PrivApprox
Privacy-Preserving Stream Analytics
https://privapprox.github.io

Do Le Quoc, Martin Beck,
Pramod Bhatotia, Ruichuan Chen, Christof Fetzer, Thorsten Strufe

July 2017
Motivation

Clients → Analysts

Private data

Recommendation, Ads
Motivation

Clients

Private data

Recommendation, Ads

Analysts

Strong privacy guarantee
Motivation

Clients

Private data

Recommendation, Ads

Analysts

Strong privacy guarantee

High utility analytics in real-time
Motivation

How to preserve users’ privacy while supporting high-utility data analytics for low-latency stream processing?
State-of-the-art systems

Clients
State-of-the-art systems

Clients

- Personal data should be stored locally under the clients’ control
State-of-the-art systems

Clients

Personal data should be stored locally under the clients’ control
State-of-the-art systems

Clients
State-of-the-art systems
State-of-the-art systems
State-of-the-art systems
State-of-the-art systems

Clients

Answer query

Aggregator

Analyst
State-of-the-art systems

Clients → Answer query → Aggregator → Analyst

Add noise
State-of-the-art systems

Clients → Aggregator

Answer query → Add noise

Privacy-preserving output → Analyst
State-of-the-art systems

Clients → Aggregator

Answer query → Add noise → Differential Privacy → Privacy-preserving output → Analyst
State-of-the-art systems

Clients → Answer query

Aggregator

Add noise

Privacy-preserving output

Differential Privacy

Analyst

Limitations:

- Privacy-preserving output
- Add noise
- Differential Privacy
State-of-the-art systems

Limitations:
- Deal with only “single-shot” batch queries 😞
State-of-the-art systems

Limitations:
- Deal with only “single-shot” batch queries 😞
- Require synchronization between system components 😞
State-of-the-art systems

Limitations:
- Deal with only “single-shot” batch queries 😞
- Require synchronization between system components 😞
- Require a trusted aggregator 😞
PrivApprox

Clients

PrivApprox

Analyst
PrivApprox

Clients

PrivApprox:

Analyst
PrivApprox

Clients

PrivApprox

Analyst

PrivApprox:
- Supports stream processing with low latency 😊
PrivApprox

Clients

PrivApprox:

• Supports **stream processing** with **low latency** 😊
• Enables a truly **synchronization-free** distributed architecture 😊
PrivApprox

PrivApprox:
- Supports **stream processing** with **low latency** 😊
- Enables a truly **synchronization-free** distributed architecture 😊
- Requires lower trust in aggregator 😊
Outline

- Motivation
- Overview
- Design
- Evaluation
System overview

Clients

PrivApprox

Analyst
System overview

Clients

PrivApprox

( Query, budget )

Analyst

5
System overview

Execution budget:
- **Latency/throughput** guarantees
- Desired **computing resources** for query processing
- Desired accuracy
System overview

Execution budget:

- **Latency/throughput** guarantees
- Desired **computing resources** for query processing
- Desired accuracy
System overview

Clients

PrivApprox

Analyst
System overview

Clients → PrivApprox → Result → Analyst
System overview

Clients

PrivApprox

Approximate computing

Result

Analyst
System overview

PrivApprox

Approximate computing

Low latency

Clients

Result

Analyst
System overview

PrivApprox

Clients

Approximate computing + Randomized response

Low latency

Result

Analyst
System overview

PrivApprox

Approximate computing + Randomized response

Low latency + Privacy

Clients → PrivApprox → Result → Analyst
System overview

PrivApprox

Approximate computing + Randomized response

Clients

Result

Analyst
System overview

Clients → PrivApprox

- Approximate computing
- Randomized response

Result → Analyst

Zero-knowledge Privacy
System overview

PrivApprox

Approximate computing + Randomized response

Zero-knowledge Privacy

Zero-knowledge Privacy ≥ Differential Privacy
#1: Approximate computing
#1: Approximate computing

State-of-the-art-systems

Compute → Add noise → (Privacy-preserving) approximate output
#1: Approximate computing

State-of-the-art-systems

Idea: To achieve low latency, compute over a sub-set of data items instead of the entire data-set

Compute → Add noise → (Privacy-preserving) approximate output
#1: Approximate computing

**State-of-the-art-systems**

Compute → Add noise → (Privacy-preserving) approximate output

**Idea:** To achieve low latency, compute over a sub-set of data items instead of the entire data-set

**Approximate computing**

Take a sample → Compute → Approximate output ± Error bound
#2: Randomized response
#2: Randomized response

Idea: To preserve privacy, clients may not need to provide truthful answers every time
#2: Randomized response

Idea: To preserve privacy, clients may not need to provide truthful answers every time.
#2: Randomized response

Idea: To preserve privacy, clients may not need to provide truthful answers every time

