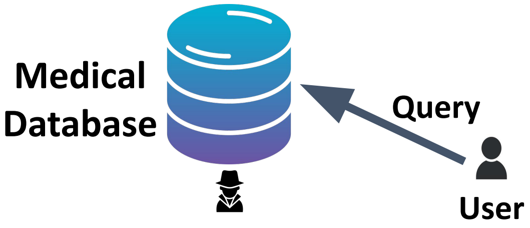
# Private Aggregate Queries to Untrusted Databases

Syed Mahbub Hafiz*, **Chitrabhanu Gupta***, Warren Wnuck, Brijesh Vora, and Chen-Nee Chuah

**University of California, Davis**

# Aggregate Information Retrieval with Privacy

- Motivation: Data provider can observe all queries run on their database by any user, the computations taking place on the server, and which database rows are scanned



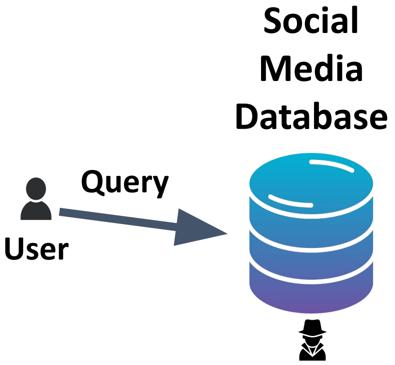
- **Goal:** Retrieve information from an untrusted database without revealing specific queries, even in the presence of *t* colluding database servers

```
SELECT COUNT(user_id)
FROM patients
WHERE is_smoker = 'yes'
AND cancer_flag = 1
```

# Aggregate Information Retrieval with Privacy

- Motivation: Data provider can observe all queries run on their database by any user, the computations taking place on the server, and which database rows are scanned

- **Goal:** Retrieve information from an untrusted database without revealing specific queries, even in the presence of *t* colluding database servers

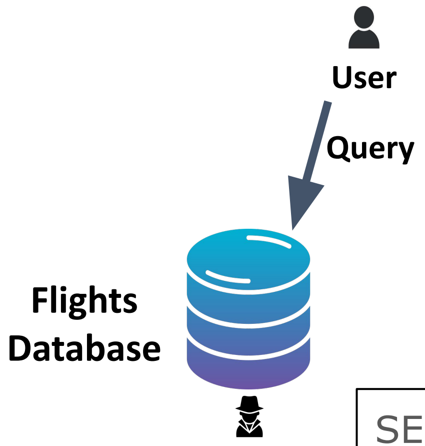


SELECT SUM(***num_likes***)
FROM ***tweets***
WHERE ***user_id*** = **‘20124’**
AND ***date*** ≤ **getdate()**

# Aggregate Information Retrieval with Privacy

- Motivation: Data provider can observe all queries run on their database by any user, the computations taking place on the server, and which database rows are scanned

- **Goal:** Retrieve information from an untrusted database without revealing specific queries, even in the presence of *t* colluding database servers



```
SELECT SUM(price)
FROM flights
WHERE flight_id in
('1120', '4268')
```

