PIR with Low Storage Overhead: Coding instead of Replication

Arman Fazeli\textsuperscript{(1)}, Alexander Vardy\textsuperscript{(1)}, Eitan Yaakobi\textsuperscript{(2)}

\textsuperscript{(1)} University of California San Diego \hspace{2cm} \textsuperscript{(2)} Technion Israel Institute of Technology
What's Private Information Retrieval (PIR)?

- Alice wants to read the $i^{th}$ bit from a database $x=(x_1,...,x_n)$.
- 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 \( \mathbf{a} \cdot \mathbf{x} = \sum a_j x_j \)
- Second server returns \( (\mathbf{e}_i + \mathbf{a}) \cdot \mathbf{x} = x_i + \sum a_j x_j \)
- 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 $x$.
- 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
  
  $$A(k,j,x_1,q_j)+A(k,j,x_2,q_j) = A(k,j,x_1+x_2,q_j)$$
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, Journal 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
  \( x_1 = (x_1, \ldots, x_{n/2}) \), \( x_2 = (x_{n/2+1}, \ldots, x_n) \)
- Store in three servers \( x_1 \), \( x_2 \), \( x_1 + x_2 \)
- Assume Alice wants to read bit \( i \in [n/2] \)
- Alice chooses \( a = (a_1, \ldots, a_{n/2}) \)
- First server receives \( a \) and returns \( a \cdot x_1 = \sum a_j x_j \)
- Second server receives \( a + e_i \) and returns \( (e_i + a) \cdot x_2 = x_{n/2+i} + \sum a_j x_{n/2+j} \)
- Third server receives \( a + e_i \) and returns \( (e_i + a) (x_1 + x_2) = x_i + x_{n/2+i} + \sum a_j x_j + \sum a_j x_{n/2+j} \)
- Alice calculates \( a \cdot x_1 + (e_i + a) \cdot x_2 + (e_i + a) (x_1 + x_2) = x_i \)
- 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 \(x_1,...,x_s\)
- \(m\) servers \(S_1,...,S_m\), each stores a function of \(x_1,...,x_s\)
- Alice wants to retrieve \(x_i\), without revealing \(i\)
- A protocol \(P^*\) with three algorithms \(P^*(Q^*,A^*,C^*)\)
  - Alice randomly generates \(m\) queries \(Q^*(m,s,n;i)=(q_1,...,q_m)\) and sends to the servers
  - Each server responds with \(a_j=A^*(m,s,j,x,q_j)\)
  - Alice computes \(x_i\) by \(C^*(m,s,n;i,a^*_1,...,a^*_m)\)

Requirements:

- **Privacy**: each server learns no information about \(i\)
- **Correctness**: \(C(m,s,n;i,a^*_1,...,a^*_m) = x_i\)

Communication complexity: number of uploaded and downloaded bits

Storage overhead: ratio between stored and information bits
The database is partitioned into four parts
\[ x_1 = (x_1, \ldots, x_{n/4}), x_2 = (x_{n/4+1}, \ldots, x_{n/2}), x_3 = (x_{n/2+1}, \ldots, x_{3n/4}), x_4 = (x_{3n/4+1}, \ldots, x_n) \]

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 \( Q^*(8, 4, n; i) = (q_1, q_2, q_3, q_2, q_3, q_2, q_3) \)

Retrieval:
- \( a'_1 = a_1 = A(3, 1, x_1, q_1) \)
- \( a'_2 = a_2 + a_5 = A(3, 2, x_2, q_2) + A(3, 2, x_1 + x_2, q_2) = A(3, 2, x_2, q_2) \)
- \( a'_3 = a_4 + a_8 = A(3, 3, x_4, q_3) + A(3, 3, x_4 + x_1, q_3) = A(3, 3, x_1, q_3) \)

- \( C^*(8, 4, n; i, a_1, \ldots, a_8) = C(3, n/4; i, a'_1, a'_2, a'_3) = C(3, n/4; i, A(3, 1, x_1, q_1), A(3, 2, x_2, q_2), A(3, 3, x_1, q_3)) = x_i \)

Correctness and Privacy: from \( P \)

Communication complexity: same as the one for \( P \)

Storage overhead: 2 (instead of 3)

---

**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 \( t \)
  - Each server responds with \( a_j = A^*(m, s, j, x_i, q_j) \)
  - 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
- Storage overhead: ratio between stored and information bits

