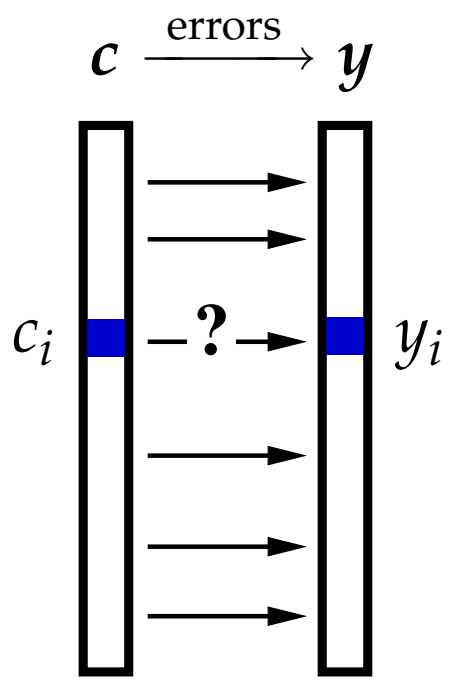
Majority-logic decodable codes

Majority-logic decoding originated with the work of Reed and Massey over 50 years ago. 100s of papers in the 1960s and 1970s... now forgotten.

Definition: Majority-logic decodable codes

A linear code $\mathbb{C}$ of length n is ***majority-logic decodable*** *with parameter J iff for each position $i \in [n]$, there exist J* ***parity-checks orthogonal on this position:***

$$\begin{array}{c|c} & i \\ \hline 11\ldots11 & \\ 1 & 11\ldots1 \\ \vdots & \ddots \\ 1 & 11\ldots1 \\ 1 & 11\ldots1 \end{array} \Bigg\} J \text{ codewords of } \mathbb{C}^{\perp}$$



Majority-logic decoding: Given y, evaluate the J orthogonal parity-checks for each position i. Then:

- $y_i = c_i$ and $\leqslant t$ other errors $\implies$ at least $J - t$ **checks evaluate to** 0
- $y_i \neq c_i$ and $\leqslant t$ other errors $\implies$ at least $J - t$ **checks evaluate to** 1

There is an error at position i iff a ***majority of the J checks*** evaluate to 1.

PIR codes from majority-logic codes

Lemma 7: PIR codes from majority-logic codes

Let $\mathbb{C}$ be a majority-logic decodable code with parameter J. Then $\mathbb{C}$ is also a k-server PIR code with $k = J + 1$.

Proof. It is easy to see that a ***systematic generator matrix*** G for $\mathbb{C}$ has property P_k with $k = J + 1$. Since G is systematic, the column in position i is e_i.

$$G = \left[\begin{array}{c|c|c|c|c|c} & e_i & & & & \end{array}\right]$$

$$\left.\begin{array}{l} 11\ldots1\ 1 \\ 1\ 11\ldots1 \\ \vdots \\ 1 \\ 1 \end{array}\right\} \begin{array}{l} \\ \\ \ddots \\ 11\ldots1 \\ 11\ldots1 \end{array} \quad J \text{ codewords of } \mathbb{C}^{\perp}$$

PIR codes from majority-logic codes

Lemma 7: PIR codes from majority-logic codes

Let $\mathbb{C}$ be a majority-logic decodable code with parameter J. Then $\mathbb{C}$ is also a k-server PIR code with $k = J + 1$.

Proof. It is easy to see that a ***systematic generator matrix*** G for $\mathbb{C}$ has property P_k with $k = J + 1$. Since G is systematic, the column in position i is $\boldsymbol{e}_i$.

$$G = \left[\begin{array}{c|c|c|c|c|c} & \boldsymbol{e}_i & & & & \end{array}\right]$$

$$\left.\begin{array}{l} 11\ldots1\ 1 \\ \quad 1\ 11\ldots1 \\ \quad\ \vdots \qquad\quad \ddots \\ \quad\ 1 \qquad\qquad\quad 11\ldots1 \\ \quad\ 1 \qquad\qquad\qquad\quad 11\ldots1 \end{array}\right\} J \text{ codewords of } \mathbb{C}^{\perp}$$

PIR codes from majority-logic codes

Lemma 7: PIR codes from majority-logic codes

Let $\mathbb{C}$ be a majority-logic decodable code with parameter J. Then $\mathbb{C}$ is also a k-server PIR code with $k = J + 1$.

Proof. It is easy to see that a ***systematic generator matrix*** G for $\mathbb{C}$ has property P_k with $k = J + 1$. Since G is systematic, the column in position i is $\boldsymbol{e}_i$.

$$G = \begin{array}{|c|c|c|c|c|c|} \hline \mathcal{R}_1 & \boldsymbol{e}_i & \mathcal{R}_2 & \cdots & \mathcal{R}_{J-1} & \mathcal{R}_J \\ \hline \end{array}$$

$$\left.\begin{array}{l} 11\ldots1\ 1 \\ \quad\ 1\ 11\ldots1 \\ \quad\ \vdots \qquad\quad \ddots \\ \quad\ 1 \qquad\qquad\quad 11\ldots1 \\ \quad\ 1 \qquad\qquad\qquad\quad 11\ldots1 \end{array}\right\} J \text{ codewords of } \mathbb{C}^{\perp}$$

Thus $\{\boldsymbol{e}_i\}$ and $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_J$ are ***disjoint sets of columns*** of G that add to $\boldsymbol{e}_i$. $\square$

PIR codes from majority-logic codes

Lemma 7: PIR codes from majority-logic codes

Let $\mathbb{C}$ be a majority-logic decodable code with parameter J. Then $\mathbb{C}$ is also a k-server PIR code with $k = J + 1$.

Numerous algebraic constructions of cyclic majority-logic decodable codes are known. For example, Reed-Muller codes, BCH codes, and other codes invariant under the ***group of affine permutations:***

$$\alpha^i \mapsto \beta\alpha^i + \gamma \qquad \text{for all } i = 0, 1, \ldots, 2^m - 2 \text{ and } \beta, \gamma \in \mathrm{GF}(2^m)$$

T. Kasami, S. Lin, and W.W. Peterson, Some results on cyclic codes which are invariant under the affine group, *Information and Control*, vol. **2**, pp. 475–496, November 1968.

Theorem: Doubly-transitive majority-logic codes

Let $n = 2^{2ab} - 1$ and let $\mathbb{C}$ be a binary cyclic code of length n and co-dimension $(2^{b+1} - 1)^a - 1$. Then $\mathbb{C}$ is majority-logic decodable with parameter $J = 2^a + 1$.

