Recent Advances in Private Information Retrieval

David Wu

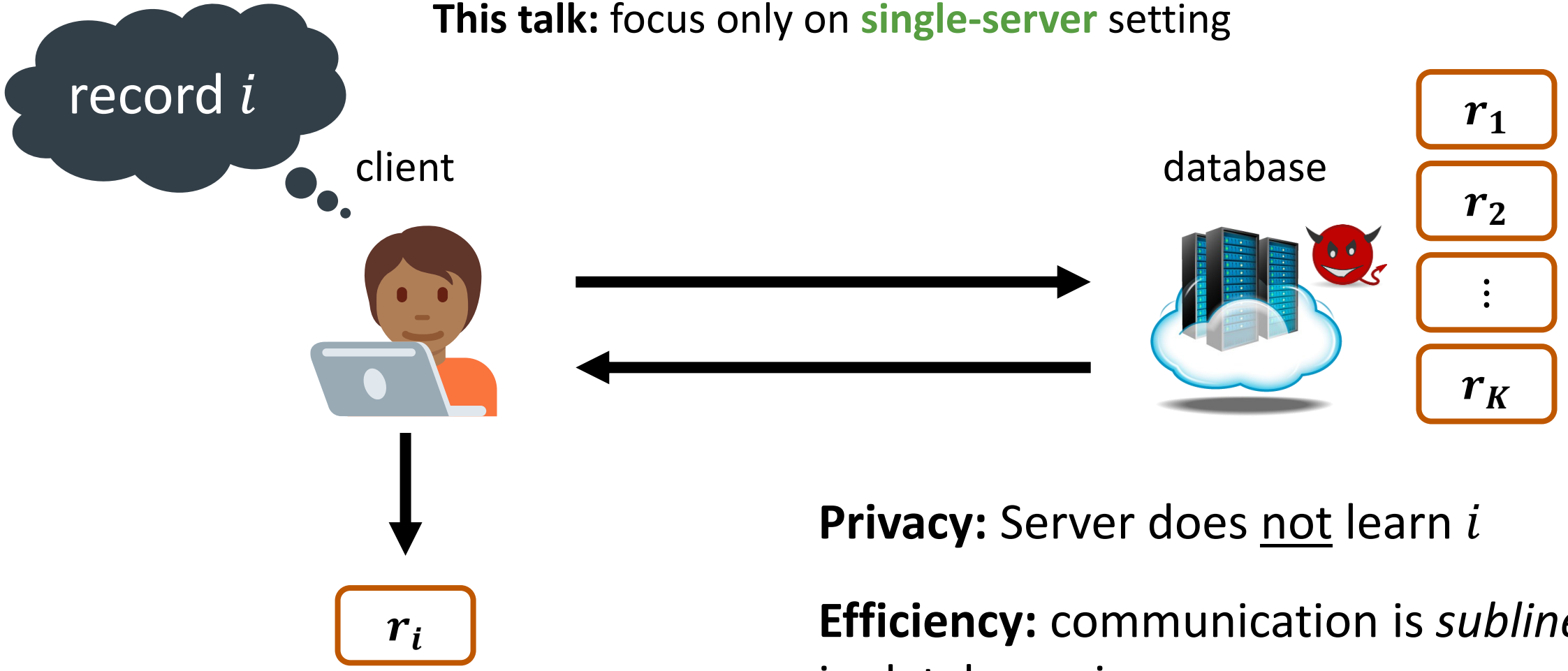
June 2026

based on joint works with Samir Menon and Sidaarth Sabhnani

Private Information Retrieval (PIR)

[CGKS95]

This talk: focus only on **single-server** setting



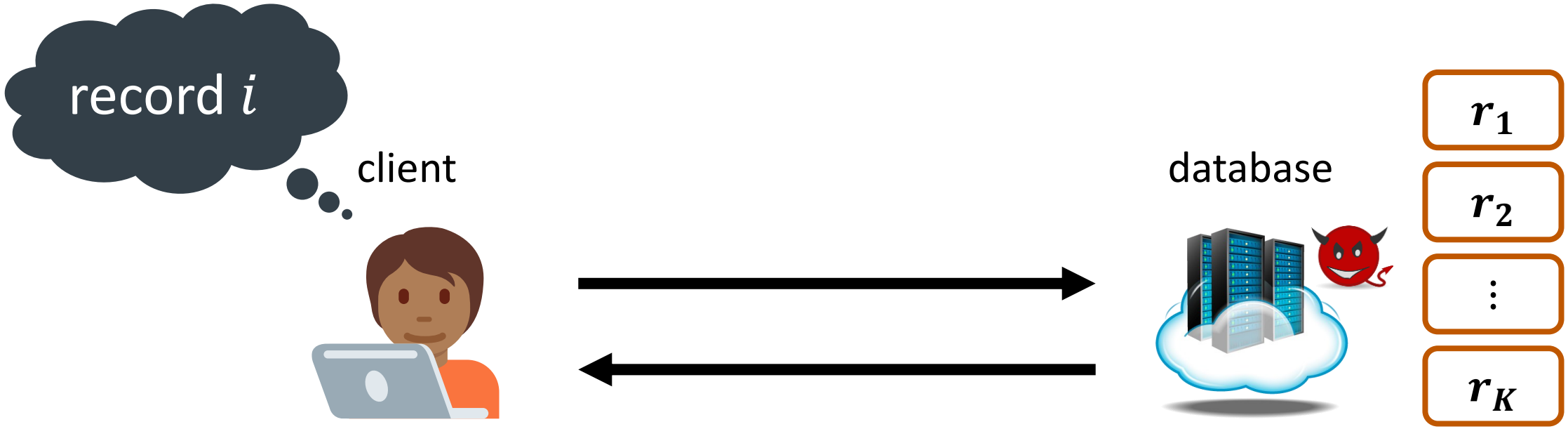
Privacy: Server does not learn i

Efficiency: communication is *sublinear* in database size

Correctness: Client learns r_i

Private Information Retrieval (PIR)