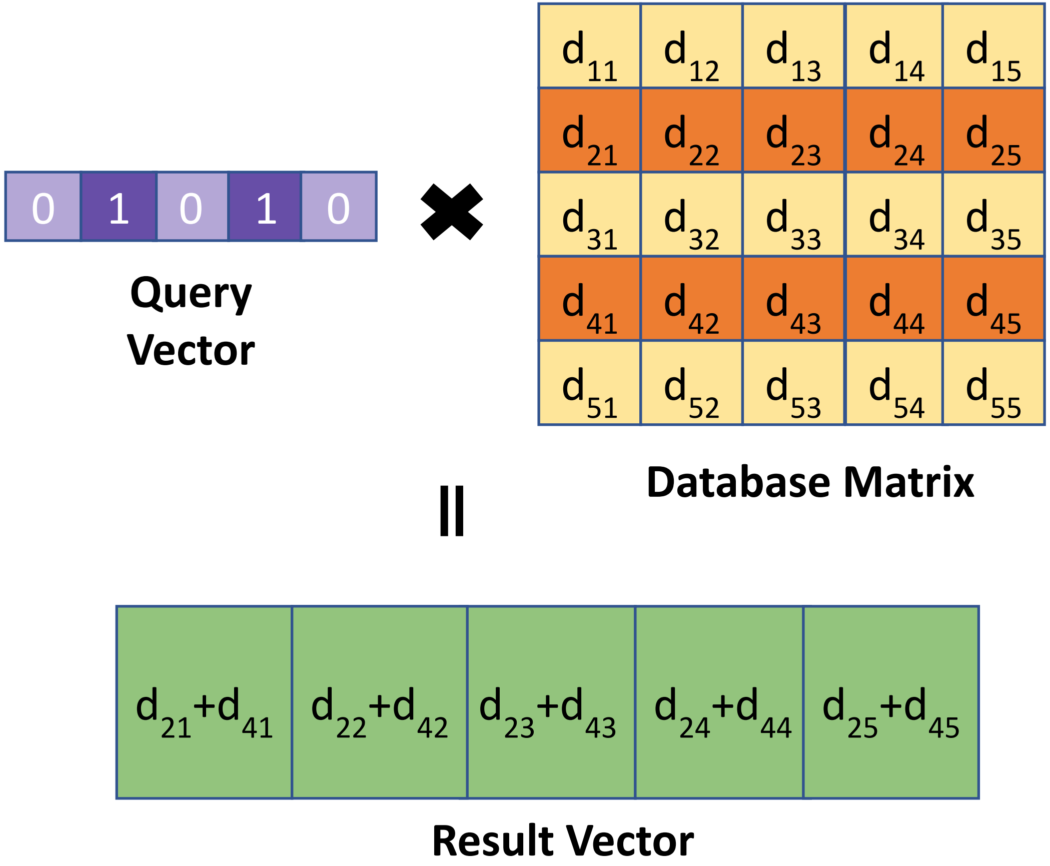
# Vector Matrix Model

- Database modeled as an *r x s* matrix where *r* corresponds to the number of data blocks (or rows) [*Goldberg, 2007*]

- To fetch the block of data, r-dimensional query vector encoded with a 1 in the i-th position and 0s at every other index

- Product of this query vector with the database matrix produces the desired block of data

- However, this procedure is not private and so we use linear secret sharing



$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} \\ d_{41} & d_{42} & d_{43} & d_{44} & d_{45} \\ d_{51} & d_{52} & d_{53} & d_{54} & d_{55} \end{bmatrix} = \begin{bmatrix} d_{21}+d_{41} & d_{22}+d_{42} & d_{23}+d_{43} & d_{24}+d_{44} & d_{25}+d_{45} \end{bmatrix}$$

Query Vector

Database Matrix

Result Vector

# Making VMM Private for Information Retrieval

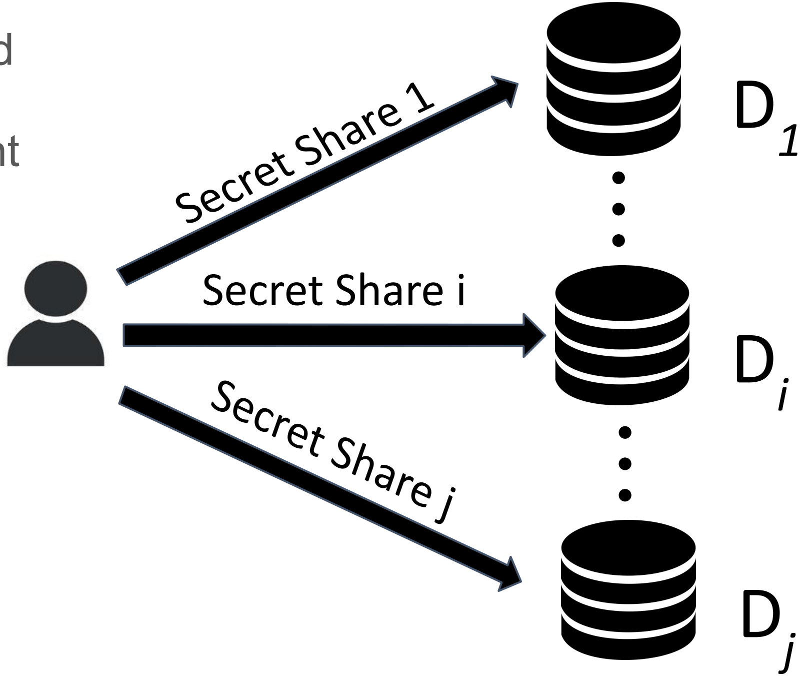
- User shares query vector component-wise across servers, share vectors are multiplied with copies of database matrix hosted in each server, and user receives independent products from each server

