CS 350S: Privacy-Preserving Systems

Private Information Retrieval I

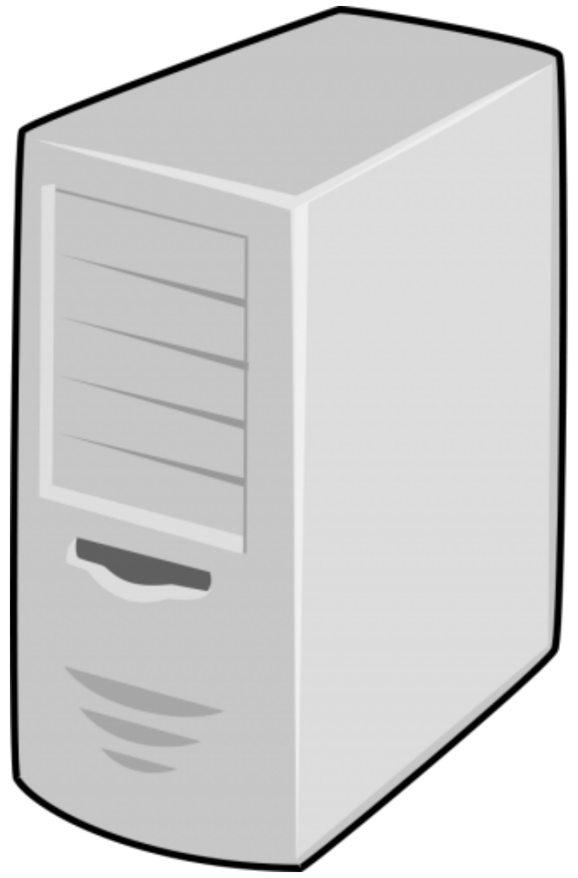
Outline

1. Overview
2. Two-server PIR construction
3. Single-server PIR construction
4. PIR with preprocessing overview

Users need to make sensitive queries to databases

Lung cancer

Record 1

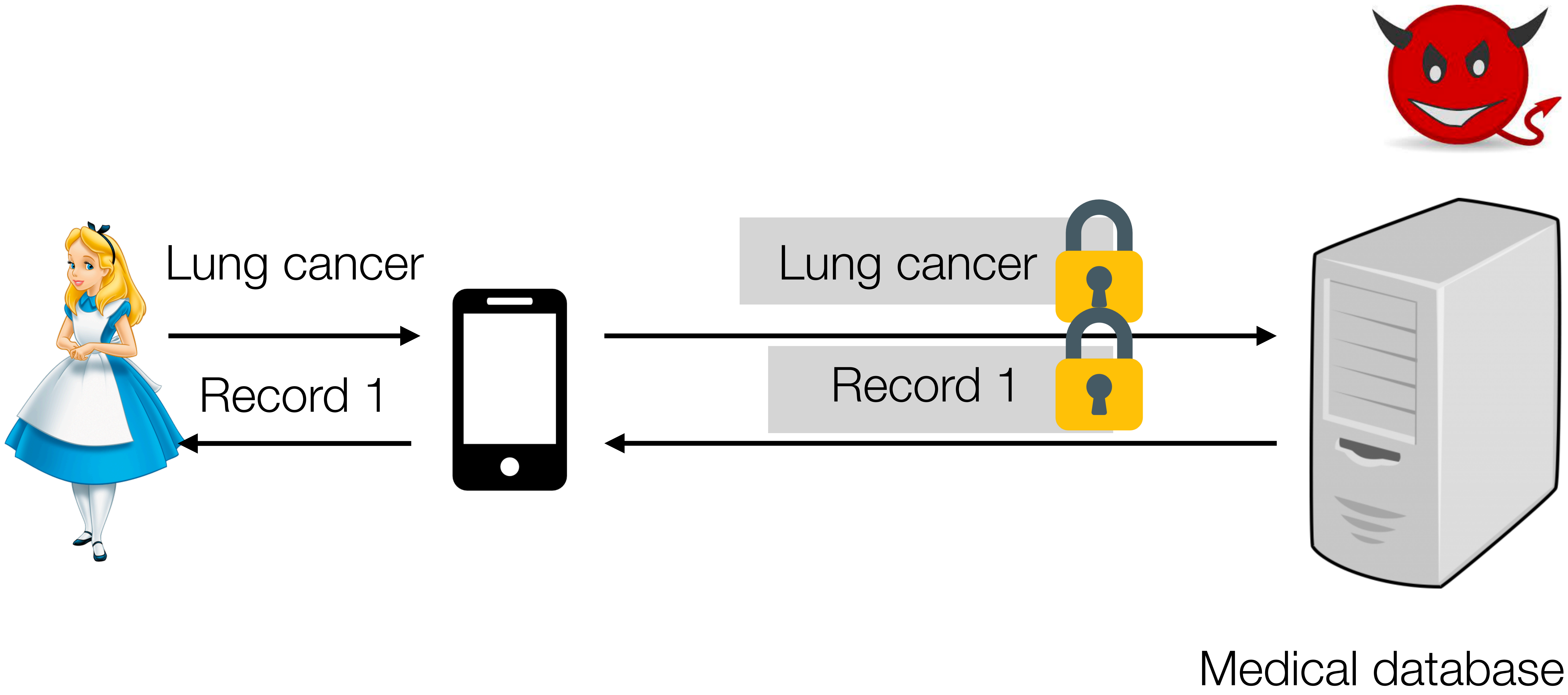


Medical database

Heart disease	Record 1
Lung cancer	Record 2
Diabetes	Record 3
…	…

Private information retrieval

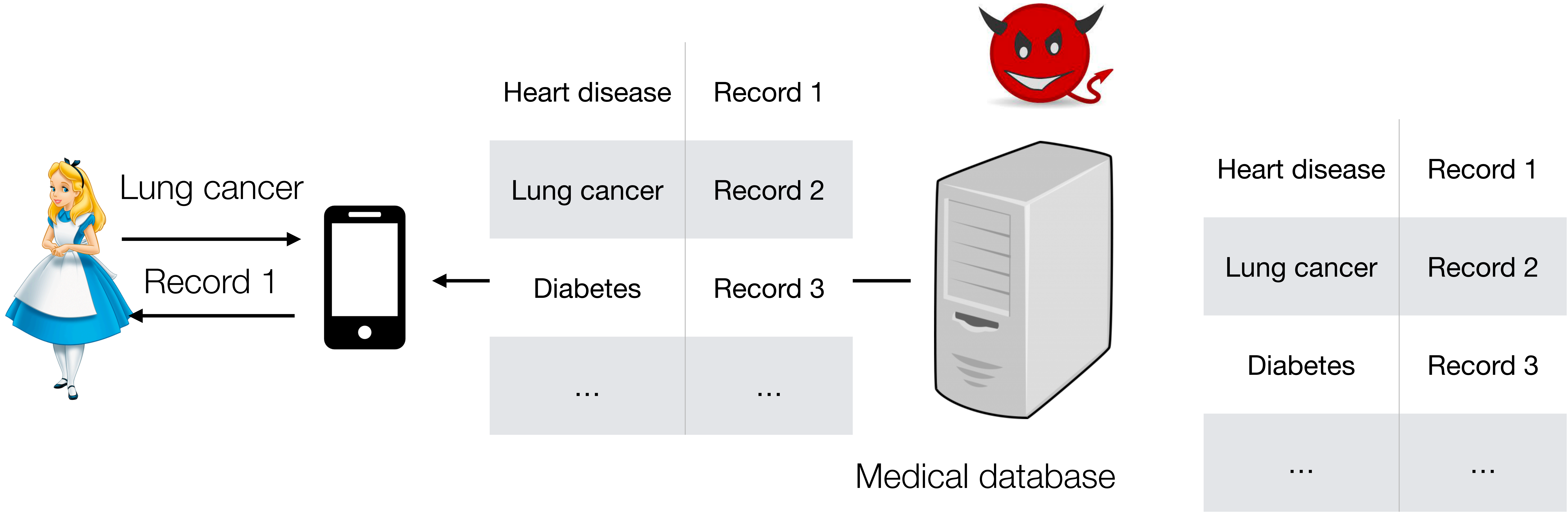
Goal: Attacker that controls the server learns nothing about a user's query



Heart disease	Record 1
Lung cancer	Record 2
Diabetes	Record 3
…	…

Can also use to fetch articles, images, podcasts, movies, etc. from a remote server

Trivial solution: download the whole database

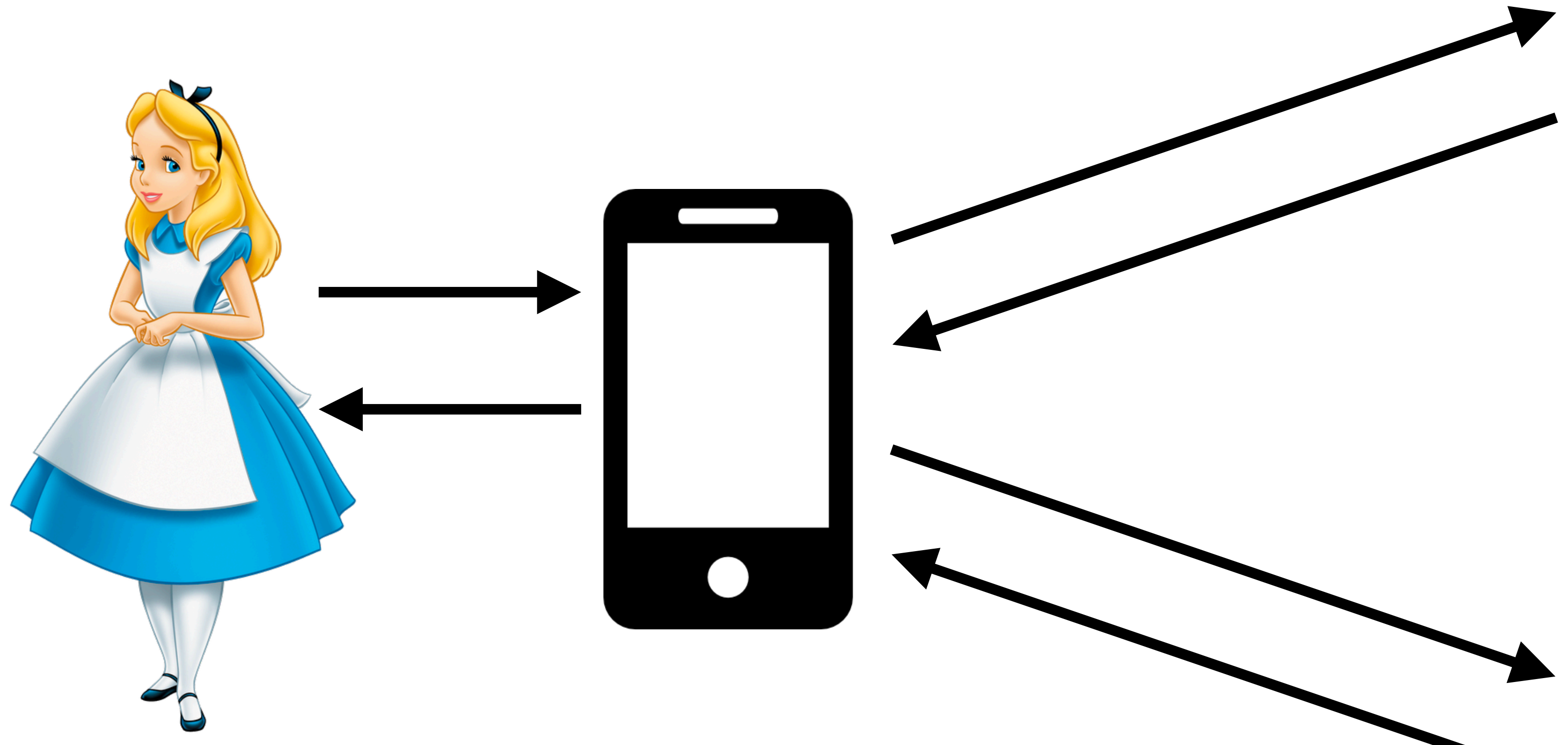


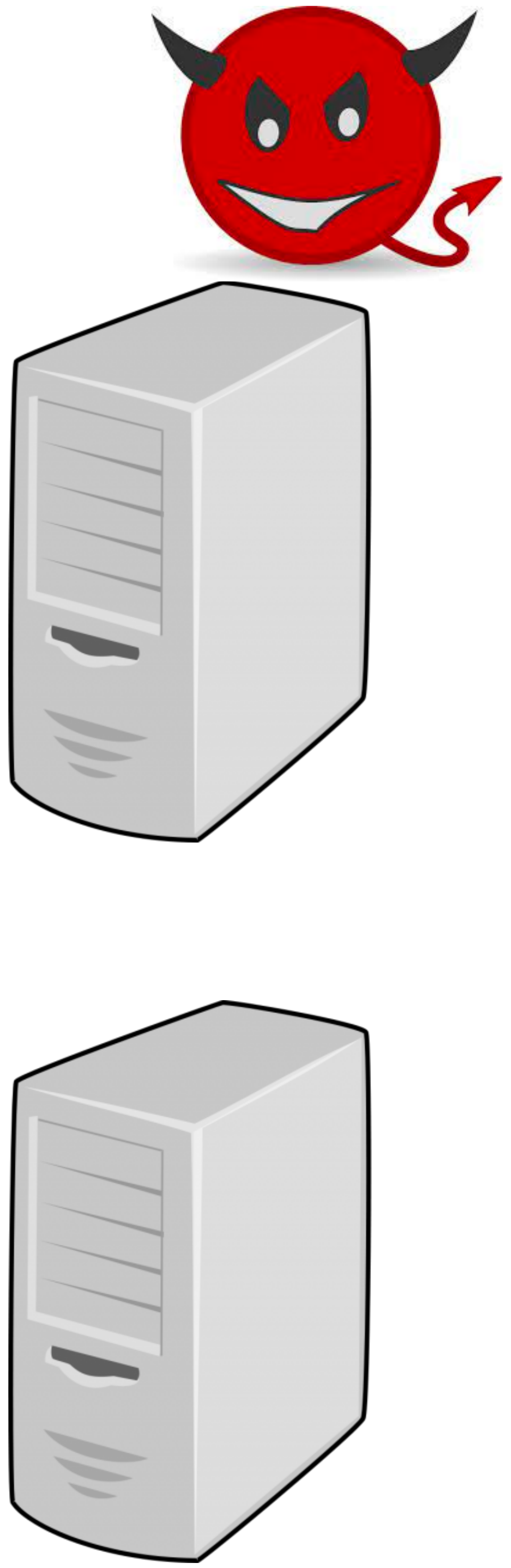
Want to privately query a database with communication *sublinear* in database size

