



SimplePIR: Simple and Fast Single-Server Private Information Retrieval

Alexandra Henzinger

MIT

Matthew M. Hong

MIT

Henry Corrigan-Gibbs

MIT

Sarah Meiklejohn

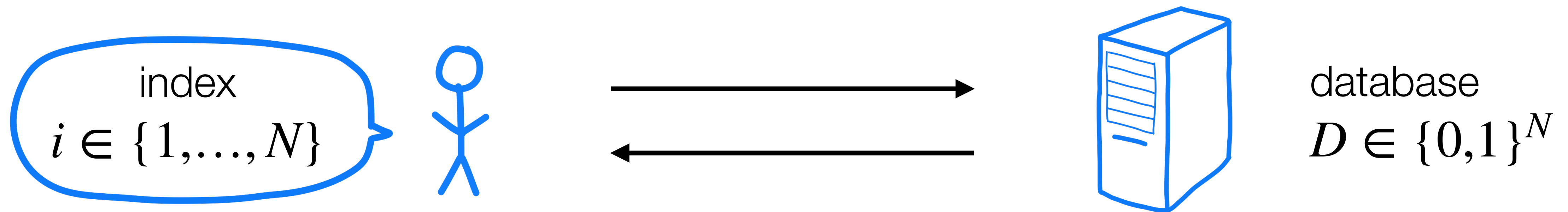
Google

Vinod Vaikuntanathan

MIT

Private information retrieval [CGKS95, KO97]

Goal: Privately read an entry from a remote database

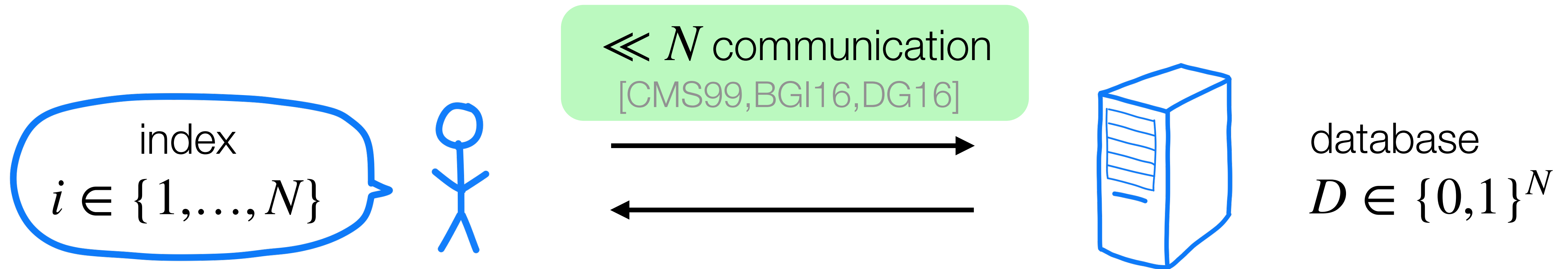


Correctness: learns i -th entry in D ,
if the server is **honest**

Privacy: learns nothing about i ,
even if the server is **malicious**

Private information retrieval [CGKS95, KO97]

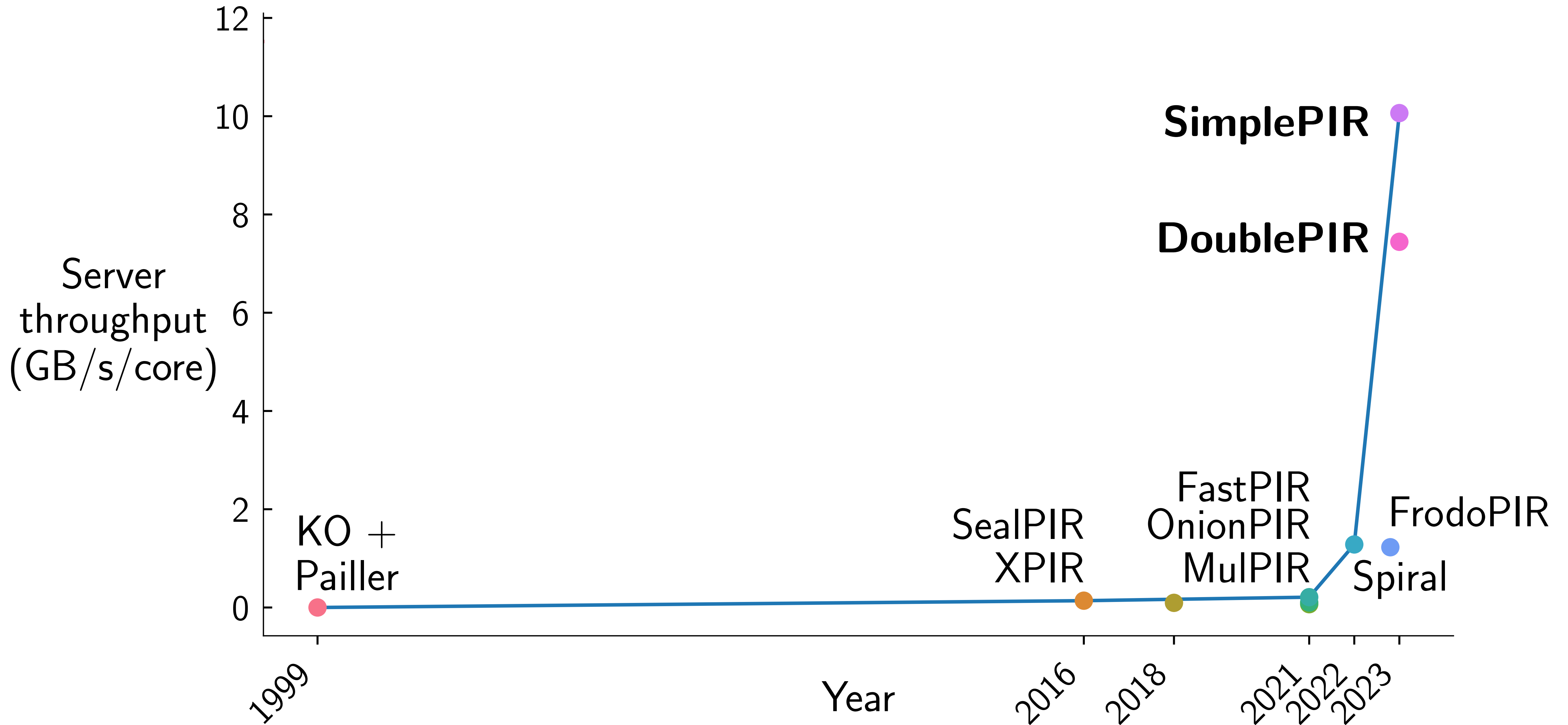
Goal: Privately read an entry from a remote database



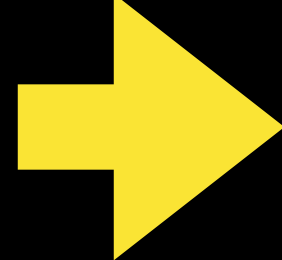
However, PIR inherently requires **lots of server-side computation**.