- User performs component-wise reconstruction using responses received from the servers to obtain desired block of data

# Making VMM Private for Information Retrieval

- User shares query vector component-wise across servers, share vectors are multiplied with copies of database matrix hosted in each server, and user receives independent products from each server

- User performs component-wise reconstruction using responses received from the servers to obtain desired block of data



# Proposed PIR: Indexes of Aggregate Queries

- Data structure that maps the database into another matrix, designed to serve specific queries

- Each column corresponds to a row in the database, each row corresponds to a unique value of an attribute in the database

- Multiple indexes of queries can be batched together if dimensions same [*Hafiz-Henry, 2017*]

D =

| Hospitalization_ID | Patient_ID | Admit_Date | Gender_ID | Days_Hospitalized | State_ID |
|---|---|---|---|---|---|
| 1 | 1 | 01-02-2022 | 1 (Male) | 10 | 2 (OR) |
| 2 | 2 | 01-04-2022 | 1 (Male) | 2 | 1 (CA) |
| 3 | 3 | 08-06-2022 | 2 (Female) | 14 | 3 (WA) |
| 4 | 1 | 07-23-2022 | 1 (Male) | 2 | 2 (OR) |
| 5 | 3 | 09-01-2022 | 2 (Female) | 7 | 3 (WA) |
| 6 | 4 | 05-14-2022 | 3 (Other) | 2 | 1 (CA) |

$$\prod_{\text{patient}} := \begin{array}{r|cccccc|} \text{patient 1} & 1 & 0 & 0 & 1 & 0 & 0 \\ \text{patient 2} & 0 & 1 & 0 & 0 & 0 & 0 \\ \text{patient 3} & 0 & 0 & 1 & 0 & 1 & 0 \\ \text{patient 4} & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

# Sample Query

D =

| Hospitalization_ID | Patient_ID | Admit_Date | Gender_ID | Days_Hospitalized | State_ID |
|---|---|---|---|---|---|
| 1 | 1 | 01-02-2022 | 1 (Male) | 10 | 2 (OR) |
| 2 | 2 | 01-04-2022 | 1 (Male) | 2 | 1 (CA) |
| 3 | 3 | 08-06-2022 | 2 (Female) | 14 | 3 (WA) |
| 4 | 1 | 07-23-2022 | 1 (Male) | 2 | 2 (OR) |
| 5 | 3 | 09-01-2022 | 2 (Female) | 7 | 3 (WA) |
| 6 | 4 | 05-14-2022 | 3 (Other) | 2 | 1 (CA) |