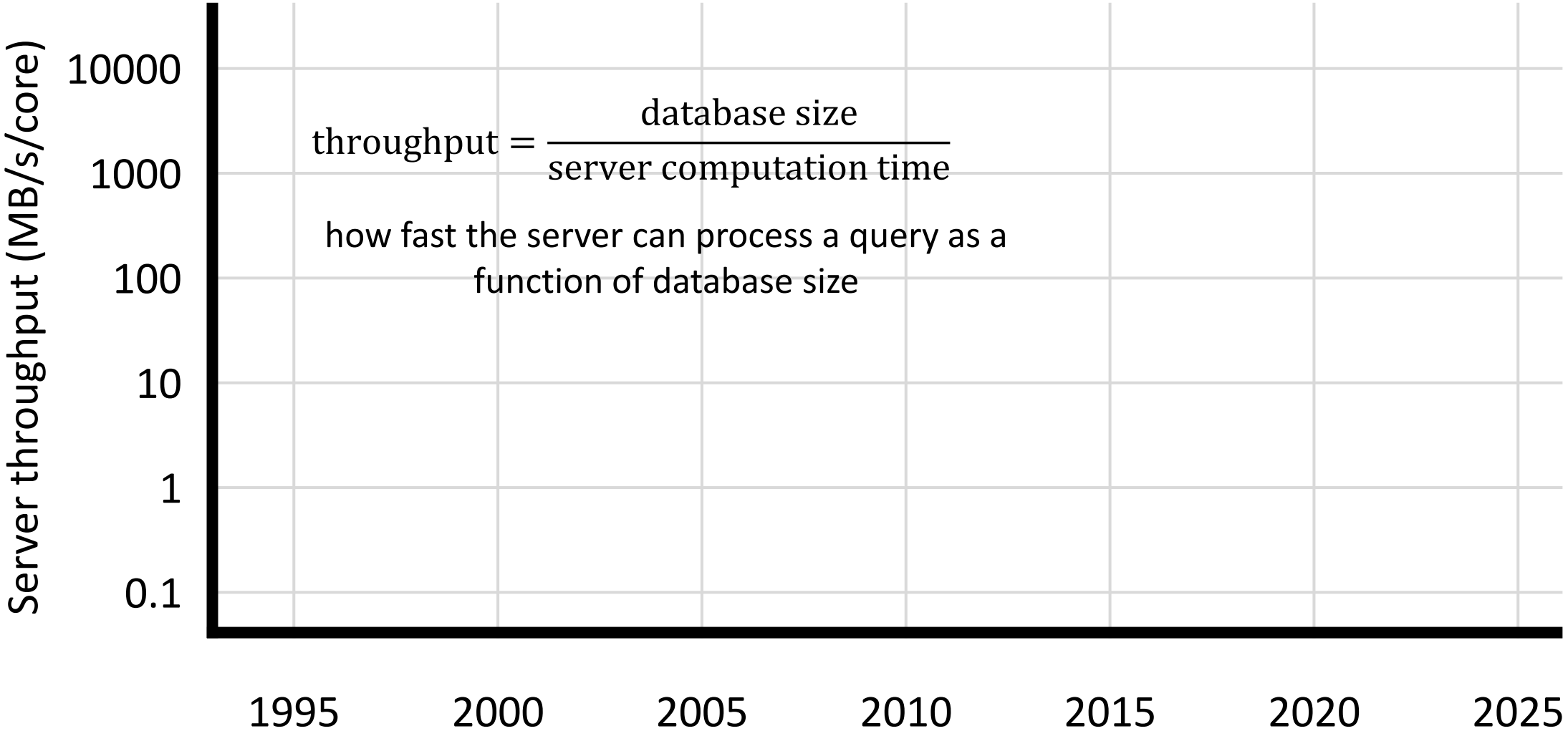
[CGKS95]



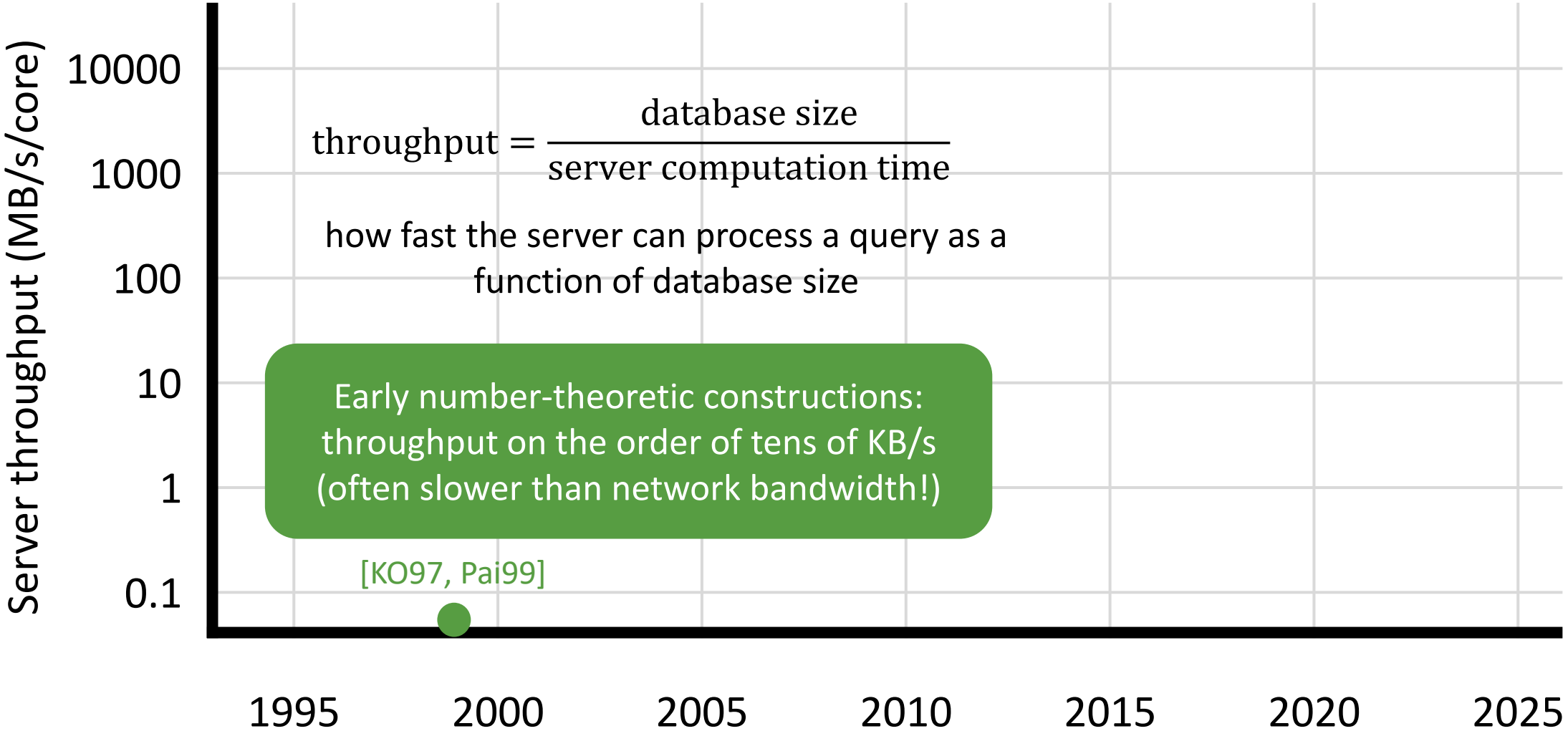
Basic building block in many privacy-preserving protocols

- Metadata-private messaging
- Certificate transparency auditing
- Private content delivery
- Contact discovery
- Private web search
- Private navigation
- Private contact tracing
- Private DNS
- Private blocklist lookup

