Trusted Computing and Analytics

N. Asokan
(joint work with Mika Juuti, Jian Liu, Samuel Marchal and Andrew Paverd)
Established in 2010, named in honour of Alvar Aalto, the famous Finnish architect.

Science and art meet technology and business.
Promoting entrepreneurship

70 to 100 Companies are founded every year in our ecosystem

MIT Skolltech initiative rated Aalto’s innovation ecosystem among the top-5 rising stars in the world

Entrepreneurship is a more popular career option than ever – in the last four years, over 2,000 students have studied through the Aalto Ventures Program

50% of Finnish startups that originate from universities come from the Aalto community
Nobody in their right mind would come to Helsinki in November.
About Us

Aalto Department of Computer Science
• Features in top-100 in CS world rankings; AI, algorithms, security & privacy, …
  http://cs.aalto.fi/

Secure Systems Group (Asokan)
• 10 senior researchers, 5-10 MSc students; Part of the previous ICRI-SC
  http://cs.aalto.fi/secure_systems

Me
• In academia since 2012; Earlier: industrial research labs (IBM ZRL and Nokia NRC)
  Twitter: @nasokan, https://asokan.org/asokan/
Aalto University plans in CARS Lab

(Machine) Learning

Consensus

Recovery

Trusted Computing
How do we evaluate ML-based systems?

Effectiveness of inference
• accuracy/score measures on held-out test set?

Performance
• inference speed and memory consumption?

Hardware/software requirements
• e.g. memory/processor limitations, or specific software library?
Meeting requirements in the presence of an adversary
Machine learning pipeline

Legend
○ entities
□ components
1. Malicious data owners

<table>
<thead>
<tr>
<th>Attack target</th>
<th>Risk</th>
<th>Remedies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model (integrity)</td>
<td>Data poisoning [1, 2]</td>
<td>Access control, Robust estimators, Active learning (human-in-the-loop learning), Outlier removal / normality models</td>
</tr>
</tbody>
</table>

2. Malicious pre-processor

<table>
<thead>
<tr>
<th>Attack target</th>
<th>Risk</th>
<th>Remedies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model (integrity)</td>
<td>Data poisoning</td>
<td>Access control, Robust estimators, Active learning (human-in-the-loop learning), Outlier removal / normality models</td>
</tr>
<tr>
<td>Training data (confidentiality)</td>
<td>Unauthorized data use (e.g. profiling)</td>
<td>Adding noise (e.g. differential privacy), Oblivious aggregation [1] (e.g., homomorphic encryption)</td>
</tr>
</tbody>
</table>

[1] Heikkila et al. ”Differentially Private Bayesian Learning on Distributed Data”, NIPS’17
3. Malicious model trainer

<table>
<thead>
<tr>
<th>Attack target</th>
<th>Risk</th>
<th>Remedies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data (confidentiality)</td>
<td>Unauthorized data use</td>
<td>Oblivious training (learning with encrypted data) [1]</td>
</tr>
<tr>
<td>(e.g. profiling)</td>
<td>(e.g. profiling)</td>
<td></td>
</tr>
</tbody>
</table>

4. Malicious inference service provider

<table>
<thead>
<tr>
<th>Attack target</th>
<th>Risk</th>
<th>Remedies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference queries/results (confidentiality)</td>
<td>Unauthorized data use (e.g. profiling)</td>
<td>Oblivious inference [1,2,3]</td>
</tr>
</tbody>
</table>

5. Malicious client

<table>
<thead>
<tr>
<th>Attack target</th>
<th>Risk</th>
<th>Remedies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data</td>
<td>Membership inference</td>
<td>Minimize information leakage in responses</td>
</tr>
<tr>
<td></td>
<td>Model inversion</td>
<td>Differential privacy</td>
</tr>
<tr>
<td>(confidentiality)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normality model for client queries</td>
</tr>
</tbody>
</table>

Oblivious Neural Network Predictions via MiniONN Transformations

Jian Liu, Mika Juuti, Yao Lu, N. Asokan
ACM CCS 2017
Machine learning as a service (MLaaS)

Client

Inference service provider

Input

Predictions

violation of clients’ privacy
Running predictions on client-side

- Model theft
- Evasion
- Model inversion
Oblivious Neural Networks (ONN)

Given a neural network, is it possible to make it oblivious?

- server learns nothing about clients' input;
- clients learn “nothing” about the model.
Example: CryptoNets

- Uses fully homomorphic encryption (FHE)
- High throughput for batch queries from same client
- High overhead for single queries: 297.5s and 372MB
- Only supports low-degree polynomials

[GDLLNW16] CryptoNets, ICML 2016
MiniONN: Overview

Blinded input

oblivious protocols

Blinded predictions

**Lightweight primitives:**
- Additively homomorphic encryption (with SIMD)
- Secure two-party computation
Example \( z = W' \cdot f(W \cdot x + b) + b' \)

\[
\begin{align*}
 x &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},
 W &= \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix},
 b &= \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},
 W' &= \begin{bmatrix} w'_{1,1} & w'_{1,2} \\ w'_{2,1} & w'_{2,2} \end{bmatrix},
 b' &= \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix}
\end{align*}
\]

All operations are in a finite field.
Core idea: use secret sharing for oblivious computation

For a layer with input $x$ and output $y$, at the beginning:
arrange for client & server to have shares $x^c$ and $x^s$ s.t $x^s + x^c = x$