<table>
<thead>
<tr>
<th>Server</th>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( q_2 )</td>
<td>( a_1 = A^*(8, 4, 1, c_1, q_2) = A(3, 2, x_1, q_2) )</td>
</tr>
<tr>
<td>2</td>
<td>( q_1 )</td>
<td>( a_2 = A^*(8, 4, 2, c_2, q_1) = A(3, 1, x_2, q_1) )</td>
</tr>
<tr>
<td>3</td>
<td>( q_3 )</td>
<td>( a_3 = A^*(8, 4, 3, c_3, q_3) = A(3, 3, x_3, q_3) )</td>
</tr>
<tr>
<td>5</td>
<td>( q_2 )</td>
<td>( a_5 = A^*(8, 4, 5, c_5, q_2) = A(3, 2, c_5 = x_1 + x_2, q_2) )</td>
</tr>
<tr>
<td>6</td>
<td>( q_3 )</td>
<td>( a_6 = A^*(8, 4, 6, c_6, q_3) = A(3, 3, c_6 = x_2 + x_3, q_3) )</td>
</tr>
<tr>
<td>8</td>
<td>( q_3 )</td>
<td>( a_8 = A^*(8, 4, 8, c_8, q_3) = A(3, 3, c_8 = x_4 + x_1, q_3) )</td>
</tr>
</tbody>
</table>
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 $R_{i,1},..., R_{i,k}$ 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.

```
( 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)
```
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
  - $A(v^2,3) \leq (v+1)^2-1 = v^2+2v$
  - **Storage overhead:** $(v^2+2v)/v^2 = 1+2/v \Rightarrow 1$ 😊
  - For any $s$, $k$: $A(s,k) \leq s+(k-1)s^{(k-2)/(k-1)}$
  - **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 $R_{i,1},..., R_{i,k}$ 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, \[ A(s,k) \approx s + s^{1/2} \]
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 $k$, $A(s,k) \approx s + s^{\frac{1}{2}}$

$$h_3 = (0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1)$$
$$h_{1+5} = (0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1)$$
$$h_{0+2+6} = (1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1)$$
$$h_7 = (0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1)$$
### Summary: Constructions with Fixed $k$