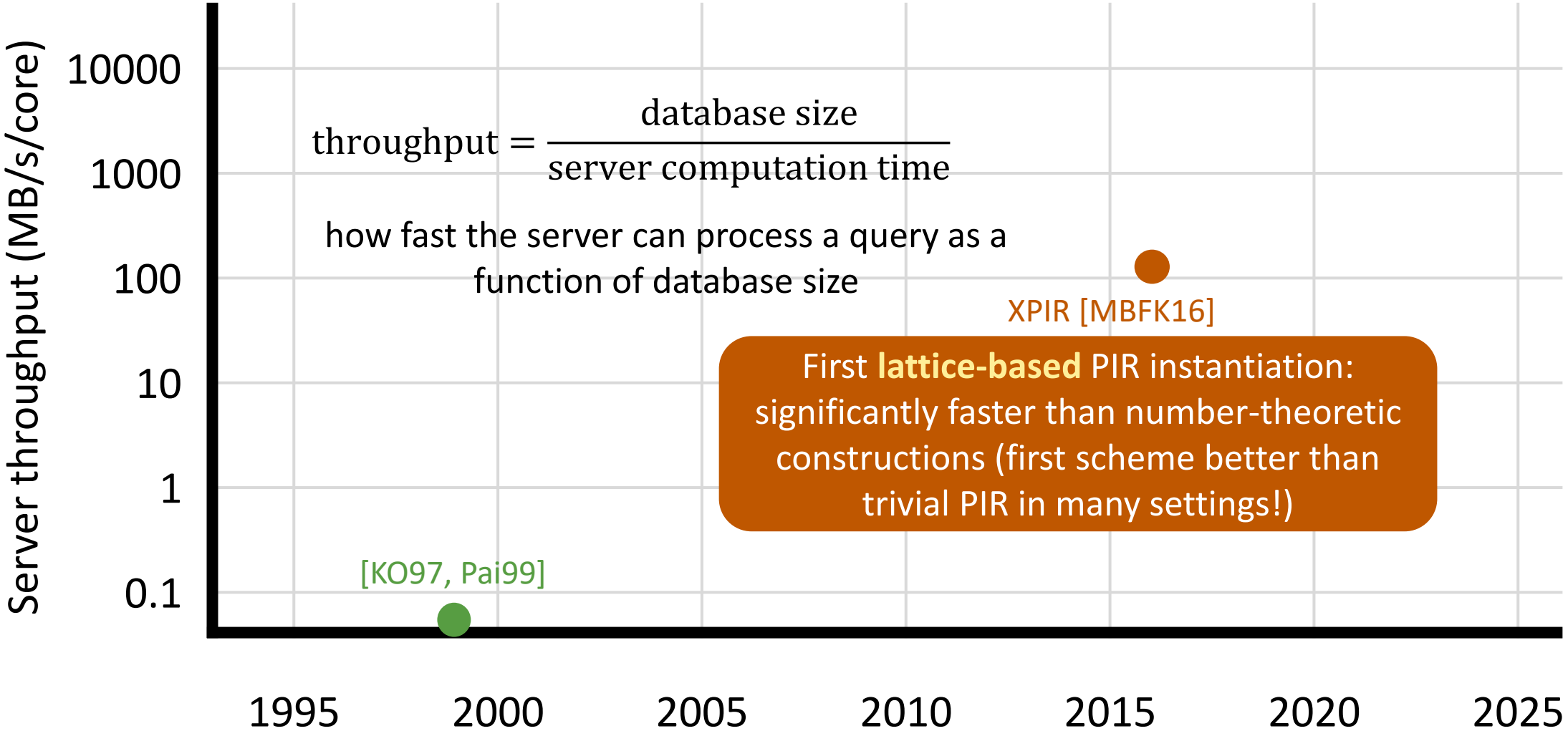
25 Years of PIR Research



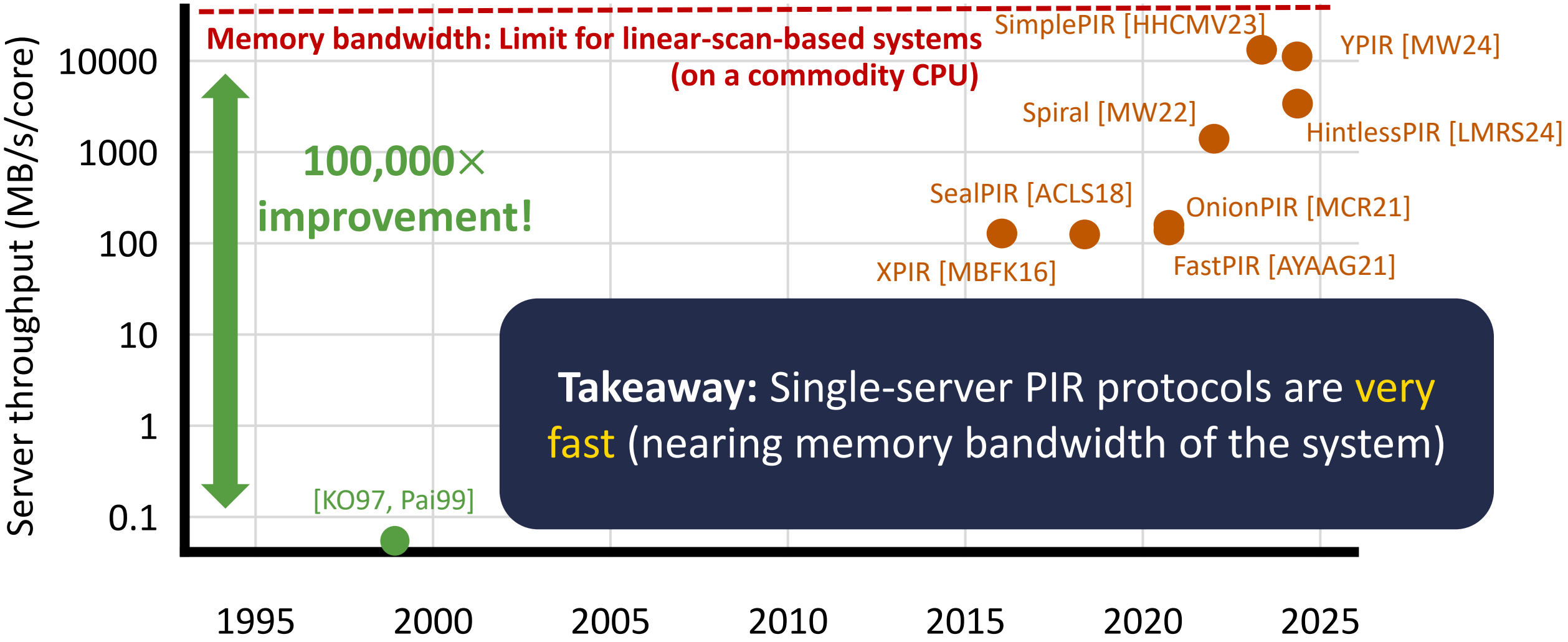
25 Years of PIR Research



25 Years of PIR Research



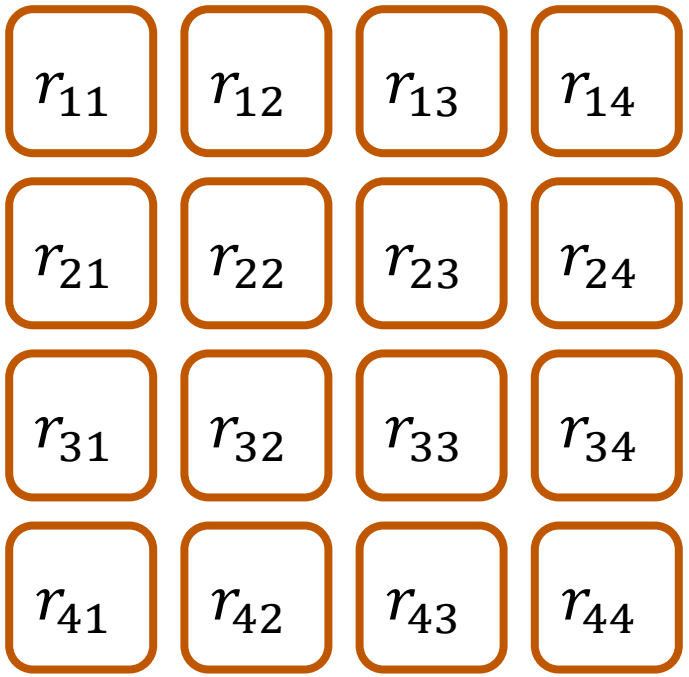
25 Years of PIR Research



PIR from Homomorphic Encryption

[KO97]

Starting point: a $\sqrt{K}$ construction (K = number of records)

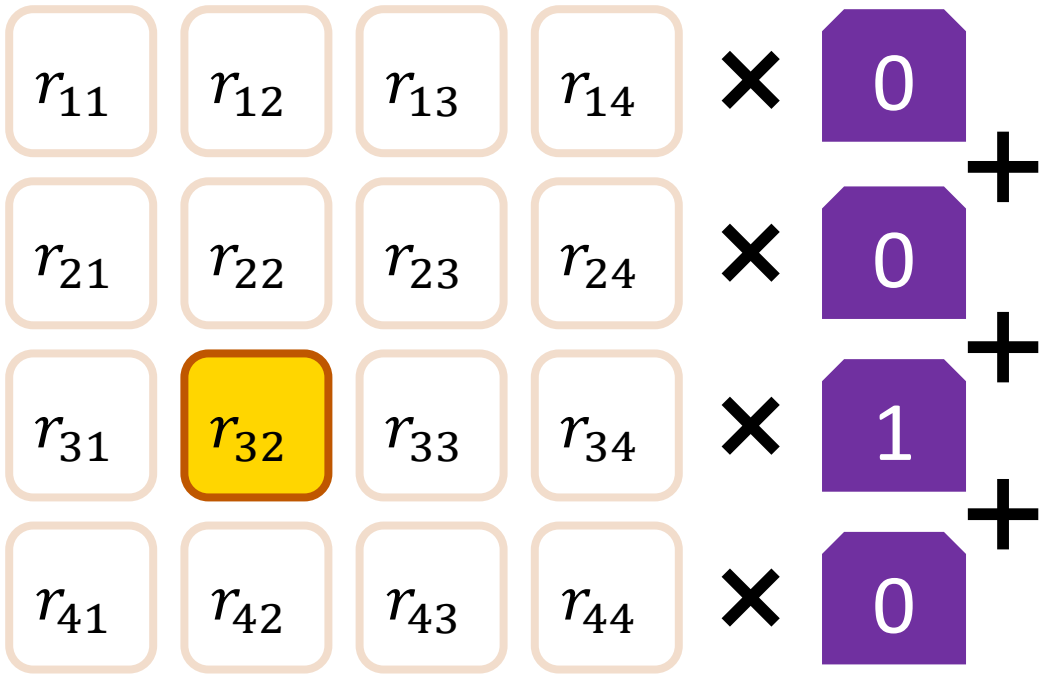


Arrange the database as a $\sqrt{K}$-by-$\sqrt{K}$ matrix