- ▶ Database read with no privacy: $O(1)$ time
- ▶ Database read with privacy: $\geq N$ time* (even with crypto) [BIM04, PY22]

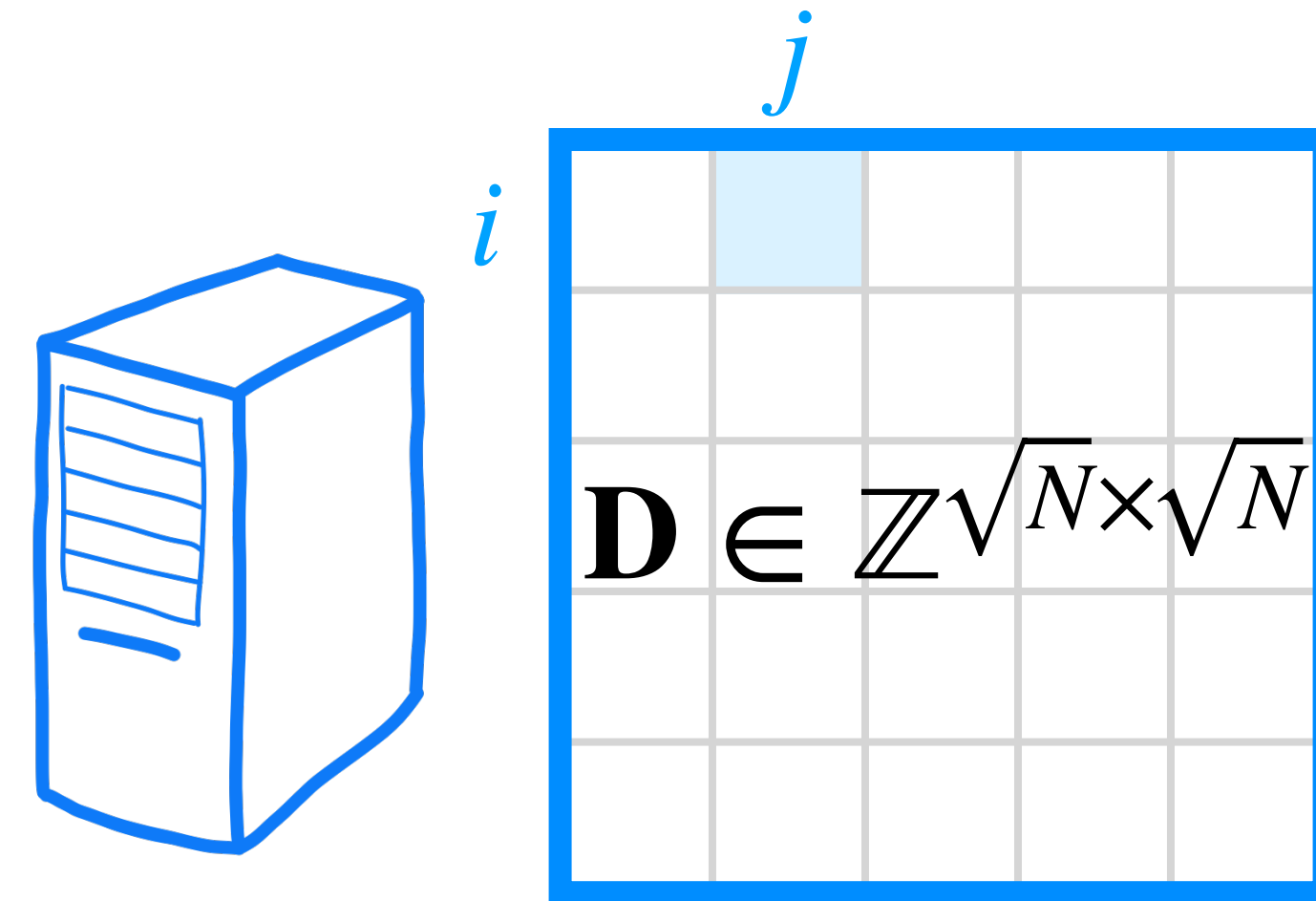
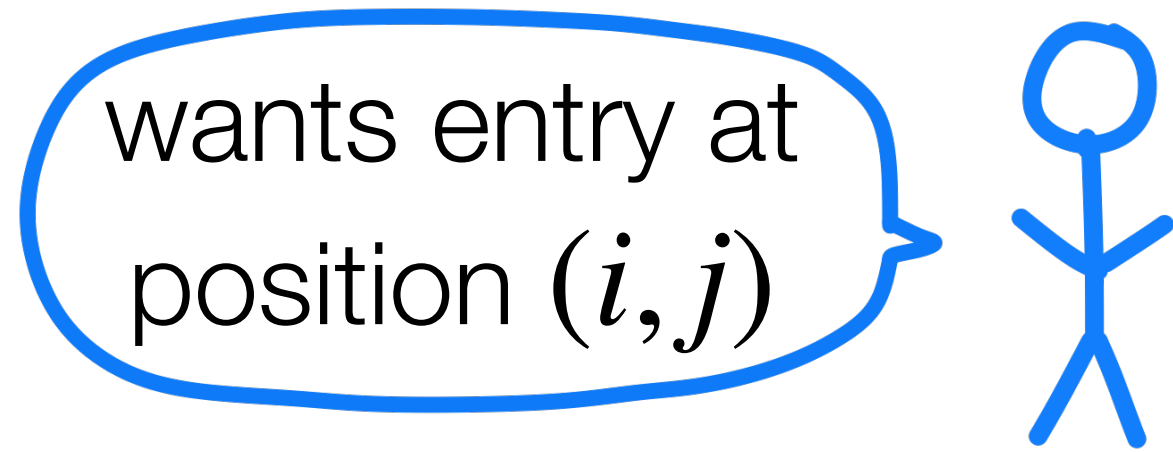
25 years of work on single-server PIR



This talk

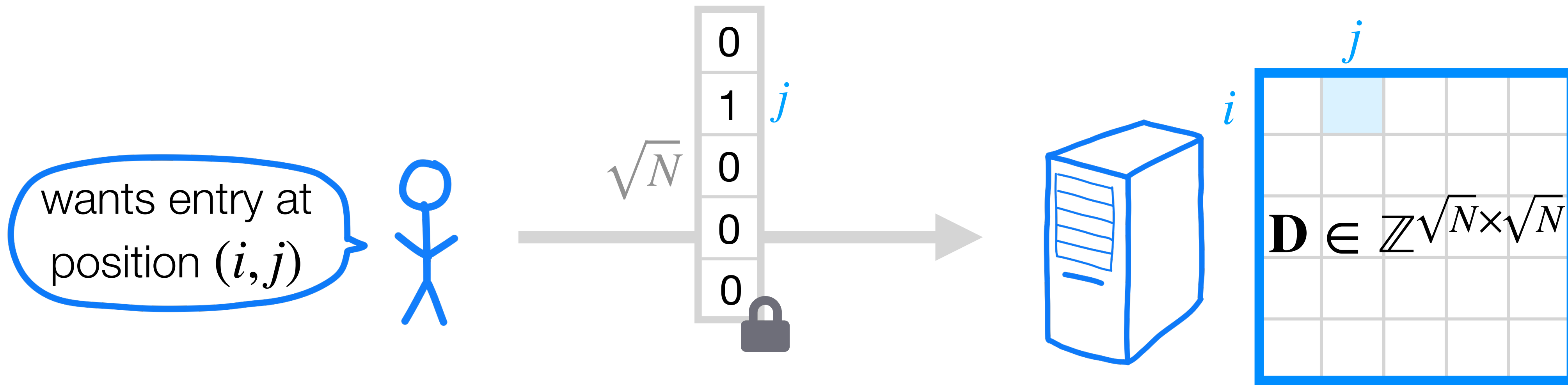
- 
1. New tool: Fast linearly homomorphic encryption
 2. Our results
 - SimplePIR: High throughput
 - DoublePIR: Less communication
 3. Evaluation

Starting point: a classic PIR scheme [KO97]