Two models

Multi-server PIR [CGKS'95]

Protects privacy if at least one server is honest

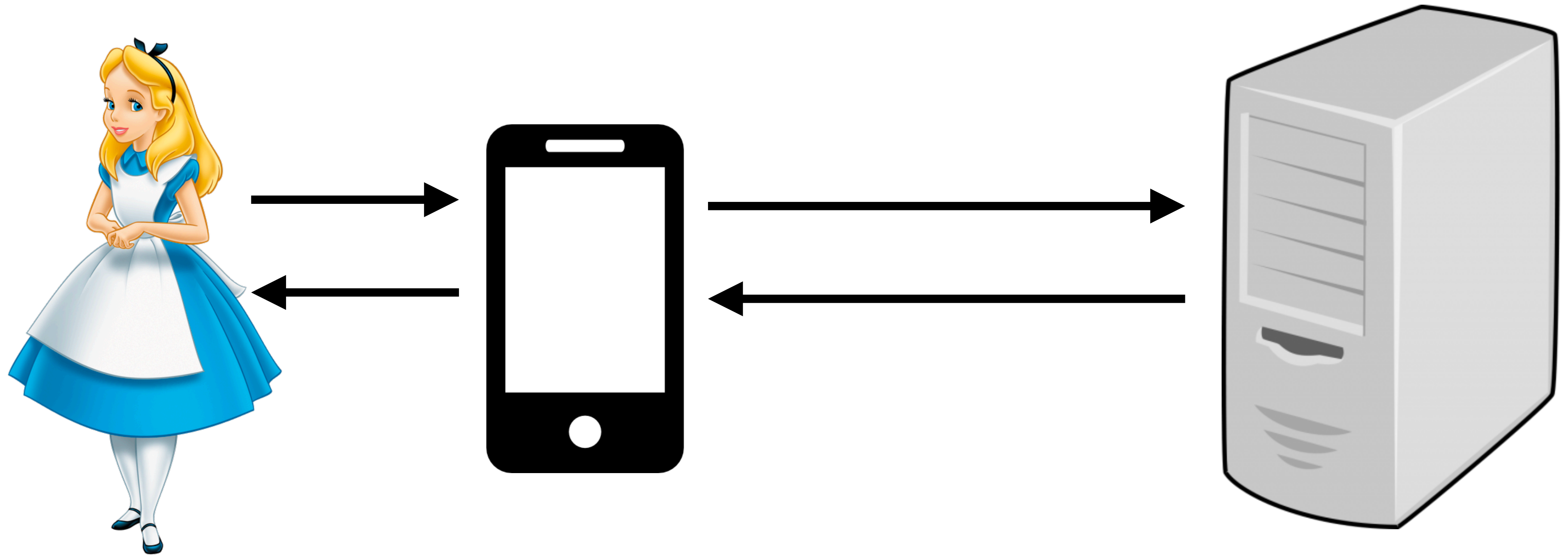




Can be information-theoretic

Single-server PIR [KO'97]

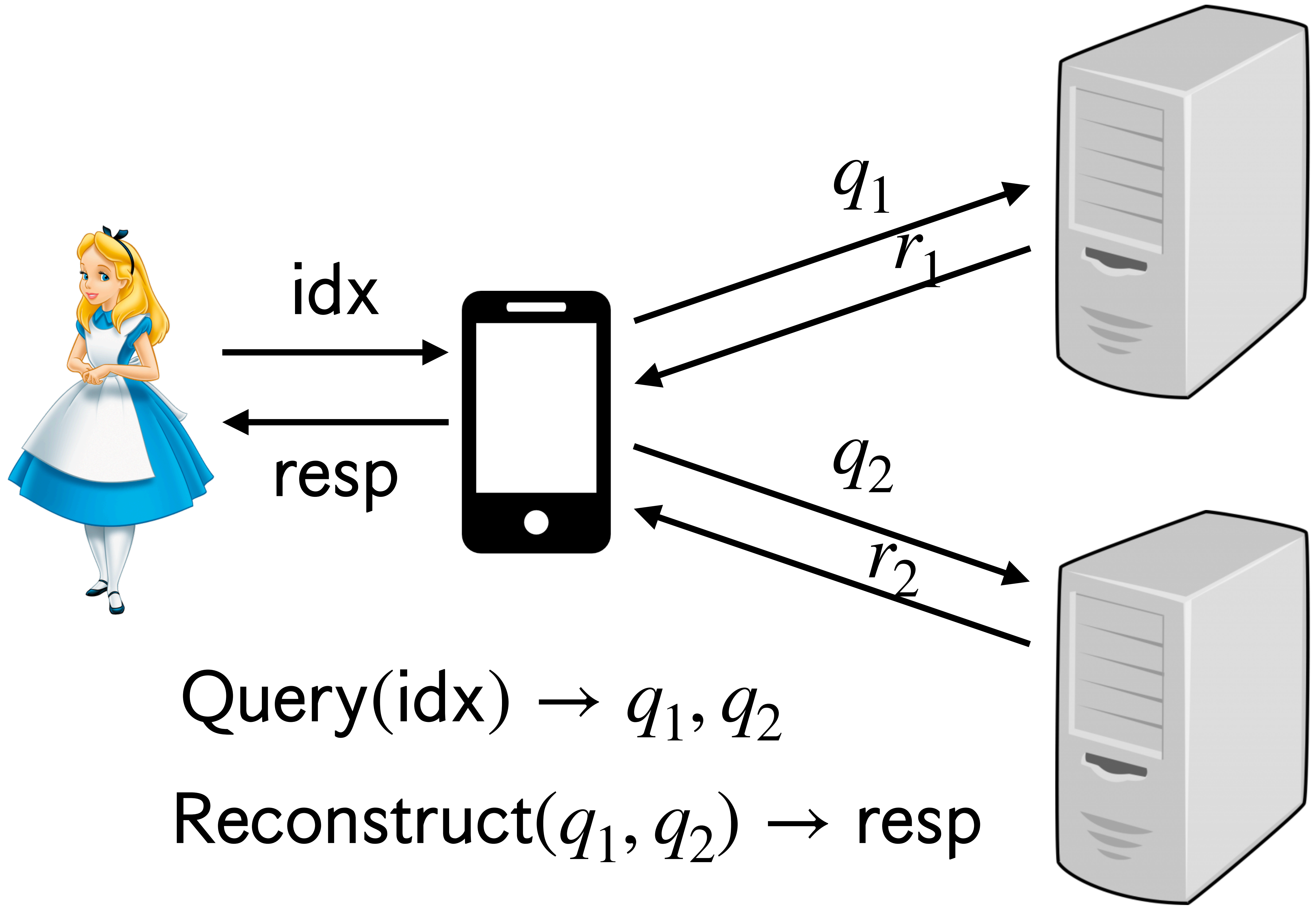
Protects privacy if the server is compromised



Requires a cryptographic assumption

Two-server PIR

$\mathsf{Answer}(\mathsf{db}, q_1) \to r_1$



$\mathsf{Query}(\mathsf{idx}) \to q_1, q_2$

$\mathsf{Reconstruct}(q_1, q_2) \to \mathsf{resp}$

$\mathsf{Answer}(\mathsf{db}, q_2) \to r_2$

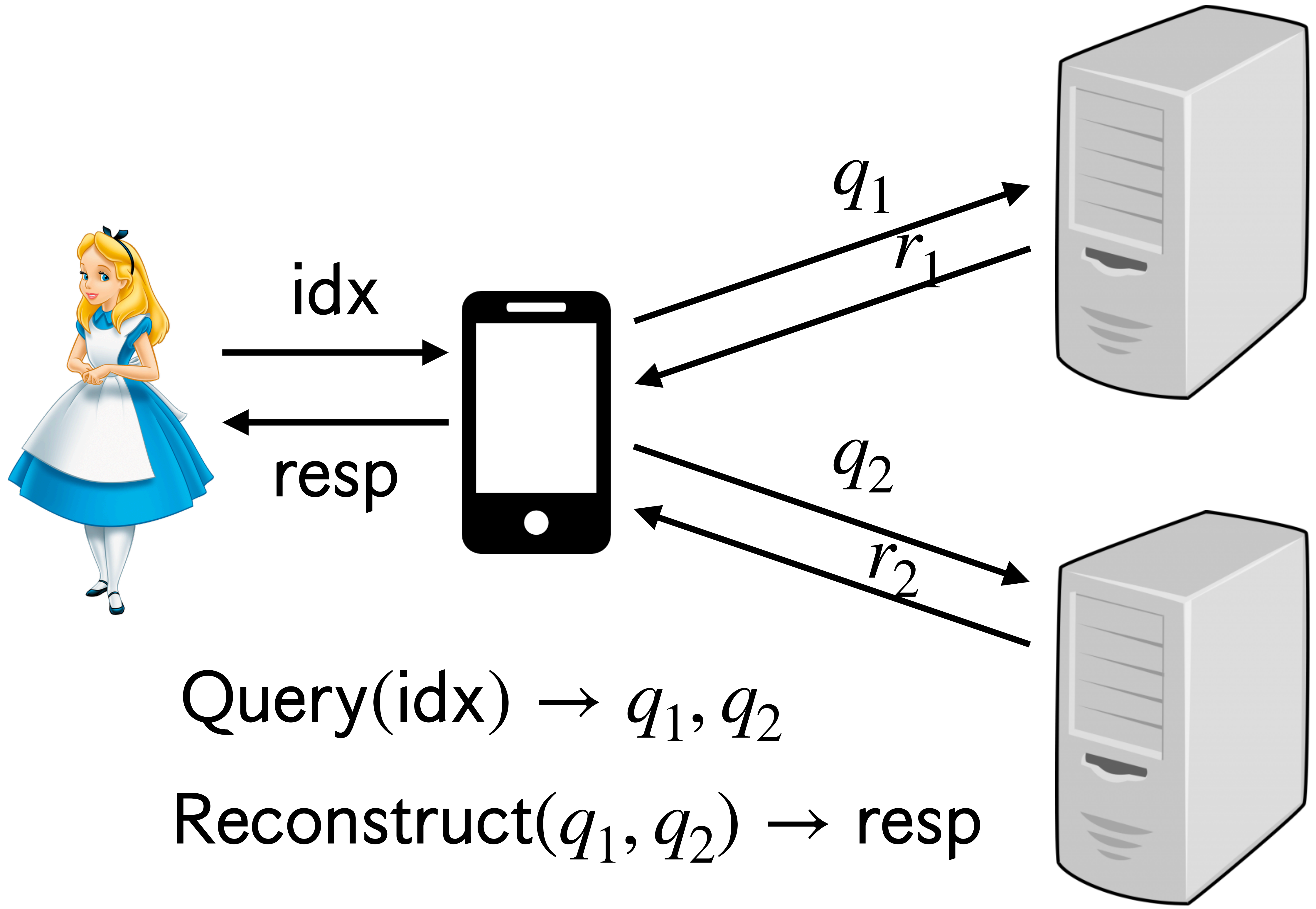
For database with n elements, query space Q, values in $\{0,1\}$, and response space R

$\mathsf{Query} : [n] \to Q^2$
$\mathsf{Answer} : \{0,1\}^n \times Q \to R$
$\mathsf{Reconstruct} : R^2 \to \{0,1\}$

