Incremental Offline/Online PIR

Yiping Ma    Ke Zhong    Tal Rabin    Sebastian Angel

Appeared at USENIX Security 2022
Private Information Retrieval (PIR): Basics
[CGKS95, KO97]

**Server** has a database
\[ x = x_1 x_2 \ldots x_n \in \{0,1\}^n \]

**Client** wants to read an item \( x_i \) without revealing the index \( i \)

**Correctness**
“Client gets the bits it wants”
(with overwhelming probability)

**Security**
“Server learns nothing from client’s queries”
(information-theoretic or computational)

We will later discuss privacy against clients.
Private Information Retrieval (PIR): Why interesting?

Server has a database
\[ x = x_1 x_2 \ldots x_n \in \{0,1\}^n \]

Client wants to read an item \( x_i \)
without revealing the index \( i \)

Black-box extensions
- longer entries [CGN98],
- keyword query [CGN98],
- batched fetching [IKOS04], etc.

Real-world Applications
Private Information Retrieval (PIR): Applications

**Server** has a database
\[ x = x_1 x_2 \ldots x_n \in \{0,1\}^n \]

**Client** wants to read an item \( x_i \)
without revealing the index \( i \)

---

**Systems built from PIR**

- Private contact discovery [DP5, PETS15]
- Private stream service [Popcorn, NSDI16]
- Metadata-private messaging [Pung, OSDI16]
- Private search [DORY, OSDI20; Coeus, SOSP21]
- Safe browsing [Checklist, Sec21]
- Private key-value store [Pantheon, VLDB23]

... and many more
Private Information Retrieval (PIR): Applications

**Client** wants to read an item $x_i$ without revealing the index $i$

**Server** has a database

$$x = x_1 x_2 ... x_n \in \{0,1\}^n$$

**Query**

**Answer**

Systems built from PIR

Private contact discovery [DP5, PETS15]

Private stream service [Popcorn, NSDI16]

Metadata-private messaging [Pung, OSDI16]

Private search [DORY, OSDI20; Coeus, SOSP21]

Safe browsing [Checklist, Sec21]

Private key-value store [Pantheon, VLDB23]

... and many more
Private Information Retrieval (PIR): Efficiency

Server has a database
\( x = x_1 x_2 \ldots x_n \in \{0,1\}^n \)

Client wants to read an item \( x_i \)
without revealing the index \( i \)

Cost is critical for applications:

- Communication
- Computation
  “work” [BIM04]:
  \#bits the server reads
- Storage

Measure in terms of database size \( n \)
Reducing **Communication** Cost for PIR

- Trivial PIR has **linear** communication
Reducing \textbf{Communication} Cost for PIR

- Trivial PIR has linear communication
- PIR with \textit{sublinear} communication

\begin{itemize}
  \item Non-colluding servers, database replicated
    - Information-theoretic [CGKS95, BIK04, Yek08, Efr12, DG16, ...]
    - Computational [BGI16, ...]
  \item Single server
    - Information-theoretic [KO97]
      - Not possible unless linear communication
    - Computational [KO97, CMS99, KO00, GR05, OS07, ...]
\end{itemize}
Reducing **Computation** Cost for PIR?

- Trivial PIR has linear computation or “work” [BIM00]
- PIR with sublinear computation?

**A lower bound** [BIM00]:

If the database has no redundancy (i.e., no extra storage at the server or the client), answering a single query requires in expectation $\Omega(n)$ total server “work”, where $n$ is database size.
Reducing **Computation** Cost for PIR?

- Trivial PIR has linear computation or “work” [BIM00]
- PIR with sublinear computation?

**A lower bound** [BIM00]:

If the database has no redundancy (i.e., no extra storage at the server or the client), answering a single query requires in expectation $\Omega(n)$ total server “work”, where $n$ is database size.

_linear computation makes PIR hard to scale to large databases_
Reducing **Computation** Cost for PIR?

- Trivial PIR has linear computation or “work” [BIM00]
- PIR with sublinear computation: hope?

