Secure Grouping and Aggregation with MapReduce

Radu Ciucanu  Matthieu Giraud
Pascal Lafourcade  Lihua Ye

28 July 2018
SECRIPT, Porto
Example of Grouping and Aggregation

<table>
<thead>
<tr>
<th>Name</th>
<th>Department</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Computer Science</td>
<td>1900</td>
</tr>
<tr>
<td>Bob</td>
<td>Mathematics</td>
<td>1750</td>
</tr>
<tr>
<td>Mallory</td>
<td>Computer Science</td>
<td>1800</td>
</tr>
<tr>
<td>Oscar</td>
<td>Physics</td>
<td>2000</td>
</tr>
<tr>
<td>Carol</td>
<td>Mathematics</td>
<td>1600</td>
</tr>
</tbody>
</table>
## Example of Grouping and Aggregation

<table>
<thead>
<tr>
<th>Name</th>
<th>Department</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Computer Science</td>
<td>1900</td>
</tr>
<tr>
<td>Bob</td>
<td>Mathematics</td>
<td>1750</td>
</tr>
<tr>
<td>Mallory</td>
<td>Computer Science</td>
<td>1800</td>
</tr>
<tr>
<td>Oscar</td>
<td>Physics</td>
<td>2000</td>
</tr>
<tr>
<td>Carol</td>
<td>Mathematics</td>
<td>1600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Department</th>
<th>COUNT</th>
<th>SUM</th>
<th>AVG</th>
<th>MAX</th>
<th>MIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>2</td>
<td>3700</td>
<td>1850</td>
<td>1900</td>
<td>1800</td>
</tr>
<tr>
<td>Mathematics</td>
<td>2</td>
<td>3350</td>
<td>1675</td>
<td>1750</td>
<td>1600</td>
</tr>
</tbody>
</table>
MapReduce

- Partitioning input data
- Scheduling program execution on machines
- Performing the shuffle
- Handling machine failures

### Programmer gives:

- Input files
- **Map** and **Reduce**

![MapReduce Diagram]

Input 1 | Input 2 | Input 3
--------|--------|--------
Basket 1 | Basket 2 | Basket 3

Map 1 | Map 2 | Map 3
------|------|------
Basket 1 | Basket 2 | Basket 3

Shuffle

Reduce 1 | Reduce 2
--------|--------
Basket 1 | Basket 2

Output 1 | Output 2
---------|---------
Bottle 1 | Bottle 2
Grouping and Sum with MapReduce

\[ \gamma_{\text{Dept}, \text{SUM}(\text{Salary})}(D) \]

**Map:**
\[ \mathcal{M} \rightarrow \mathcal{R} : \{ (\pi_{\text{Dept}}(t), \pi_{\text{Salary}}(t)) \}_{t \in D} \]

**Reduce:**
Input: (\text{key}, \text{values})
\[ \text{sum} = \sum_{\pi_{\text{Dept}}(t) \in \text{values}} \pi_{\text{Salary}}(t) \]
\[ \mathcal{R} \rightarrow \mathcal{P} : (\pi_{\text{Dept}}(t), \text{sum}). \]
Security Model

Cloud is **honest-but-curious**

Data Owner $\rightarrow f \rightarrow$ Cloud $\rightarrow f(\cdot) \rightarrow$ User

Security properties

- Secrecy of $\square$ and $f(\square)$
- User queries $f(\square)$ but cannot learn $\square$
## Contributions

### Secure MapReduce Algorithms:

- COUNT
- SUM
- AVG
- MAX
- MIN

### Secure Private Approach

- Cloud nodes do not learn
- Cloud nodes do not learn $f()$
- User does not learn
Outline

Cryptography
  Pseudo-Random Permutation
  Partial Homomorphic Encryption
  Order Preserving Encryption

Secure-Private MapReduce for COUNT, SUM and AVG

Secure-Private MapReduce for MIN and MAX

Security and Performances

Conclusion
Idea:

\[ f : \{0, 1\}^n \times \{0, 1\}^{n_0} \rightarrow \{0, 1\}^{n_1} \]

- Deterministic
- Result indistinguishable from a random
- Not invertible

Notation: Data owner picks a key \( k \) and uses \( f_k(m) \)
Fully Homomorphic Encryption (Gentry 2009)