PIR from Homomorphic Encryption

[KO97]

Starting point: a $\sqrt{K}$ construction (K = number of records)



Encrypt a 0/1 vector indicating the row containing the desired record

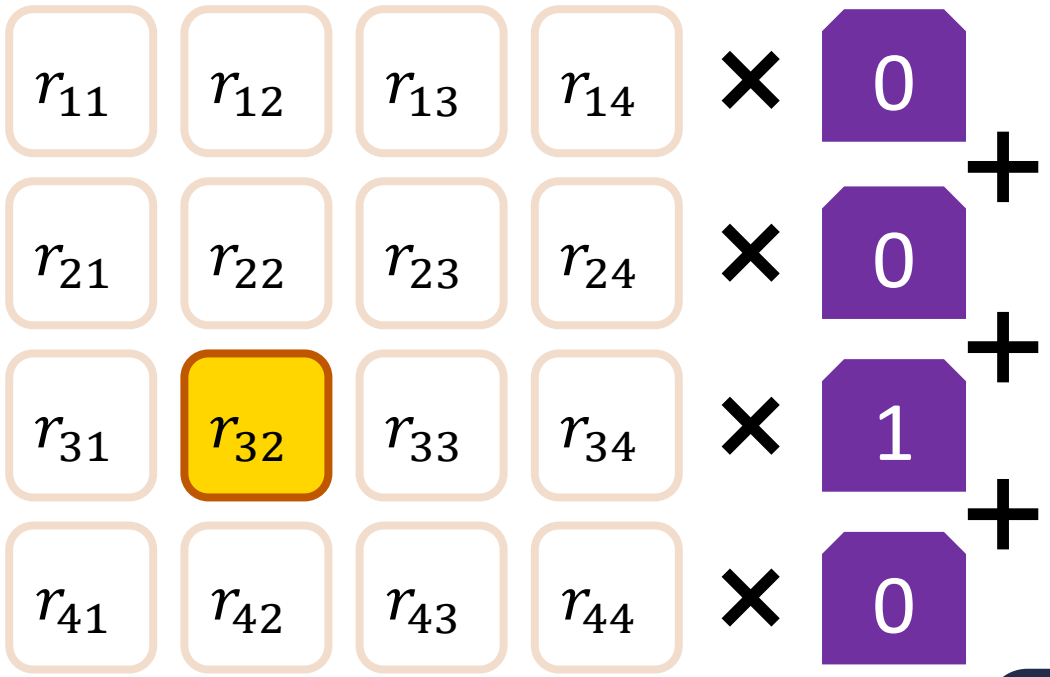
Arrange the database as a $\sqrt{K}$-by-$\sqrt{K}$ matrix

Homomorphically compute product between query vector and database matrix

PIR from Homomorphic Encryption

[KO97]

Starting point: a $\sqrt{K}$ construction (K = number of records)