**A lower bound [BIM00]:**

If the database has no redundancy (i.e., no extra storage at the server or the client), answering a single query requires in expectation $\Omega(n)$ total server “work”, where $n$ is database size.
Reducing **Computation** Cost for PIR: An important idea

Push (necessary) linear work to an offline stage and generate **hints** along the way...

and get sublinear computation for each query with the hints!

[BIM00, CHR17, BIPW17, HOWW18, PPY18, CK20, SACM21, CHK22, ...]
Reducing **Computation** Cost for PIR: An important idea

Push (necessary) linear work to an offline stage and generate *hints* along the way...

and get sublinear computation for each query with the hints!

[**BIM00**, **CHR17**, **BIPW17**, **HOWW18**, **PPY18**, **CK20**, **SACM21**, **CHK22**, ...]

**PIR with preprocessing**

Superlinear-sized hints at the server
Reducing **Computation** Cost for PIR: An important idea

Push (necessary) linear work to an offline stage and generate **hints** along the way...

and get sublinear computation for each query with the hints!

[BIM00, CHR17, BIPW17, HOWW18, PPY19, CK20, SACM21, CHK22, ...]

Our work is based on this flavor

Offline/Online model

Sublinear-sized hints at the client
PIR with Preprocessing: Are we done?

- Preprocessing phase: (super) linear computation, generate hints
  - Preprocessing can be done only by the servers [BIM00, BIPW17, CHR17] or interactive with clients (per-client) [PPY18, CK20, KC21, SACM21, CHK22]
  - Hints can be stored at the server or the clients

- Query phase: sublinear computation for each query utilizing the hints
  - #Queries can be unbounded or polynomially many

Sublinear computation and sublinear communication
PIR with Preprocessing in Applications
PIR with Preprocessing in Applications

9:00 AM Preprocessing

9:10 AM Answer 3 queries

9:12 AM Add 2 more items

Hint no longer work!
PIR with Preprocessing: More things to do

Databases are not static, but old hints no longer work!
Features of common applications

- A handful of changes (small compared to the database size)
- Changes periodically happen
PIR with Preprocessing: Patches

**Features of common applications**
- A handful of changes (small compared to the database size)
- Changes periodically happen

**Computation linear in the database size per change**

**Rerun preprocessing**
Features of common applications
• A handful of changes (small compared to the database size)
• Changes periodically happen

PIR with Preprocessing: Patches

Rerun preprocessing

Computation linear in the database size per change

More databases

Make many queries
Our approach to handle dynamic database preserves all the properties of the solutions for the static database
PIR with Mutable Preprocessing

Preprocessing for updates

A single database

Minimum necessary server work: linear in additions

A small hint
PIR with Mutable Preprocessing

A single database

Preprocessing for updates

Minimum necessary server work:
linear in #additions

A new hint for the updated database
PIR with Mutable Preprocessing

Preprocessing for updates

A single database

Query phase

Sublinear computation

Sublinear communication
Rest of This Talk

• Motivation

• Our solution: update existing hints at a cost proportional to the changes
  • Based on [CK20], [SACM21]
  • Experimental evaluation

• Open questions
Rest of This Talk

• Motivation

• Our solution: update existing hints at a cost proportional to the changes
  • Based on [CK20], [SACM21]
  • Experimental evaluation

• Discussion
Rest of This Talk

• Motivation

• Our solution: update existing hints at a cost proportional to the changes
  • Based on [CK20], [SACM21]
  • Experimental evaluation

• Discussion
A Two-Server Offline/Online PIR [CK20]

- Generate “hints” in offline phase for online queries

Non-colluding servers, database replicated

Offline server Online server
A Two-Server Offline/Online PIR [CK20]

• Generate “hints” in offline phase for online queries
A Two-Server Offline/Online PIR [CK20]

Offline phase

\[ T = \lambda \sqrt{n} \] random subsets of \([n]\), each of size \(s = \sqrt{n}\). Denote as \(S_1, S_2, ..., S_T\).