PIR codes from majority-logic codes

Lemma 7: PIR codes from majority-logic codes

Let $\mathbb{C}$ be a majority-logic decodable code with parameter J. Then $\mathbb{C}$ is also a k-server PIR code with $k = J + 1$.

Numerous algebraic constructions of cyclic majority-logic decodable codes are known. For example, Reed-Muller codes, BCH codes, and other codes invariant under the ***group of affine permutations:***

$$\alpha^i \mapsto \beta\alpha^i + \gamma \qquad \text{for all } i = 0, 1, \ldots, 2^m - 2 \text{ and } \beta, \gamma \in \mathrm{GF}(2^m)$$

T. Kasami, S. Lin, and W.W. Peterson, Some results on cyclic codes which are invariant under the affine group, *Information and Control*, vol. **2**, pp. 475–496, November 1968.

Theorem: Doubly-transitive majority-logic codes

Let $n = 2^{2ab} - 1$ and let $\mathbb{C}$ be a binary cyclic code of length n and co-dimension $(2^{b+1} - 1)^a - 1$. Then $\mathbb{C}$ is majority-logic decodable with parameter $J = 2^a + 1$.

As a corollary to this theorem and Lemma 7, whenever the number of servers is of the form $k = 4, 6, 10, \ldots, 2^a + 2$, we have:

$$\rho(s, k) = O\left(\sqrt{s}\right)$$

Construction from certain set systems

Definition: Almost disjoint k-covers

*Let $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ be a collection of subsets of $[s]$. We say that $\mathcal{A}$ is a **k-cover of** $[s]$ if every $i \in [s]$ belongs to at least k of the subsets in $\mathcal{A}$. We say that these subsets are **almost disjoint** if any two of them intersect in at most one element.*

Given any collection $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ of subsets of $[s]$, we construct a **systematic** $(s+r, s)$ **linear code** $\mathbb{C}(\mathcal{A})$ as follows. To each message $x = (x_1, x_2, \ldots, x_s)$, we ***append*** r ***parity bits*** given by:

$$c_1 = \sum_{j \in A_1} x_j\,, \quad c_2 = \sum_{j \in A_2} x_j\,, \quad \cdots \quad, c_r = \sum_{j \in A_r} x_j$$

Construction from certain set systems

Definition: Almost disjoint k-covers

*Let $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ be a collection of subsets of $[s]$. We say that $\mathcal{A}$ is a **k-cover of** $[s]$ if every $i \in [s]$ belongs to at least k of the subsets in $\mathcal{A}$. We say that these subsets are **almost disjoint** if any two of them intersect in at most one element.*

Given any collection $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ of subsets of $[s]$, we construct a **systematic $(s+r, s)$ linear code $\mathbb{C}(\mathcal{A})$** as follows. To each message $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$, we ***append*** r ***parity bits*** given by:

$$c_1 = \sum_{j \in A_1} x_j\,, \quad c_2 = \sum_{j \in A_2} x_j\,, \quad \cdots \quad, c_r = \sum_{j \in A_r} x_j$$

Construction from certain set systems

Definition: Almost disjoint k-covers

*Let $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ be a collection of subsets of $[s]$. We say that $\mathcal{A}$ is a **k-cover of** $[s]$ if every $i \in [s]$ belongs to at least k of the subsets in $\mathcal{A}$. We say that these subsets are **almost disjoint** if any two of them intersect in at most one element.*

Given any collection $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ of subsets of $[s]$, we construct a **systematic** $(s+r, s)$ **linear code** $\mathbb{C}(\mathcal{A})$ as follows. To each message $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$, we ***append*** r ***parity bits*** given by:

$$c_1 = \sum_{j \in A_1} x_j\,, \quad c_2 = \sum_{j \in A_2} x_j\,, \quad \cdots \quad , c_r = \sum_{j \in A_r} x_j$$

Lemma 8: PIR codes from almost disjoint k-covers

*Suppose that $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ is a $(k-1)$-cover of $[s]$ and the sets in $\mathcal{A}$ are almost disjoint. Then the resulting $(s+r, s)$ code $\mathbb{C}(\mathcal{A})$ **is a k-server PIR code.***

Construction from certain set systems

Definition: Almost disjoint k-covers

*Let $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ be a collection of subsets of $[s]$. We say that $\mathcal{A}$ is a **k-cover of** $[s]$ if every $i \in [s]$ belongs to at least k of the subsets in $\mathcal{A}$. We say that these subsets are **almost disjoint** if any two of them intersect in at most one element.*

Given any collection $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ of subsets of $[s]$, we construct a **systematic $(s+r, s)$ linear code $\mathbb{C}(\mathcal{A})$** as follows. To each message $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$, we ***append* r *parity bits*** given by:

$$c_1 = \sum_{j \in A_1} x_j\,, \quad c_2 = \sum_{j \in A_2} x_j\,, \quad \cdots \quad , c_r = \sum_{j \in A_r} x_j$$

Lemma 8: PIR codes from almost disjoint k-covers

*Suppose that $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ is a $(k-1)$-cover of $[s]$ and the sets in $\mathcal{A}$ are almost disjoint. Then the resulting $(s+r, s)$ code $\mathbb{C}(\mathcal{A})$ **is a k-server PIR code.***

Proof. Given $i \in [s]$, find $k-1$ subsets in $\mathcal{A}$ that contain i. W.l.o.g., suppose these subsets are $A_1, A_2, \ldots, A_{k-1}$. Let $A'_j = A_j \setminus \{i\}$ for all j. Then the sets $A'_1, A'_2, \ldots, A'_{k-1}$ are disjoint. These sets give rise **to k disjoint recovery equations:**

$$x_i \;=\; c_1 + \sum_{j \in A'_1} x_j \;=\; c_2 + \sum_{j \in A'_2} x_j \;=\; \cdots \;=\; c_{k-1} + \sum_{j \in A'_{k-1}} x_j$$

□

Construction from certain set systems

Definition: Almost disjoint k-covers

*Let $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ be a collection of subsets of $[s]$. We say that $\mathcal{A}$ is a **k-cover of $[s]$** if every $i \in [s]$ belongs to at least k of the subsets in $\mathcal{A}$. We say that these subsets are **almost disjoint** if any two of them intersect in at most one element.*

Given any collection $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ of subsets of $[s]$, we construct a **systematic** $(s+r, s)$ **linear code** $\mathbb{C}(\mathcal{A})$ as follows. To each message $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$, we ***append*** r ***parity bits*** given by:

$$c_1 = \sum_{j \in A_1} x_j\,, \quad c_2 = \sum_{j \in A_2} x_j\,, \quad \cdots \quad , c_r = \sum_{j \in A_r} x_j$$

Lemma 8: PIR codes from almost disjoint k-covers

*Suppose that $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ is a $(k-1)$-cover of $[s]$ and the sets in $\mathcal{A}$ are almost disjoint. Then the resulting $(s+r, s)$ code $\mathbb{C}(\mathcal{A})$ **is a k-server PIR code.***

Corollary 9: PIR codes from almost disjoint k-covers

If there exists an almost disjoint $(k-1)$-cover of $[s]$ with r sets, then $\rho(s,k) \leqslant r$.

