Secure MLaaS with Temper: Trusted and Efficient Model Partitioning and Enclave Reuse

Fabing Li\textsuperscript{1,2}, Xiang Li\textsuperscript{3}, Mingyu Gao\textsuperscript{2,3,4}

\textsuperscript{1} Xi’an Jiaotong University  
\textsuperscript{2} Institute for Interdisciplinary Information Core Technology  
\textsuperscript{3} Tsinghua University  
\textsuperscript{4} Shanghai Artificial Intelligence Lab Shanghai

https://github.com/tsinghua-ideal/TEMPER-Secure-MLaaS
Background

- DNN is widely-used in many applications

- Machine Learning as a Service (MLaaS) becomes popular
  - Accessibility: simplified and user-friendly interfaces
  - Rapid Prototyping and Development
  - Scalability, Flexibility and Reduced Costs
  - Integration and Compatibility with Existing Workflows

Face recognition  Autonomous driving  Voice recognition  Smart Assistant
Security Issues

Data Breaches
- Cyber-attacks
- Adversarial Attacks
- Secure Runtime

Data Privacy
- Untrusted Service Provider
- Malicious Tenants
- Data Isolation

Regulatory Compliance
- GDPR
- HIPAA
- Data Visibility

🤔🤔 How to solve these issues?
Solutions

- Trusted Execution Environment (TEE) oriented approaches
- Cryptography oriented approaches

<table>
<thead>
<tr>
<th>APPROACH</th>
<th>SECURITY LEVEL</th>
<th>EFFICIENCY</th>
<th>LIMITATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEE</td>
<td>System</td>
<td>Relatively High</td>
<td>Hardware Side-channels</td>
</tr>
<tr>
<td>Cryptography</td>
<td>Cryptographic</td>
<td>Low</td>
<td>Accuracy Computation Communication</td>
</tr>
<tr>
<td>• HE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• MPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

🌟🌟 We focus on TEE-based MLaaS !
Intel SGX

- Enclave: a hardware-protected memory region
- Provide attestation for code and data inside the enclave
- Obstruct OS-level and physical attackers
- High accessibility: Azure, Alibaba Cloud, etc.
Secure MLaaS

- Integrate runtime with Intel SGX SDK
- Start and load the model weights into the enclave
- Verify the runtime by attestation after initialization
- Receive user requests and make inferences
Goals

- Not sacrificing the security
- Unchanged accuracy
- Comparable efficiency and scalability
Challenges

Why is TEE-based MLaaS slow?

- Enclave initialization
  - Attestation
- Model Loading
  - Load model weights
- Secure Paging
  - Performance degradation due to limited EPC size

Performance breakdown of baseline secure MLaaS
Overview

👉 Approaches

💡 Enclave reuse for enclave initialization and model loading

💡 Model partitioning for secure paging

[Diagram showing model instances and user interactions]

Legends:
- Untrusted
- Trusted
- Trusted Third Party

Intel Attestation Service (IAS)
Enclave Reuse

- Long running enclave
  - Leader-Worker topology
  - Preloaded model weights

👍👍 No Model Loading and Scalability

🤔🤔 How to solve the security issues?
  e.g. attestation for enclaves, data interference, data residual

Cloud Resource Management
Enclave Reuse

- Traditional attestation
  - One report for one user
  - One report for one enclave

💡💡 Report validation happens without the enclave

🤔 Can we verify a pre-generated report?
Enclave Reuse

- Traditional attestation
  - One report for one user
  - One report for one enclave

💡💡 Report validation happens without the enclave

🤔 Can we verify a pre-generated report?
Enclave Reuse

- Traditional attestation
  - One report for one user
  - One report for one enclave

💡💡 Report validation happens without the enclave

🤔 Can we verify a pre-generated report?
Enclave Reuse

- Report reuse
  - One report for all users
  - One report for all enclaves

👍 No more enclave initialization
Enclave Reuse

• Report reuse
  • One report for all users
  • One report for all enclaves

👍 No more enclave initialization
Enclave Reuse

• Report reuse
  • One report for all users
  • One report for all enclaves

👍 No more enclave initialization
Enclave Reuse

- NN models are stateless
- Models are read-only for inferences
- Data streams flow with deterministic sizes and at fixed time

<table>
<thead>
<tr>
<th>Batch 1</th>
<th>Data</th>
<th>Data</th>
<th>Data</th>
<th>Padding</th>
<th>Padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch 2</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Batch 3</td>
<td>Data</td>
<td>Data</td>
<td>Padding</td>
<td>Padding</td>
<td>Padding</td>
</tr>
<tr>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>Batch n</td>
<td>Data</td>
<td>Padding</td>
<td>Padding</td>
<td>Padding</td>
<td>Padding</td>
</tr>
</tbody>
</table>
Model Partitioning

🔍 How to get the optimal runtime efficiency?