Encrypt a 0/1 vector indicating the row containing the desired record

Arrange the database as a $\sqrt{K}$-by-$\sqrt{K}$ matrix

Database is in the clear, so *additive* homomorphism suffices

PIR from Homomorphic Encryption

[KO97]

Starting point: a $\sqrt{K}$ construction (K = number of records)

Client decrypts to learn records

Encrypt a 0/1 vector indicating the row containing the desired record

r_{31} r_{32} r_{33} r_{34}

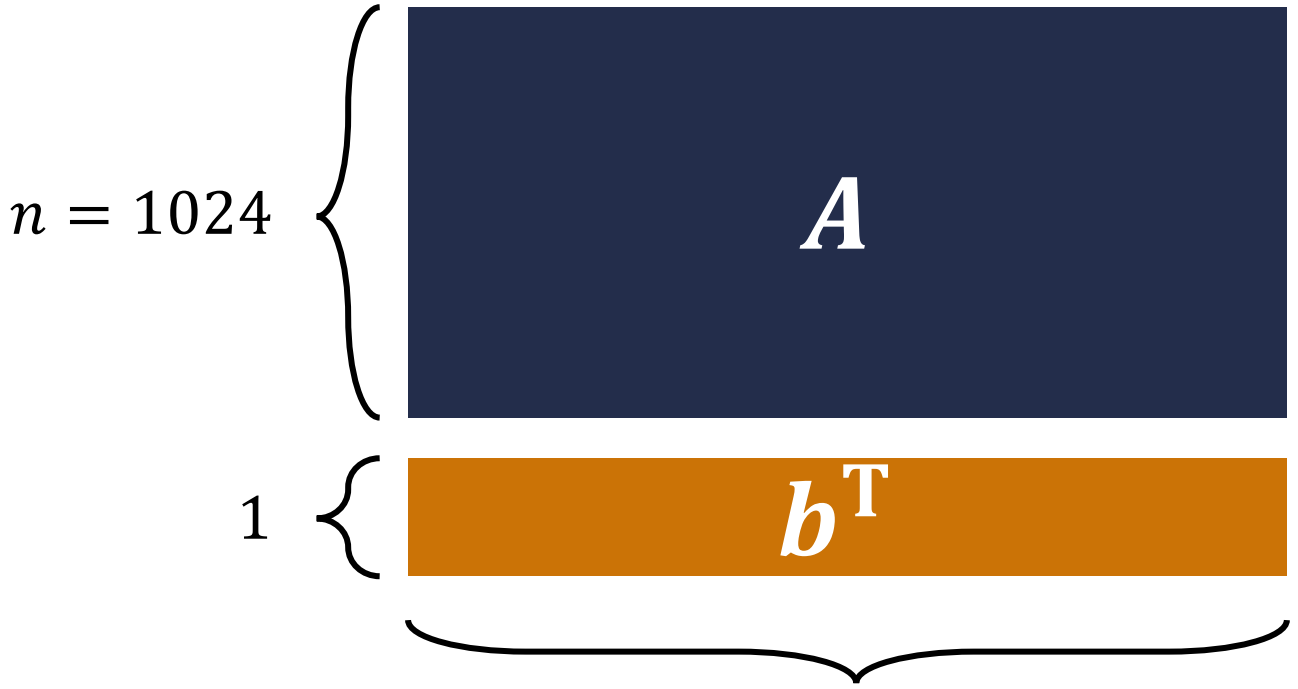
Response size: $O_\lambda(\sqrt{K})$

Homomorphically compute product between query vector and database matrix

SimplePIR: Lightweight PIR from LWE

[HHCMV23]

Building block: Regev's linearly homomorphic encryption from LWE [Reg05]



Elements of $\boldsymbol{A}$ are random 32-bit integers

$\boldsymbol{b}$ is a ciphertext (encrypts an indicator vector)

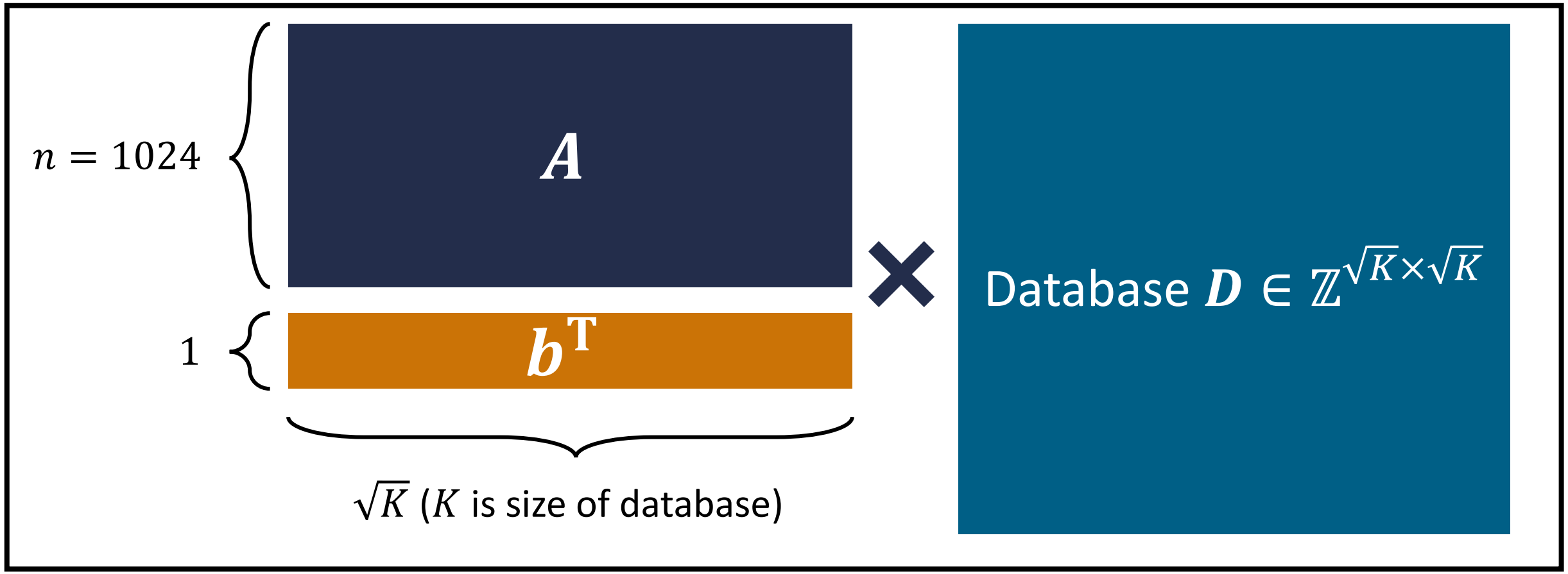
Query structure in SimplePIR

All components are elements of $\mathbb{Z}_q = \mathbb{Z}_{2^{32}}$ (i.e., 32-bit integers)

SimplePIR: Lightweight PIR from LWE

[HHCMV23]

