Authenticated private information retrieval

USENIX Security Symposium 2023

Simone Colombo  
EPFL

Kirill Nikitin  
NYGC & Columbia University

Henry Corrigan-Gibbs  
MIT

David J. Wu  
UT Austin

Bryan Ford  
EPFL
Private information retrieval (PIR) [CGKS95]

holds index $i \in \{1, \ldots, N\}$

holds database $d \in \mathbb{F}_N^\sqrt{N}$
Private information retrieval (PIR) [CGKS95]

holds index \( i \in \{1, \ldots, N\} \)

learns \( d_i \)

holds database \( d \in \mathbb{F}^N \)
Private information retrieval (PIR) [CGKS95]

holds index $i \in \{1, \ldots, N\}$

learns $d_i$

holds database $d \in \mathbb{F}^N$

learns nothing
Private information retrieval (PIR) \[^{[CGKS95,WYGVZ17]}\]

holds function \( f: \mathbb{F}^N \rightarrow \mathbb{F} \)

holds database \( d \in \mathbb{F}^N \)
Private information retrieval (PIR) \cite{CGKS95,WYGVZ17}

holds function $f : \mathbb{F}^N \rightarrow \mathbb{F}$

learns $f(d)$

holds database $d \in \mathbb{F}^N$
Private information retrieval (PIR) \cite{CGKS95,WYGVZ17}

holds function \( f: \mathbb{F}^N \rightarrow \mathbb{F} \)

holds database \( d \in \mathbb{F}^N \)

learns \( f(d) \)

learns nothing
An example application: PGP key server
An example application: PGP key server
An example application: PGP key server

Bob's public key?

$\text{pk}_{\text{Bob}}$

PGP key server
PIR does not consider integrity

holds index \( i \in \{1, \ldots, N\} \)

holds database \( d \in \mathbb{F}^N \)

learns nothing
PIR does not consider integrity

holds index $i \in \{1, \ldots, N\}$

holds database $d \in \mathbb{F}^N$

learns wrong $d'_i$

learns nothing
PIR does not consider integrity

holds index $i \in \{1, \ldots, N\}$

learns wrong $\text{pk}_{\text{adversary}}$

holds database $d \in \mathbb{F}^N$

learns nothing
PIR and authentication are not enough

holds index $i \in \{1, \ldots, N\}$

holds database $d \in \mathbb{F}^N$
PIR and authentication are not enough

holds index \( i \in \{1, \ldots, N\} \)

if \( \text{Verify}(pk, d_i, \sigma_i) = T \) return \( d_i \)
else abort

holds database \( d \in \mathbb{F}^N \)
PIR and authentication are not enough

holds index $i \in \{1, \ldots, N\}$

if $\text{Verify}(pk, d_i, \sigma_i) = T$ return $d_i$
else abort
PIR and authentication are not enough

holds index $i \in \{1, \ldots, N\}$

if $\text{Verify}(pk, d_i, \sigma_i) = T$ return $d_i$

else abort

holds database $d \in \mathbb{F}^N$
PIR and authentication are not enough

holds index $i \in \{1, \ldots, N\}$

if $\text{Verify}(pk, d_i, \sigma_i) = T$ return $d_i$

else abort

holds database $d \in \mathbb{F}^N$
PIR and authentication are not enough

holds index $i \in \{1, \ldots, N\}$

holds database $d \in \mathbb{F}^N$

if $\text{Verify}(pk, d_i, \sigma_i) = T$ return $d_i$

else abort

The accept/reject bit reveals if the client is reading the $i$th entry: selective-failure attack [KS06].
PIR and authentication are not enough

holds index $i \in \{1,\ldots,N\}$

holds database $d \in \mathbb{F}^N$

A new primitive is necessary: authenticated private information retrieval.

Related works require a majority of honest servers for recovery [BS02,BS07,G07,DGN12,K19,YXB02], stronger assumptions [ZS14] or do not consider selective-failure attacks [KO97,WZ18,ZWH21].

if Verify(pk, $d_i, \sigma_i$) = $\top$ return $d_i$

else abort

The accept/reject bit reveals if the client is reading the $i^{th}$ entry: selective-failure attack [KS06].
Authenticated PIR properties

- Correctness: If client and server are honest, the client recovers $pk_{Bob}$. 
Authenticated PIR properties

• Correctness: If client and server are honest, the client recovers $pk_{Bob}$.