Idea:

Perform ANY computations on encrypted data

\[ \forall f, \forall x_i, f(\mathcal{E}_k(x_1), \ldots, \mathcal{E}_k(x_n)) = \mathcal{E}_k(f(x_1, \ldots, x_n)) \]

- Not yet efficient enough
Partial Homomorphic Encryption

<table>
<thead>
<tr>
<th>Paillier’s Cryptosystem (1999)</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Public Key encryption</td>
</tr>
<tr>
<td>- Probabilistic encryption</td>
</tr>
<tr>
<td>- $E_{pk}(x + y) = E_{pk}(x) \cdot E_{pk}(y)$</td>
</tr>
</tbody>
</table>

$$E_{pk}(x \cdot y) = (E_{pk}(x))^y$$
Oder Preserving Encryption


Let $c_1 = E_k(m_1)$ and $c_2 = E_k(m_2)$

if $m_1 < m_2$ then $c_1 < c_2$

- Symmetric encryption
Outline

Cryptography
  Pseudo-Random Permutation
  Partial Homomorphic Encryption
  Oder Preserving Encryption

Secure-Private MapReduce for COUNT, SUM and AVG

Secure-Private MapReduce for MIN and MAX

Security and Performances

Conclusion
COUNT, SUM and AVG

Preprocessing on data

- All data are encrypted with Paillier with $pk_U$
- All data $d$ have $f_k(d)$

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>CS</td>
<td>1900</td>
</tr>
<tr>
<td>B</td>
<td>Math</td>
<td>1750</td>
</tr>
<tr>
<td>M</td>
<td>CS</td>
<td>1800</td>
</tr>
<tr>
<td>O</td>
<td>Phy</td>
<td>2000</td>
</tr>
<tr>
<td>C</td>
<td>Math</td>
<td>1600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_k(A),\varepsilon_{pk_U}(A)$</td>
<td>$f_k(CS), \varepsilon_{pk_U}(CS)$</td>
<td>$f_k(1900), \varepsilon_{pk_U}(1900)$</td>
</tr>
<tr>
<td>$f_k(B),\varepsilon_{pk_U}(B)$</td>
<td>$f_k(Math), \varepsilon_{pk_U}(Math)$</td>
<td>$f_k(1750), \varepsilon_{pk_U}(1750)$</td>
</tr>
<tr>
<td>$f_k(M),\varepsilon_{pk_U}(M)$</td>
<td>$f_k(CS), \varepsilon_{pk_U}(CS)$</td>
<td>$f_k(1800), \varepsilon_{pk_U}(1800)$</td>
</tr>
<tr>
<td>$f_k(O),\varepsilon_{pk_U}(O)$</td>
<td>$f_k(Phy), \varepsilon_{pk_U}(Phy)$</td>
<td>$f_k(2000), \varepsilon_{pk_U}(2000)$</td>
</tr>
<tr>
<td>$f_k(C),\varepsilon_{pk_U}(C)$</td>
<td>$f_k(Math), \varepsilon_{pk_U}(Math)$</td>
<td>$f_k(1600), \varepsilon_{pk_U}(1600)$</td>
</tr>
</tbody>
</table>
Secure Private COUNT

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_k(A), \varepsilon_{pk_U}(A)$</td>
<td>$f_k(CS), \varepsilon_{pk_U}(CS)$</td>
<td>$f_k(1900), \varepsilon_{pk_U}(1900)$</td>
</tr>
<tr>
<td>$f_k(B), \varepsilon_{pk_U}(B)$</td>
<td>$f_k(Math), \varepsilon_{pk_U}(Math)$</td>
<td>$f_k(1750), \varepsilon_{pk_U}(1750)$</td>
</tr>
<tr>
<td>$f_k(M), \varepsilon_{pk_U}(M)$</td>
<td>$f_k(CS), \varepsilon_{pk_U}(CS)$</td>
<td>$f_k(1800), \varepsilon_{pk_U}(1800)$</td>
</tr>
<tr>
<td>$f_k(O), \varepsilon_{pk_U}(O)$</td>
<td>$f_k(Phy), \varepsilon_{pk_U}(Phy)$</td>
<td>$f_k(2000), \varepsilon_{pk_U}(2000)$</td>
</tr>
<tr>
<td>$f_k(C), \varepsilon_{pk_U}(C)$</td>
<td>$f_k(Math), \varepsilon_{pk_U}(Math)$</td>
<td>$f_k(1600), \varepsilon_{pk_U}(1600)$</td>
</tr>
</tbody>
</table>

\[ \gamma_{A, \text{COUNT}(*)}(D) \]

