UNIVERSITY OF WATERLOO

KTH VETENSKAP OCH KONST

Aalto University

Towards Verifiable Properties of AI systems

via Hardware-Assisted Attestations

N. Asokan

https://asokan.org/asokan/

@asokan.org 𝕏 @nasokan

(Joint work with Vasisht Duddu, Oskari Järvinen, Lachlan J. Gunn, Prach Chantasantitam, Adam Caulfield)

AI Regulations are Emerging

Various jurisdictions have announced forthcoming AI regulations[1,2,3]

- E.g., Requirements governing distribution of (demographic) attributes in training datasets

Executive Order 14110 [4]
*… incorporation of equity principles in AI-enabled technologies used in the health and human services sector, **using disaggregated data on affected populations and representative population data sets** when developing new models, monitoring algorithmic performance against discrimination and bias in existing models, and helping to identify and mitigate discrimination and bias in current systems;*

AI Act [5]
*The data sets should also have the appropriate **statistical properties, including as regards the persons or groups of persons in relation to whom the high-risk AI system is intended to be used**, with specific attention to the mitigation of possible biases in the data sets, that are likely to affect the health and safety of persons, have a negative impact on fundamental rights or lead to discrimination prohibited under Union law, especially where data outputs influence inputs for future operations*

Designed to ensure that AI models have desirable properties

- Representativeness, fairness, privacy, robustness, transparency, etc.

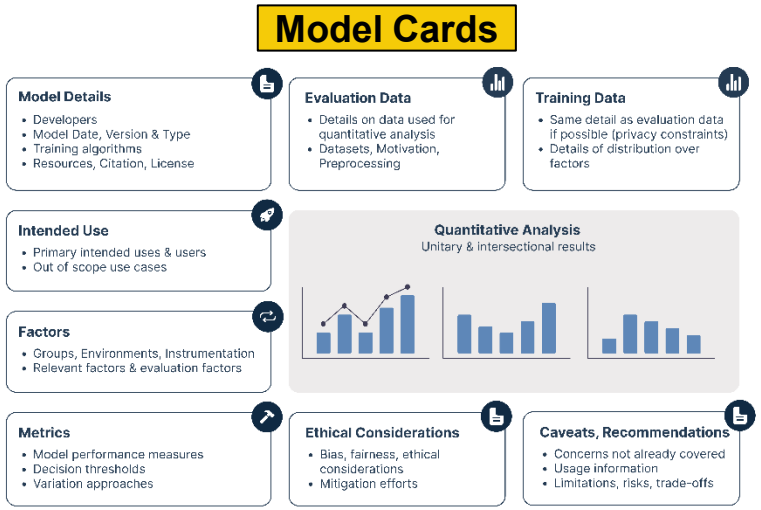
[1] European Commission, *General-Purpose AI Code of Practice*, 2025
[2] United Kingdom Parliament, *Artificial Intelligence (Regulation) Bill*, 2025
[3] Brazilian Senate, *Brazil AI Act*, 2024
[4] Executive Office of the US President, *Executive Order 14110: Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence*, 2023
[5] European Parliament, *AI Act: Recital 67*, 2021

Mechanisms to Advertise Model Properties Exist

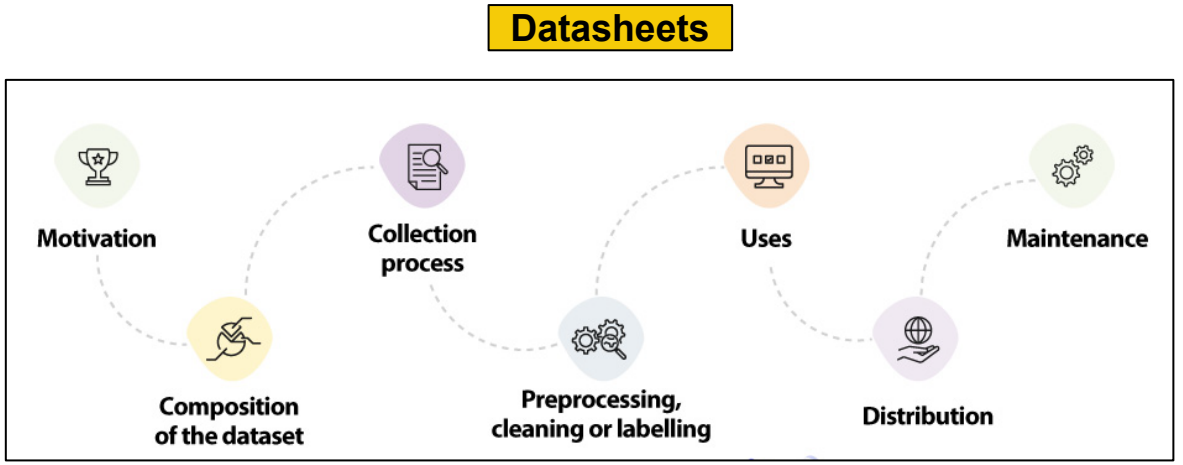
AI model providers use "nutrition labels" to advertise model properties

Model cards (for model properties)[1], datasheets (for datasets)[2,3]

- Adapted by Google, Huggingface and others



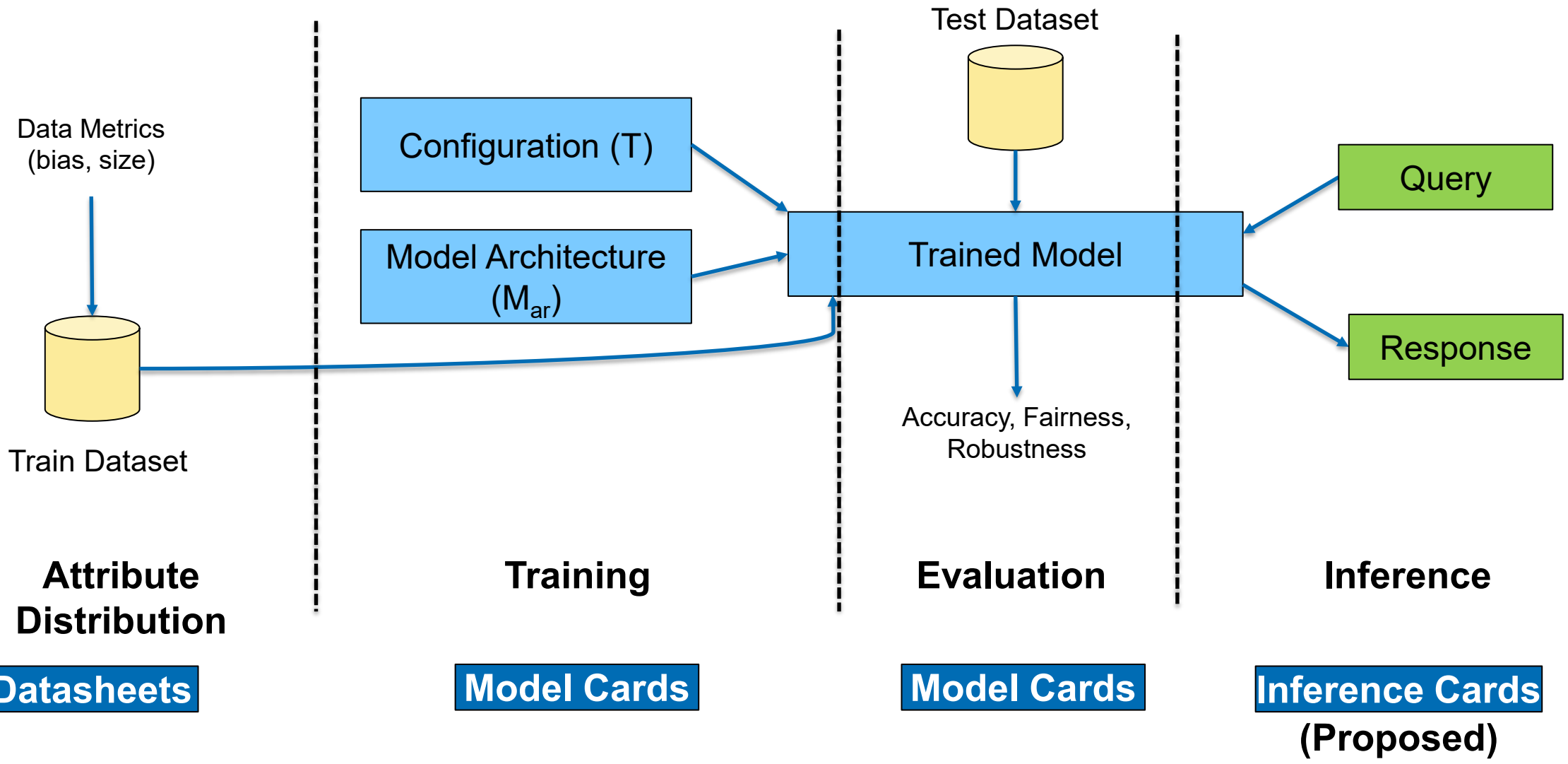
Source: https://www.trail-ml.com/blog/ml-model-cards



Source: https://datos.gob.es/en/blog/data-documentation-datasheets-datasets

[1] Mitchell et al. *Model Cards for Model Reporting*, FAccT 2019
[2] Gebru et al. *Datasheets for datasets*, Communications of ACM 2021
[3] Pushkarna et al. *Data Cards: Purposeful and Transparent Dataset Documentation for Responsible AI*, FAccT 2022

Types of AI Property Cards



The Need for Verifiable Properties of AI Systems

How to verify compliance with regulation/policy/standard?

