Information-Theoretic Multi-Server Client Preprocessing PIR

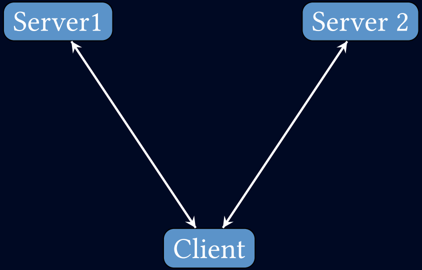
Jaspal Singh
Purdue University
Georgia Tech

Yu Wei
Georgia Tech

Vassilis Zikas
Georgia Tech

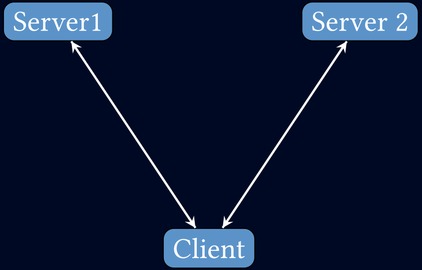
TCC 2024

Private Information Retrieval



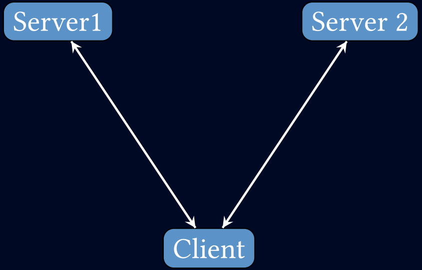
- interactive protocol between client and server(s)

Private Information Retrieval



- interactive protocol between client and server(s)
- server(s) input database D of size n
- client inputs index $i \in \{0, 1, \ldots, n-1\}$

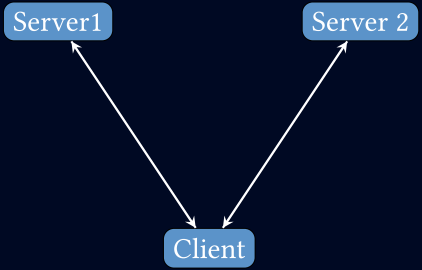
Private Information Retrieval



- interactive protocol between client and server(s)
- server(s) input database D of size n
- client inputs index $i \in \{0, 1, \ldots, n-1\}$

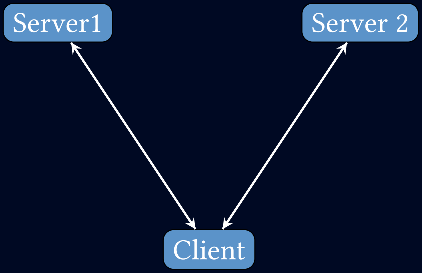
- Correctness: Client outputs $D[i]$
- Privacy: Server learns nothing about i

Private Information Retrieval



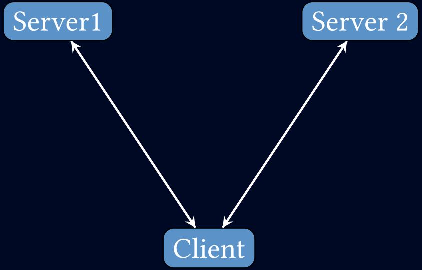
- Traditional goal of PIR - reduce the communication complexity

Private Information Retrieval



- Traditional goal of PIR - reduce the communication complexity
- [CKGS98] show an $\Omega(n)$ computation lower bound for server complexity

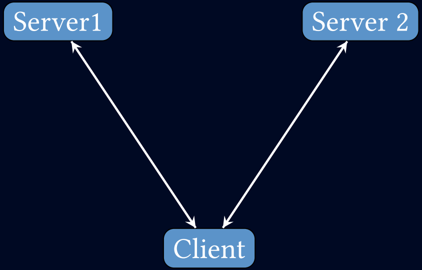
Private Information Retrieval



- Traditional goal of PIR - reduce the communication complexity
- [CKGS98] show an $\Omega(n)$ computation lower bound for server complexity

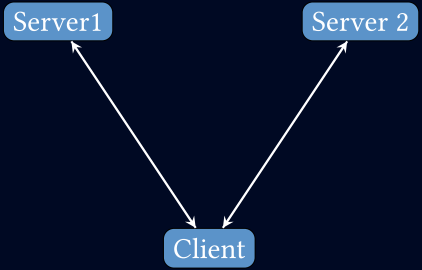
Need for different PIR models to overcome this lower bound!

