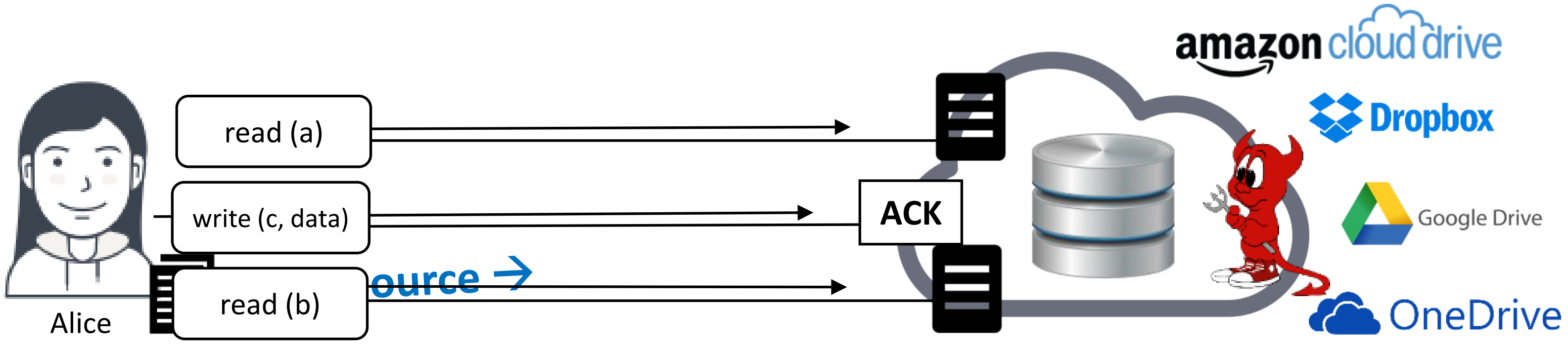
Towards Practical, Scalable and Private Management of Cloud Data

Amr El Abbadi
Department of Computer Science
University of California at Santa Barbara

In Collaboration with Ishtiyaque Ahmad, Sujaya Maiyya, Divy Agrawal, Trinabh Gupta, Rachel Lin, Stefano Tessaro, etc

Outsourced Private Data



Security Concerns?

Confidentiality of Data

Typical Solution

Encryption

GOAL

Developing algorithms that can answer queries over securely outsourced data without fetching all data

Lots of on-going works

Outsourced Private Data

Security Concerns?

Confidentiality of Data

Encryption

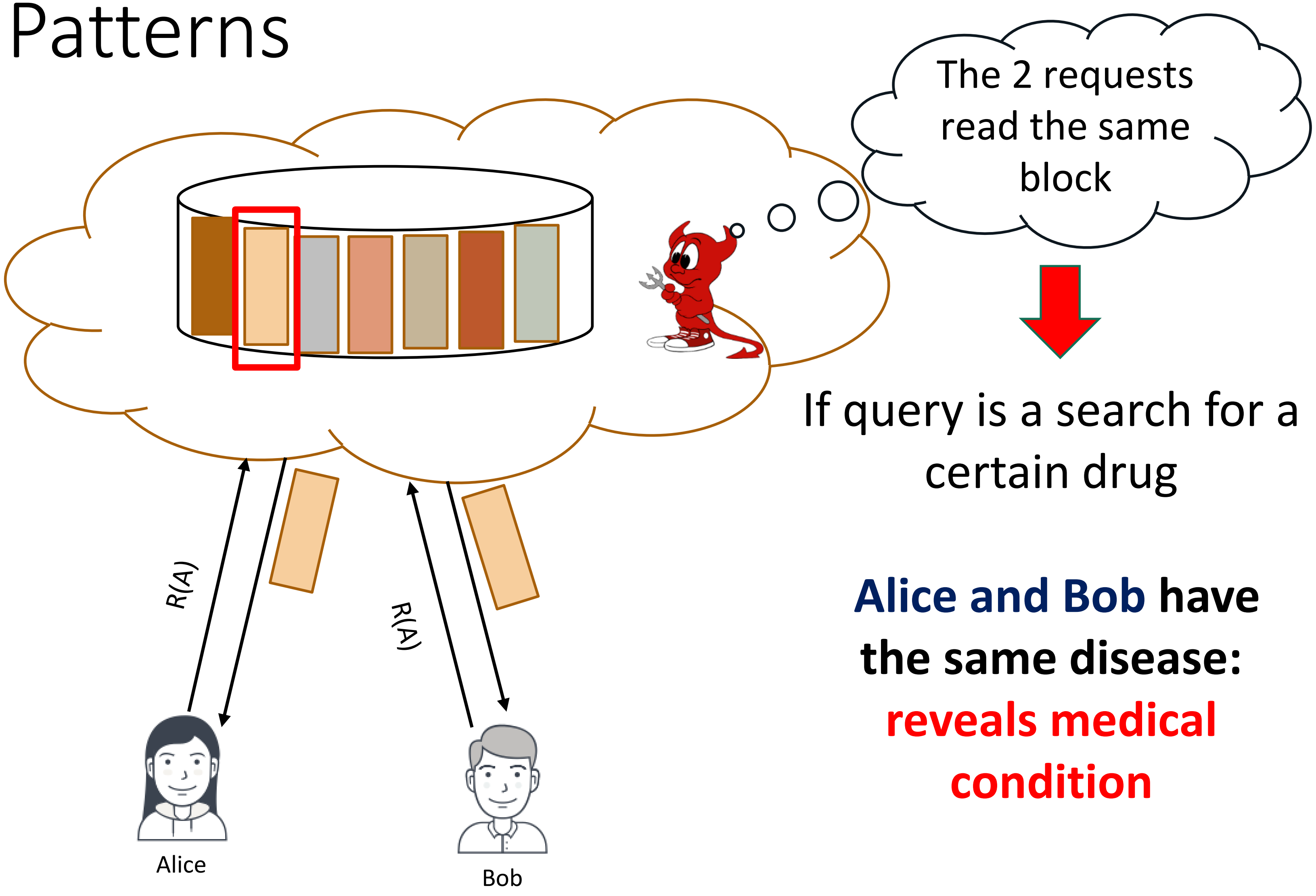
Is encryption enough?

Encryption alone is not enough!!!

Access patterns can leak sensitive information
[Islam et al. NDSS’12]

read(1), read(1) vs read(1), write(3)

Access Patterns



If query is a search for a certain drug

Alice and Bob have the same disease: reveals medical condition

Data Privacy

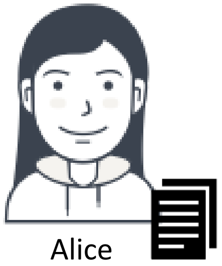
- Sensitive data needs to be kept private
- Encryption is a good start but isn't good enough
- Honest-But-Curious attackers could observe the ***access pattern***
  - Which data item is being accessed?
  - When it was accessed?
  - How frequently?
- The access pattern can be used to leak information about the actual data content and user activity

We need mechanisms to ensure

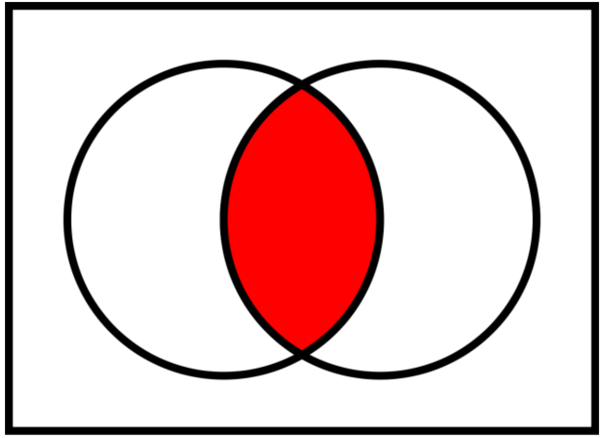
data security and privacy

that are scalable, efficient and fault-tolerant

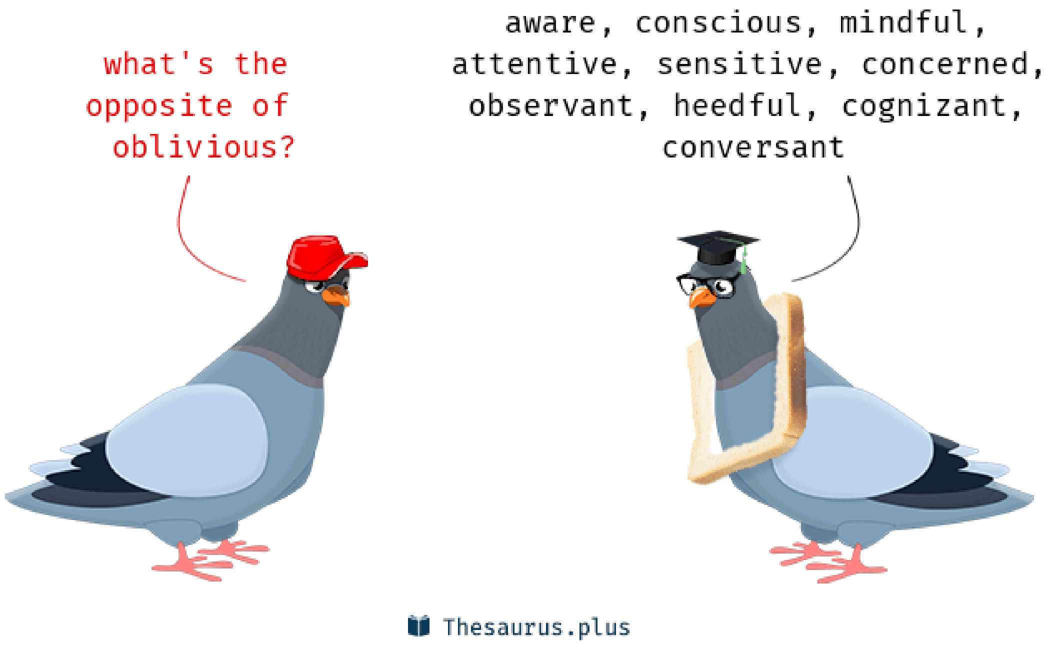
Lets Start with Private User Data



→ outsource →



Oblivious RAM



ORAM Problem Statement

- Clients wish to outsource data to an untrusted cloud storage
- Honest-But-Curious cloud can control & observe network & cloud storage
- Keep the **data** and **access pattern** private

Client 1

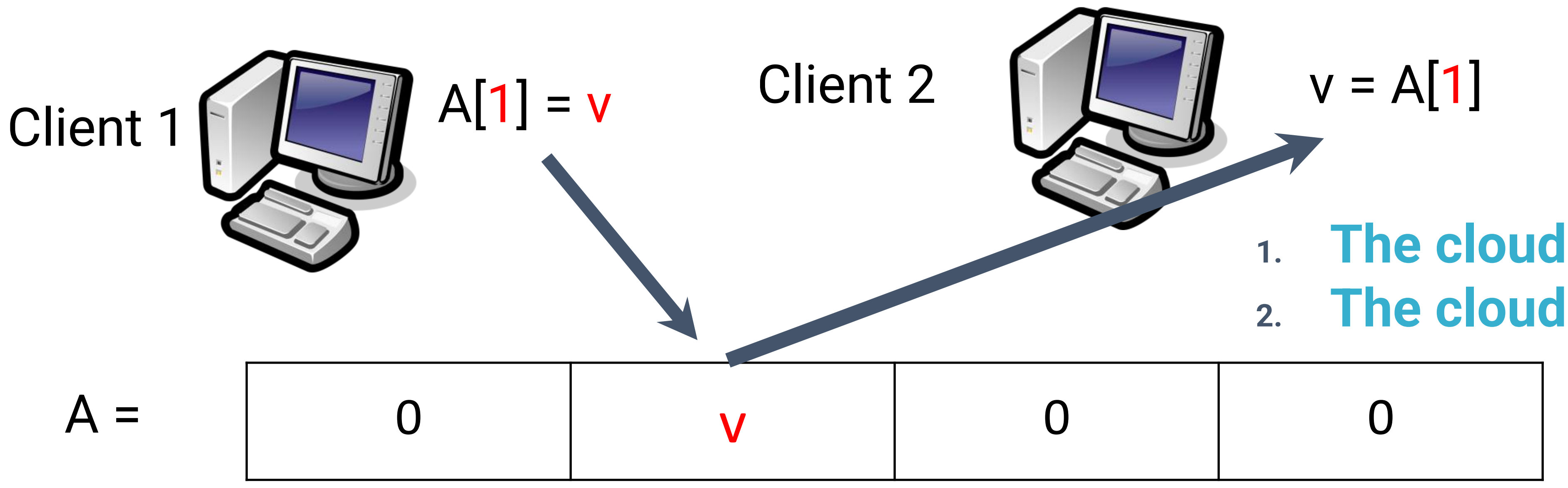
Client 2

A =

0	0	0	0

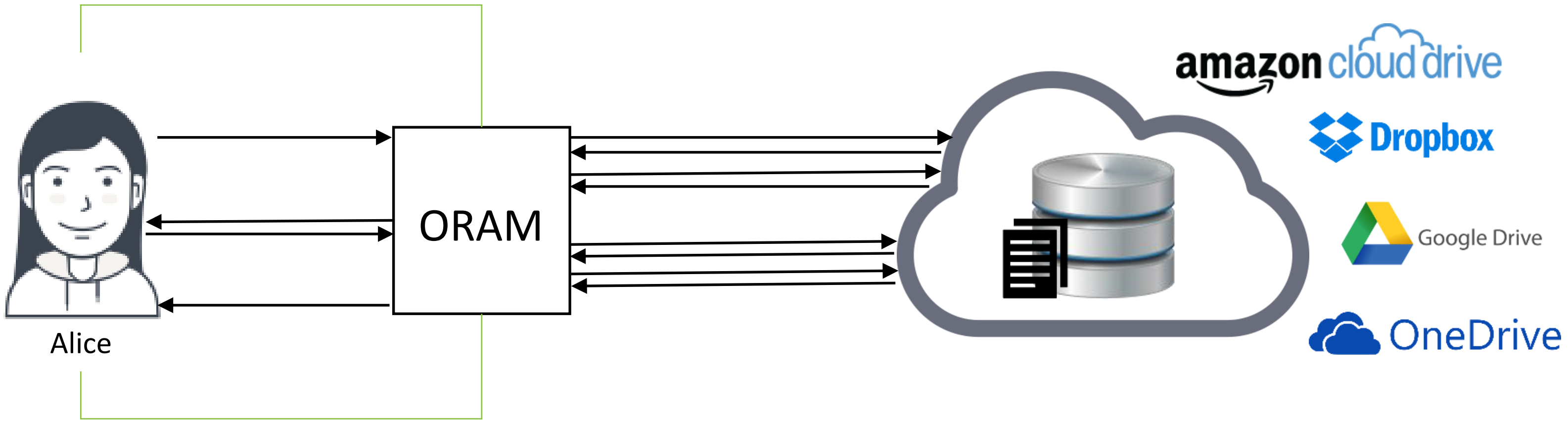
ORAM Problem Statement

- Clients wish to outsource data to an untrusted cloud storage
- Honest-But-Curious cloud can control & observe network & cloud storage
- Keep the **data** and **access pattern** private



1. **The cloud** never learns **value v**
2. **The cloud** never learns **index 1**

Outsourced Private Data



Goal: Oblivious Access

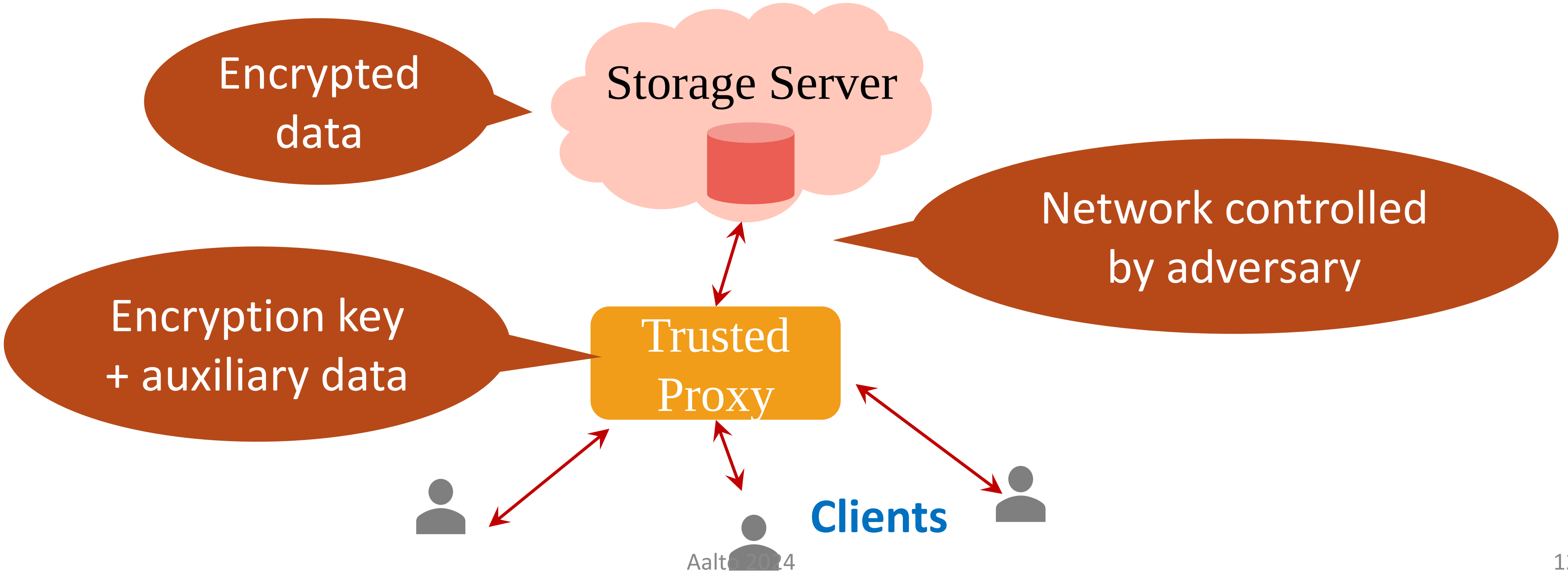
Translate each logical access
to a sequence of random-looking accesses

OBLIVIOUS RAM (ORAM)