Note: **Query** is randomized and takes security parameter as implicit input

Two-server PIR definitions

$\mathsf{Answer}(\mathsf{db}, q_1) \to r_1$



$\mathsf{Query}(\mathsf{idx}) \to q_1, q_2$

$\mathsf{Reconstruct}(q_1, q_2) \to \mathsf{resp}$

$\mathsf{Answer}(\mathsf{db}, q_2) \to r_2$

***Correctness*: For all $n \in \mathbb{N}$, db $\in \{0,1\}^n$, idx $\in [n]$, the probability that:**

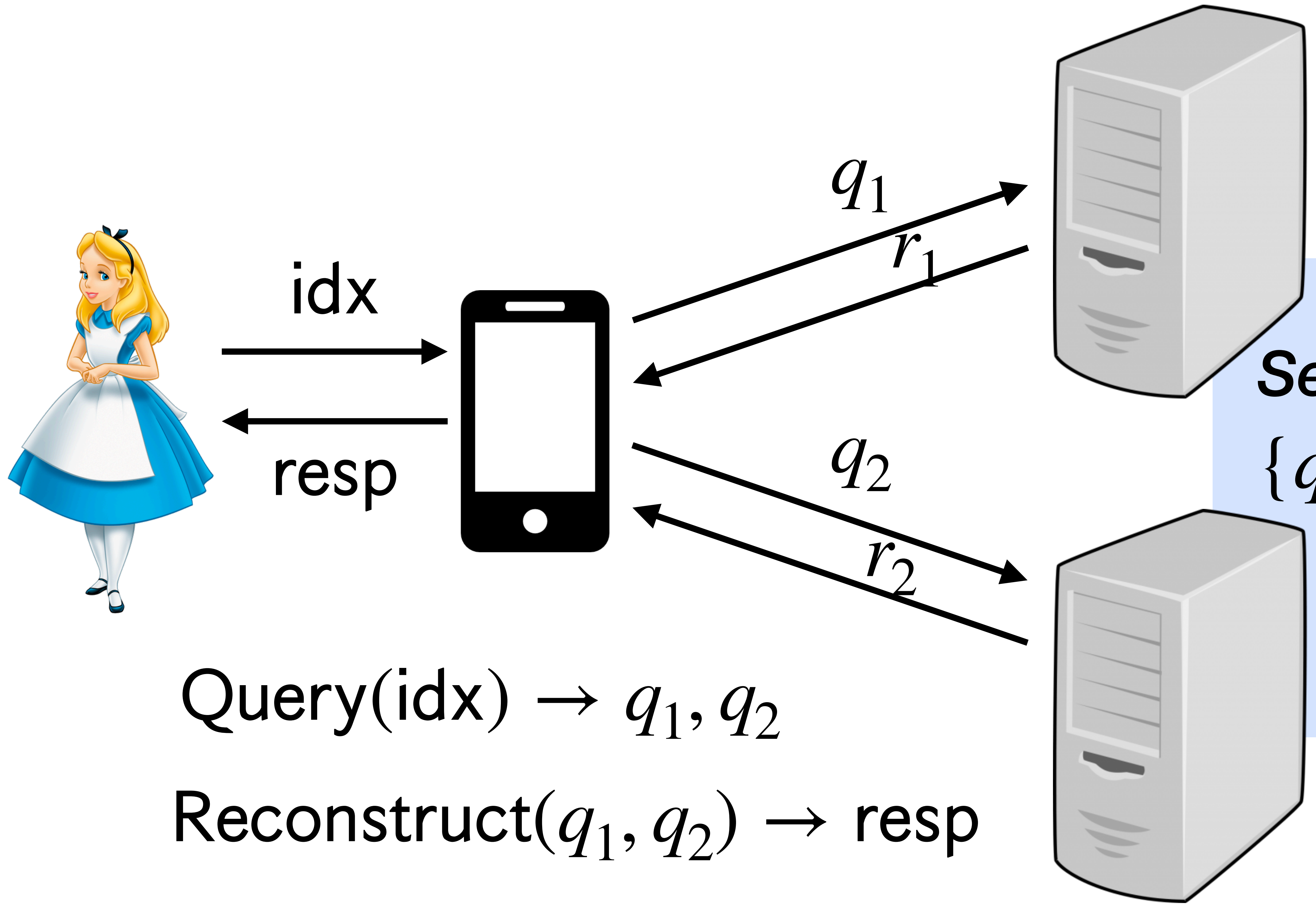
$\mathsf{Query}(\mathsf{idx}) \to q_1, q_2$
$\mathsf{Answer}(db, q_i) \to r_i$ for $i \in \{1,2\}$
$\mathsf{Reconstruct}(q_1, q_2) = \mathsf{db}_{\mathsf{idx}}$

is 1.

Informally, the client gets the requested record

Two-server PIR definitions

Answer(db, q_1) $\rightarrow r_1$



Answer(db, q_2) $\rightarrow r_2$

Security: **For all** $i, j \in [n], b \in \{1,2\}$

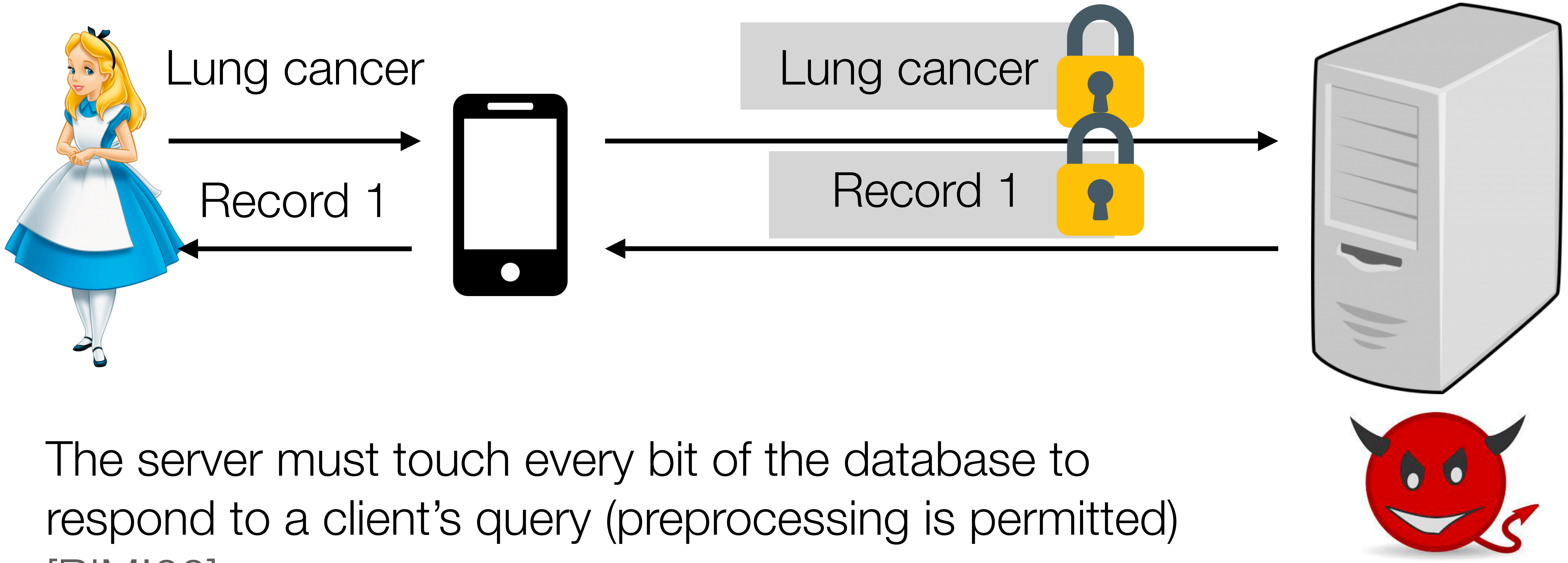
$\{q_b : \text{Query}(i) \rightarrow q_1, q_2\} \approx \{q'_b : \text{Query}(j) \rightarrow q'_1, q'_2\}$

Informally, no one server can learn the client's query

What we can hope for with PIR

Is it possible to build a PIR scheme that only touches part of the database?

No! If query execution only touches part of the database, then an attacker can learn which part of the database is *not* accessed by the query



Heart disease	Record 1
Lung cancer	Record 2
Diabetes	Record 3
…	…

The server must touch every bit of the database to respond to a client's query (preprocessing is permitted) [BIM'00]

ORAM vs PIR

ORAM

- One client, one server
- Private reads + writes
- Database is hidden from the server
- Stateful client
- Stateful server
- Constructions with polylogarithmic overheads

PIR

- Many clients, one server
- Private reads (no private writes)
- Database is visible to the server
- Traditionally no stateful client (some recent work leveraging stateful client)
- Server state does not change (after possible preprocessing)
- Must touch every bit of database

Outline

1. Overview
2. **Two-server PIR construction**
3. Single-server PIR construction
4. PIR with preprocessing overview

Background: secret sharing

Split a value $x \in \{0,1\}$ into secret shares $[x]_1 \in \{0,1\}, [x]_2 \in \{0,1\}$ such that
$[x]_1 + [x]_2 \mod 2 = x$

Information-theoretic privacy: Given just $[x]_b$ for $b \in \{1,2\}$, adversary learns no information about x

Operations (all mod 2):

- Given x, generate secret shares by randomly sampling $[x]_1$ and setting $[x]_2 = x - [x]_1$
- Given $[x]_1, [x]_2$, reconstruct x by computing $[x]_1 + [x]_2 = x$

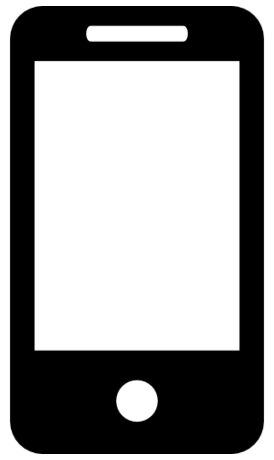
Can extend beyond a single bit (e.g., supports bitstrings, fields, rings)

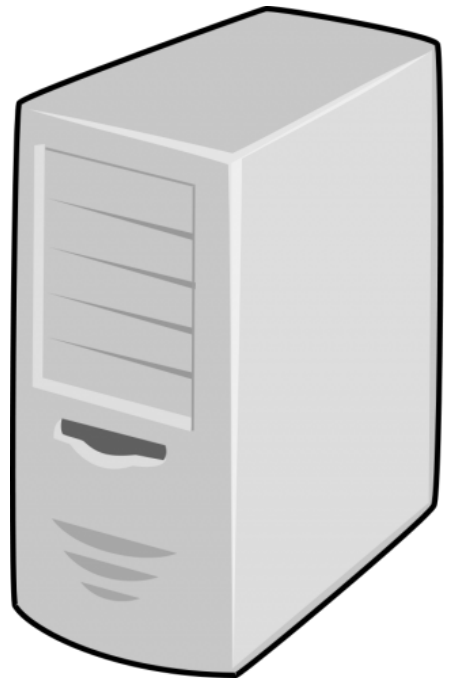
Computing on secret shares:

- Can add secret shares: $[x] + [y] = [x + y]$
- Can multiply by a constant: $c \cdot [x] = [c \cdot x]$ (by extension)

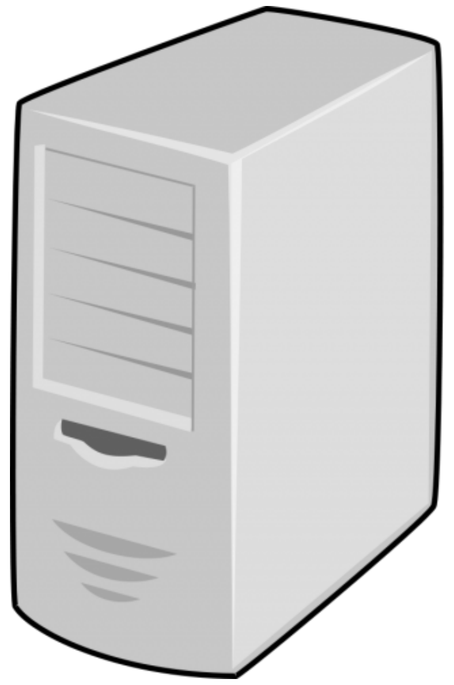
2-server PIR scheme

Write database as $\sqrt{n} \times \sqrt{n}$ matrix





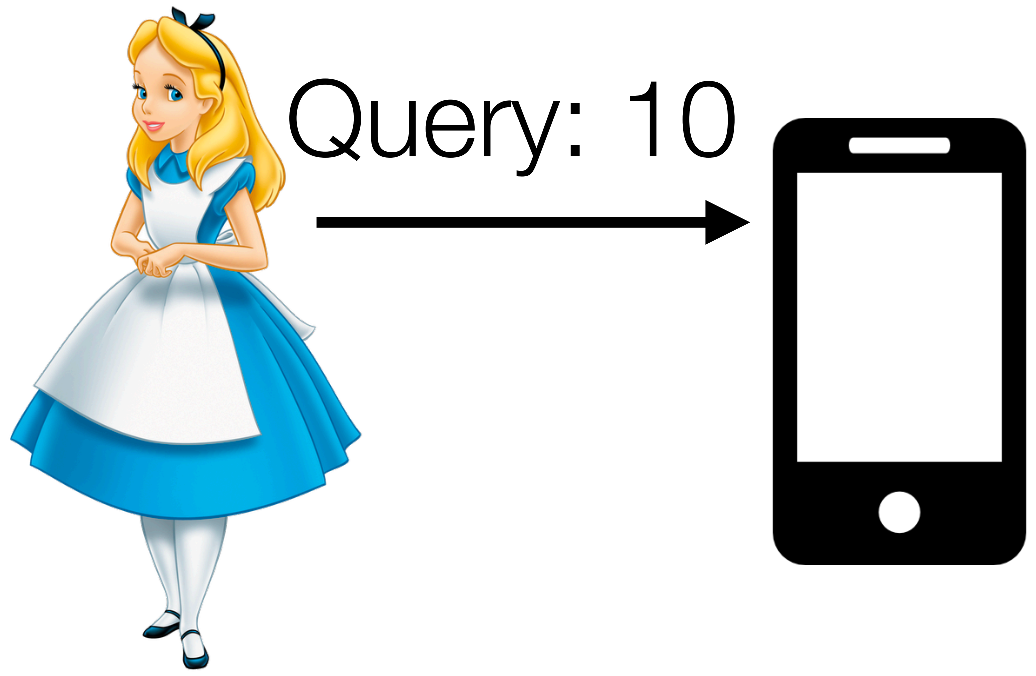
$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix}$$