Construction from certain set systems

Definition: Almost disjoint k-covers

*Let $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ be a collection of subsets of $[s]$. We say that $\mathcal{A}$ is a **k-cover of $[s]$** if every $i \in [s]$ belongs to at least k of the subsets in $\mathcal{A}$. We say that these subsets are **almost disjoint** if any two of them intersect in at most one element.*

Given any collection $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ of subsets of $[s]$, we construct a **systematic** $(s+r, s)$ **linear code** $\mathbb{C}(\mathcal{A})$ as follows. To each message $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$, we ***append*** r ***parity bits*** given by:

$$c_1 = \sum_{j \in A_1} x_j\,, \quad c_2 = \sum_{j \in A_2} x_j\,, \quad \cdots \quad , c_r = \sum_{j \in A_r} x_j$$

Lemma 8: PIR codes from almost disjoint k-covers

*Suppose that $\mathcal{A} = \{A_1, A_2, \ldots, A_r\}$ is a $(k-1)$-cover of $[s]$ and the sets in $\mathcal{A}$ are almost disjoint. Then the resulting $(s+r, s)$ code $\mathbb{C}(\mathcal{A})$ **is a k-server PIR code.***

Corollary 9: PIR codes from almost disjoint k-covers

If there exists an almost disjoint $(k-1)$-cover of $[s]$ with r sets, then $\rho(s,k) \leqslant r$.

Where can we get almost disjoint k-covers or small size r?

PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2, q, r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example:

	1	2	3	4	5	6	7
B_1:	●	●	●	–	–	–	–
B_2:	–	●	–	●	–	●	–
B_3:	●	–	–	●	●	–	–
B_4:	●	–	–	–	–	●	●
B_5:	–	●	–	–	●	–	●
B_6:	–	–	●	●	–	–	●
B_7:	–	–	●	–	●	●	–

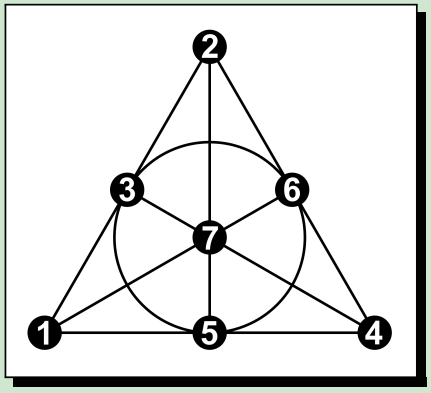
PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2,q,r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example: Fano plane $\mathcal{S}(2,3,7)$

	1	2	3	4	5	6	7
B_1:	●	●	●	–	–	–	–
B_2:	–	●	–	●	–	●	–
B_3:	●	–	–	●	●	–	–
B_4:	●	–	–	–	–	●	●
B_5:	–	●	–	–	●	–	●
B_6:	–	–	●	●	–	–	●
B_7:	–	–	●	–	●	●	–



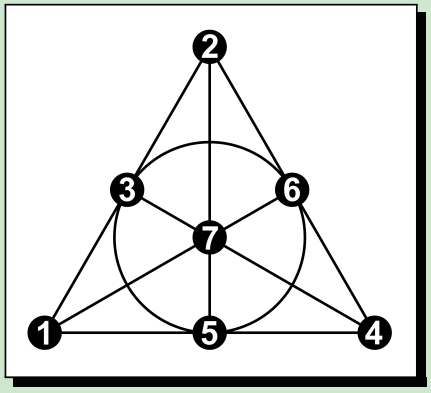
PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2,q,r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example: Fano plane $\mathcal{S}(2,3,7)$

	1	2	3	4	5	6	7
B_1:	●	●	●	–	–	–	–
B_2:	–	●	–	●	–	●	–
B_3:	●	–	–	●	●	–	–
B_4:	●	–	–	–	–	●	●
B_5:	–	●	–	–	●	–	●
B_6:	–	–	●	●	–	–	●
B_7:	–	–	●	–	●	●	–



Observation: There are $b = \binom{r}{2}/\binom{q}{2}$ blocks in $\mathcal{B}$ and each point is contained in $(r-1)/(q-1)$ of them. Moreover, any two blocks intersect in at most one point.

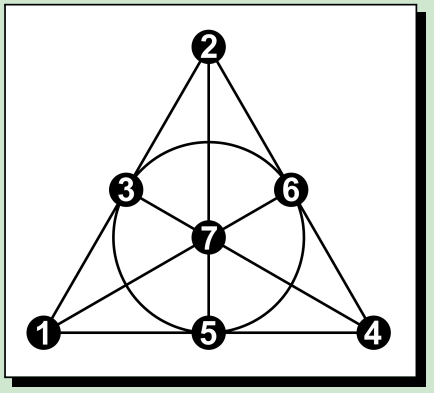
PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2,q,r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example: Fano plane $\mathcal{S}(2,3,7)$

	1	2	3	4	5	6	7
A_1:	●	●	●	–	–	–	–
A_2:	–	●	–	●	–	●	–
A_3:	●	–	–	●	●	–	–
A_4:	●	–	–	–	–	●	●
A_5:	–	●	–	–	●	–	●
A_6:	–	–	●	●	–	–	●
A_7:	–	–	●	–	●	●	–



Observation: There are $b = \binom{r}{2}/\binom{q}{2}$ blocks in $\mathcal{B}$ and each point is contained in $(r-1)/(q-1)$ of them. Moreover, any two blocks intersect in at most one point.