Initially proposed by **[Goldreich and Ostrovsky, JACM’96]**

Oblivious RAM (ORAM)[1] mitigates access pattern attacks

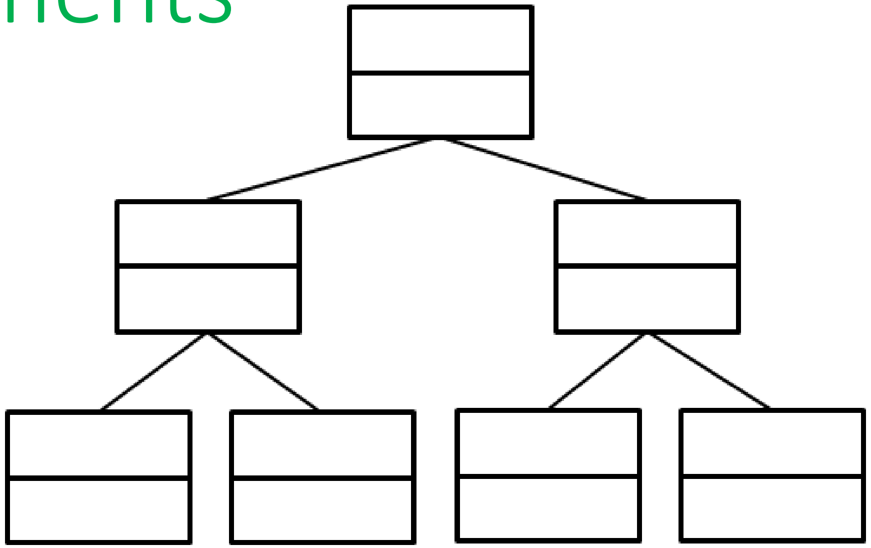
- Core idea: make all data accesses **look random**
- Supports single item **Get-Put** operations
- Typical (but not all) ORAM architecture



[1] O. Goldreich & R. Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM, 1996*

Tree-based ORAM Developments

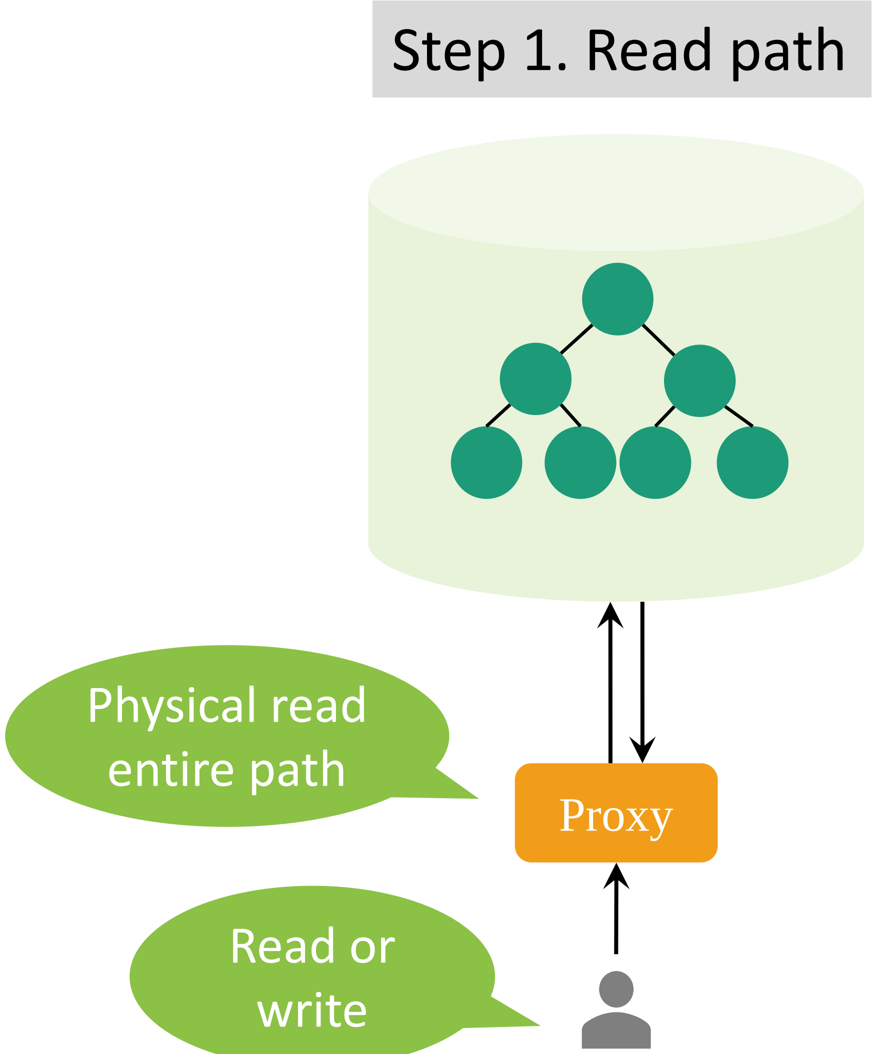
- Initial construction by [Shi et al. 2011]

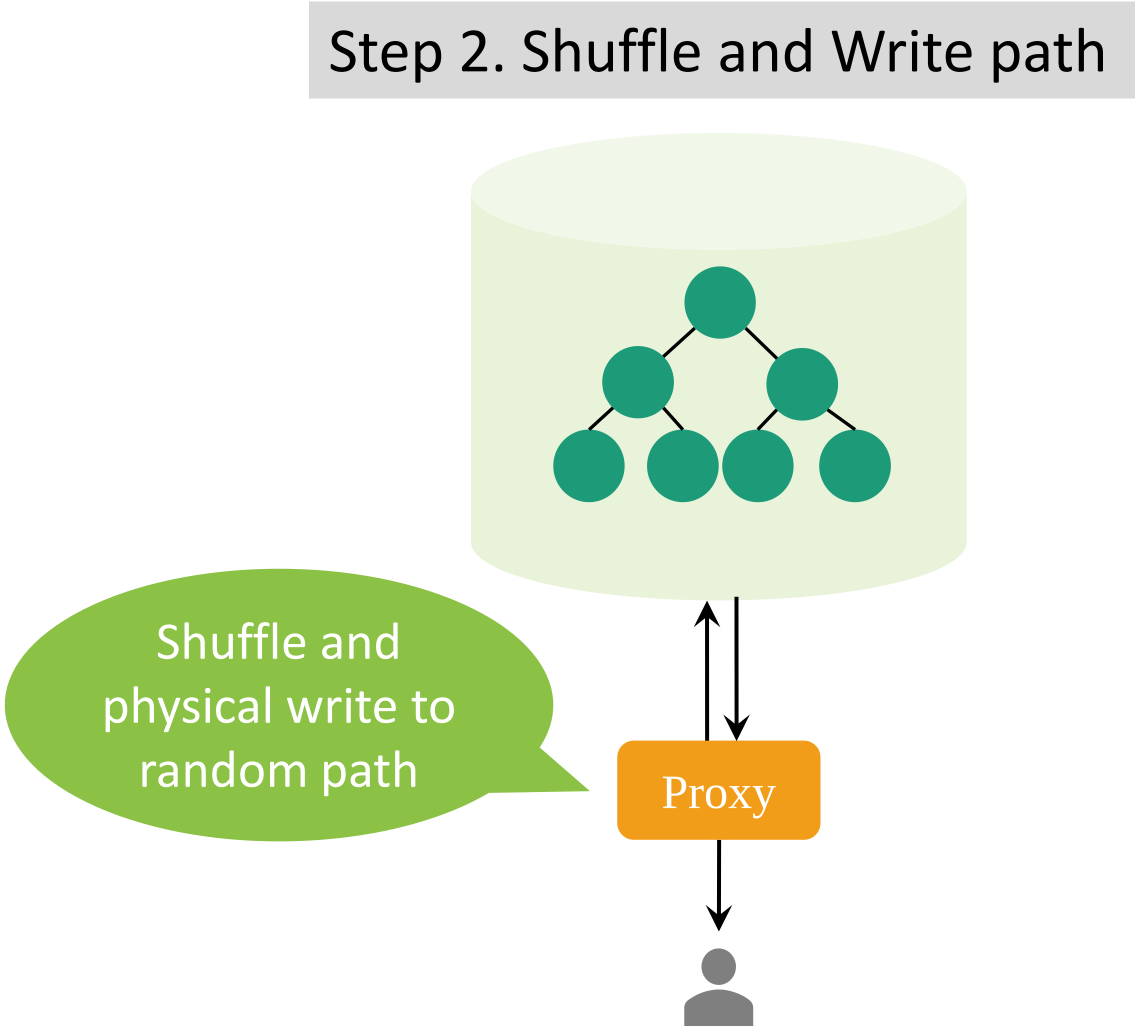


More practical and famous solution

- Path ORAM: an extremely simple oblivious RAM protocol [Stefanov et al. CCS'13]

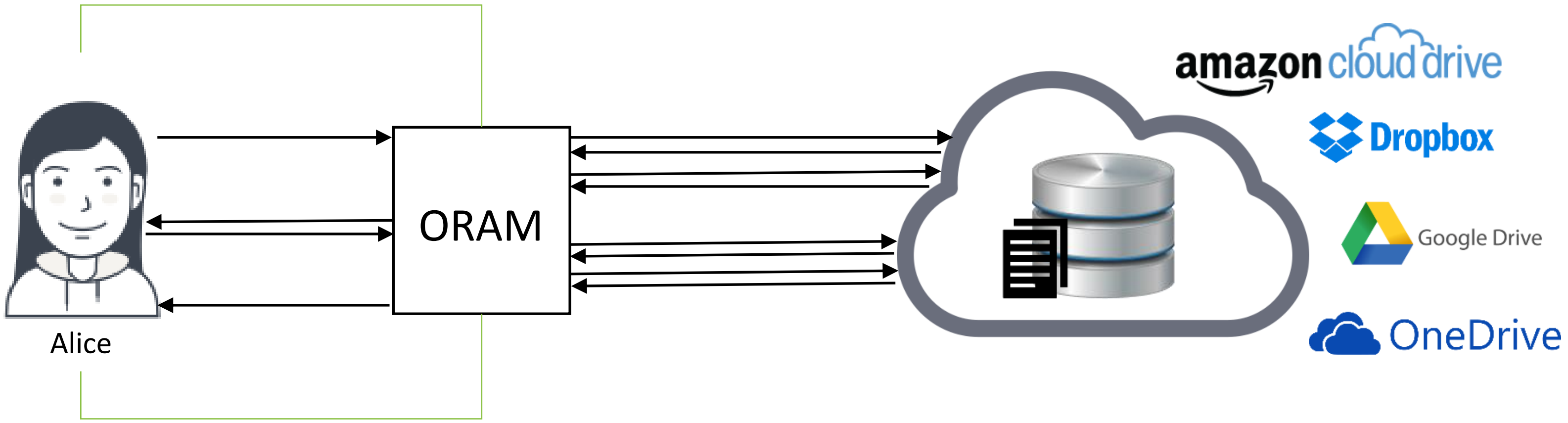
1000 feet overview of ORAM (PathORAM[1])





[1] E. Stefanov, et al. "Path ORAM: an extremely simple oblivious RAM protocol." *Proceedings of the 2013 ACM SIGSAC*. 2013.

ORAM Status circa 2016



Goal: Oblivious Access

Translate each logical access
to a sequence of random-looking accesses

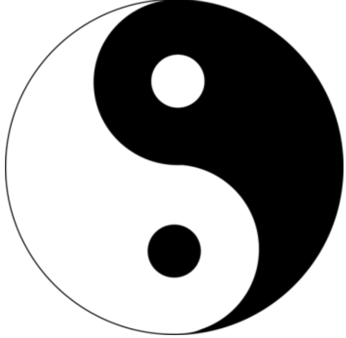
OBLIVIOUS RAM (ORAM)

Initially proposed by [Goldreich and Ostrovsky, JACM'96]

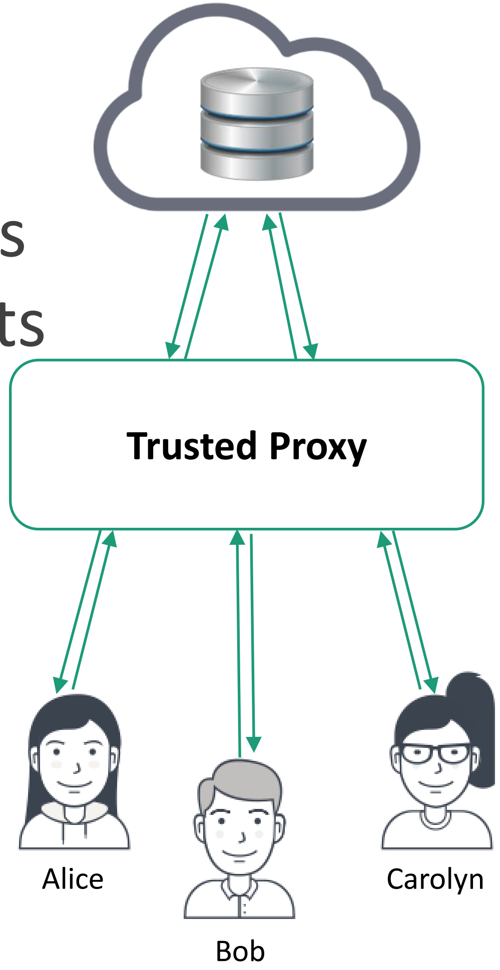
More solutions: MG'11, DB'11, ES'11, EK'12, ES'12, PW'12, ES'13, CG'13, KC'13, KC'14, LR'15, TM'15 , SD'16, …

Single client

TaoStore: Tree-Based Asynchronous Oblivious Store



- Fully concurrent and asynchronous oblivious access
- Concurrent and non-blocking processing of requests
  - Makes tree-based ORAM concurrent



Goal: Improve overall performance

[Sahin, Zakhary, El Abbadi, Lin Tessaro. **TaoStore: Overcoming Asynchronicity in Oblivious Data Storage.** S&P'16]

Current ORAM data stores are **not** fault tolerant

Loose access to entire data!!

Maiyya et al. QuORAM: A Quorum Replicated Fault-Tolerant ORAM Datastore. Usenix Security 2022

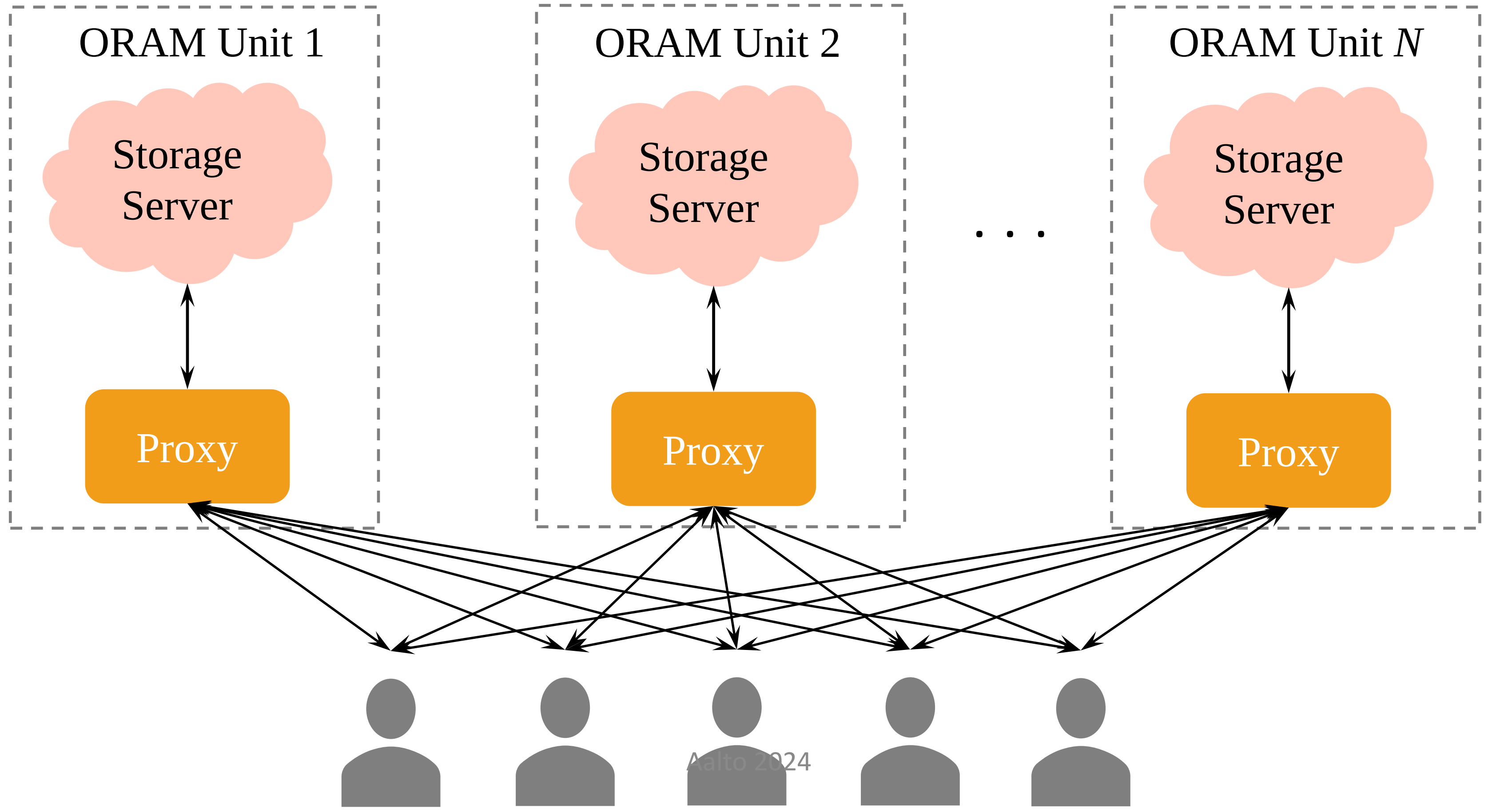
Obladi [1] provides data durability guarantees using fault-tolerant cloud db

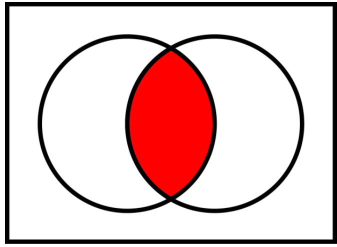
- No liveness: single point of failure when proxy crashes

[1] N. Crooks, et al. "Obladi: Oblivious serializable transactions in the cloud." *13th OSDI* 2018.