Notation \([n] := \{1, 2, ..., n\}\)

\(\lambda\) ensures the subsets cover \([n]\) with high probability.

Database \(x = x_1 x_2 ... x_n\) replicated among two servers

Offline server

Online server
A Two-Server Offline/Online PIR [CK20]

Offline phase

\[ T = \lambda \sqrt{n} \]

\( \lambda \) ensures the subsets cover \([n]\) with high probability.

Client generates \( T = \lambda \sqrt{n} \) random subsets of \([n]\), each of size \( s = \sqrt{n} \).

deno as \( S_1, S_2, ..., S_T \).
A Two-Server Offline/Online PIR [CK20]

Offline phase

$S_1, S_2, ..., S_T$

Offline server

Online server
A Two-Server Offline/Online PIR [CK20]

Client generates random subsets of \([n]\), \(S_1, S_2, \ldots, S_T\), all of size \(s\).

\(k_1, k_2, \ldots, k_T\)
A Two-Server Offline/Online PIR [CK20]

Offline phase

where $h_j = \bigoplus_{\ell \in S_j} x_{\ell}$ ($j \in [T]$)
A Two-Server Offline/Online PIR [CK20]

Offline phase

\[(S_1, h_1), (S_2, h_2), \ldots, (S_T, h_T)\]
where \(h_j = \bigoplus_{\ell \in S_j} x_\ell \quad (j \in [T])\)
A Two-Server Offline/Online PIR [CK20]

Offline phase

\( O(\lambda n) \) server computation
\( O(\lambda \sqrt{n}) \) client storage

\((S_1, h_1), (S_2, h_2), \ldots, (S_T, h_T)\)

where \( h_j = \bigoplus_{\ell \in S_j} x_{j,\ell} \) \((j \in [T])\)

Offline server

Online server
A Two-Server Offline/Online PIR [CK20]

Online phase

Query index $i \in [n]$: find a set $S_j$ that contains $i$

$$(S_1, h_1), (S_2, h_2), \ldots, (S_T, h_T)$$

where $h_j = \bigoplus_{\ell \in S_j} x_\ell \ (j \in [T])$

Membership check: can be done efficiently via one PRP call
A Two-Server Offline/Online PIR [CK20]

Online phase

Query index $i \in [n]:$ find a set $S_j$ that contains $i$

$(S_j, h_j)$

Online server

Offline server
A Two-Server Offline/Online PIR [CK20]

Online phase

Query index $i \in [n]$: find a set $S_j$ that contains $i$

$(S_j, h_j)$

$$h^* = \bigoplus_{j \in S_j \setminus \{i\}} x_j$$

Offline server

Online server
With exact probability $1 - \frac{s-1}{n}$, remove $i$ from $S_j$;
With the remaining probability, remove another random element from $S_j$. 

$= S_j^*$. 

A Two-Server Offline/Online PIR [CK20]
**A Two-Server Offline/Online PIR [CK20]**

With exact probability \(1 - \frac{s-1}{n}\), remove \(i\) from \(S_j\);

With the remaining probability, remove another random element from \(S_j\).

\[ S_j^* = \{19, 51, 3, 67, \ldots \} \]

\(= S_j^*\)
A Two-Server Offline/Online PIR [CK20]

Online phase

\[(S_j, h_j)\]

With exact probability \(1 - \frac{s-1}{n}\), remove \(i\) from \(S_j\);
With the remaining probability, remove another random element from \(S_j\).

\(\text{Offline server}
\begin{align*}
&\text{A random subset of} \\
&\text{[}n\text{] with size } s - 1 \\
&\text{\(S_j^*\) generated from} \\
&\text{the query on } i
\end{align*}
\= S_j^*

\begin{align*}
\text{Online server}
\end{align*}
A Two-Server Offline/Online PIR [CK20]

Online phase

$O(\sqrt{n})$ server computation

$(S_j, h_j)$

Compute $x_i = h_j^* \oplus h_j$. 

