TVA: A multi-party computation system for secure and expressive time series analytics

Muhammad Faisal
PhD Student
Boston University
mfaisal@bu.edu

Jerry Zhang
MS Student
UCSD
jerryzhang@ucsd.edu

John Liagouris
Assistant Professor
Boston University
liagos@bu.edu

Vasiliki Kalavri
Assistant Professor
Boston University
vkalavri@bu.edu

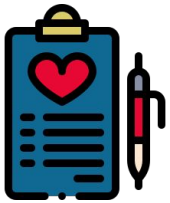
Mayank Varia
Associate Professor
Boston University
varia@bu.edu

BOSTON UNIVERSITY

CASP Systems

https://sites.bu.edu/casp/

Secure Time Series Analytics



Mobile health analytics

Query:
Effect of insulin during *eating periods*.

Protect:
Individuals health records.

Resource Optimization

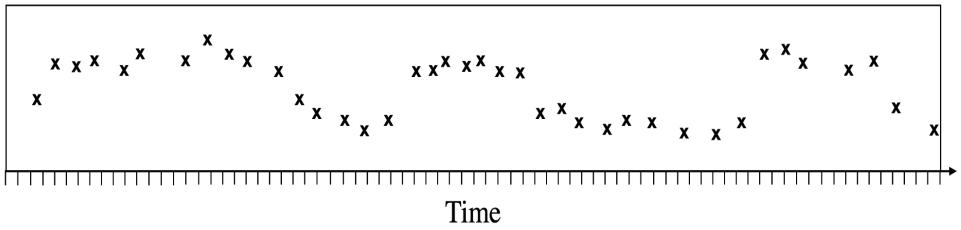
Query:
Resource utilization per *job stages*.

Protect:
Sensitive telemetry data.

Energy Consumption Monitoring

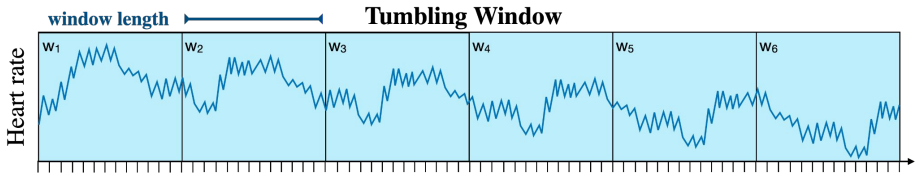
Query:
Energy demand peak and supply peak *per hour*.

Protect:
Smart Grid
Individual's privacy.



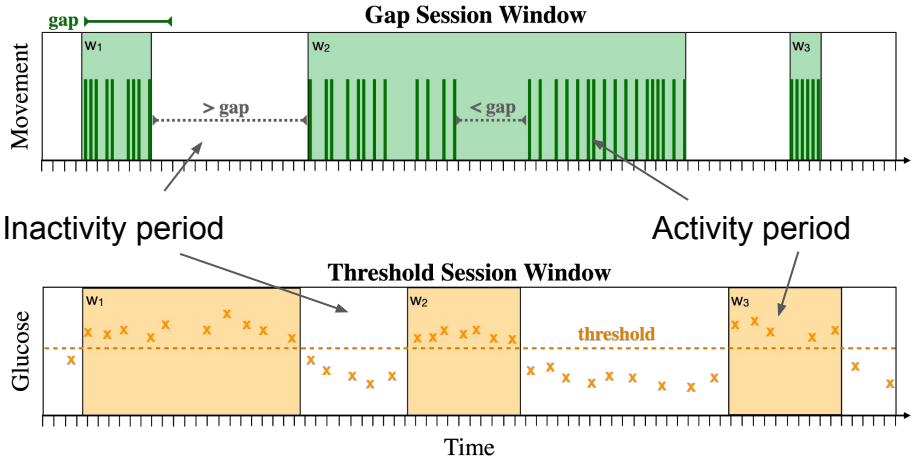
Windows: monitoring the evolution of metrics over time

- **Tumbling Window:**
  groups records into fixed-length time buckets.