QuORAM system and threat model

- Tolerates ***f*** failures out of ***2f+1*** replicas
- **Honest but curious** adversary: can control storage server & all communication channels





QuORAM's 3-fold goal:

Hide access pattern + Replicate data + Provide linearizable semantics

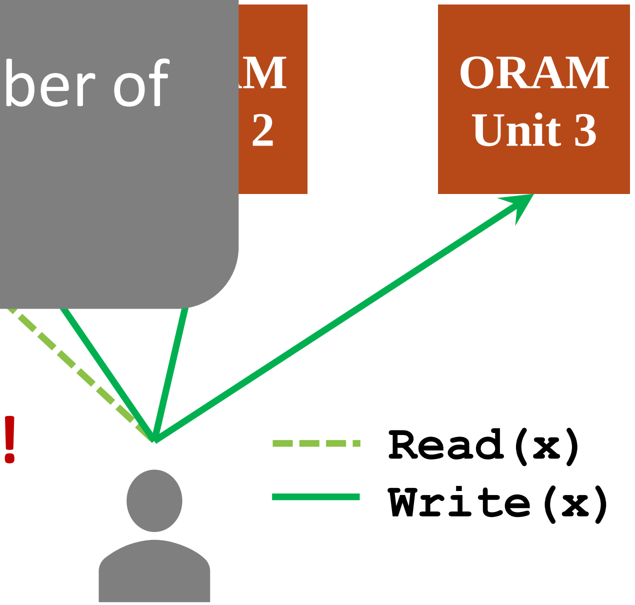
- Linearizability: all read/write operations on a data item **appear** to be linear
- Data replication can be **expensive** due to geo-distributed replicas
- Use efficient replication protocol?

Solution 1: Efficient replication protocol

- Protocols such as Virtual Partitions [1,2], Hermes[3] or CRAQ[4] are re

- Read from: **on**
  Write to: **ALL**

- Number of message exchanges **reveal type of operation!!**



Need a protocol that accesses same number of replicas for reads and writes

[1]El Abbadi et al "An Efficient, Fault-Tolerant Protocol for Replicated Data Management", PODS 1984
[2] El Abbadi and Toueg "Availability in Partitioned Replicated Databases." PODS 1985.
[3] Antonios Katsarakis, et al. "Hermes: A fast, fault-tolerant and linearizable replication protocol." *ASPLOS* 2020.
[4] Jeff Terrace and Michael J. Freedman. "Object Storage on CRAQ: High-throughput Chain Replication for Read-mostly Workloads." USENIX Annual Technical Conference 2009

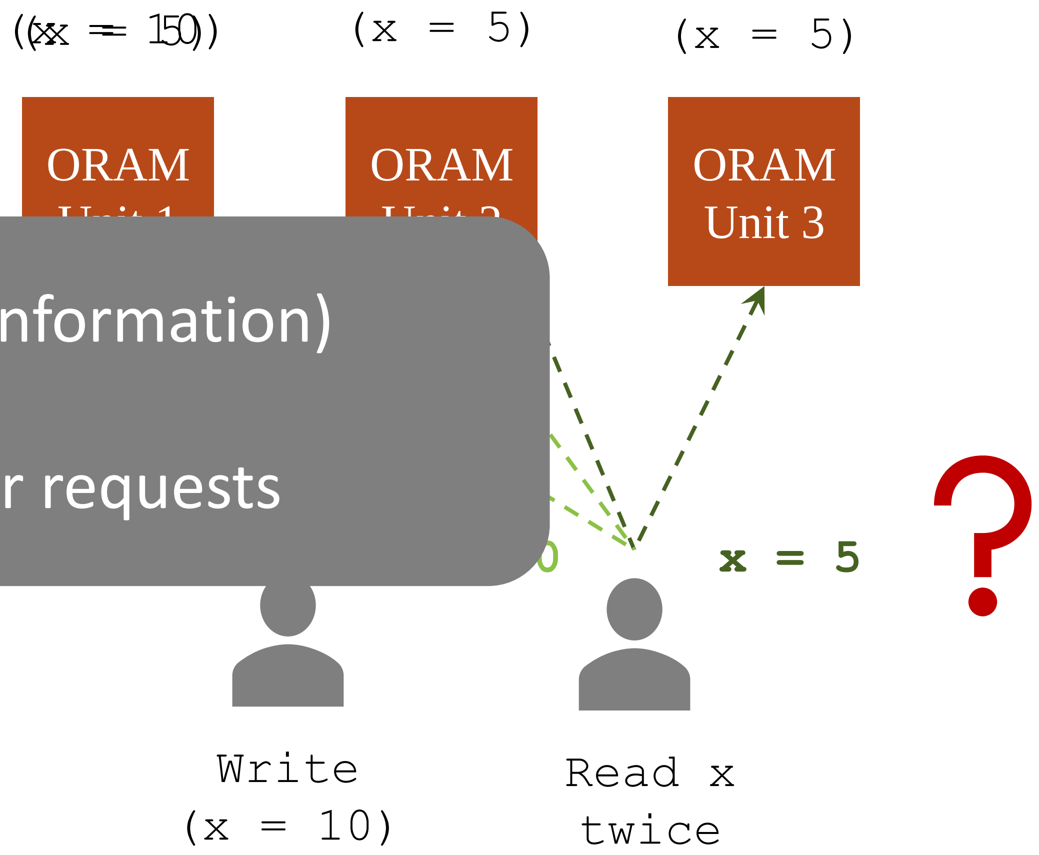
Solution 2: Access the same number of replicas for read and writes

- Single round

- Read from:
  Write to: a
- **Note: maj**

Need **locking** (may leak information)
or
another round to order requests

- Concurrent & conflicting operations may **violate linearizability!!**

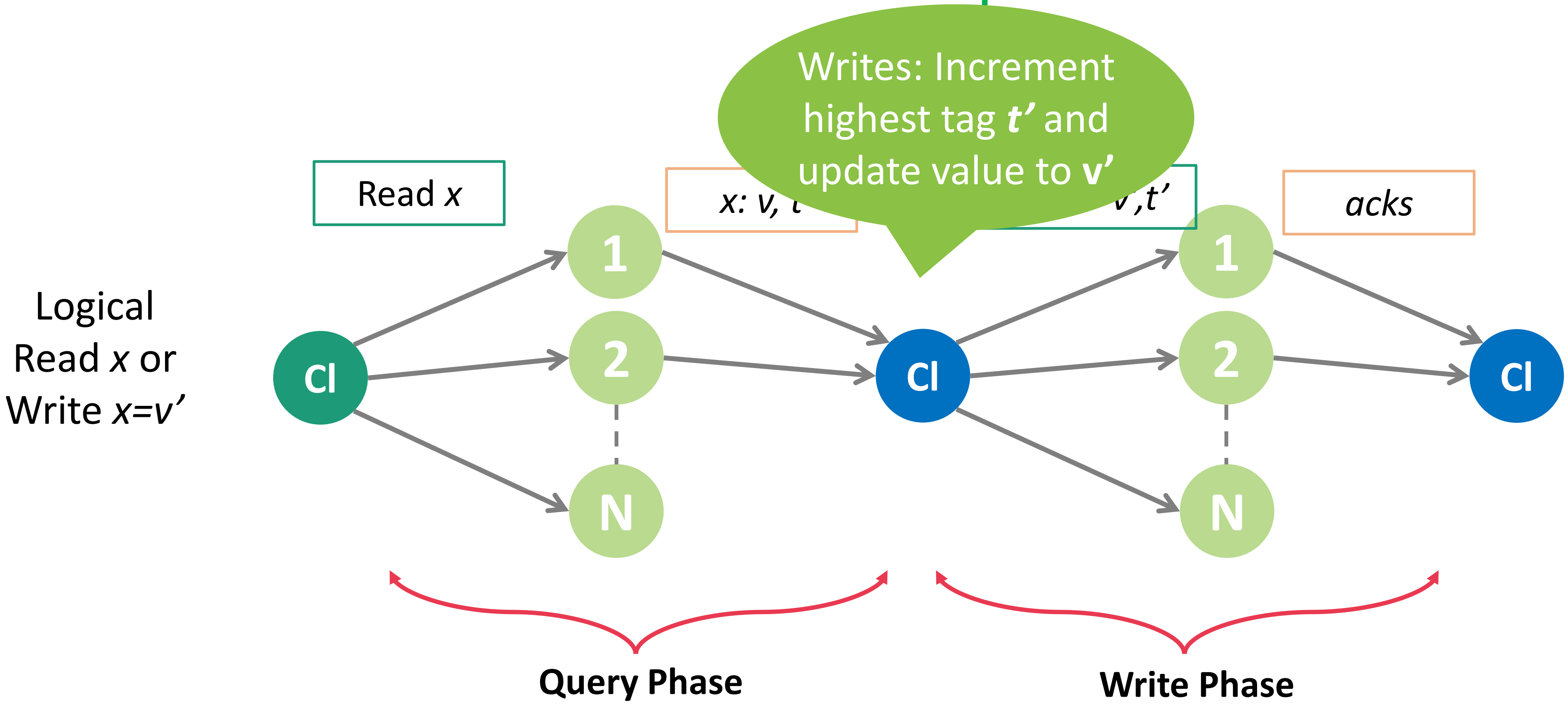


QuORAM Replication

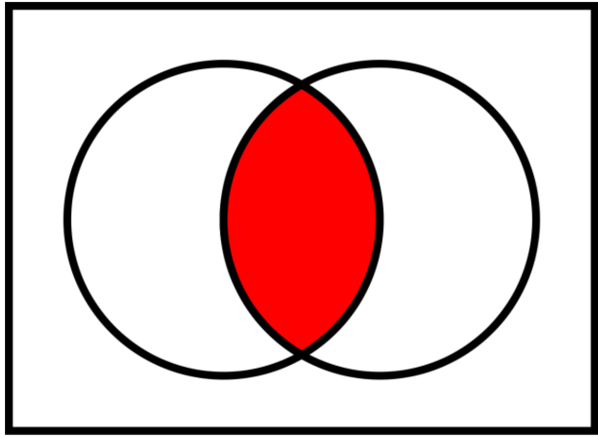
- Inspired by Lynch and Shvartsman's [1] solution
- Data item *x* has value *v* and a monotonically increasing tag *t*
- Two phase replication: Query + Propagate
  - Each logical request → two ORAM requests

[1] N. Lynch, and A. Shvartsman. "Robust emulation of shared memory using dynamic quorum-acknowledged broadcasts." *27th International Symposium on Fault Tolerant Computing*. IEEE, 1997.

QuORAM Replication



QuORAM's 3-fold goal:

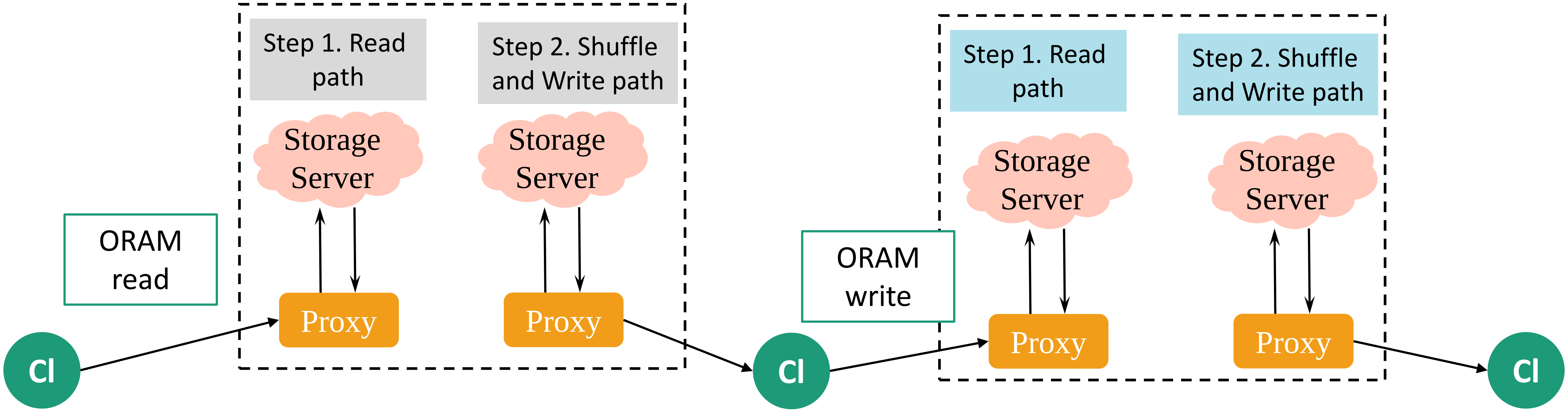


Hide access pattern + Replicate data + Provide linearizable semantics

- We use **TaORAM** [1] – extends PathORAM to include **concurrency** and asynchronous settings

[1] C.Sahin, et al. "Taostore: Overcoming asynchronicity in oblivious data storage." *IEEE S&P*, 2016.

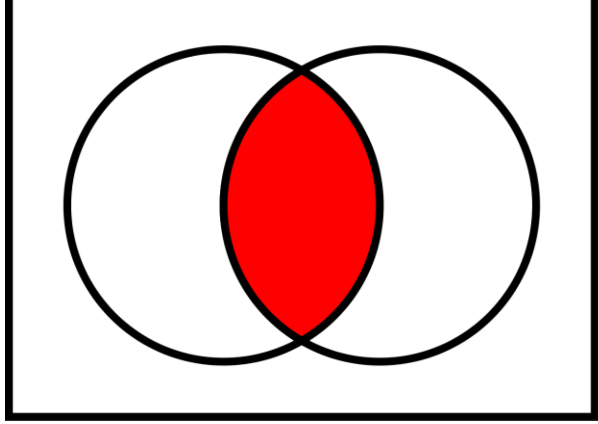
Unchanged ORAM scheme **double fetches** the same path

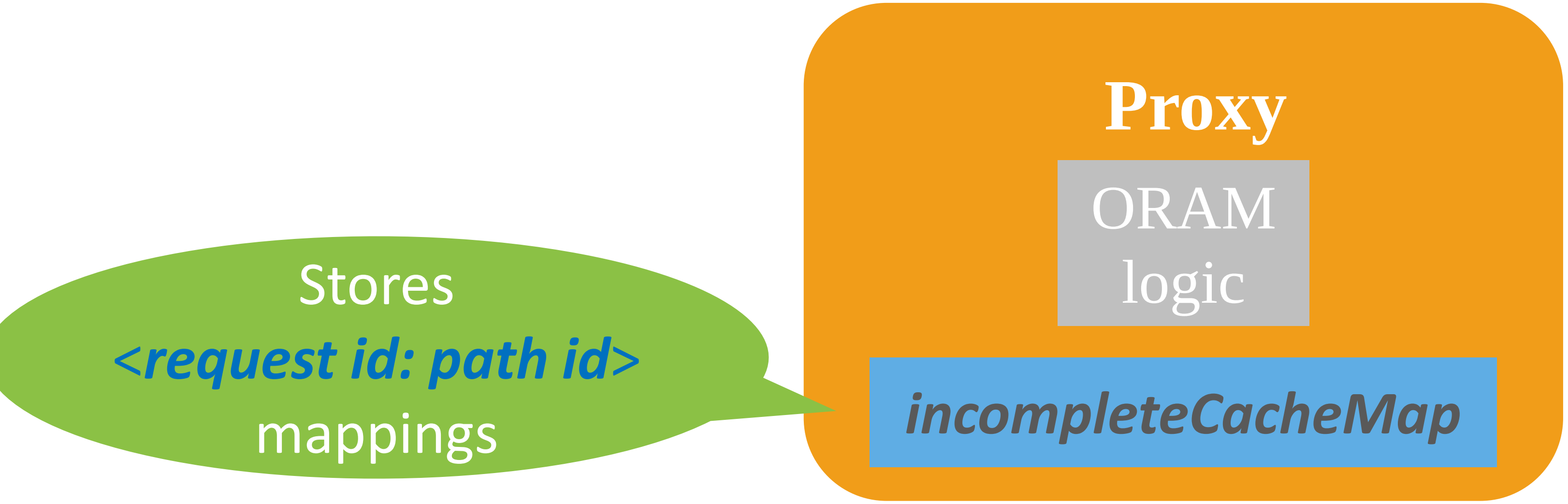


- QuORAM replication: 2 rounds (query + propagate)
- Each ORAM operation: 2 rounds (read path + shuffle & write path)