**Map:**
\[ M \rightarrow R: \{ (\pi_A f_k(t), (\pi_A \varepsilon_{pk_U}(t), \varepsilon_{pk_U}(1))) \} \] \[ t \in D \]

**Reduce:**
\[ \text{count} = \varepsilon_{pk_U}(\sum_{\pi_A f_k(t) \in \text{values}} 1) = \prod_{\pi_A f_k(t) \in \text{values}} \varepsilon_{pk_U}(1) \]
\[ R \rightarrow P : (\pi_A \varepsilon_{pk_U}(t), \text{count}). \]
Secure Private SUM

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_k(A)$, $\mathcal{E}_{pk_U}(A)$</td>
<td>$f_k(\text{CS})$, $\mathcal{E}_{pk_U}(\text{CS})$</td>
<td>$f_k(1900)$, $\mathcal{E}_{pk_U}(1900)$</td>
</tr>
<tr>
<td>$f_k(B)$, $\mathcal{E}_{pk_U}(B)$</td>
<td>$f_k(\text{Math})$, $\mathcal{E}_{pk_U}(\text{Math})$</td>
<td>$f_k(1750)$, $\mathcal{E}_{pk_U}(1750)$</td>
</tr>
<tr>
<td>$f_k(M)$, $\mathcal{E}_{pk_U}(M)$</td>
<td>$f_k(\text{CS})$, $\mathcal{E}_{pk_U}(\text{CS})$</td>
<td>$f_k(1800)$, $\mathcal{E}_{pk_U}(1800)$</td>
</tr>
<tr>
<td>$f_k(O)$, $\mathcal{E}_{pk_U}(O)$</td>
<td>$f_k(\text{Phy})$, $\mathcal{E}_{pk_U}(\text{Phy})$</td>
<td>$f_k(2000)$, $\mathcal{E}_{pk_U}(2000)$</td>
</tr>
<tr>
<td>$f_k(C)$, $\mathcal{E}_{pk_U}(C)$</td>
<td>$f_k(\text{Math})$, $\mathcal{E}_{pk_U}(\text{Math})$</td>
<td>$f_k(1600)$, $\mathcal{E}_{pk_U}(1600)$</td>
</tr>
</tbody>
</table>

$\gamma_{A, \text{SUM}(B)}(D)$

Map:
$\mathcal{M} \rightarrow \mathcal{R}$: \[\{ (\pi_A f_k(t), (\pi_A \mathcal{E}_{pk_U}(t), \pi_B \mathcal{E}_{pk_U}(t))) \}_{t \in D}\]

Reduce:
$\text{sum} = \mathcal{E}_{pk_U}(\sum_{t \in \text{values}} \pi_B(t)) = \prod_{t \in \text{values}} \pi_A f_k(t) \mathcal{E}_{pk_U}(t)$

$\mathcal{R} \rightarrow \mathcal{P}$: \((\pi_A \mathcal{E}_{pk_U}(t), \text{sum})\).
Secure Private AVG

\[ \gamma_{A,\text{AVG}}(B)(D) \]

**Map:**
\[ \mathcal{M} \rightarrow \mathcal{R}: \{(A_{f_k}(t), (A_{\mathcal{E}_{pk_U}}(t), B_{\mathcal{E}_{pk_U}}(t), \mathcal{E}_{pk_U}(1)))\}_{t \in D} \]

