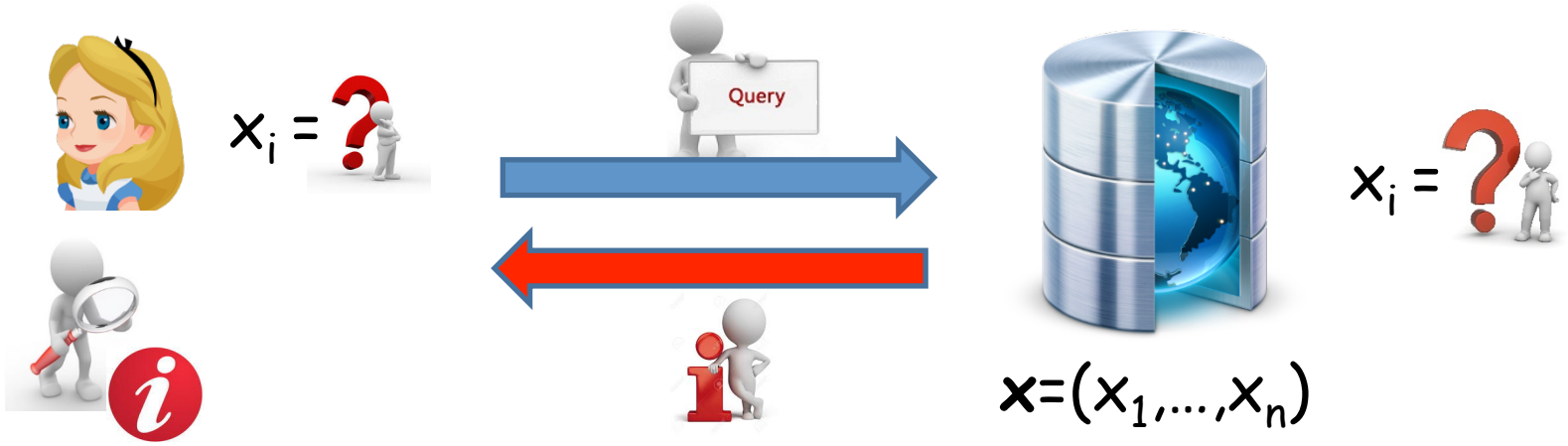
PIR with Low Storage Overhead: Coding instead of Replication

Arman Fazeli[1], Alexander Vardy[1], Eitan Yaakobi[2]