- **4 rounds of communication!!!**

QuORAM avoids double-fetching of paths by tracking ongoing requests



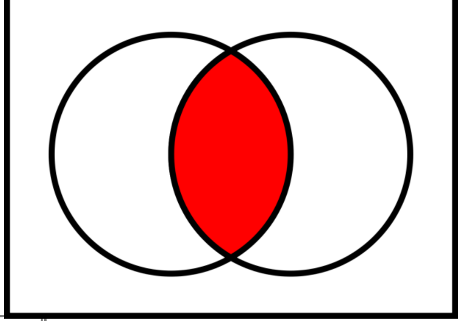


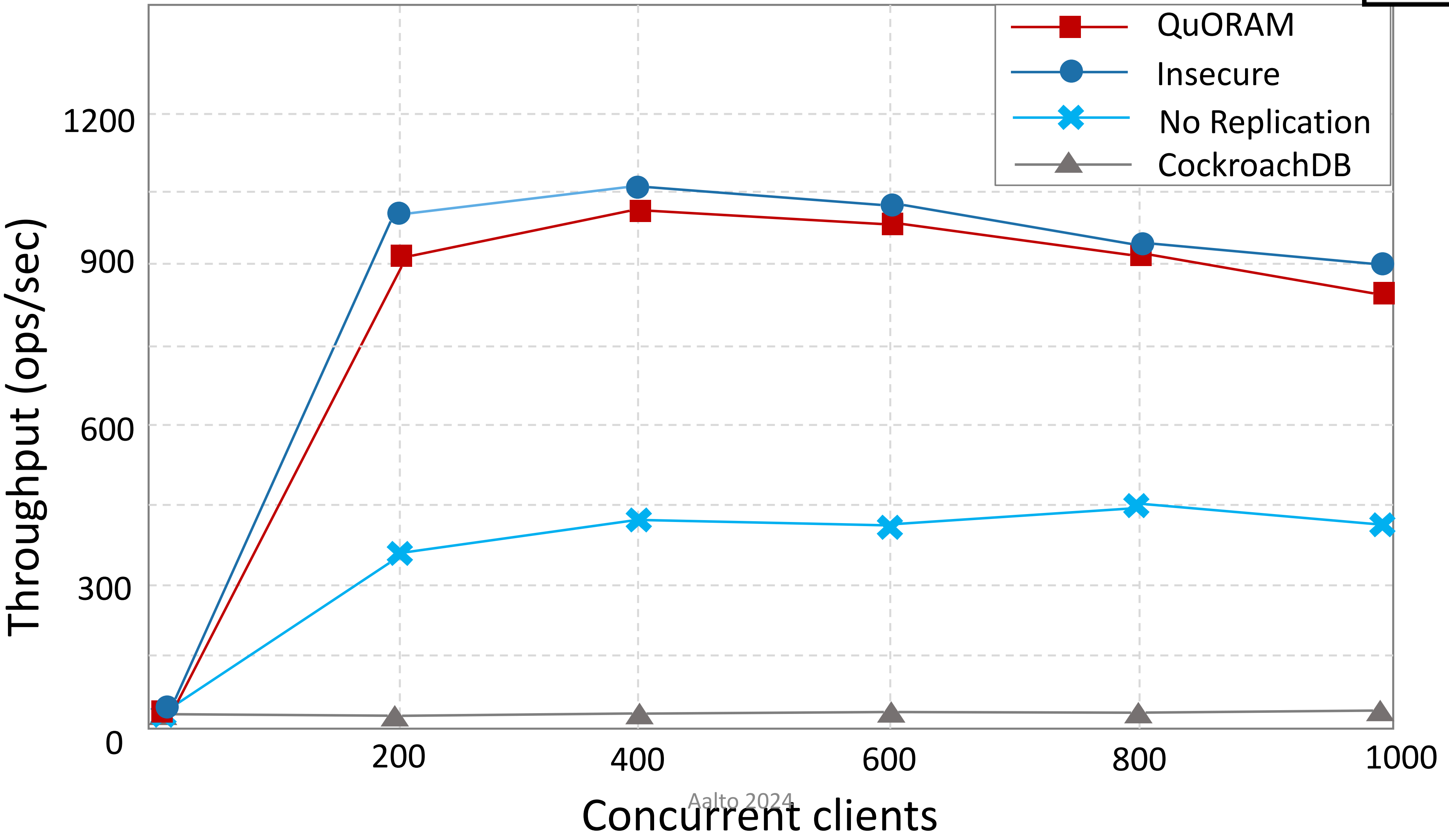
Check details in USENIX Security'22 paper

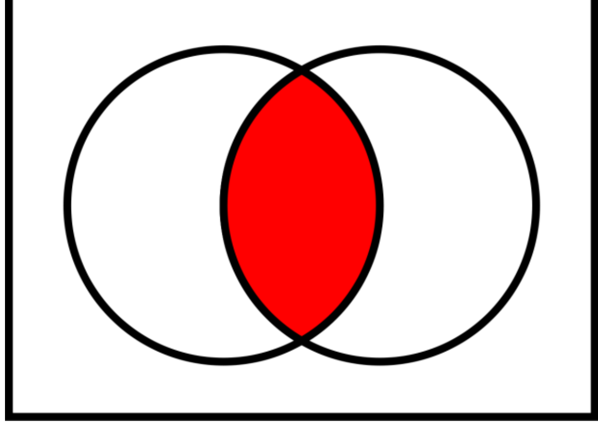
- Store <***request id: path id***> of requests that finished query but not propagate phase
- Proxies **fetch a path only once** for the query phase

Maiyya, Ibrahim, Scarberry, Agrawal, El Abbadi, Lin, Tessaro, Zakhary
QuORAM: A Quorum Replicated Fault-Tolerant ORAM Datastore. Usenix Security 2022

Throughput: higher is better

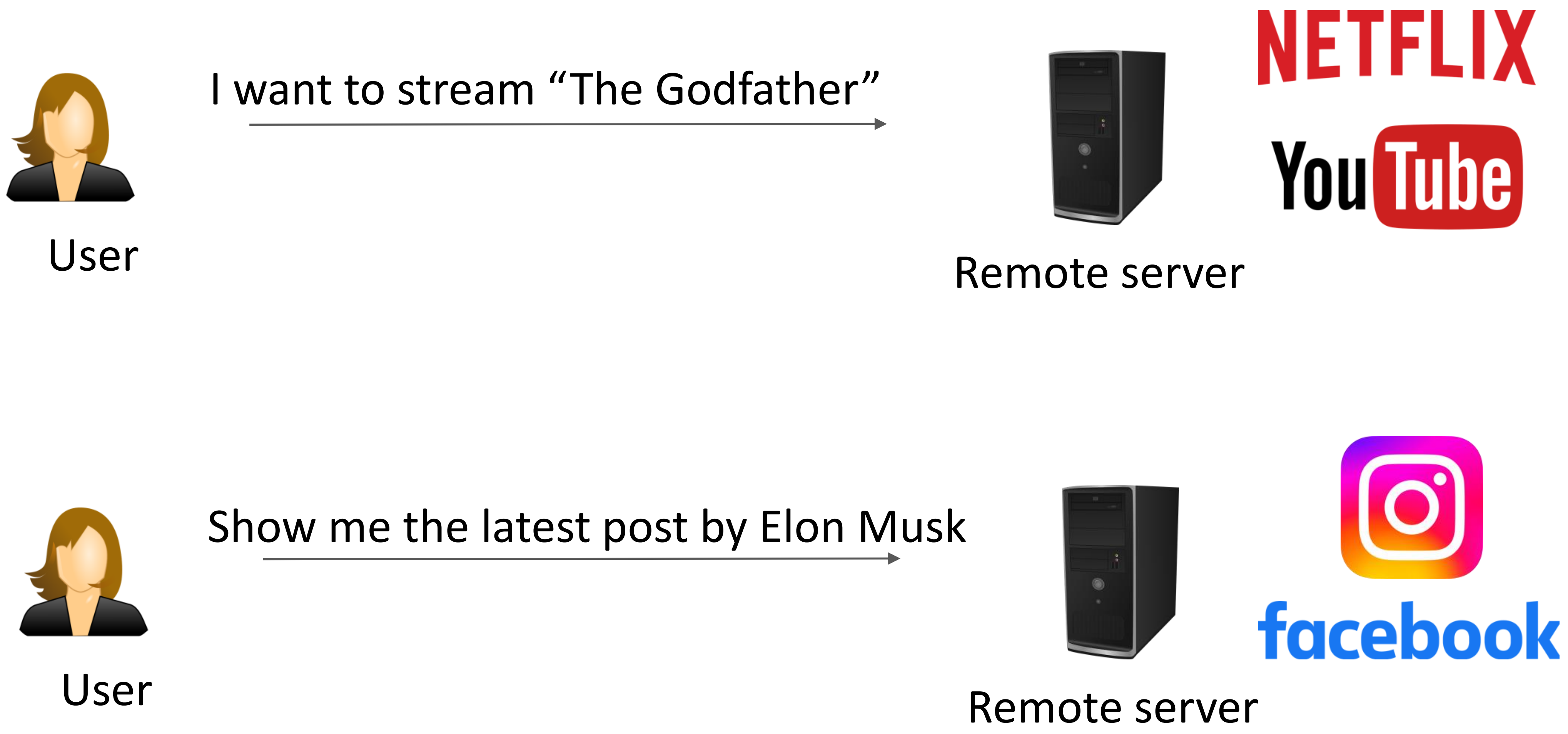




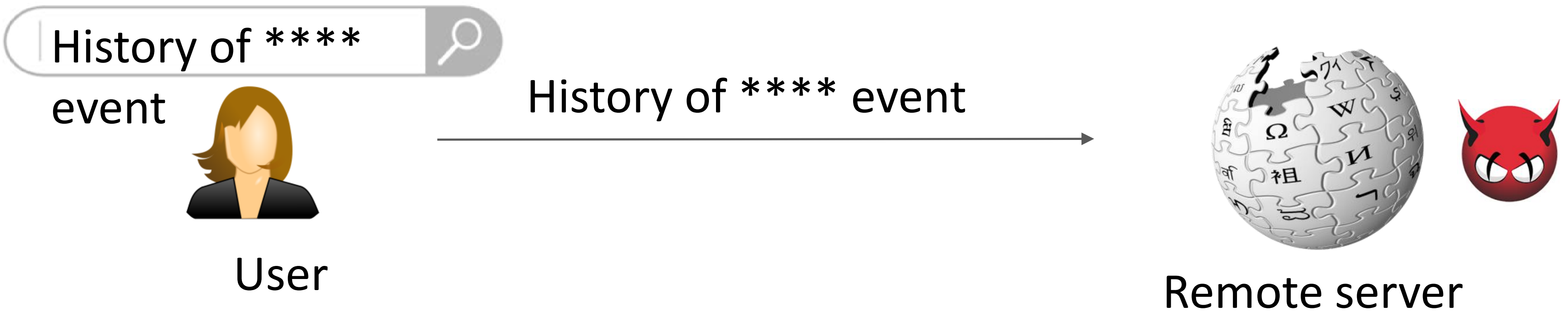


We can have Privacy AND Fault-tolerance!

But, much of the content on the Internet is in *public data repositories*



Both users and service providers want to hide access patterns over public repositories



User may:

- Consider queries private
- Belong to a vulnerable population or a minority group

Server can be:

- Hacked by an outsider
- Compromised by an insider
- Coerced by a nation state [1, 2]

1. Brian Fung. *Analysis: There is now some public evidence that China viewed TikTok data.* CNN, 2023.
2. Sapna Maheshwari and Ryan Mac. *Driver's Licenses, Addresses, Photos: Inside How TikTok Shares User Data.* New York Times, 2023

How can we hide access patterns (queries) over public data repositories?

History of pride parade

User

History of pride parade

Remote server

Cannot use:

- Encryption
- ORAM
- CryptDB-like solution

Private Information Retrieval: Retieval by Location

k_0	0
k_1	1
k_2	2
…	…
k_{n-1}	n-1

Give me the *i*-th value

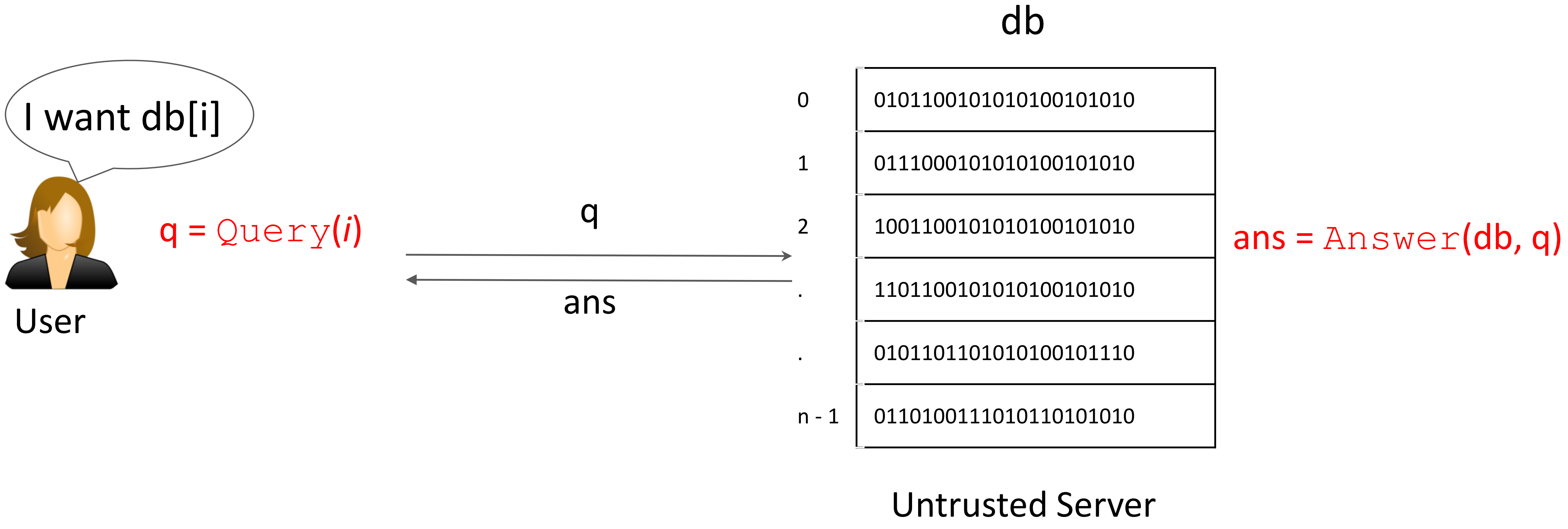
0	v_0
1	v_1
2	v_2
…	…
n-1	v_{n-1}

Untrusted Server

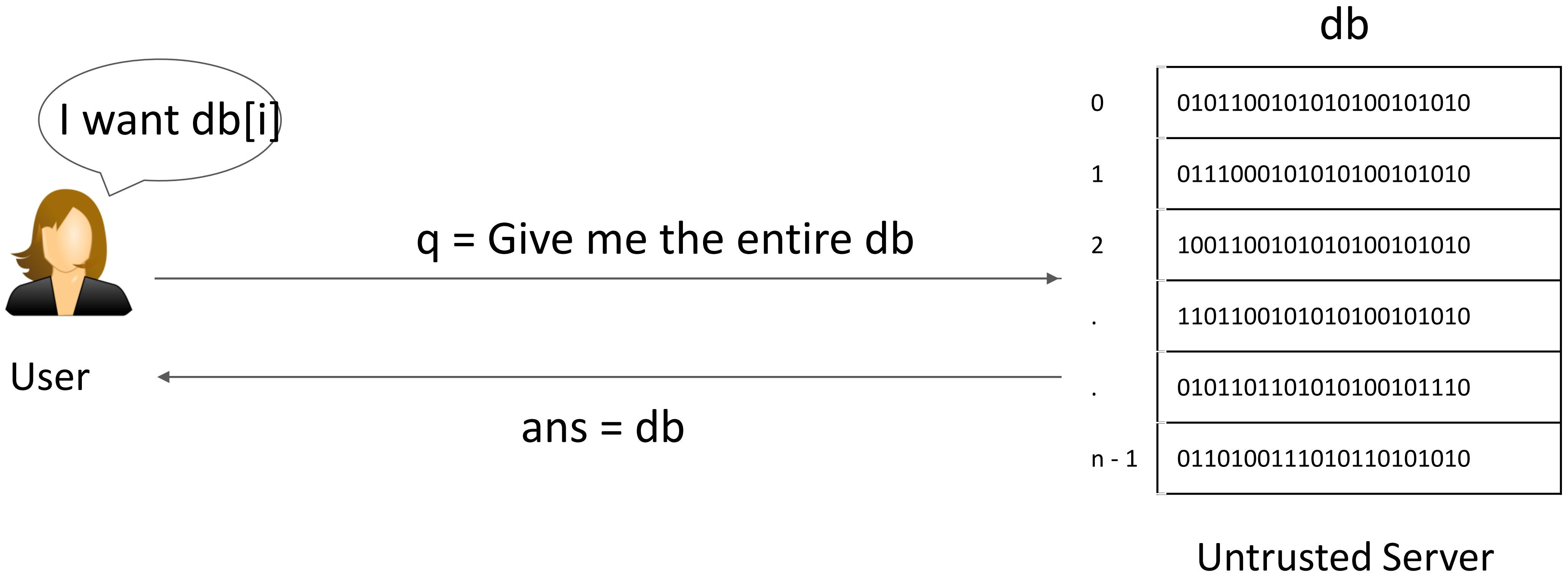
Client has (key, location) mapping

How can the client privately retrieve the value corresponding to a given location?