Client
#2: Randomized response

**Idea:** To preserve privacy, clients may not need to provide truthful answers every time.
#2: Randomized response

**Idea:** To preserve privacy, clients may not need to provide truthful answers every time.
#2: Randomized response

**Idea:** To preserve privacy, clients may not need to provide truthful answers every time
#2: Randomized response

**Idea:** To preserve privacy, clients may not need to provide truthful answers every time

Provides **plausible deniability** for clients responding to sensitive queries; achieves **differential privacy** (RAPPOR [CCS’14])
Outline

• Motivation
• Overview
• Design
• Evaluation
Query model
Query model

Divide answer’s value range into **buckets**, enforce a **binary answer** in each bucket
Query model

Divide answer’s value range into buckets, enforce a **binary answer** in each bucket

**Query:** SELECT age FROM clients WHERE city = ‘Santa Clara’
Query model

Divide answer’s value range into **buckets**, enforce a **binary answer** in each bucket

**Query:** SELECT age FROM clients WHERE city = ‘Santa Clara’

1-20 21-30 31-40 41-50 51-60 >60
Query model

Divide answer’s value range into **buckets**, enforce a **binary answer** in each bucket

Query: SELECT age FROM clients WHERE city = ‘Santa Clara’

<table>
<thead>
<tr>
<th>Age</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21-30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31-40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41-50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>51-60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Age: 31
Query model

Divide answer’s value range into **buckets**, enforce a **binary answer** in each bucket

Query: SELECT age FROM clients WHERE city = ‘Santa Clara’

![Age: 31](image)

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-20</td>
<td>0</td>
</tr>
<tr>
<td>21-30</td>
<td>0</td>
</tr>
<tr>
<td>31-40</td>
<td>1</td>
</tr>
<tr>
<td>41-50</td>
<td>0</td>
</tr>
<tr>
<td>51-60</td>
<td>0</td>
</tr>
<tr>
<td>&gt;60</td>
<td>0</td>
</tr>
</tbody>
</table>

Client cannot arbitrarily manipulate answers
Workflow: Submit query

Aggregator -> Analyst

(Query, budget)
Workflow: Submit query

Clients

Aggregator

Analyst

Cost-Function(budget)

System parameters:
- Sampling parameter
- Randomized response parameters

(Query, budget)
Workflow: Submit query

Clients

(Query, parameters) → Aggregator

Cost-Function (budget) → Analyst

System parameters:
• Sampling parameter
• Randomized response parameters
Workflow: Answer query
Workflow: Answer query

Client
Workflow: Answer query

Client

Step #1

Sampling
(Flip a coin to decide to answer query or not)
Workflow: Answer query

Step #1: Sampling
(Flip a coin to decide to answer query or not)

Step #2: Randomized Response
Workflow: Answer query

Step #1: Sampling (Flip a coin to decide to answer query or not)
Step #2: Randomized Response
Step #3: Send randomized answer
Workflow: Answer query

**Step #1**
- **Sampling**
  - (Flip a coin to decide to answer query or not)

**Step #2**
- **Randomized Response**

**Step #3**
- **Send randomized answer**

**Zero-knowledge privacy**
Workflow: Answer query

Step #1: Sampling (Flip a coin to decide to answer query or not)

Step #2: Randomized Response

Step #3: Send randomized answer

Zero-knowledge privacy

See the paper for details!
Workflow: Answer query

Clients ➔ Randomized answers ➔ Aggregator
Workflow: Answer query

Clients → Randomized answers → Aggregator → Approximate result ± Error bound → Analyst
Workflow: Answer query

Clients → Randomized answers → Aggregator → Approximate result ± Error bound → Analyst

Lack of anonymity and unlinkability?
#3: Anonymity and unlinkability
#3: Anonymity and unlinkability

Idea: XOR-based Encryption
#3: Anonymity and unlinkability

Idea: XOR-based Encryption
#3: Anonymity and unlinkability

**Idea:** XOR-based Encryption

Client

Encrypt answer $M$:
GenerateKey $\rightarrow M_k$
$M \ XOR \ M_k \rightarrow M_E$
#3: Anonymity and unlinkability

**Idea:** XOR-based Encryption

Encrypt answer $M$:
- GenerateKey $\rightarrow M_k$
- $M \text{ XOR } M_k \rightarrow M_E$
#3: Anonymity and unlinkability

**Idea:** XOR-based Encryption

Encrypt answer $M$:
- GenerateKey $\rightarrow M_k$
- $M \text{ XOR } M_k \rightarrow M_E$

Decrypt answer $M_E$:
- $M_E \text{ XOR } M_k \rightarrow M$
Implementation
Implementation

Clients

Proxy

Proxy

Aggregator

Analyst
Implementation
Implementation
Outline

• Motivation
• Overview
• Design
• Evaluation
Experimental setup

• Evaluation questions
  • Utility vs privacy
  • Throughput & latency
  • Network overhead
Experimental setup

