ML Property Attestation using TEEs

1. Introduction
   - Measured model and dataset metrics used to demonstrate the quality of models & inferences
   - Need to link dataset, training parameters to model, model to inference input/output
   - New advances (e.g., Intel AMX+SGXv2) allow training/running complex models within TEEs

2. The problem
   - Cryptographic proofs inefficient or don’t scale
   - ML-based methods are inaccurate
   - Current methods focus only on specific properties
   - Current certification services require outsourcing both training and inference

3. Our solution
   Use remote attestation to run ML software and prove properties like:
   - Which model produced an inference
   - How accurate is the model
   - How was the model trained
   - What data was used to train it
   - How representative was the training set

4. Implementation
   - SGX enclaves perform ML tasks and attest process/performance claims
   - Verifier combines attestations to link output to input, model, training dataset
   - Trust in claims derives from trust in TEE

5. Conclusion
   TEE-based ML property attestation is efficient, scalable & versatile

---

Introduction to Attested ML architecture. Enclaves (represented as boxes) hosting models measure/attest metrics for training data, model, and inference operations for confidence in model & inferences.

<table>
<thead>
<tr>
<th></th>
<th>I/O Binding (100 operations)</th>
<th>Accuracy (ms)</th>
<th>Proof of Training (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>32029ms</td>
<td>36470ms</td>
<td>36.5s</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>0.5ms</td>
<td>294ms</td>
<td>4.4s</td>
</tr>
<tr>
<td>Computation</td>
<td>70.1ms</td>
<td>3490ms</td>
<td>514s</td>
</tr>
<tr>
<td>Proving</td>
<td>6.6ms</td>
<td>5.68ms</td>
<td>0.005s</td>
</tr>
</tbody>
</table>

Run-time for different types of attestation (average of 10 runs).

---

Project Link:
https://ssg-research.github.io/mlsec/mlattestation
Abstract—Providers of machine-learning (ML)-based services make various claims about their models, e.g., accuracy, fairness, or the provenance and representativeness of the data used to train it. Regulators and potential clients must convince themselves that these claims are accurate. Prior works have used purely ML approaches or cryptographic primitives to prove certain properties, such as distributional properties or proof of training. However, these are often narrowly focused (lacking versatility), inefficient or inaccurate. There is a need to efficiently audit different types properties across the ML model training and inference pipeline. Trusted computing researchers introduced the notion of property attestation during ML training and inference. We identify this to the notion of property attestation to prove various properties about a local computing system to remote parties. Recent developments make it possible to run and even train models inside hardware-assisted trusted execution environments (TEEs). We propose TEE-based ML property attestation to efficiently furnish attestations for various ML properties for training and inference. It scales to multiple verifiers, and is independent of ML model configuration.

1. Introduction

Machine learning (ML) models are increasingly being used for high-stakes decision making like medical diagnosis, job screening, and loan applications. This has raised concerns about the different risks to data privacy, fairness and robustness. In response, several jurisdictions have set up regulations to ensure that the training process and model’s behaviour during inference are as expected [3]. For example showing that (a) a model meets a desirable level of accuracy or guarantees fairness, privacy or robustness, without disclosing the model, or (b) distributional properties of the dataset used to train a model, or the training configuration. The model trainer who makes various claims, such as the identity or distributional properties of the dataset used to train a model, or the training configuration. The model trainer may try to fool the attestation mechanism to make false claims. Later an untrusted ML service provider is required to attest to properties about the inference process, e.g., that the output was generated from a specific model on a specific input.

Requirements. An ideal ML attestation scheme must be:

R1 Efficient (low-overhead generation and verification, even with large models or expensive-to-compute properties).

R2 Scalable (supports large numbers of provers/verifiers).

R3 Versatile (able to prove a wide variety of claims, and new claims created with minimal effort.)

No prior attestation scheme using cryptographic primitives satisfy all the three requirements. Secure MPC-based attestation is applicable across different models, but lack scalability and efficiency due to the large number of interactions, and needing to train the model for each verification [1]. Zero-knowledge proofs (ZKPs) are scalable to larger numbers of verifiers but lack efficiency and require properties that can be adapted to the ZKP scheme [2].