• Privacy: The server(s) learns nothing about the content of the client’s query, even if the server(s) learns whether the client aborted during reconstruction.
Authenticated PIR properties

- Correctness: If client and server are honest, the client recovers $pk_{Bob}$.
- Privacy: The server(s) learns nothing about the content of the client’s query, even if the server(s) learns whether the client aborted during reconstruction.

Selective-failure attacks.
Authenticated PIR properties

- Correctness: If client and server are honest, the client recovers $pk_{Bob}$.
- Privacy: The server(s) learn nothing about the content of the client’s query, even if the server(s) learn whether the client aborted during reconstruction.
- Integrity: The client either outputs the authentic $pk_{Bob}$ or aborts, except with negligible probability.
How to define authentic data?

Multi-server schemes: honest server’s view of the database.
How to define authentic data?

Multi-server schemes: honest server’s view of the database.
How to define authentic data?

Multi-server schemes: honest server’s view of the database.
How to define authentic data?

Multi-server schemes: honest server’s view of the database.

Single-server schemes: digest of the true database.
Our results: multi-server schemes
Our results: multi-server schemes

(1) Multi-servers, single-record query

Given a Merkle-tree scheme, on a database of size $N$
- the per-query communication is $O(\log N)$, same as unauthenticated PIR,
- the integrity error is negligible.

See paper
Our results: multi-server schemes

(1) Multi-servers, single-record query

Given a Merkle-tree scheme, on a database of size $N$
- the per-query communication is $O(\log N)$, same as unauthenticated PIR,
- the integrity error is negligible.

(2) Two-servers, single-record and aggregate queries

Given PRG and a field $\mathbb{F}$, on a database of size $N$
- the per-query communication is $O(\log N)$, same as unauthenticated PIR,
- the integrity error is $1/|\mathbb{F}|$
Our results: single-server schemes
Our results: single-server schemes

(3) Single-record query from LWE

Under the LWE secret dimension $s$ and ciphertext modulus $q$, on a $N$-bit database

- the client downloads a one-time digest of size $n\sqrt{N}$ elements of $\mathbb{Z}_q$,
- the per-query communication cost is $2\sqrt{N}$ elements of $\mathbb{Z}_q$,
- the integrity error is roughly $\sqrt{N}/q$, can be amplified generically.

See paper
Our results: single-server schemes

(3) Single-record query from LWE

Under the LWE secret dimension $s$ and ciphertext modulus $q$, on a $N$-bit database
- the client downloads a one-time digest of size $n\sqrt{N}$ elements of $\mathbb{Z}_q$,
- the per-query communication cost is $2\sqrt{N}$ elements of $\mathbb{Z}_q$,
- the integrity error is roughly $\sqrt{N}/q$, can be amplified generically.

(4) Single-record query from DDH

Under the DDH assumption in a group $\mathbb{G}$, on a $N$-bit database
- the client downloads a one-time digest of size $\sqrt{N}$ elements of $\mathbb{G}$,
- the per-query communication cost is $2\sqrt{N}$ elements of $\mathbb{G}$,
- the integrity error is negligible.
Classic multi-server PIR [CGKS95]

pk_{Bob} is in $d_{22}$, i.e., 2nd column

<table>
<thead>
<tr>
<th></th>
<th>$\sqrt{N}$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{12}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{22}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{32}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{42}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
pk_{Bob} is in $d_{22}$, i.e., 2\textsuperscript{nd} column
Classic multi-server PIR [CGKS95]

pk_{Bob} is in \( d_{22} \), i.e., 2nd column

additive-secret sharing
Classic multi-server PIR [CGKS95]

pk_{Bob} is in $d_{22}$, i.e., $2^{nd}$ column

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\begin{array}{cccc}
\sqrt{N} \\
\sqrt{N} \\
\sqrt{N} \\
\sqrt{N} \\
\end{array}

\begin{array}{cccc}
d_{12} &  &  &  \\
d_{22} &  &  &  \\
d_{32} &  &  &  \\
d_{42} &  &  &  \\
\end{array}

= secret shares
Classic multi-server PIR \[\text{[CGKS95]}\]

pk_{Bob} is in \(d_{22}\), i.e., 2\textsuperscript{nd} column

\[\begin{array}{c}
0 \\
1 \\
0 \\
0
\end{array}\]
Classic multi-server PIR \cite{CGKS95}

pk\textsubscript{Bob} is in \(d_{22}\), i.e., 2\textsuperscript{nd} column

\[pk\textsubscript{Bob} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\]