Session Windows

- **Gap Session Window:**
  groups records with timestamps difference less than the *'gap'* into the same session.

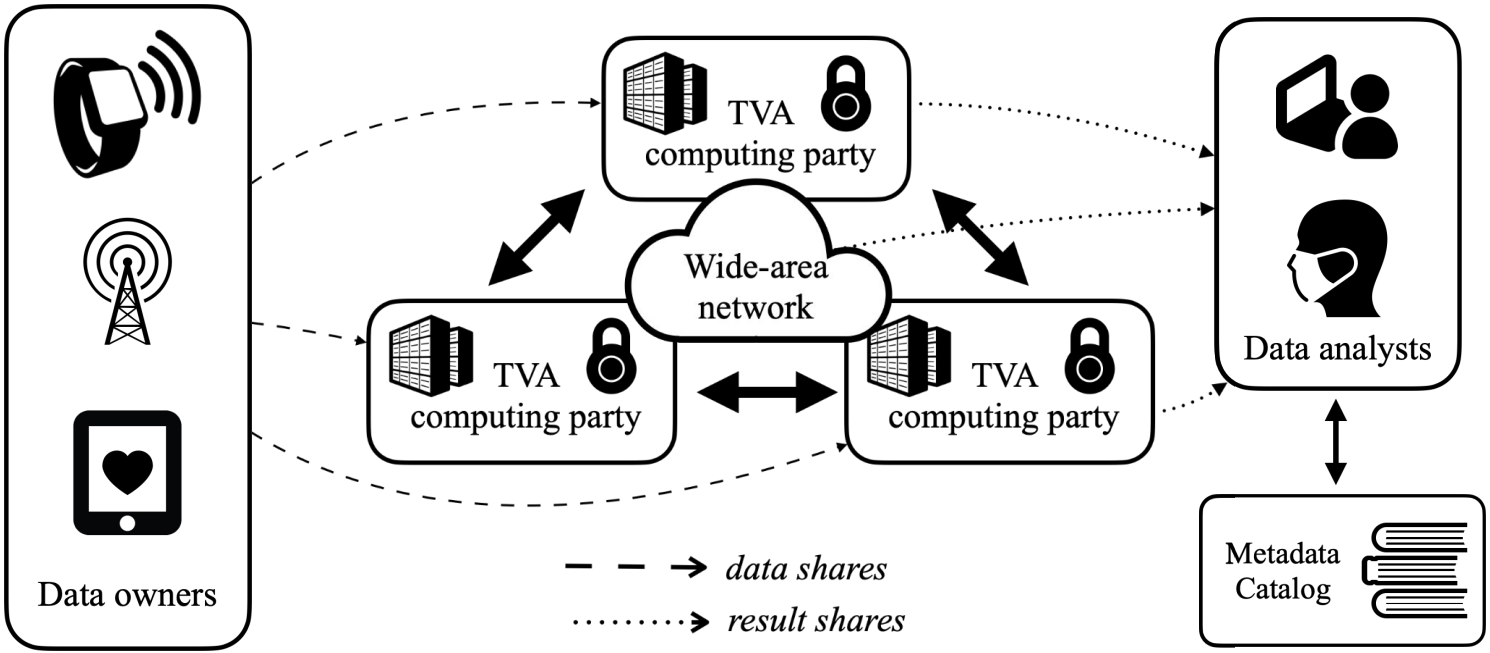


- **Threshold Session Window:**
  groups consecutive records whose attribute values are above the `*threshold*` into the same session.

Requirements for secure and expressive time series analytics

	No Leakage	Time Snapshots	Windows	Query Composition	Unordered and irregular timestamps
Waldo (S&P 22)	✓	✓	✗	(✓)	✗
TimeCrypt (NSDI 20)	(✓)	✓	(✓)	✗	✗
Zeph (OSDI 21)	(✓)	✓	(✓)	✗	✗
TVA	✓	✓	✓	✓	✓

System Overview



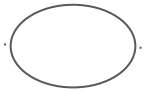
TVA contributions

Secure Window Assignment Protocols

Division for tumbling window assignment - Sessionization

Efficient Operator Composition

Arbitrary complex analytics



Semi-honest and Malicious Security

3PC replicated secret sharing scheme – 4PC Fantastic Four

System Design and Implementation

Efficient multi-threaded runtime, high-level dataflow API and vectorized execution.