For more on Steiner systems and their properties, see the following papers:

G.D. Cohen anf P. Frankl, On generalized perfect codes and Steiner systems, *Annals of Discrete Mathematics*, **18**, pp. 197–200, 1983.

G.D. Cohen and B. Montaron, Empilements parfaits de boules dans les espaces vectoriels binaires, *Compte Rendus de l'Academie des Sciences*, **288**, pp. 579–582, 1979.

B. Montaron and G.D. Cohen, Codes parfaits binaires a plusieurs rayons, *Revue du Centre d'Études Théoriques de la Détection et Communication*, **NS1979-2**, pp. 35–58, 1979.

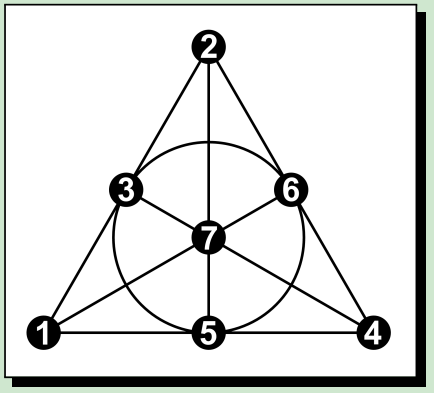
PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2,q,r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example: Fano plane $\mathcal{S}(2,3,7)$

	1	2	3	4	5	6	7
A_1:	●	●	●	–	–	–	–
A_2:	–	●	–	●	–	●	–
A_3:	●	–	–	●	●	–	–
A_4:	●	–	–	–	–	●	●
A_5:	–	●	–	–	●	–	●
A_6:	–	–	●	●	–	–	●
A_7:	–	–	●	–	●	●	–



Observation: There are $b = \binom{r}{2}/\binom{q}{2}$ blocks in $\mathcal{B}$ and each point is contained in $(r-1)/(q-1)$ of them. Moreover, any two blocks intersect in at most one point.

Conclusion: The blocks of a Steiner system $\mathcal{S}(2,q,s)$ form an almost disjoint $(s-1)/(q-1)$-cover of $[s]$. Therefore, when such Steiner systems exist, we have

$$\rho(s,k) \leqslant \text{number of blocks in } \mathcal{S}(2,q,s) = \frac{s(s-1)}{q(q-1)} = \frac{s(k-1)^2}{s+k}$$

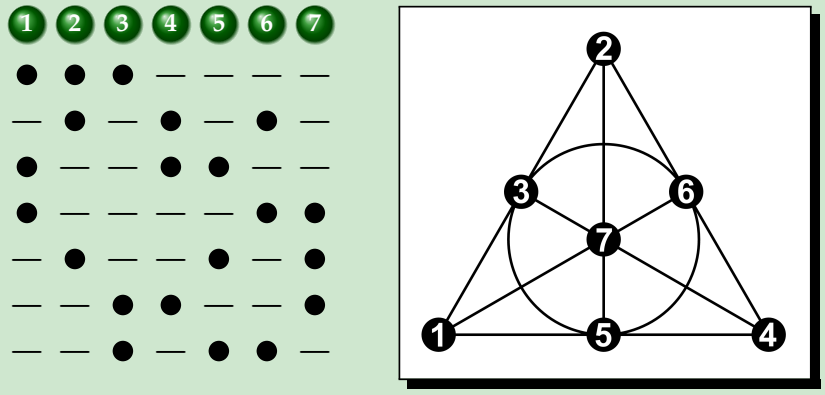
where $k = (s-1)/(q-1)+1$.

PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2,q,r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example: Fano plane $\mathcal{S}(2,3,7)$



Observation: There are $b = \binom{r}{2}/\binom{q}{2}$ blocks in $\mathcal{B}$ and each point is contained in $(r-1)/(q-1)$ of them. Moreover, any two blocks intersect in at most one point.

Conclusion: The blocks of a Steiner system $\mathcal{S}(2,q,s)$ form an almost disjoint $(s-1)/(q-1)$-cover of $[s]$. Therefore, when such Steiner systems exist, we have

$$\rho(s,k) \leqslant \text{number of blocks in } \mathcal{S}(2,q,s) = \frac{s(s-1)}{q(q-1)} = \frac{s(k-1)^2}{s+k}$$

where $k = (s-1)/(q-1)+1$. But, by Fisher's inequality (*# blocks* $\geqslant$ *# points*), this gives $\rho(s,k) \leqslant s$ at best.

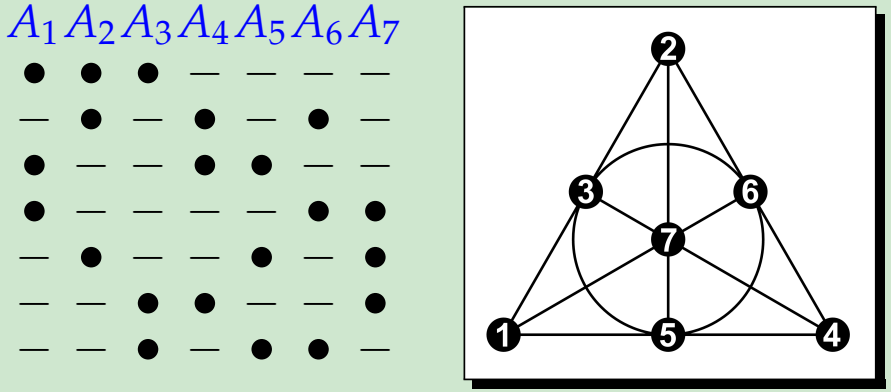
We can do much better with Steiner systems!

PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2,q,r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example: Fano plane $\mathcal{S}(2,3,7)$



Lemma 10: PIR codes from Steiner systems

Let $\mathcal{S}(2,q,r)$ be a Steiner system. For each $v \in V$, let $A_v \subset \mathcal{B}$ be the set of blocks that contain v. Then the sets $\{A_v : v \in V\}$ form an almost disjoint q-cover of $[b]$.