Building block: Regev's linearly homomorphic encryption from LWE [Reg05]



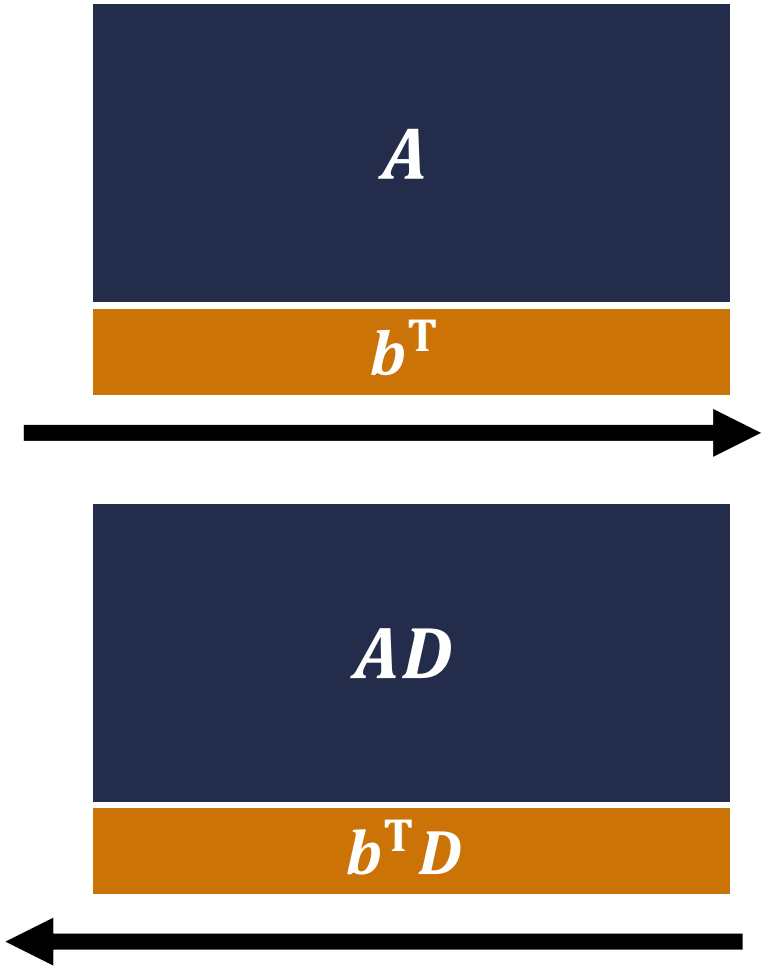
All components are elements of $\mathbb{Z}_q = \mathbb{Z}_{2^{32}}$ (i.e., 32-bit integers)

SimplePIR: Lightweight PIR from LWE

[HHCMV23]

To recover record in row j:

$q = 2^{32}, n = 2^{10}$



Query size:
$(n+1)\sqrt{K}$ elements over $\mathbb{Z}_q$

Database
$D \in \mathbb{Z}^{\sqrt{N} \times \sqrt{N}}$

Response size:
$(n+1)\sqrt{K}$ elements over $\mathbb{Z}_q$

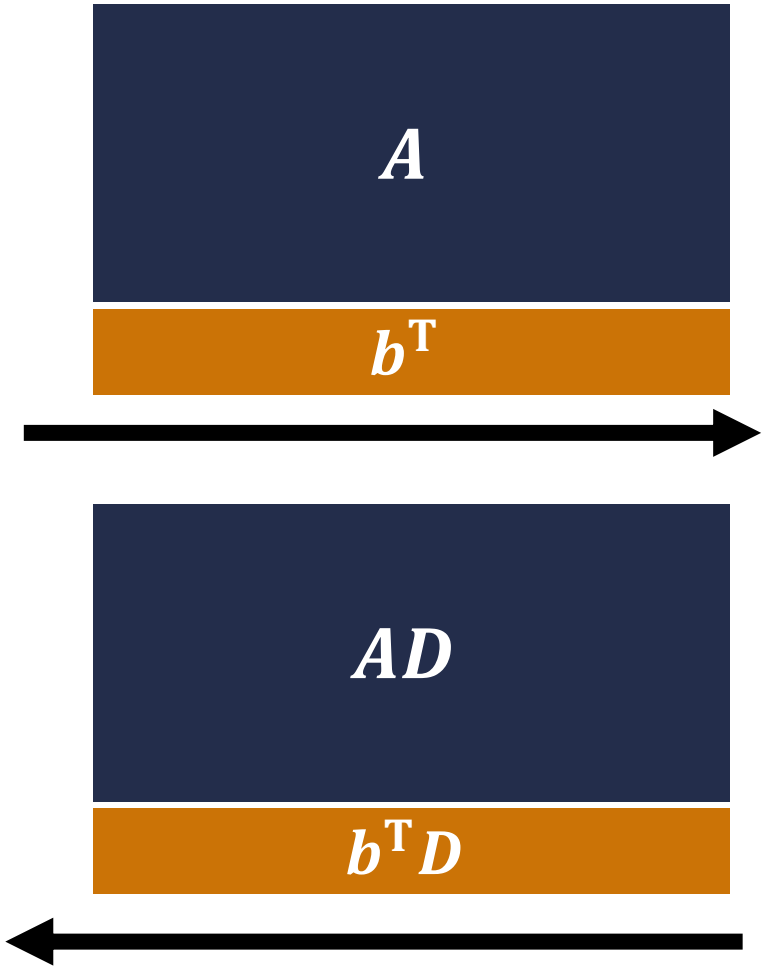
Server computation:
$2(n+1)K$ operations over $\mathbb{Z}_q$

2000 operations per database entry!

SimplePIR: Lightweight PIR from LWE

[HHCMV23]

Key insight in SimplePIR: $\boldsymbol{A}$ is query-independent so move computation of $\boldsymbol{AD}$ offline