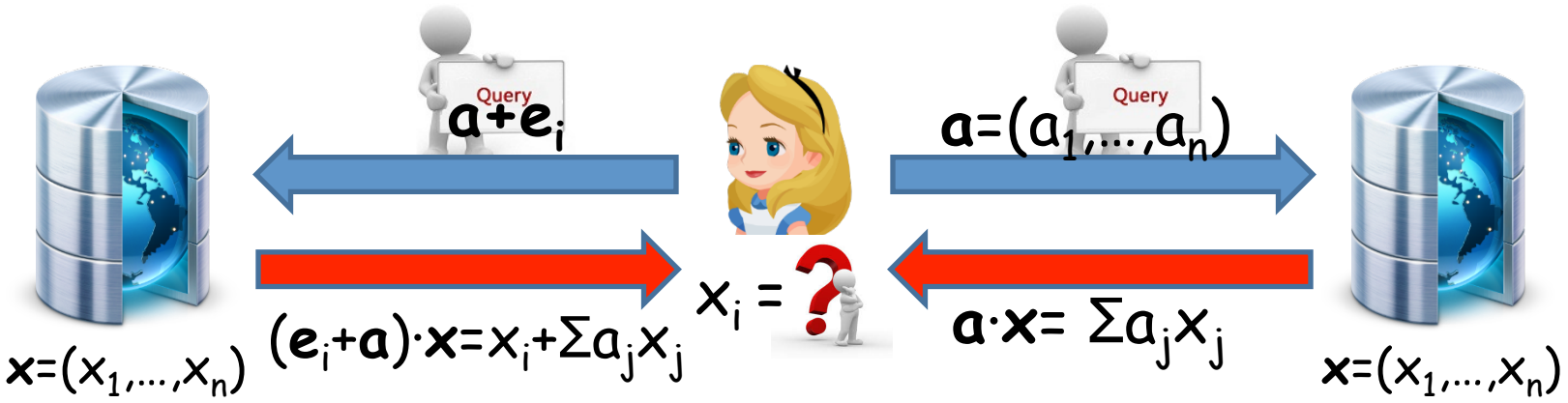
(1) University of California San Diego

(2) Technion Israel Institute of Technology

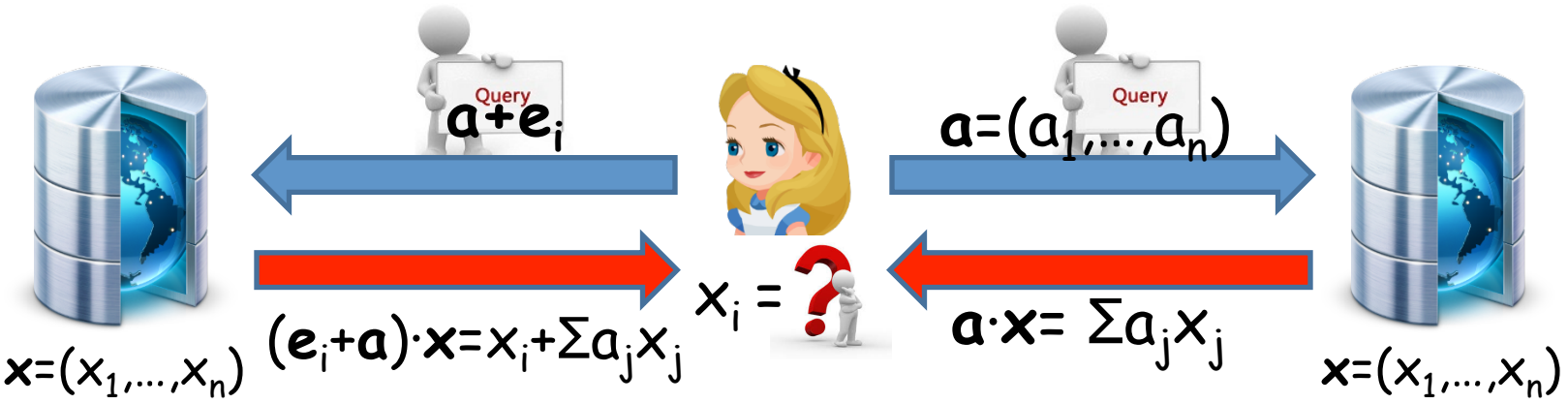
What's Private Information Retrieval (PIR)?



- Alice wants to read the i^{th} bit from a database $x=(x_1 \ldots x_n)$
- But the database can deduce **nothing** about the value of i…
- **The problem**: minimize the communication complexity
- Naïve solution: read the entire database…
  - In fact, cannot do better if there is only a single database
- Two models:
  - Information theoretic PIR
  - Computational PIR



- **Example**: Two-server PIR
  - Alice chooses uniformly at random a binary vector $\mathbf{a}=(a_1,\ldots,a_n)$
  - First server receives $\mathbf{a}$
  - Second server receives $\mathbf{a}+\mathbf{e}_i$
  - First server returns $\mathbf{a}\cdot\mathbf{x} = \Sigma a_j x_j$
  - Second server returns $(\mathbf{e}_i+\mathbf{a})\cdot\mathbf{x} = x_i + \Sigma a_j x_j$
  - Alice calculates $\mathbf{a}\cdot\mathbf{x} + (\mathbf{e}_i+\mathbf{a})\cdot\mathbf{x} = x_i$
  - Correctness, Privacy ✔
  - Communication complexity:
    - Download – 2 bits ☺
    - Upload – 2n bits ☹



Definition: a k-server PIR scheme consists of

- k servers $S_1,\ldots,S_k$ each stores the database $\mathbf{x}$
- Alice wants to retrieve x_i, without revealing i
- A protocol P with three algorithms P(Q,A,C)
  - Alice randomly generates k queries $Q(k,n;i)=(q_1,\ldots,q_k)$ and sends to the servers
  - Each server responds with $a_j=A(k,j,x,q_j)$
  - Alice computes x_i by $C(k,n;i,a_1,\ldots,a_k)$
- **Requirements**:
  - **Privacy**: each server learns no information about i
  - **Correctness**: $C(k,n;i,a_1,\ldots,a_k) = x_i$
- **Communication complexity**: number of uploaded and downloaded bits
- **Storage overhead**: ratio between stored and information bits
- A protocol P(Q,A,C) is called a **linear PIR protocol** if