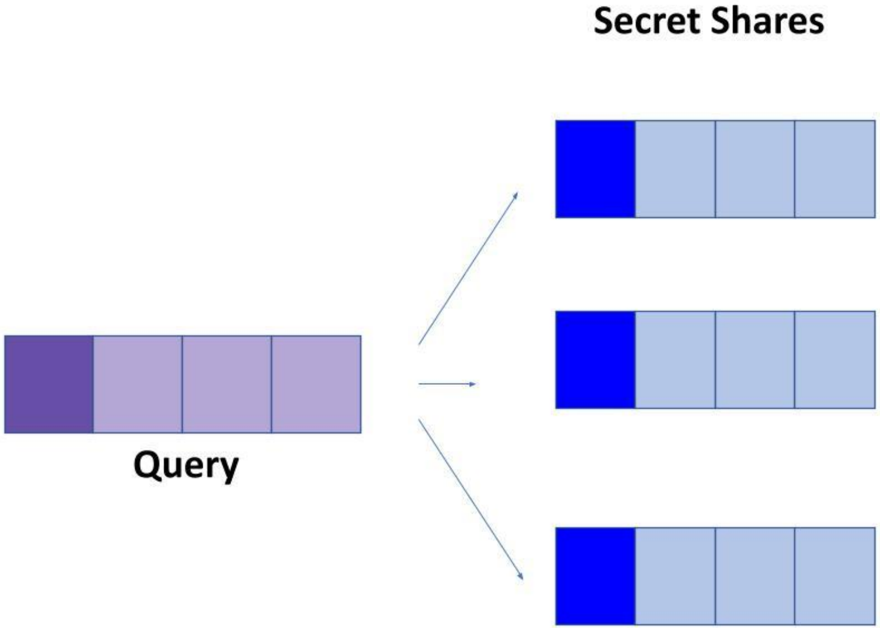
$$\prod_{\text{patient}} := \begin{array}{l} \text{patient 1} \\ \text{patient 2} \\ \text{patient 3} \\ \text{patient 4} \end{array} \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
SELECT SUM(Days_Hospitalized)
From D
WHERE Patient_ID = 1
```

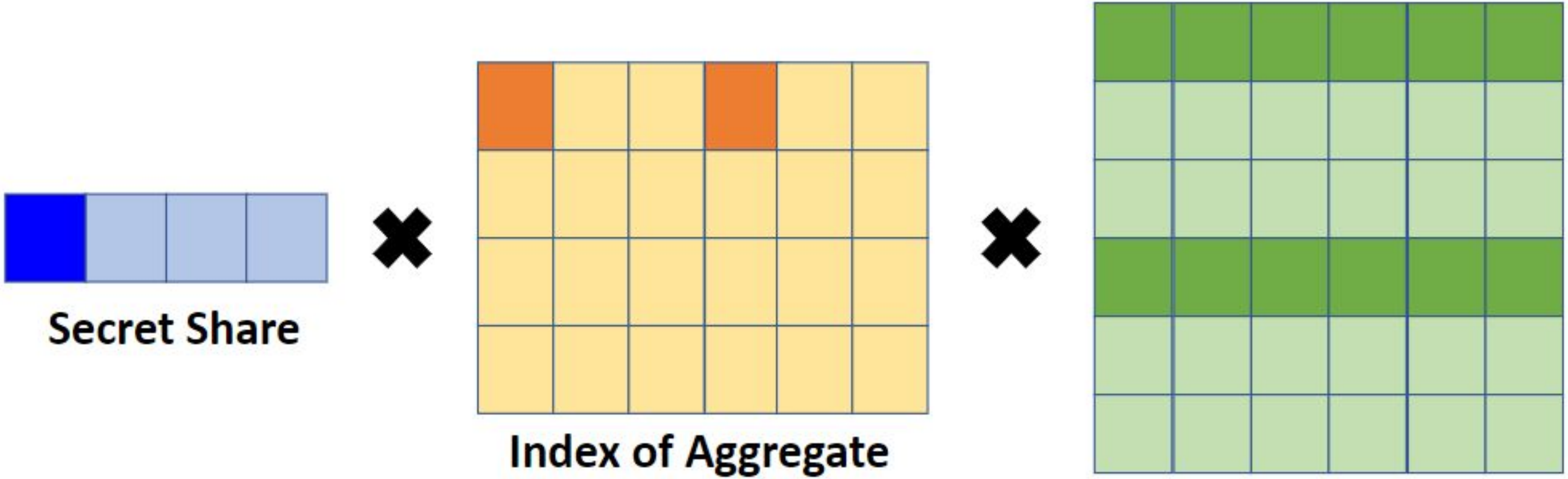
| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |

Query Vector

# Protocol Schematic



# Protocol Schematic

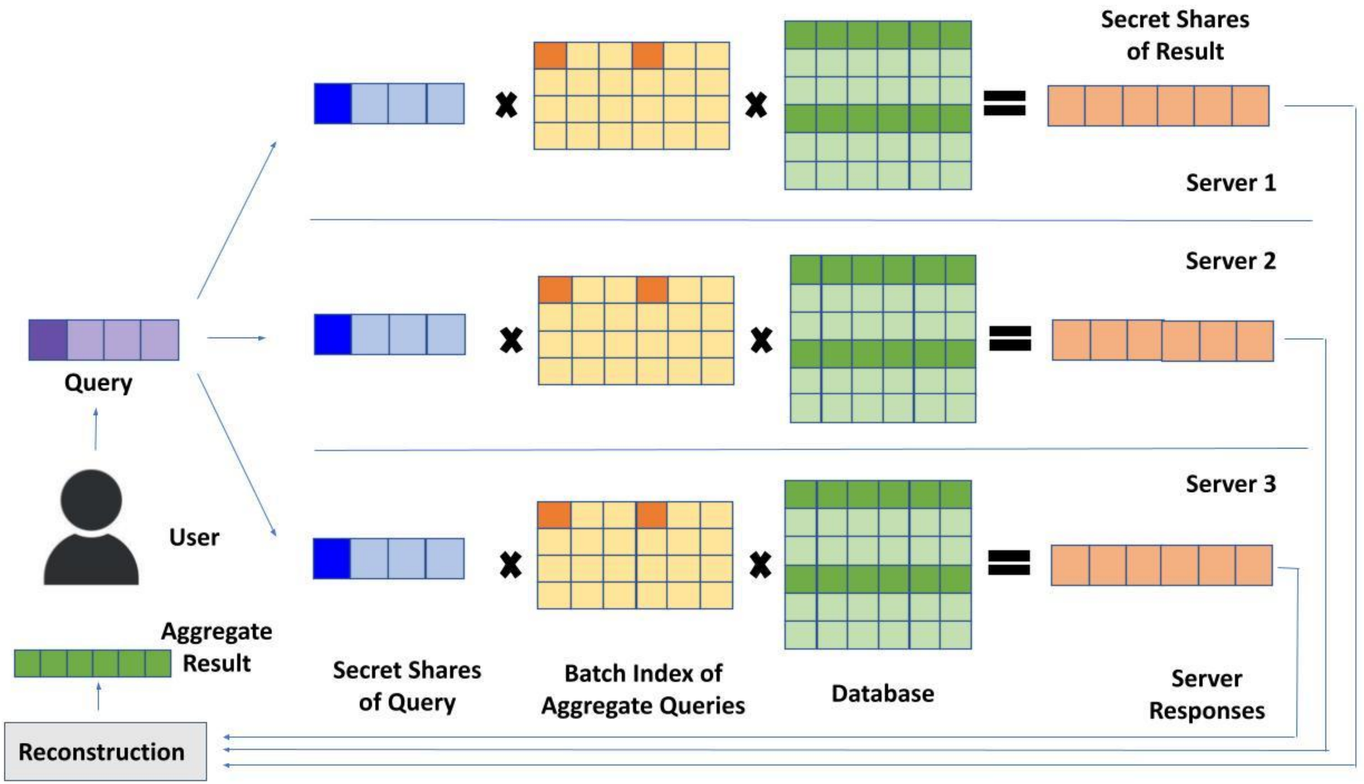


Secret Share

Index of Aggregate Queries

Copy of Database

# Protocol Schematic



# Case Studies

X (Formerly Twitter)

- Scraped 1,004,129 tweets with politically relevant hashtags such as ‘USElections’, ‘Trump’, ‘Biden’
- 2 indexes of queries batched to serve queries about like counts and retweet counts, each index of query of dimension 333,286 × 1,004,129
- Each row in the index of queries corresponds to a unique user in the scraped database

MIMIC 3

- Clinical dataset of hospitalization records
- First set batches 4 indexes of queries to serve 4 different queries, each index of query matrix is of dimension 4 x 58,976, with each row corresponding to a different value of admission type
- Second set batches 2 indexes of queries to serve 2 queries, each index of query is of dimension 1,400 × 4,156,450, with each row corresponding to a different patient

# Case Studies

X (Formerly Twitter)