Database
$D \in \mathbb{Z}^{\sqrt{K} \times \sqrt{K}}$

Query size:
$(n+1)\sqrt{K}$ elements over $\mathbb{Z}_q$

Response size:
$(n+1)\sqrt{K}$ elements over $\mathbb{Z}_q$

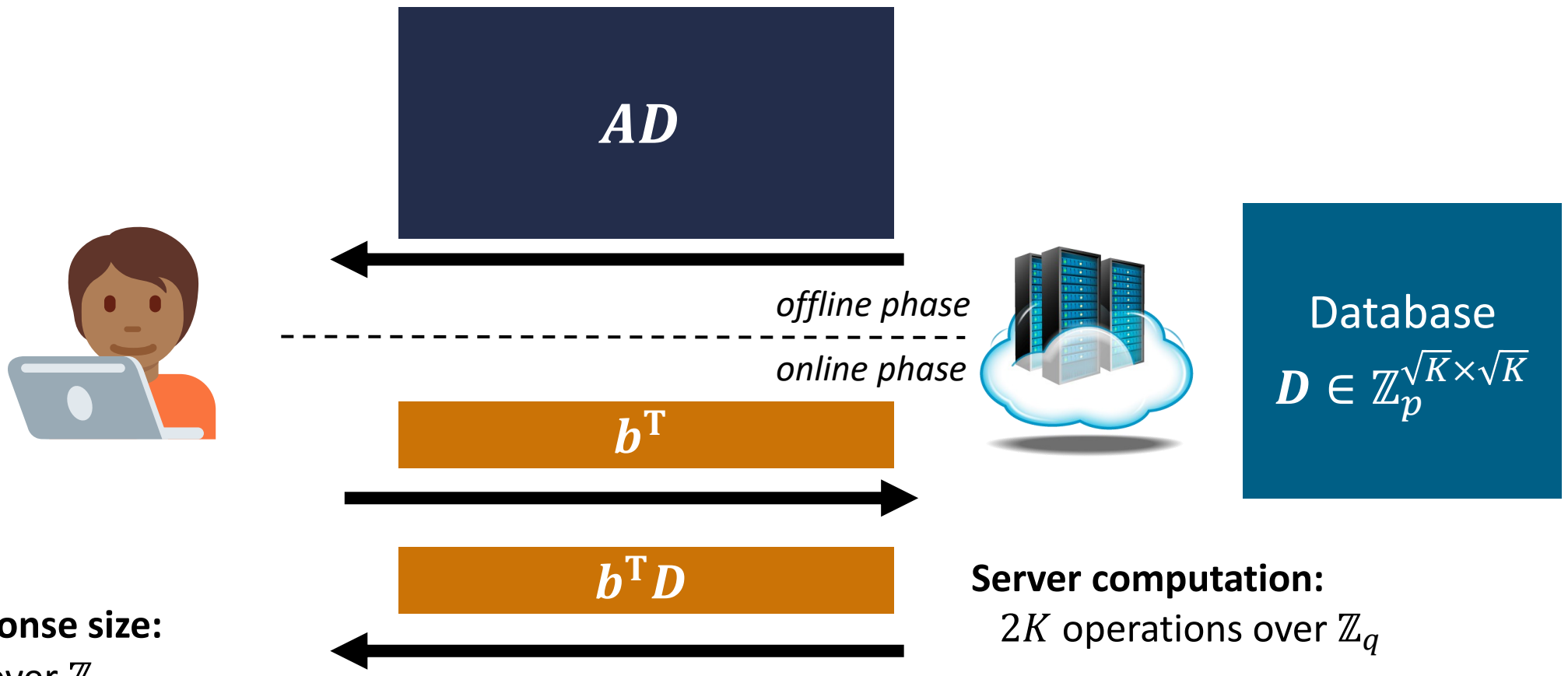
Server computation:
$2(n+1)K$ operations over $\mathbb{Z}_q$

2000 operations per database entry!

SimplePIR: Lightweight PIR from LWE

[HHCMV23]

Key insight in SimplePIR: $\boldsymbol{A}$ is query-independent so move computation of $\boldsymbol{AD}$ offline



Query and response size:
$\sqrt{K}$ elements over $\mathbb{Z}_q$

Server computation:
$2K$ operations over $\mathbb{Z}_q$

2 operations per database entry

SimplePIR: Lightweight PIR from LWE

[HHCMV23]

Key insight in SimplePIR: $\boldsymbol{A}$ is query-independent so move computation of $\boldsymbol{AD}$ offline

Limitation: hint is **large** ($n \times$ larger than response) and needs to be updated whenever database changes

$\boldsymbol{AD}$

offline phase

online phase

$\boldsymbol{b}^{\mathrm{T}}$

$\boldsymbol{b}^{\mathrm{T}}\boldsymbol{D}$

Database $\boldsymbol{D} \in \mathbb{Z}_p^{\sqrt{K} \times \sqrt{K}}$

On modern processors, can process query at *memory bandwidth* (best possible if protocol performs linear-scan)

2 operations per database entry

Query and response size:
$\sqrt{N}$ elements over $\mathbb{Z}_q$

PIR with *Silent* Preprocessing

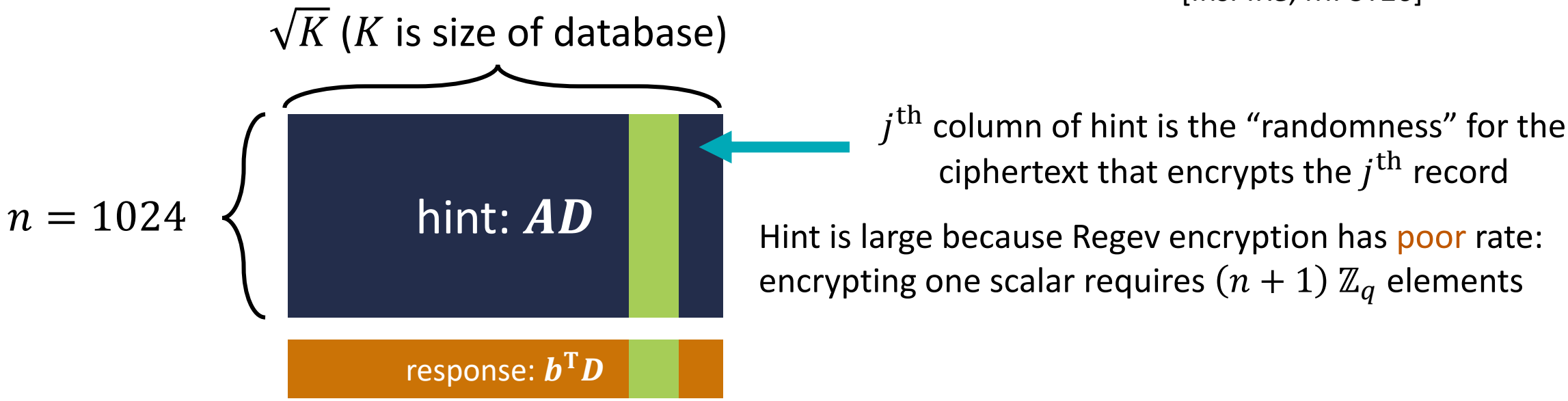
Silent preprocessing: allow **offline** preprocessing, but **no communication**

No need to manage hints; better suited for dynamic databases

Approach: *compress* the hint and include as part of the response

Why is the SimplePIR hint so large?