Running Example: Mobile Health Analytics

Running Example: Mobile Health Analytics

Query:

“During each patient’s eating period, count the number of insulin doses they have taken.”

Running Example: Mobile Health Analytics

```
// Data schema
TS ts = get_shares({"TIMESTAMP", "PATIENT_ID",
                    "GLUCOSE","INSULIN",
                    "TOTAL_INSULIN_EVENTS"});

// Window aggregation
TS res = ts.keyBy("PATIENT_ID")
           .threshold_window("TIMESTAMP", "GLUCOSE", 120)
           .aggregate("INSULIN", "TOTAL_INSULIN_EVENTS",
                      Agg::COUNT);
```

Query Implemented using TVA API

Running Example: Mobile Health Analytics

```
// Data schema
TS ts = get_shares({"TIMESTAMP", "PATIENT_ID",
                    "GLUCOSE","INSULIN",
                    "TOTAL_INSULIN_EVENTS"});

// Window aggregation
TS res = ts.keyBy("PATIENT_ID")
           .threshold_window("TIMESTAMP", "GLUCOSE", 120)
           .aggregate("INSULIN", "TOTAL_INSULIN_EVENTS",
                      Agg::COUNT);
```

Query Implemented using TVA API

Running Example: Mobile Health Analytics

```
// Data schema
TS ts = get_shares({"TIMESTAMP", "PATIENT_ID",
                    "GLUCOSE","INSULIN",
                    "TOTAL_INSULIN_EVENTS"});

// Window aggregation
TS res = ts.keyBy("PATIENT_ID")
           .threshold_window("TIMESTAMP", "GLUCOSE", 120)
           .aggregate("INSULIN", "TOTAL_INSULIN_EVENTS",
                      Agg::COUNT);
```

Query Implemented using TVA API

Running Example: Mobile Health Analytics

```
// Data schema
TS ts = get_shares({"TIMESTAMP", "PATIENT_ID",
                    "GLUCOSE","INSULIN",
                    "TOTAL_INSULIN_EVENTS"});

// Window aggregation
TS res = ts.keyBy("PATIENT_ID")
           .threshold_window("TIMESTAMP", "GLUCOSE", 120)
           .aggregate("INSULIN", "TOTAL_INSULIN_EVENTS",
                      Agg::COUNT);
```

Query Implemented using TVA API

Running Example: Threshold Session Window

TIMESTAMP	PATIENT_ID	GLUCOSE	INSULIN
1	V3G2Q0	80	0
3	V3G2Q0	118	0
7	V3G2Q0	123	1
8	V3G2Q0	130	0
12	V3G2Q0	112	1
15	V3G2Q0	125	0
16	V3G2Q0	126	0
20	V3G2Q0	90	0

TIMESTAMP

Glucose measurement time

PATIENT_ID

Unique identifier for each patient

GLUCOSE

Measured Glucose value (mg/dL).

INSULIN

Indicates an insulin dose was given

Running Example: Threshold Session Window

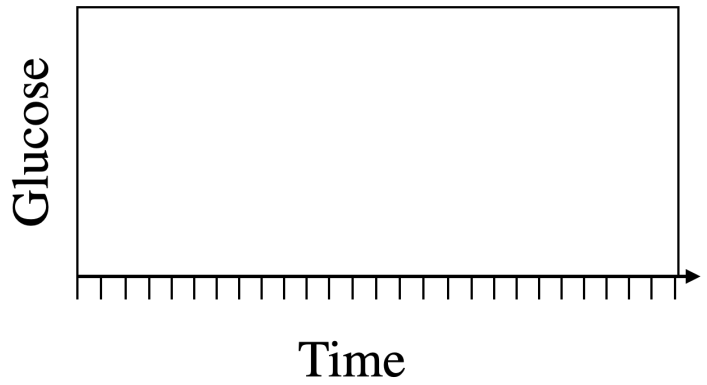
TIMESTAMP	PATIENT_ID	GLUCOSE	INSULIN
1	V3G2Q0	80	0
3	V3G2Q0	118	0
7	V3G2Q0	123	1
8	V3G2Q0	130	0
12	V3G2Q0	112	1
15	V3G2Q0	125	0
16	V3G2Q0	126	0
20	V3G2Q0	90	0