\textit{additive-secret sharing}

\[N = \begin{bmatrix} d_{12} & 0 \\ d_{22} & 1 \\ d_{32} & 0 \\ d_{42} & 0 \end{bmatrix}\]

\[\sqrt{N} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\]

\[\sqrt{N} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\]

\[\sqrt{N} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\]

\[\sqrt{N} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}\]
pk_{Bob} is in $d_{22}$, i.e., 2nd column
pk_{Bob} is in \( d_{22} \), i.e., 2\(^{nd}\) column

additive-secret sharing

\[ d_{12} \quad d_{22} \quad d_{32} \quad d_{42} \]

= secret shares

\[ \sqrt{N} \]

\[ \begin{array}{c|c|c|c|c}
0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array} \]

[CGKS95]
Classic multi-server PIR [CGKS95]

pk_{Bob} is in $d_{22}$, i.e., 2\textsuperscript{nd} column

$$d_{12} \quad d_{22} \quad d_{32} \quad d_{42}$$

$$+$$

$$d_{12} \quad d_{22} \quad d_{32} \quad d_{42}$$

= secret shares

$$\sqrt{N}$$
**Classic multi-server PIR** [CGKS95]

pk\(_{\text{Bob}}\) is in \(d_{22}\), i.e., 2\(^{\text{nd}}\) column

\[
\begin{array}{c}
\begin{array}{c}
d_{12} \\
d_{22} \\
d_{32} \\
d_{42}
\end{array}
\end{array} + \begin{array}{c}
\begin{array}{c}
d_{12} \\
d_{22} \\
d_{32} \\
d_{42}
\end{array}
\end{array}
= \begin{array}{c}
\begin{array}{c}
d_{12} \\
d_{22} \\
d_{32} \\
d_{42}
\end{array}
\end{array}
\]

 additive-secret sharing

\[\sqrt{N}\]

\[
\begin{array}{c}
\begin{array}{c}
d_{12} \\
d_{22} \\
d_{32} \\
d_{42}
\end{array}
\end{array}
= \text{secret shares}
\]
pk_{Bob} is in $d_{22}$, i.e., 2nd column

additive-secret sharing

correctness
Classic multi-server PIR [CGKS95]

pk_{Bob} is in $d_{22}$, i.e., 2nd column

privacy

additive-secret sharing

correctness
**Classic multi-server PIR** [CGKS95]

pk_{Bob} is in \( d_{22} \), i.e., 2\(^{nd} \) column.

- **Privacy**
- **Additive-secret sharing**
- **Correctness**

communication \( O(\sqrt{N}) \)

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]
Classic multi-server PIR [CGKS95]

Additive-secret sharing

$0$

$1$

$0$

$0$

$0$

$1$

$1$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$1$

$1$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$

$0$
Classic multi-server PIR [CGKS95]

\[ d_{12} + d_{32} + d_{42} = \text{secret shares} \]

\[ \sqrt{N} \]

\[ \sqrt{N} \]

\[ \sqrt{N} \]

\[ \sqrt{N} \]
Classic multi-server PIR [CGKS95]

\[\begin{array}{c}
\text{additive-secret sharing}
\end{array} \]
Key idea: two correlated queries, one for data and one to authenticate
Authenticated multi-server PIR
Authenticated multi-server PIR

samples random $\alpha \in_{R} \mathbb{F}$
Authenticated multi-server PIR

samples random $\alpha \in \mathbb{R} \setminus \mathbb{F}$
Authenticated multi-server PIR

samples random $\alpha \in_R \mathbb{F}$

1  0  1  0

additive-secret sharing

0  0  0  0
0  1  0  1
0  0  0  0
0  0  0  0

$d_{12}$
$d_{22}$
$d_{32}$
$d_{42}$

$\sqrt{N}$

$\sqrt{N}$

$\sqrt{N}$

$\sqrt{N}$

samples random $\alpha \in_R \mathbb{F}$
Authenticated multi-server PIR

samples random $\alpha \in_R \mathbb{F}$

additive-secret sharing

$\mathbb{F}$
Authenticated multi-server PIR

samples random $\alpha \in_R \mathbb{F}$

additive-secret sharing
Authenticated multi-server PIR

samples random $\alpha \in R \mathbb{F}$
Authenticated multi-server PIR

samples random $\alpha \in_R \mathbb{F}$
Authenticated multi-server PIR

samples random $\alpha \in_R \mathbb{F}$
Authenticated multi-server PIR

samples random $\alpha \in_R \mathbb{F}$

$$d_{12} + d_{22} + d_{32} + d_{42} = \alpha d_{12} + \alpha d_{22} + \alpha d_{32} + \alpha d_{42}$$
Authenticated multi-server PIR integrity
Authenticated multi-server PIR integrity