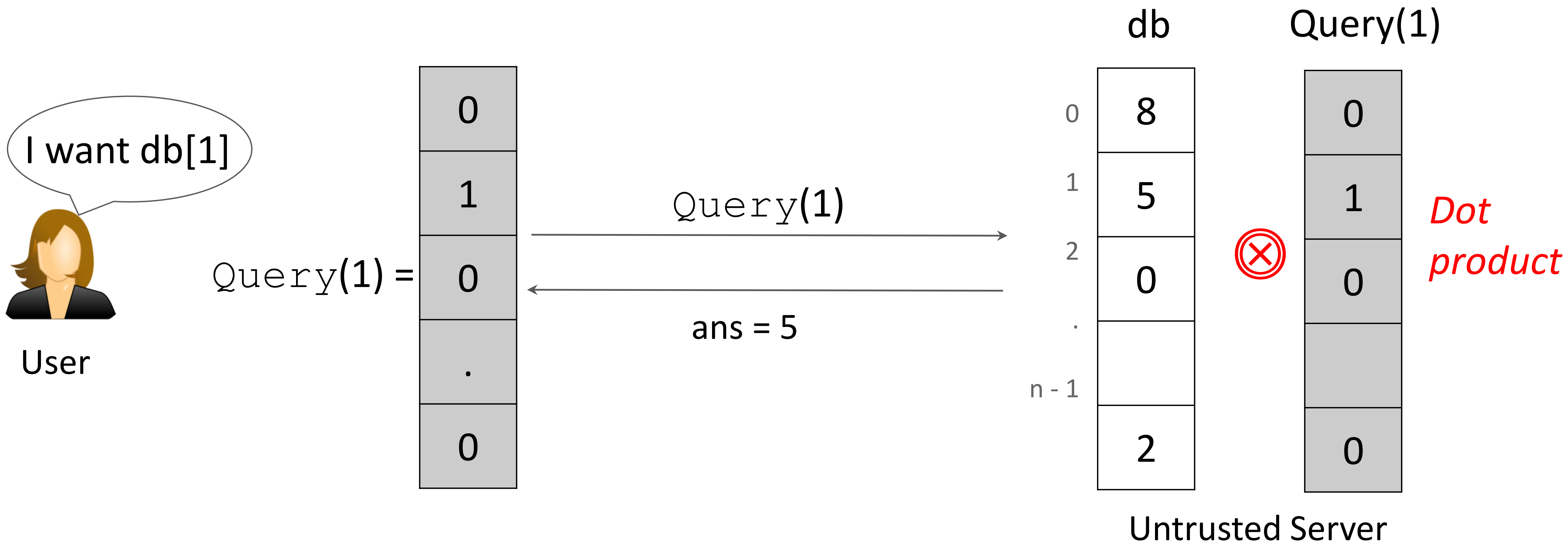
This area originated as private information retrieval (PIR) in 1995 (Chor et al. FOCS '95)



One trivial solution to private information retrieval

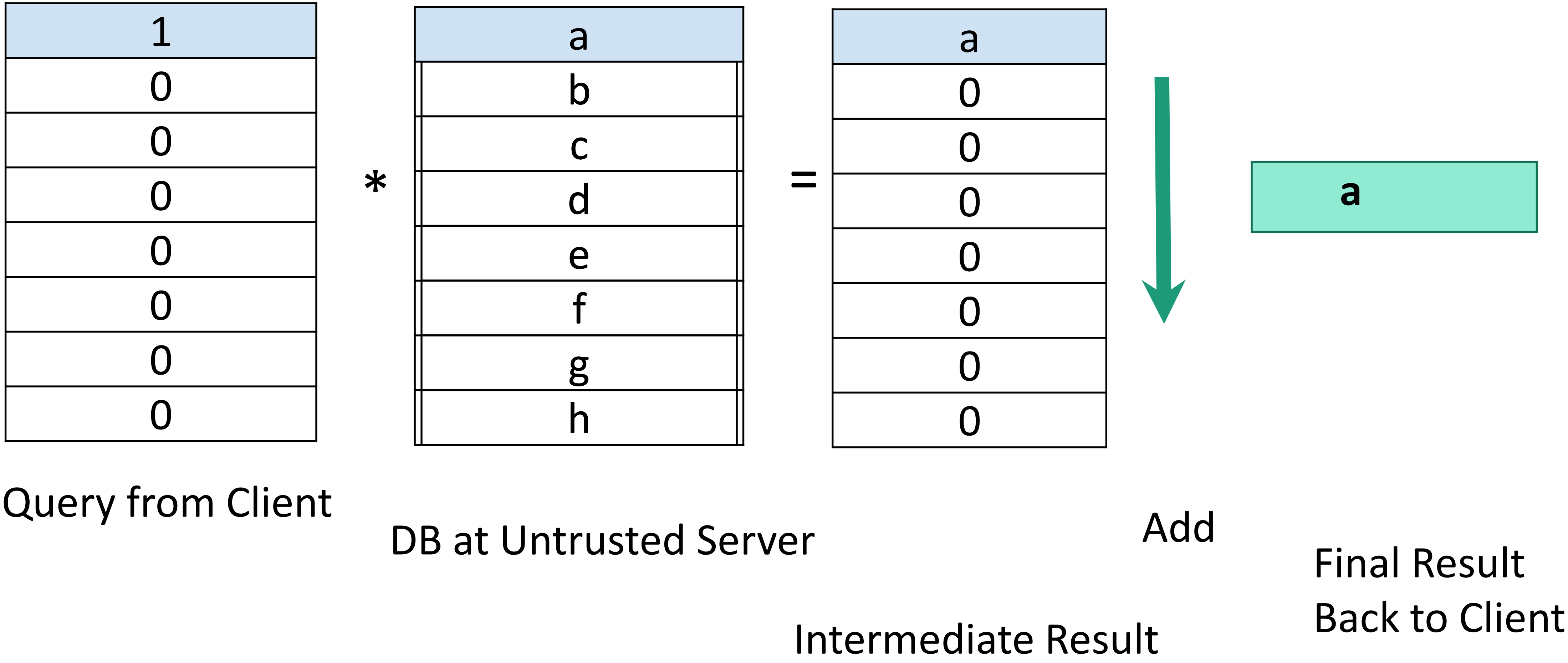


PIR Foundations



Retrieval is equivalent to computing a dot product

Basic Idea in CPIR



How to achieve this in a Private manner?

Homomorphic Encryption

Homomorphic Encryption

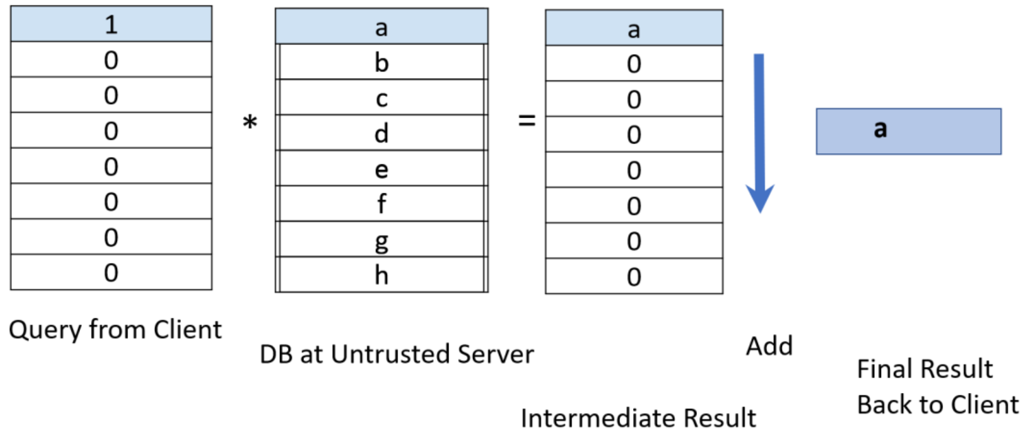
A form of encryption which allows specific types of computations to be carried out on cipher texts without decrypting it.

Partially Homomorphic
- Either additive or multiplicative
- Paillier, El Gamal, …etc.

Fully Homomorphic
- Breakthrough by Gentry 09.
- Supports computations for any arbitrary function
- Can be inefficient for arbitrary Boolean functions
- However:
  - Quantum Secure
  - Can be much faster if properly used—our plan!

Challenges

Basic Idea in PIR



Query size is too large: cipher text * size of database.

- For example, for 32K entries in db and cipher text size 64K
- A query is 64K * 32K= 2 GigaByte!

Much of the research on PIR is on reducing request size and server-side compute overhead

Overhead	High-level technique
Request size	• Recursion (Stern 1998) • Cryptographic query compression (SealPIR '18)
Server-side compute	• PIR with preprocessing (Beimel et al. '00, SimplePIR '23) • Lattice-based cryptography (FastPIR '21)

How to reduce server-side overhead?

Pay linear overhead but improve the constant

Key techniques in FastPIR (OSDI ‘21)

- Use lattice-based additive homomorphic encryption schemes
- Single-input multiple data (SIMD) capabilities
- Query and response compression using homomorphic rotation operations

For more details:

Check our Addra OSDI ’21 paper

Come on Friday

Ahmad, Yang, Agrawal, El Abbadi, Gupta

Addra: Metadata-private voice communication over fully untrusted infrastructure.

Retrieval by key

I’m interested in value of key k

Give me value for key k

Client retrieves:

- v, if (k,v) at Server
- ∅, otherwise

k_0	v_0
k_1	v_1
k_2	v_2
…	…
k_{n-1}	v_{n-1}

Untrusted Server

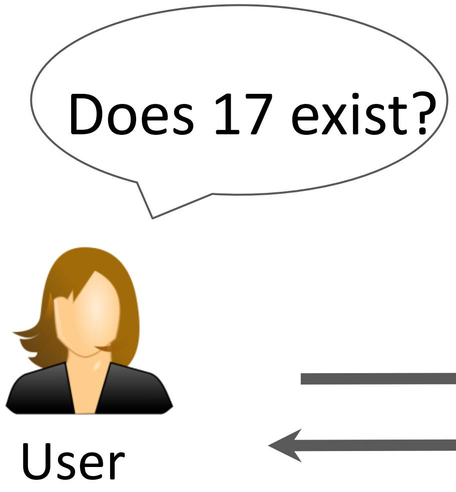
How can the client privately retrieve the value corresponding to a given key?

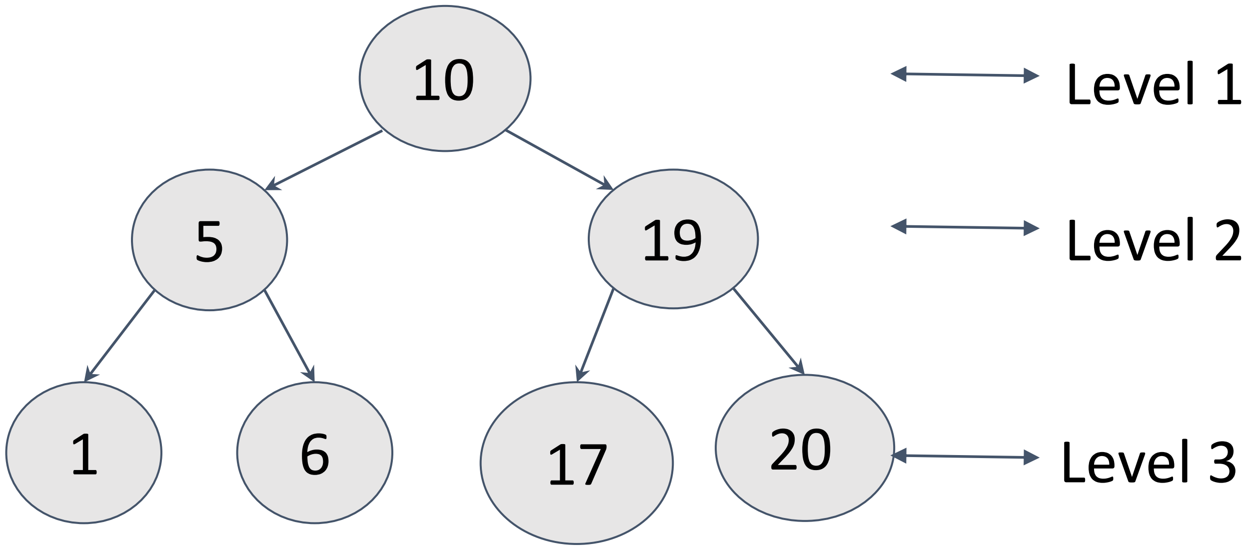
PIR-by-keywords: Muti-round PIR-by-index

(Chor et al. TOC '98)

