Repeatable Oblivious Shuffling of Large Outsourced Data Blocks

Zhilin Zhang+, Ke Wang, Weipeng Lin, Ada Wai-Chee Fu, Raymond Chi-Wing Wong
+Simon Fraser University, Amazon
Outsourcing in the Cloud

2019 Public cloud services market >$206.2 B

Source: Gartner’s annual forecast of worldwide public cloud service revenue
Sensitive data must be encrypted before putting on the cloud server.
Secure Computation Outsourcing

Trusted client

Semi-trusted Server

Encrypted Data

Computational Task

Result
Encryption is Insufficient

Input: \([a], [b]\)

Task:

\[
\text{if } a>b: \\
\text{branch 1} \quad \text{a=2, b=1} \\
\text{else:} \\
\text{branch 2} \quad \text{a=1, b=2}
\]

Oblivious algorithm: make the control flow be independent of the input data

- oblivious transfer/ sorting/ shuffling, etc.
Problem

Oblivious Shuffling (OS)

A shuffling of $n$ encrypted data blocks $[B] = ([B_1], \cdots, [B_n])$ according to a permutation $\pi$ is oblivious if the server is unable to infer $\pi$.

Untrackable

which is which
Application

- Private data access (hide access pattern)
- Private data integration/sharing (hide data source)
- Coin mixing in cryptocurrency (hide owner anonymity)
State of the Art

All existing OS methods rely on the movement of outsourced data to the client.

download for shuffling

heavy communication for shuffling
large-sized blocks

download for peel-off
Repeatable Oblivious Shuffle

Definition

An oblivious shuffle of $[B] = ([B_1], \cdots, [B_n])$ is repeatable if it is performed by the server without increasing encryption layers.
Preliminaries

Homomorphic matrix multiplication

\[
[M_1] \odot M_2 = [M_1 \cdot M_2]
\]

Matrix based data shuffling

\[
B \cdot \pi = (B_1, B_2) \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \Rightarrow (B_2, B_1)
\]
Main Idea

Key Requirements

• **repeatability**: server side shuffling, no increase in encryption layers

• **obliviousness**: shuffling must be oblivious

\[ H = \pi \]

split the information of \( \pi \) into plaintext \( H \) and some ciphertext \([H_A]\)
Formalization

\[ [B^{(\eta)}] \leftarrow ROS(\pi^{(\eta)}, [B^{(\eta-1)}]) \]

- data after shuffling
  \[ [B^{(\eta)}] = [B \cdot \pi^{\eta-1} \cdot \pi(\eta)] \]
- permutation matrix
- data before shuffling
  \[ [B^{(\eta-1)}] = [B \cdot \pi^{\eta-1}] \]

server side shuffling

single layer encryption

hide \( \pi^{(\eta)} \)
Construction

1. pick $\pi^{(\eta)}$

2. compute $H^{(\eta)}$ and $[H_A^{(\eta)}]$

3. compute the shuffling result by

$$H^{(\eta)}$$ and $[H_A^{(\eta)}]$
Analysis

Correctness

\[
\pi^{\eta-1} \cdot \pi^{(\eta)} = H_A^{(\eta)} \cdot \pi^{\eta-1}
\]

Obliviousness

\[
\pi^{\eta-1} \cdot \pi^{(\eta)} = H_A^{(\eta)} \cdot \pi^{\eta-1}
\]
## Experimental Settings

<table>
<thead>
<tr>
<th></th>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Our approach</strong></td>
<td><strong>ROS</strong></td>
<td>Server-side shuffling without increasing encryption layer</td>
</tr>
<tr>
<td><strong>Baseline</strong></td>
<td><strong>ClientShuffle</strong></td>
<td>Client-side shuffling (download data for every shuffling)</td>
</tr>
<tr>
<td></td>
<td>LayeredShuffle ((l = 2))</td>
<td>Service-side shuffling with increasing encryption layers</td>
</tr>
<tr>
<td></td>
<td>LayeredShuffle ((l = 10))</td>
<td>Service-side shuffling with increased encryption layers</td>
</tr>
</tbody>
</table>
Effect of Block Size $m$

Shuffle cost w.r.t. block size $m$ (MB) ($n = 4$, ClientShuffle has no server computation and thus not reported)
Effect of Block Number $n$

Shuffle cost w.r.t. block number $n$ (m=10 MB, ClientShuffle has no server computation and not reported)