M²R: ENABLING STRONGER PRIVACY IN MAPREDUCE COMPUTATION

TIEN TUAN ANH DINH, PRATEEK SAXENA, EE-CHIEN CHANG, BENG CHIN OOI, AND CHUNWANG ZHANG, NATIONAL UNIVERSITY OF SINGAPORE

PRESENTED BY: RAVEEN WIJEWICKRAMA

DATE: 03/02/2017
2 OUTLINE

• Introduction
• Proposed System
• Performance Evaluation
• Conclusion
3 INTRODUCTION

- Privacy of data in cloud
- Encryption of data
- Computations on encrypted data: challenging
- Privacy-preserving computation
  - trusted computing support for hardware-isolated computation
  - purely cryptographic techniques
PROBLEM STATEMENT

• Enabling privacy-preserving distributed computation in an un-trusted cloud.
• Distributed computing
  • model in which components of a software system are shared among multiple computers to improve efficiency and performance.
• Focus: privacy in the popular MapReduce framework
5 CONTRIBUTIONS

• Privacy-preserving distributed computation
  • Defines a new level of privacy for MapReduce
  • No algorithmic restructuring needed

• Attacks
  • Mere encryption of data is insufficient

• Simple Design
  • Non-intrusive Architectural change to MapReduce
BACKGROUND
7 MapReduce

- A software framework
- Enables distributed processing of massive data sets across many servers. E.g. Apache Hadoop
- Units of computation: Map & Reduce
  - Map: Takes a set of data and convert it to another form. Output: key-value tuples
  - Reduce: takes input from Map. Combines data tuples into a smaller set of tuples.
8 MapReduce
9 THREAT MODEL

• Passive/honest-but curious attacker
  • Tries to learn sensitive info without deviating from the protocol
• Active/malicious attacker
  • Deviates from protocol and tamper with any data under its control
• Direct Attacks
  • Observing data passing bet. computation nodes
  • Sabotaging computation of each map/reduce instance
10 BASELINE SYSTEM

- Computation units
  - Hardware-isolated
  - Executed privately
  - Side channels are protected

- Intermediate data (bet. Computation units)
  - Encrypted (with authenticated encryption)
  - Decrypted only in a trusted environment

- Can only invoke with its complete input set.
PROBLEM DEFINITION

- Ideally
  - only input/output size, execution time should be leaked

- Baseline System
  - Input, output size, processing time of computation units
  - Data flow among the computation units (MAP→REDUCE)
PROBLEM DEFINITION

• Allowable leakage: $\Psi$
  - Input, output size, processing time of computation units

• Privacy modulo-PSI
A provisioning protocol for a program is modulo-$\Psi$ private if, for any adversary $A$ executing the MapReduce protocol, there is an adversary $\overline{A}$ with access only to $\Psi$, such that the output of $A$ and $\overline{A}$ are indistinguishable.
ASSUMPTIONS

• Underlying hardware sufficiently protects each computation unit from malware and snooping attacks.
• Information leakage via side-channels from a computation unit is minimal.
• To enable arbitrary computation on encrypted data, decryption keys need to be made available to each hardware-isolated computation unit.
WORD-COUNT JOB IN MAP-REDUCE

("Map Reduce (MR) Framework [Gerardnico]", 2017)
ATTACKS

ATTACKS

• Passive Attacks
  • Data flow patterns
  • Order of execution
  • Time of access

• Active Attacks
  • Tuple tampering
    • The adversary may attempt to duplicate or eliminate an entire output tuple-set produced by a computation unit.
  • Misrouting tuples
    • The adversary can reorder intermediate tuples or route data blocks intended for one reduce unit to another.
• Computationally expensive
  • $O(\log^k N)$ for each access when no. of tuples=N
  • With sorting algorithm (used for grouping)
    • $O(\log^{k+1} N)$
    • 30 to 100 times slow
Authors’ observation

• Fixed sequence of data access patterns
  • Cycles of tuple writes followed by reads.

Authors’ solution

• re-write intermediate encrypted tuples with re-randomized tuple keys
  • Such that there is no linkability between the re-randomized tuples and the original encrypted map output tuples.
PROPOSED SYSTEM: M²R

Authors’ solution

• Use a secure mix-network (a cascaded mix-network)
  • To securely shuffle tuples

• Procedure
  • Cascading k-mixing steps
  • mixT units (trusted computation units)
  • Execution distributed over multiple nodes: mixers
PROPOSED SYSTEM: M²R
PROPOSED SYSTEM: M^2R

- Each mixT takes a fixed amount of T tuples
- In each step of the cascade, the mixer utilizes N/T mixT units for mixing (total no.) N tuples.
- At K = \log \frac{N}{T}
  - the strongest possible unlinkability
- MixT unit
  - Decrypts the tuples received
  - Randomly permutes them (O(n) algorithm)
  - Re-encrypts the permuted tuples with new randomly chosen symmetric key.
PROPOSED SYSTEM: M$^2$R

- Secure Grouping
  - Original grouping algorithm can be used.
  - Takes the output of the cascade-mix, group the tuples with the same-key & forwards to the reducers.