• Evaluation questions
  • Utility vs privacy
  • Throughput & latency
  • Network overhead

See the paper for more results!
Experimental setup

• Evaluation questions
  • Utility vs privacy
  • Throughput & latency
  • Network overhead

• Testbed
  • Cluster: 44 nodes
  • Dataset: NYC Taxi ride records, household electricity usage

See the paper for more results!
Accuracy vs privacy
Accuracy vs privacy

- Randomization parameters #1 (p = 0.6, q = 0.6)
- Randomization parameters #2 (p = 0.9, q = 0.6)
The lower the better

Accuracy loss  vs  Privacy level

- Accuracy loss (%)
- Privacy level ($\varepsilon_{zk}$)

Trade-off between utility and privacy

Randomization parameters #1 ($p = 0.6$, $q = 0.6$)
Randomization parameters #2 ($p = 0.9$, $q = 0.6$)
Accuracy vs privacy

The lower the better

Trade-off between utility and privacy

Accuracy loss (%)

Privacy (ε zk)

Sampling Fraction (%)

Randomization parameters #1 (p = 0.6, q = 0.6)

Randomization parameters #2 (p = 0.9, q = 0.6)

Accuracy loss vs Privacy level

Randomization parameters #1 (p = 0.6, q = 0.6)

Randomization parameters #2 (p = 0.9, q = 0.6)

Accuracy loss vs Privacy level

Trade-off between utility and privacy
Accuracy vs privacy

The lower the better

Trade-off between utility and privacy
Accuracy vs privacy

The lower the better

Trade-off between utility and privacy
Accuracy vs privacy

Trade-off between utility and privacy

The lower the better

Accuracy loss: Red line for Randomization parameters #1 (p = 0.6, q = 0.6) and Blue line for Randomization parameters #2 (p = 0.9, q = 0.6)

Sampling Fraction (%)

Privacy level: Squares for Privacy level

Privacy (\(\varepsilon_{zk}\))
Throughput
Throughput

- NYC Taxi Ride
- Household Electricity

Throughput (K)

#nodes
Throughput

Throughput (K)

#nodes

NYC Taxi Ride
Household Electricity

The higher the better
Throughput

The higher the better

NYC Taxi Ride vs Household Electricity

Throughput (K)

#nodes
Throughput

Throughput (K)

#nodes

NYC Taxi Ride

Household Electricity

The higher the better

~8X speedup when going from one node to 20 nodes
Latency
Latency

NYC Taxi Ride  Household Electricity

Total processing time (seconds)

Sampling fraction (%)
Latency

The lower the better

Total processing time (seconds)

Sampling fraction (%)
Latency

The lower the better

NYC Taxi Ride    Household Electricity

Total processing time (seconds)

Sampling fraction (%)
Latency

The lower the better

~1.66X lower than the native execution with sampling fraction of 60%

NYC Taxi Ride

Household Electricity

Native
Network overhead
Network overhead

NYC Taxi Ride  Household Electricity

Network traffic (GB)
0  100  200  300  400  500  600

Sampling fraction (%)  Native
Network overhead

![Graph showing network traffic (GB) vs. sampling fraction (%). The lower the better.](Image)

- NYC Taxi Ride
- Household Electricity
Network overhead

The lower the better
Network overhead

The lower the better

~1.6X lower than the native execution with sampling fraction of 60%
Conclusion

PrivApprox: a privacy-preserving stream analytics system over distributed datasets
Conclusion

**PrivApprox:** a privacy-preserving stream analytics system over distributed datasets

Privacy
Zero-knowledge privacy
## Conclusion

**PrivApprox**: a privacy-preserving stream analytics system over distributed datasets

<table>
<thead>
<tr>
<th>Privacy</th>
<th>Zero-knowledge privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practical</td>
<td>Adaptive execution based on query budget</td>
</tr>
</tbody>
</table>
**Conclusion**

**PrivApprox**: a privacy-preserving stream analytics system over distributed datasets

<table>
<thead>
<tr>
<th>Privacy</th>
<th>Zero-knowledge privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practical</td>
<td>Adaptive execution based on query budget</td>
</tr>
<tr>
<td>Efficient</td>
<td>Randomized response &amp; sampling techniques</td>
</tr>
</tbody>
</table>
## Conclusion

**PrivApprox**: a privacy-preserving stream analytics system over distributed datasets

<table>
<thead>
<tr>
<th>Privacy</th>
<th>Zero-knowledge privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practical</td>
<td>Adaptive execution based on query budget</td>
</tr>
<tr>
<td>Efficient</td>
<td>Randomized response &amp; sampling techniques</td>
</tr>
</tbody>
</table>

Thank you!

[https://privapprox.github.io](https://privapprox.github.io)