Traditional approaches (like verification by an authority) may not work for AI systems

- **Release of some information may be subject to other regulation**
  - E.g., health-related sensitive data

- **Third parties may need to check compliance before official verification**
  - Fast-moving ecosystem

Need a way to attest to claimed properties without leaking any sensitive data

Existing Property Attestation Mechanisms

Machine Learning (ML)-based Attestations

Error-prone and not robust: e,g.,

- proof of learning[1,2]
- re-purposing distribution inference for attesting attribute distribution properties[3]

Cryptographic Attestations (e.g., Zero-knowledge Proofs, Multi-party Computation)

Inefficient: e,g.,

- ~13 minutes for inference (I/O) attestation (e.g., using ZKPs with LLMs[4])
- ~15 minutes per iteration of gradient descent for proof of training[5]
- Sometimes need to retrain model each time[3]

Not Versatile: Limited to crypto-friendly properties

[1] Zhang et al. *"Adversarial Examples" for Proof- of-Learning*, IEEE S&P 2022
[2] Fang et al. *Proof of Learning is more Broken than You Think*, IEEE EuroS&P 2023
[3] Duddu et al. *Attesting Distributional Properties of Machine Learning Training Data*, ESORICS 2024
[4] Sun et al. *zkLLMs: Zero Knowledge Proofs for Large Language Models*, ACM CCS 2024
[5] Abbaszadeh et al. *Zero-Knowledge Proofs of Training for Deep Neural Networks*, ACM CCS 2024

Desiderata for ML Property Attestation

R1 Efficient

Incur low computation overhead

R2 Versatile

Support various ML properties for training and inference

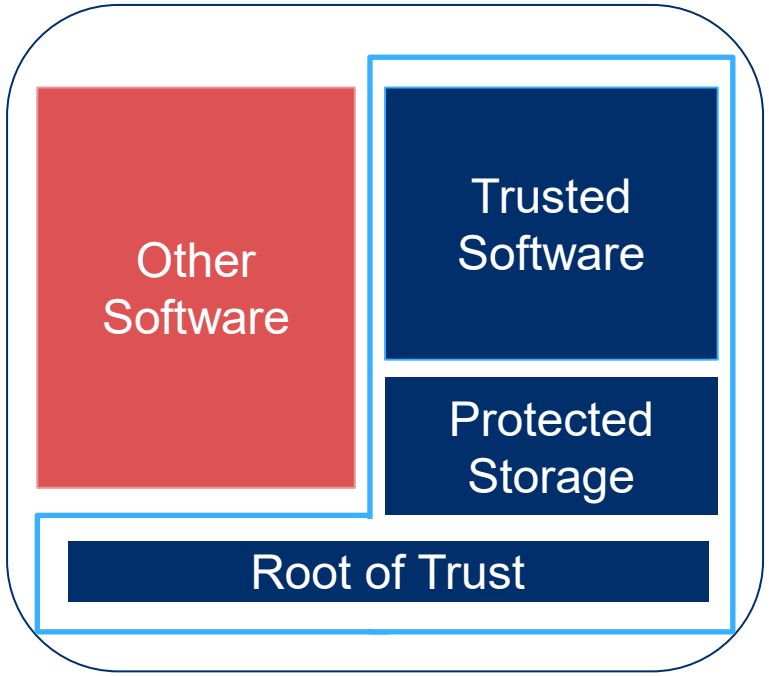
R3 Scalable

Support multiple verifiers

R4 Robust

Resist evasion of attestations by malicious prover

Hardware-assisted TEEs are Pervasive



Hardware support for

- **Isolated execution: Isolated Execution Environment**
- **Protected storage: Sealing**
- **Ability to convince remote verifiers: (Remote) Attestation**

Trusted Execution Environments (TEEs)
Operating in parallel with "rich execution environments" (REEs)

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

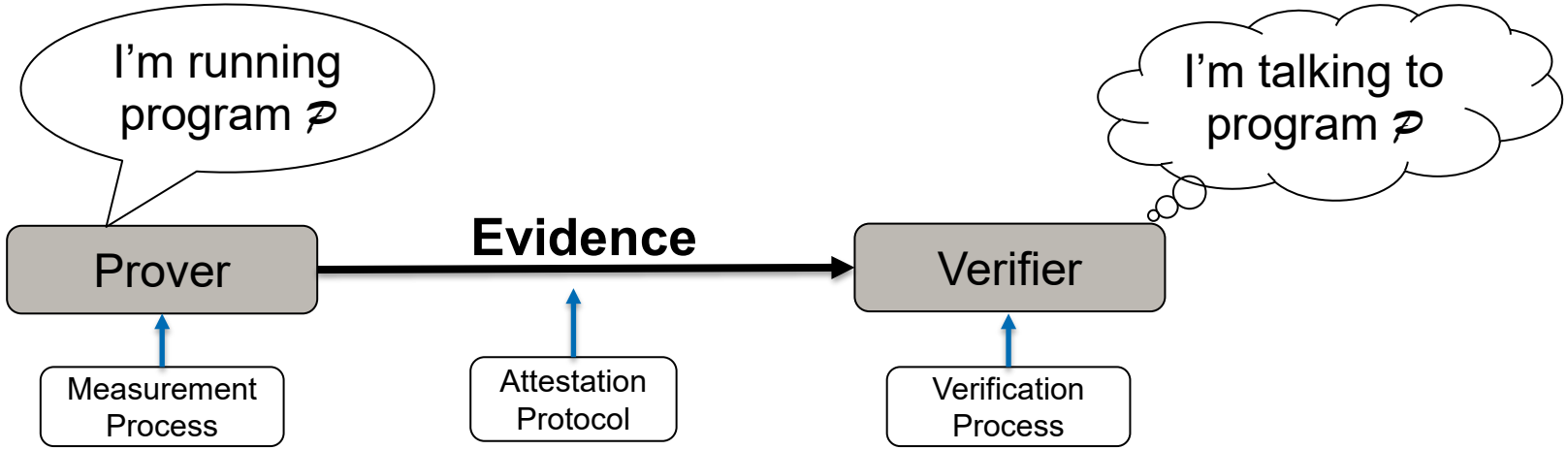
Intel Software Guard Extensions (SGX) and Trust Domain eXtensions (TDX)

https://www.intel.com/…/securing-your-trust-boundary-with-intel-sgx-and-intel-tdx.html

[1] Asokan et al. *Mobile Trusted Computing*, Proceedings of the IEEE, 102(8) 2014
[2] Ekberg et al. *Untapped potential of trusted execution environments*, IEEE S&P Magazine, 12:04 2014

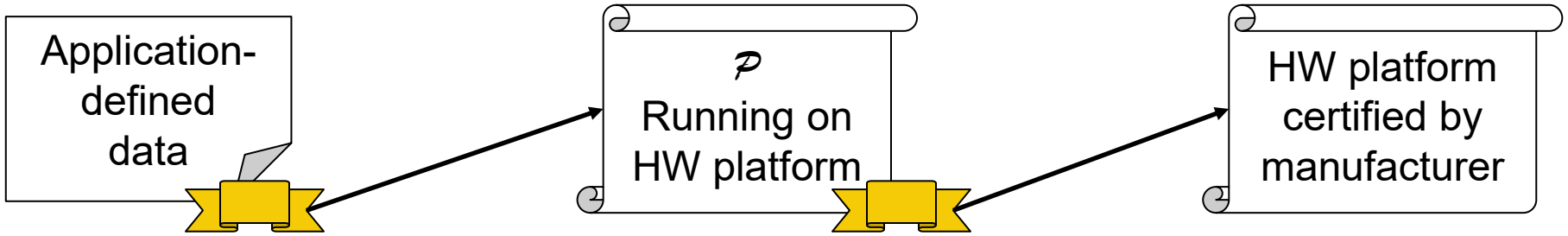
What is Remote Attestation?

Verifier ascertains current state and/or behavior of Prover



A practical mental model for SGX and TDX attestation:

Certificate showing that something came from software with a certain hash



Can TEEs Enable ML Property Attestation?

