Amortized PIR via Batch Codes

Eyal Kushilevitz

Talk is based on:

"Batch Codes and their Applications"
Ishai, K, Ostrovsky, Sahai
STOC 2004

PIR = Private Information Retrieval

Private Information Retrieval (PIR)

Data: n-bit string x

User: wishes

– to retrieve x_i; and

– to keep i private

Remark: hides identity of data-items; not the mere existence of interaction with the user

Some "Solutions"

Solution 1: user asks for a copy of x
Drawback: communication complexity is n bits

Solution 2: ask for additional random indices
Drawback: gives partial information

Solution 3: Employ general crypto machinery to compute $f(x, i) = x_i$ privately
Drawback: highly inefficient (poly(n) comm.)

Solution 4: Anonymity (e.g., via anonymizers)
Drawback: addresses different concern: hides identity of user; not that x_i is retrieved (+trust issues)

[CGKS]: information-theoretic PIR in this setting requires $\Omega(n)$ bits

PIR – Setting and Assumptions

Information-Theoretic PIR [CGKS,...]

– k *separated* servers $S_1, \ldots, S_k$

– Replication: identical copies of x

– Each *single* server gets no info on i

Computational PIR [CG,KO,CMS,...]

– Computational Assumptions

– Computational Privacy

– Replication not needed

Known Communication Upper-Bounds

Information-Theoretic PIR

– k servers, $n^{1/\Omega(k)}$ comm. [CGKS,Amb,...,BIKR]

– $\log n$ servers, polylog(n) comm. [BF,CGKS]

Computational PIR

communication poly($\log n$,security-parameter) under appropriate computational assumption [CG,KO,CMS,...]

PIR – Related Work

- Extensions:
  - Symmetric PIR ($\approx \binom{n}{1}$-OT) [GIKM,NP]
  - t-privacy [CGKS,IK,BI]

$\vdots$

- Locally decodable codes (with k probes) [KT]
- Lower bounds for PIR [Mann] and LDCs [KT,GKST,AJKLR]
- Different settings [OS,GGM,DIO,BIM,...]
- Applications of PIR [CGN,NN,FIMNSW,CIKRRW,...]

Amortization

What follows applies to both types of PIR.

- Communication efficient PIR
  $\Rightarrow$ time complexity at servers a bottleneck.
  (current protocols: linear time *per query*)
- Linear time is a lower bound
- Solutions:
  - Preprocessing [BIM] (mainly multi-server)
  - amortization
    (with/without pre-processing)

Rest of the Talk:

- Amortization via hashing
- Amortization via batch codes

Hashing-based Amortization

User wishes to retrieve k indices $i_1, \ldots, i_k$.

Idea: Use hashing as a "random" partition h of indices into $m \approx k$ buckets of $\approx n/m$ size.
h independent of $i_1, \ldots, i_k \Rightarrow$ w/prob $\geq 1 - 2^{-O(\sigma)}$ in each bucket $\leq \alpha = \sigma \log m$ of i_j's

Meta Scheme:

1. User picks $h : [n] \to [m]$ from appropriate $\mathcal{H}$ and sends h to server(s)
2. Apply PIR α times to each bucket

Analysis: "Small" number of buckets; each is "not-too-large"; need only "few" bits from it.
$\Rightarrow$ time$= m \cdot T(n/m) \cdot \sigma \log k \approx n \cdot \sigma \log k$

Hashing-based Amortization (cont.)

This is a pretty good solution but...

Some disadvantages:

- Not perfect – non-perfect correctness; can be replaced by non-perfect privacy.
- Useless for "small" k.
- Desirable: hash "once and for all"
  But $\forall h$ $\exists$"bad" k-tuples of queries.

Batch Codes

Definition: (n, N, k, m) batch-code maps n-bit string x to m strings ("buckets") of total length N s.t. $\forall$ k-tuple of bits of x (a "batch") can be decoded using at most one bit from each bucket.

Motivation: natural notion
+ applications: Load-balancing, Amortized PIR...

More general definitions: e.g., allow t reads from each bucket; multisets

Some Examples

Naive Codes:

$(n, N = kn, k, m = k)$ (replication)

$(n, N = n, k, m = n)$ (one bit per bucket)

Goal: Simultaneously good “rate” & few buckets

Example: $(n, 1.5n, 2, 3)$ batch-code:

$x = (L, R)$ where $|L| = |R| = n/2$.

Encoding: $x \to (L, R, L \oplus R)$.

Amortized PIR from Batch Codes

Fix a batch code.

Meta Scheme

1. Preprocessing: Server(s) apply the code to x and get m buckets.
2. User, based on $i_1, \ldots, i_k$ computes the indices it needs from each bucket
3. For each bucket:
   User and Server(s) apply PIR scheme to retrieve an index from this bucket

Complexity:
time $\sum_{i=1}^{m} T(N_i)$, communication $\sum_{i=1}^{m} C(N_i)$
No error

Remark: independent of the PIR protocol

Some specific amortization results

- Using $(n, N = O(n), k, m = O(k^{1+\varepsilon}))$ code

$\Rightarrow$ $\exists$PIR scheme to retrieve k indices,
w/total linear time at servers
and communication $O(k^{1+\varepsilon}) \cdot C(N/k^{1+\varepsilon})$.

- Using efficient transformation

$$\binom{n}{k}\text{-PIR} \Rightarrow \binom{n}{k}\text{-OT} \quad \text{[NP]}$$

$\Rightarrow$ amortized $\binom{n}{k}$-OT

Open Problems

- Amortization in the multi-user setting (some positive results shown in [BIM])
- ...