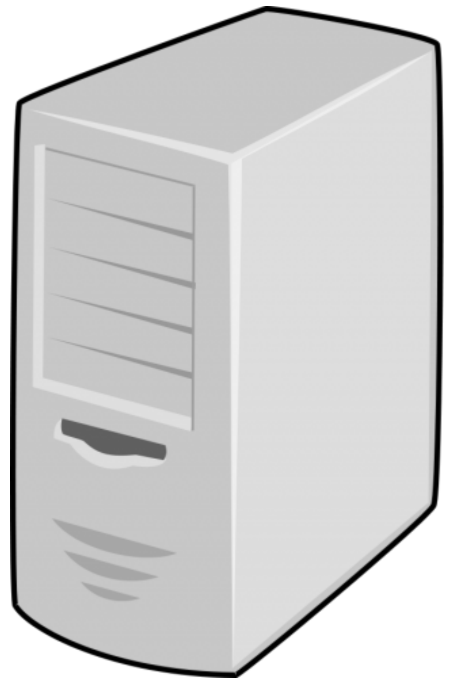


$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix}$$

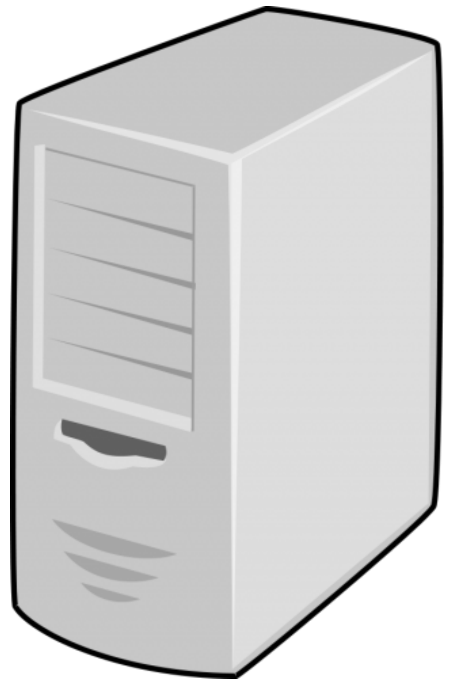
2-server PIR scheme

Write database as $\sqrt{n} \times \sqrt{n}$ matrix





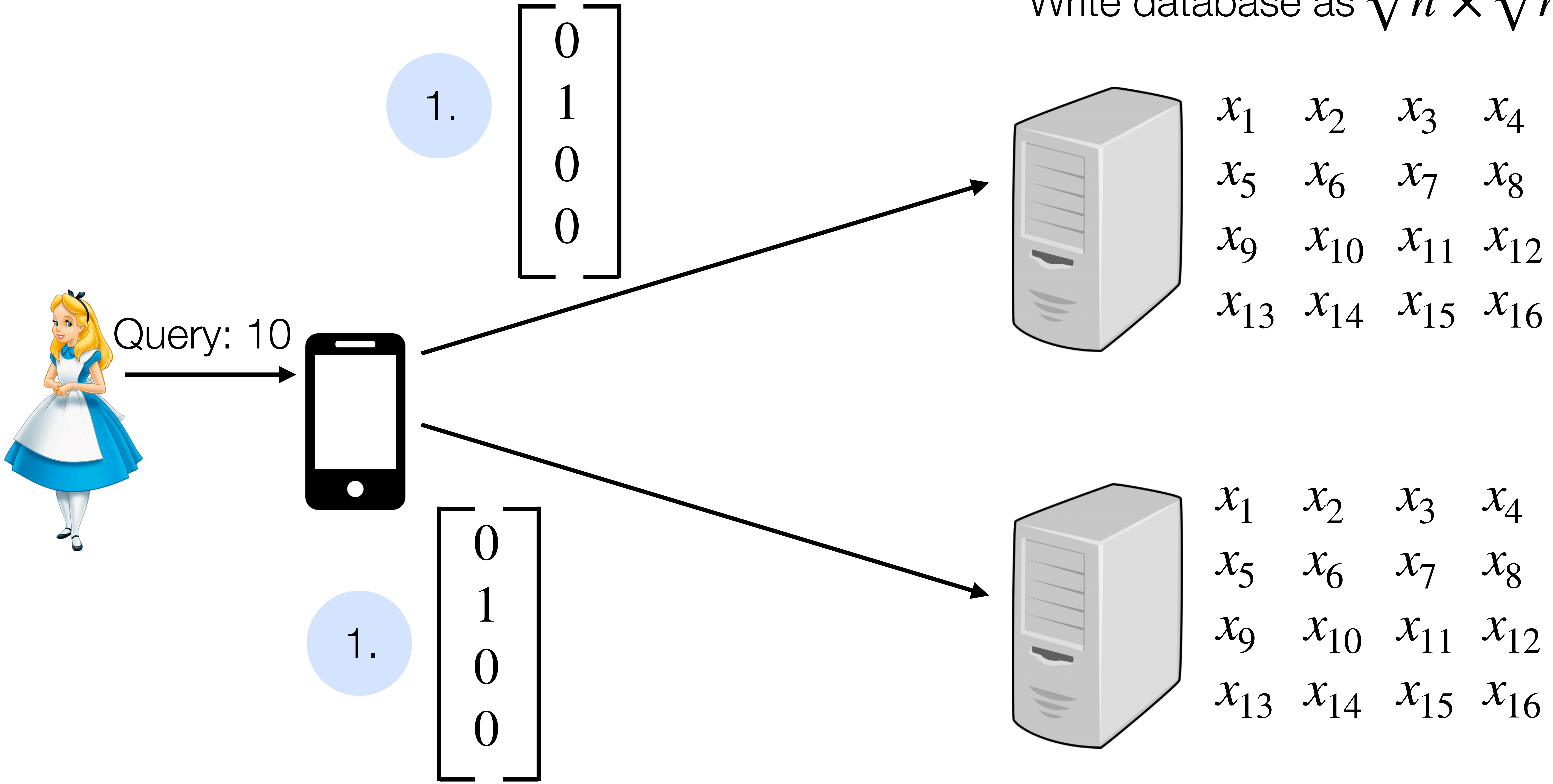
$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix}$$



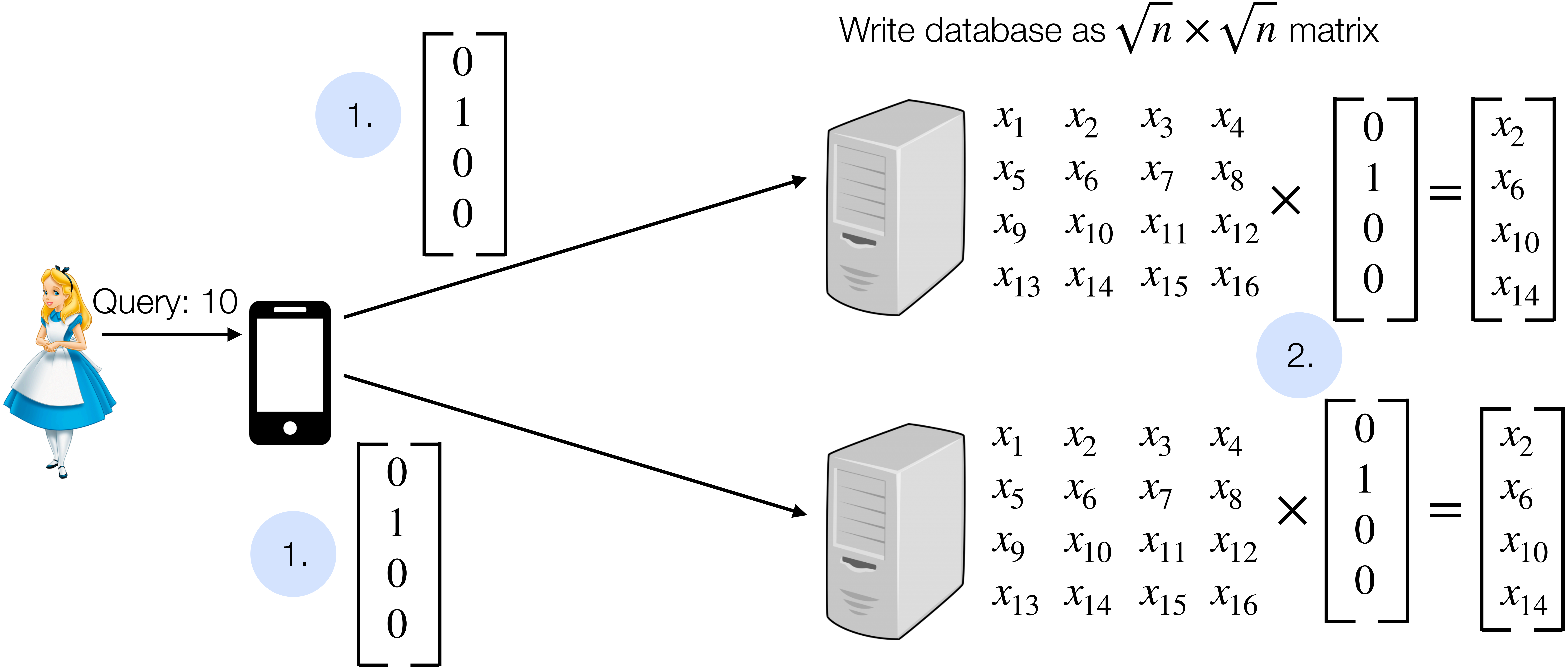
$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix}$$

2-server PIR scheme

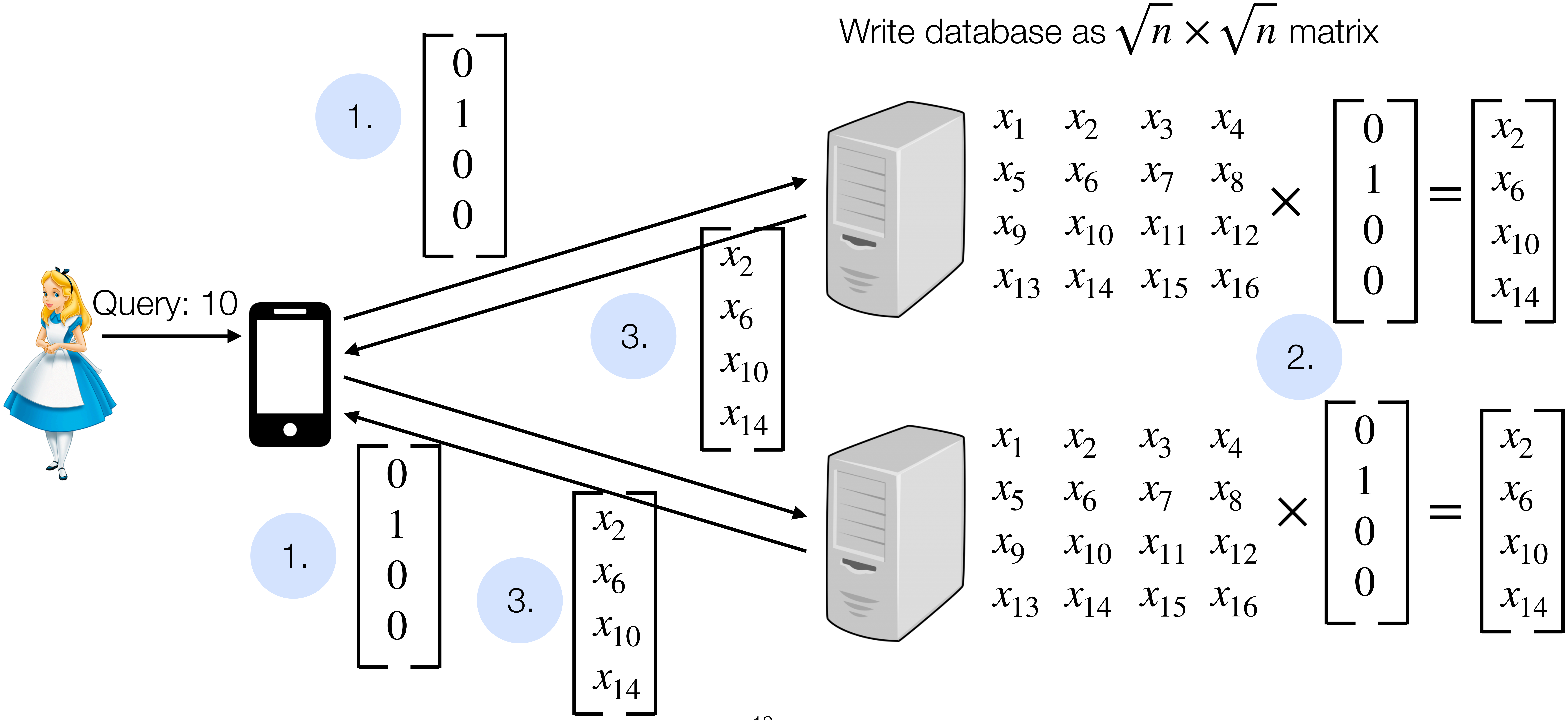
Write database as $\sqrt{n} \times \sqrt{n}$ matrix