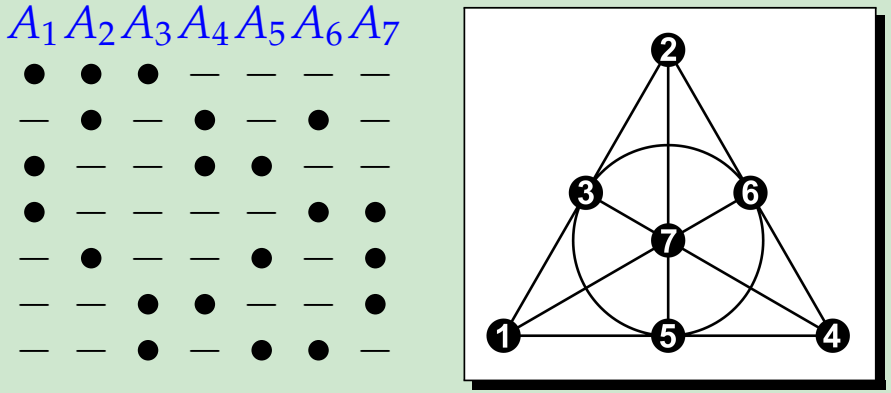
Proof. For any pair of points u and v, there is only one block that contains both. Hence $|A_v \cap A_u| = 1$, and the sets $\{A_v : v \in V\}$ are almost disjoint. □

PIR codes from Steiner systems

Let V be a set with r elements, called **points.** A ***Steiner system*** $\mathcal{S}(2,q,r)$ is a collection $\mathcal{B}$ of subsets of V of size q, called **blocks,** such that every pair of points is contained in exactly one block.

- Such a system is an example of a *balanced incomplete block design.*

Example: Fano plane $\mathcal{S}(2,3,7)$



Lemma 10: PIR codes from Steiner systems

Let $\mathcal{S}(2,q,r)$ be a Steiner system. For each $v \in V$, let $A_v \subset \mathcal{B}$ be the set of blocks that contain v. Then the sets $\{A_v : v \in V\}$ form an almost disjoint q-cover of $[b]$.

Proof. For any pair of points u and v, there is only one block that contains both. Hence $|A_v \cap A_u| = 1$, and the sets $\{A_v : v \in V\}$ are almost disjoint. □

By Wilson's theorem, a Steiner system $\mathcal{S}(2,q,r)$ exists for all sufficiently large r whenever $(q-1)|(r-1)$ and $q(q-1)|r(r-1)$. Combining this theorem with Lemma 10 and Corollary 9, we have:

$$\rho(s,k) = O\left(\sqrt{s}\right) \quad \textit{for all fixed } k$$

PIR codes from bipartite graphs

Let $\mathcal{G} = (U, V; \mathcal{E})$ be a bipartite graph, with bipartition U, V and edge set $\mathcal{E}$. We consider the neighborhoods $N(v) = \{u \in U : (u, v) \in \mathcal{E}\}$ of vertices in V.

Lemma 11: PIR codes from bipartite graphs

*If $\mathcal{G}$ has **no 4-cycles**, then the neighborhoods of vertices in V, namely the set $\{N(v) : v \in V\}$, form an almost disjoint k-cover of U, where $k = \min_{u \in U} \deg(u)$.*

Proof. Assume to the contrary that there are vertices $v_1, v_2 \in V$ such that $|N(v_1) \cap N(v_2)| \geqslant 2$. Let u_1, u_2 be some two vertices in $N(v_1) \cap N(v_2)$. Then the induced subgraph on $\{v_1, v_2, u_1, u_2\}$ is $K_{2,2}$ which is a 4-cycle in $\mathcal{G}$. □

Note: [DGRS15] use a similar construction for batch codes, but with girth$(\mathcal{G}) \geqslant 8$.

PIR codes from bipartite graphs

Let $\mathcal{G} = (U, V; \mathcal{E})$ be a bipartite graph, with bipartition U, V and edge set $\mathcal{E}$. We consider the neighborhoods $N(v) = \{u \in U : (u, v) \in \mathcal{E}\}$ of vertices in V.

Lemma 11: PIR codes from bipartite graphs

*If $\mathcal{G}$ has **no 4-cycles**, then the neighborhoods of vertices in V, namely the set $\{N(v) : v \in V\}$, form an almost disjoint k-cover of U, where $k = \min_{u \in U} \deg(u)$.*

Proof. Assume to the contrary that there are vertices $v_1, v_2 \in V$ such that $|N(v_1) \cap N(v_2)| \geqslant 2$. Let u_1, u_2 be some two vertices in $N(v_1) \cap N(v_2)$. Then the induced subgraph on $\{v_1, v_2, u_1, u_2\}$ is $K_{2,2}$ which is a 4-cycle in $\mathcal{G}$. □

Note: [DGRS15] use a similar construction for batch codes, but with girth$(\mathcal{G}) \geqslant 8$.

PIR codes from bipartite graphs

Let $\mathcal{G} = (U, V; \mathcal{E})$ be a bipartite graph, with bipartition U, V and edge set $\mathcal{E}$. We consider the neighborhoods $N(v) = \{u \in U : (u, v) \in \mathcal{E}\}$ of vertices in V.

Lemma 11: PIR codes from bipartite graphs

*If $\mathcal{G}$ has **no 4-cycles**, then the neighborhoods of vertices in V, namely the set $\{N(v) : v \in V\}$, form an almost disjoint k-cover of U, where $k = \min_{u \in U} \deg(u)$.*

Proof. Assume to the contrary that there are vertices $v_1, v_2 \in V$ such that $|N(v_1) \cap N(v_2)| \geqslant 2$. Let u_1, u_2 be some two vertices in $N(v_1) \cap N(v_2)$. Then the induced subgraph on $\{v_1, v_2, u_1, u_2\}$ is $K_{2,2}$ which is a 4-cycle in $\mathcal{G}$. □

- Given s and k, we would like to construct a bipartite graph $\mathcal{G} = (U, V; \mathcal{E})$ with the following properties:

$$|U| = s \qquad \min_{u \in U} \deg(u) = k - 1 \qquad \text{girth}(\mathcal{G}) \geqslant 6$$

If we can do this, then $\rho(s, k) \leqslant |V|$ by Corollary 10. What is the least possible number of vertices in V for such a graph?

Note: [DGRS15] use a similar construction for batch codes, but with $\text{girth}(\mathcal{G}) \geqslant 8$.

PIR codes from bipartite graphs

Let $\mathcal{G} = (U, V; \mathcal{E})$ be a bipartite graph, with bipartition U, V and edge set $\mathcal{E}$. We consider the neighborhoods $N(v) = \{u \in U : (u, v) \in \mathcal{E}\}$ of vertices in V.