Recent developments make ML training/inference within TEEs feasible (efficient)

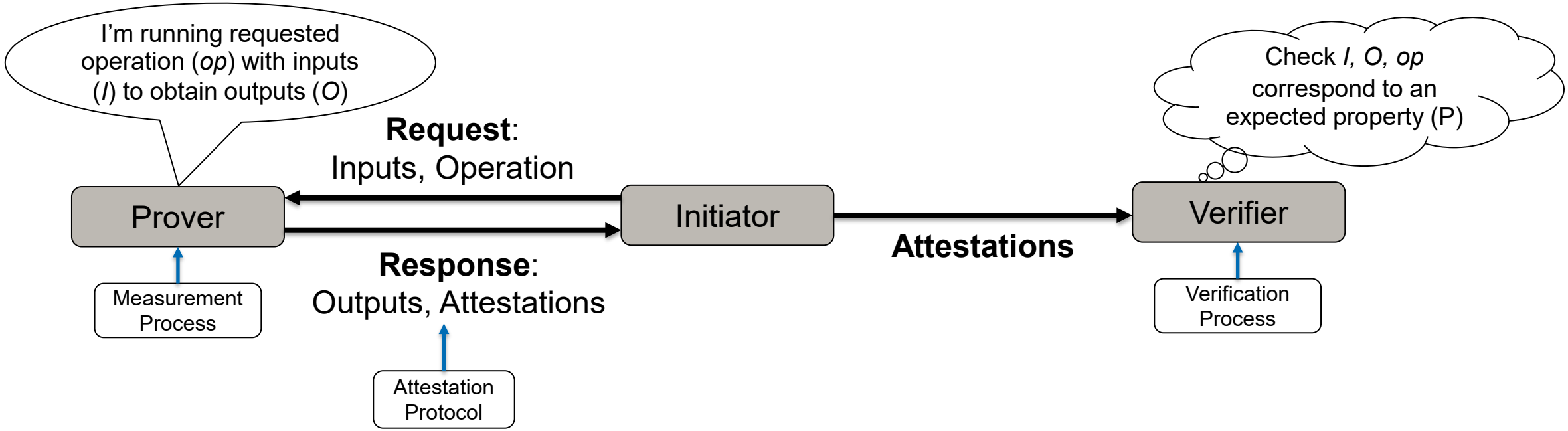
- Intel's AMX extensions[1], NVIDIA's H100 GPU[2]
- Available in cloud computing platforms

Property Attestation for TEEs

- Remote attestation was extended to properties of binaries running inside TEEs[3]
- Can we adapt this for attesting ML properties?

[1] Google Cloud Team, *We tested Intel's AMX CPU accelerator for AI and here's what we learned*, 2024
[2] Zhu et al. *Confidential Computing on NVIDIA's H100 GPU: A Performance Benchmark Study*, 2024
[3] Sadeghi and Stuble, *Property-based attestation for computing platforms: caring about properties, not mechanisms*, ACM NSPW'04

Enabling non-interactive property attestation



Initiator ←→ Prover

- Initiator specifies operation type and inputs (challenge, datasets/models, configs)
- Prover provides outputs and attestations

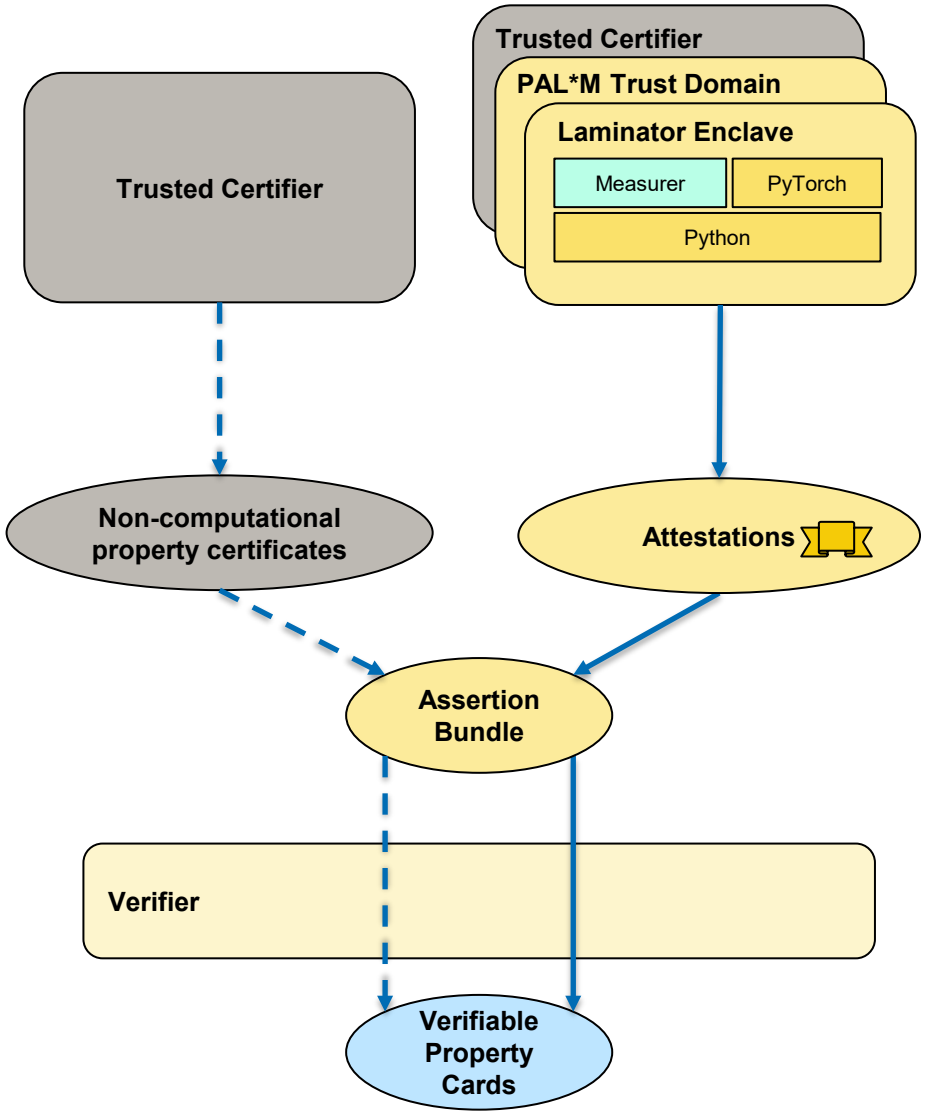
Initiator ←→ Verifier

- Non-interactive with respect to operation
- Initiator provides evidence to Verifier
- Verifier performs verification process

Our Frameworks

Measurer within TEE measures desired property

TEE produces attestation (property card fragment)



Property attestations

- Laminator[1]: SGX-based for classifiers
- PAL*M[2]: TDX-based for large generative models