3. TEE-based ML Property Attestations

TEEs establish bindings between key components: the model and its inputs/outputs, the model and its accuracy with respect to a test dataset, the model and its training dataset and configuration, and distributional properties of a dataset. These bindings are implemented in normal PyTorch-based Python code, making the approach highly versatile. Bindings are signed with the TEE’s secret attestation key, yielding attestations. As these can be generated
by anyone with the hardware, and validated by anyone, this approach is inherently scalable. These attestations enable verifiers to draw conclusions about the model and training dataset properties during training and inference. We describe the different types of attestations supported by our framework.

**Attestations made once per dataset.**
- **model provenance**: the model has been certified by a trusted organisation, linking it to a well-known identity.
- **distributional property attestation** shows that distributional properties (e.g., sex ratio) in the training dataset matches with required properties specified by the verifier. This is done once for the training dataset but has to be combined with proof of training to draw meaningful conclusions.

**Attestations made once per model.**
- **proof of training**: a model was trained on a certain dataset with a certain algorithm and configuration.
- **accuracy attestation**: achievement of some accuracy, as measured with a test dataset.
- **fairness attestation**: achievement of a fairness metric (e.g., accuracy parity, subgroup error rates).
- **robustness attestation**: achievement of some level of robustness, measured with a test dataset containing adversarial examples.

**Attestations made once per inference.**
- **input-model-output attestation** shows that a specific output was generated from the model for a given input. Combining these attestations allows the client verifying them to obtain end-to-end guarantees, e.g. “this inference is the result of applying model $M$ over input $I$, where $M$ was trained on dataset $D$ (which has distribution $F$), and $M$ has accuracy $p$ when measured using test set $T$”.

4. Preliminary Results

We implement the framework using the Intel SGX TEE, and the Gramine library to run Python programs inside it.

**Experimental Setup.** We use the CIFAR10 benchmark dataset which consists of 60000 $32 \times 32$ colour images belonging to one of ten classes. We use 50000 training images and 10000 test images. We consider a convolutional neural network three convolution layers of dimensions: [32, 64, 128] and is trained for 10 epochs.

For metrics, we consider the CPU-based baseline that performs the same computation outside of the SGX enclave, and without the attestation. The computation is divided into:
- **Startup**: time to initialize the Gramine-based SGX enclave (except the baseline) and start the Python runtime.
- **Data pre-processing**: time to read the data from disk, as well as performing any measurements or preprocessing.
- **Computation**: the time spent on the inference, training, or property measurement operation.
- **Proving**: time spent generating the attestation.

We demonstrate efficiency by showing that proving makes up only a small part of the overall execution time.

**Results.** The performance of proof of training is measured by training the convolutional model on the CIFAR10 dataset, performance shown in Table 1. We then measure its accuracy, performance shown in Table 2. Not shown are fairness and robustness attestation, computed similarly. For input-model-output attestation, we ran 10 runs of 100 inferences, with performance shown in Table 3.

5. Summary

These findings highlight the efficiency of our framework, with minimal overhead incurred during attestation generation. Notably, this initial startup cost is followed by the ability for limitless verification by any number of parties, affirming the scalability of our approach. Furthermore, TEEs exhibit versatility in training various models, exemplified by their successful implementation with convolutional neural networks. In contrast, zero-knowledge proofs, constrained by model-dependent proofs, are restricted to simpler models.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (mean ± s.d.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>36.510 ± 0.126s</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>4.35 ± 0.02s</td>
</tr>
<tr>
<td>Computation</td>
<td>514 ± 7s</td>
</tr>
<tr>
<td>Proving</td>
<td>0.0054s ± 0.00015s</td>
</tr>
</tbody>
</table>

**TABLE 2: Accuracy attestation performance.**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (mean ± s.d.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>36470 ± 171ms</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>294 ± 2ms</td>
</tr>
<tr>
<td>Computation</td>
<td>3490 ± 101ms</td>
</tr>
<tr>
<td>Proving</td>
<td>5.68ms ± 0.196ms</td>
</tr>
</tbody>
</table>

**TABLE 3: Input-model-output attestation performance.**

References