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 ongoing works
Outsourced Private Data

Security Concerns?
Confidentiality of Data
Encryption

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

The 2 requests read the same block

If query is a search for a certain drug

Alice and Bob have the same disease: reveals medical condition

Alice
Bob
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.
Let's Start with **Private User Data**
Oblivious RAM

what's the opposite of oblivious?

aware, conscious, mindful, attentive, sensitive, concerned, observant, heedful, cognizant, conversant
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

A =

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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

\[
\begin{array}{c|c|c|c|c}
\hline
  & 0 & v & 0 & 0 \\
\hline
\end{array}
\]

1. The cloud never learns value \( v \)
2. The cloud never learns index 1
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

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])

Step 1. Read path

- Physical read entire path
- Read or write
- Proxy

Step 2. Shuffle and Write path

- Shuffle and physical write to random path
- Proxy

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, LR’15, TM’15, SD’16, ...
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

Current ORAM data stores are not fault tolerant

Loose access to entire data!!


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

- No liveness: single point of failure when proxy crashes

QuORAM system and threat model

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

QuORAM’s 3-fold goal:

- Hide access pattern
- Replicate data
- Provide linearizable semantics
Solution 1: Efficient replication protocol

- Protocols such as Virtual Partitions [1,2], Hermes[3] or CRAQ[4] are read-optimized.
  - Read from: one replica
  - Write to: ALL (available) replicas

- Number of message exchanges reveal type of operation!!

---

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

- Single round
  - Read from: a majority
  - Write to: a majority
- Note: majority sets must intersect
- Concurrent & conflicting operations may violate linearizability!!

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

Write \((x = 10)\) 
Read \(x\) twice

ORAM Unit 1
ORAM Unit 2
ORAM Unit 3

\((x = 5)\) 
\((x = 5)\) 
\((x = 5)\)
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 $\rightarrow$ two ORAM requests

QuORAM Replication

Query Phase

Write Phase

Logical Read $x$ or Write $x = v'$

Read $x$

Writes: Increment highest tag $t'$ and update value to $v'$

$x: v, t = v', t'$

acks
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

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

- Store \(<request \text{ id: path id}>\) 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

![Graph showing throughput vs. concurrent clients for different database configurations]

- QuORAM
- Insecure
- No Replication
- CockroachDB
We can have Privacy AND Fault-tolerance!
But, much of the content on the Internet is in *public data repositories*.

**User**

I want to stream “The Godfather”

Remote server

**User**

Show me the latest post by Elon Musk

Remote server
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]

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

History of pride parade

Cannot use:
- Encryption
- ORAM
- CryptDB-like solution
Private Information Retrieval: Retrieval by Location

Client has (key, location) mapping

Give me the $i$-th value

Untrusted Server

<table>
<thead>
<tr>
<th>$k_0$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>1</td>
</tr>
<tr>
<td>$k_2$</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$k_{n-1}$</td>
<td>$n-1$</td>
</tr>
</tbody>
</table>

| 0 | $v_0$ |
| 1 | $v_1$ |
| 2 | $v_2$ |
| ... | ... |
| $n-1$ | $v_{n-1}$ |

*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)

User

I want $db[i]$

$q = \text{Query}(i)$

Untrusted Server

$q$

$ans$

$$
\begin{align*}
\text{db} & \\
0 & : 0101100101010100101010 \\
1 & : 0111000101010100101010 \\
2 & : 1001100101010100101010 \\
\vdots & \\
n-1 & : 0101101101010100101110 \\
\end{align*}
$$
One trivial solution to private information retrieval

User

q = Give me the entire db

ans = db

I want db[i]

Untrusted Server

| 0  | 010110001010100100101010 |
| 1  | 011100010101010010101010 |
| 2  | 100110010101010010101010 |
| ... | ... |
| n-1 | 011010011101011010101010 |

db
Retrieval is equivalent to computing a dot product
Basic Idea in CPIR

Query from Client

DB at Untrusted Server

Intermediate Result

Add

Final Result Back to Client
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

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.

<table>
<thead>
<tr>
<th>Overhead</th>
<th>High-level technique</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request size</strong></td>
<td>• Recursion (Stern 1998)</td>
</tr>
<tr>
<td></td>
<td>• Cryptographic query compression (SealPIR ‘18)</td>
</tr>
<tr>
<td><strong>Server-side compute</strong></td>
<td>• PIR with preprocessing (Beimel et al. ‘00, SimplePIR ‘23)</td>
</tr>
<tr>
<td></td>
<td>• Lattice-based cryptography (FastPIR ‘21)</td>
</tr>
</tbody>
</table>
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

Ahmad, Yang, Agrawal, El Abbadi, Gupta
Addra: Metadata-private voice communication over fully untrusted infrastructure.
Retrieval by key

I’m interested in the value of key $k$

Client retrieves:
- $v$, if $(k,v)$ at Server
- $\emptyset$, otherwise

Give me value for key $k$

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\}

Does 17 exist?

# of Rounds: height of tree
Pantheon: A single round approach for PIR-by-keywords

- Does $k_2$ exist?

