Secure Systems Group, Aalto University†, University of Waterloo+

Vasisht Duddu+, Oskari Järvinen†, Lachlan J. Gunn†, N. Asokan+†

ML Property Attestation using TEEs

- Clients of ML-based services cannot verify that responses come from the right model
- Algorithms, datasets, and training parameters cannot be verified after training
- **ML property attestation** can prove such properties efficiently and scalably

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

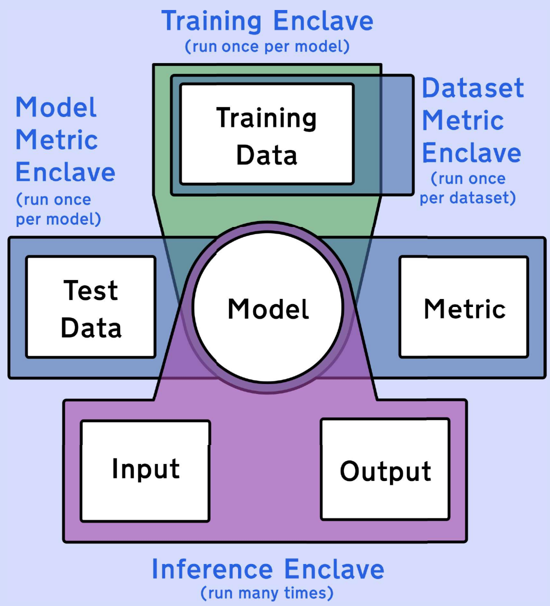
3 Our solution

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



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

	I/O Binding (100 operations)	Accuracy	Proof of Training
Startup	32029ms	36470ms	36.5s
Preprocessing	0.5ms	294ms	4.4s
Computation	70.1ms	3490ms	514s
Proving	**6.6ms**	**5.68ms**	**0.005s**

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

5 Conclusion

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

Aalto University

Project Link:
https://ssg-research.github.io/mlsec/mlattestation

UNIVERSITY OF WATERLOO

Poster: Machine Learning Property Attestations using TEEs

Vasisht Duddu*, Oskari Järvinen†, Lachlan J. Gunn†, N. Asokan*†
*University of Waterloo, †Aalto University
vasisht.duddu@uwaterloo.ca, oskari.jarvinen@aalto.fi, lachlan@gunn.ee, asokan@acm.org

***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* 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 training data complies with regulations, without disclosing the data.

ML property attestation [1] includes technical schemes by which a prover (e.g., a model trainer) can demonstrate relevant properties, during training and inference, about the model to a verifier. Current attestation schemes have various drawbacks in terms of accuracy [1], efficiency [1], [2], or scalability to multiple verifiers [1]. Further, prior works are limited to specific attestations such as proof of training (that a model was trained on some dataset) [2] or distributional property attestations (showing the training dataset of a model satisfies distributional properties) [1]. There is a need to efficiently attest different aspects about ML training and inference. We identify **efficiency**, **scalability**, and **versatility** as important criteria for attestation schemes.

Trusted Execution Environments (TEEs) already have the notion of hardware-assisted *remote attestation* to prove local system or software configuration to a remote verifier. The trusted computing research community has extended this to the notion of *property attestation* [6], and recent developments by hardware vendors [4], [5], make it possible to run and train ML models efficiently inside TEEs. Hence, we identify TEEs as the choice for realizing efficient, scalable and versatile ML property attestation. We are the first to propose a software framework for *TEE based ML property attestation* which satisfies this need.

More about this project is available at https://ssg-research.github.io/mlsec/mlattestation.

2. Problem Statement

Our goal is to design a scheme which can furnish property attestation during ML training and inference.

Setting. A model is trained by an untrusted *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 [1]. 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 32×32 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.

TABLE 1: Time to produce a proof of training (mean ± s.d.).

Operation	**Time** (mean ± s.d.)
Startup	36.510 ± 0.320s
Preprocessing	4.35 ± 0.02s
Computation	514 ± 7s
Proving	0.0054s ± 0.00015s

TABLE 2: Accuracy attestation performance.

Operation	**Time**
Startup	36470 ± 171ms
Preprocessing	294 ± 2ms
Computation	3490 ± 101ms
Proving	5.68ms ± 0.196ms

TABLE 3: Input-model-output attestation performance.

Operation	**Time** (mean ± s.d.)
Startup	32029 ± 187ms
Preprocessing	0.512 ± 1.71ms
Computation	70.1 ± 40.6ms
Proving	6.58 ± 1.10ms

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.

References

[1] V. Duddu *et al.*, "Attesting distributional properties of training data for machine learning," in *ESORICS*, 2024, arXiv:2308.09552.

[2] S. Garg *et al.*, "Experimenting with zero-knowledge proofs of training," in *CCS*, 2023. [Online]. Available: https://eprint.iacr.org/2023/1345

[3] High-Level Expert Group on AI, "Ethics guidelines for trustworthy AI," European Commission, Brussels, Report, Apr. 2019.

[4] Intel, "Accelerate AI workloads with Intel Advanced Matrix Extensions (Intel AMX)," Solution Brief, 2024.

[5] Nvidia, "NVIDIA H100 tensor core GPU," Datasheet, 2024.

[6] A.-R. Sadeghi and C. Stüble, "Property-based attestation for computing platforms: caring about properties, not mechanisms," in *NSPW*, 2004.