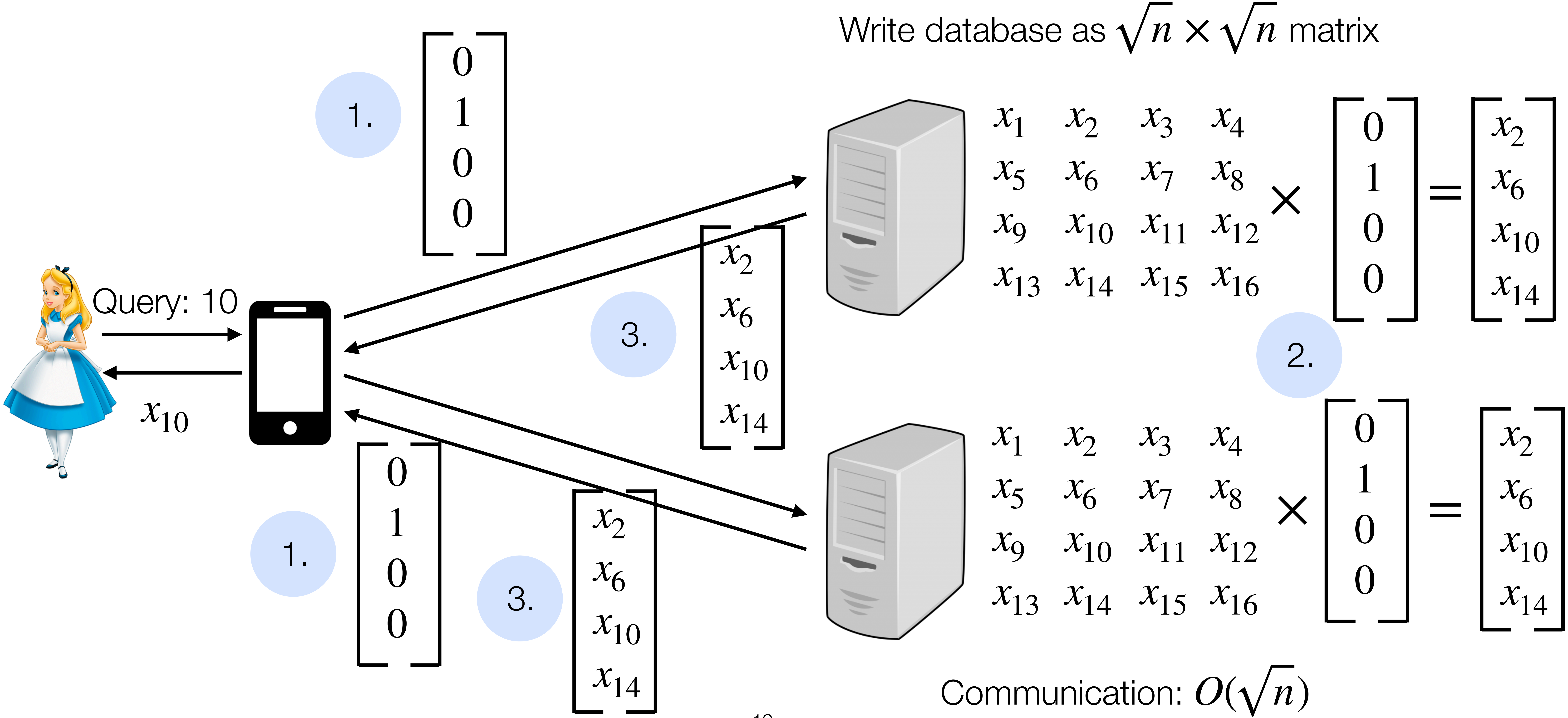
2-server PIR scheme



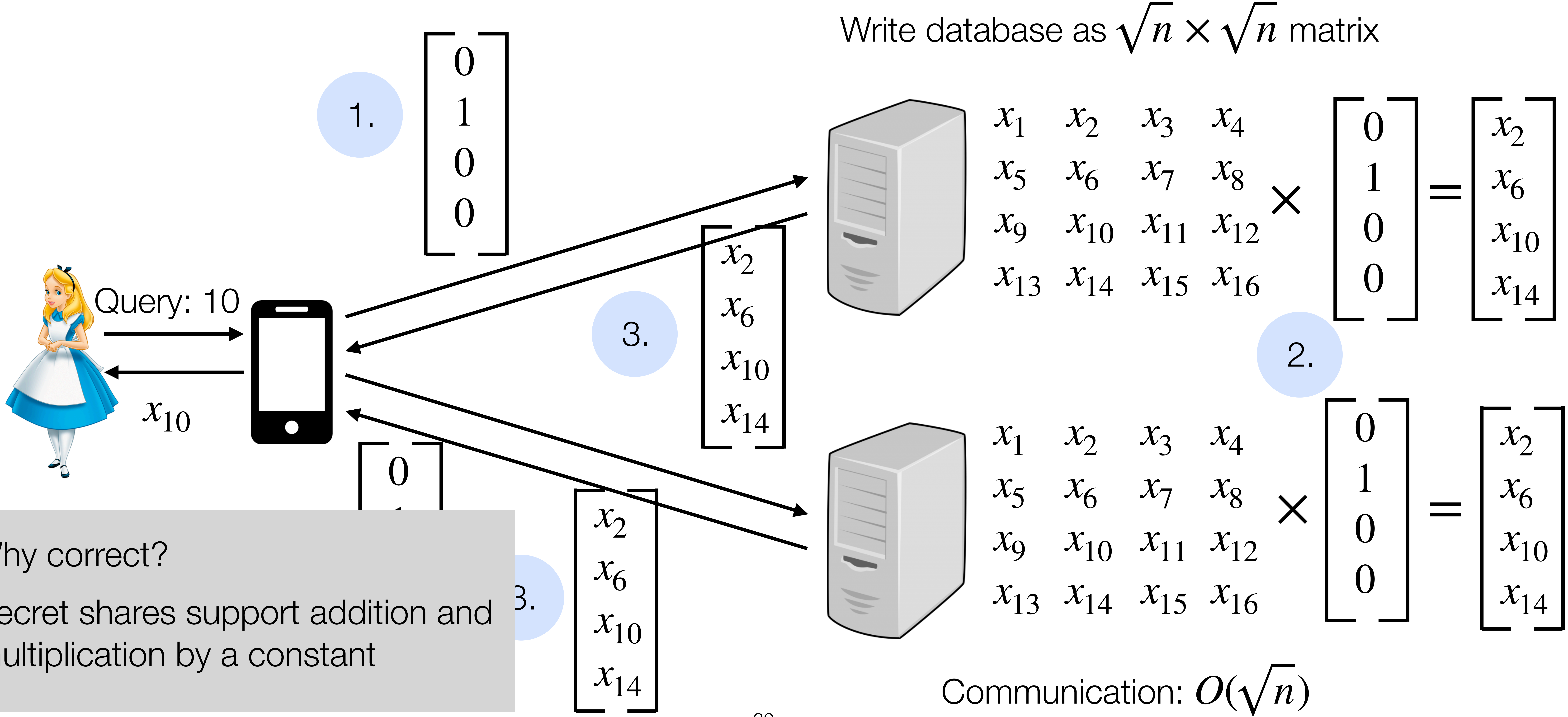
2-server PIR scheme



2-server PIR scheme



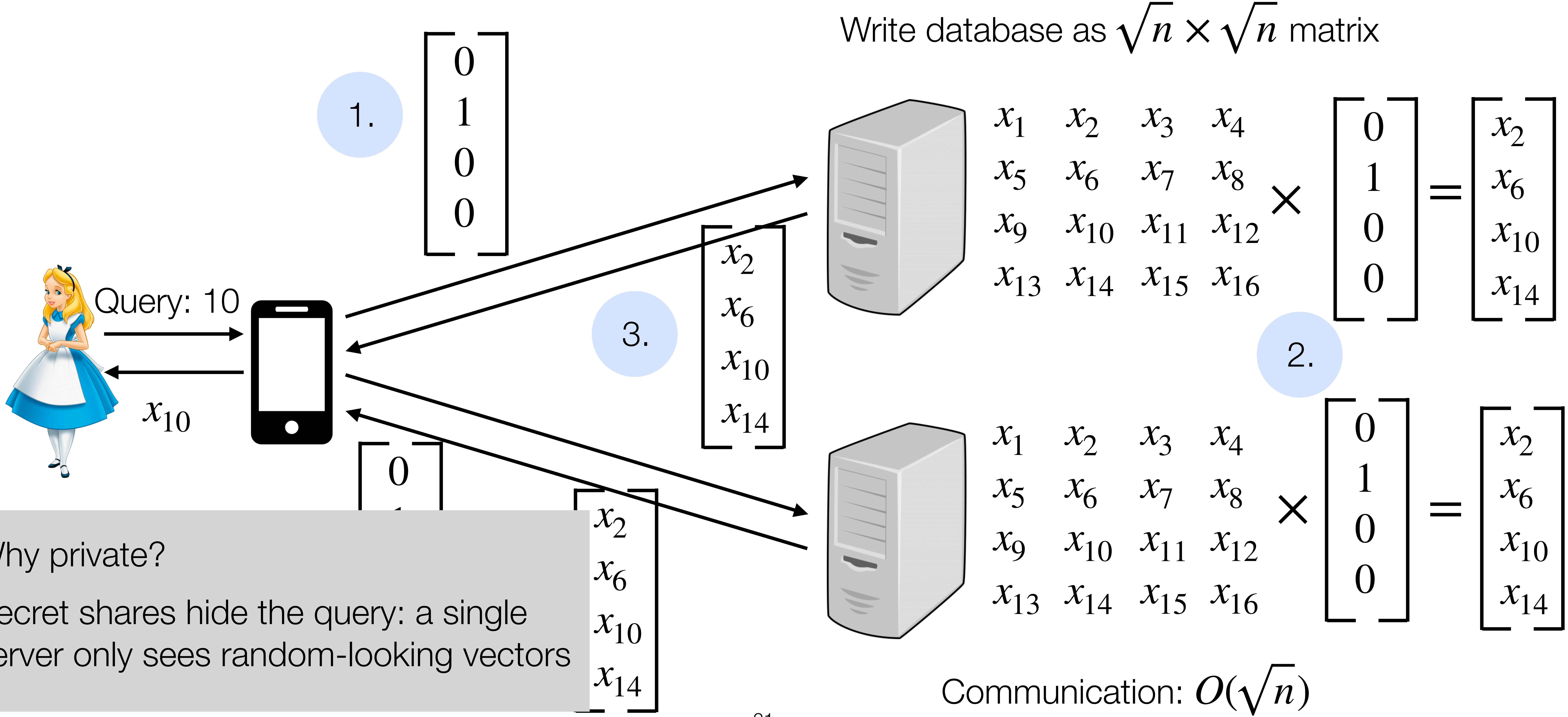
2-server PIR scheme



Why correct?

Secret shares support addition and multiplication by a constant

2-server PIR scheme

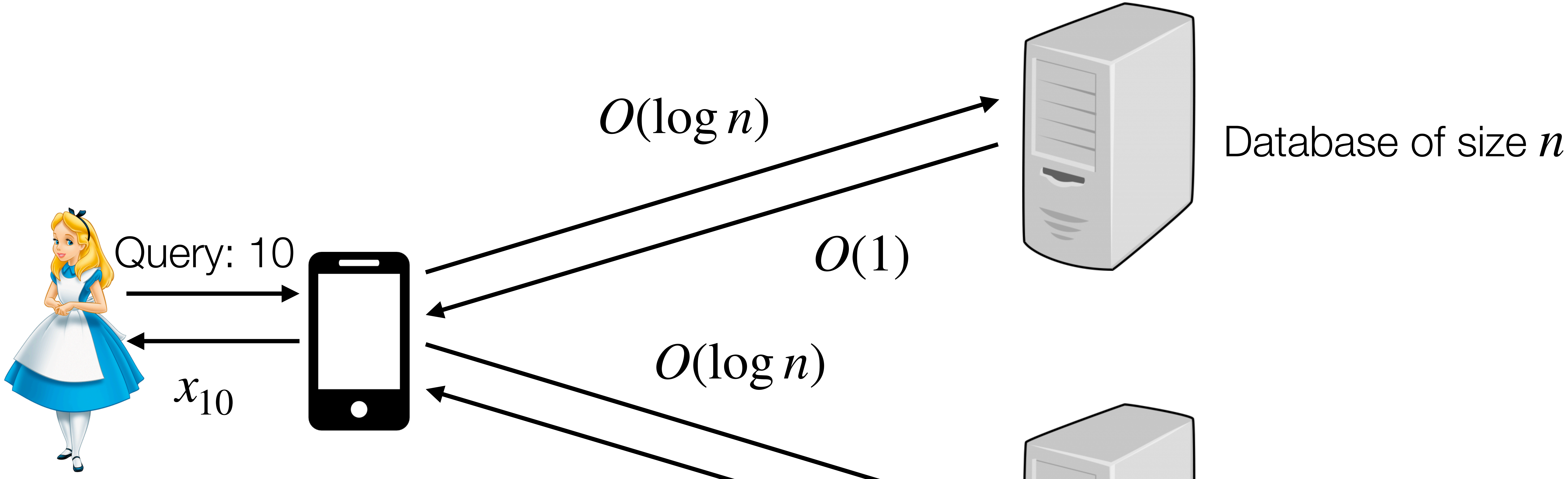


Why private?

Secret shares hide the query: a single server only sees random-looking vectors

Reducing bandwidth in two-server PIR

Tool: Distributed point functions [GI14] (more next class)



Outline

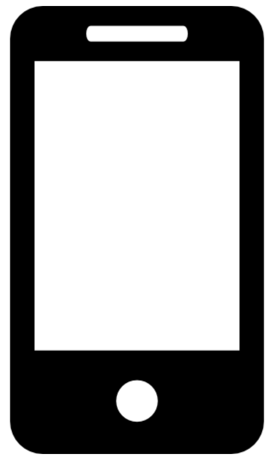
1. Overview
2. Two-server PIR construction
3. **Single-server PIR construction**
4. PIR with preprocessing overview

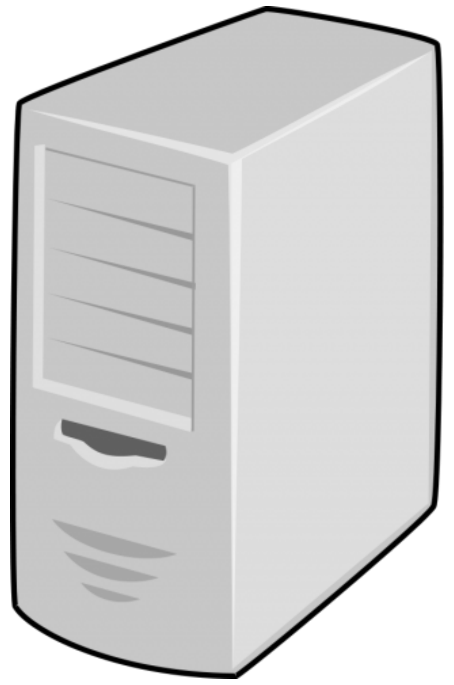
Background: additively homomorphic encryption

Encryption scheme that supports:

- Adding ciphertexts: $\mathsf{Enc}_k(x) + \mathsf{Enc}_k(y) = \mathsf{Enc}_k(x + y)$
- Multiplication by a constant (by extension): $c \cdot \mathsf{Enc}_k(x) = \mathsf{Enc}_k(c \cdot x)$