PIR with Client Preprocessing



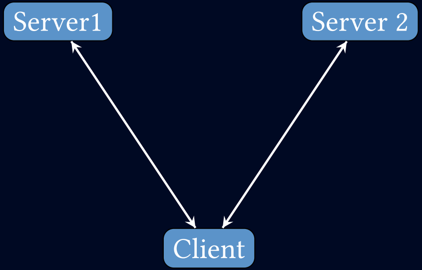
- Protocol has two phases: offline and online

PIR with Client Preprocessing



- Protocol has two phases: offline and online
- expensive offline phase with linear computation overhead
- Client stores a local state

PIR with Client Preprocessing



- Protocol has two phases: offline and online
- expensive offline phase with linear computation overhead
- Client stores a local state
- Sub-linear online phase

Previous Works (PIR with Client Preprocessing)

constructions (# servers)	communication	computation	client storage	assumptions
PRP-PIR [CGK20] (2)	$\widetilde{O}(n^{1/2})$[1]	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	OWF

[1] $\widetilde{O}$ notation hides poly log factors

Previous Works (PIR with Client Preprocessing)

constructions (# servers)	communication	computation	client storage	assumptions
PRP-PIR [CGK20] (2)	$\widetilde{O}(n^{1/2})$[1]	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	OWF
Shi et al. [SACM21] (2)	$(\mathrm{poly}(\log n))$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	LWE
TreePIR [LP23] (2)	$(\mathrm{poly}(\log n))$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	DDH
Ghoshal et al. [GZS24] (2)	$\widetilde{O}(n^{1/4})$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	OWF

[1] $\widetilde{O}$ notation hides poly log factors

Previous Works (PIR with Client Preprocessing)

constructions (# servers)	communication	computation	client storage	assumptions
PRP-PIR [CGK20] (2)	$\widetilde{O}(n^{1/2})$[1]	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	OWF
Shi et al. [SACM21] (2)	$(\mathrm{poly}(\log n))$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	LWE
TreePIR [LP23] (2)	$(\mathrm{poly}(\log n))$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	DDH
Ghoshal et al. [GZS24] (2)	$\widetilde{O}(n^{1/4})$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	OWF

Does there exist non-trivial protocols for unconditional multi-server PIR with client preprocessing?

[1] $\widetilde{O}$ notation hides poly log factors

Previous Works (PIR with Client Preprocessing)

constructions (# servers)	communication	computation	client storage	assumptions
PRP-PIR [CGK20] (2)	$\widetilde{O}(n^{1/2})$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	OWF
Shi et al. [SACM21] (2)	$(\mathrm{poly}(\log n))$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	LWE
TreePIR [LP23] (2)	$(\mathrm{poly}(\log n))$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	DDH
Ghoshal et al. [GZS24] (2)	$\widetilde{O}(n^{1/4})$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(\sqrt{n})$	OWF
Our Work $(2t)$	$\widetilde{O}(n^{1/2})$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(n^{1/2+1/2t})$	–
Set $t = 1/2 \log n$ $(\log n)$	$\widetilde{O}(n^{1/2})$	$\widetilde{O}(\sqrt{n})$	$\widetilde{O}(n^{1/2})$	–

CGK Paradigm: Privately Puncturable Pseudo-random Sets

CGK Paradigm: Privately Puncturable Pseudo-random Sets

PPPS (Gen, Set, Test, Punc)

- $k \leftarrow \mathrm{Gen}()$
- $S \leftarrow \mathrm{Set}(k)$
- $b \leftarrow \mathrm{Test}(k, x)$
- $\hat{k} \leftarrow \mathrm{Punc}(k, x)$

CGK Paradigm: Privately Puncturable Pseudo-random Sets

PPPS (Gen, Set, Test, Punc)

- $k \leftarrow \mathsf{Gen}()$
- $S \leftarrow \mathsf{Set}(k)$
- $b \leftarrow \mathsf{Test}(k, x)$
- $\hat{k} \leftarrow \mathsf{Punc}(k, x)$

Security properties:

- $S \cong$ random subset of size $\sqrt{n}$

CGK Paradigm: Privately Puncturable Pseudo-random Sets

PPPS (Gen, Set, Test, Punc)