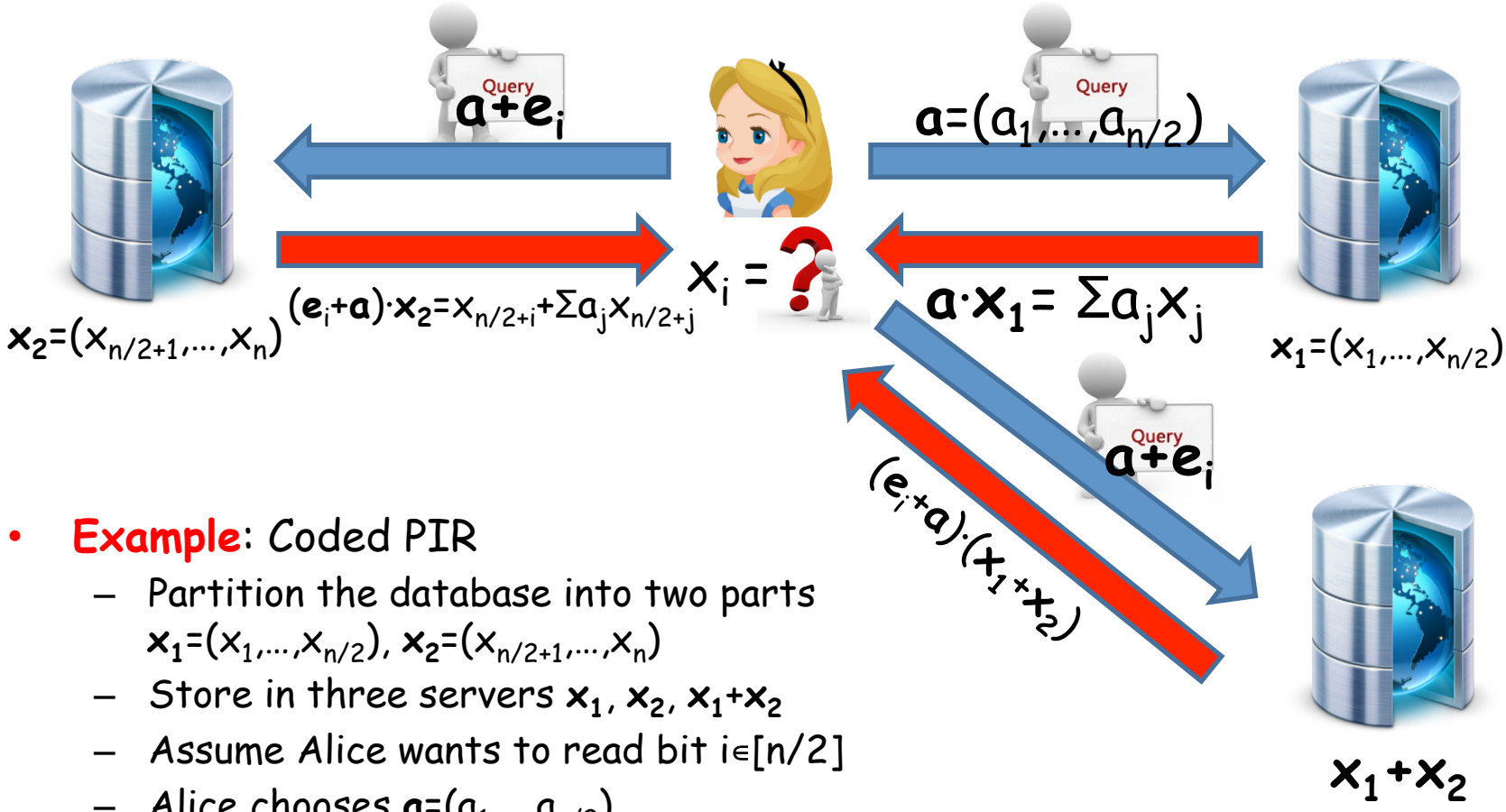
$$A(k,j,\mathbf{x}_1,q_j)+ A(k,j,\mathbf{x}_2,q_j) = A(k,j,\mathbf{x}_1+\mathbf{x}_2,q_j)$$