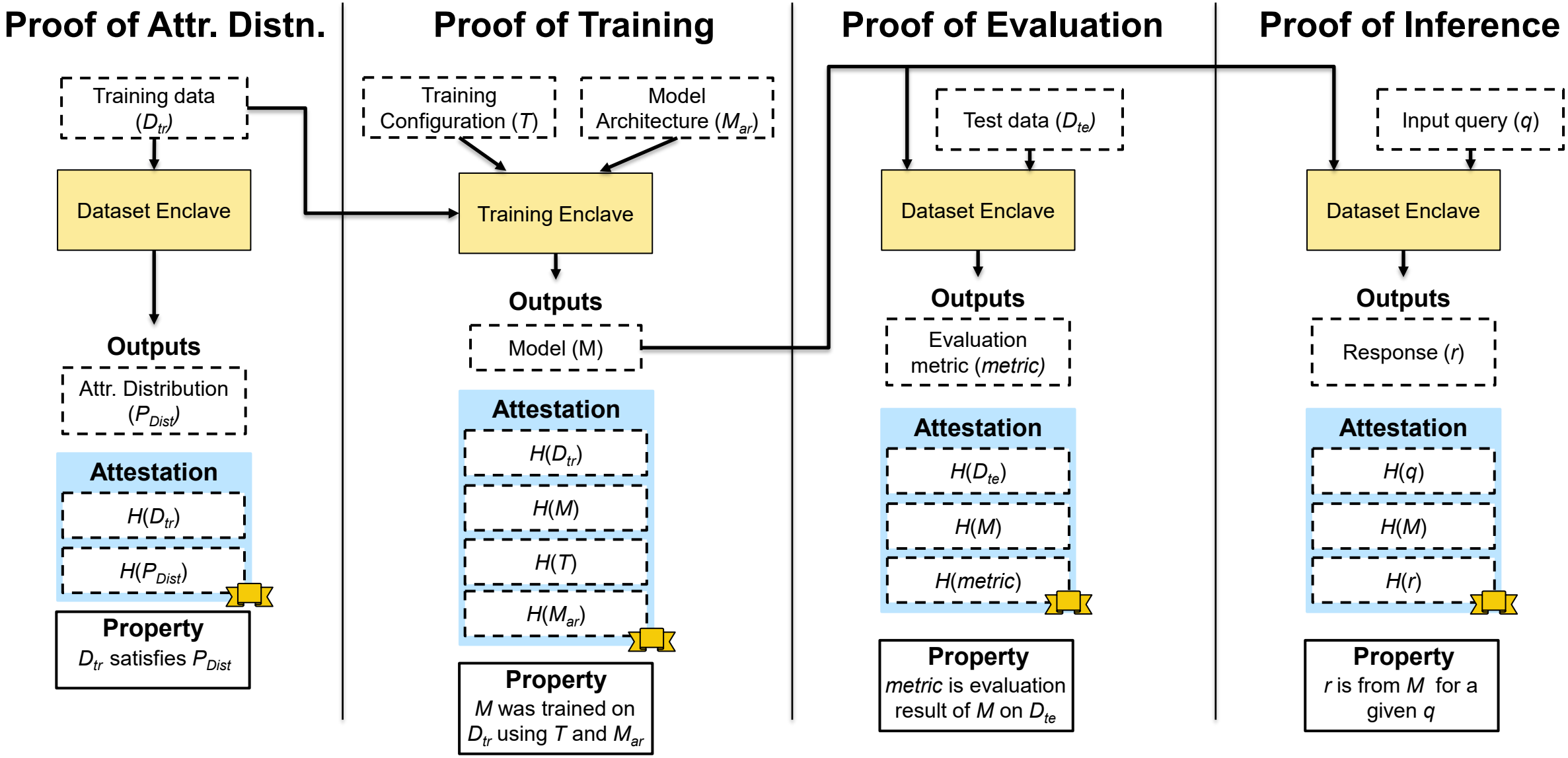
Assertion bundle

- combines certificates and attestations from various sources
- checkable by Verifier to realize verifiable property cards

[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025

[2] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

ML Property Attestations in Laminator[1]



[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025

Beyond Laminator[1]…

Laminator[1] is limited due to use of SGX

- Runs small models (classifiers), cannot efficiently support generative models
- Large generative models require GPU for realistic performance

This motivates PAL*M[2] for verifiable **P**roperty **A**ttestations of **L**arge *G*enerative **M**odels

- Runs natively on Intel TDX
- Uses GPU (NVIDIA H100 CC) instead of CPU-only implementation
- Support generative models (e.g., LLMs)

[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025
[2] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

Challenges in PAL*M[1]

How to handle CPU-GPU operations accurately & efficiently

- Operations use GPU for practical performance
- Must prove GPU in use is trusted
- Must verify (1) GPU is trusted and (2) CPU used trusted GPU

How to account for large datasets

- Standard ML frameworks may use memory-mapping
- Thus, dataset resides outside trust boundary
- Training may involve randomly sampling from dataset

How to define properties relevant to generative models

- Account for common practices like fine-tuning, inference “sessions”

