Opaque: An Oblivious and Encrypted Distributed Analytics Platform

[NSDI’17]

Joint work with: Wenting Zheng, Ankur Dave, Jethro Beekman, Raluca Ada Popa, Joseph Gonzalez, and Ion Stoica

UC Berkeley
Complex analytics run on sensitive data

client

sensitive data

cloud provider

Spark SQL
MLLib
GraphX
Spark Streaming
NEWS ANALYSIS

No, your data isn't secure in the cloud

In 2012, Google alone received 21,389 government requests for information affecting 33,634 user accounts

By Lucas Mearian
Senior Reporter, Computerworld | AUG 13, 2013 7:00 AM PT

How to Tell If Your Cloud Provider Can Read Your Data

by Rich Mogull

With the tremendous popularity of services like Dropbox and iCloud there is, rightfully, an incredible amount of interest in cloud data security. Once we start hosting our most sensitive data with cloud services (or any third-party provider) it’s only natural to wonder how secure our data is when it’s in the hands of others. But sometimes it’s hard to figure out exactly who can look at our information, especially since buzzwords like “secure” and “encrypted” don’t necessarily mean you...
Cloud attackers

client

cloud provider

sensitive data
Threat model

Attacker has full access to all cloud software
How to protect data and computation while preserving functionality?

relational algebra
Hardware enclaves 101
Hardware enclaves (Intel SGX)

- Hardware-enforced isolated execution environment
- Data decrypted only on the processor
- Protect against an attacker who has root access or compromised OS
Remote attestation

Enables verifying which code runs in the enclave and performing key exchange.
Attacker is restricted to software attacks only, and does not exploit timing. **Attacker controls the software stack.**
Access patterns leakage
Access patterns

machine 0

processor

network messages

machine 1

memory

addresses
## Example: network access pattern leakage

```sql
SELECT count(*) FROM medical
GROUP BY disease
```
Example: network access pattern leakage

Public information:
Diabetes twice as common as cancer
Example: network access pattern leakage

Learns that Alice has cancer.
Leakage from prior work

- Memory access patterns attacks [XCP15] extracted complete text documents and photo outlines

- Network access patterns [OCF+15] extracted age, gender, address of individuals
Goal: oblivious distributed analytics

access patterns are independent of data content
Opaque*: oblivious and encrypted distributed analytics platform

* Oblivious Platform for Analytic QUERies
Security guarantees (informal)

- **Data encryption and authentication**

- **Computation integrity:** the client can check that the computation result was not affected by an attacker

- **Obliviousness:** The memory and network accesses of a query is the same for *any* two inputs with the same size characteristics (input/outputs)
  - When enabling padding, Opaque hides output sizes as well
Achieving practical obliviousness is not easy

Obliviousness typically comes with high overheads

- For example, the state-of-the-art system, ObliVM, is six orders of magnitude slower than regular computation
Opaque components

**Oblivious query planning**
- Cost model
- Rule-based opt.
- Cost-based opt.

**Distributed oblivious operators**
- Oblivious Filter
- Oblivious Aggregation
- Oblivious Join

**Computation verification**

**Data encryption and authentication**
query = SELECT sum(*)
FROM table
Problem: cloud can alter distributed computation

- Drop data
- Modify data
- Skip task
- Replay old state
Example: drop data

query = SELECT sum(*) FROM table
Self-verifying computation

Invariant: if computation does not abort, the execution completed so far is correct

If the computation is complete, then the entire query was executed correctly
Self-verifying computation

Task 13

Task 14

Task 15

query = SELECT sum(*)
FROM table
Opaque components

Oblivious query planning
- Cost model
- Rule-based opt.
- Cost-based opt.

Distributed oblivious operators
- Oblivious Filter
- Oblivious Aggregation
- Oblivious Join

Computation verification

Data encryption and authentication
Oblivious aggregation

SELECT count(*) FROM medical GROUP BY disease

There can be many partitions
Oblivious aggregation

SELECT count(*) FROM medical GROUP BY disease

Map

Sort

Oblivious sort
[CLRS, Leighton ‘85]
Oblivious aggregation

SELECT count(*) FROM medical GROUP BY disease

[CLRS, Leighton '85]

Oblivious sort

Map

Sort

Cancer

Diabetes

13744

98329

12809

29489

18740

32591

...
Oblivious aggregation

SELECT count(*) FROM medical GROUP BY disease

The "Diabetes" group is split!

How to aggregate obliviously and in parallel?
It can span over many partitions
Oblivious aggregation

SELECT count(*) FROM medical GROUP BY disease

Scan Boundary processing Scan
Oblivious aggregation

```
SELECT count(*) FROM medical GROUP BY disease
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>13744</td>
<td></td>
<td>Cancer</td>
</tr>
<tr>
<td>98329</td>
<td></td>
<td>Cancer</td>
</tr>
<tr>
<td>12809</td>
<td></td>
<td>Diabetes</td>
</tr>
<tr>
<td>129489</td>
<td></td>
<td>Diabetes</td>
</tr>
<tr>
<td>18740</td>
<td></td>
<td>Diabetes</td>
</tr>
<tr>
<td>32591</td>
<td></td>
<td>Diabetes</td>
</tr>
</tbody>
</table>

Scan

Boundary processing

Scan
Oblivious aggregation

SELECT count(*) FROM medical GROUP BY disease

Sort

Oblivious sort
(CLRS, Leighton ‘85)

Final result

Aggregation has two sorts…

Can we do better?