BigSecret: A Secure Data Management Framework for Key-Value Stores

Murat Kantarcioglu
Motivation

- Increased network traffic, and number of users in the last decade
  - Simultaneously millions of users want to surf on popular web sites, e.g. Facebook, Twitter, Amazon
- Traditional databases may cause bottlenecks
  - ACID properties may not be needed
  - Some applications may tolerate inconsistencies in the data
  - A user wouldn’t wait too much for a web page to open
- Retrieving data should be fast!
Motivation

- Many companies have started adopting and providing Key-Value store solutions, e.g. Amazon’s SimpleDB, Google’s AppEngine
  - Very fast data retrieval and sending
  - Scalable
  - Allows dynamic data structure
- In a Key-Value store, data consists of a Key and Value pairing
  - Both are uninterpreted array of bytes
  - Stored based on some rules
    - Some may store in sorted order
    - Some may store using hash functions
Motivation

• A Data Owner can outsource data to any number of public cloud providers
  – May also use its own cloud infrastructure, i.e. private cloud
• When public, sensitive information needs to be protected
• Securing data and providing efficient querying mechanisms is though
Aim

- Provide scalable, efficient and secure solutions to outsource Key-Value data
- Utilize any existing cloud infrastructures to
  - Improve performance
  - Reduce the overall monetary cost
  - Reduce the overall sensitive data disclosure
Overview of the Solution: BigSecret

Client - 1

Proxy

BigSecret

Aux. Data

Cloud Provider-1

Cloud Provider-k

Client - n
Outline

• How do we share data and workload on all providers?
  – Monetary metrics
  – Security metrics
  – Performance metrics
  – Heuristic solution

• How do we store data in encrypted form?
  – Transformation of data
  – Transformation of queries

• Experiments
Data and Workload Partitioning Problem

• Given:
  – A dataset of Key-Value pairs, $D$
  – A workload on the dataset, $Q$
  – A set of providers, $P$

• First, partition workload over the providers such that
  – The total execution time of the workload is minimized
  – The total amount of monetary cost from cloud usage is below a limit
  – Expected amount of sensitive data disclosure is below a limit
Data and Workload Partitioning Problem

- Second, partition data based on queries
  - All Key-Value pairs needed to answer a query is given to that provider

```
BigSecret

q_1, q_2
D_{q_1}, D_{q_2}

Cloud Provider-1

D_{q_3}, D_{q_4}

q_3, q_4

Cloud Provider-k
```
Data and Workload Partitioning Problem

- Each provider $P_i$ is given:
  - Partitioned workload $Q_{P_i}$
  - Partitioned dataset $D_{P_i}$

- Total monetary cost of a workload $Q_{P_i}$ on the provider $P_i$ is:
  \[ t(Q_{P_i}, P_i) := s(D_{P_i}, P_i) + \sum_{q \in Q_{P_i}} f(q) \times c(q, P_i) \]

- Total monetary cost of the partitioned workload $Q_{P_1}, \ldots, Q_{P_k}$:
  \[ \sum t(Q_{P_i}, P_i) \leq C_{cost} \]
Data and Workload Partitioning Problem

- Expected number of sensitive Key-Value entries in $\mathcal{D}_{P_i}$ is represented as $sens(\mathcal{D}_{P_i})$
- Total expected disclosure:
  \[ \sum w_{P_i} \times sens(\mathcal{D}_{P_i}) \leq C_{risk} \]
• Expected execution time of a workload $Q_{P_i}$ on $P_i$
  
  - $r(Q_{P_i}, P_i) := \sum_{q \in Q_{P_i}} f(q) \times r(q, P_i)$
  - $r(q, P_i) := \frac{io(q)}{pow(P_i)}$

• Minimize the total execution time:
  
  - $OptRun(Q_{P_1}, \ldots, Q_{P_k}) := \sum_{P_i \in \mathcal{P}} r(Q_{P_i}, P_i)$

• We call this particular partitioning problem over a set of providers as Multi-Cloud Partitioning Problem (MCPP), which is proven to be NP-Hard
Heuristic Solution to MCPP