[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

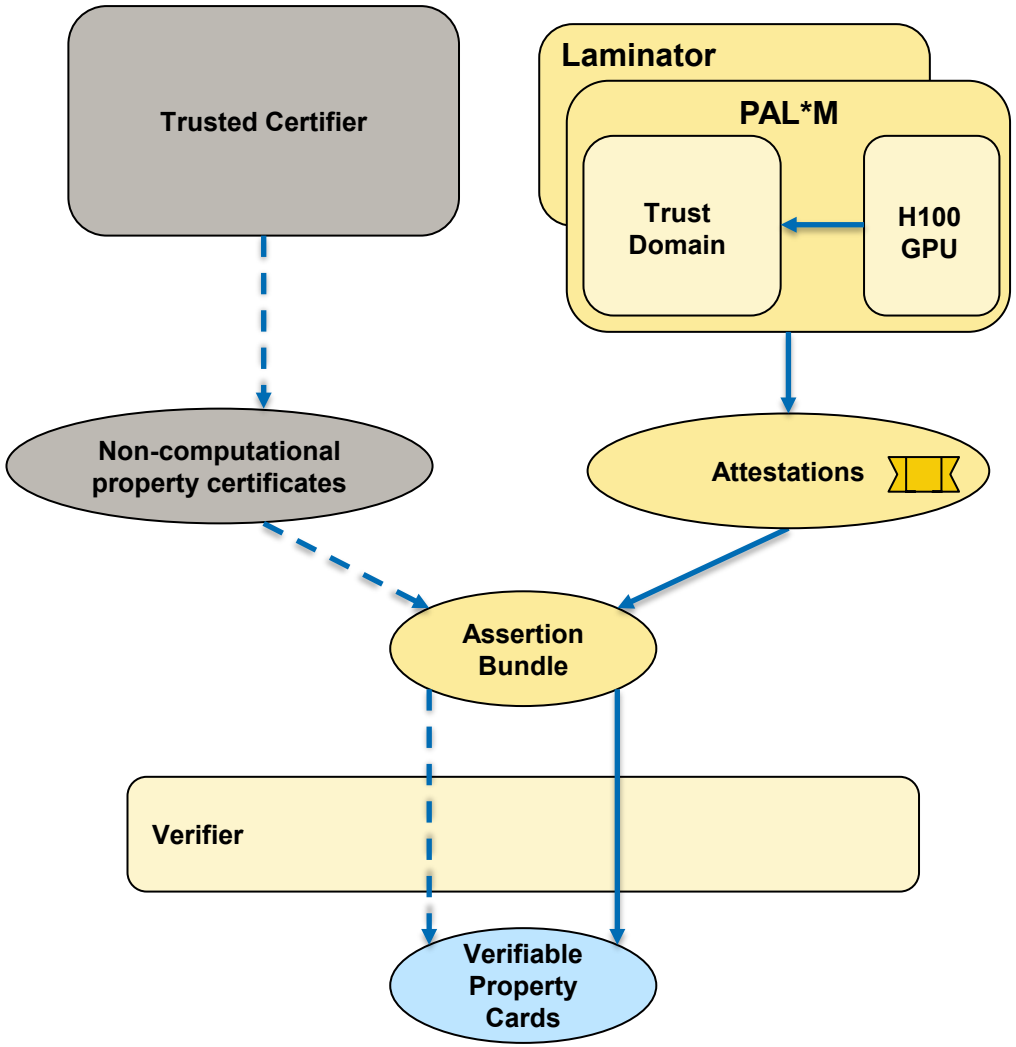
Challenge: CPU-GPU settings

PAL*M extends trust boundary to GPU

Leverages GPU TEE: NVIDIA[1] H100

- attests its own configuration GPU_{Att}

Property measurement includes GPU_{Att}

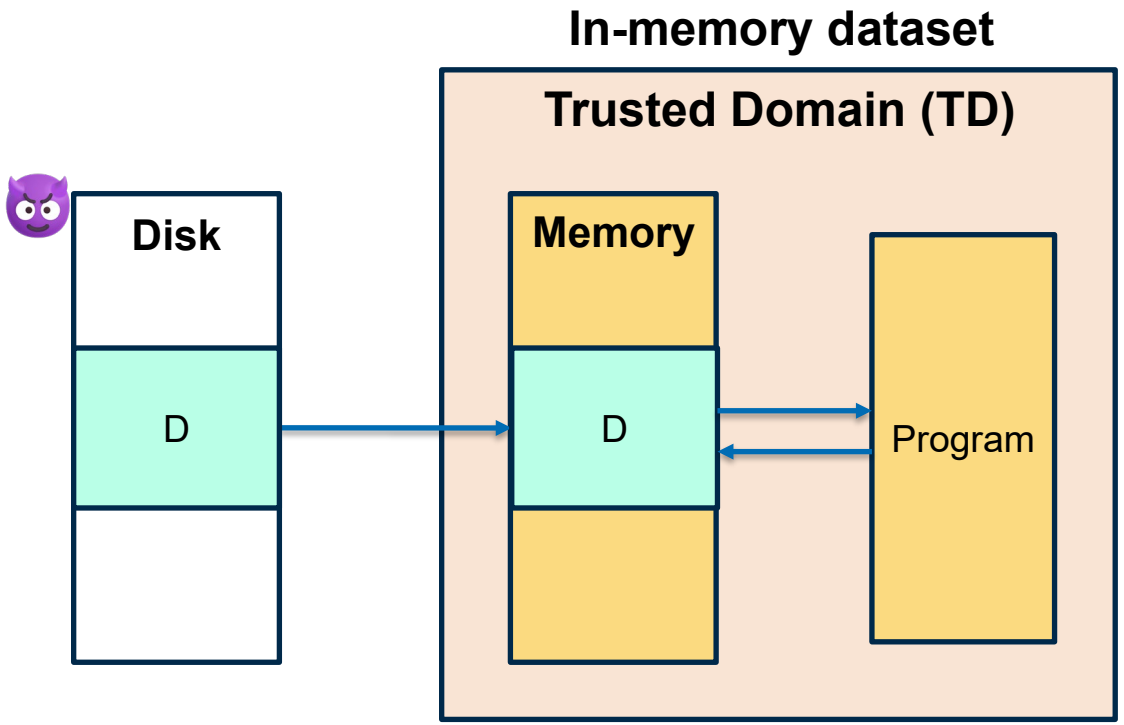


[1] NVIDIA Cloud & Data Center, *NVIDIA H100*, Accessed: 2025

Challenge: Handling large datasets

Case 1: In-memory dataset

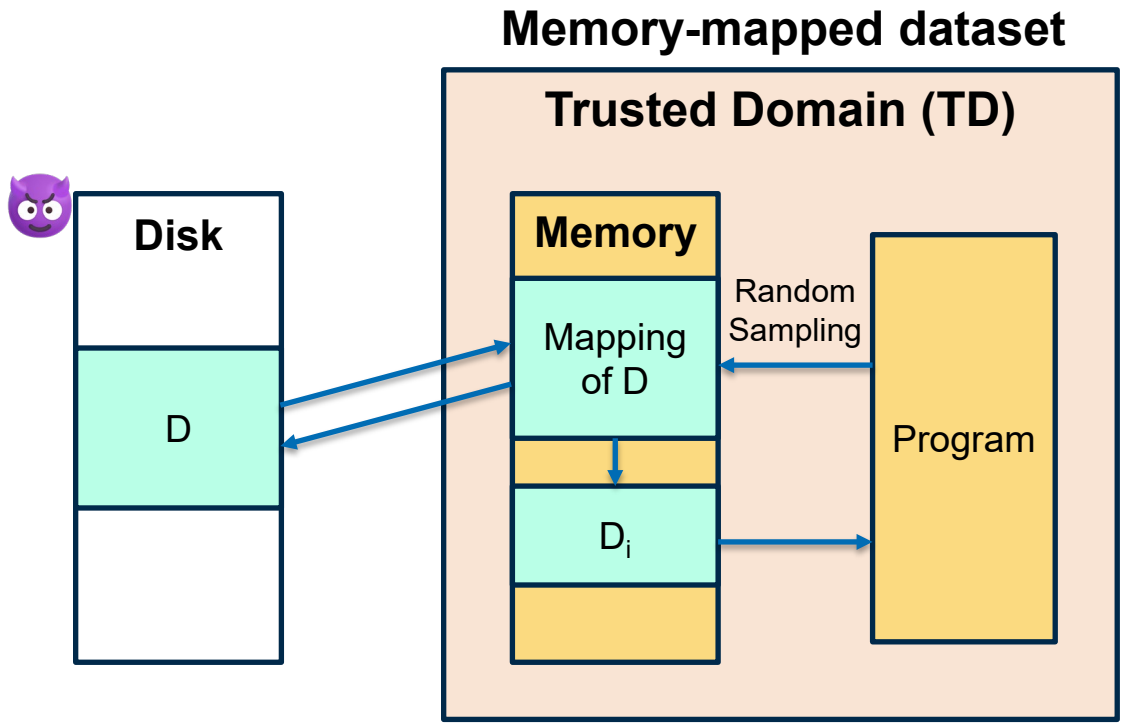
- Less commonly used
- D stays in TD memory
- High memory costs
- Integrity measurement is straightforward:
  - Load, measure, use



Challenge: Handling large datasets

Case 1: In-memory dataset
- Less commonly used
- D stays in TD memory
- High memory costs
- Integrity measurement is straightforward:
  - Load, measure, use

Case 2: Memory-mapped
- Commonly used
- Memory map in TD memory, D stays on disk
- Low memory costs
- Integrity measurement not straightforward
  - Vulnerable to time-of-check to time-of-use attacks
  - Training may use a random sequence of training dataset



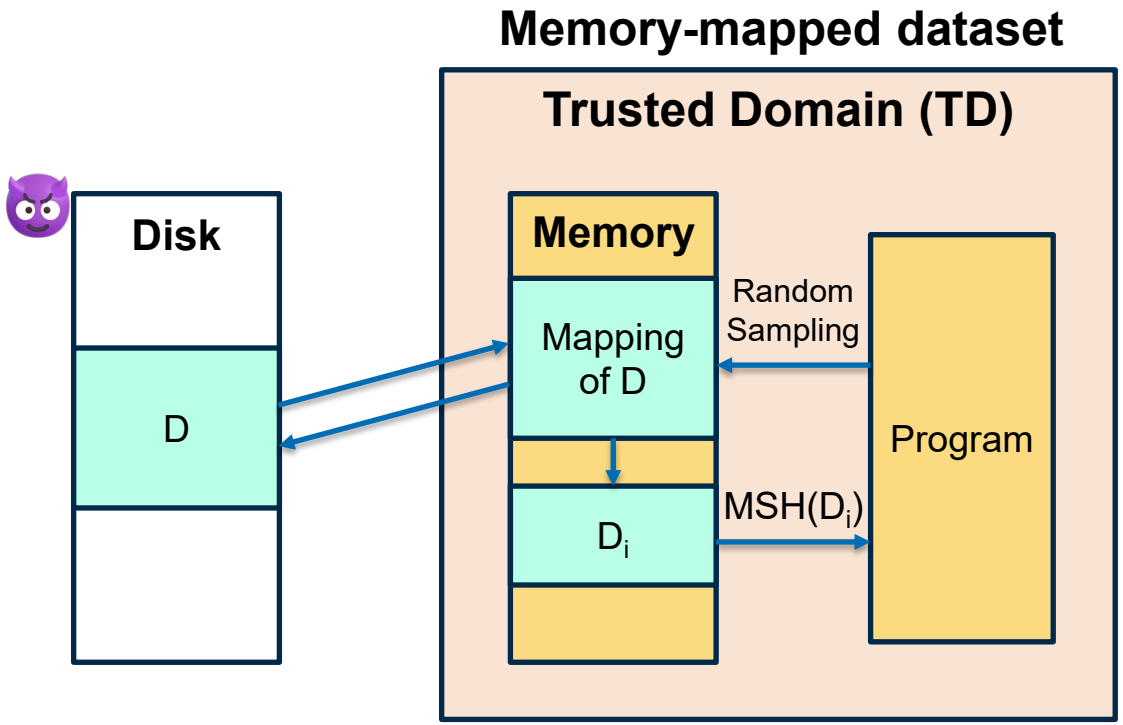
Challenge: Handling large datasets

Memory-mapped

- Use increment multiset hash (MSH)[1]
- Produces a unique hash for a set
- Incremental: adds one record at a time
- Produces unique hash regardless of order

Result:

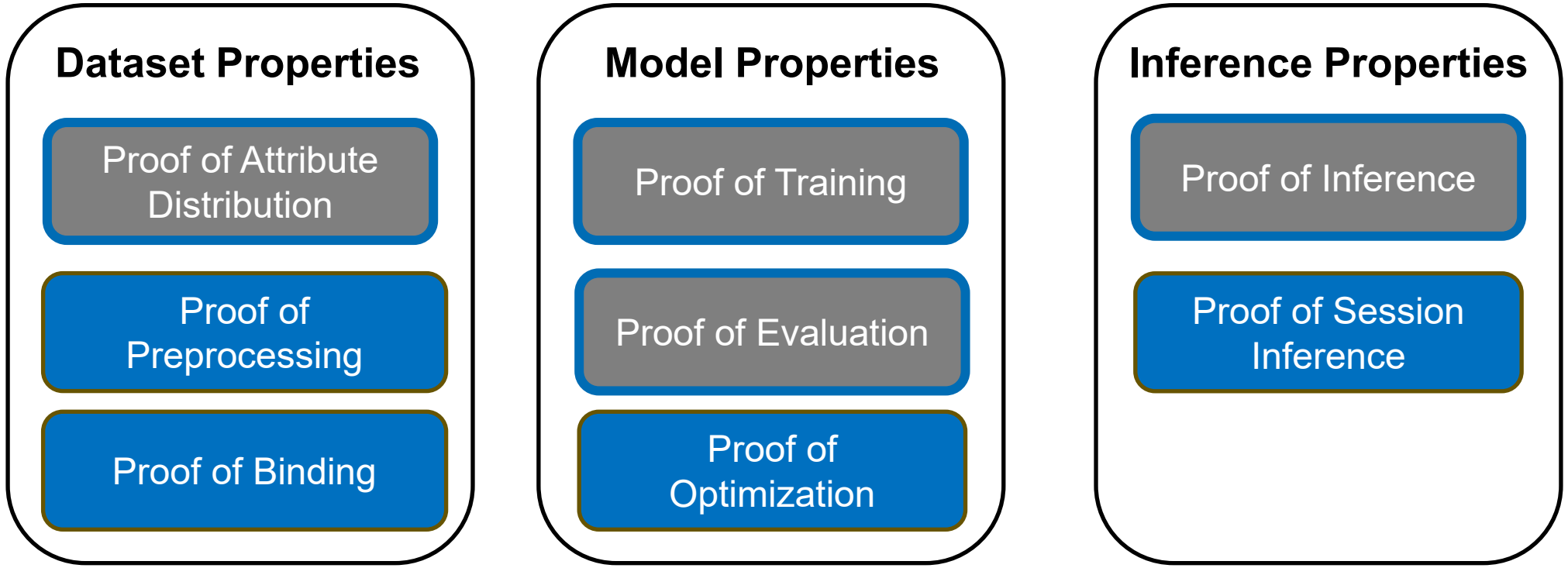
- Final MSH represents entire dataset
- Dataset remains in external storage
- Tampering detectable
- Added performance cost vs. typical hash
- For generative models, this cost is:
  - incurred once and
  - minimal for multi-time operation