Lemma 11: PIR codes from bipartite graphs

*If $\mathcal{G}$ has **no 4-cycles**, then the neighborhoods of vertices in V, namely the set $\{N(v) : v \in V\}$, form an almost disjoint k-cover of U, where $k = \min_{u \in U} \deg(u)$.*

Proof. Assume to the contrary that there are vertices $v_1, v_2 \in V$ such that $|N(v_1) \cap N(v_2)| \geqslant 2$. Let u_1, u_2 be some two vertices in $N(v_1) \cap N(v_2)$. Then the induced subgraph on $\{v_1, v_2, u_1, u_2\}$ is $K_{2,2}$ which is a 4-cycle in $\mathcal{G}$. □

- Given s and k, we would like to construct a bipartite graph $\mathcal{G} = (U, V; \mathcal{E})$ with the following properties:

$$|U| = s \qquad \min_{u \in U} \deg(u) = k - 1 \qquad \text{girth}(\mathcal{G}) \geqslant 6$$

If we can do this, then $\rho(s, k) \leqslant |V|$ by Corollary 10. What is the least possible number of vertices in V for such a graph? Using the best known results on bipartite cages, we get:

$$\rho(s, k) = O\left(\sqrt{s}\right) \quad \textit{for all fixed } k$$

Note: [DGRS15] use a similar construction for batch codes, but with $\text{girth}(\mathcal{G}) \geqslant 8$.

PIR codes from constant-weight codes

Definition: Constant-weight codes

Let $A_2(n, d, w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

PIR codes from constant-weight codes

Definition: Constant-weight codes

Let $A_2(n,d,w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

To learn more about constant-weight codes and their properties, consult the following papers:

Ch. Bachoc, V. Chandar, G.D. Cohen, P. Solé, and A. Tchamkerten, On bounded weight codes, *IEEE Trans. Information Theory*, **57**, pp. 6780–6787, October 2011.

G.D. Cohen, P. Solé, and A. Tchamkerten, Heavy weight codes, *Proceedings IEEE International Symp. Information Theory*, pp. 1120–1124, Austin, TX., June 2010.

PIR codes from constant-weight codes

Definition: Constant-weight codes

Let $A_2(n,d,w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

Observations:

- $d = 2w$ if and only if any two codewords have ***disjoint supports:***

PIR codes from constant-weight codes

Definition: Constant-weight codes

Let $A_2(n,d,w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

Observations:

- $d = 2w$ if and only if any two codewords have ***disjoint supports:***

- $d = 2w - 2$ iff any two codewords ***intersect in at most one position:***

PIR codes from constant-weight codes

Definition: Constant-weight codes

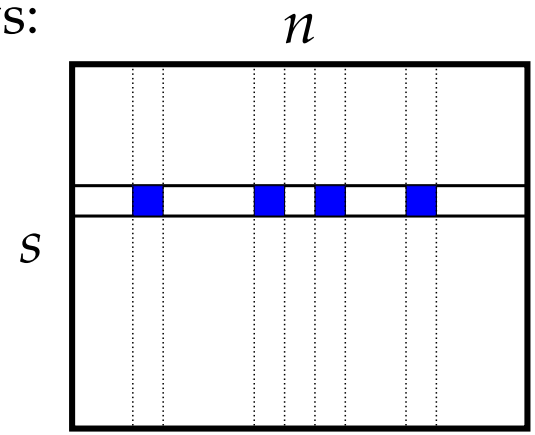
Let $A_2(n,d,w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

Observations:

- $d = 2w$ if and only if any two codewords have ***disjoint supports:***

- $d = 2w - 2$ iff any two codewords ***intersect in at most one position:***

Now let $s = A_2(n, 2w-2, w)$, and consider the $s \times n$ matrix having the codewords of $\mathbb{C}$ as its rows:



As the weight of each row is w, columns form a w-cover of $[s]$.

PIR codes from constant-weight codes

Definition: Constant-weight codes

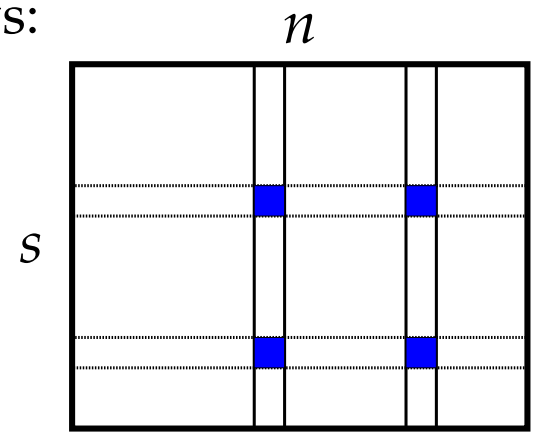
Let $A_2(n,d,w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

Observations:

- $d = 2w$ if and only if any two codewords have ***disjoint supports:***

- $d = 2w - 2$ iff any two codewords ***intersect in at most one position:***

Now let $s = A_2(n, 2w-2, w)$, and consider the $s \times n$ matrix having the codewords of $\mathbb{C}$ as its rows:



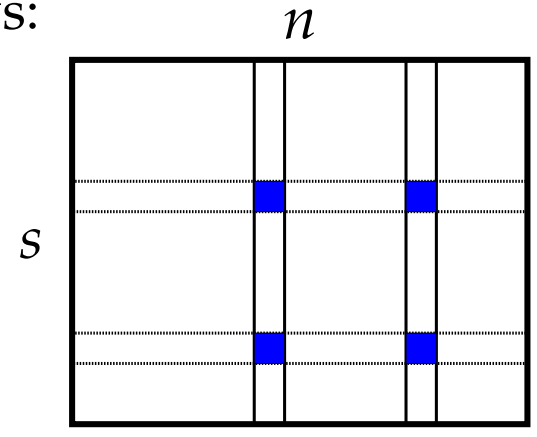
As the row supports are almost disjoint, the column supports are also almost disjoint.

PIR codes from constant-weight codes

Definition: Constant-weight codes

Let $A_2(n,d,w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

Now let $s = A_2(n, 2w-2, w)$, and consider the $s \times n$ matrix having the codewords of $\mathbb{C}$ as its rows:



As the row supports are almost disjoint, the column supports are also almost disjoint.