Single-server PIR scheme

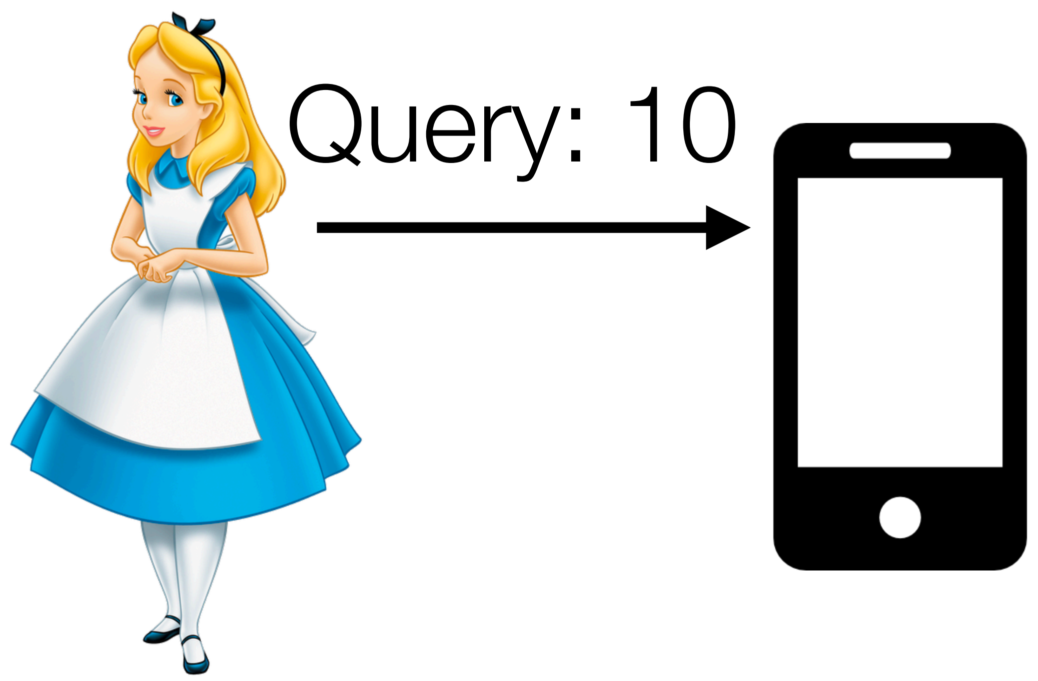


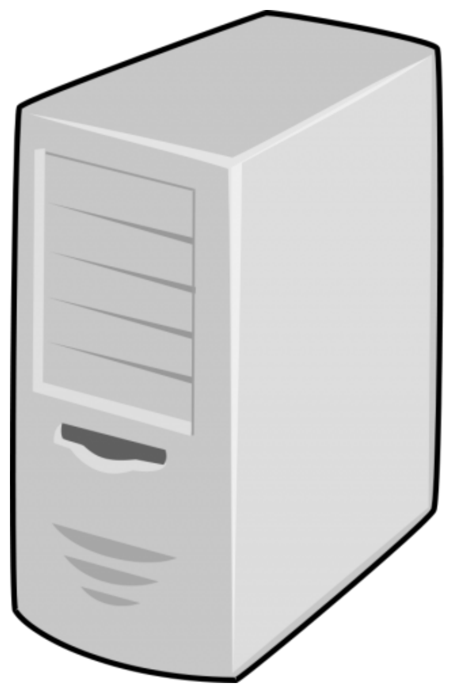


Write database as $\sqrt{n} \times \sqrt{n}$ matrix

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix}$$

Single-server PIR scheme





Write database as $\sqrt{n} \times \sqrt{n}$ matrix

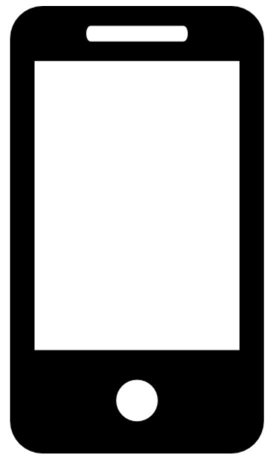
$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{array}$$

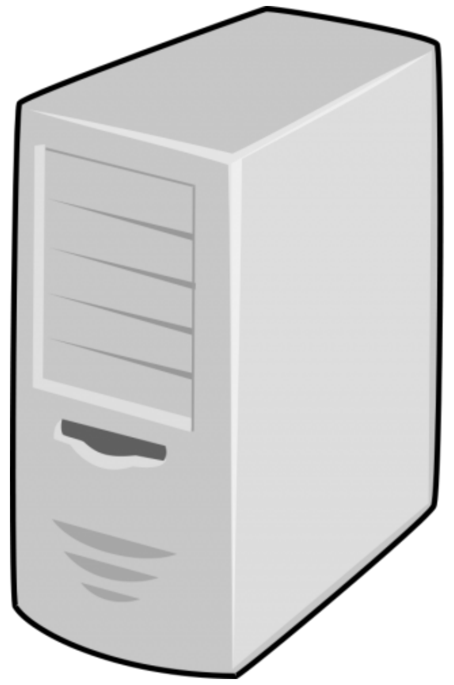
Single-server PIR scheme

1.

$$\begin{bmatrix} \text{Enc}_k(0) \\ \text{Enc}_k(1) \\ \text{Enc}_k(0) \\ \text{Enc}_k(0) \end{bmatrix}$$

Query: 10





Write database as $\sqrt{n} \times \sqrt{n}$ matrix

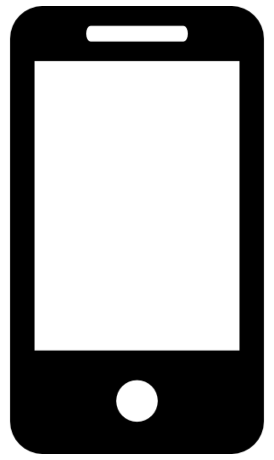
$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix}$$

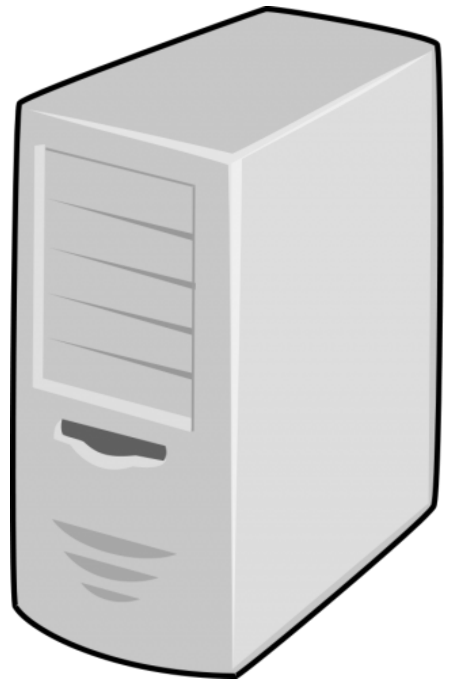
Single-server PIR scheme

1.

$$\begin{bmatrix} \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(1) \\ \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(0) \end{bmatrix}$$

Query: 10





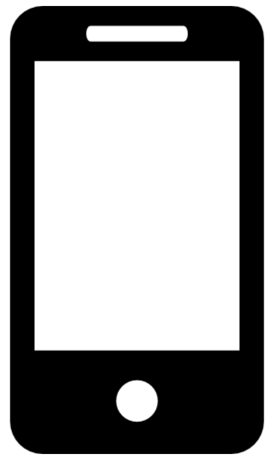
Write database as $\sqrt{n} \times \sqrt{n}$ matrix

2.

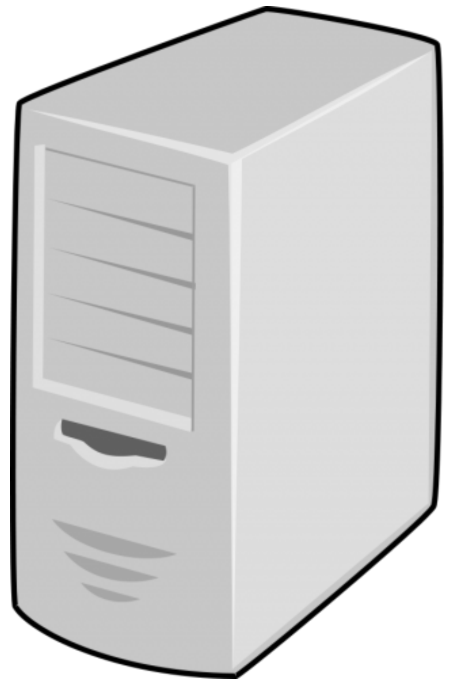
$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix} \times \begin{bmatrix} \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(1) \\ \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(0) \end{bmatrix} = \begin{bmatrix} \mathsf{Enc}_k(x_2) \\ \mathsf{Enc}_k(x_6) \\ \mathsf{Enc}_k(x_{10}) \\ \mathsf{Enc}_k(x_{14}) \end{bmatrix}$$

Single-server PIR scheme

Query: 10



1. $\begin{bmatrix} \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(1) \\ \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(0) \end{bmatrix}$

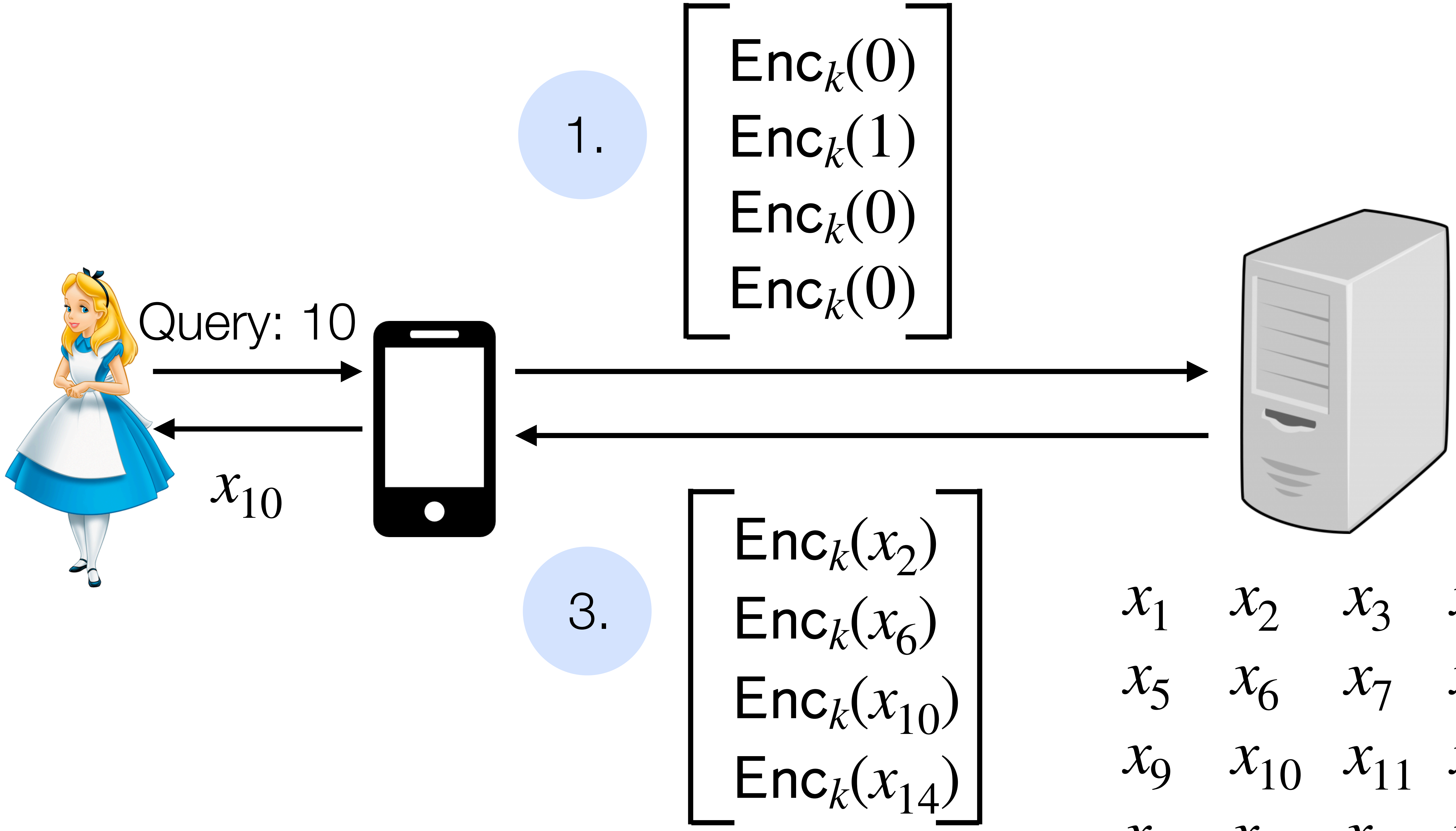


Write database as $\sqrt{n} \times \sqrt{n}$ matrix

2. $$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix} \times \begin{bmatrix} \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(1) \\ \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(0) \end{bmatrix} = \begin{bmatrix} \mathsf{Enc}_k(x_2) \\ \mathsf{Enc}_k(x_6) \\ \mathsf{Enc}_k(x_{10}) \\ \mathsf{Enc}_k(x_{14}) \end{bmatrix}$$

3. $\begin{bmatrix} \mathsf{Enc}_k(x_2) \\ \mathsf{Enc}_k(x_6) \\ \mathsf{Enc}_k(x_{10}) \\ \mathsf{Enc}_k(x_{14}) \end{bmatrix}$

Single-server PIR scheme



Write database as $\sqrt{n} \times \sqrt{n}$ matrix

2.

$$\begin{matrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{matrix} \times \begin{bmatrix} \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(1) \\ \mathsf{Enc}_k(0) \\ \mathsf{Enc}_k(0) \end{bmatrix} = \begin{bmatrix} \mathsf{Enc}_k(x_2) \\ \mathsf{Enc}_k(x_6) \\ \mathsf{Enc}_k(x_{10}) \\ \mathsf{Enc}_k(x_{14}) \end{bmatrix}$$

Outline

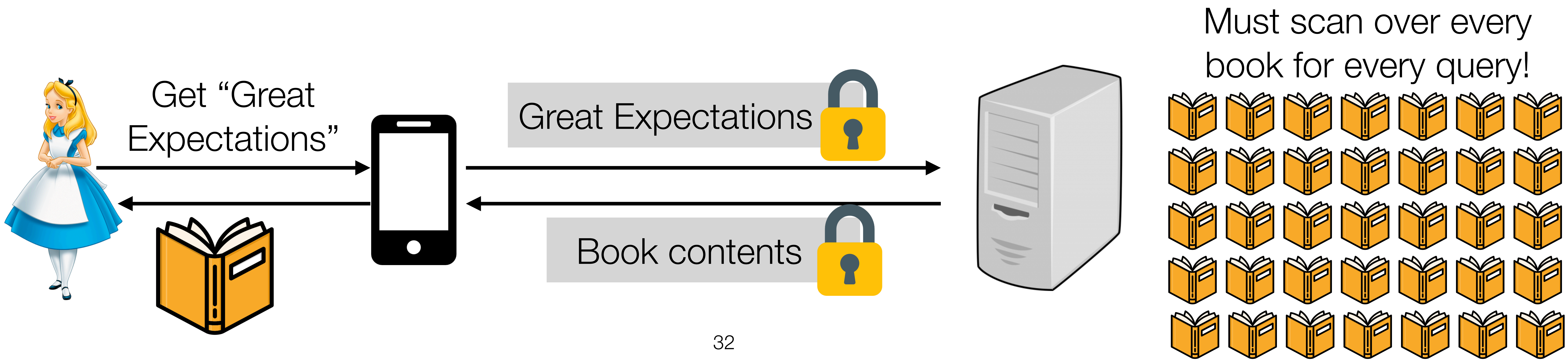
1. Overview
2. Two-server PIR construction
3. Single-server PIR construction
4. **PIR with preprocessing overview**