Previous Work

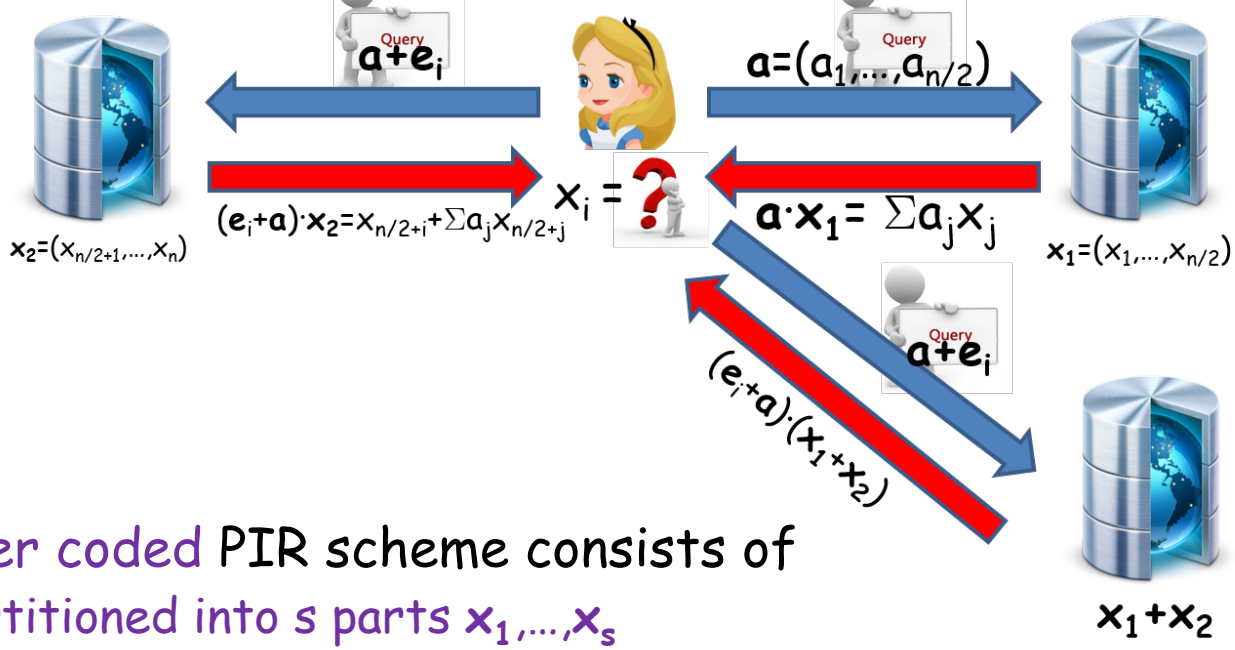
- **Chor, Kushilevitz, Goldreich, Sudan**, *Private information retrieval*, FOCS, '95
- **Ambainis**, *Upper bound on communication complexity of private information retrieval*, ICALP '97
- **Kushilevitz, Ostrovsky**, *Replication is not needed: single database, computationally-private information retrieval*, FOCS '97
- **Beimel, Ishai, Malkin**, Reducing the servers computation in private information retrieval: PIR with preprocessing, CRYPTO '00
- **Beimel, Ishai, Kushilevitz, Raymond**, *Breaking the $O(n^{1/(2k-1)})$ barrier for information theoretic private information retrieval*, FOCS '02
- **Beimel, Ishai, Kushilevitz**, *General constructions for information-theoretic private information retrieval*, Journal of Computer and System Sciences, '05
- **Woodruff, Yekhanin**, A geometric approach to information-theoretic private information retrieval, CCC '05
- **Yekhanin**, *Towards 3-query locally decodable codes of subexponential length*, Joural ACM, '08
- **Efremenko**, *3-query locally decodable codes of subexponential length*, STOC '09
- **Dvir, Gopi**, *2-server PIR with sub-polynomial communication*, '14

Related Work

- **Shah, Rashmi, Ramchandran**, *One extra bit of download ensures perfectly private information retrieval*, ISIT '14
- **Chan, Ho, Yamamoto**, *Private information retrieval for coded storage*, '14
- **Augot, Levy-Dit-Vehel, Shikfa**, A storage-efficient and robust private information retrieval scheme allowing few servers, '14
- **Ishai, Kushilevitz, Ostrovsky, Sahai**, *Batch codes and their applications*, STOC '04
- **Dimakis, Gal, Rawat, Song**, *Batch Codes through dense graphs without short cycles*, '14



- **Example**: Coded PIR
  - Partition the database into two parts $x_1=(x_1,\dots,x_{n/2})$, $x_2=(x_{n/2+1},\dots,x_n)$
  - Store in three servers x_1, x_2, x_1+x_2
  - Assume Alice wants to read bit $i\in[n/2]$
  - Alice chooses $a=(a_1,\dots,a_{n/2})$
  - First server receives a and returns $a\cdot x_1=\Sigma a_j x_j$
  - Second server receives $a+e_i$ and returns $(e_i+a)\cdot x_2= x_{n/2+i}+\Sigma a_j x_{n/2+j}$
  - Third server receives $a+e_i$ and returns $(e_i+a)\cdot(x_1+x_2)= x_i+x_{n/2+i}+\Sigma a_j x_j+\Sigma a_j x_{n/2+j}$
  - Alice calculates $a\cdot x_1+(e_i+a)\cdot x_2+(e_i+a)\cdot(x_1+x_2)= x_i$
  - Correctness and Privacy ✔
  - **Communication complexity**: Download – 3 bits ☺, Upload – 1.5n bits ☹
  - **Storage overhead**: 1.5