Theorem 12: PIR codes from constant-weight codes

$$\rho(s,k) \leqslant \text{ the smallest } n \text{ such that } A_2(n, 2k-4, k-1) \geqslant s$$

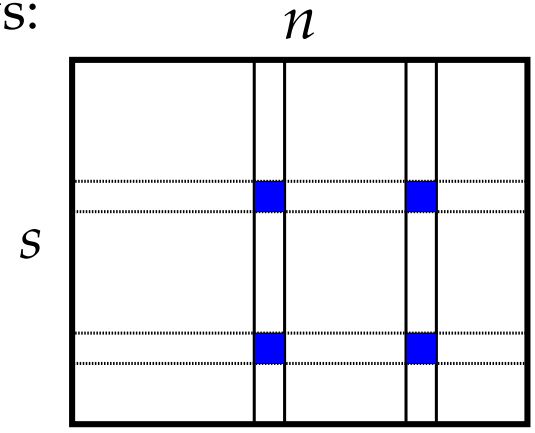
supports are almost disjoint ***$(k-1)$-cover of $[s]$***

PIR codes from constant-weight codes

Definition: Constant-weight codes

Let $A_2(n,d,w)$ be the number of codewords in the largest binary code $\mathbb{C}$ of length n and minimum distance d such that all the codewords of $\mathbb{C}$ have weight w.

Now let $s = A_2(n, 2w-2, w)$, and consider the $s \times n$ matrix having the codewords of $\mathbb{C}$ as its rows:



As the row supports are almost disjoint, the column supports are also almost disjoint.

Theorem 12: PIR codes from constant-weight codes

$$\rho(s,k) \leqslant \text{ the smallest } n \text{ such that } A_2(n, 2k-4, k-1) \geqslant s$$

supports are almost disjoint ***$(k-1)$-cover of*** $[s]$

For example, for $k = 3$ we conclude that $\rho(s,3)$ is upper bounded by the smallest n such that $n(n-1) \geqslant 2s$. In general, we again have $\rho(s,k) = O\left(\sqrt{s}\,\right)$.

Tables of short PIR codes

number k of servers emulated

number s of database parts	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1	3	4	6	7	9	10	12	13	15	16	18	19	21	22
3	1	3	4	7	8	10	11	14	15	17	18	21	22	24	25
4	1	4	5	7	8	10	11	15	16	19	20	22	23	25	26
5	1	4	5	7	8	13	14	17	18	20	21	23	24	25	26
6	1	4	5	7	8	14	15	18	19	21	22	28	29	32	33
7	1	5	6	7	8	15	16	20	21	22	23	30	31	35	36
8	1	5	6	11	12	15	16	24	25	29	30	35	36	39	40
9	1	5	6	12	13	15	16	25	26	30	31	37	38	40	41
10	1	5	6	13	14	15	16	26	27	31	32	39	40	41	42
11	1	6	7	13	14	21	22	30	31	37	38	39	40	41	42
12	1	6	7	13	14	21	22	30	31	37	38	39	40	41	42
13	1	6	7	13	14	21	22	30	31	37	38	39	40	41	42
14	1	6	7	14	15	21	22	30	31	37	38	39	40	41	42
15	1	6	7	15	16	21	22	30	31	37	38	39	40	41	42
16	1	7	8	16	17	21	22	30	31	45	46	51	52	55	56
17	1	7	8	16	17	21	22	30	31	46	47	55	56	60	61
18	1	7	8	16	17	21	22	30	31	47	48	56	57	61	62
19	1	7	8	16	17	21	22	30	31	48	49	57	58	62	63
20	1	7	8	16	17	21	22	30	31	49	50	58	59	63	64
21	1	7	8	18	19	27	28	30	31	52	53	59	60	70	71
22	1	8	9	18	19	28	29	30	31	53	54	61	62	71	72
23	1	8	9	19	20	28	29	30	31	54	55	62	63	73	74
24	1	8	9	19	20	28	29	30	31	55	56	63	64	74	75

Redundancy $\rho(s,k)$ of the best-known PIR codes

Tables of short PIR codes

number k of servers emulated

number s of database parts	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	1	3	4	6	7	9	10	12	13	15	16	18	19	21	22
3	1	3	4	7	8	10	11	14	15	17	18	21	22	24	25
4	1	4	5	7	8	10	11	15	16	19	20	22	23	25	26
5	1	4	5	7	8	13	14	17	18	20	21	23	24	25	26
6	1	4	5	7	8	14	15	18	19	21	22	28	29	32	33
7	1	5	6	7	8	15	16	20	21	22	23	30	31	35	36
8	1	5	6	11	12	15	16	24	25	29	30	35	36	39	
9	1	5	6	12	13	15	16	25	26	30	31	37	38		
10	1	5	6	13	14	15	16	26	27	31	32	39			
11	1	6	7	13	14	21	22	30	31	37	38				
12	1	6	7	13	14	21	22	30	31	37					42
13	1	6	7	13	14	21	22	30	31					41	42
14	1	6	7	14	15	21	22	30					40	41	42
15	1	6	7	15	16	21	22					39	40	41	42
16	1	7	8	16	17	21	22				46	51	52	55	56
17	1	7	8	16	17	21					46	47	55	56	60	61
18	1	7	8	16	17					47	48	56	57	61	62
19	1	7	8	16					31	48	49	57	58	62	63
20	1	7	8	16				30	31	49	50	58	59	63	64
21	1	7	8	18			28	30	31	52	53	59	60	70	71
22	1	8	9	18		28	29	30	31	53	54	61	62	71	72
23	1	8	9	19	20	28	29	30	31	54	55	62	63	73	74
24	1	8	9	19	20	28	29	30	31	55	56	63	64	74	75

Improve any entry in this table!