Assume keys are integers and arranged in a BST

K = {1, 5, 6, 10, 17, 19, 20}

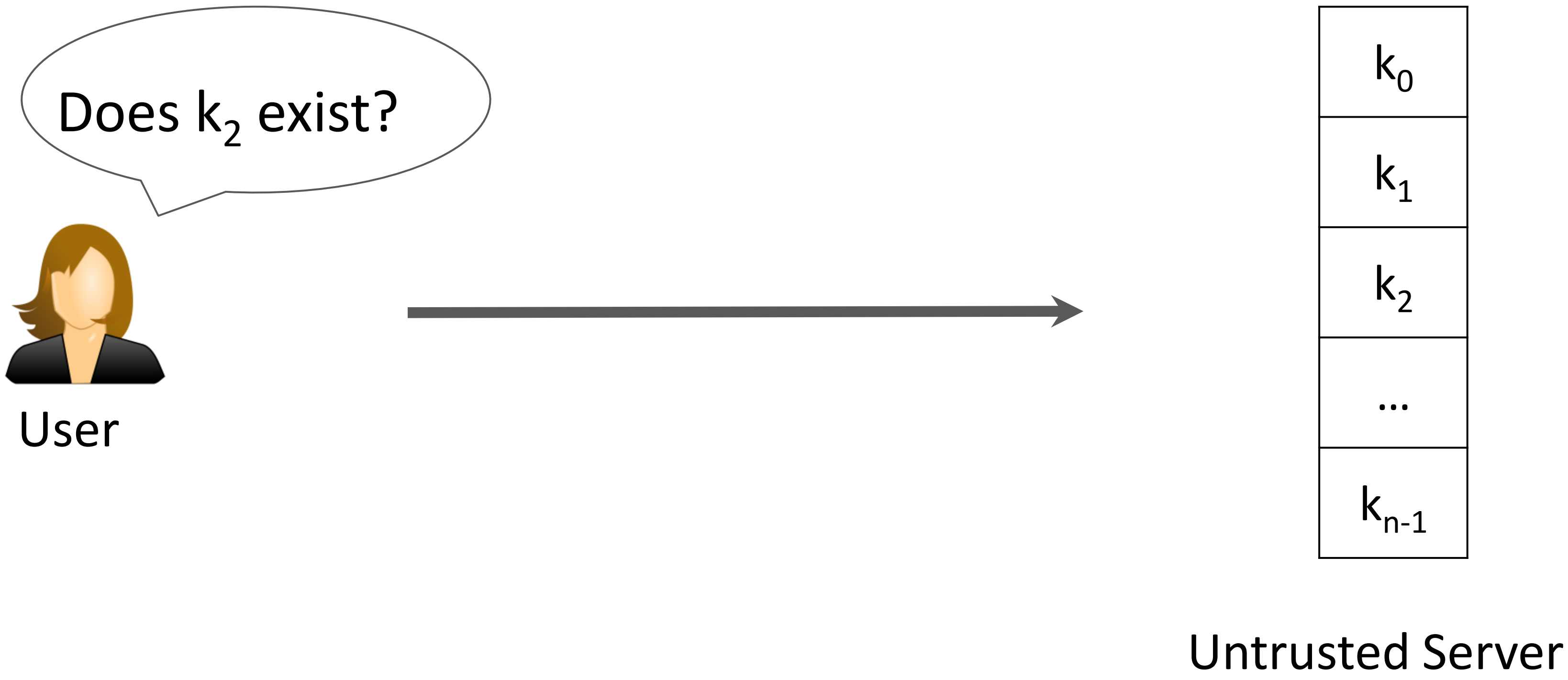




# of Rounds: height of tree

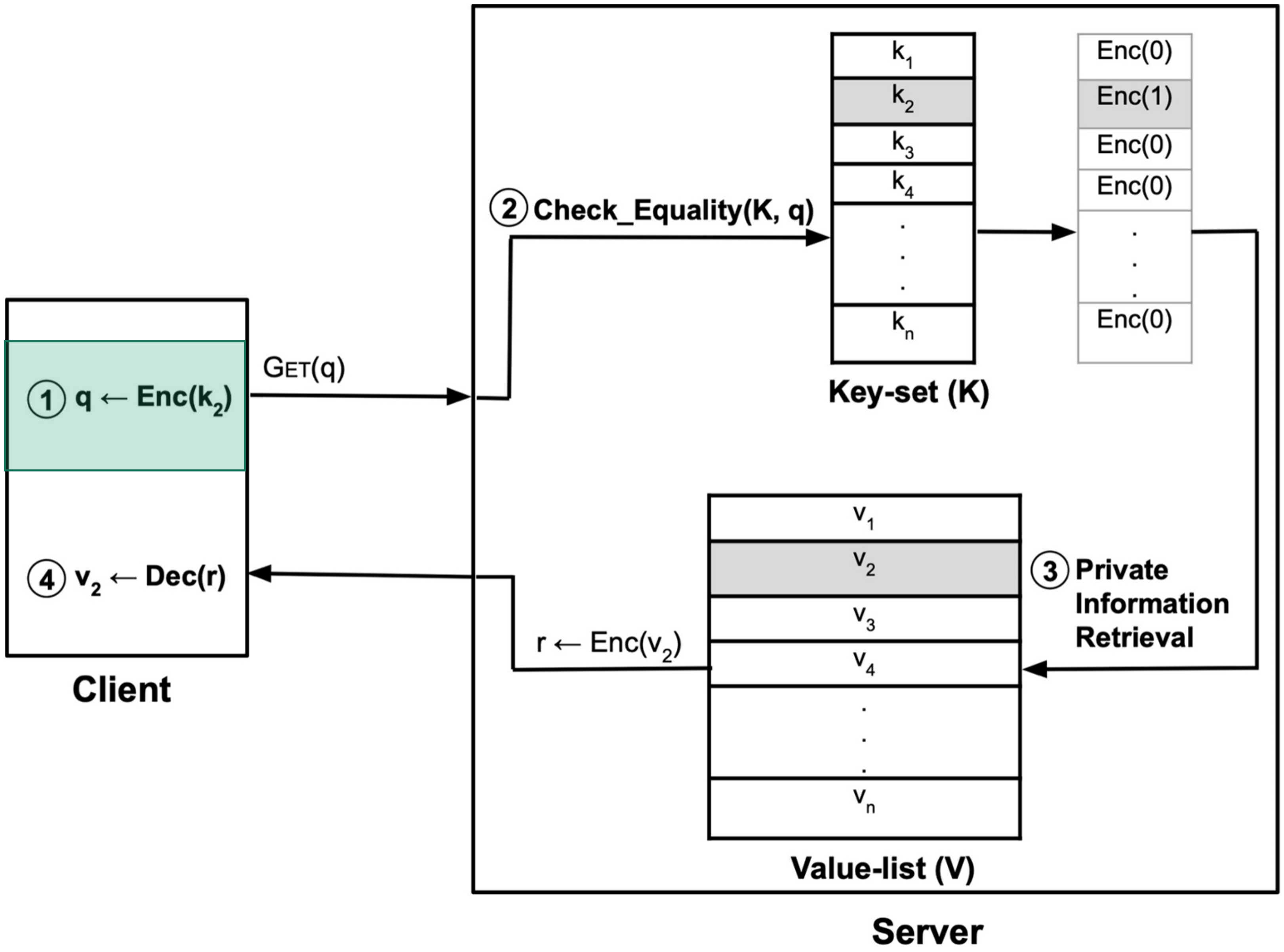
Untrusted Server

Pantheon: A single round approach for PIR-by-keywords

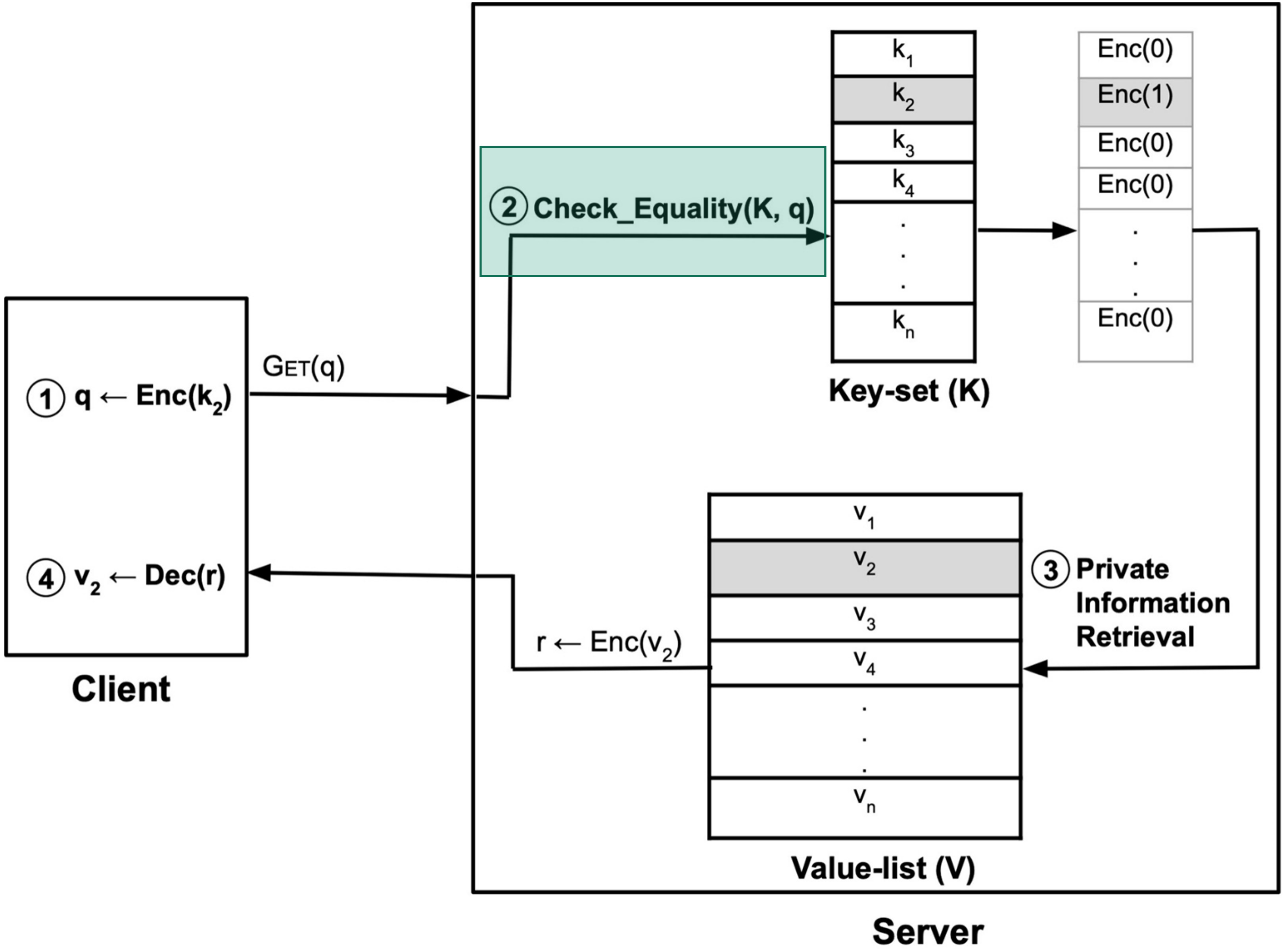


- *Can we make the checking single-round?*
- *Can we make the user independent of the number of keys (n)?*