```
SELECT SUM(like_count) FROM
twitter_data WHERE user_id =
‘100012’
```

```
SELECT COUNT(*) FROM twitter_data
WHERE user_id = ‘100012’ AND
no_retweets = 0
```

MIMIC 3

```
SELECT SUM(hospitalization_duration)
FROM admissions WHERE subject_id =
‘100012’ AND admission_type =
‘EMERGENCY’
```

```
SELECT COUNT(*) FROM admissions
WHERE admission_type = ‘URGENT’
```

# Case Study Results

| Case Study Database | Index of Aggregate Queries for | Index Matrix Dimension | Index Generation Time (secs) | Batching Time for Multiple Indexes (mins) | Additional Data Structure Storage Size (MiB) | VSpM Throughput on GPU (clients/sec) | Server Response Generation Time (secs) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All Attributes | Essential Attributes | Baseline |
| **MIMIC 3** | Admission Type | 4 x 58,976 | 0.06 | 0.002 | 0.76 | 20,534.12 | **0.11** | **0.04** | **0.11** |
| | Ethnicity | | 0.39 | | | | | | |
| | Latest Admission | | 0.98 | | | | | | |
| | Oldest Admission | | 0.95 | | | | | | |
| | Dosage | 1,400 x 4,156,450 | 2.12 | 0.101 | 34.34 | 4,412.80 | **0.26** | **0.10** | **7.37** |
| | Stay Duration | | 2.03 | | | | | | |

# Case Study Results

| Case Study Database | Index of Aggregate Queries for | Index Matrix Dimension | Index Generation Time (secs) | Batching Time for Multiple Indexes (mins) | Additional Data Structure Storage Size (MiB) | VSpM Throughput on GPU (clients/sec) | Server Response Generation Time (secs) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All Attributes | Essential Attributes | Baseline |
| **MIMIC 3** | Admission Type | 4 x 58,976 | 0.06 | 0.002 | 0.76 | 20,534.12 | **0.11** | **0.04** | **0.11** |
| | Ethnicity | | 0.39 | | | | | | |
| | Latest Admission | | 0.98 | | | | | | |
| | Oldest Admission | | 0.95 | | | | | | |
| | Dosage | 1,400 x 4,156,450 | 2.12 | 0.101 | 34.34 | 4,412.80 | **0.26** | **0.10** | **7.37** |
| | Stay Duration | | 2.03 | | | | | | |

