SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning

Nishat Koti, Mahak Pancholi, Arpita Patra, Ajith Suresh
Indian Institute of Science, Bangalore

30th USENIX Security Symposium

*Full Version: https://eprint.iacr.org/2020/592

Privacy in Machine Learning

- Machine Learning (ML)
  - Automobile, Healthcare, Finance, .....

Image credits: Abdul Wahid
Privacy in Machine Learning

- Machine Learning (ML)
  - Automobile, Healthcare, Finance, .....
- Phases: Training & Inference

Image credits: Abdul Wahid
Privacy in Machine Learning

ML Training

Data owner

Data owner

ML Inference

Client

Query

Result

Data owner

Data owner

aws

Azure

IBM Cloud

Google Cloud
Privacy in Machine Learning

• Machine Learning (ML)
  • Automobile, Healthcare, Finance, ....

• Phases: Training & Inference

• More the data -> better the model
Privacy in Machine Learning

• Machine Learning (ML)
  • Automobile, Healthcare, Finance, ..... 

• Phases: Training & Inference 

• More the data -> better the model 

• Privacy of the data ??
Secure Outsourced Computation (SOC)

- Computation -> hired powerful servers
Secure Outsourced Computation (SOC)

• Computation -> hired powerful servers

• Average end users benefit
Secure Outsourced Computation (SOC)

• Computation -> hired powerful servers

• Average end users benefit

• Solves issue for large population
Secure Outsourced Computation (SOC)

- Computation -> hired powerful servers
- Average end users benefit
- Solves issue for large population

- Desired properties:
  - Privacy
  - Efficiency
  - Robustness
Secure Outsourced Computation (SOC)

- Computation -> hired powerful servers
- Average end users benefit
- Solves issue for large population

Desired properties:
- Privacy
- Efficiency
- Robustness

Secure Multiparty Computation (MPC)
Secure Multiparty Computation [Yao’82]

- MPC allows a set of parties with private inputs to compute some joint function of their inputs
Secure Multiparty Computation [Yao’82]

- MPC allows a set of parties with private inputs to compute some joint function of their inputs.

- Properties of MPC:
  - Correctness – Parties should obtain the correct function output.
Secure Multiparty Computation [Yao’82]

• MPC allows a set of parties with private inputs to compute some joint function of their inputs

• Properties of MPC:
  - **Correctness** – Parties should obtain the correct function output
  - **Privacy** – Nothing more than the function output should be revealed
Secure Multiparty Computation [Yao’82]

- Adversary
  - Semi-honest vs Malicious
Secure Multiparty Computation [Yao’82]

- Adversary
  - Semi-honest vs Malicious
  - Honest majority vs Dishonest majority
Secure Multiparty Computation [Yao’82]

- Security
  - Abort
Secure Multiparty Computation [Yao’82]

- Security
  - Abort
  - Fairness
Secure Multiparty Computation [Yao’82]

- Security
  - Abort
  - Fairness
  - Guaranteed Output Delivery (GOD)
SWIFT Protocol
SWIFT Protocol

- 3PC and 4PC protocols for malicious corruption

- MPC for small population
- Efficiency and simplicity
Swift Protocol

• 3PC and 4PC protocols for malicious corruption

• Honest-majority setting
  • at most 1 corruption
• 3PC and 4PC protocols for malicious corruption

• Honest-majority setting
  • at most 1 corruption

• Guaranteed Output Delivery
SWIFT Protocol

- 3PC and 4PC protocols for malicious corruption
- Honest-majority setting
  - at most 1 corruption
- Guaranteed Output Delivery
- Preprocessing Model
  - preprocessing phase

- input-independent computation
- relatively slow and expensive
SWIFT Protocol

- 3PC and 4PC protocols for malicious corruption
- Honest-majority setting
  - at most 1 corruption
- Guaranteed Output Delivery
- Preprocessing Model
  - preprocessing phase
  - online phase

- input-dependent computation
- super fast
SWIFT Protocol: Results

- Communication cost per multiplication gate

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>Boyle et al. 19</td>
<td>-</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>BLAZE [PS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td>SWIFT (3PC)</td>
<td><strong>SWIFT (3PC)</strong></td>
<td><strong>3</strong></td>
<td><strong>3</strong></td>
<td><strong>GOD</strong></td>
</tr>
</tbody>
</table>
SWIFT Protocol: Results