Running Example: Threshold Session Window

TIMESTAMP	GLUCOSE
1	80
3	118
7	123
8	130
12	112
15	125
16	126
20	90

Note

Let’s plot Glucose vs Time for user `V3G2Q0` below

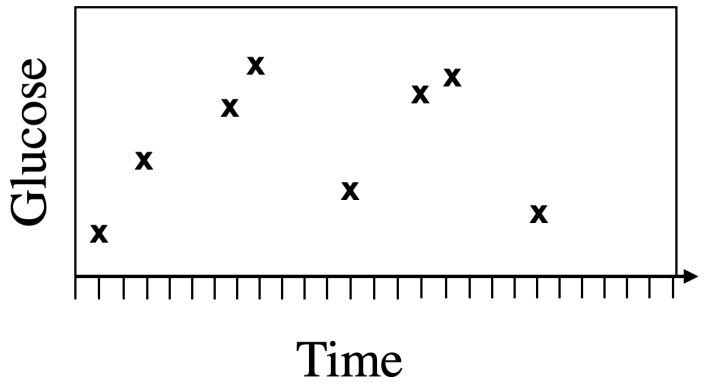


Running Example: Threshold Session Window

TIMESTAMP	GLUCOSE
1	80
3	118
7	123
8	130
12	112
15	125
16	126
20	90

Note

Let’s plot Glucose vs Time for user `V3G2Q0` below



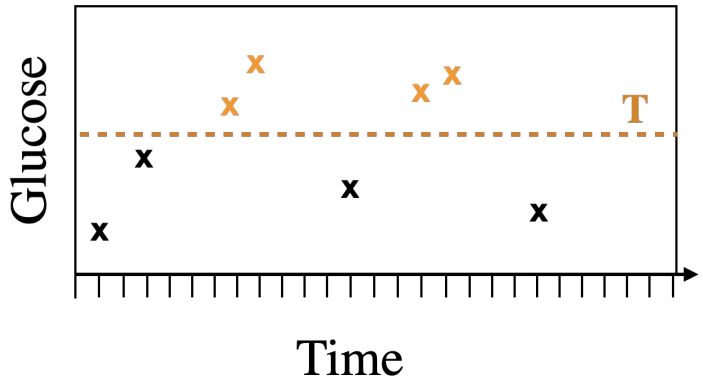
Naive sessionization: **O(n)** rounds

Running Example: Threshold Session Window

TIMESTAMP	GLUCOSE	G>T?
1	80	0
3	118	0
7	123	1
8	130	1
12	112	0
15	125	1
16	126	1
20	90	0

STEP 1A

We evaluate the data points whose values are above the threshold `T`.

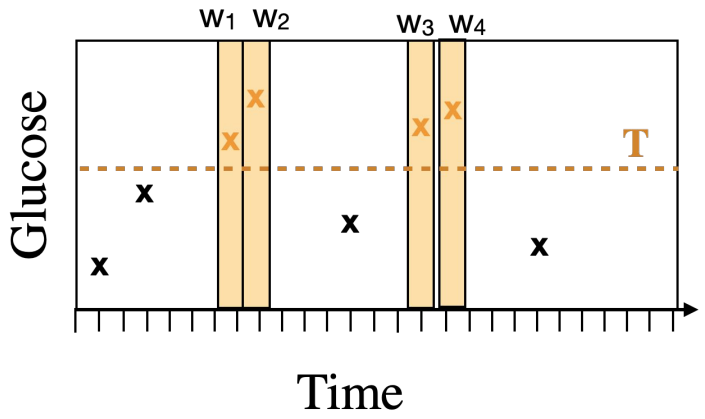


Running Example: Threshold Session Window

TIMESTAMP	GLUCOSE	G>T?	WID
1	80	0	-1
3	118	0	-1
7	123	1	7
8	130	1	8
12	112	0	-1
15	125	1	15
16	126	1	16
20	90	0	-1

STEP 1B `SESSIONSTART`

We assign initial window ID equal to the timestamp according to the threshold condition.