- $k \leftarrow \mathsf{Gen}()$
- $S \leftarrow \mathsf{Set}(k)$
- $b \leftarrow \mathsf{Test}(k, x)$
- $\hat{k} \leftarrow \mathsf{Punc}(k, x)$

Security properties:

- $S \cong$ random subset of size $\sqrt{n}$
- $\mathsf{Test}(k, x) = [x \stackrel{?}{\in} \mathsf{Set}(k)]$

CGK Paradigm: Privately Puncturable Pseudo-random Sets

PPPS (Gen, Set, Test, Punc)

- $k \leftarrow \mathsf{Gen}()$
- $S \leftarrow \mathsf{Set}(k)$
- $b \leftarrow \mathsf{Test}(k, x)$
- $\hat{k} \leftarrow \mathsf{Punc}(k, x)$

Security properties:

- $S \cong$ random subset of size $\sqrt{n}$
- $\mathsf{Test}(k, x) = [x \overset{?}{\in} \mathsf{Set}(k)]$
- $\mathsf{Set}(\mathsf{Punc}(k, x)) = \mathsf{Set}(k) \setminus \{x\}$

CGK Paradigm: Privately Puncturable Pseudo-random Sets

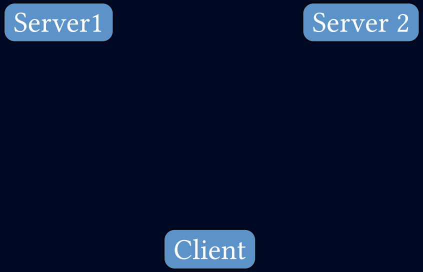
PPPS (Gen, Set, Test, Punc)

- $k \leftarrow \mathsf{Gen}()$
- $S \leftarrow \mathsf{Set}(k)$
- $b \leftarrow \mathsf{Test}(k, x)$
- $\hat{k} \leftarrow \mathsf{Punc}(k, x)$

Security properties:

- $S \cong$ random subset of size $\sqrt{n}$
- $\mathsf{Test}(k, x) = [x \overset{?}{\in} \mathsf{Set}(k)]$
- $\mathsf{Set}(\mathsf{Punc}(k, x)) = \mathsf{Set}(k) \setminus \{x\}$
- Punctured key $\hat{k}$ hides x

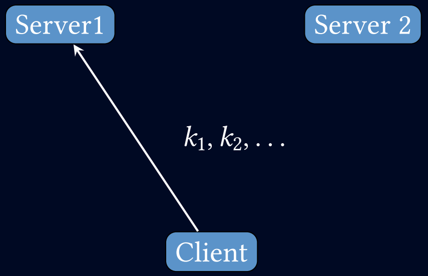
PIR with client preprocessing [CGK20]



PIR with client preprocessing [CGK20]

Offline:

- Client sends $\widetilde{O}(\sqrt{n})$ PPPS keys $k_1, k_2, \ldots$ to server 1

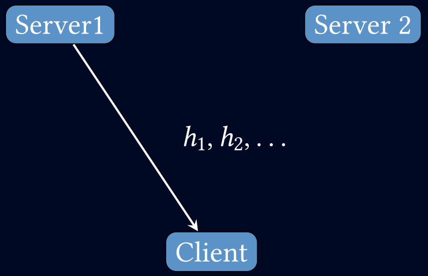


PIR with client preprocessing [CGK20]

Offline:

- Client sends $\widetilde{O}(\sqrt{n})$ PPPS keys $k_1, k_2, \ldots$ to server 1
- server responds with hint bits $h_1, h_2, \ldots$

$$S_i \leftarrow \mathrm{Set}(k_i)$$
$$h_t = \oplus_{j \in S_t} D[j]$$



PIR with client preprocessing [CGK20]

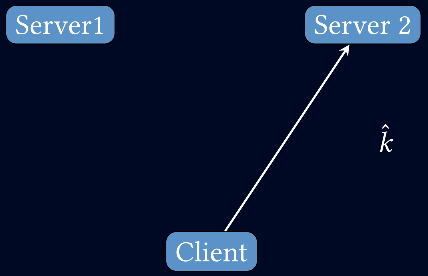
Offline:

- Client sends $\widetilde{O}(\sqrt{n})$ PPPS keys $k_1, k_2, \ldots$ to server 1
- server responds with hint bits $h_1, h_2, \ldots$

$$S_i \leftarrow \mathsf{Set}(k_i)$$
$$h_t = \oplus_{j \in S_t} D[j]$$

Online: (Client inputs x)

- Client finds $k = k_i$ with $\mathsf{Test}(k, x) =$true and send $\hat{k} = \mathsf{Punc}(k, x)$ to server 2



PIR with client preprocessing [CGK20]

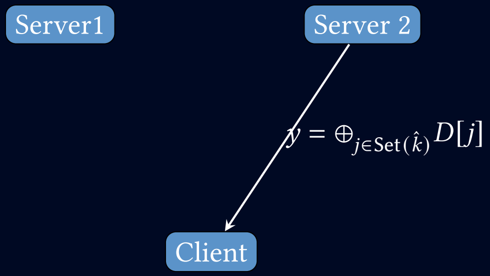
Offline:

- Client sends $\widetilde{O}(\sqrt{n})$ PPPS keys $k_1, k_2, \ldots$ to server 1
- server responds with hint bits $h_1, h_2, \ldots$

$$S_i \leftarrow \mathsf{Set}(k_i)$$
$$h_t = \oplus_{j \in S_t} D[j]$$

Online: (Client inputs x)

- Client finds $k = k_i$ with $\mathsf{Test}(k, x)$ =true and send $\hat{k} = \mathsf{Punc}(k, x)$ to server 2
- server 2 responds with $y \leftarrow \oplus_{j \in S} D[j]$ where $S = \mathsf{Set}(\hat{k})$



PIR with client preprocessing [CGK20]

Server1

Server 2

Client

Offline:

- Client sends $\widetilde{O}(\sqrt{n})$ PPPS keys $k_1, k_2, \ldots$ to server 1
- server responds with hint bits $h_1, h_2, \ldots$

$$S_i \leftarrow \mathsf{Set}(k_i)$$
$$h_t = \oplus_{j \in S_t} D[j]$$

Online: (Client inputs x)

- Client finds $k = k_i$ with $\mathsf{Test}(k, x) = \mathsf{true}$ and send $\hat{k} = \mathsf{Punc}(k, x)$ to server 2
- server 2 responds with $y \leftarrow \oplus_{j \in S} D[j]$ where $S = \mathsf{Set}(\hat{k})$
- Client outputs $y \oplus h_i = D[x]$

PIR with client preprocessing [CGK20]

Offline:

- Client sends $\widetilde{O}(\sqrt{n})$ PPPS keys $k_1, k_2, \ldots$ to server 1
- server responds with hint bits $h_1, h_2, \ldots$

$$S_i \leftarrow \mathsf{Set}(k_i)$$
$$h_t = \oplus_{j \in S_t} D[j]$$

Online: (Client inputs x)

- Client finds $k = k_i$ with $\mathsf{Test}(k, x) = \mathsf{true}$ and send $\hat{k} = \mathsf{Punc}(k, x)$ to server 2
- server 2 responds with $y \leftarrow \oplus_{j \in S} D[j]$ where $S = \mathsf{Set}(\hat{k})$
- Client outputs $y \oplus h_i = D[x]$

Server1

Server 2

Client

Online complexity:

- Client computation: $\widetilde{O}(\sqrt{n})$
- Server computation: $\widetilde{O}(\sqrt{n})$
- bandwidth: $\widetilde{O}(\sqrt{n})$

Key Challenge

- Design PPPS from information-theoretic primitives
- Short PPPS key representation that supports efficient test, puncturing and set enumeration

Key Challenge

- Design PPPS from information-theoretic primitives
- Short PPPS key representation that supports efficient test, puncturing and set enumeration

Key insights:

- For PIR correctness, only need each random set to contain indivdual elements from domain with probability $1/\sqrt{n}$
- Multi-server PIR scheme is compatible with a weaker PPPS: puncturing a PPPS key gives multiple disjoint keys/sets

Privately Multi-Puncturable Random Sets (PMPRS)

Privately Multi-Puncturable Random Sets (PMPRS)

(n, t)-**PMPRS** (Gen, Set, Test, Punc
, SerEval)