PIR lower bound

Recall: The server must touch every bit of the database to respond to a client's query [BIM'00]

- Otherwise the server can learn information about what the client's query could *not* have been via which parts of the database are *not* accessed

Limitation to deployment: scaling to large datasets is challenging



PIR lower bound

Recall: The server must touch every bit of the database to respond to a client's query [BIM'00]

- Otherwise the server can learn information about what the client's query could *not* have been via which parts of the database are *not* accessed

Limitation to deployment: scaling to large datasets is challenging

Opportunity: Preprocessing

- Preprocess the data and then serve many user queries

PIR with preprocessing

Idea: push expensive, linear scan to a step that runs before the client submits its query

Two flavors that we'll talk about:

- Two-server online-offline private information retrieval
- Doubly efficient private information retrieval

(Others that we won't have time to talk about)

PIR with preprocessing

Idea: push expensive, linear scan to a step that runs before the client submits its query

Two flavors that we'll talk about:

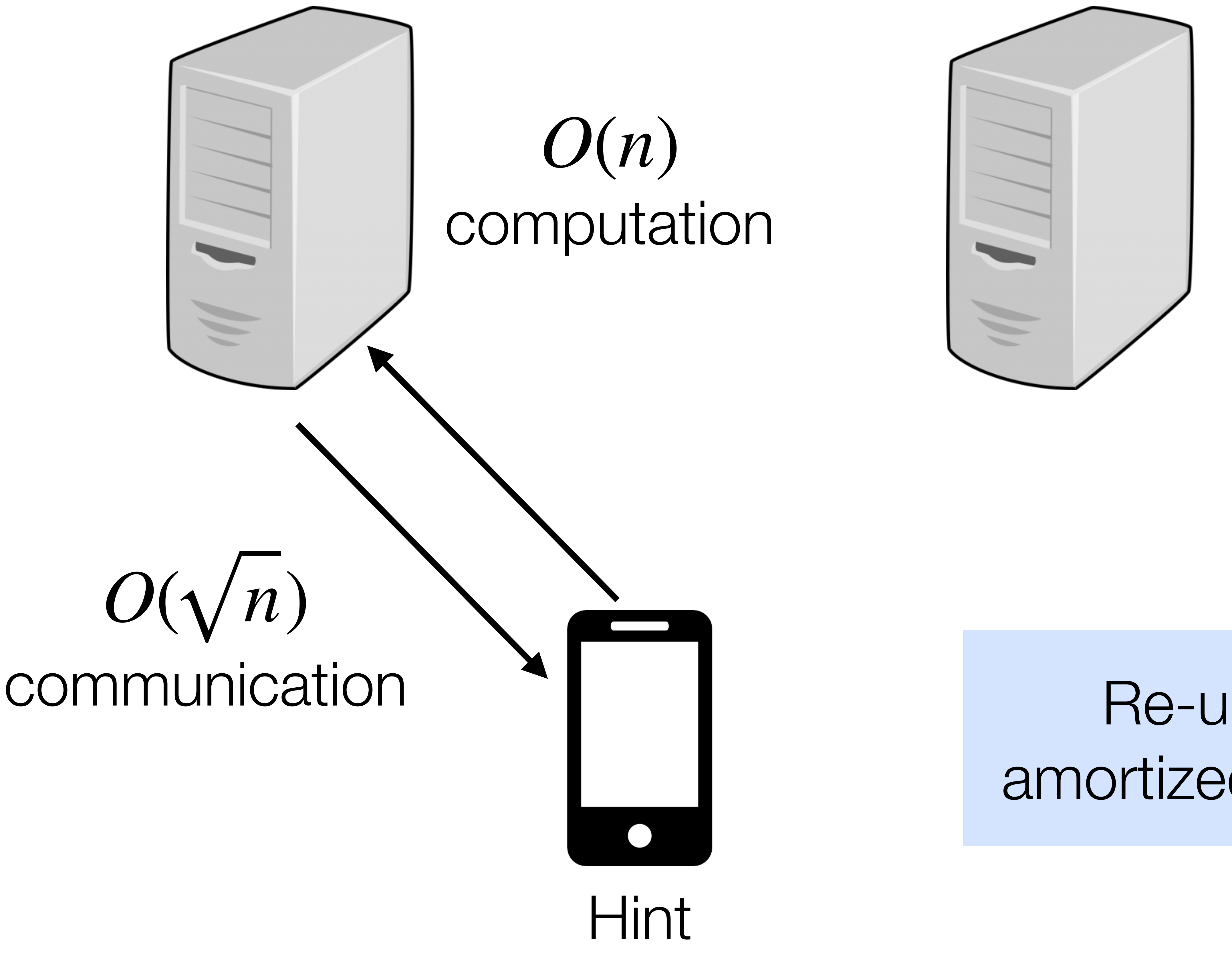
- **Two-server online-offline private information retrieval**
- Doubly efficient private information retrieval

(Others that we won't have time to talk about)

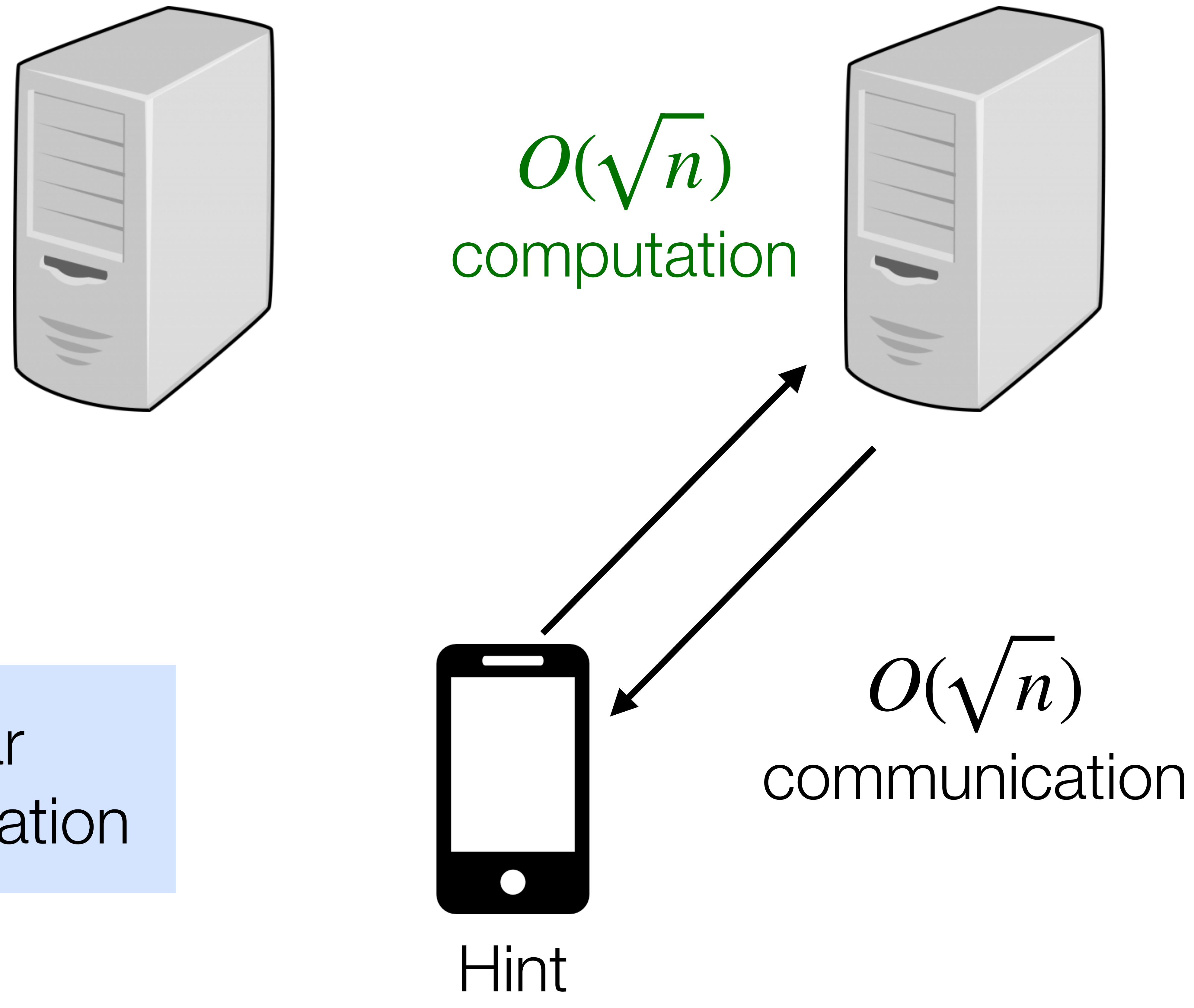
Two-server online-offline PIR [CK'20]

Database size n

Offline phase (before query time)



Online phase (at query time)