\[
\alpha \cdot \begin{pmatrix}
    d_{12} \\
    d_{22} \\
    d_{32} \\
    d_{42}
\end{pmatrix}
+ \begin{pmatrix}
    d_{12} \\
    d_{22} \\
    d_{32} \\
    d_{42}
\end{pmatrix}
= \begin{pmatrix}
    ad_{12} \\
    ad_{22} \\
    ad_{32} \\
    ad_{42}
\end{pmatrix}
+ \begin{pmatrix}
    ad_{12} \\
    ad_{22} \\
    ad_{32} \\
    ad_{42}
\end{pmatrix}
\]
Authenticated multi-server PIR integrity

\[ \alpha \cdot \left( \begin{array}{cccc} d_{12} & d_{12} & \alpha d_{12} & \alpha d_{12} \\ d_{22} & d_{22} & \alpha d_{22} & \alpha d_{22} \\ d_{32} & d_{32} & \alpha d_{32} & \alpha d_{32} \\ d_{42} & d_{42} & \alpha d_{42} & \alpha d_{42} \end{array} \right) = \left( \begin{array}{c} d_{12} \\ d_{22} \\ d_{32} \\ d_{42} \end{array} \right) + \left( \begin{array}{c} \alpha d_{12} \\ \alpha d_{22} \\ \alpha d_{32} \\ \alpha d_{42} \end{array} \right) \]

return second element of

\[ d_{12} + d_{22} + d_{32} + d_{42} \]
Authenticated multi-server PIR integrity

if $\alpha \cdot \begin{pmatrix} d_{12} \\ d_{22} \\ d_{32} \\ d_{42} \end{pmatrix} + \begin{pmatrix} d_{12} \\ d_{22} \\ d_{32} \\ d_{42} \end{pmatrix} = \begin{pmatrix} ad_{12} \\ ad_{22} \\ ad_{33} \\ ad_{44} \end{pmatrix} + \begin{pmatrix} \alpha d_{12} \\ \alpha d_{22} \\ \alpha d_{32} \\ \alpha d_{42} \end{pmatrix}$

return second element of

else abort
Authenticated multi-server PIR integrity

\[
\begin{align*}
\text{if } & \alpha \cdot \begin{pmatrix}
d_{12} \\ d_{22} \\ d_{32} \\ d_{42}
\end{pmatrix} + \begin{pmatrix}
d_{12} \\ d_{22} \\ d_{32} \\ d_{42}
\end{pmatrix} = \begin{pmatrix}
ad_{12} \\ ad_{22} \\ ad_{32} \\ ad_{42}
\end{pmatrix} + \\
& \text{return second element of } \begin{pmatrix}
d_{12} \\ d_{22} \\ d_{32} \\ d_{42}
\end{pmatrix} + \begin{pmatrix}
d_{12} \\ d_{22} \\ d_{32} \\ d_{42}
\end{pmatrix} = \begin{pmatrix}
d_{12} \\ d_{22} \\ d_{32} \\ d_{42}
\end{pmatrix} \\
\text{else abort}
\end{align*}
\]

communication $O(\sqrt{N})$, see paper for $O(\log N)$ with function secret sharing [BGI16]
Evaluation: single-record queries

Cost of retrieving a 1KiB record
Evaluation: aggregate queries

SELECT COUNT(*) FROM keys WHERE email LIKE "%s"

User-time ratio

Bandwidth ratio

SELECT COUNT(*) FROM keys WHERE email LIKE "%s"
Evaluation: aggregate queries

SELECT COUNT(*) FROM keys WHERE email LIKE "%s"

ratio of authenticated and classic unauthenticated PIR

User-time ratio

Bandwidth ratio

SELECT COUNT(*) FROM keys WHERE email LIKE "%s"
Evaluation: aggregate queries

SELECT COUNT(*) FROM keys WHERE email LIKE "%s"

User-time ratio

Bandwidth ratio

Ratio of authenticated and classic unauthenticated PIR

Count emails that end with string “s”

SELECT COUNT(*) FROM keys WHERE email LIKE "%s"
DEMO TIME
Conclusion

• New integrity definition for PIR schemes: either authentic record or abort.
  • In multi-server setting comes almost for free.
  • In single-server setting imposes 30-100× overhead: can we do better?
• Key directory service: PoC, but not deployed yet.
• Keyd: https://keyd.org/.