Redundancy $\rho(s,k)$ of the best-known PIR codes

Tables of short PIR codes

number k of servers emulated

number s of database parts	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00	11.00	12.00	13.00	14.00	15.00	16.00
2	1.50	2.50	3.00	4.00	4.50	5.50	6.00	7.00	7.50	8.50	9.00	10.00	10.50	11.50	12.00
3	1.33	2.00	2.33	3.33	3.67	4.33	4.67	5.67	6.00	6.67	7.00	8.00	8.33	9.00	9.33
4	1.25	2.00	2.25	2.75	3.00	3.50	3.75	4.75	5.00	5.75	6.00	6.50	6.75	7.25	7.50
5	1.20	1.80	2.00	2.40	2.60	3.60	3.80	4.40	4.60	5.00	5.20	5.60	5.80	6.00	6.20
6	1.17	1.67	1.83	2.17	2.33	3.33	3.50	4.00	4.17	4.50	4.67	5.67	5.83	6.33	6.50
7	1.14	1.71	1.86	2.00	2.14	3.14	3.29	3.86	4.00	4.14	4.29	5.29	5.43	6.00	6.14
8	1.13	1.63	1.75	2.38	2.50	2.88	3.00	4.00	4.13	4.63	4.75	5.38	5.50	5.88	6.00
9	1.11	1.56	1.67	2.33	2.44	2.67	2.78	3.78	3.89	4.33	4.44	5.11	5.22	5.44	5.56
10	1.10	1.50	1.60	2.30	2.40	2.50	2.60	3.60	3.70	4.10	4.20	4.90	5.00	5.10	5.20
11	1.09	1.55	1.64	2.18	2.27	2.91	3.00	3.73	3.82	4.36	4.45	4.55	4.64	4.73	4.82
12	1.08	1.50	1.58	2.08	2.17	2.75	2.83	3.50	3.58	4.08	4.17	4.25	4.33	4.42	4.50
13	1.08	1.46	1.54	2.00	2.08	2.62	2.69	3.31	3.38	3.85	3.92	4.00	4.08	4.15	4.23
14	1.07	1.43	1.50	2.00	2.07	2.50	2.57	3.14	3.21	3.64	3.71	3.79	3.86	3.93	4.00
15	1.07	1.40	1.47	2.00	2.07	2.40	2.47	3.00	3.07	3.47	3.53	3.60	3.67	3.73	3.80
16	1.06	1.44	1.50	2.00	2.06	2.31	2.38	2.88	2.94	3.81	3.88	4.19	4.25	4.44	4.50
17	1.06	1.41	1.47	1.94	2.00	2.24	2.29	2.76	2.82	3.71	3.76	4.24	4.29	4.53	4.59
18	1.06	1.39	1.44	1.89	1.94	2.17	2.22	2.67	2.72	3.61	3.67	4.11	4.17	4.39	4.44
19	1.05	1.37	1.42	1.84	1.89	2.11	2.16	2.58	2.63	3.53	3.58	4.00	4.05	4.26	4.32
20	1.05	1.35	1.40	1.80	1.85	2.05	2.10	2.50	2.55	3.45	3.50	3.90	3.95	4.15	4.20
21	1.05	1.33	1.38	1.86	1.90	2.29	2.33	2.43	2.48	3.48	3.52	3.81	3.86	4.33	4.38
22	1.05	1.36	1.41	1.82	1.86	2.27	2.32	2.36	2.41	3.41	3.45	3.77	3.82	4.23	4.27
23	1.04	1.35	1.39	1.83	1.87	2.22	2.26	2.30	2.35	3.35	3.39	3.70	3.74	4.17	4.22
24	1.04	1.33	1.38	1.79	1.83	2.17	2.21	2.25	2.29	3.29	3.33	3.63	3.67	4.08	4.13

Storage overhead of the best-known PIR codes

Tables of short PIR codes

number k of servers emulated

number s of database parts	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00	11.00	12.00	13.00	14.00	15.00	16.00
2	1.50	2.50	3.00	4.00	4.50	5.50	6.00	7.00	7.50	8.50	9.00	10.00	10.50	11.50	12.00
3	1.33	2.00	2.33	3.33	3.67	4.33	4.67	5.67	6.00	6.67	7.00	8.00	8.33	9.00	9.33
4	1.25	2.00	2.25	2.75	3.00	3.50	3.75	4.75	5.00	5.75	6.00	6.50	6.75	7.25	7.50
5	1.20	1.80	2.00	2.40	2.60	3.60	3.80	4.40	4.60	5.00	5.20	5.60	5.80	6.00	6.20
6	1.17	1.67	1.83	2.17	2.33	3.33	3.50	4.00	4.17	4.50	4.67	5.67	5.83	6.33	6.50
7	1.14	1.71	1.86	2.00	2.14	3.14	3.29	3.86	4.00	4.14	4.29	5.29	5.43	6.00	6.14
8	1.13	1.63	1.75	2.38	2.50	2.88	3.00	4.00	4.13	4.63	4.75	5.38	5.50	5.88	6.00
9	1.11	1.56	1.67	2.33	2.44	2.67	2.78	3.78	3.89	4.33	4.44	5.11	5.22	5.44	5.56
10	1.10	1.50	1.60	2.30	2.40	2.50	2.60	3.60	3.70	4.10	4.20	4.90	5.00	5.10	5.20
11	1.09	1.55	1.64	2.18	2.27	2.91	3.00	3.73	3.82	4.36	4.45	4.55	4.64	4.73	4.82
12	1.08	1.50	1.58	2.08	2.17	2.75	2.83	3.50	3.58	4.08	4.17	4.25	4.33	4.42	4.50
13	1.08	1.46	1.54	2.00	2.08	2.62	2.69	3.31	3.38	3.85	3.92	4.00	4.08	4.15	4.23
14	1.07	1.43	1.50	2.00	2.07	2.50	2.57	3.14	3.21	3.64	3.71	3.79	3.86	3.93	4.00
15	1.07	1.40	1.47	2.00	2.07	2.40	2.47	3.00	3.07	3.47	3.53	3.60	3.67	3.73	3.80
16	1.06	1.44	1.50	2.00	2.06	2.31	2.38	2.88	2.94	3.81	3.88	4.19	4.25	4.44	4.50
17	1.06	1.41	1.47	1.94	2.00	2.24	2.29	2.76	2.82	3.71	3.76	4.24	4.29	4.53	4.59
18	1.06	1.39	1.44	1.89	1.94	2.17	2.22	2.67	2.72	3.61	3.67	4.11	4.17	4.39	4.44
19	1.05	1.37	1.42	1.84	1.89	2.11	2.16	2.58	2.63	3.53	3.58	4.00	4.05	4.26	4.32
20	1.05	1.35	1.40	1.80	1.85	2.05	2.10	2.50	2.55	3.45	3.50	3.90	3.95	4.15	4.20
21	1.05	1.33	1.38	1.86	1.90	2.29	2.33	2.43	2.48	3.48	3.52	3.81	3.86	4.33	4.38
22	1.05	1.36	1.41	1.82	1.86	2.27	2.32	2.36	2.41	3.41	3.45	3.77	3.82	4.23	4.27
23	1.04	1.35	1.39	1.83	1.87	2.22	2.26	2.30	2.35	3.35	3.39	3.70	3.74	4.17	4.22
24	1.04	1.33	1.38	1.79	1.83	2.17	2.21	2.25	2.29	3.29	3.33	3.63	3.67	4.08	4.13

Storage overhead of the best-known PIR codes

Thank you for your attention!

Please send you queries...