- 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

1. $q \leftarrow \text{Enc}(k_2)$

2. **Check Equality**($K, q$)

   - $k_1$
   - $k_2$
   - $k_3$
   - $k_4$
   - ...
   - $k_n$

3. **Private Information Retrieval**

   - $\text{Enc}(0)$
   - $\text{Enc}(1)$
   - $\text{Enc}(0)$
   - $\text{Enc}(0)$
   - ...
   - $\text{Enc}(0)$

4. $v_2 \leftarrow \text{Dec}(r)$

Client

Server

$\text{Get}(q)$

$r \leftarrow \text{Enc}(v_2)$
Pantheon: A single round approach
Pantheon: A single round approach

1. \( q \leftarrow \text{Enc}(k_2) \)
2. \( \text{Check\_Equality}(K, q) \)
3. \( r \leftarrow \text{Enc}(v_2) \)
4. \( v_2 \leftarrow \text{Dec}(r) \)

Key-set (K)

Value-list (V)

Private Information Retrieval
Pantheon: A single round approach

1. $q \leftarrow \text{Enc}(k_2)$
2. Check\_Equality($K$, $q$)
3. Private Information Retrieval
4. $v_2 \leftarrow \text{Dec}(r)$

Key-set ($K$)

Value-list ($V$)

Client

Server
Pantheon: A single round approach

1. $q \leftarrow \text{Enc}(k_2)$

2. Check_Equality($K$, $q$)

3. Private Information Retrieval

4. $v_2 \leftarrow \text{Dec}(r)$

Key-set ($K$)

Value-list ($V$)

Server

Client
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,
  - $a^{(p-1)} \mod p \equiv 1$
  - Otherwise, if $a$ is zero, then $a^{(p-1)} \mod p \equiv 0$

Enables to distinguish between zero and non-zero value!
Oblivious equality checking

Step 1: Subtraction

query
12
12
12
12

-  
K
4
7
12
10

=  

Step 2: Exponentiation

8
5
0
2

Step 3: Complement

1
0
0
1

Fermat

1’s complement

0
0
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

<table>
<thead>
<tr>
<th>matched doc1</th>
</tr>
</thead>
<tbody>
<tr>
<td>matched doc2</td>
</tr>
<tr>
<td>matched doc3</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Simple IR with ranking in one round of communication

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

“red apple”
Simple IR with ranking in one round of communication

“red apple”

<table>
<thead>
<tr>
<th>Query Scorer</th>
<th>tf-idf matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>apple</td>
</tr>
<tr>
<td>Doc1</td>
<td>0.5</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Simple IR with ranking in one round of communication

“red apple”

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>bat</th>
<th>red</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Query Scorer

tf-idf matrix
Simple IR with ranking in one round of communication

matrix-vector multiplication

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>bat</th>
<th>red</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>...</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Simple IR with ranking in one round of communication

Query Scorer

tf-idf matrix

<table>
<thead>
<tr>
<th>apple</th>
<th>bat</th>
<th>red</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Simple IR with ranking in one round of communication

Server picks top-\(k\) scores

\[\text{idx}_1, \ldots, \text{idx}_K\]
Simple IR with ranking in one round of communication
Simple IR with ranking in one round of communication

Server sends top-k documents to client

D[\text{idx}_1],..., D[\text{idx}_K]
Simple IR with ranking in one round of communication

Client reads Relevant Document

D[idx*]
Information on search keywords or retrieved document leaks privacy
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”

| Doc | 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”

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>bat</th>
<th>red</th>
<th>....</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>...</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Coeus: A novel 3-round protocol

Round 1: Query scoring
“red apple”

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>bat</th>
<th>red</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>...</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Query Scorer
tf-idf matrix
Coeus: A novel 3-round protocol

Round 1: Query scoring

“red apple”

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>bat</th>
<th>red</th>
<th>....</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>...</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Coeus: A novel 3-round protocol

Round 1: Query scoring

Secure matrix-vector multiplication

Query Scorer

tf-idf matrix

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>bat</th>
<th>red</th>
<th>....</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>...</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Coeus: A novel 3-round protocol

Round 1: Query scoring

Secure matrix-vector multiplication

<table>
<thead>
<tr>
<th>Query Scorer</th>
<th>tf-idf matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>bat</td>
</tr>
<tr>
<td>Doc1</td>
<td>0.5</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Coeus: A novel 3-round protocol**

**Round 1: Query scoring**

<table>
<thead>
<tr>
<th>Score1</th>
<th>Score2</th>
<th>Score3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**Secure matrix-vector multiplication**

**Query Scorer**

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>bat</th>
<th>red</th>
<th>....</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>0.5</td>
<td>0.2</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>Doc2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>...</td>
</tr>
<tr>
<td>Doc3</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**tf-idf matrix**
Coeus: A novel 3-round protocol

Round 1: Query scoring
Pick top-k highest scores

$\text{idx}_1, \ldots, \text{idx}_K$
Coeus: A novel 3-round protocol

Round 2: Document Retrieval

Document Provider (D)

<table>
<thead>
<tr>
<th>doc1</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc2</td>
</tr>
<tr>
<td>doc3</td>
</tr>
<tr>
<td>doc4</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Round 2: Document Retrieval

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

\[
\text{Enc}(D[idx_1], \ldots, D[idx_K])
\]

**Private Information Retrieval**

**Coeus: A novel 3-round protocol**

**Document Provider (D)**

| doc1 | doc2 | doc3 | doc4 | ...
|------|------|------|------|------
|      |      |      |      |      |
|      |      |      |      |      |
|      |      |      |      |      |
|      |      |      |      |      |
|      |      |      |      |      |
Coeus: A novel 3-round protocol

Round 2: Document Retrieval

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

Introduce 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

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

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

Issue 2: Documents padded to largest size
Solution: Bin-packing reduces padding
A novel 3-round protocol

**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
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 $\log(\text{db size})$ bounded.
  • Experimentally: size < $\log(\text{db size})$