[1] Clarke, Dwaine, et al. *Incremental multiset hash functions and their application to memory integrity checking*, International conference on the theory and application of cryptology and information security, 2003

Challenge: Defining Generative AI Properties

Starting with verifiable ML properties from Laminator[1]…



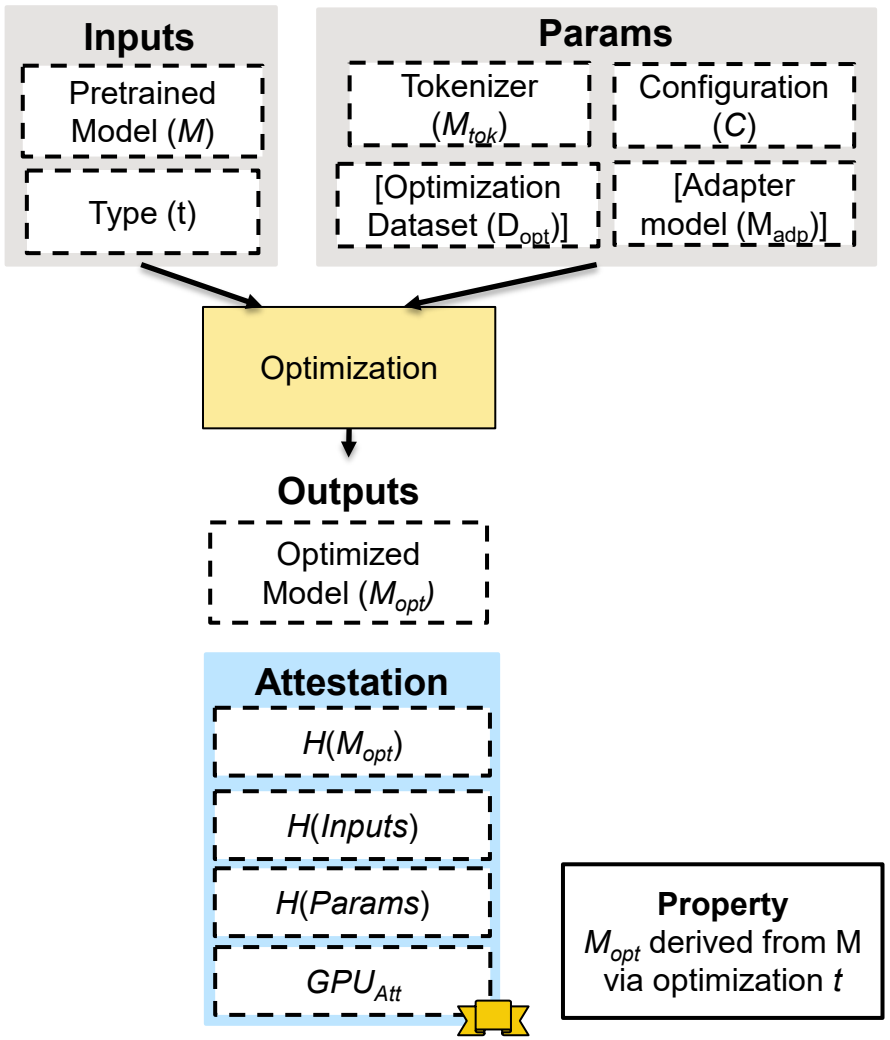
PAL*M[2] introduces methods to handle characteristics of generative models

- Unique operations of generative models
- Operations along generative-model pipeline
- GPU use for any operation

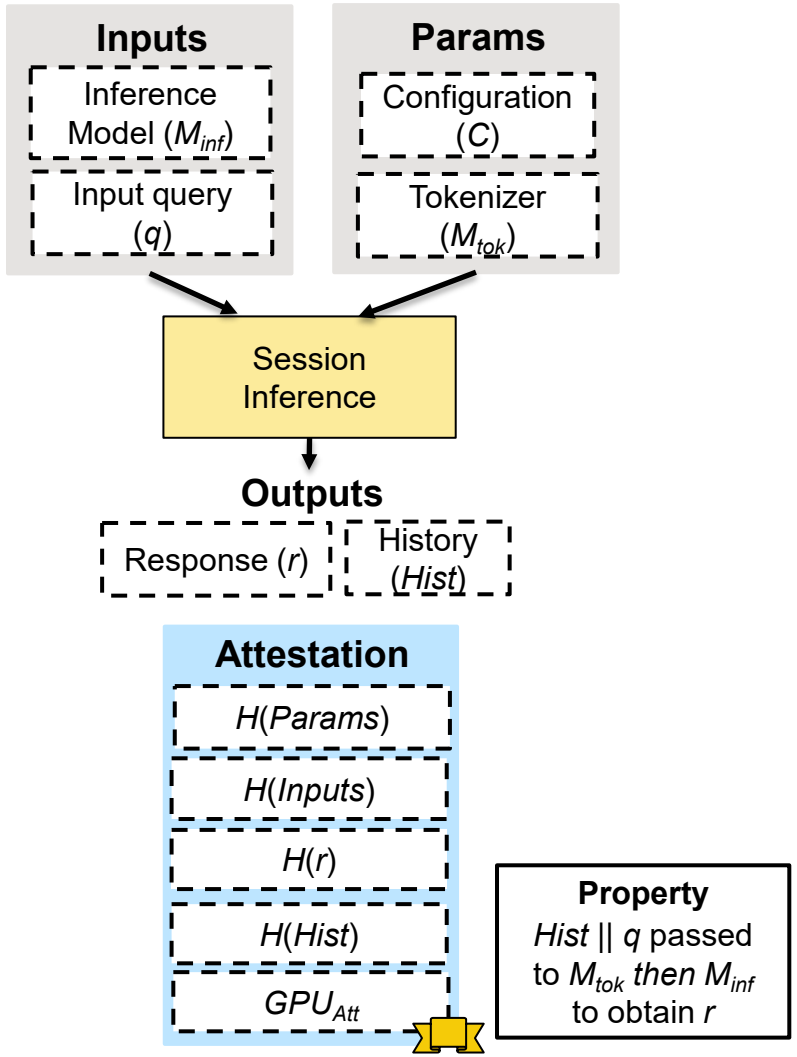
[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025
[2] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

New Properties in PAL*M[1]