- Extra step in cascade
  - Key-component of the output tuples are encrypted using a deterministic symmetric encryption algorithm Fs (with a secret key s).
  - $\{Fs(a), E(a,b)\}$ where E(.): a probabilistic encryption scheme

- Hence,
  - two shuffled tuples with the same tuple-keys will have the same ciphertext for the key-component
  - So grouping algorithm can group them without decrypting the tuples.
PROPOSED SYSTEM: M^2R
PROPOSED SYSTEM: M\textsuperscript{2}R

Integrity Check

- **groupT**
  - Checks to ensure that ordering of grouped tuples passed to reduceT is preserved.
  - Checks for duplicated tuples
  - Checks to ensure that no tuples are dropped.
PROPOSED SYSTEM: M²R

- Achieves private modulo
IMPLEMENTATION

• A modified standard Hadoop implementation
  • to invoke mixT & groupT units
  • 90 LoC to TCB
  • mapT & reduceT: same as baseline system

• Execution
  • Client encrypts & uploads data to M²R nodes
  • User submits M²R applications & finally decrypts the results.
PERFORMANCE EVALUATION

Benchmarks

• HiBench suite: a standard benchmark for Hadoop

Setup

• Bet. 1 to 4 compute nodes (mappers & reducers)
• Bet. 1 to 4 mixer nodes (for mix-network)
• Results shown: From running 4 compute nodes & 4 mixers averaged over 10 executions.
PERFORMANCE EVALUATION

Summary of the porting effort and TCB increase for various M2R applications

<table>
<thead>
<tr>
<th>Job</th>
<th>LoC changed (vs. Hadoop job)</th>
<th>TCB increase (vs. Hadoop codebase)</th>
<th>Input size (vs. plaintext size)</th>
<th>Shuffled bytes</th>
<th># App hypercalls</th>
<th># Platform hypercall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordcount</td>
<td>10 (15%)</td>
<td>370 (0.14%)</td>
<td>2.1G (1.06×)</td>
<td>4.2G</td>
<td>3277173</td>
<td>35</td>
</tr>
<tr>
<td>Index</td>
<td>28 (24%)</td>
<td>370 (0.14%)</td>
<td>2.5G (1.15×)</td>
<td>8G</td>
<td>3277173</td>
<td>59</td>
</tr>
<tr>
<td>Grep</td>
<td>13 (13%)</td>
<td>355 (0.13%)</td>
<td>2.1G (1.06×)</td>
<td>75M</td>
<td>3277174</td>
<td>10</td>
</tr>
<tr>
<td>Aggregate</td>
<td>16 (18%)</td>
<td>395 (0.15%)</td>
<td>2G (1.19×)</td>
<td>289M</td>
<td>18121377</td>
<td>12</td>
</tr>
<tr>
<td>Join</td>
<td>30 (22%)</td>
<td>478 (0.16%)</td>
<td>2G (1.19×)</td>
<td>450M</td>
<td>11010647</td>
<td>14</td>
</tr>
<tr>
<td>Pagerank</td>
<td>42 (20%)</td>
<td>429 (0.15%)</td>
<td>2.5G (4×)</td>
<td>2.6G</td>
<td>1750000</td>
<td>21</td>
</tr>
<tr>
<td>KMeans</td>
<td>113 (7%)</td>
<td>400 (0.12%)</td>
<td>1G (1.09×)</td>
<td>11K</td>
<td>12000064</td>
<td>8</td>
</tr>
</tbody>
</table>

- All jobs have TCB increases of fewer than 500 LoC, merely 0.16% of the Hadoop codebase.
## PERFORMANCE EVALUATION

Overall running time (s) of M2R applications in comparison with other systems

<table>
<thead>
<tr>
<th>Job</th>
<th>Baseline (vs. no encryption)</th>
<th>M²R (% increase vs. baseline)</th>
<th>Download-and-complete (x M²R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordcount</td>
<td>570 (221)</td>
<td>1156 (100%)</td>
<td>1859 (1.6x)</td>
</tr>
<tr>
<td>Index</td>
<td>666 (423)</td>
<td>1549 (130%)</td>
<td>2061 (1.3x)</td>
</tr>
<tr>
<td>Grep</td>
<td>70 (48)</td>
<td>106 (50%)</td>
<td>1686 (15.9x)</td>
</tr>
<tr>
<td>Aggregate</td>
<td>125 (80)</td>
<td>205 (64%)</td>
<td>9140 (44.6x)</td>
</tr>
<tr>
<td>Join</td>
<td>422 (211)</td>
<td>510 (20%)</td>
<td>5716 (11.2x)</td>
</tr>
<tr>
<td>Pagerank</td>
<td>521 (334)</td>
<td>755 (44%)</td>
<td>1209 (1.6x)</td>
</tr>
<tr>
<td>KMeans</td>
<td>123 (71)</td>
<td>145 (17%)</td>
<td>6071 (41.9x)</td>
</tr>
</tbody>
</table>
Normalized break-down time for M2R applications.

Running time: time taken by $mapT + reduceT$ + time taken by the secure shuffler. The rest comes from the Hadoop runtime.
Remaining leakage of M²R applications, compared with that in the baseline system.

<table>
<thead>
<tr>
<th>Job</th>
<th>M²R</th>
<th>Baseline (additional leakage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordcount</td>
<td># unique words + count</td>
<td>word-file relationship</td>
</tr>
<tr>
<td>Index</td>
<td># unique words + count</td>
<td>word-file relationship</td>
</tr>
<tr>
<td>Grep</td>
<td>nothing</td>
<td>nothing</td>
</tr>
<tr>
<td>Aggregate</td>
<td># groups + group size</td>
<td>record-group relationship</td>
</tr>
<tr>
<td>Join</td>
<td># groups + group size</td>
<td>record-group relationship</td>
</tr>
<tr>
<td>Pagerank</td>
<td>node in-degree</td>
<td>whole input graph</td>
</tr>
<tr>
<td>KMeans</td>
<td>nothing</td>
<td>nothing</td>
</tr>
</tbody>
</table>
CONCLUSION

• The proposed design demonstrates
  • a newly defined level of security
  • lower performance overhead
  • Requires only a small TCB.
• M²R shows that the
  • System requires only a little effort to port legacy MapReduce applications
  • Scalable


QUESTIONS?
THANK YOU!