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

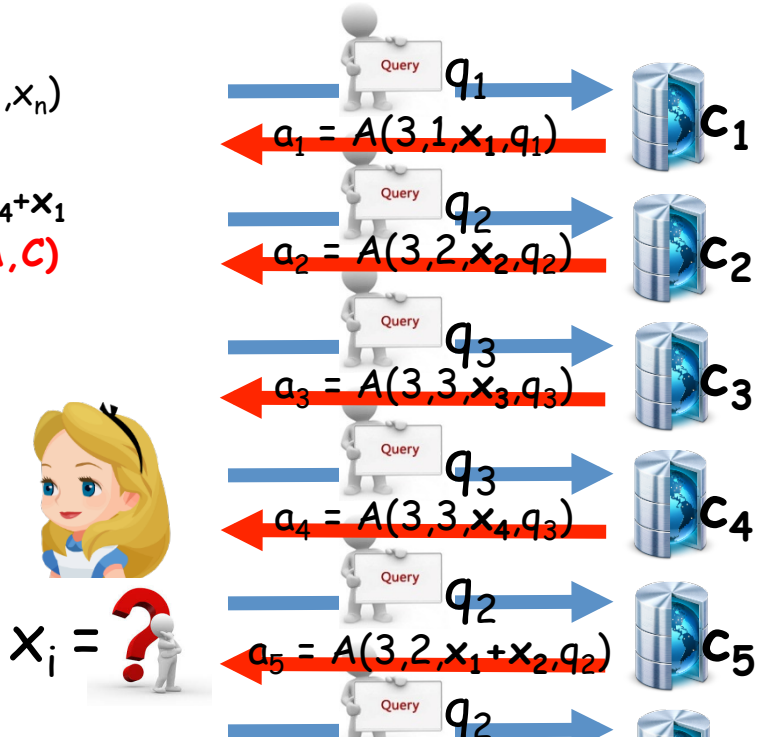
- Database **x**, which is partitioned into s parts $x_1,\ldots,x_s$
- m servers $S_1,\ldots,S_m$, each stores a function of $x_1,\ldots,x_s$
- Alice wants to retrieve x_i, without revealing i
- A protocol P^* with three algorithms $P^*(Q^*,A^*,C^*)$
  - Alice randomly generates m queries $Q^*(m,s,n;i)=(q_1,\ldots,q_m)$ and sends to the servers
  - Each server responds with $a_j=A^*(m,s,j,x,q_j)$
  - Alice computes x_i by $C^*(m,s,n;i,a_1,\ldots,a_m)$
- **Requirements**:
  - **Privacy**: each server learns no information about i
  - **Correctness**: $C(m,s,n;i,a_1,\ldots,a_m) = x_i$
- **Communication complexity**: number of uploaded and downloaded bits
- **Storage overhead**: ratio between stored and information bits

- The database is partitioned into four parts $x_1=(x_1,\dots,x_{n/4}), x_2=(x_{n/4+1},\dots,x_{n/2}), x_3=(x_{n/2+1},\dots,x_{3n/4}), x_4=(x_{3n/4+1},\dots,x_n)$
- Store in eight servers $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$:

  $c_1=x_1,\ c_2=x_2,\ c_3=x_3,\ c_4=x_4,\ c_5=x_1+x_2,\ c_6=x_2+x_3,\ c_7=x_3+x_4,\ c_8=x_4+x_1$
- Assume there exists a **3-server linear PIR protocol P(Q,A,C)**
- Assume Alice wants to read bit $i\in[n/4]$
  - Alice invokes $Q(3,n/4;i) = (q_1, q_2, q_3)$ and assigns the queries

    $Q^*(8,4,n;i) = (q_1, q_2, q_3, q_3, q_2, q_2, q_3, q_3)$
- Retrieval:
  - $a'_1 = a_1 = A(3,1,x_1,q_1)$
  - $a'_2 = a_2 + a_5 = A(3,2,x_2,q_2) + A(3,2,x_1+x_2,q_2) = A(3,2,x_1,q_2)$
  - $a'_3 = a_4 + a_8 = A(3,3,x_4,q_3) + A(3,3,x_4+x_1,q_3) = A(3,3,x_1,q_3)$
- $C^*(8,4,n;i,a_1,\dots,a_8) = C(3,n/4;i,a'_1,a'_2,a'_3) =$

  $C(3,n/4;i,\ A(3,1,x_1,q_1),\ A(3,2,x_1,q_2),\ A(3,3,x_1,q_3)) = x_i$
- **Correctness and Privacy**: from P
- **Communication complexity**: same as the on
- **Storage overhead**: 2 (instead of 3)



Server	Query	Response
1	q_2	$a_1 = \mathcal{A}^*(8,4,1,c_1,q_2) = \mathcal{A}(3,2,x_1,q_2)$
2	q_1	$a_2 = \mathcal{A}^*(8,4,2,c_2,q_1) = \mathcal{A}(3,1,x_2,q_1)$
3	q_3	$a_3 = \mathcal{A}^*(8,4,3,c_3,q_3) = \mathcal{A}(3,3,x_3,q_3)$
5	q_2	$a_5 = \mathcal{A}^*(8,4,5,c_5,q_2) = \mathcal{A}(3,2,c_5 = x_1+x_2,q_2)$
6	q_3	$a_6 = \mathcal{A}^*(8,4,6,c_6,q_3) = \mathcal{A}(3,3,c_6 = x_2+x_3,q_3)$

$a_8 = A(3,3,x_4+x_1,q_3)$ c_8

Server	Query	Response
1	q_1	$a_1 = \mathcal{A}^*(8,4,1,c_1,q_1) = \mathcal{A}(3,1,x_1,q_1)$
2	q_2	$a_2 = \mathcal{A}^*(8,4,2,c_2,q_2) = \mathcal{A}(3,2,x_2,q_2)$
4	q_3	$a_4 = \mathcal{A}^*(8,4,4,c_4,q_3) = \mathcal{A}(3,3,x_4,q_3)$
5	q_2	$a_5 = \mathcal{A}^*(8,4,5,c_5,q_2) = \mathcal{A}(3,2,c_5 = x_1+x_2,q_2)$
8	q_3	$a_8 = \mathcal{A}^*(8,4,8,c_8,q_3) = \mathcal{A}(3,3,c_8 = x_4+x_1,q_3)$