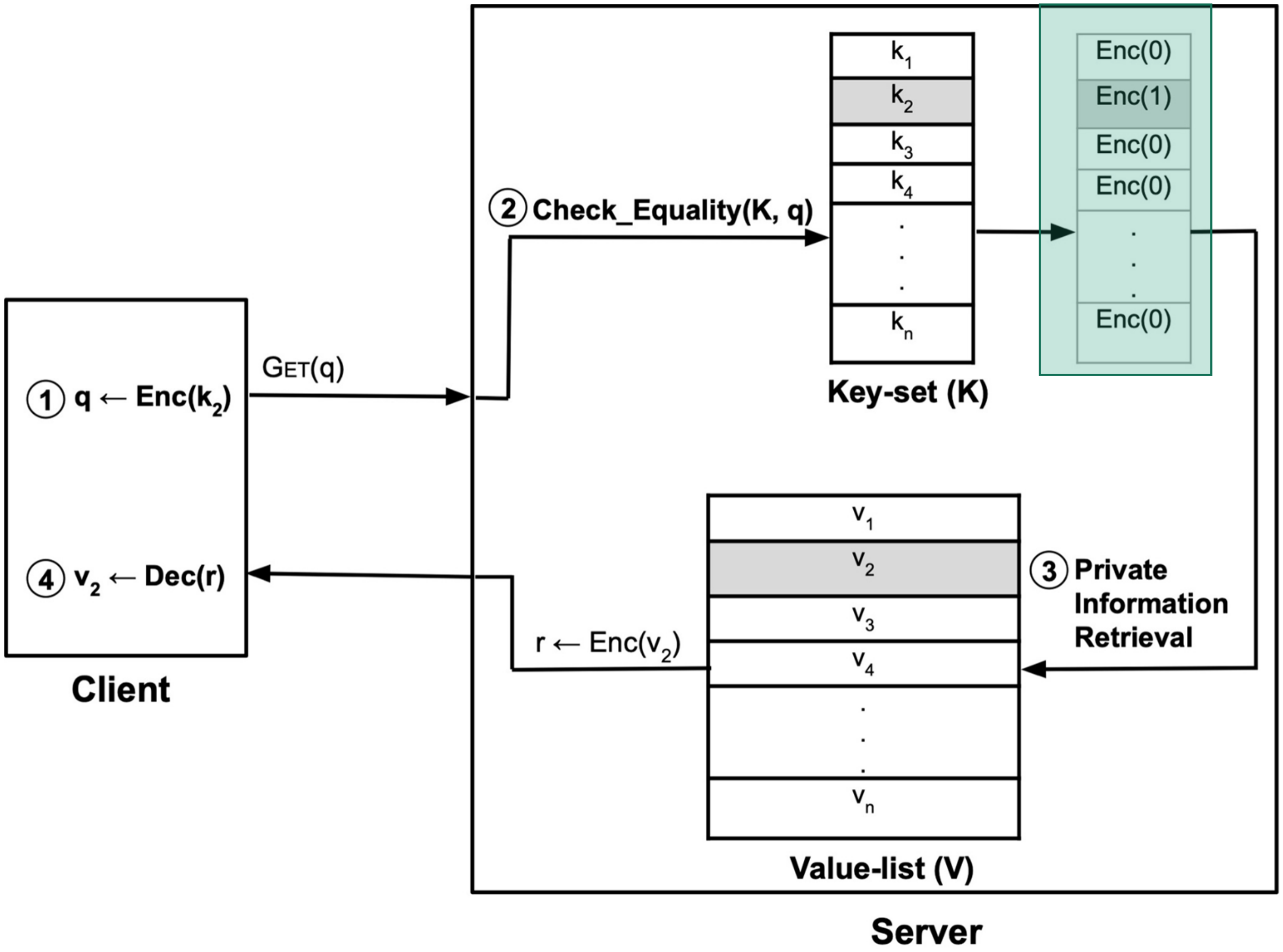
Pantheon: A single round approach



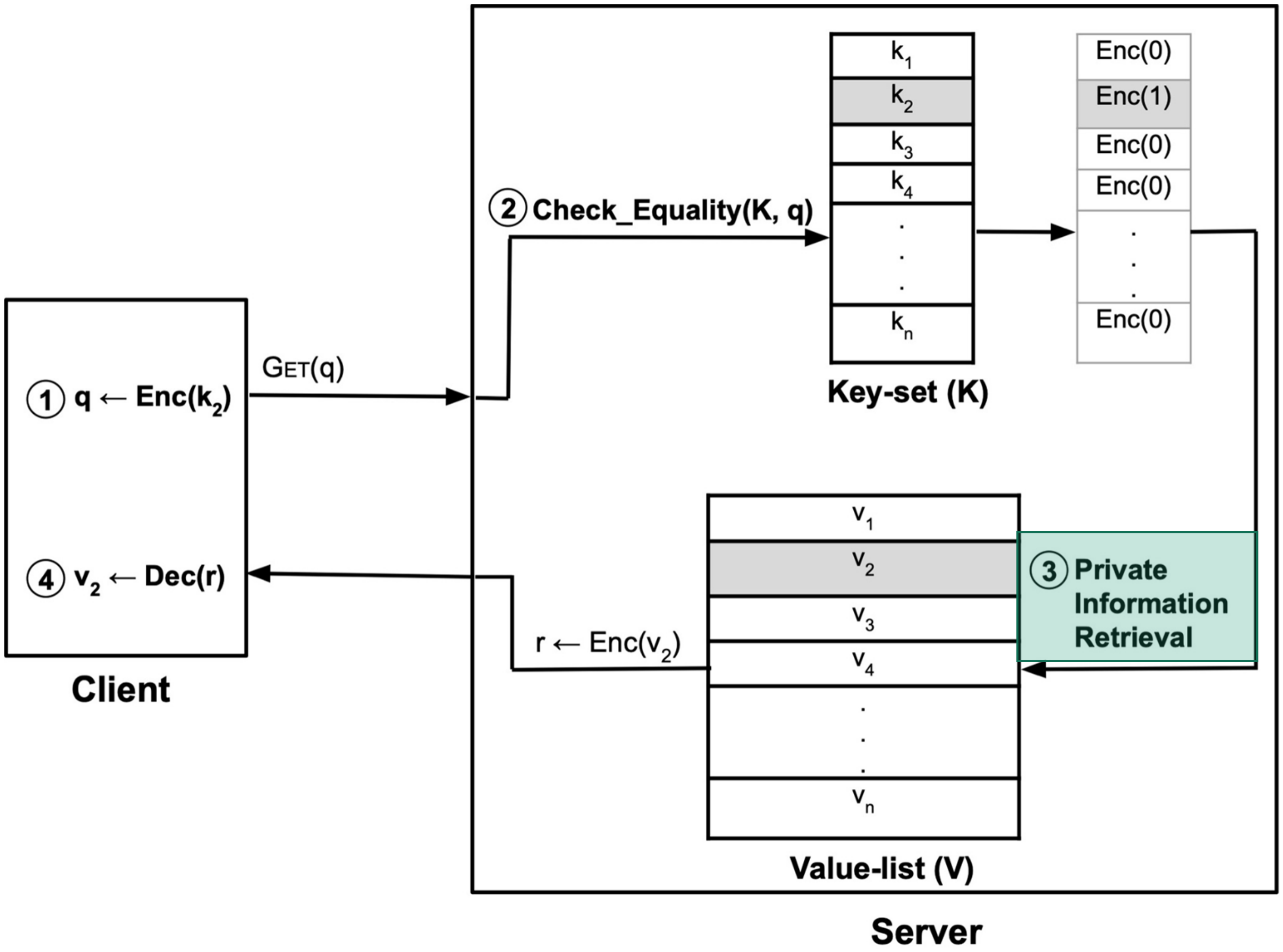
Pantheon: A single round approach



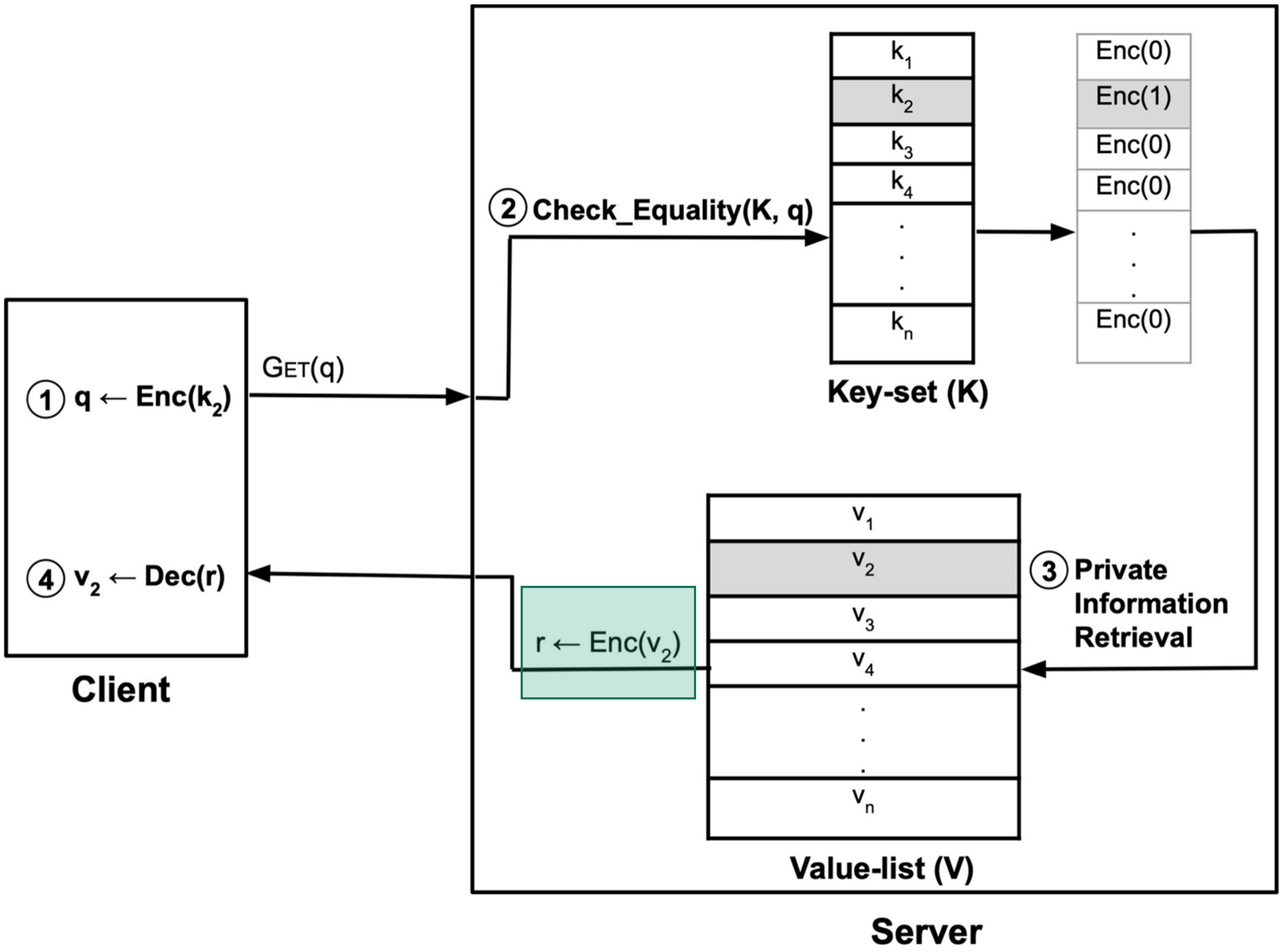
Pantheon: A single round approach



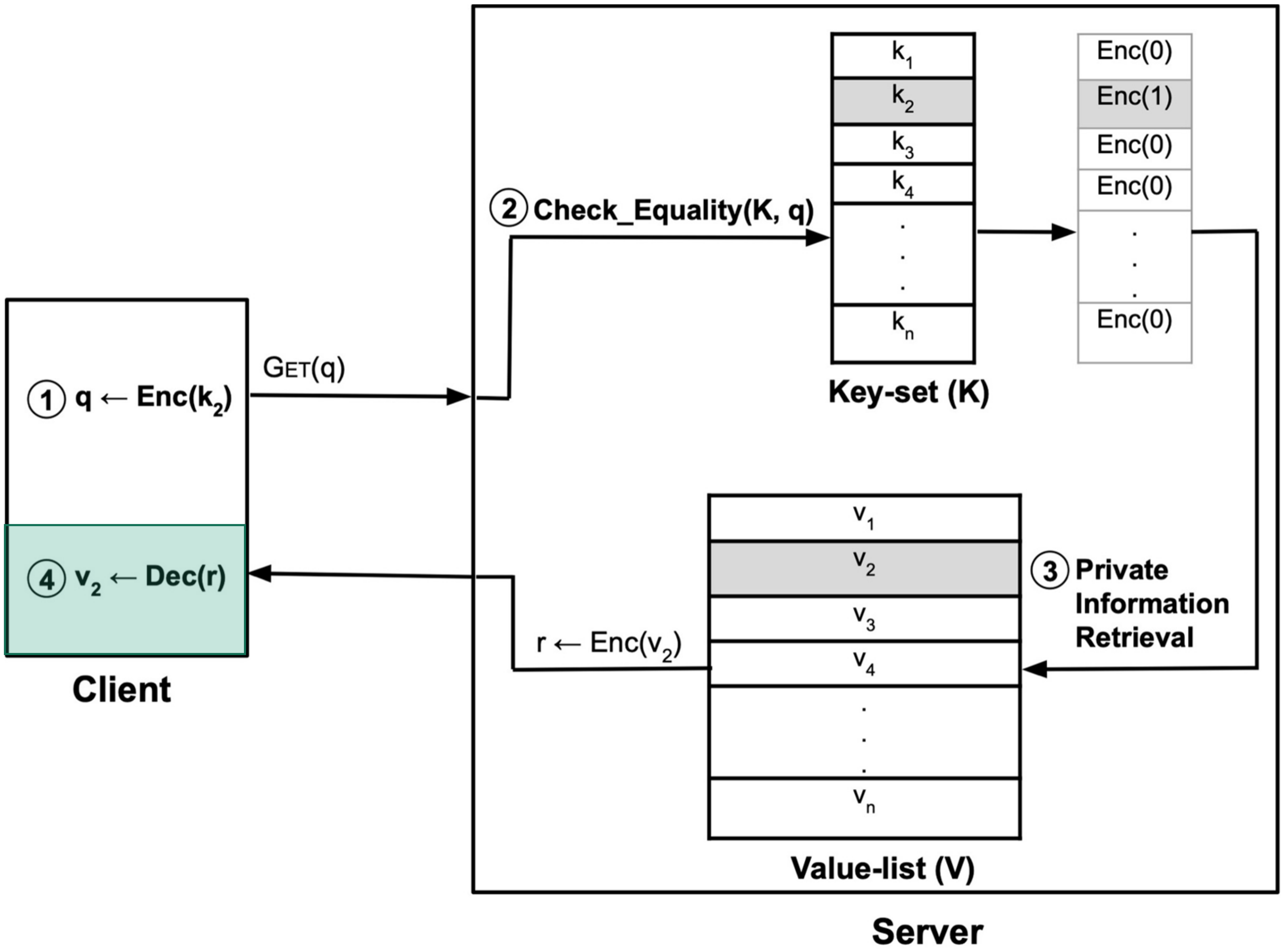
Pantheon: A single round approach



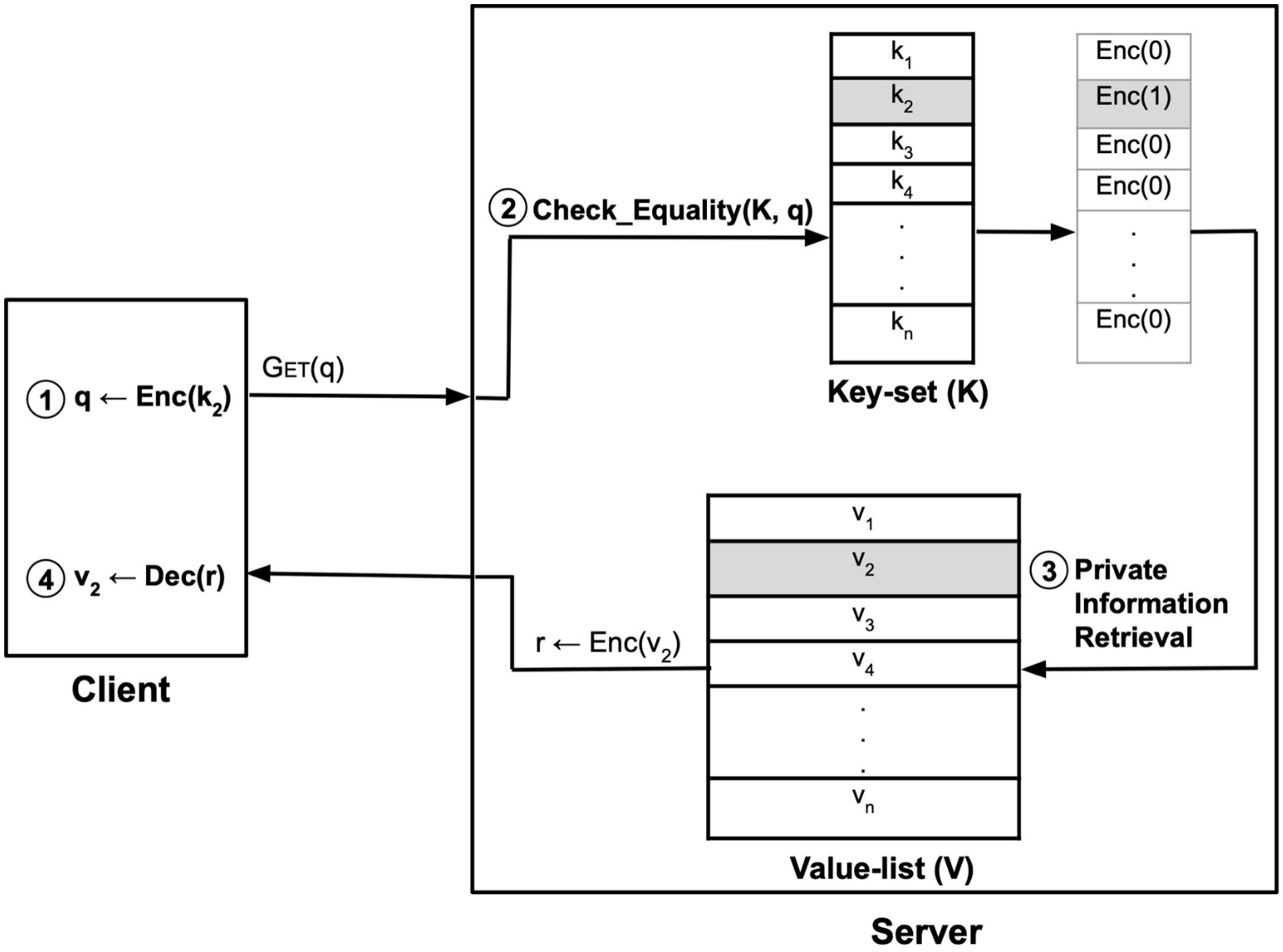
Pantheon: A single round approach



Pantheon: A single round approach



Pantheon: A single round approach



Key challenge:

Checking equality obliviously

Pantheon: An efficient and scalable solution

A new approach for homomorphic equality check

- A number theoretic approach based on Fermat's little theorem
- if p is a prime number and a is a non-zero number not divisible by p, then,
  - $a^{(p-1)} \bmod p \equiv 1$
  - Otherwise, if a is zero, then $a^{(p-1)} \bmod p \equiv 0$

Enables to distinguish between zero and non-zero value!

Oblivious equality checking

Step 1: Subtraction — *Step 2: Exponentiation* — *Step 3: Complement*

query		K			Fermat		1's complement	
12	−	4	=	8	→	1	→	0
12		7		5		1		0
12		12		0		0		1
12		10		2		1		0

Additional techniques to achieve scalability

- SIMD batching to reduce homomorphic multiplications
- Optimal parameter selection to reduce homomorphic exponentiation cost
- Parallelization → vertical scalability
- Coordinator-worker deployment → horizontal scalability
- Query compression to reduce bandwidth usage

Details available in the paper!

Pantheon: An efficient and scalable solution

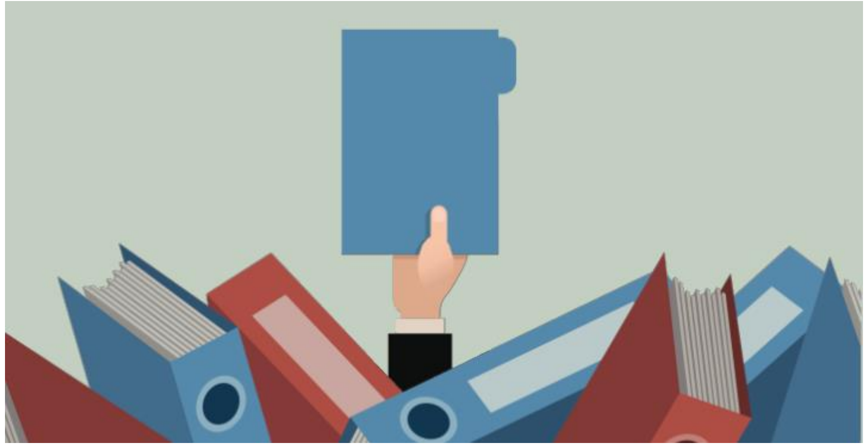
For more details, please check VLDB 2023 paper

And next Tuesday Lecture

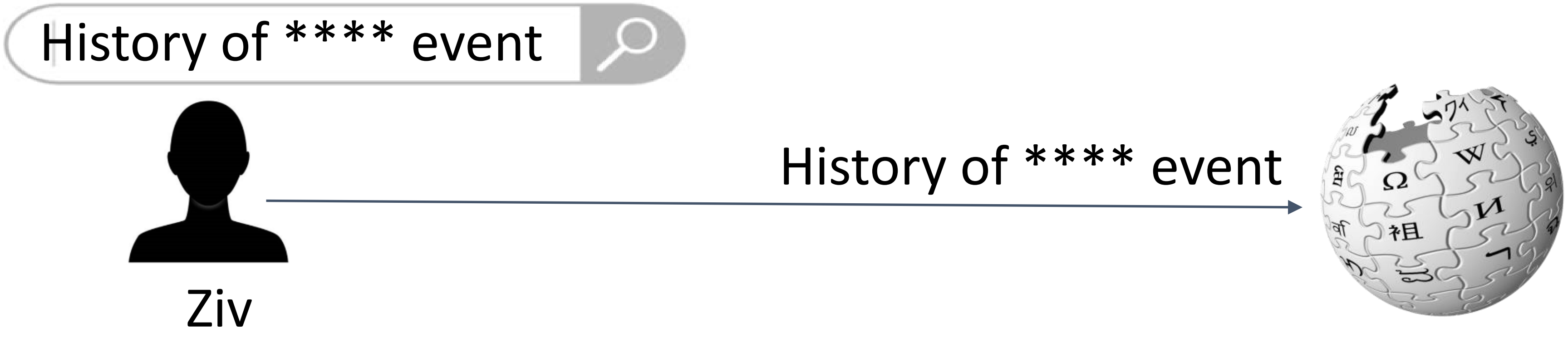
Ahmad, Agrawal, El Abbadi, Gupta

Pantheon: Private Retrieval from Public Key-Value Store.

What if we don't know which document we are interested in, and want to retrieve the top-k qualifying documents obliviously from a public repository?



Information on search keywords or retrieved document leaks privacy



Information on search keywords or retrieved document leaks privacy

Information on search keywords or retrieved document leaks privacy

Ziv

Ranked list

matched doc1
matched doc2
matched doc3
…
…

Simple IR with ranking in one round of communication

Use term frequency-Inverse document frequency (tf-idf) from Information Retrieval (IR)

“red apple”

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

Simple IR with ranking in one round of communication

"red apple"

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

Simple IR with ranking in one round of communication

“red apple”

1
0
1
…

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**….**
Doc1	0.5	0.2	0	…
Doc2	0.8	0.1	0.1	…
Doc3	0	0	0.6	…
…	…	…	…	…
…	…	…	…	…

Simple IR with ranking in one round of communication

matrix-vector multiplication

1
0
1
...

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

Simple IR with ranking in one round of communication

Score1
Score2
Score3
…
…

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**….**
Doc1	0.5	0.2	0	…
Doc2	0.8	0.1	0.1	…
Doc3	0	0	0.6	…
…	…	…	…	…
…	…	…	…	…

Simple IR with ranking in one round of communication

Server picks top-k scores

$\text{idx}_1, \ldots., \text{idx}_K$

Document Provider (D)

doc1
doc2
doc3
doc4
...
...
...

Simple IR with ranking in one round of communication

Server retrieve top-k documents

Information Retrieval

$idx_1, \ldots., idx_K$

$D[idx_1], \ldots., D[idx_K]$

$D[idx_1], \ldots., D[idx_K]$

Document Provider (D)

doc1
doc2
doc3
doc4
…
…
…

Simple IR with ranking in one round of communication

Server sends top-k documents to client

D[idx_1],...., D[idx_K]

Document Provider (D)

doc1
doc2
doc3
doc4
...
...
...

Simple IR with ranking in one round of communication

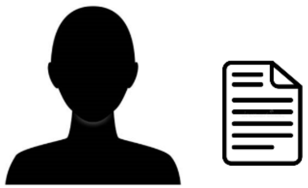
Client reads Relevant Document

D[idx*]

Document Provider (D)

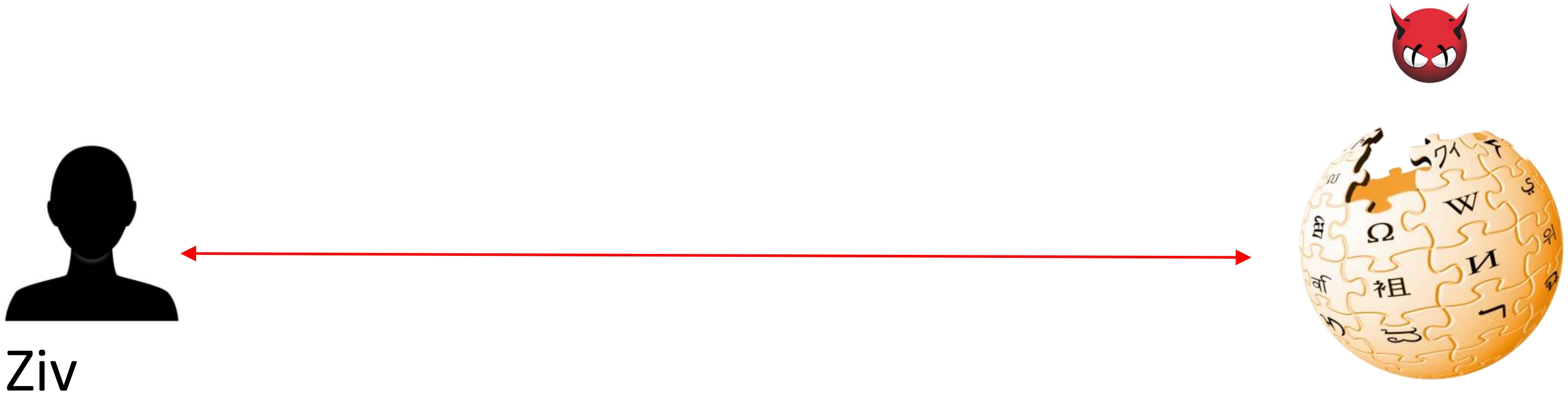
doc1
doc2
doc3
doc4
…
…
…

Information on search keywords or retrieved document leaks privacy



Ziv

Information on search keywords or retrieved document leaks privacy

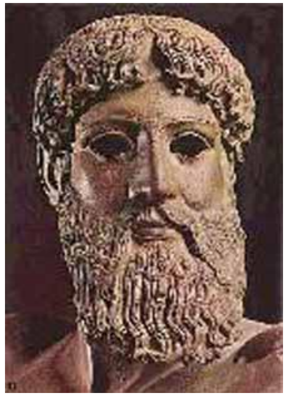


*Can Ziv search and retrieve documents privately **without trusting anybody**?*

Challenges for Privacy

- User query needs to be private & server performs matrix multiplication
  - ➔ Encrypt (using FHE) query vector
- Result scores are encrypted, so server cannot rank
  - ➔ user needs to rank, hence multi round protocol
- Server should not know which documents are retrieved
  - ➔ User Private Information Retrieval (PIR) at server.

Coeus achieves its performance with two key ideas



Idea 1: A novel 3-round protocol

Idea 2: Efficient large-scale secure matrix-vector multiplication

Key challenge:

Need to process the entire state

- Millions of documents of different sizes
  - Example : 5 million Wiki articles
- Hundreds of billions of matrix entries
  - 65,536 keywords
  - Over 327 billion matrix elements

Ranking can be achieved using *homomorphic encryption*

Start with 2 round protocol (adopted from works on private data [1], [2])

Round 1: Query scoring

"red apple"

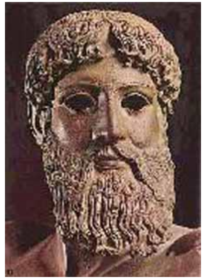
Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

1. M. Strizhov and I. Ray. *Multi-keyword similarity search over encrypted cloud data.* In IFIP international information security conference, 2014.
2. J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li. *Toward secure multikey-word top-k retrieval over encrypted cloud data.* IEEE transactions on dependable and secure computing, 2013

Coeus: A novel 3-round protocol



Round 1: Query scoring

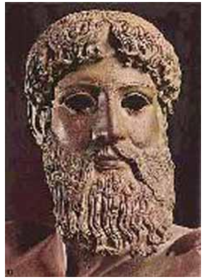
"red apple"

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

Coeus: A novel 3-round protocol



Round 1: Query scoring

"red apple"

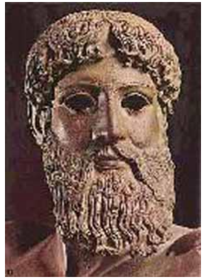
1
0
1
...

Query Scorer

tf-idf matrix

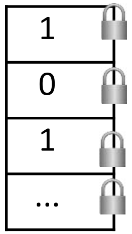
	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

Coeus: A novel 3-round protocol



Round 1: Query scoring

"red apple"

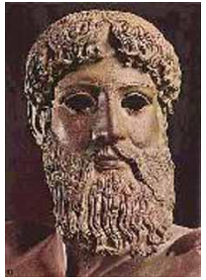


Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

Coeus: A novel 3-round protocol



Round 1: Query scoring

Secure matrix-vector multiplication

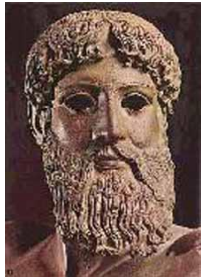
1
0
1
...

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	...
Doc2	0.8	0.1	0.1	...
Doc3	0	0	0.6	...
...	...	...	...	...
...	...	...	...	...

Coeus: A novel 3-round protocol



Round 1: Query scoring

Secure matrix-vector multiplication

Enc(Score1)
Enc(Score2)
Enc(Score3)
…
…

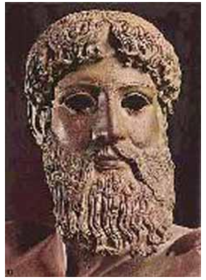
Enc(Score1)
Enc(Score2)
Enc(Score3)
…
…

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**....**
Doc1	0.5	0.2	0	…
Doc2	0.8	0.1	0.1	…
Doc3	0	0	0.6	…
…	…	…	…	…
…	…	…	…	…

Coeus: A novel 3-round protocol



Round 1: Query scoring

Score1
Score2
Score3
…
…

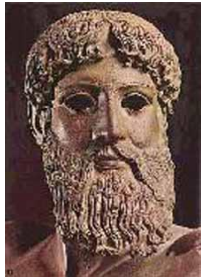
Secure matrix-vector multiplication

Query Scorer

tf-idf matrix

	apple	**bat**	**red**	**….**
Doc1	0.5	0.2	0	…
Doc2	0.8	0.1	0.1	…
Doc3	0	0	0.6	…
…	…	…	…	…
…	…	…	…	…

Coeus: A novel 3-round protocol

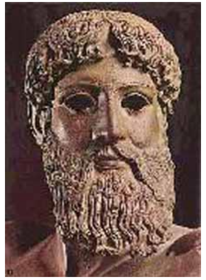


Round 1: Query scoring

Pick top-k highest scores

idx_1,...., idx_K

Coeus: A novel 3-round protocol



Round 2: Document Retrieval

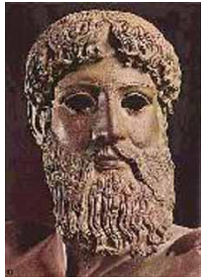
Enc(idx_1,...., idx_K)

Enc(idx_1,...., idx_K)

Document Provider (D)

doc1
doc2
doc3
doc4
...
...
...

Coeus: A novel 3-round protocol



Round 2: Document Retrieval

- **Multi-retrieval PIR**, extension of PIR to download k elements privately in a single retrieval.

Private Information Retrieval

Enc(D[idx_1],...., D[idx_K])

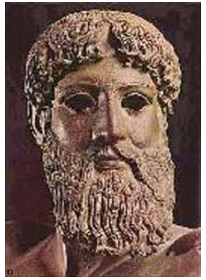
Enc(idx_1,...., idx_K)

Enc(D[idx_1],...., D[idx_K])

Document Provider (D)

doc1
doc2
doc3
doc4
...
...
...