[Tiptoe; HDCZ23]
[HintlessPIR; LMRS24]
[YPIR; MW24]
[InsPIRe; MPSY26]



Higher Rate using Polynomial Rings

Common technique to get better rate: use polynomial rings

Ciphertext space:

$$\mathbb{Z}_q^{n+1} \longrightarrow R_q^2 = \left(\mathbb{Z}_q[x]/(x^n+1)\right)^2$$

degree-n polynomials with coefficients in $\mathbb{Z}_q$

Each ciphertext now encrypts a degree-n polynomial

$$\text{rate} = \frac{\text{size of plaintext}}{\text{size of ciphertext}} = \frac{1}{(n+1)\log q} \longrightarrow \frac{n}{2n\log q} = \frac{1}{2\log q}$$

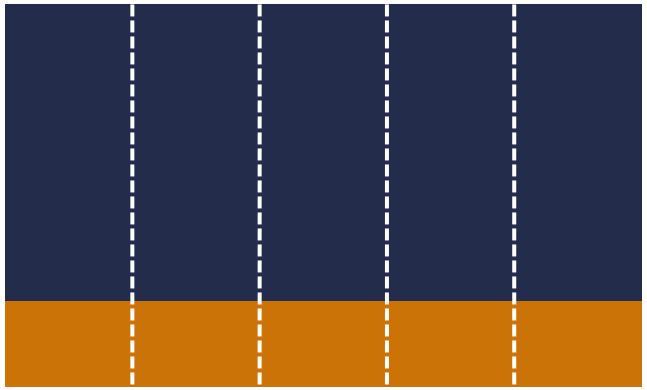
*rings allow **packing** more data into each ciphertext*

Recall $n = 2^{10} = 1024$, so using rings gives over 500× reduction in size

YPIR: Lightweight Hint Compression

[MW24]

YPIR approach: pack LWE ciphertexts into RLWE ciphertexts [CDKS21]



Each column is an LWE ciphertext encrypting a record z_i

Limitation: LWE ciphertext has **poor** rate

Idea: Convert LWE encryption of z_i into RLWE encryption of z_i

- Bootstrapping (i.e., homomorphic decryption) [HDCZ23, LMRS24]
- Homomorphic trace evaluation [MW24, MPSY26]

Packing: Evaluate linear function

$$(z_1, \ldots, z_n) \mapsto z_1 + z_2 x + \cdots + z_n x^{n-1}$$

This is a linear function over the ring

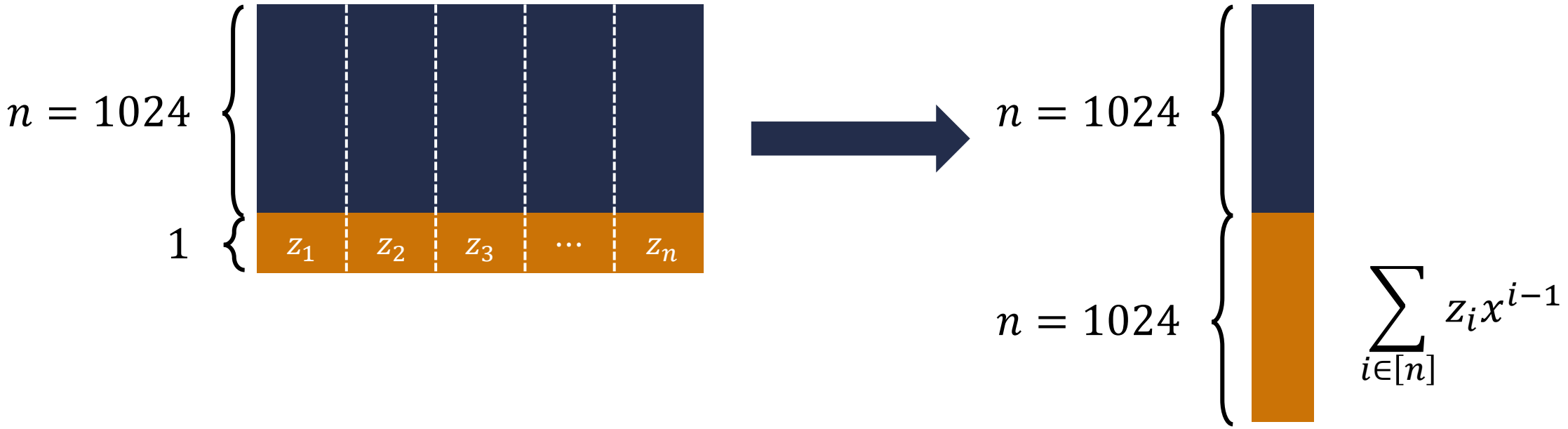
Result: RLWE encryption of polynomial $z_1 + z_2 x + \cdots + z_n x^{n-1}$

Ciphertext consists of just two ring elements

YPIR: Lightweight Hint Compression

[MW24]

YPIR approach: pack LWE ciphertexts into RLWE ciphertexts [CDKS21]



$n(n+1)\ \mathbb{Z}_q$ elements → $2n\ \mathbb{Z}_q$ elements

over $500\times$ reduction

YPIR: Lightweight Hint Compression

[MW24]

Compute:

Database $D \in \mathbb{Z}^{\sqrt{K} \times \sqrt{K}}$

AD

No communication!

offline phase

online phase

b^T

packing keys

Performance Comparison

[MW24]

Setup: 8 GB database, 1-bit records

Corresponds to application for certificate transparency auditing

[HHCMV23] uses DoublePIR (recursive version of SimplePIR to reduce hint size)

		DoublePIR	HintlessPIR	YPIR
Offline	Download	14 MB	–	–
Online	Upload	960 KB	1.4 MB	1.5 MB
	Download	12 KB	1.7 MB	12 KB
	Throughput	12.5 GB/s	4.9 GB/s	11.6 GB/s

1.6× larger queries and 93% of the throughput of fastest scheme

Hardware Acceleration

[SW26]

On commodity CPUs: memory bandwidth is ~ 10 GB/s

SimplePIR family of protocols: memory bandwidth is the bottleneck

On GPUs: memory bandwidth (for plain matrix multiplication) is ~ 2 TB/s when saturating available compute

Single PIR query does not saturate the available compute

Idea: cross-client batching (process multiple independent queries together)

Main protocol is computing $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{D}$, where $\boldsymbol{b}$ is the query vector and $\boldsymbol{D}$ is the database

If ℓ queries $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_\ell$ arrive together, can stack them together and compute $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}$ where i^{th} row of $\boldsymbol{B}^{\mathrm{T}}$ is $\boldsymbol{b}_i^{\mathrm{T}}$

Can also recast ring packing as a matrix multiplication

For 8 GB database (on a single Nvidia L4 GPU):

Single-query throughput: 143 GB/s

128-query effective throughput: 8.5 TB/s

Saturates available compute

On an A100 and batch size of 1024, can achieve effective throughput over 20 TB/s

Effective throughput for k queries = $\frac{k \cdot |\mathrm{DB}|}{\mathrm{time}}$

Private Wikipedia Demo

[SW26]

https://www.cs.utexas.edu/~dwu4/pir-demo

Demo built by Sidaarth Sabhnani

Open Problems

Reduce concrete communication costs of PIR with silent preprocessing

- Current approaches all have $O(\sqrt{K})$ communication – can we get to $\mathrm{polylog}(K)$ with similar (concrete) computational overhead

Practical **doubly-efficient** PIR (single-server ***or*** multi-server)

What will it take to deploy PIR in practical systems?

- **Apple:** private caller ID lookup and private image search
- **Meta:** advanced browsing protection (URL filtering)

Thank you!