**Reduce:**
- \( \text{count} = \mathcal{E}_{pk_U}(\sum_{\pi A_{f_k}(t) \in \text{values}} 1) = \prod_{\pi A_{f_k}(t) \in \text{values}} \mathcal{E}_{pk_U}(1) \)
- \( \text{sum} = \mathcal{E}_{pk_U}(\sum_{\pi A_{f_k}(t) \in \text{values}} B(t)) = \prod_{\pi A_{f_k}(t) \in \text{values}} B_{\mathcal{E}_{pk_U}}(t) \)

\[ \mathcal{R} \rightarrow \mathcal{P} : (A_{\mathcal{E}_{pk_U}}(t), (\text{sum}, \text{count})) \].
Outline

Cryptography
  Pseudo-Random Permutation
  Partial Homomorphic Encryption
  Order Preserving Encryption

Secure-Private MapReduce for COUNT, SUM and AVG

Secure-Private MapReduce for MIN and MAX

Security and Performances

Conclusion
### Preprocessing on data

- All data are encrypted with OPE with a shared key $K_{DU}$
- And encrypted with the public key of the node $pk_C$
- All data $d$ have $f_k(d)$
Secure Private MIN

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_k(A), E_kOU(A)</td>
<td>f_k(CS), E_kOU(CS)</td>
<td>f_k(1900), E_kOU(1900)</td>
</tr>
<tr>
<td>f_k(B), E_kOU(B)</td>
<td>f_k(Math), E_kOU(Math)</td>
<td>f_k(1750), E_kOU(1750)</td>
</tr>
<tr>
<td>f_k(M), E_kOU(M)</td>
<td>f_k(CS), E_kOU(CS)</td>
<td>f_k(1800), E_kOU(1800)</td>
</tr>
<tr>
<td>f_k(O), E_kOU(O)</td>
<td>f_k(Phy), E_kOU(Phy)</td>
<td>f_k(2000), E_kOU(2000)</td>
</tr>
<tr>
<td>f_k(C), E_kOU(C)</td>
<td>f_k(Math), E_kOU(Math)</td>
<td>f_k(1600), E_kOU(1600)</td>
</tr>
</tbody>
</table>

\(\gamma_{A, \text{MIN}(B)}(D)\)

**Map:**
\[ M \rightarrow \mathcal{R} : \{(\pi_A f_k(t), (\pi_A E_{pkU}(t), \pi_B(t)))\} \quad t \in D \]

**Reduce:**
\[ M = \min_{\pi_A f_k(t) \in \text{values}\mathcal{D}} \pi_B(t) \]
\[ \mathcal{R} \rightarrow \mathcal{P} : (\pi_A E_{pkU}(t), M) \]
Outline

Cryptography
  Pseudo-Random Permutation
  Partial Homomorphic Encryption
  Order Preserving Encryption

Secure-Private MapReduce for COUNT, SUM and AVG

Secure-Private MapReduce for MIN and MAX

Security and Performances

Conclusion
Security

Theorem

The SP-SUM, SP-COUNT, SP-AVG, SP-MIN, and SP-MAX protocols securely compute the grouping and aggregation in the ROM in the presence of honest-but-curious adversary even if cloud nodes collude.
Combiners and Improvements

<table>
<thead>
<tr>
<th>COUNT</th>
<th>SUM</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Map can perform some aggregations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MAX & MIN

We can split it into 2 rounds to counter possible frequency attacks against OPE
Performances of COUNT, SUM & AVG

- Avg without combiner
- Avg with combiner
- Sum without combiner
- Sum with combiner
- Count without combiner
- Count with combiner

Seconds vs. Number of tuples/k

- Avg without combiner
- Avg with combiner
- Sum without combiner
- Sum with combiner
- Count without combiner
- Count with combiner
Performances of MIN

Number of tuples/k

Seconds

Nosecure 1Round
Nosecure 2Round
Secure 1Round
Secure 2Round
Outline

Cryptography
- Pseudo-Random Permutation
- Partial Homomorphic Encryption
- Order Preserving Encryption

Secure-Private MapReduce for COUNT, SUM and AVG

Secure-Private MapReduce for MIN and MAX

Security and Performances

Conclusion
Conclusion

- Secure-Private MapReduce: COUNT, SUM, AVG, & MAX MIN
- Using Paillier and OPE
- Honest-but-curious adversary

Next step

- Combinaisons of COUNT, SUM, AVG, MAX & MIN
Questions?

pascal.lafourcade@uca.fr