$h_j^* = \bigoplus_{\ell \in S_j^*} x_\ell$
A Two-Server Offline/Online PIR [CK20]

Correctness: with probability $1 - \frac{s-1}{n}$ (can be made negl in followup work [KC21])

Compute $x_i = h_j^* \oplus h_j$.

$(S_j, h_j)$

$h_j^* = \bigoplus_{\ell \in S_j^*} x_\ell$
A Two-Server Offline/Online PIR [CK20]

Security:
Offline phase is query-independent;
Online server sees a random subset $S_j^*$

$$(S_j, h_j)$$

Compute $x_i = h_j^* \oplus h_j$.

$h_j^* = \bigoplus_{\ell \in S_j^*} x_\ell$
A Two-Server Offline/Online PIR [CK20]

\[(S_1, h_1), \ldots, (S_j, h_j), \ldots, (S_T, h_T)\]

Compute \(x_i = h_j^* \oplus h_j.\)

\(h_j^* = \bigoplus_{\ell \in S_j^*} x_{\ell}\)

A set cannot be reused
A Two-Server Offline/Online PIR [CK20]

\[(S_1, h_1), ..., (S_j, h_j), ..., (S_T, h_T)\]

Compute \(x_i = h_j^* \oplus h_j\).
Rest of This Talk

• Motivation

• Our solution
  • Background of [CK20], [SACM21]
  • Mutable preprocessing based on [CK20]
  • Experimental evaluation

• Open questions
Rest of This Talk

• Motivation

• Our solution
  • Background of [CK20], [SACM21]
  • Mutable preprocessing based on [CK20]
  • Experimental evaluation

• Open questions
Types of Database Changes

• Additions: we will show next
• Deletions: more involved
• In-place edits: easy to see
Update Hints for Additions

Start with a toy version: add a single item

$S_1, S_2, ..., S_T \leftarrow \{0, 1\}^n$

$h_1, h_2, ..., h_T \leftarrow [n]$
Update Hints for Additions

Start with a toy version: add a single item

$$S_1, S_2, ..., S_T \leftarrow [n]$$
$$h_1, h_2, ..., h_T$$

$$S'_1, S'_2, ..., S'_T \leftarrow [n + 1]$$
$$h'_1, h'_2, ..., h'_T$$
Update Hints for Additions

Start with a toy version: add a single item

\[ S_1, S_2, \ldots, S_T \leftarrow [n] \]
\[ h_1, h_2, \ldots, h_T \]

With small server computation

\[ S'_1, S'_2, \ldots, S'_T \leftarrow [n + 1] \]
\[ h'_1, h'_2, \ldots, h'_T \]
Update Hints for Additions

Start with a toy version: add a single item

Client updates each $S_j$ for $j \in [T]$:
- With probability $p = s/(n + 1)$, use $n + 1$ to replace a random element in $S_j$;
- With probability $1 - p$, do nothing.
Update Hints for Additions

Start with a toy version: add a single item

\[ S_1, S_2, ..., S_T \leftarrow [n] \]
\[ h_1, h_2, ..., h_T \]

\[ \oplus \text{ for a changed set} \]

\[ S_1', S_2', ..., S_T' \leftarrow [n + 1] \]
\[ h_1', h_2', ..., h_T' \]

Analysis: roughly \( s/n \) portion of sets will change by one element; note that \( s \) is very small compared to \( n \)
**Update Hints for Additions**

Start with a toy version: add a single item

\[ S_1, S_2, \ldots, S_T \leftarrow [n] \]

\[ h_1, h_2, \ldots, h_T \]

\[ S_1', S_2', \ldots, S_T' \leftarrow [n + 1] \]

\[ h_1', h_2', \ldots, h_T' \]

**Analysis:** roughly \( s/n \) portion of sets will change by one element; note that \( s \) is very small compared to \( n \)

**Cost:** in expectation \( \lambda \) XOR operations; whereas redoing preprocessing takes \( O(\lambda n) \) XOR operations