- We approach the problem using Hill-Climbing technique
- First assign each query to a provider so that the initial constraints are met
- Then, iterate over each query and check
  - Better performance can be achieved
  - Constraints are still met
- Finish when no further improvements can be made
Background - HBase

- Apache’s open source Key-Value store implementation, designed after Google’s BigTable

- Key consists of four parts:
  - row-key (row)
  - family (fam)
  - qualifier (qua)
  - timestamp (ts)

- Provides 4 operations:
  - Put, Get, Delete, and Scan
Data Transformation

- Data is transformed into encrypted form using “Encryption Models”

<table>
<thead>
<tr>
<th></th>
<th>Model-1</th>
<th>Model-2</th>
<th>Model-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>$\text{Map}(row)$</td>
<td>$H(row)$</td>
<td>$H(row)$</td>
</tr>
<tr>
<td>fam</td>
<td>$\text{Map}(fam)$</td>
<td>$H(fam)$</td>
<td>0</td>
</tr>
<tr>
<td>qua</td>
<td>$\text{Map}(qua)|E(\text{KEY})$</td>
<td>$H(qua)|E(\text{KEY})$</td>
<td>$E(\text{KEY})$</td>
</tr>
<tr>
<td>ts</td>
<td>$\text{Map}(ts)$</td>
<td>$H(ts)$</td>
<td>1</td>
</tr>
<tr>
<td>val</td>
<td>$E(val)$</td>
<td>$E(val)$</td>
<td>$E(val)$</td>
</tr>
</tbody>
</table>
Query Translation

• Given a Put query, translation for Model-2:
  – put(“Jake”, “personal”, “height”, “170cm”, 1)
  – put(H(“Jake”), H(“personal”), H(“height”)||E(KEY),
    E(“170cm”), H(1))
  – KEY = “Jakepersonalheight1”

• Given a Get query, translation for Model-2:
  – get(“Jake”, 0, ∞, “personal”)
  – get(H(“Jake”), 0, ∞, H(“personal”))
Experiments

• Performed experiments using Yahoo! Cloud Serving Benchmark
• Created tables consisting of 1, 2, 4, 8, 16, and 32 Millions of rows
  – Each row has 10 Key-Value entries of 100B
• Created 3 different workloads
  – 1K queries for single-cloud experiments
  – 100K queries for multi-cloud experiments

<table>
<thead>
<tr>
<th></th>
<th>Workload-1</th>
<th>Workload-2</th>
<th>Workload-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put (%)</td>
<td>5</td>
<td>95</td>
<td>25</td>
</tr>
<tr>
<td>Get (%)</td>
<td>95</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>Scan (%)</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>
Single-Cloud Experiments

Plaintext

Model-1

Model-2

Model-3

Total Time (sec)

Table Size (M rows)

Workload - 1
Single-Cloud Experiments

![Graph showing total time (sec) vs. table size (M rows) for different models with a workload of 2.](image)

- **Plaintext**
- **Model-1**
- **Model-2**
- **Model-3**

Table Size (M rows):
- 1
- 2
- 4
- 8
- 16
- 32

Total Time (sec):
- 1
- 10
- 100

Workload - 2
Single-Cloud Experiments

Table Size (M rows) vs. Total Time (sec) for Workload - 3.
Multi-Cloud Experiments

- Same cluster is used with two different settings:
  - Provider-1: Plaintext storage, $w_{P_1}$ is 1
  - Provider-2: Uses Model-2, $w_{P_2}$ is 0.7
- Both have the same pricing policies
  - Amazon S3, EC2, and EMR pricing
- Monetary cost constraint varies between $700 and $3700
- All data is assumed to be sensitive
Multi-Cloud Experiments

Workload - 3
Conclusion

- We proposed efficient and secure storage techniques, specially designed for Key-Value stores.
- We formalized how to partition data and workload on a multi-cloud setup with monetary, sensitivity disclosure, and performance constraints.
- We implemented BigSecret on Hbase, and evaluated the performance.
Conclusion

• We plan to add support for other Key-Value stores
• BigSecret will be open-source
Thank You