Coeus: A novel 3-round protocol



Private Information Retrieval

Document Provider (D)

doc1	
doc2	
doc3	doc3
doc4	doc4
…	
…	
…	

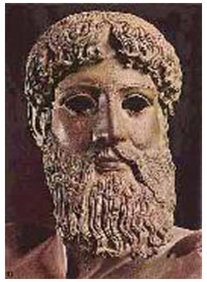
Round 2: Document Retrieval

$\text{Enc}(\text{idx}_1, \ldots, \text{idx}_K)$

$\text{Enc}(\text{D}[\text{idx}_1], \ldots, \text{D}[\text{idx}_K])$

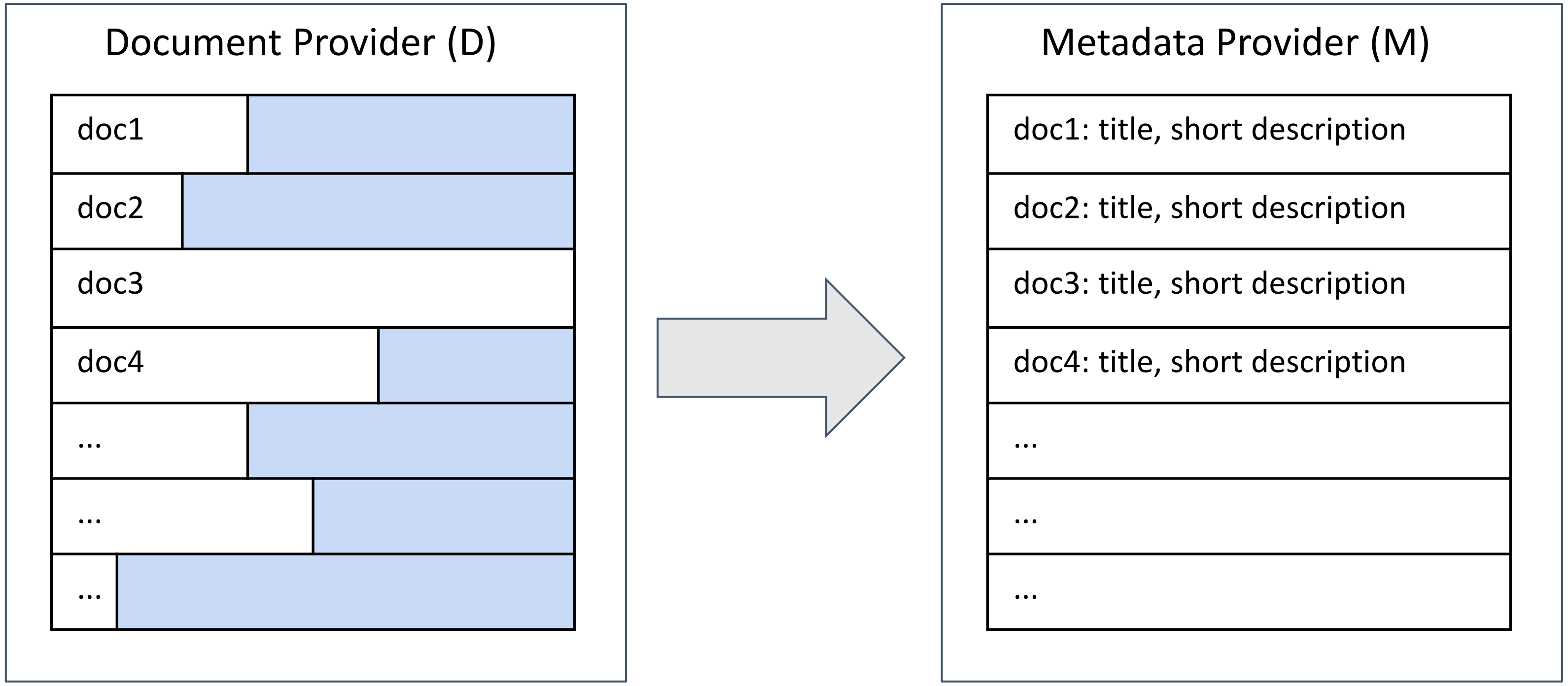
Issue 1: Retrieval of all Top-k documents is expensive

Issue 2: Documents need to be padded to the largest size for privacy



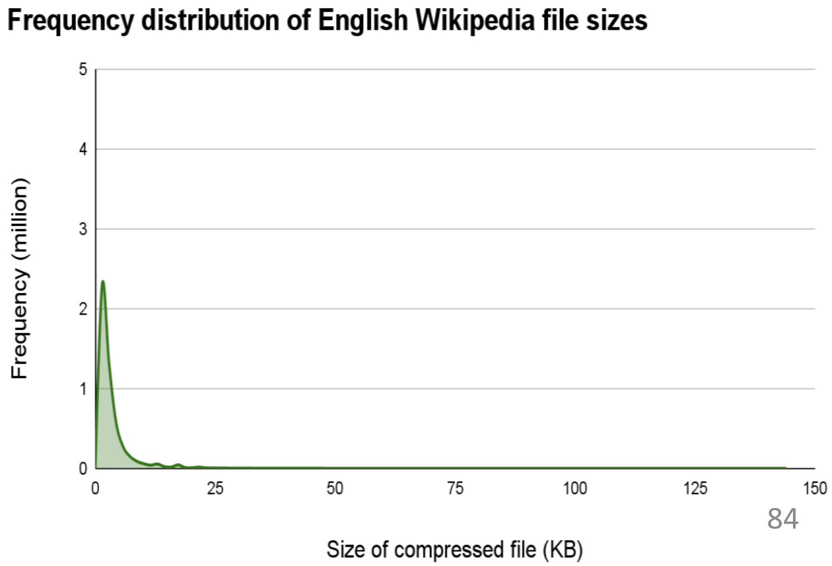
A novel 3-round protocol

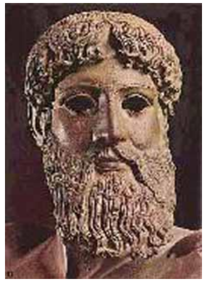
Intodúce Metadata Abstract Summary



Wikipedia's Short Descriptions,

- Concise explanation of page.
- Used in Wikipedia mobile
- Helps identify desired article.
- Title: 255 Bytes
- Description: 40 Bytes

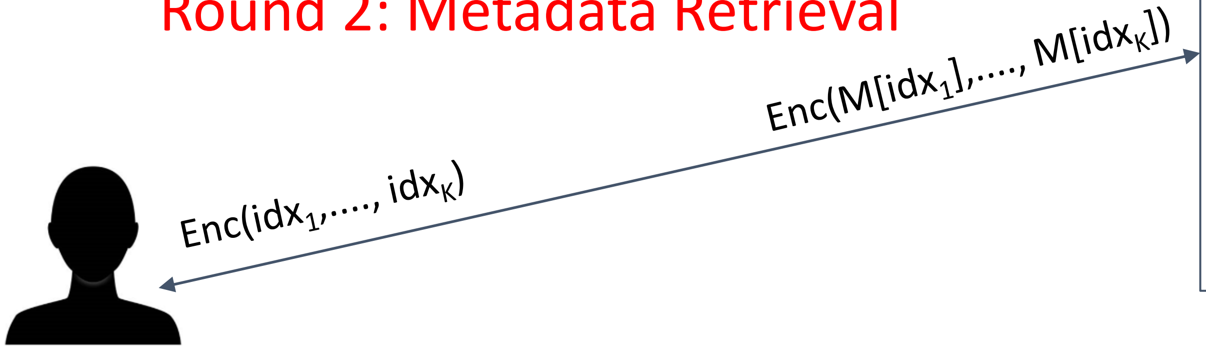




A novel 3-round protocol

Round 2: Document Retrieval

Round 2: Metadata Retrieval



Metadata Provider (M)

doc1: title, short description
doc2: title, short description
doc3: title, short description
doc4: title, short description
…
…
…

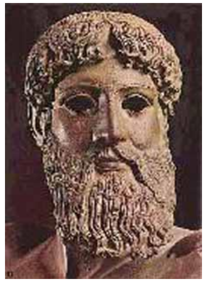
Document Provider (D)

doc1
doc2
doc3
doc4
…
…
…

Issue 1: Multi-retrieval costlier than single retrieval

Solution: * One Round for Multi-retrieval of metadata,

* One Round for retrieval of single document



A novel 3-round protocol

Round 3: Document Retrieval

Enc(idx*)

Enc(D[idx*])

Metadata Provider (M)

doc1: title, short description
doc2: title, short description
doc3: title, short description
doc4: title, short description
…
…
…

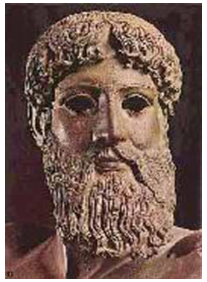
Document Provider (D)

doc1
doc2
doc3
doc4
…
…
…

Issue 1: Multi-retrieval costlier than single retrieval

Solution: * One Round for Multi-retrieval for metadata,

* One Round for retrieval of single document



A novel 3-round protocol

Metadata Provider (M)

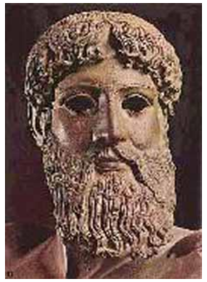
doc1: title, short description
doc2: title, short description
doc3: title, short description
doc4: title, short description
…
…
…

Document Provider (D)

doc1
doc2
doc3
doc4
…
…
…

Issue 2: Documents padded to largest size

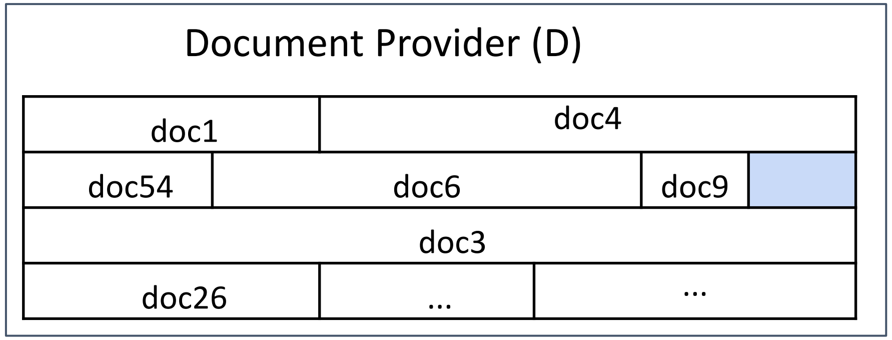
Solution: Bin-packing reduces padding



A novel 3-round protocol

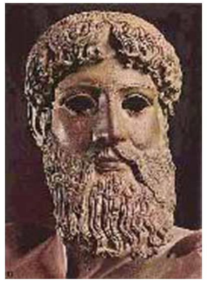
Metadata Provider (M)

doc1: title, short description
doc2: title, short description
doc3: title, short description
doc4: title, short description
…
…
…



Issue 2: Documents padded to largest size

Solution: Bin-packing reduces padding



A novel 3-round protocol

Add location in Metadata

In experiments:

- Padding size: 670.8GiB
- Packing size: 13.1 GiB
- ~50x compaction

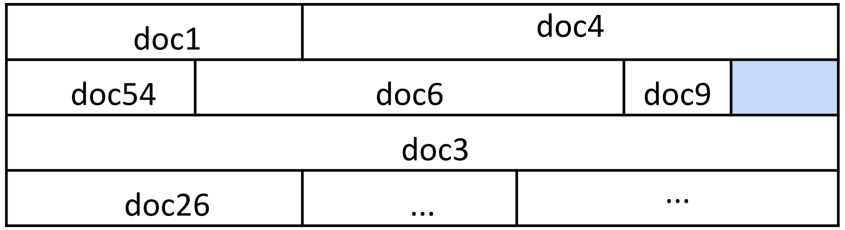
Issue 2: Documents padded to largest size

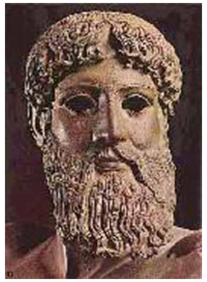
Solution: Bin-packing reduces padding

Metadata Provider (M)

doc1: title, short description, location
doc2: title, short description, location
doc3: title, short description, location
doc4: title, short description, location
…
…
…

Document Provider (D)





Efficient large-scale secure matrix multiplication

Issue: Matrix contains several hundred billion elements

Solution: Optimizations on top of Halevi-Shoup algorithm

- Remove redundant calls to expensive homomorphic rotation
- Re-order operations to amortize cost of rotations
- Efficiently distribute the workload among a cluster of machines

Check our SOSP 2021 paper for more details. Next Tuesday Lecture

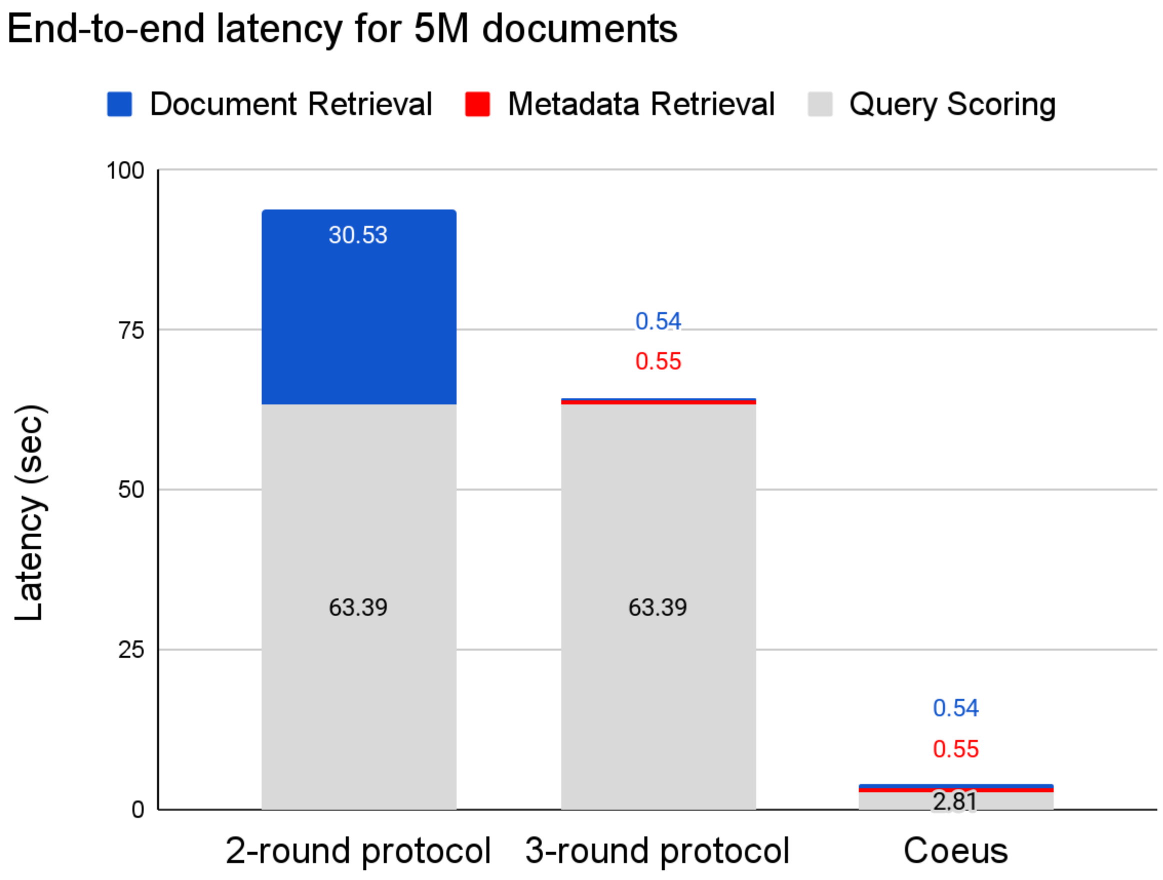
Ahmad, Sarker, Agrawal, El Abbadi, Gupta

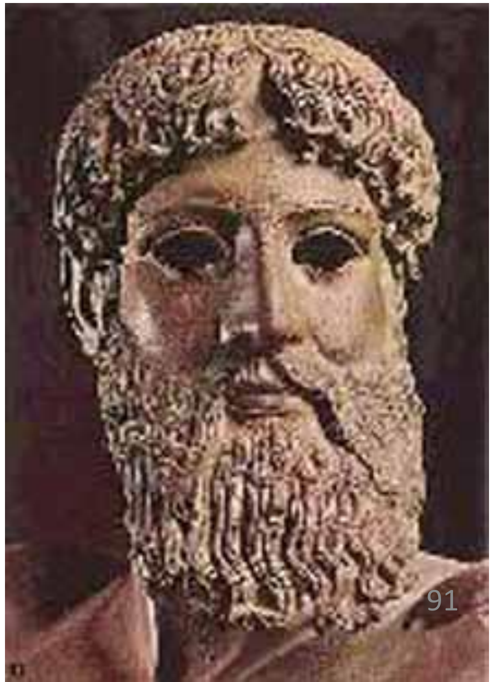
Coeus: A System for Oblivious Document Ranking and Retrieval.

Coeus achieves its performance with two key ideas

A novel 3-round protocol

An efficient algorithm for large-scale secure matrix-vector multiplication

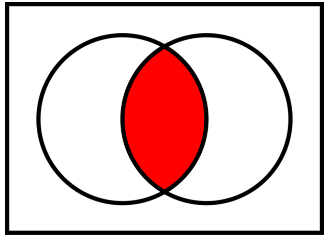




Conclusion

- Private Access of Private and Public Data
- Scalability
- Fault-tolerance
- Practical
- Efficient
- Expressive

QuORAM - Efficient Replication



- QuORAM optimizes the propagate phase by caching blocks in IncompleteCache
- **Excess Blocks - Blocks that TaoStore would have deleted but QuORAM can't delete because they have ongoing operations**
- Challenge: Proxy's memory can theoretically grow to be unbounded!
  - Background Daemon that all blocks at a preset interval.
  - Theoretically: prove stash size is log(db size) bounded.
  - Experimentally: size < log(db size)