Definition: An (m,s)-server coded PIR scheme consists of
- Database x, which is partitioned into s parts $x_1,\dots,x_s$
- m servers $S_1,\dots,S_m$, each stores a function of $x_1,\dots,x_s$
- Alice wants to retrieve x_i, without revealing i
- A protocol P^* with three algorithms $P^*(Q^*,A^*,C^*)$
  - Alice randomly generates m queries $Q^*(m,s,n;i)=(q_1,\dots,q_m)$ and sends t
  - Each server responds with $a_j=A^*(m,s,j,x,q_j)$
  - Alice computes x_i by $C^*(m,s,n;i,a_1,\dots,a_m)$
- **Requirements**:
  - **Privacy**: each server learns no information about i
  - **Correctness**: $C(m,s,n;i,a_1,\dots,a_m) = x_i$
- **Communication complexity**: number of uploaded and downloaded
- **Storage overhead**: ratio between stored and information bits

How to Construct Coded PIR Protocols?

- **Two ingredients**:
  - A k-server linear PIR protocol
  - An [m,s] linear code with special properties
- **Definition**: A binary [m,s] linear code is a **k-server PIR code** if for every information bit u_i, $i \in [s]$, there exist k mutually disjoint sets $R_{i,1},\ldots, R_{i,k}$ such that u_i is a linear function of the bits in every set
- **Example**:
  - $c_1=x_1$, $c_2=x_2$, $c_3=x_3$, $c_4=x_4$, $c_5=x_1+x_2$, $c_6=x_2+x_3$, $c_7=x_3+x_4$, $c_8=x_4+x_1$
  - [8,4] 3-server PIR code

$$(c_1,\ldots,c_8) = (x_1,x_2,x_3,x_4)\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

How to Construct Coded PIR Protocols?

- **Theorem**: If there exist
  - k-server linear PIR protocol P
  - [m,s] k-server PIR code C

  then there exists an (m,s)-server coded PIR protocol P^*
- **Communication complexity**:
  - $U^*(P^*;n,m,s) = m \cdot U(P;n/s,k)$
  - $D^*(P^*;n,m,s) = m \cdot D(P;n/s,k)$
- **Storage overhead**: $m \cdot (n/s)/n = m/s$

- **Definition**: A binary [m,s] linear code is a **k-server PIR code** if for every information bit u_i, $i \in [s]$, there exist k mutually disjoint sets $R_{i,1},\ldots, R_{i,k}$ such that u_i is a linear function of the bits in every set

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

How to construct PIR codes?

- **Goal**: given s (number of chunks) and k (PIR protocol), find the smallest m such that there exists an [m,s] k-server PIR code
- Denote $A(s,k) = m$
- **Example**: for s=4 chunks and k=3, m=8, $A(4,3)=8$
- **Example**: $A(s,2)=s+1$
  - A simple parity code $(x_1,\ldots x_s) \Rightarrow (x_1,\ldots,x_s,x_{s+1})$
  - **Storage overhead**: $(s+1)/s = 1+1/s \rightarrow 1$ ☺

- **Definition**: A binary [m,s] linear code is a **k-server PIR code** if for every information bit u_i, $i\in[s]$, there exist k mutually disjoint sets $R_{i,1},\ldots, R_{i,k}$ such that u_i is a linear function of the bits in every set

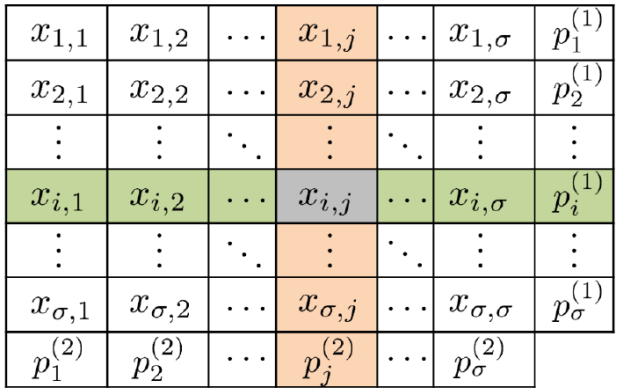
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

How to construct PIR codes?

- **Goal**: given s (number of chunks) and k (PIR protocol), find the smallest m such that there exists an [m,s] k-server PIR code
- Denote $A(s,k) = m$
- **Example**: for 4 chunks and k=3, m=8, $A(4,3)=8$
- Strong connections with:
  - Codes with locality and availability
  - One-step majority logic codes
  - Constant-weight codes
  - Combinatorial designs such as Steiner systems, difference sets and more

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

How to construct PIR codes?