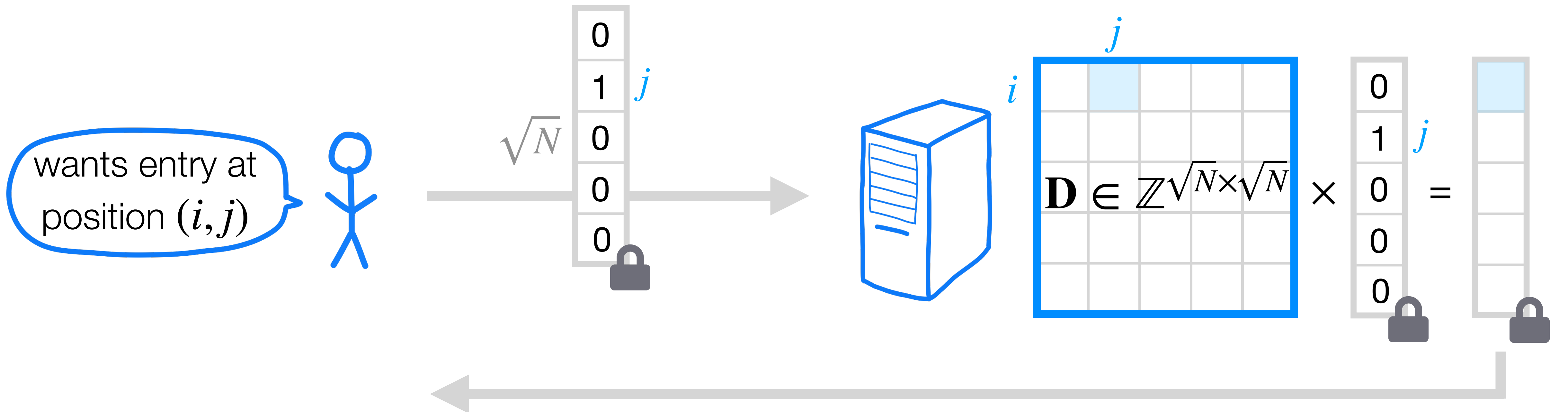
🔒 denotes linearly homomorphic encryption

Starting point: a classic PIR scheme [KO97]



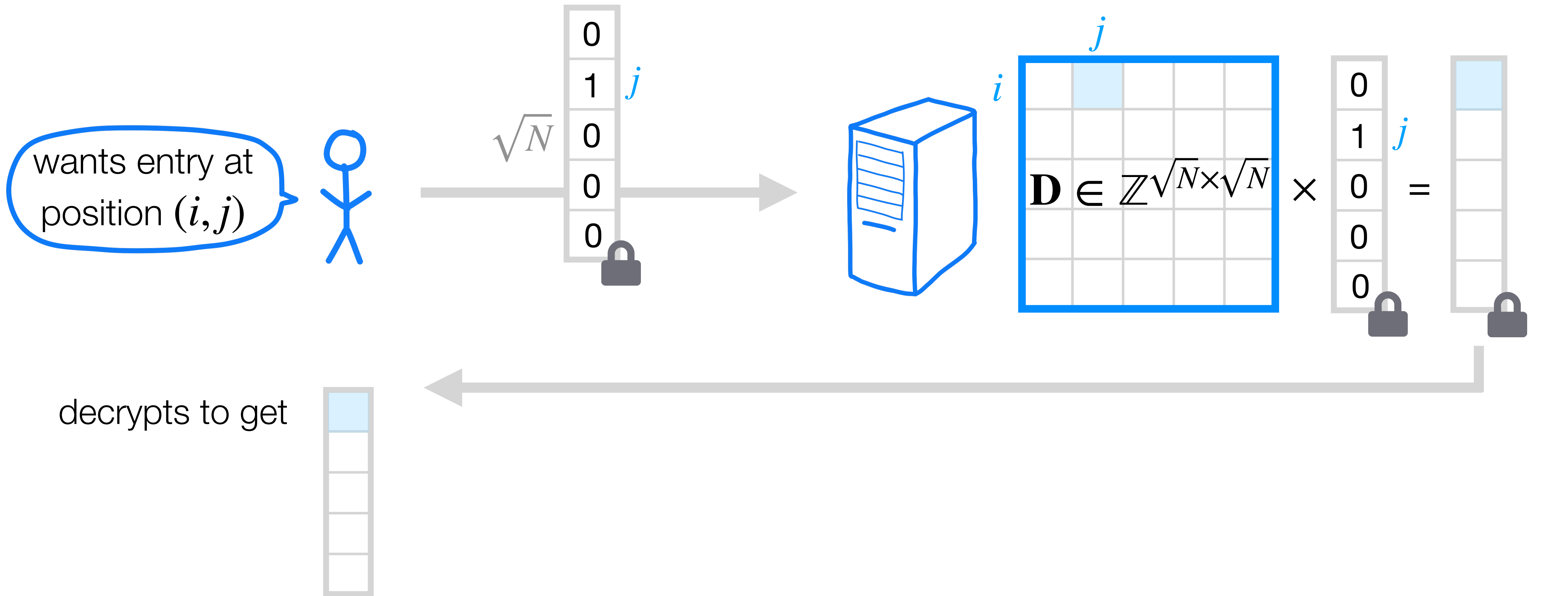
🔒 denotes linearly homomorphic encryption

Starting point: a classic PIR scheme [KO97]

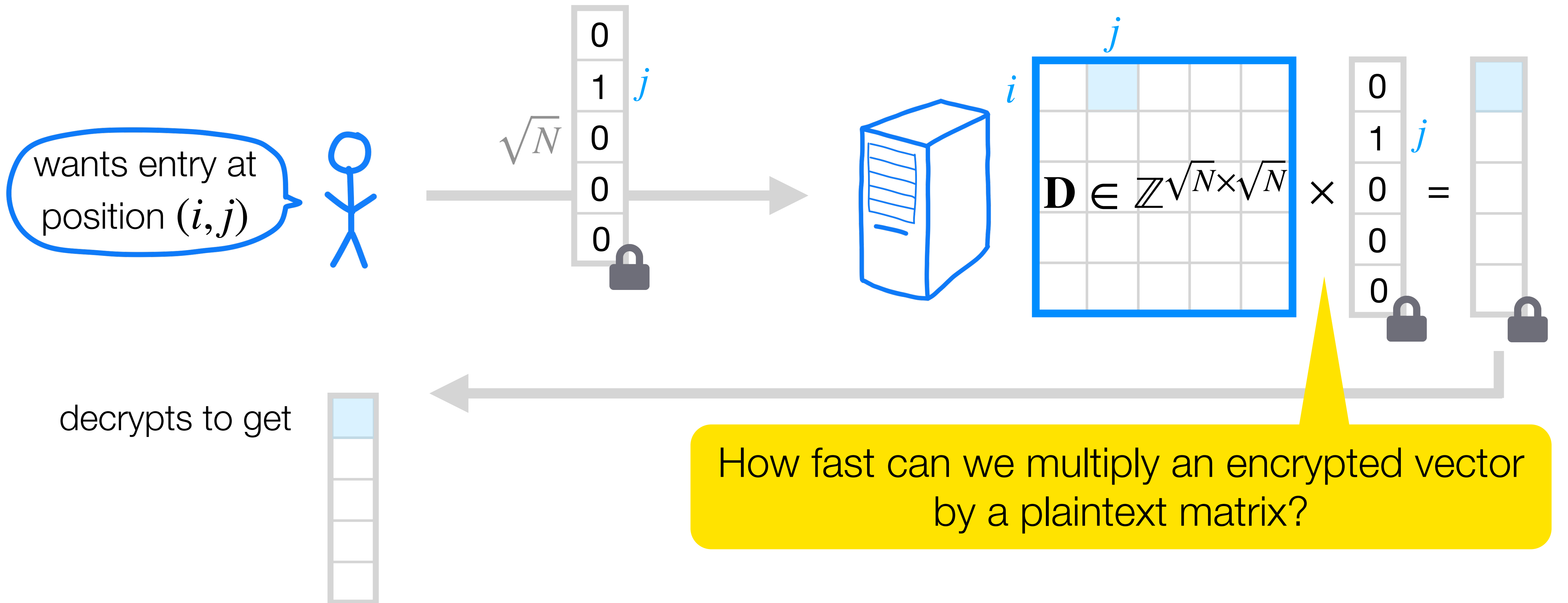


🔒 denotes linearly homomorphic encryption

Starting point: a classic PIR scheme [KO97]