- Communication cost per multiplication gate

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>Boyle et al. 19</td>
<td>-</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>3PC</td>
<td>BLAZE [PS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td>SWIFT (3PC)</td>
<td></td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
</tbody>
</table>
## SWIFT Protocol: Results

- Communication cost per multiplication gate

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>Boyle et al. 19</td>
<td>-</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>3PC</td>
<td>BLAZE [PS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td>SWIFT (3PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
<td></td>
</tr>
</tbody>
</table>
SWIFT Protocol: Results

- Communication cost per multiplication gate

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>Boyle et al. 19</td>
<td>-</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>BLAZE [PS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>SWIFT (3PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>4PC</td>
<td>Trident [CRS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>FLASH [BCPS20]</td>
<td>6</td>
<td>6</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>SWIFT (4PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
</tbody>
</table>
SWIFT Protocol: Results

- Communication cost per multiplication gate

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>Boyle et al. 19</td>
<td>-</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>BLAZE [PS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>SWIFT (3PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>4PC</td>
<td>Trident [CRS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>FLASH [BCPS20]</td>
<td>6</td>
<td>6</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>SWIFT (4PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
</tbody>
</table>
SWIFT Protocol: Results

- Communication cost per multiplication gate

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>Boyle et al. 19</td>
<td>-</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>BLAZE [PS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>SWIFT (3PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>4PC</td>
<td>Trident [CRS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>FLASH [BCPS20]</td>
<td>6</td>
<td>6</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>SWIFT (4PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
</tbody>
</table>
SWIFT Protocol: Results in PPML

- PPML Building Blocks
  - Dot Product

\[ \vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i \]
SWIFT Protocol: Results in PPML

- PPML Building Blocks
  - Dot Product

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>BLAZE [PS20]</td>
<td>3d</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>SWIFT (3PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>4PC</td>
<td>Trident [CRS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>FLASH [BCPS20]</td>
<td>6</td>
<td>6</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>SWIFT (4PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
</tbody>
</table>

Communication cost per dot product (d-length vectors)

\[ \vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i \]
**SWIFT Protocol: Results in PPML**

- **PPML Building Blocks**
  - **Dot Product**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>BLAZE [PS20]</td>
<td>3d</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>SWIFT (3PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>4PC</td>
<td>Trident [CRS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>FLASH [BCPS20]</td>
<td>6</td>
<td>6</td>
<td>GOD</td>
</tr>
</tbody>
</table>

**Communication cost per dot product (d-length vectors)**

\[
\vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i
\]
SWIFT Protocol: Results in PPML

- PPML Building Blocks
  - Dot Product
  - Truncation
  - Comparison
  - Bit to arithmetic conversions
  - Non-linear activation functions

<table>
<thead>
<tr>
<th>Setting</th>
<th>Reference</th>
<th>Pre-processing (#elements)</th>
<th>Online (#elements)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>3PC</td>
<td>BLAZE [PS20]</td>
<td>3d</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>SWIFT (3PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
<tr>
<td>4PC</td>
<td>Trident [CRS20]</td>
<td>3</td>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td></td>
<td>FLASH [BCPS20]</td>
<td>6</td>
<td>6</td>
<td>GOD</td>
</tr>
<tr>
<td></td>
<td>SWIFT (4PC)</td>
<td>3</td>
<td>3</td>
<td>GOD</td>
</tr>
</tbody>
</table>

Communication cost per dot product (d-length vectors)

\[
\vec{x} \odot \vec{y} = \sum_{i=1}^{d} x_i y_i
\]
SWIFT Joint Message Passing (jmp) primitive

Sender 1

$\mathbf{m}$

Receiver

$\mathbf{m}$

Sender 2
SWIFT Joint Message Passing (jmp) primitive

Sender 1 \( m \)

\( F_{jmp} \)

Sender 2 \( m' \)