- $k \leftarrow \mathsf{Gen}()$
- $S \leftarrow \mathsf{Set}(k)$
- $b \leftarrow \mathsf{Test}(k, x)$
- $((S_0, k_0, \mathsf{ind}_0), , \ldots,$
  $(S_{t-1}, k_{t-1}, \mathsf{ind}_{t-1})) \leftarrow \mathsf{Punc}(k, x)$
- $\vec{v}_i \leftarrow \mathsf{SerEval}(i, k_i, DB)$

Privately Multi-Puncturable Random Sets (PMPRS)

(n,t)**-PMPRS** (Gen, Set, Test, Punc , SerEval)

- $k \leftarrow \mathsf{Gen}()$
- $S \leftarrow \mathsf{Set}(k)$
- $b \leftarrow \mathsf{Test}(k, x)$
- $((S_0, k_0, \mathsf{ind}_0), \ldots, (S_{t-1}, k_{t-1}, \mathsf{ind}_{t-1})) \leftarrow \mathsf{Punc}(k, x)$
- $\vec{v}_i \leftarrow \mathsf{SerEval}(i, k_i, DB)$

Security properties:

- **randomness** For any $x \in [n]$, $Pr(x \in S) = 1/\sqrt{n}$
- **Privacy**: key k_i hides x
- **Correctness**:
  - $S \setminus \{x\} = \dot{\bigcup} S_i$
  - for $i \in [t]$, $\vec{v}_i[\mathsf{ind}_i] = \oplus_{j \in S_i} DB[j]$

Privately Multi-Puncturable Random Sets (PMPRS)

(n, t)**-PMPRS** (Gen, Set, Test, Punc , SerEval)

- $k \leftarrow \mathsf{Gen}()$
- $S \leftarrow \mathsf{Set}(k)$
- $b \leftarrow \mathsf{Test}(k, x)$
- $((S_0, k_0, \mathsf{ind}_0), \ldots, (S_{t-1}, k_{t-1}, \mathsf{ind}_{t-1})) \leftarrow \mathsf{Punc}(k, x)$
- $\vec{v}_i \leftarrow \mathsf{SerEval}(i, k_i, DB)$

Security properties:

- **randomness** For any $x \in [n]$, $Pr(x \in S) = 1/\sqrt{n}$
- **Privacy**: key k_i hides x
- **Correctness**:
  - $S \setminus \{x\} = \dot{\bigcup} S_i$
  - for $i \in [t]$, $\vec{v}_i[\mathsf{ind}_i] = \oplus_{j \in S_i} DB[j]$

Efficient PMPRS: $\widetilde{O}(1)$ Test and $\widetilde{O}(\sqrt{n})$ SerEval complexity; $\widetilde{O}(n^{1/2t})$ key-size

Main Results

Theorem (PMPRS)

There exist (n, t)-PMPRS with key size $\widetilde{O}(n^{1/2t})$, Test and $\mathsf{SerEval}$ complexity $\widetilde{O}(1)$ and $\widetilde{O}(\sqrt{n})$ respectively

Theorem (IT client preprocessing PIR)

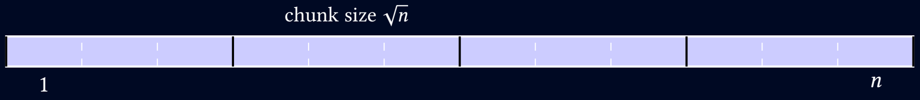
(n, t)-PMPRS $\implies$ t server IT PIR with preprocessing with client state size $\widetilde{O}(n^{1/2+1/2t})$, online client/server $\widetilde{O}(\sqrt{n})$ and bandwidth $\widetilde{O}(\sqrt{n})$

$(n, 1/2 \log n)$-**PMPRS** (where n is even power of 2)

$(n, 1/2 \log n)$-PMPRS (where n is even power of 2)

well-partitioned random set

- Contain one random element from each chunk
- chunk $(x) = \lfloor x/\sqrt{n} \rfloor$, offset$(x) = (x \% \sqrt{n})$



$(n, 1/2 \log n)$-**PMPRS** (where n is even power of 2)

Let $t = 1/2 \log_2 n$ and $m = \sqrt{n}$

$(n, 1/2 \log n)$-**PMPRS** (where n is even power of 2)