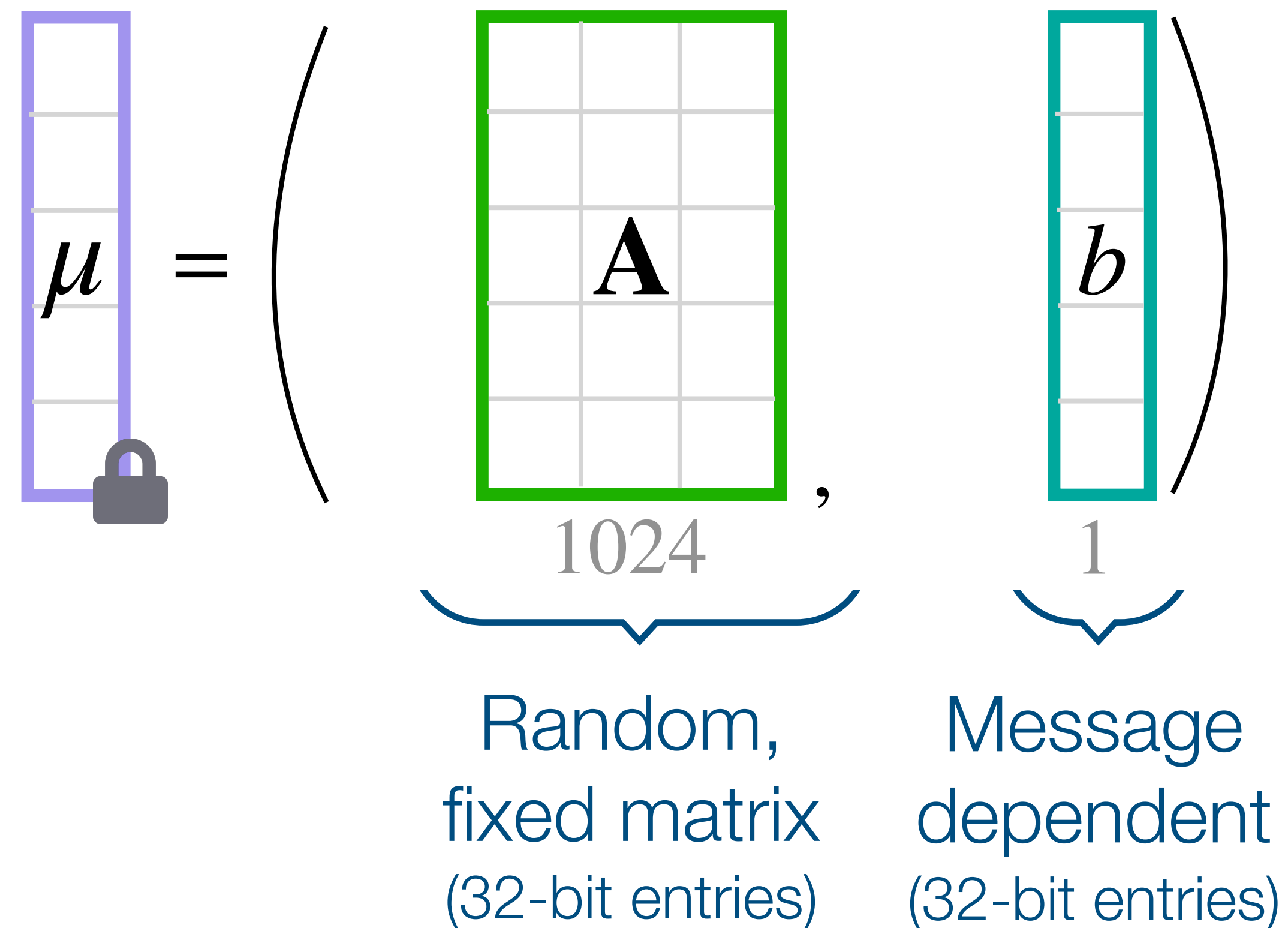
Starting point: a classic PIR scheme [KO97]



🔒 denotes linearly homomorphic encryption

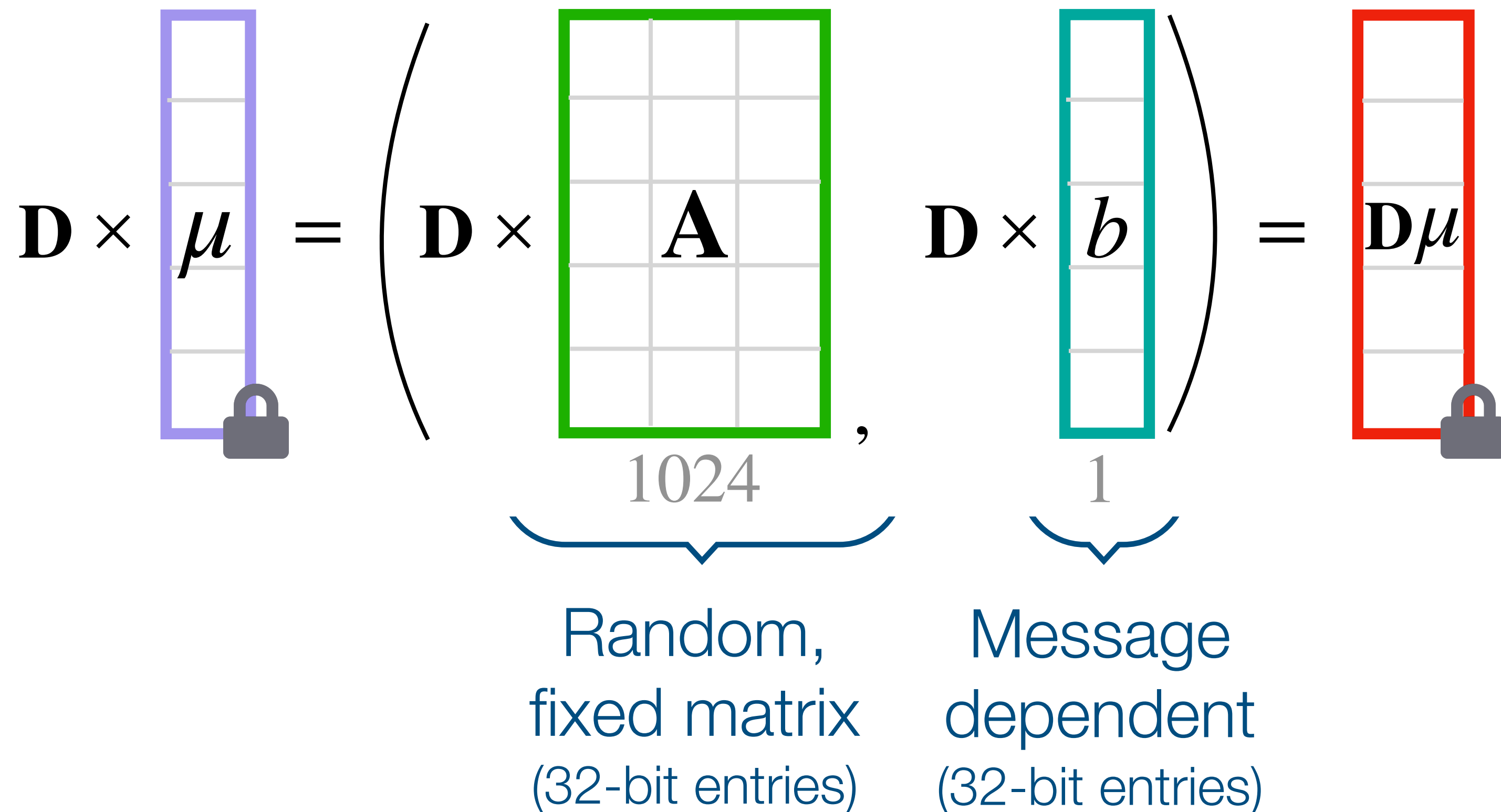
New tool: Fast linearly homomorphic encryption

Regev encryption [Reg09] of a message $\mu \in \mathbb{Z}^{\sqrt{N}}$ with 1-byte entries:



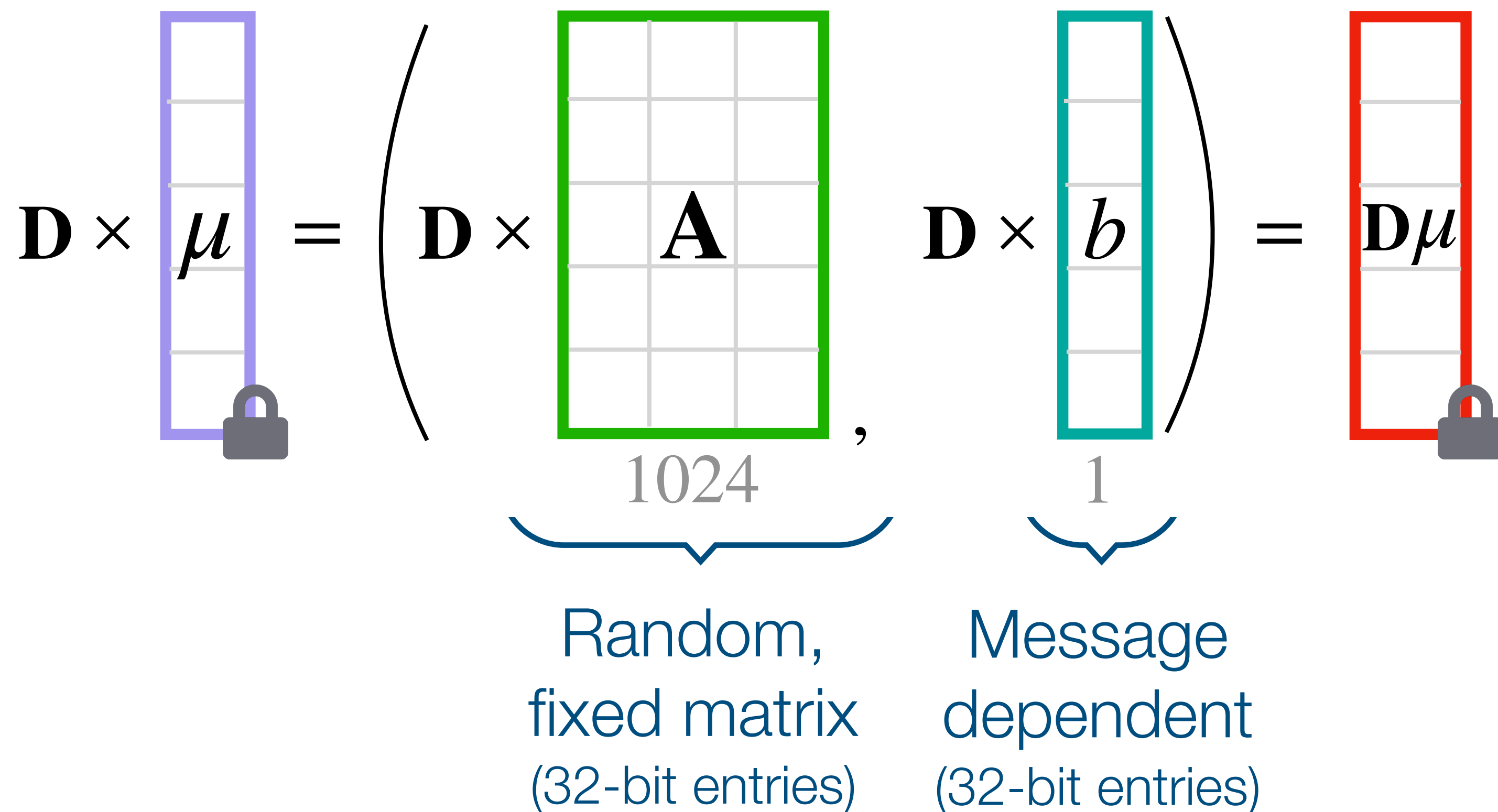
New tool: Fast linearly homomorphic encryption

Regev encryption [Reg09] of a message $\mu \in \mathbb{Z}^{\sqrt{N}}$ with 1-byte entries:



New tool: Fast linearly homomorphic encryption

Regev encryption [Reg09] of a message $\mu \in \mathbb{Z}^{\sqrt{N}}$ with 1-byte entries:



Our observations:

- ▶ Precompute $\mathbf{D} \times \mathbf{A}$ once
99.9% of the work
 - ▶ The per-message work is $\mathbf{D} \times b$
one 32-bit mul per entry in \mathbf{D}
- ⇒ Multiplying an encrypted vector with a plaintext matrix is cheap *with preprocessing*.