Assuming \( s = \sqrt{n} \) and \( T = \lambda \sqrt{n} \)
Update Hints for Additions

Start with a toy version: add a single item

\[ S_1, S_2, ..., S_T \leftarrow [n] \]
\[ h_1, h_2, ..., h_T \]

\[ S'_1, S'_2, ..., S'_T \leftarrow [n+1] \]
\[ h'_1, h'_2, ..., h'_T \]

Analysis: roughly \( s/n \) portion of sets will change by one element; note that \( s \) is very small compared to \( n \)

Cost: in expectation \( \lambda \) XOR operations; whereas redoing preprocessing takes \( O(\lambda n) \) XOR operations
Update Hints for Additions

- A toy version of adding a single item
- Can be extended for adding a batch of items
- Can be further extended to adding multiple batches of items
Update Hints for Additions

Adding a batch of items: recursively apply the toy approach

For each set $S_j$:
- Let $w \leftarrow \text{HG}(\text{total} = n + m, \text{featured} = m, \#\text{samples} = s)$
- Randomly choose $w$ elements in $S_j \subset \{1, \ldots, n\}$ to be replaced with
  $w$ random elements in $\{n + 1, \ldots, n + m\}$

Equivalent to sampling from hypergeometric distribution

In expectation $O(\lambda m)$ server work
Update Hints for Additions

Reducing communication: making the key representation compatible

\[ k_1, \ldots, k_T \rightarrow S_1, S_2, \ldots, S_T \]

Remember [CK20] builds pseudorandom sets from a PRP: \( K \times [n] \rightarrow [n] \): A random subset in \([n]\) with size \( s \) is represented by a key \( k \in K \), and the set is \{PRP(k, 1), PRP(k, 2), ..., PRP(k, s)\}
Update Hints for Additions

Reducing communication: making the key representation compatible

\[ k_1, \ldots, k_T \rightarrow S_1, S_2, \ldots, S_T \]

Remember [CK20] builds pseudorandom sets from a PRP: \( \mathcal{K} \times [n] \rightarrow [n] \):
A random subset in \([n]\) with size \(s\) is represented by a key \(k \in \mathcal{K}\), and the set is

\[
Set(k, n, s) := \{\text{PRP}(k, 1), \text{PRP}(k, 2), \ldots, \text{PRP}(k, s)\}
\]
Update Hints for Additions

Reducing communication: making key representation compatible

Equivalent to sampling from hypergeometric distribution

For each set $S_j$:

- Let $w \leftarrow HG(total = n + m, \text{featured} = m, \#samples = s)$
- Let $U^1 = Set(k, n, s - w)$ “kick out $w$ random old elements”
- Let $U^2 = Set(k, m, w) + n$ “add $w$ random new elements”
- Set $S_j \leftarrow U^1 \cup U^2$ “represented by “$k, s - w, w”"

Server: 2$w$ PRP calls per set
Total #PRP calls in expectation $\Theta(m)$

Client: no need to store set elements (indices) explicitly

{0, 1}^{n+m}
Update Hints for Additions

Preprocessing for updates
“Notify”

Additions happen at both servers

Existing hints $(S_1, h_1), \ldots, (S_T, h_T)$

#Additions $m$

Offline server

Online server
Update Hints for Additions

Preprocessing for updates
“Hint request”

Offline server

Online server

\[(k_1, w_1), \ldots, (k_T, w_T)\]
Update Hints for Additions

Preprocessing for updates “Hint response”

Parity difference: $\Delta_1, \ldots, \Delta_T$

$O(\lambda m)$ server computation

Offline server

Online server
Update Hints for Additions

Preprocessing for updates "Hint update"

Parity difference: \( \Delta_1, \ldots, \Delta_T \)

Client computes new hints
\((S'_1, h'_1), \ldots, (S'_T, h'_T)\)
where \( h'_j := \Delta_j \oplus h_j \).
Update Hints for Additions

Online phase
(minor change)

Stored hints: \((S_1', h_1'), \ldots, (S_T', h_T')\)

Query index \(i \in [n]\): find a set \(S_j'\) that contains \(i\)

Updated database \(\{0,1\}^{n'}\), where \(n' = n + m\).
Update Hints for Additions

Online phase (minor change)

With probability $1 - \frac{s-1}{n'}$, remove $i$ from $S_j'$; With the remaining probability remove another random element from $S_j'$.