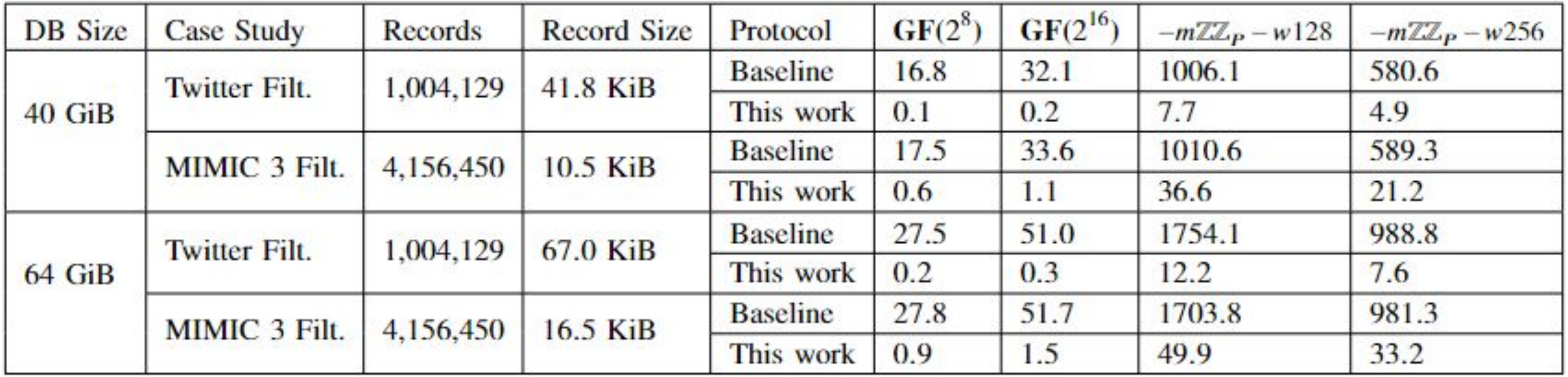
# Case Study Results

| Case Study Database | Index of Aggregate Queries for | Index Matrix Dimension | Index Generation Time (secs) | Batching Time for Multiple Indexes (mins) | Additional Data Structure Storage Size (MiB) | VSpM Throughput on GPU (clients/sec) | Server Response Generation Time (secs) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All Attributes | Essential Attributes | Baseline |
| **MIMIC 3** | Admission Type | 4 x 58,976 | 0.06 | 0.002 | 0.76 | 20,534.12 | **0.11** | **0.04** | **0.11** |
| | Ethnicity | | 0.39 | | | | | | |
| | Latest Admission | | 0.98 | | | | | | |
| | Oldest Admission | | 0.95 | | | | | | |
| | Dosage | 1,400 x 4,156,450 | 2.12 | 0.101 | 34.34 | 4,412.80 | **0.26** | **0.10** | **7.37** |
| | Stay Duration | | 2.03 | | | | | | |

# Case Study Results

| Case Study Database | Index of Aggregate Queries for | Index Matrix Dimension | Index Generation Time (secs) | Batching Time for Multiple Indexes (mins) | Additional Data Structure Storage Size (MiB) | VSpM Throughput on GPU (clients/sec) | Server Response Generation Time (secs) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | **All Attributes** | **Essential Attributes** | **Baseline** |
| **MIMIC 3** | Admission Type | 4 x 58,976 | 0.06 | 0.002 | 0.76 | 20,534.12 | **0.11** | **0.04** | **0.11** |
| | Ethnicity | | 0.39 | | | | | | |
| | Latest Admission | | 0.98 | | | | | | |
| | Oldest Admission | | 0.95 | | | | | | |
| | Dosage | 1,400 x 4,156,450 | 2.12 | 0.101 | 34.34 | 4,412.80 | **0.26** | **0.10** | **7.37** |
| | Stay Duration | | 2.03 | | | | | | |

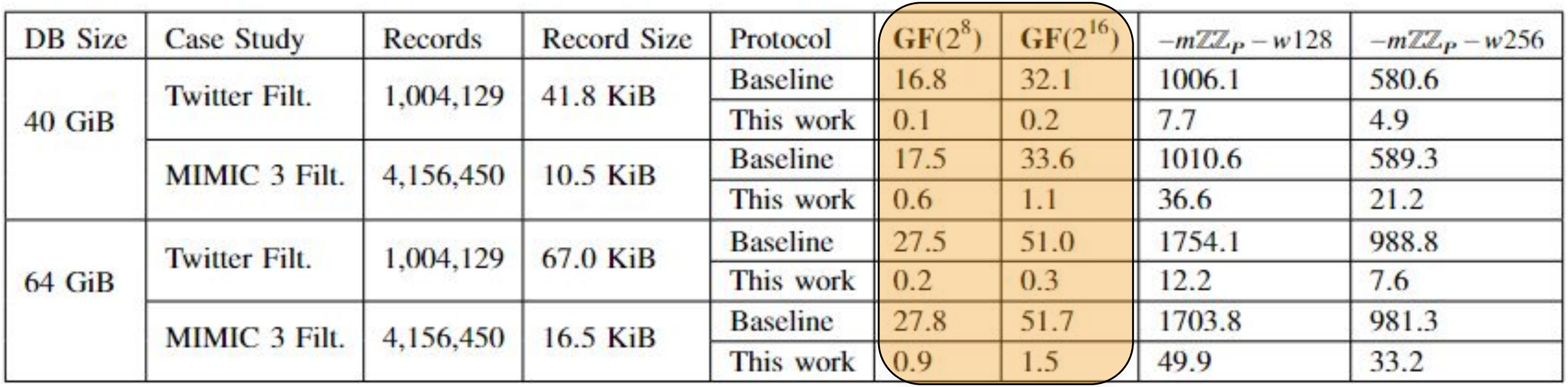
# Server Response Generation on Larger Databases



| DB Size | Case Study | Records | Record Size | Protocol | GF($2^8$) | GF($2^{16}$) | $-m\mathbb{Z}_p - w128$ | $-m\mathbb{Z}_p - w256$ |
|---|---|---|---|---|---|---|---|---|