This talk

1. New tool: Fast linearly homomorphic encryption

 2. Our results

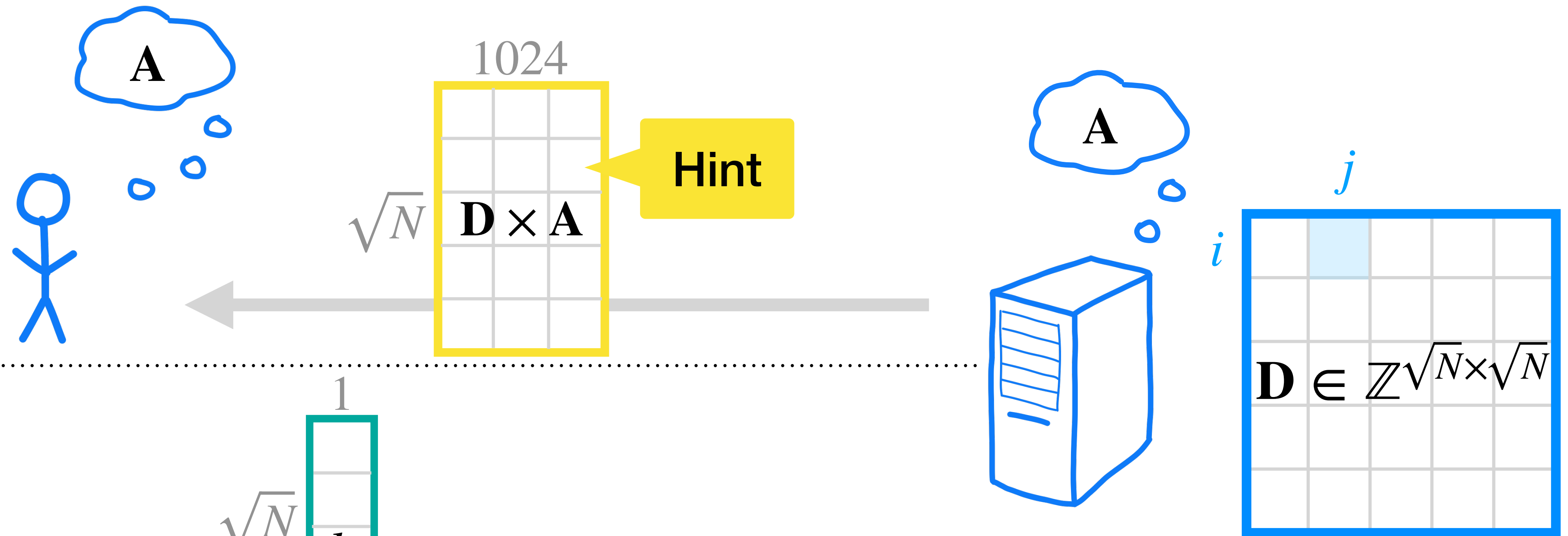
➤ SimplePIR: High throughput

➤ DoublePIR: Less communication

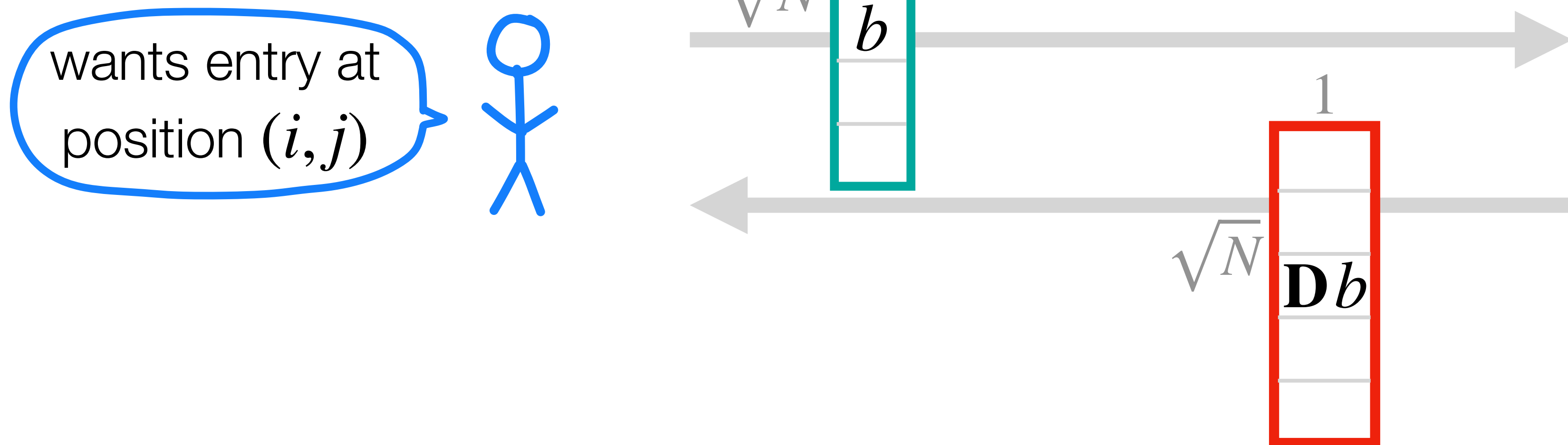
3. Evaluation

SimplePIR: fast PIR with one-time preprocessing

Offline

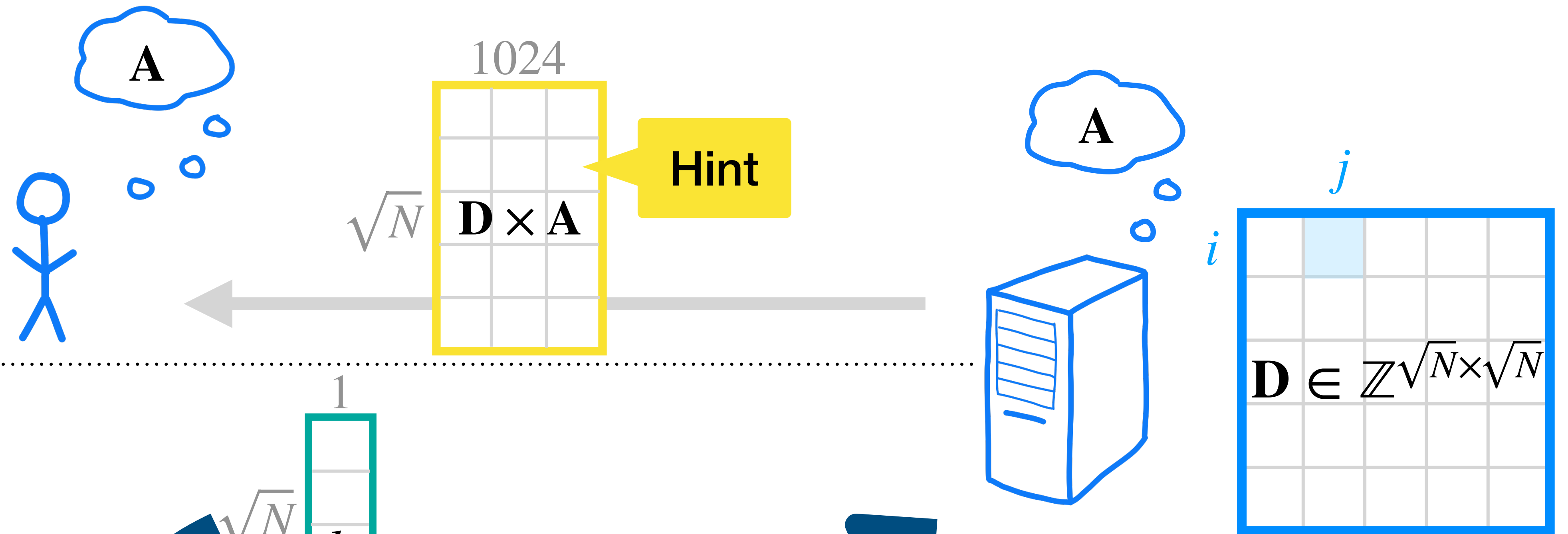


Online

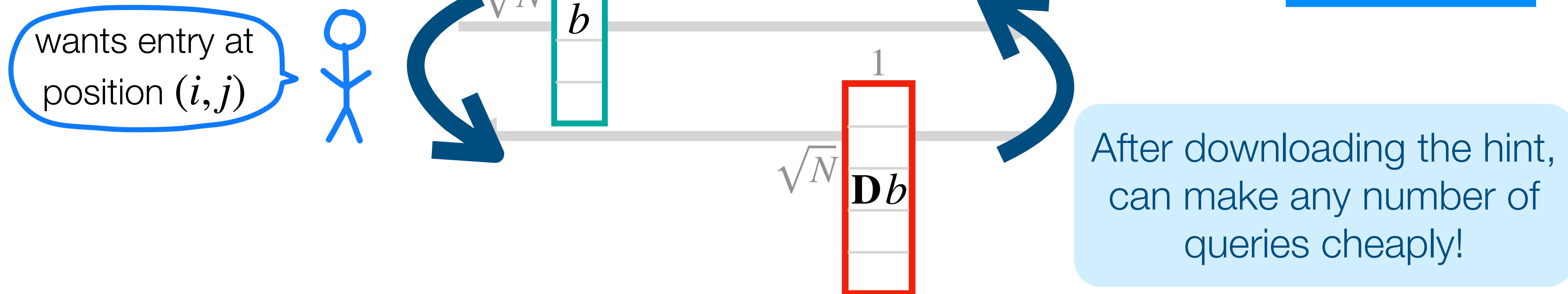


SimplePIR: fast PIR with one-time preprocessing

Offline



Online



Result 1: SimplePIR

Assuming LWE, we build single-server PIR where, on an N -byte database,

- ▶ the client downloads a one-time “hint” of size $4096 \cdot \sqrt{N}$ bytes,
- ▶ the per-query communication is $8 \cdot \sqrt{N}$ bytes, and
- ▶ the server performs N 32-bit adds and muls per query [**10 GB/s/core**].

Result 2: DoublePIR

On databases with 1-byte entries, we can shrink the hint to a constant 16 MB with a small decrease in throughput [**7 GB/s/core**].

Result 1: SimplePIR

Assuming LWE, we build single-server PIR where, on an N -byte database,

- ▶ the client downloads a one-time “hint” of size $4096 \cdot \sqrt{N}$ bytes,
- ▶ the per-query communication is $8 \cdot \sqrt{N}$ bytes, and
- ▶ the server performs N 32-bit adds and muls per query [**10 GB/s/core**].

Result 2: DoublePIR

On databases with **1-byte entries**, we can shrink the hint to a constant **16 MB** with a small decrease in throughput [**7 GB/s/core**].