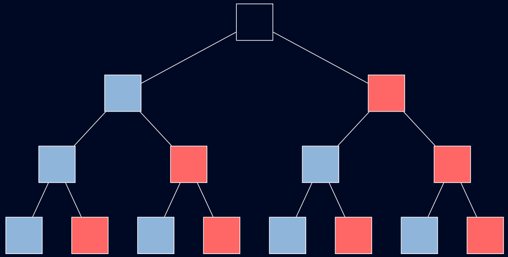
Let $t = 1/2 \log_2 n$ and $m = \sqrt{n}$

- $R \leftarrow \text{Gen}()$ outputs for $i \in [t]$:
  R[i][0] , R[i][1] $\in [m]$

$(n, 1/2 \log n)$-**PMPRS** (where n is even power of 2)

Let $t = 1/2 \log_2 n$ and $m = \sqrt{n}$

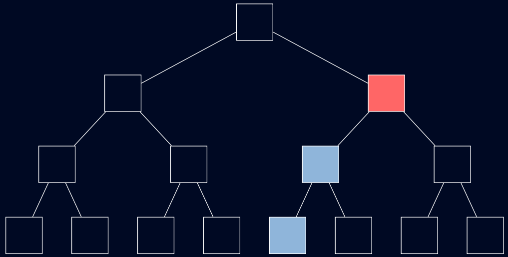
- $R \leftarrow \text{Gen}()$ outputs for $i \in [t]$:
  R[i][0] , R[i][1] $\in [m]$



$(n, 1/2 \log n)$-**PMPRS** (where n is even power of 2)

Let $t = 1/2 \log_2 n$ and $m = \sqrt{n}$

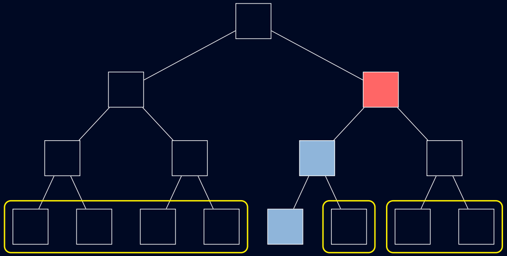
- $R \leftarrow \mathrm{Gen}()$ outputs for $i \in [t]$: R[i][0], R[i][1] $\in [m]$
- $\mathrm{Test}(R, x)$:
  - $(c_0, \ldots, c_{t-1}) \leftarrow$ bit-dec(chunk (x))
  - check if $\oplus_i \boxed{R[i][c_i]} = \mathrm{offset}(x)$?



$(n, 1/2 \log n)$-**PMPRS** (where n is even power of 2)

Let $t = 1/2 \log_2 n$ and $m = \sqrt{n}$

- $R \leftarrow \mathrm{Gen}()$ outputs for $i \in [t]$:
  $\boxed{\mathrm{R[i][0]}}$, $\boxed{\mathrm{R[i][1]}}$ $\in [m]$
- $\mathrm{Test}(R, x)$:
  - $(c_0, \ldots, c_{t-1}) \leftarrow$ bit-dec(chunk (x))
  - check if $\oplus_i$ $\boxed{R[i][c_i]}$ $= \mathrm{offset}(x)$?
- $\mathrm{Set}(k)$
- $\mathrm{Punc}(k, x)$
  - Each key k_i contains offset values of set S_i
- $\mathrm{SerEval}(D, k_i)$
  - for each guess of chunk indexes of S_i compute database xor bit



Conclusion

Conclusion

- New PMPRS primitive and construction
- Efficient (n, t)-PMPRS $\implies$ non-trivial unconditional PIR with client preprocessing ($2t$ servers)
  - General construction based on arbitrary d-ary trees
  - technical challenge: ensuring SerEval cost is $O(\sqrt{n})$
- **Extension**: Improved online communication complexity $O(n^{o(1)+1/2t})$ assuming $4t$ servers

Conclusion

- New PMPRS primitive and construction
- Efficient (n, t)-PMPRS $\implies$ non-trivial unconditional PIR with client preprocessing ($2t$ servers)
  - General construction based on arbitrary d-ary trees
  - technical challenge: ensuring SerEval cost is $O(\sqrt{n})$
- **Extension**: Improved online communication complexity $O(n^{o(1)+1/2t})$ assuming $4t$ servers

Parallel Work [ISW24]

- Also study PIR with client preprocessing in IT setting (1 and 2 server setting)
- different techniques and lower bound proof
- Higher complexity for online server cost, offline bandwidth, and other measures

Thanks

eprint: `https://eprint.iacr.org/2024/780.pdf`