Proof of Optimization

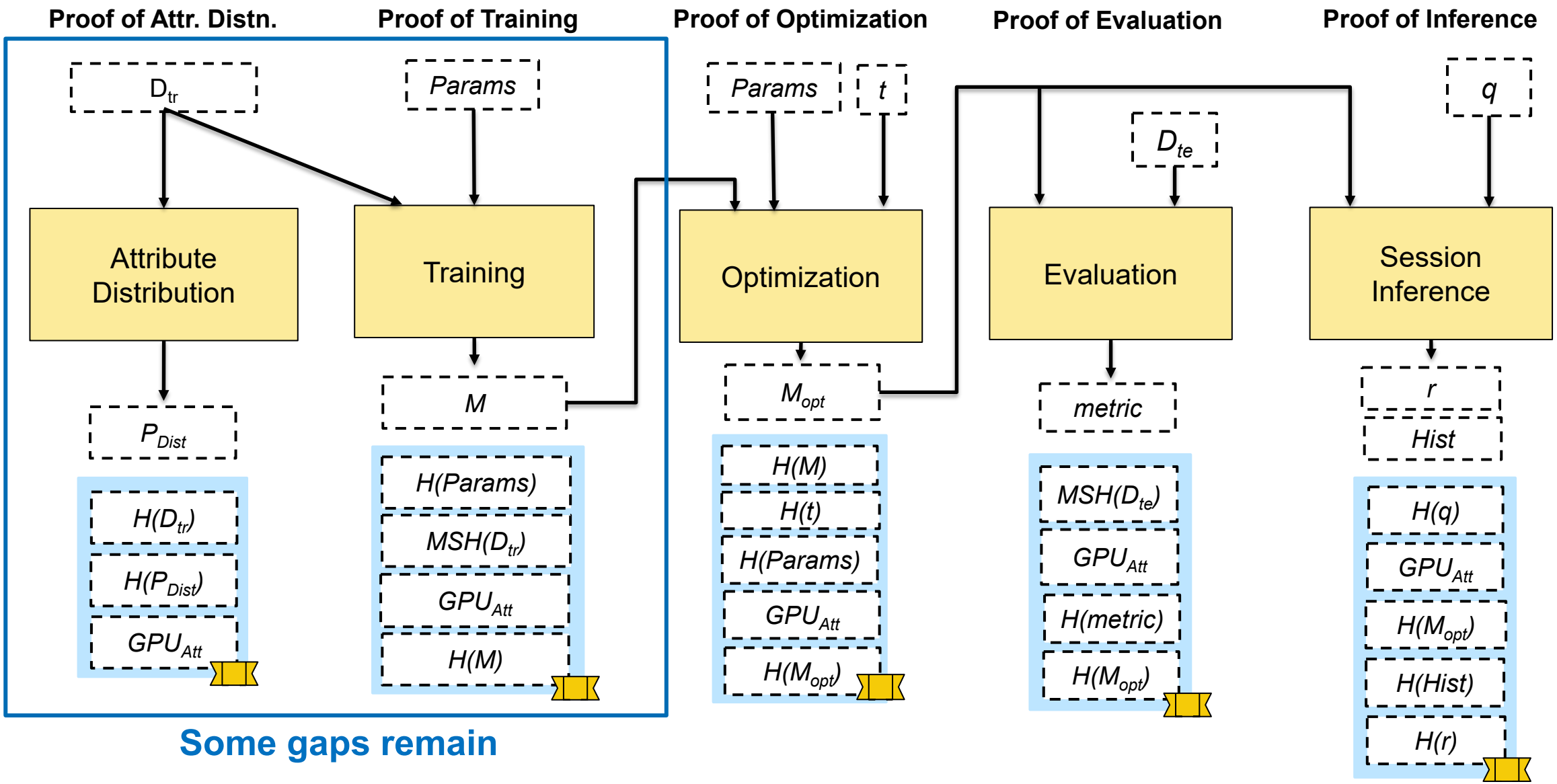


Proof of Session Inference



[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

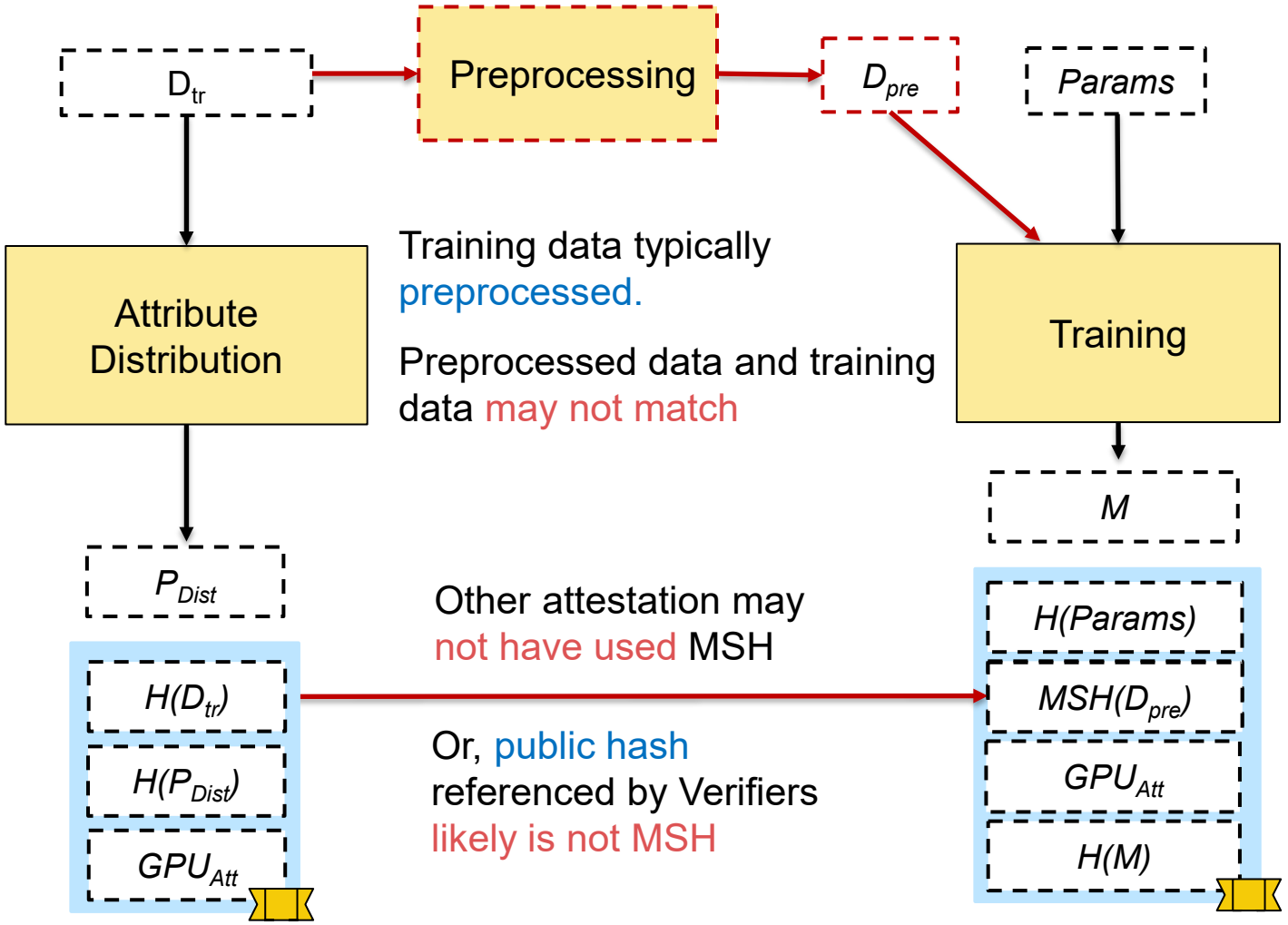
New Properties in PAL*M [1] : The Big Picture



Some gaps remain

[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

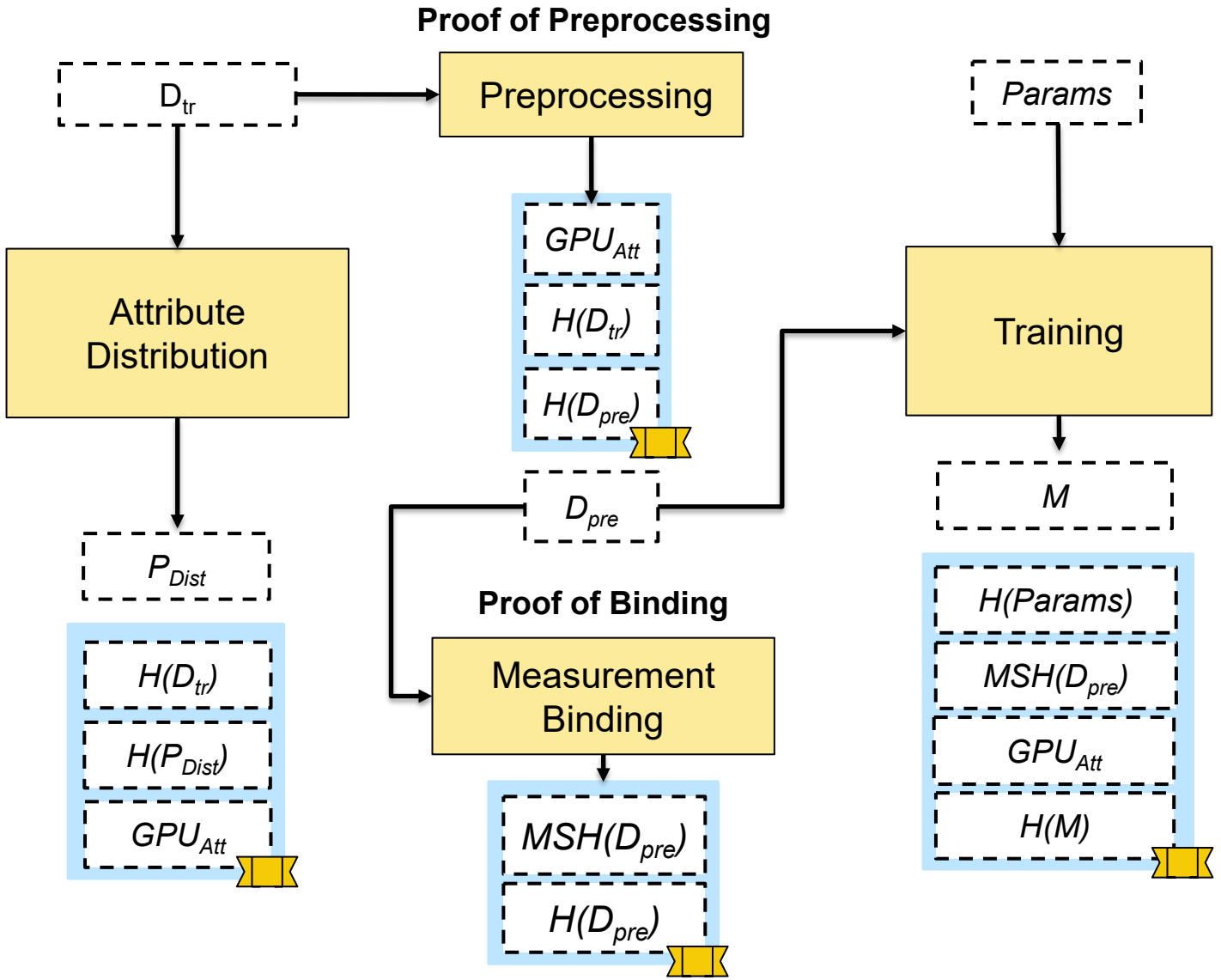
New Properties in PAL*M [1] : Property Disconnects