| 40 GiB | Twitter Filt. | 1,004,129 | 41.8 KiB | Baseline | 16.8 | 32.1 | 1006.1 | 580.6 |
| | | | | This work | 0.1 | 0.2 | 7.7 | 4.9 |
| | MIMIC 3 Filt. | 4,156,450 | 10.5 KiB | Baseline | 17.5 | 33.6 | 1010.6 | 589.3 |
| | | | | This work | 0.6 | 1.1 | 36.6 | 21.2 |
| 64 GiB | Twitter Filt. | 1,004,129 | 67.0 KiB | Baseline | 27.5 | 51.0 | 1754.1 | 988.8 |
| | | | | This work | 0.2 | 0.3 | 12.2 | 7.6 |
| | MIMIC 3 Filt. | 4,156,450 | 16.5 KiB | Baseline | 27.8 | 51.7 | 1703.8 | 981.3 |
| | | | | This work | 0.9 | 1.5 | 49.9 | 33.2 |

Response times for all modulus bit sizes are in seconds

# Server Response Generation on Larger Databases



| DB Size | Case Study | Records | Record Size | Protocol | GF($2^8$) | GF($2^{16}$) | $-m\mathbb{Z}_p - w128$ | $-m\mathbb{Z}_p - w256$ |
|---|---|---|---|---|---|---|---|---|
| 40 GiB | Twitter Filt. | 1,004,129 | 41.8 KiB | Baseline | 16.8 | 32.1 | 1006.1 | 580.6 |
| | | | | This work | 0.1 | 0.2 | 7.7 | 4.9 |
| | MIMIC 3 Filt. | 4,156,450 | 10.5 KiB | Baseline | 17.5 | 33.6 | 1010.6 | 589.3 |
| | | | | This work | 0.6 | 1.1 | 36.6 | 21.2 |
| 64 GiB | Twitter Filt. | 1,004,129 | 67.0 KiB | Baseline | 27.5 | 51.0 | 1754.1 | 988.8 |
| | | | | This work | 0.2 | 0.3 | 12.2 | 7.6 |
| | MIMIC 3 Filt. | 4,156,450 | 16.5 KiB | Baseline | 27.8 | 51.7 | 1703.8 | 981.3 |
| | | | | This work | 0.9 | 1.5 | 49.9 | 33.2 |

Response times for all modulus bit sizes are in seconds

# Takeaways

- Novel framework that augments conventional IT-PIR protocols (e.g., Goldberg’s IT-PIR) with aggregate queries
    - Constructions of effective indexes of aggregate queries comprising new standard aggregate vector

- Simulated real-world applications to benchmark performance and scalability of proposed PIR scheme with aggregate queries

- Efficient implementation of our framework on GPU can achieve fast query response time while assuring the privacy of aggregate queries

# Thank You