See paper

This talk

1. New tool: fast linearly homomorphic encryption

2. Our results

➤ SimplePIR: High throughput

➤ DoublePIR: Less communication

 3. Evaluation

Implementation



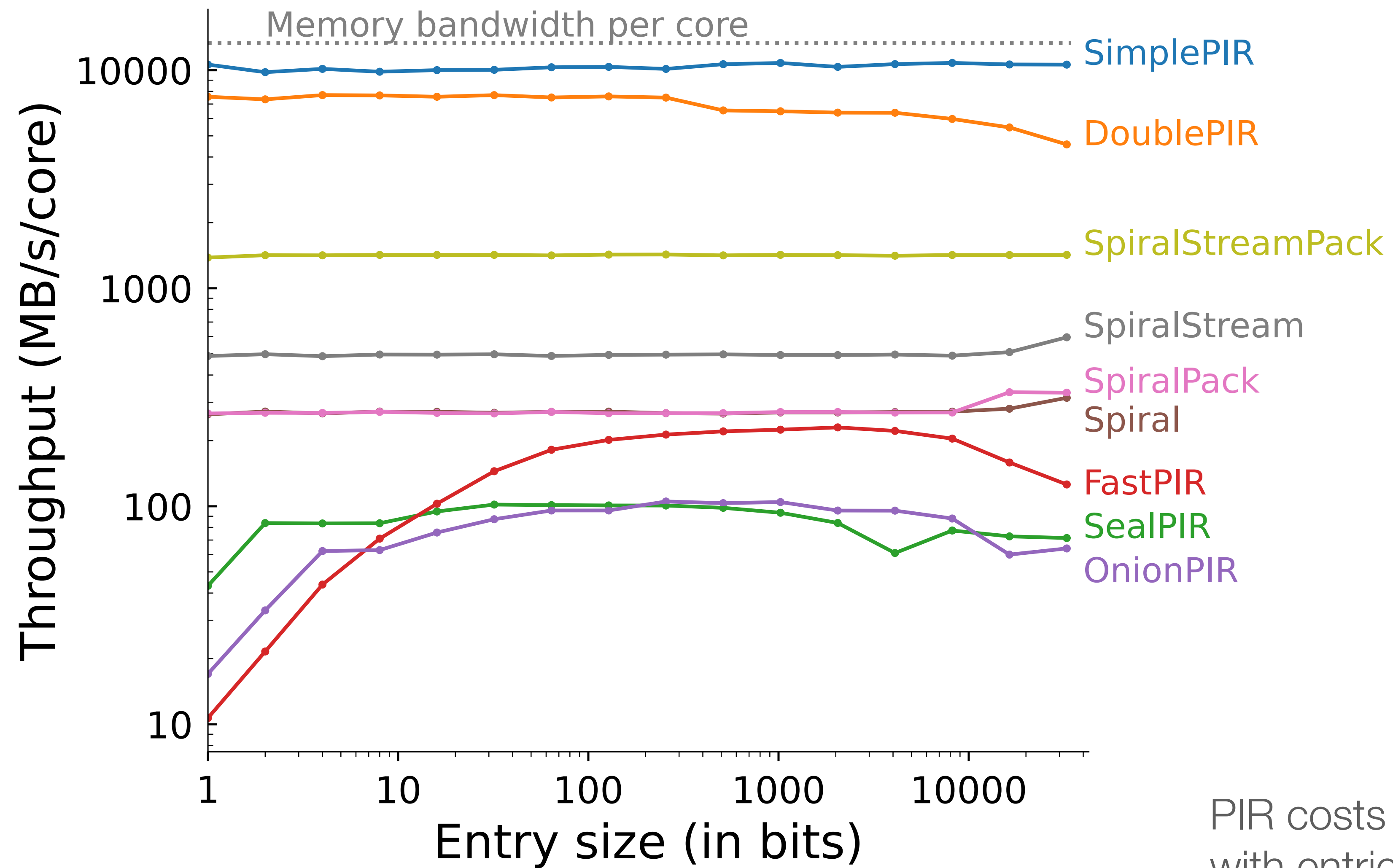
Open-source code at github.com/ahenzinger/simplepir

- ▶ 1,600 lines of Go/C for SimplePIR and DoublePIR
- ▶ No external libraries
- ▶ For speed: SIMD instructions, optimized memory access locality

Experimental setup

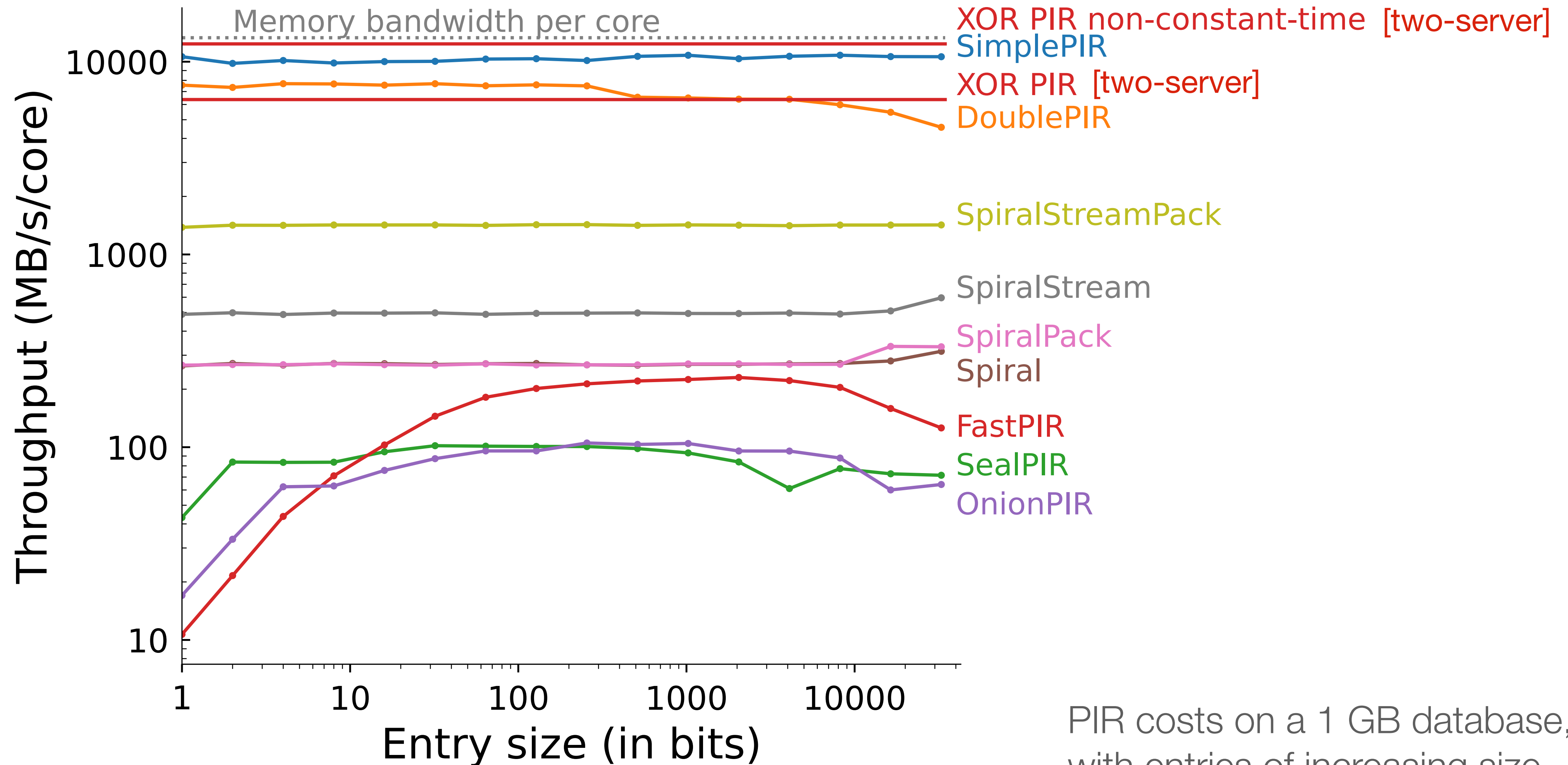
- ▶ Server runs on an AWS c5n.metal instance
- ▶ All experiments on a 1 GB database, with a single thread of execution