- **Alleviate the secure paging**
  - Partition the model to fit in the limited EPC size

- **Acceptable communication cost**
  - Replace model loading with inter-enclave communication
Model Partitioning

Optimization Goals

Is it worth to have some secure paging cost to avoid large communication overheads?

$$\min \left\{ n \times \max_i \{ t_{\text{comp},i}, t_{\text{comm},i} \} \right\} \quad \leftarrow \text{high-throughput}$$

$$\min \left\{ \sum_i t_{\text{comp},i} + t_{\text{comm},i} \right\} \quad \leftarrow \text{low-latency}$$

$t_{\text{comp}}$: Computation costs $\quad \leftarrow$ Memory consumption

$t_{\text{comm}}$: Communication costs $\quad \leftarrow$ Data Size

Basic units

- Fusing small operators
- Split large operators

ยาย Find the best partition, and do it faster
Model Partitioning

- Latency Estimation Model
  - Computation cost
    - Memory ballooning for latency with secure paging
    - Normal execution for latency without secure paging
  - Communication cost

- Solving Optimized Partitioning
  - Partition (10 to 100) units to (1 to n) enclaves
  - Exhaustive search

👍 Get all possible costs

😒 Not so slow, compared to latency estimation
Evaluation

Implementation
Intel SGX v2.9
Fortanix Rust enclave development platform (EDP)
TVM v0.7 for ML models

Platform
4 servers
Intel Core i7-9700 CPU
1 Gbps Ethernet

Experiment Setup
ImageNet
MobileNetV1, ResNet18/50/152, VGG19, InceptionV3, and DenseNet201
Evaluation

• Baselines: runtime optimization and careful memory management
Latency

- Comparison when optimizing for low-latency

😊 10 × speedup against TensorSCONE
😊 4.9 × speedup against Myelin
😊 2.2 × speedup against Lasagna
😊 2.1 × slowdown for the untrusted
Throughput

- Throughput using batch sizes 1, 4, and 16

😊 1.8 ×, 2.1 ×, and 1.2 × higher throughput over Lasagna with batch size 1, 4, and 16

🧐 Increasing batch sizes will not always result in throughput improvements, because larger batch sizes incur more secure paging
Report generation inside the enclave takes close to half a second.

😊 4 × faster on overall performance.

<table>
<thead>
<tr>
<th>Attestation (msec)</th>
<th>Server</th>
<th>User</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>462.43</td>
<td>111.25</td>
<td>573.68</td>
</tr>
<tr>
<td>TEMPER</td>
<td>30.48</td>
<td>112.97</td>
<td>143.45</td>
</tr>
</tbody>
</table>
Partitioning Strategies

ılma The effect of appropriate secure paging instead of strict partition size

💡 It is necessary to allow secure paging sometimes

<table>
<thead>
<tr>
<th>Model</th>
<th>TEMPER(img/sec/server)</th>
<th>DNN-Partition (img/sec/server)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV1</td>
<td>41.57</td>
<td>41.53</td>
</tr>
<tr>
<td>ResNet18</td>
<td>23.96</td>
<td>15.37</td>
</tr>
<tr>
<td>ResNet50</td>
<td>9.41</td>
<td>2.47</td>
</tr>
<tr>
<td>ResNet152</td>
<td>1.89</td>
<td>2.67</td>
</tr>
<tr>
<td>VGG19</td>
<td>0.60</td>
<td>0.39</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>4.39</td>
<td>1.06</td>
</tr>
<tr>
<td>DenseNet201</td>
<td>4.62</td>
<td>2.74</td>
</tr>
</tbody>
</table>

* DNN-Partition assumes a heterogeneous system with many accelerators and CPUs
Conclusion

👍 In-depth analysis on TEE-based secure MLaaS designs and identify three key performance inefficiencies: enclave initialization, model loading, and limited trusted memory space.

👍 Propose a trusted and efficient MLaaS system, TEMPER, improving performance while not sacrificing security guarantees or inference accuracy.

👍 Outperform the SOTA baseline by over $2 \times$ in terms of latency and throughput

Questions?