<table>
<thead>
<tr>
<th>Code construction</th>
<th>Upper bound on $A(s, k)$</th>
<th>Asymptotic redundancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cubic construction</td>
<td>$A(s, k) \leq s + (k - 1) \left[ s^{\frac{1}{k-1}} \right]^{k-2}$</td>
<td>$O(s^{1-\frac{1}{k-1}})$</td>
</tr>
<tr>
<td>Steiner System</td>
<td>$A \left( \frac{n(n-1)}{(k-1)(k-2)}, k \right) \leq n + \frac{n(n-1)}{(k-1)(k-2)}$</td>
<td>$O(s^{\frac{1}{2}})$</td>
</tr>
<tr>
<td>Type-1 DTI codes (1)</td>
<td>$A \left( 2^{2\theta \ell} - (2^{\theta+1} - 1)^\ell, 2^\ell + 2 \right) \leq 2^{2\theta \ell} - 1$</td>
<td>$O(s^{\frac{1}{2}})$</td>
</tr>
<tr>
<td>Type-1 DTI codes (2)</td>
<td>$A \left( (2^\lambda - 1)^\ell - 1, 2^\ell \right) \leq 2^{\lambda \ell} - 1$</td>
<td>$O(s^{1-\frac{1}{\ell}})$</td>
</tr>
<tr>
<td>Constant weight codes</td>
<td>$A \left( \binom{n}{2}, 3 \right) \leq \binom{n}{2} + n$</td>
<td>$O(s^{\frac{1}{2}})$</td>
</tr>
</tbody>
</table>
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 $2^{s-1} | k$
<table>
<thead>
<tr>
<th>s \ k</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1.50</td>
<td>5</td>
<td>2.50</td>
<td>6</td>
<td>3.00</td>
<td>9</td>
<td>4.50</td>
<td>12</td>
<td>6.00</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1.33</td>
<td>6</td>
<td>2.00</td>
<td>7</td>
<td>2.33</td>
<td>11</td>
<td>3.67</td>
<td>14</td>
<td>4.67</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1.25</td>
<td>8</td>
<td>2.00</td>
<td>9</td>
<td>2.25</td>
<td>12</td>
<td>3.00</td>
<td>15</td>
<td>3.75</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1.20</td>
<td>10</td>
<td>2.00</td>
<td>11</td>
<td>2.20</td>
<td>13</td>
<td>2.60</td>
<td>19</td>
<td>3.80</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>1.17</td>
<td>11</td>
<td>1.83</td>
<td>12</td>
<td>2.00</td>
<td>14</td>
<td>2.33</td>
<td>21</td>
<td>3.50</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1.14</td>
<td>12</td>
<td>1.71</td>
<td>13</td>
<td>1.86</td>
<td>15</td>
<td>2.14</td>
<td>23</td>
<td>3.29</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>1.13</td>
<td>13</td>
<td>1.63</td>
<td>14</td>
<td>1.75</td>
<td>20</td>
<td>2.50</td>
<td>28</td>
<td>4.00</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>1.11</td>
<td>14</td>
<td>1.56</td>
<td>15</td>
<td>1.67</td>
<td>23</td>
<td>2.56</td>
<td>30</td>
<td>4.22</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>1.10</td>
<td>17</td>
<td>1.70</td>
<td>18</td>
<td>1.80</td>
<td>24</td>
<td>2.40</td>
<td>35</td>
<td>4.10</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>1.09</td>
<td>19</td>
<td>1.73</td>
<td>20</td>
<td>1.82</td>
<td>25</td>
<td>2.27</td>
<td>37</td>
<td>4.55</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>1.08</td>
<td>20</td>
<td>1.67</td>
<td>21</td>
<td>1.75</td>
<td>26</td>
<td>2.17</td>
<td>39</td>
<td>4.35</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>1.08</td>
<td>21</td>
<td>1.62</td>
<td>22</td>
<td>1.69</td>
<td>27</td>
<td>2.08</td>
<td>41</td>
<td>4.15</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>1.07</td>
<td>22</td>
<td>1.57</td>
<td>23</td>
<td>1.64</td>
<td>29</td>
<td>2.07</td>
<td>43</td>
<td>4.07</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>1.07</td>
<td>23</td>
<td>1.53</td>
<td>24</td>
<td>1.60</td>
<td>30</td>
<td>2.27</td>
<td>44</td>
<td>4.23</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>1.06</td>
<td>24</td>
<td>1.50</td>
<td>25</td>
<td>1.56</td>
<td>37</td>
<td>2.31</td>
<td>45</td>
<td>4.81</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>1.06</td>
<td>27</td>
<td>1.59</td>
<td>28</td>
<td>1.65</td>
<td>38</td>
<td>2.24</td>
<td>46</td>
<td>4.81</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>1.06</td>
<td>28</td>
<td>1.56</td>
<td>29</td>
<td>1.61</td>
<td>39</td>
<td>2.17</td>
<td>47</td>
<td>2.72</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>1.05</td>
<td>29</td>
<td>1.53</td>
<td>30</td>
<td>1.58</td>
<td>40</td>
<td>2.11</td>
<td>48</td>
<td>2.63</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>1.05</td>
<td>30</td>
<td>1.50</td>
<td>31</td>
<td>1.55</td>
<td>41</td>
<td>2.05</td>
<td>49</td>
<td>2.53</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>1.05</td>
<td>31</td>
<td>1.48</td>
<td>32</td>
<td>1.52</td>
<td>42</td>
<td>2.00</td>
<td>50</td>
<td>2.38</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>1.05</td>
<td>32</td>
<td>1.45</td>
<td>33</td>
<td>1.50</td>
<td>47</td>
<td>2.14</td>
<td>51</td>
<td>2.32</td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>1.04</td>
<td>33</td>
<td>1.43</td>
<td>34</td>
<td>1.48</td>
<td>50</td>
<td>2.17</td>
<td>52</td>
<td>2.26</td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td>1.04</td>
<td>34</td>
<td>1.42</td>
<td>35</td>
<td>1.46</td>
<td>51</td>
<td>2.13</td>
<td>53</td>
<td>2.21</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>1.04</td>
<td>35</td>
<td>1.40</td>
<td>36</td>
<td>1.44</td>
<td>52</td>
<td>2.08</td>
<td>54</td>
<td>2.16</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>1.04</td>
<td>38</td>
<td>1.46</td>
<td>39</td>
<td>1.50</td>
<td>53</td>
<td>2.04</td>
<td>55</td>
<td>2.12</td>
</tr>
<tr>
<td>27</td>
<td>28</td>
<td>1.04</td>
<td>39</td>
<td>1.44</td>
<td>40</td>
<td>1.48</td>
<td>54</td>
<td>2.00</td>
<td>56</td>
<td>2.07</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>1.04</td>
<td>40</td>
<td>1.43</td>
<td>41</td>
<td>1.46</td>
<td>55</td>
<td>1.96</td>
<td>57</td>
<td>2.04</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>1.03</td>
<td>41</td>
<td>1.41</td>
<td>42</td>
<td>1.45</td>
<td>56</td>
<td>1.93</td>
<td>58</td>
<td>2.00</td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>1.03</td>
<td>42</td>
<td>1.40</td>
<td>43</td>
<td>1.43</td>
<td>57</td>
<td>1.90</td>
<td>59</td>
<td>1.97</td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>1.03</td>
<td>43</td>
<td>1.39</td>
<td>44</td>
<td>1.42</td>
<td>58</td>
<td>1.87</td>
<td>60</td>
<td>1.94</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>1.03</td>
<td>44</td>
<td>1.38</td>
<td>45</td>
<td>1.41</td>
<td>59</td>
<td>1.84</td>
<td>61</td>
<td>1.91</td>
</tr>
</tbody>
</table>
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 ??