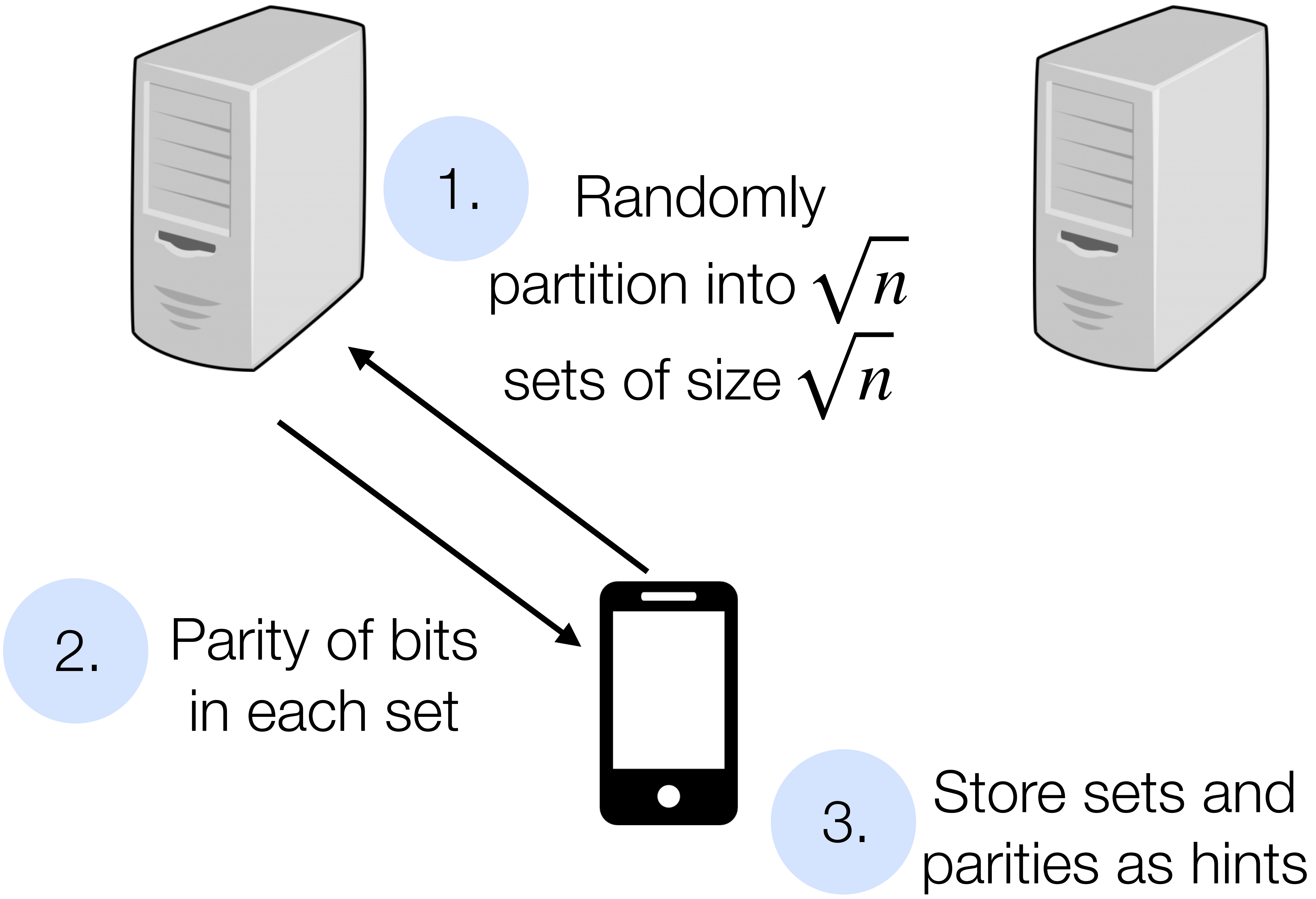


Re-use hint: sublinear amortized query computation

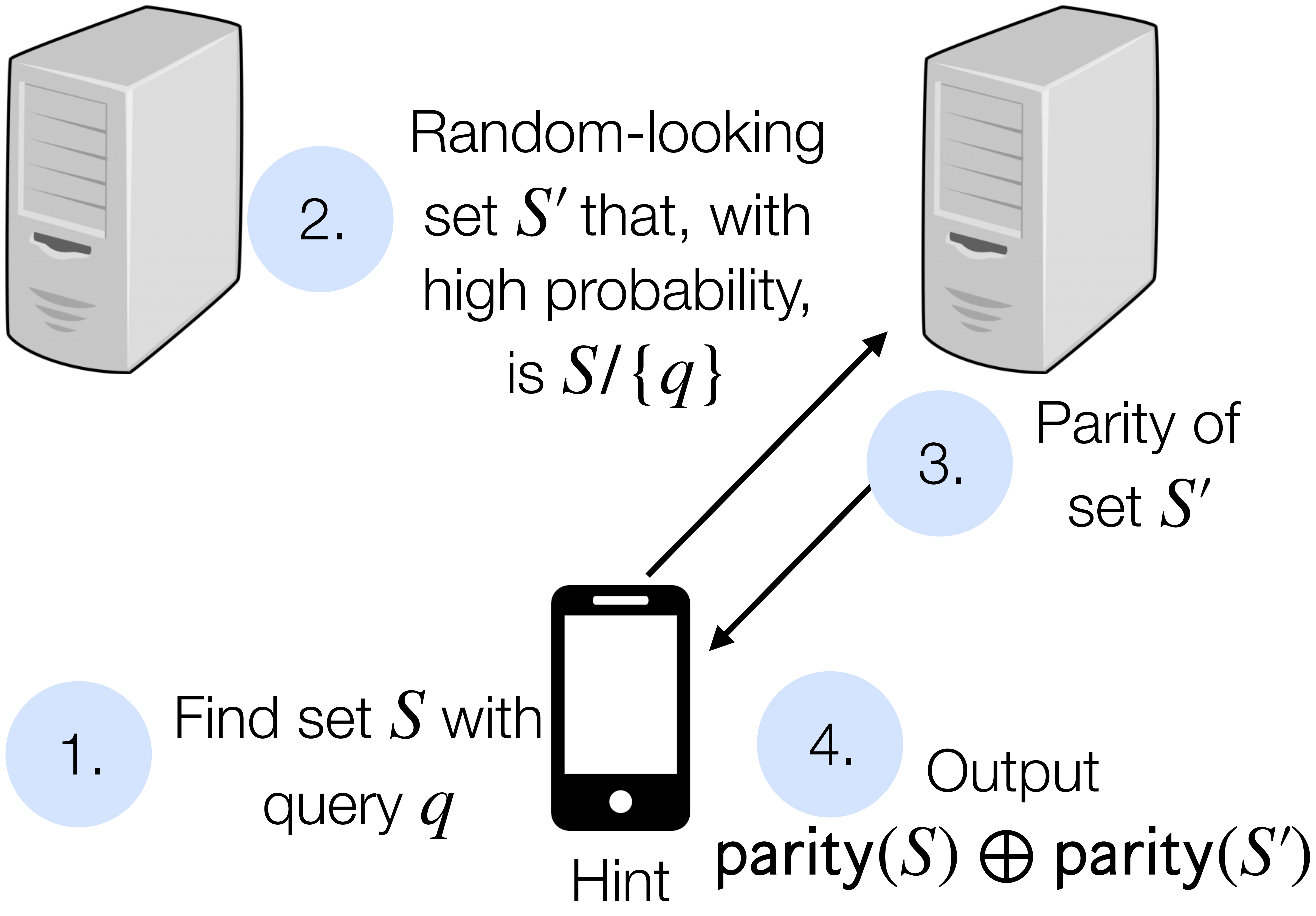
Two-server online-offline PIR (simplified) [CK'20]

Database size n

Offline phase (before query time)



Online phase (at query time)



Security [CK'20]

Database size n

Two-server online-offline PIR [CK'20]

Need a mechanism for compressing random sets:

- Puncturable pseudorandom sets: minimize offline communication + hint size

Refreshing the client hint:

- One of the sets is used for a query — how to handle future queries?
- Idea: Fetch another random subset and combine it with query results
- See paper for how to refresh hint

Extension: single-server PIR using additively homomorphic encryption

PIR with preprocessing

Idea: push expensive, linear scan to a step that runs before the client submits its query

Two flavors that we'll talk about:

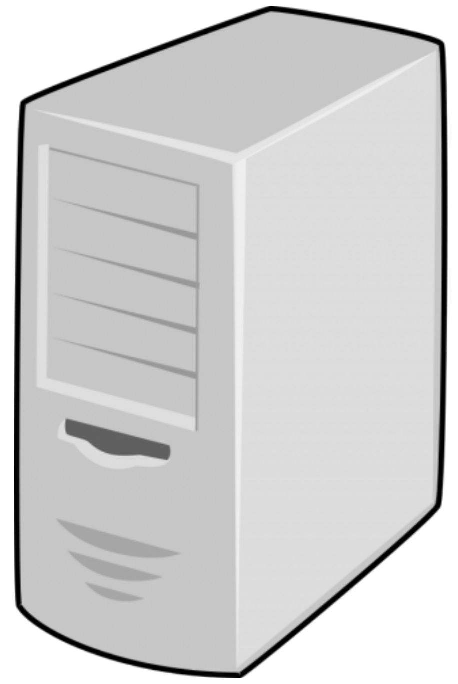
- Two-server online-offline private information retrieval
- **Doubly efficient private information retrieval**

(Others that we won't have time to talk about)

Doubly efficient PIR [LMW'23]

Preprocessing phase

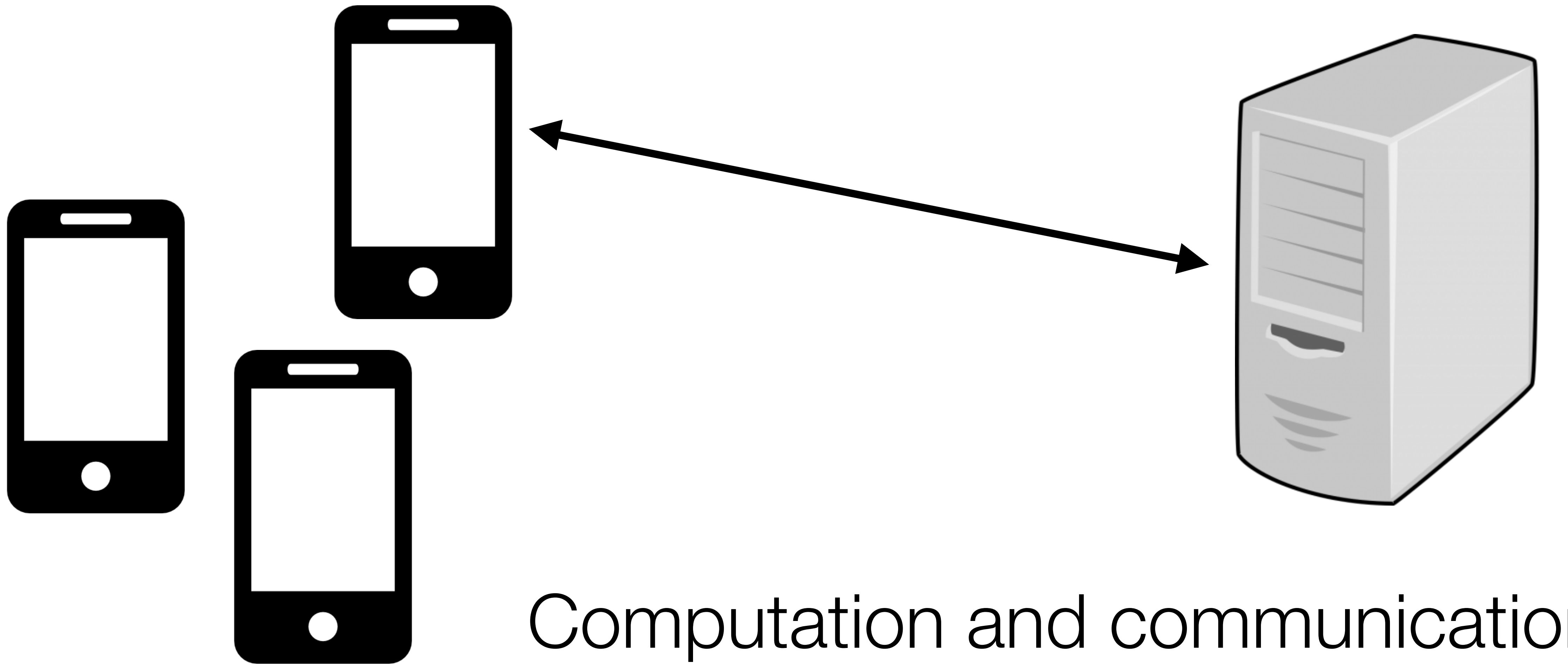
Database size n



Preprocess(db) → $\tilde{\text{db}}$

Computation $O(n^{1+\epsilon})$

Online phase



$\tilde{\text{db}}$

Computation and communication $\textbf{polylog}(n)$

High-level approach [LMW'23]

Two ingredients:

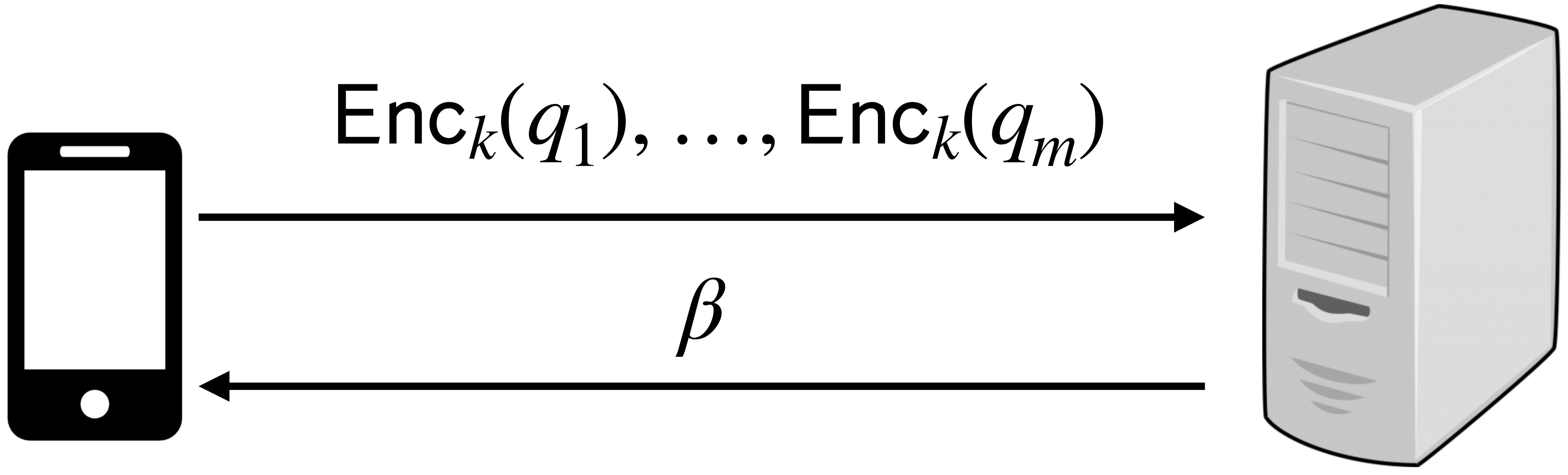
- PIR scheme from encryption scheme that can evaluate low-degree functions
- Preprocessing polynomial evaluation [Kedlaya-Umans'08]

High-level approach [LMW'23]

Two ingredients:

- *PIR scheme from encryption scheme that can evaluate low-degree functions*
- Preprocessing polynomial evaluation [Kedlaya-Umans'08]

Database size n, $m = \log_d(n)$



Encode database as f_{DB}

where $f_{\mathsf{DB}}(i_1, \ldots, i_m) = \mathsf{DB}[i]$

Decompose query into

$q_1, \ldots, q_m$ values in base d

$\beta \leftarrow f_{\mathsf{DB}}(\mathsf{Enc}_k(q_1), \ldots, \mathsf{Enc}_k(q_m))$

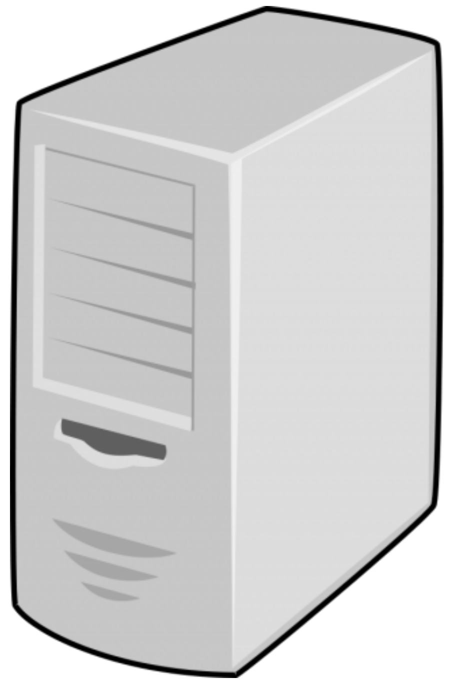
High-level approach [LMW'23]

Two ingredients:

- PIR scheme from encryption scheme that can evaluate low-degree functions
- *Preprocessing polynomial evaluation* [Kedlaya-Umans'08]

Database size n, $m = \log_d(n)$

$\mathsf{Enc}_k(q_1), \ldots, \mathsf{Enc}_k(q_m)$



Encode database as f_{DB}

where $f_{\mathsf{DB}}(i_1, \ldots, i_m) = \mathsf{DB}[i]$

Limitation: Technique for preprocessing polynomial evaluation has extremely high concrete costs

Open research problem: practical doubly efficient PIR

Deco

$q_1, \ldots,$

$\beta \leftarrow f_{\mathsf{DB}}(\mathsf{Enc}_k(q_1), \ldots, \mathsf{Enc}_k(q_m))$

Preprocess polynomial evaluation

Logistics

Project proposals due tonight at 11:59PM!

Look out for signups for meetings for feedback on project proposals.

References

Beimel, Amos, Yuval Ishai, and Tal Malkin. "Reducing the servers computation in private information retrieval: PIR with preprocessing." In *Annual International Cryptology Conference*, pp. 55-73. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

Chor, Benny, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. "Private information retrieval." *Journal of the ACM (JACM)* 45, no. 6 (1998): 965-981.

Corrigan-Gibbs, Henry, and Dmitry Kogan. "Private information retrieval with sublinear online time." In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 44-75. Cham: Springer International Publishing, 2020

Gilboa, Niv, and Yuval Ishai. "Distributed point functions and their applications." In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 640-658. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

Kushilevitz, Eyal, and Rafail Ostrovsky. "Replication is not needed: Single database, computationally-private information retrieval." In *Proceedings 38th annual symposium on foundations of computer science*, pp. 364-373. IEEE, 1997.

Lin, Wei-Kai, Ethan Mook, and Daniel Wichs. "Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE." In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pp. 595-608. 2023.

MIT 6.893 lectures