SimplePIR is the fastest known single-server PIR



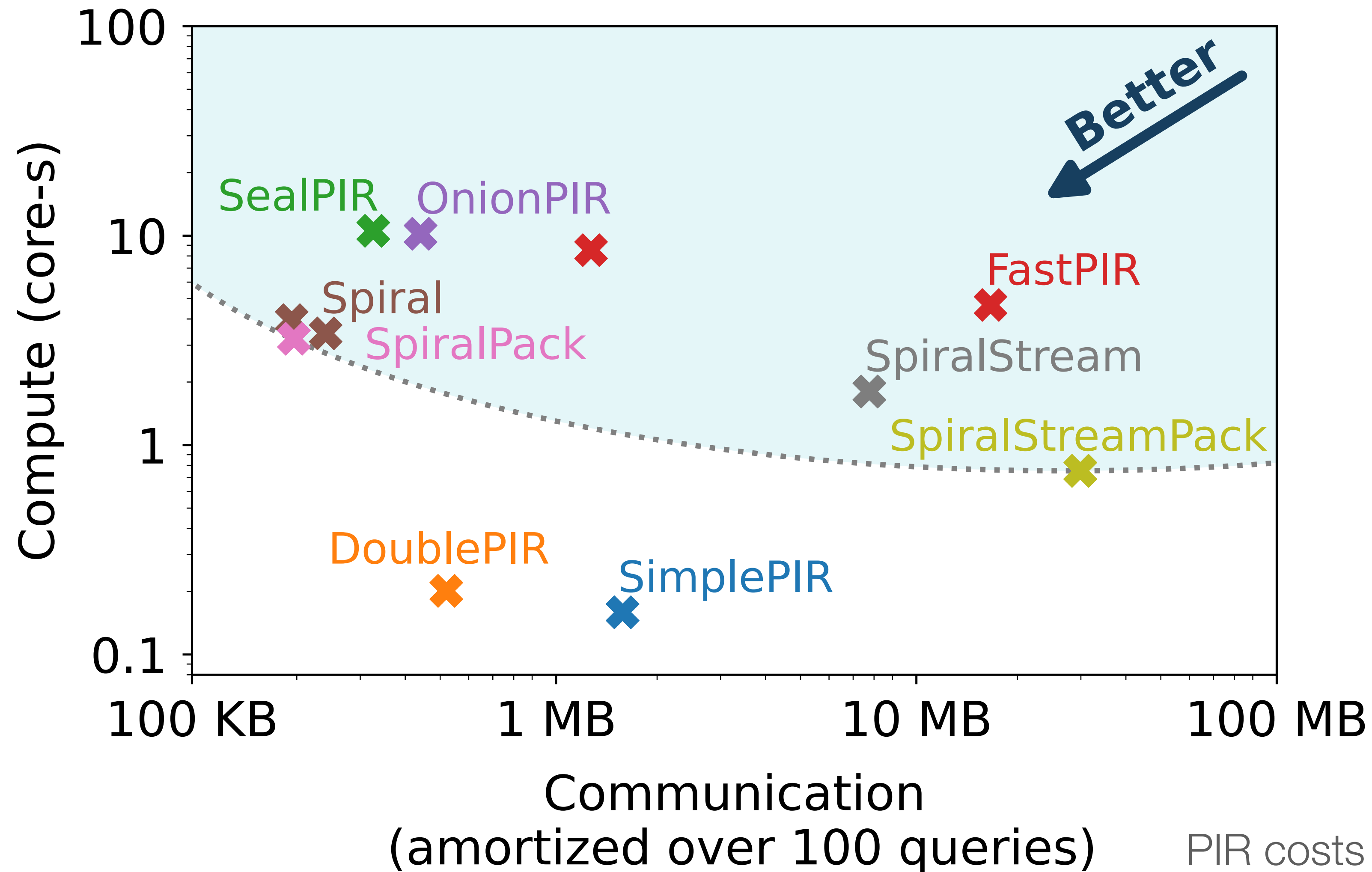
PIR costs on a 1 GB database, with entries of increasing size.

SimplePIR is almost as fast as two-server PIR



PIR costs on a 1 GB database, with entries of increasing size.

SimplePIR saves compute at the cost of communication



PIR costs on a 1 GB database,
with the optimal entry size.

With SimplePIR, PIR can be fast... what's next?

1. Reduce the communication cost.

2. Deploy PIR to protect our privacy.

- ▶ This paper: DoublePIR to privately audit webpage certificates
- ▶ Blyss demo: DoublePIR to privately check for password compromise
- ▶ SOSPI 2023: SimplePIR for private web search [HDCZ23]

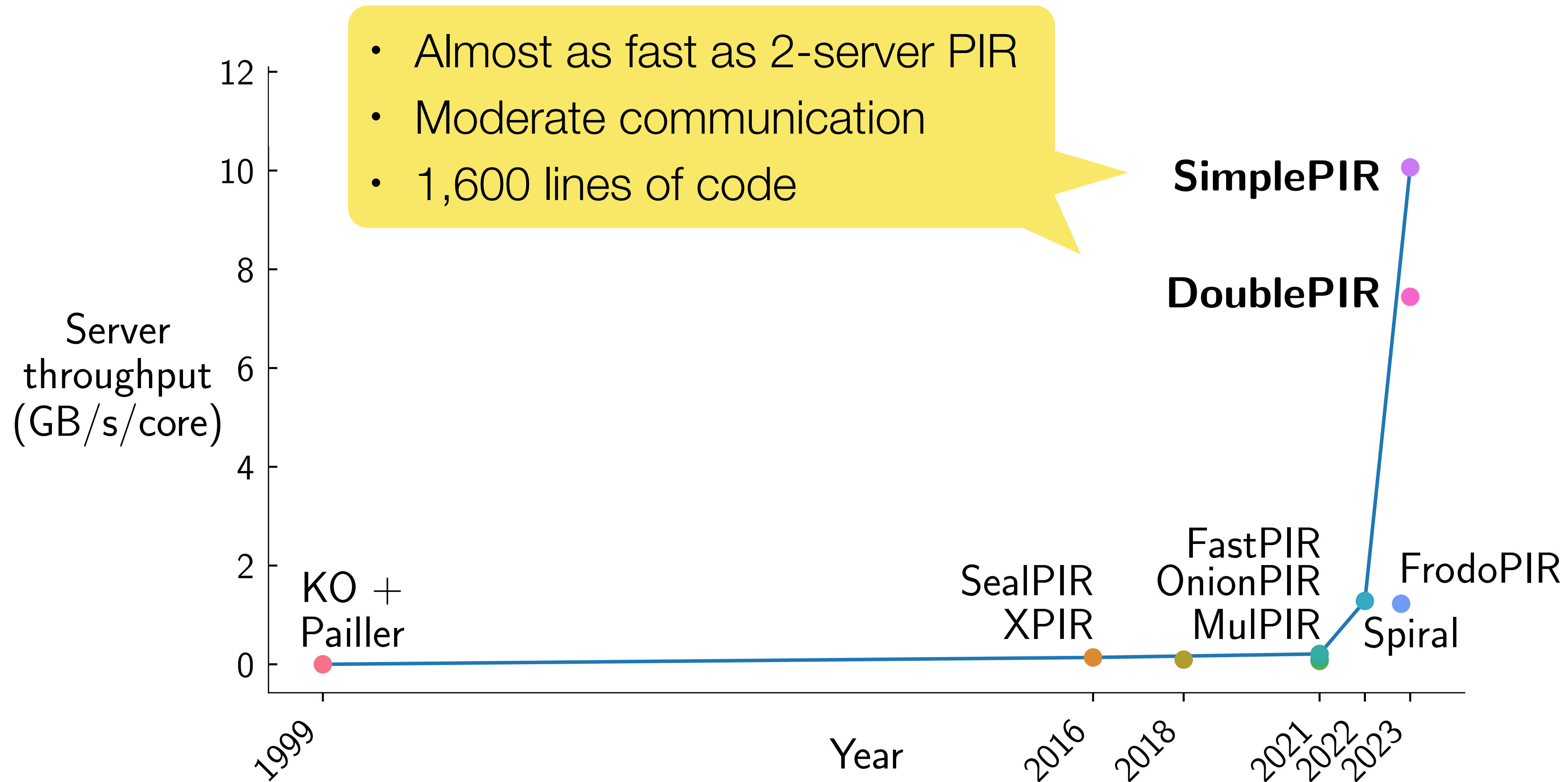
With SimplePIR, PIR can be fast... what's next?

1. Reduce the communication cost.

2. Deploy PIR to protect our privacy.

- ▶ This paper: DoublePIR to privately [audit webpage certificates](#)
- ▶ Blyss demo: DoublePIR to privately check for [password compromise](#)
- ▶ SOSp 2023: SimplePIR for private [web search](#) [HDCZ23]

See paper



Paper: eprint.iacr.org/2022/949

Code: github.com/ahenzinger/simplepir

Independent demo: playground.blyss.dev/passwords

Alexandra Henzinger

ahenz@csail.mit.edu