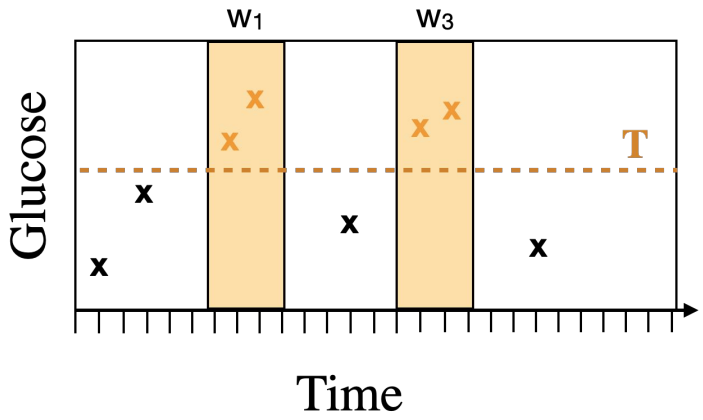
Running Example: Threshold Session Window

TIMESTAMP	GLUCOSE	G>T?	WID
1	80	0	-1
3	118	0	-1
7	123	1	7
8	130	1	7
12	112	0	-1
15	125	1	15
16	126	1	15
20	90	0	-1

O(log n) rounds

STEP 2 `SESSIONIZATION`

We merge consecutive windows giving them WID equal to the smallest WID among them.

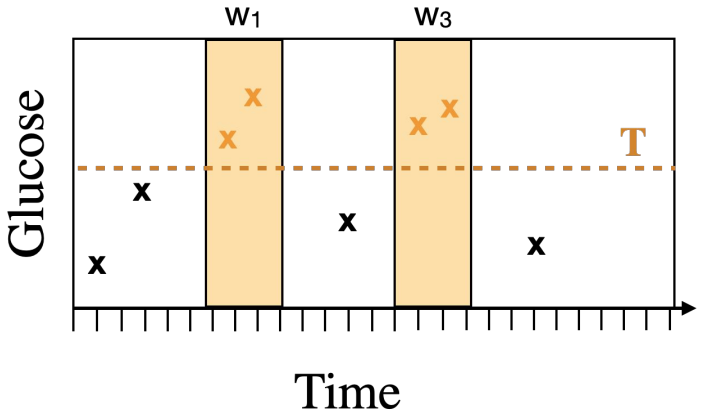


Running Example: Mobile Health Analytics

TIMESTAMP	GLUCOSE	G>T?	WID	INSULIN
1	80	0	-1	0
3	118	0	-1	0
7	123	1	7	1
8	130	1	7	0
12	112	0	-1	1
15	125	1	15	0
16	126	1	15	0
20	90	0	-1	0

Count Aggregation

Use the PATIENT_ID and WID to count the insulin events using Odd-Even Aggregation

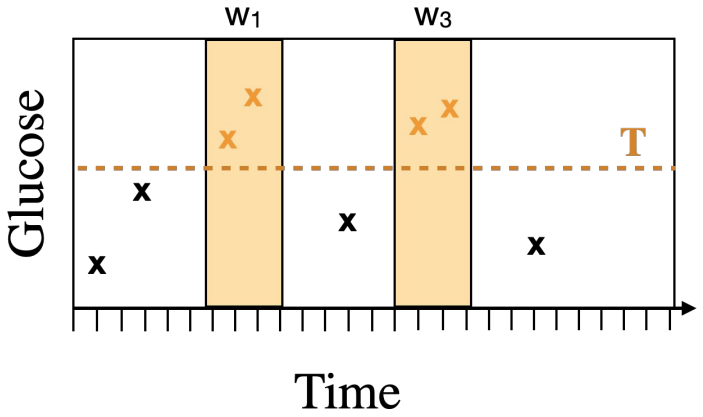


Running Example: Mobile Health Analytics

TIMESTAMP	GLUCOSE	G>T?	WID	INSULIN	#
1	80	0	-1	0	-1
3	118	0	-1	0	-1
7	123	1	7	1	1
8	130	1	7	0	-1
12	112	0	-1	1	-1
15	125	1	15	0	0
16	126	1	15	0	-1
20	90	0	-1	0	-1