Receiver
SWIFT Joint Message Passing (jmp) primitive

\[ F_{jmp} \]

\[ m = m' \]
SWIFT Joint Message Passing (jmp) primitive

Sender 1

Sender 2

$F_{jmp}$

$m \neq m'$

Receiver
SWIFT Joint Message Passing (jmp) primitive
SWIFT Joint Message Passing (jmp) primitive

\[ op = f(x, y, z) \]

Sender 1

Sender 2

Receiver
3PC Joint Message Passing (jmp) primitive

\[ m \rightarrow R \]
\[ m' \rightarrow H(m') \]

\[ H(m) \neq H(m') \]
\[ b = \begin{cases} 
1 & \text{if } H(m) \neq H(m') \\
0 & \text{otherwise}
\end{cases} \]
3PC Joint Message Passing (jmp) primitive

If $b = 1$

$S_1$ : $H(m)$

$S_2$ : $H(m')$

$R$ :

$$b = \begin{cases} 
1 & \text{if } H(m) \neq H(m') \\
0 & \text{otherwise} \end{cases}$$
3PC Joint Message Passing (jmp) primitive

\[ b = \begin{cases} 
1 & \text{if } H(m) \neq H(m') \\
0 & \text{otherwise} 
\end{cases} \]
3PC Joint Message Passing (jmp) primitive

\[ b = \begin{cases} 
1 & \text{if } H(m) \neq H(m') \\
0 & \text{otherwise} 
\end{cases} \]
3PC Joint Message Passing (jmp) primitive

\[\text{S1} \quad m \quad \text{R} \quad m' \quad \text{H}(m') \quad \text{S2}\]

\[\text{S1} \quad \text{R} \quad \text{b} \quad \text{H}(m) \overset{?}{=} \text{H}(m') \quad \text{S2}\]

\[b = \begin{cases} 1 & \text{if } H(m) \neq H(m') \\ 0 & \text{otherwise} \end{cases}\]
3PC Joint Message Passing (jmp) primitive

Send phase of jmp
3PC Joint Message Passing (jmp) primitive

Verify phase of jmp

\[ m \rightarrow R \]
\[ m' \rightarrow H(m') \]
\[ b = \begin{cases} 
1 & \text{if } H(m) \neq H(m') \\
0 & \text{otherwise}
\end{cases} \]

\[ R \rightarrow b \]

\[ \text{Verify phase of jmp} \]
4PC Joint Message Passing (jmp) primitive

\[ F_{jmp} \]

\[ m = m' \]

jmp for 4PC
4PC Joint Message Passing (jmp) primitive

\[ F_{jmp} \]

\[ m \neq m' \]

jmp for 4PC
Table 6: Logistic Regression training and inference. TP is given in (#it/min) for training and (#queries/min) for inference.
### Table 7: 3PC NN Inference. TP is given in (#queries/min).

<table>
<thead>
<tr>
<th>Network</th>
<th>Ref.</th>
<th>Online</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Latency (s)</td>
<td>Com [MB]</td>
</tr>
<tr>
<td>NN-1</td>
<td>BLAZE SWIFT</td>
<td>1.92</td>
<td>0.04</td>
</tr>
<tr>
<td>NN-2</td>
<td>BLAZE SWIFT</td>
<td>4.77</td>
<td>3.54</td>
</tr>
<tr>
<td>NN-3</td>
<td>BLAZE SWIFT</td>
<td>15.58</td>
<td>52.58</td>
</tr>
</tbody>
</table>

### Table 8: 4PC NN Inference. TP is given in (#queries/min).

<table>
<thead>
<tr>
<th>Network</th>
<th>Ref.</th>
<th>Online</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Latency (s)</td>
<td>Com [MB]</td>
</tr>
<tr>
<td>NN-1</td>
<td>FLASH SWIFT</td>
<td>1.70</td>
<td>0.06</td>
</tr>
<tr>
<td>NN-2</td>
<td>FLASH SWIFT</td>
<td>3.93</td>
<td>5.51</td>
</tr>
<tr>
<td>NN-3</td>
<td>FLASH SWIFT</td>
<td>12.65</td>
<td>82.54</td>
</tr>
</tbody>
</table>

NN-1: [MR18, PS20]  
NN-2: [LBBH98]  
NN-3: [SZ14]  

SWIFT: Benchmarking
Thank you