$S_j^* = \{2, 15, 9, \ldots\}$

Updated database $\{0, 1\}^{n'}$, where $n' = n + m$. 
Update Hints for Additions

Set\((k, n', s)\)

Updated database \(\{0,1\}^{n'}\), where \(n' = n + m\).

Online phase
(refresh)

Offline server

Online server
Update Hints for Additions

Online phase (refresh)

Set\((k, n', s), h\)

Updated database \(\{0, 1\}^{n'},\) where \(n' = n + m.\)

Offline server

Online server
More Technical Details

• Supporting multiple batches of additions
• Supporting in-place edits
• Deletions: cannot actually delete when hints are stored at the client (if client is malicious)
More Technical Details

• Supporting multiple batches of additions
• Supporting in-place edits
• Deletions: cannot actually delete when hints are stored at the client (if client is malicious)
More Technical Details

• Supporting multiple batches of additions
• Supporting in-place edits
• Deletions: cannot actually delete when hints are stored at the client (if client is malicious)
More Technical Details

- Supporting multiple batches of additions
- Supporting in-place edits
- Deletions: cannot actually delete when hints are stored at the client (if client is malicious)
High-level Ideas in [SACM21]

• Instead of sampling a set with fixed size, sample each element into a set with some probability $p$
• The set size in expectation is $np$
Mutable Preprocessing from [SACM21]

- Instead of sampling a set with fixed size, sample each element into a set with some probability $p$
- The set size in expectation is $np$
- When a new item is added (hence a new index), for each set, sample the new index into the set with probability $p$

Update sets in a way compatible with the key representation
Independent work

• [KC21, Checklist]
  • Dynamic data structure, black-box construction
  • Amortize the cost over multiple added items

• Ours: make the hints mutable
  • Utilize features of specific protocols

• Depending on concrete parameters (frequency of updates, item size, etc.), provides different benefits
Evaluation: Microbenchmarks

How does our construction save **server cost**?

Results for adding 1% data:

<table>
<thead>
<tr>
<th></th>
<th>$2^{16}$</th>
<th>$2^{18}$</th>
<th>$2^{20}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database* size</td>
<td>3.64</td>
<td>14.52</td>
<td>58.67</td>
</tr>
<tr>
<td>Initial preprocessing (sec)</td>
<td>0.07</td>
<td>0.25</td>
<td>1.03</td>
</tr>
<tr>
<td>Update hints (sec)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Each data item 32 bytes. Run on a machine with 2 GHz processor and 64 GB RAM, single thread.
Evaluation: PIR-Tor Application

- Retrieving relay description files from Tor directory servers
- Why PIR? [PIR-Tor, Sec11]
Evaluation: PIR-Tor Application

- Retrieving relay description files from Tor directory servers
- Why PIR? [PIR-Tor, Sec11]
Evaluation: PIR-Tor Application

• Retrieving relay description files from Tor directory servers
• Why PIR? [PIR-Tor, Sec11]
Evaluation: PIR-Tor Application

• A server could act as the offline server for one client; and the online server for another client

• Updates are propagated to all the servers
Evaluation: PIR-Tor Application

- A server could act as the offline server for one client; and the online server for another client
- Updates are propagated to all the servers
Evaluation: PIR-Tor Application

• A server could act as the offline server for one client; and the online server for another client
• Updates are propagated to all the servers
Evaluation: PIR-Tor Application

- A server could act as the offline server for one client; and the online server for another client
- Updates are propagated to all the servers
Thank you!