Count Aggregation

Use the PATIENT_ID and WID to count the insulin events using Odd-Even Aggregation



Evaluation and Results Highlights

Extensive performance evaluation for TVA

Different metrics

1. **Latency**: End-to-end execution time.
2. **Bandwidth**: Size of data exchanged.
3. **Cost**: Cloud service provider monetary cost.

System Performance

1. Microbenchmarking for primitives.
2. Application queries benchmarking.
3. Multi-threading scalability.
4. LAN vs. WAN.

Comparison with state-of-the-art

TVA vs. Waldo results for computations using the **WaldoTree** and **WaldoTable** data structures.

TVA satisfies time constraints of **online** queries for thousands of data owners

	Energy	mHealth	Scheduling
Input frequency	every 10s	every 5m	10k per minute
Result deadline	every 5m	every 1h	every 10m
Semi-Honest	17400	174700	52400
Malicious	8700	87300	26210

Number of Data sources (WAN)

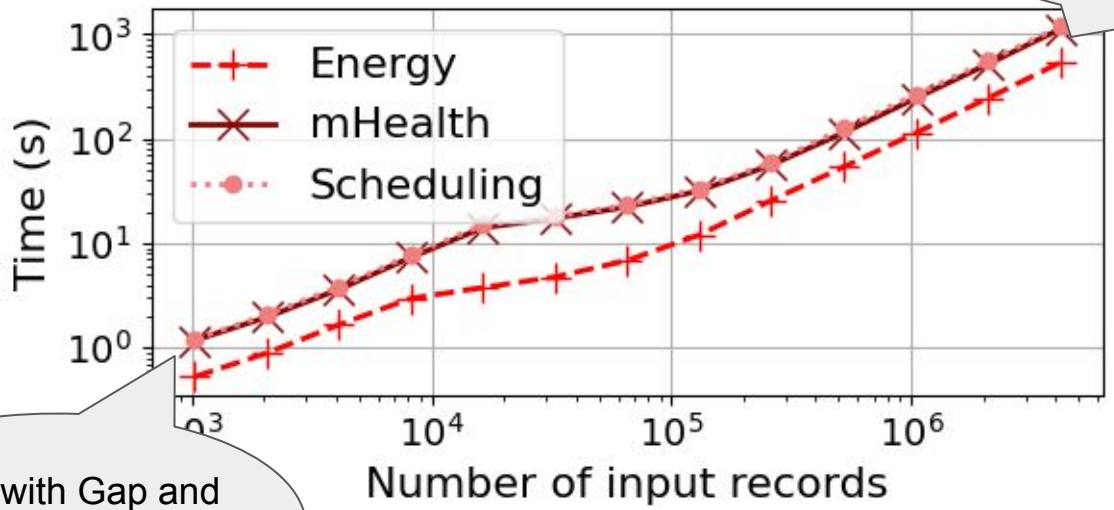
Real Time

We assume practical data input frequency and result deadline.

Practical

TVA can support thousands of data sources simultaneously.

TVA has practical performance for **historical** analysis



4 million input rows within 20 minutes.

Queries with Gap and Threshold session windows have similar latency.

AWS, r5.8xlarge

LAN

Malicious

TVA Summary

1. Easy-to-use, secure high-level time series analytics API.
2. Semi-honest and malicious security without information leakage.
3. Excellent performance for both online and historical analysis.
4. Code available at https://github.com/CASP-Systems-BU/tva

BOSTON UNIVERSITY

CASP Systems

Muhammad Faisal
PhD Student
Boston University
mfaisal@bu.edu