- Construction for k=3
  - $s = v^2$
  - A product code with a simple parity
  - We get a $[(v+1)^2-1, v^2]$ 3-server PIR code
  - $A(v^2,3) \le (v+1)^2-1 = v^2+2v$
  - **Storage overhead**: $(v^2+2v)/v^2 = 1+2/v \rightarrow 1$ ☺
  - For any s, k: $A(s,k) \le s+(k-1)s^{(k-2)/(k-1)}$

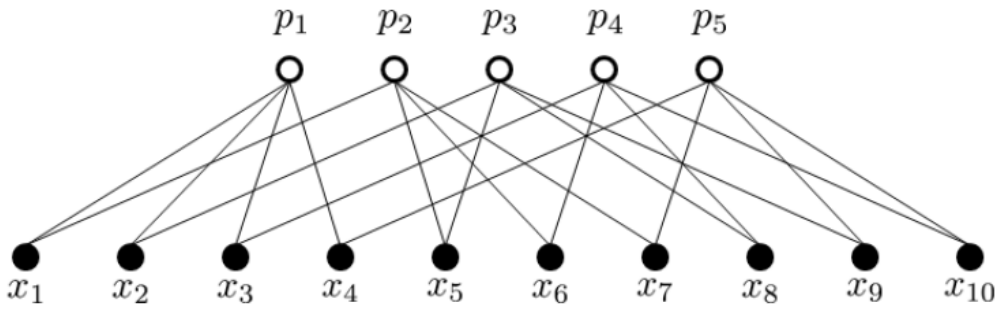
    Storage overhead: $(s+(k-1)s^{(k-2)/(k-1)})/s = 1+(k-1)s^{-1/(k-1)} \rightarrow 1$ ☺
  - **Conclusion**: for any fixed k, $A(s,k)/s \rightarrow 1$

- **Definition**: A binary [m,s] linear code is a **k-server PIR code** if for every information bit u_i, $i \in [s]$, there exist k mutually disjoint sets $R_{i,1},\ldots, R_{i,k}$ such that u_i is a linear function of the bits in every set

Construction based upon Constant-Weight Codes

- Consider the systematic generator matrix of the code
- The rows of the redundancy bits are codewords in a constant-weight code with weight 2 and min dist 2
  - in general weight k-1 and minimum distance 2(k-2)
- For k=3, all words of weight 2
  $A(v(v-1)/2,3) \leq v(v-1)/2+v$
- **Conclusion**: for fixed k, $A(s,k) \approx s + s^{\frac{1}{2}}$

$$\left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right)$$



$$\left[\begin{array}{cccccccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

One-Step Majority Logic Codes

- A method for fast decoding by only XOR and majority
  - **Costello, Lin**, *Error control coding (2nd edition)*, Pearson Higher Education, 2004
- **Example**: the [15,7] double-ECC cyclic code
  - Every bit has 5 mutually disjoint recovering sets
  - Can correct 2 errors by simple majority on four equations
  - A(7,5) = 15
- Several constructions of one-step majority logic codes
- **Conclusion**: for fixed k, $A(s,k) \approx s + s^{\frac{1}{2}}$

$$
\begin{aligned}
h_3 &= (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1) \\
h_{1+5} &= (0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1) \\
h_{0+2+6} &= (1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1) \\
h_7 &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1)
\end{aligned}
$$

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

Summary: Constructions with Fixed k

Code construction	Upper bound on $A(s,k)$	Asymptotic redundancy
Cubic construction	$A(s,k) \leqslant s + (k-1)\lceil s^{\frac{1}{k-1}} \rceil^{k-2}$	$O(s^{1-\frac{1}{k-1}})$
Steiner System	$A\left(\frac{n(n-1)}{(k-1)(k-2)}, k\right) \leqslant n + \frac{n(n-1)}{(k-1)(k-2)}$	$O(s^{\frac{1}{2}})$
Type-1 DTI codes (1)	$A\left(2^{2\theta\ell} - (2^{\theta+1}-1)^{\ell}, 2^{\ell}+2\right) \leqslant 2^{2\theta\ell} - 1$	$O(s^{\frac{1}{2}})$
Type-1 DTI codes (2)	$A\left((2^{\lambda}-1)^{\ell} - 1, 2^{\ell}\right) \leqslant 2^{\lambda\ell} - 1$	$O(s^{1-\frac{1}{\ell}})$
Constant weight codes	$A\left(\binom{n}{2}, 3\right) \leqslant \binom{n}{2} + n$	$O(s^{\frac{1}{2}})$