At the end of **oblivious processing**:
client & server to have shares $y^c$ and $y^s$ s.t $y^s + y^c = y$

https://eprint.iacr.org/2017/452

Skip to performance
Secret sharing initial input $x$

$x_1^c, x_2^c \leftarrow Z_N$

$x_1^s := x_1 - x_1^c$, $x_2^s := x_2 - x_2^c$

Note that $x^c$ is independent of $x$. Can be **pre-chosen**
Oblivious matrix multiplication $W \cdot x + b$

$$
\begin{bmatrix}
    w_{1,1} & w_{1,2} \\
    w_{2,1} & w_{2,2}
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2
\end{bmatrix}
+ \begin{bmatrix}
    b_1 \\
    b_2
\end{bmatrix}
= \begin{bmatrix}
    w_{1,1} & w_{1,2} \\
    w_{2,1} & w_{2,2}
\end{bmatrix}
\begin{bmatrix}
    x_1^s + x_1^c \\
    x_2^s + x_2^c
\end{bmatrix}
+ \begin{bmatrix}
    b_1 \\
    b_2
\end{bmatrix}

= w_{1,1}(x_1^s + x_1^c) + w_{1,2}(x_2^s + x_2^c) + b_1
+ w_{2,1}(x_1^s + x_1^c) + w_{2,2}(x_2^s + x_2^c) + b_2

= w_{1,1}x_1^s + w_{1,2}x_2^s + b_1 + w_{1,1}x_1^c + w_{1,2}x_2^c
+ w_{2,1}x_1^s + w_{2,2}x_2^s + b_2 + w_{2,1}x_1^c + w_{2,2}x_2^c

Compute locally by the server

Dot-product

A setup phase protocol can prepare “dot product triplets”

https://eprint.iacr.org/2017/452
Oblivious activation/pooling functions $f(y)$

- **Piecewise linear functions** e.g., ReLU: $x^s + x^c := \max(y^s + y^c, 0)$
  - easily computed obliviously using a garbled circuit

- **Smooth functions** e.g., Sigmoid: $x^s + x^c := 1/(1 + e^{-(y^s + y^c)})$
  - approximate by a set of piecewise linear functions
  - then compute obliviously using a garbled circuit
  - empirically: ~14 segments sufficient
Core idea: use secret sharing for oblivious computation

For a layer with input $x$ and output $y$, at the beginning:

arrange for client & server to have shares $x^c$ and $x^s$ s.t. $x^s + x^c = x$

At the end of oblivious processing:

client & server to have shares $y^c$ and $y^s$ s.t. $y^s + y^c = y$

$\text{W' \cdot [ ] + b'}$

$\text{W \cdot [ ] + b}$

$(y^c + y^s = y)$

$(x^c + x^s = x')$

$(y^c + y^s = y)$

$(x^c + x^s = x)$
## Performance (for single queries)

<table>
<thead>
<tr>
<th>Model</th>
<th>Latency (s)</th>
<th>Msg sizes (MB)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST/Square</td>
<td>0.4 (+ 0.88)</td>
<td>44 (+ 3.6)</td>
<td>98.95%</td>
</tr>
<tr>
<td>CIFAR-10/ReLU</td>
<td>472 (+ 72)</td>
<td>6226 (+ 3046)</td>
<td>81.61%</td>
</tr>
<tr>
<td>PTB/Sigmoid</td>
<td>4.39 (+ 13.9)</td>
<td>474 (+ 86.7)</td>
<td>120/114.5 (perplexity)</td>
</tr>
</tbody>
</table>

Setup phase timings in parentheses
<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>300x-700x faster than CryptoNets</td>
<td>Still ~1000x slower than without privacy</td>
</tr>
<tr>
<td>Can transform any given neural network to its oblivious variant</td>
<td>Server can no longer filter requests or do sophisticated metering</td>
</tr>
<tr>
<td></td>
<td>Assumes online connectivity to server</td>
</tr>
<tr>
<td></td>
<td>Reveals structure (but not params) of NN</td>
</tr>
</tbody>
</table>
Can trusted computing help?

Hardware support for
- Isolated execution: Trusted Execution Environment
- Protected storage: Sealing
- Ability to report status to a remote verifier: Attestation

Other Software

Trusted Software

Protected Storage

Root of Trust

Cryptocards
https://www.ibm.com/security/cryptocards/

Trusted Platform Modules
https://www.infineon.com/tpm

ARM TrustZone
https://www.arm.com/products/security-on-arm/trustzone

Intel Software Guard Extensions
https://software.intel.com/en-us/sgx
Using a client-side TEE to vet input

1. Attest client’s TEE app
2. Provision filtering policy
3. Input
4. Input, “Input/Metering Certificate”
5. MiniONN protocol + “Input/Metering Certificate”

MiniONN + policy filtering + advanced metering
Using a client-side TEE to run the model

1. Attest client’s TEE app
2. Provision model configuration, filtering policy
3. Input
4. Predictions + “Metering Certificate”
5. “Metering Certificate”

MiniONN + policy filtering + advanced metering
+ disconnected operation + performance + better privacy
- harder to reason about model secrecy
Applications of machine learning must consider threats from possible adversaries.

MiniONN: Efficiently transform any given neural network into oblivious form with no/negligible accuracy loss

Trusted Computing can help realize improved security and privacy for ML

https://eprint.iacr.org/2017/452
Fin.