Results for fixed s and k

- $A(s,k_1+k_2) \leq A(s,k_1) + A(s,k_2)$
- $A(s_1+s_2,k) \leq A(s_1,k) + A(s_2,k)$
- $A(s,k) \leq A(s,k+1) - 1$
- $A(s,k) \leq A(s+1,k) - 1$
- If k is odd then $A(s,k+1)=A(s,k)+1$
- $A(s,k) \geq (2^s-1)\cdot k/2^{s-1}$ with equality iff $2^{s-1}|k$

s\k	2		3		4		6		8		10		12		14		16	
1	2*	2.00	3*	3.00	4*	4.00	6*	6.00	8*	8.00	10*	10.0	12*	12.0	14*	14.0	16*	16.0
2	3*	1.50	5*	2.50	6*	3.00	9*	4.50	12*	6.00	15*	7.50	18*	9.00	21*	10.5	24*	12.0
3	4*	1.33	6*	2.00	7*	2.33	11*	3.67	14*	4.67	18*	6.00	21*	7.00	25*	8.33	28*	9.33
4	5*	1.25	8	2.00	9	2.25	12*	3.00	15*	3.75	20	5.00	24	6.00	27*	6.75	30*	7.50
5	6*	1.20	10	2.00	11	2.20	13	2.60	19	3.80	24	4.80	26	5.20	29	5.80	31*	6.20
6	7*	1.17	11	1.83	12	2.00	14	2.33	21	3.50	26	4.33	28	4.67	35	5.83	40	6.67
7	8*	1.14	12	1.71	13	1.86	15	2.14	23	3.29	28	4.00	30	4.29	38	5.43	43	6.14
8	9*	1.13	13	1.63	14	1.75	20	2.50	28	3.50	34	4.25	40	5.00	48	6.00	54	6.75
9	10*	1.11	14	1.56	15	1.67	23	2.56	30	3.33	38	4.22	45	5.00	53	5.89	60	6.67
10	11*	1.10	17	1.70	18	1.80	24	2.40	35	3.50	41	4.10	48	4.80	57	5.70	61	6.10
11	12*	1.09	19	1.73	20	1.82	25	2.27	37	3.36	42	3.82	50	4.55	62	5.64	67	6.09
12	13*	1.08	20	1.67	21	1.75	26	2.17	39	3.25	43	3.58	52	4.33	64	5.33	69	5.75
13	14*	1.08	21	1.62	22	1.69	27	2.08	41	3.15	44	3.38	54	4.15	66	5.08	71	5.46
14	15*	1.07	22	1.57	23	1.64	29	2.07	43	3.07	45	3.21	58	4.14	68	4.86	74	5.29
15	16*	1.07	23	1.53	24	1.60	34	2.27	44	2.93	46	3.07	62	4.13	70	4.67	80	5.33
16	17*	1.06	24	1.50	25	1.56	37	2.31	45	2.81	47	2.94	64	4.00	72	4.50	84	5.25
17	18*	1.06	27	1.59	28	1.65	38	2.24	46	2.71	48	2.82	66	3.88	76	4.47	86	5.06
18	19*	1.06	28	1.56	29	1.61	39	2.17	47	2.61	49	2.72	68	3.78	78	4.33	88	4.89
19	20*	1.05	29	1.53	30	1.58	40	2.11	48	2.53	50	2.63	70	3.68	80	4.21	90	4.74
20	21*	1.05	30	1.50	31	1.55	41	2.05	49	2.45	51	2.55	72	3.60	82	4.10	92	4.60
21	22*	1.05	31	1.48	32	1.52	42	2.00	50	2.38	52	2.48	74	3.52	84	4.00	94	4.48
22	23*	1.05	32	1.45	33	1.50	47	2.14	51	2.32	53	2.41	76	3.45	86	3.91	100	4.55
23	24*	1.04	33	1.43	34	1.48	50	2.17	52	2.26	54	2.35	78	3.39	88	3.83	104	4.52
24	25*	1.04	34	1.42	35	1.46	51	2.13	53	2.21	55	2.29	80	3.33	90	3.75	106	4.42
25	26*	1.04	35	1.40	36	1.44	52	2.08	54	2.16	56	2.24	82	3.28	92	3.68	108	4.32
26	27*	1.04	38	1.46	39	1.50	53	2.04	55	2.12	57	2.19	84	3.23	96	3.69	110	4.23
27	28*	1.04	39	1.44	40	1.48	54	2.00	56	2.07	58	2.15	86	3.19	98	3.63	112	4.15
28	29*	1.04	40	1.43	41	1.46	55	1.96	57	2.04	59	2.11	88	3.14	100	3.57	114	4.07
29	30*	1.03	41	1.41	42	1.45	56	1.93	58	2.00	60	2.07	90	3.10	102	3.52	116	4.00
30	31*	1.03	42	1.40	43	1.43	57	1.90	59	1.97	61	2.03	92	3.07	104	3.47	118	3.93
31	32*	1.03	43	1.39	44	1.42	58	1.87	60	1.94	62	2.00	94	3.03	106	3.42	120	3.87
32	33*	1.03	44	1.38	45	1.41	59	1.84	61	1.91	63	1.97	96	3.00	108	3.38	122	3.81

Conclusion & Ongoing/Future Work

- A model for distributed PIR
- Coded PIR: a general scheme to emulate conventional PIR protocols over distributed storage
- k-server PIR codes
- More results
  - Extensions for non-binary
  - Robust PIR
  - t-private PIR
  - PIR array codes

Thanks for your attention…!!!

Any Queries ??