[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

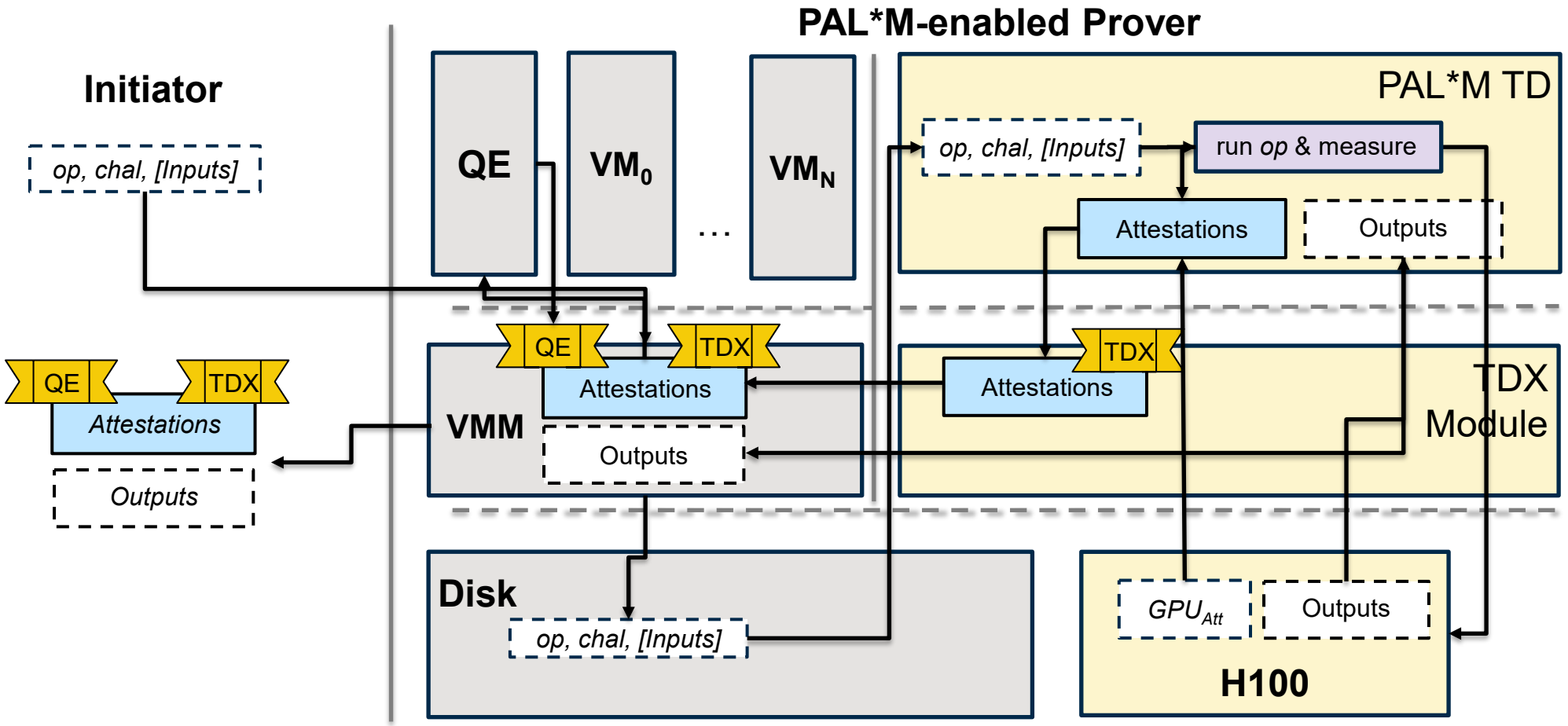
Addressing Property Disconnects

We introduce preprocessing and binding property attestation to enabling chaining



PAL*M[1] End-to-End Property Attestation

Property measurements are used with hardware-assistance in Intel TDX for attestation



[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

Experimental Setup

Datasets, models, and system configuration used for Laminator[1] & PAL*M[2]

Component	Laminator[1]	PAL*M[2]
Dataset	CENSUS (tabular) UTKFACE (images) IMDB (text)	BookCorpus Yahma/alpaca-cleaned MMLU & WMT14 CoQA
Models	MLP [128], MLP [128, 256, 512, 256] VGG11, VGG16 LSTM [64, 256, 256], LSTM [64, 256, 256, 256, 256]	Llama-3.1-8B Gemma-3-4B Phi-4-Mini
System Setup	Intel SGX Gramine	Intel TDX Ubuntu NVIDIA H100 CC

Evaluation Metric: measure additional run-time for each property attestation types

[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025
[2] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

Evaluation: Dataset & Model Properties in Laminator[1]

Input, output: measurement roughly scales with size

Attestation constant across all datasets and models

Overall, overhead is low

- Proof of Attribute Distribution: 0.36% to 2.05%
- Proof of Training: <0.01% to 0.32%
- Proof of Evaluation: <0.01% to 0.35%

[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025

Evaluation: Dataset Properties in PAL*M[1]

Proof of Attribute Distribution:

- Memory-mapped: 67.95%
- In-memory: 0.015%

Proof of Preprocessing:

- Memory-mapped: 62.55%
- In-memory: 0.06%

Proof of Binding: 69.55%

Takeaway

Expensive for memory-mapped datasets

But performed only once per unique dataset

[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

Evaluation: Model Properties in PAL*M[1]

Proof of Training

- Memory-mapped: 5.66%
- In-memory: 0.01%

Proof of Optimization (Fine-tuning)

- Memory-mapped: 0.72% to 1.35%
- In-memory: 0.09% to 0.18%

Proof of Optimization (Quantization): 4.7%

Proof of Evaluation (MMLU and BLEU score)

- Memory-mapped: 0.17% to 10.11%
- In-memory: 0.12% to 2.60%

Takeaway

Low cost to attest properties of multi-time model operations

[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

Evaluation: Efficiency for Inference Properties

In Laminator[1], baseline cost for single inference low compared to attestation

- High overhead between 39% and 3955% (aka “overhead w/ att”)

Amortizing overhead over several proofs of inference

- Generate signing keypair during initialization and attest once
- Sign each inference result for indirect, low-cost attestation (“overhead w/ sgn”)
  - Overhead between 0.17% and 1.17%

In PAL*M[2], this pattern is also observed for inference and session inference:

- Proof of Inference: 43.28% to 64.34%
- Proof of Session Inference: 3.57% to 11.03%

[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025
[2] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

Evaluation: Scalability, Versatility, Robustness

Scalable

- Measurements signed using TEE's attestation key
- Multiple verifiers can independently validate the attestations

Versatile

- Can attest any ML property that can be specified in python measurer script
- Allows external certificates and ZKP certificates

Robust

- Inherited from TEE integrity guarantees

Evaluation: Limitations

Side channel attacks

- Architectural extensions enable countermeasures[1,2]

Deployed on single CPU and GPU

- Cannot take advantage of distributed training

Execution integrity

- Guarantees are uncertain with run-time attacks

[1] ElAtali et al. *BliMe: Verifiably Secure Outsourced Computation with Hardware-Enforced Taint Tracking*, NDSS 2024
[2] ElAtali et al. *BLACKOUT: Data-Oblivious Computation wth Blinded Capabilities*, CCS 2025

Looking forward: Verifiable ML Ecosystems

Prior work[1] has proposed ecosystem graphs

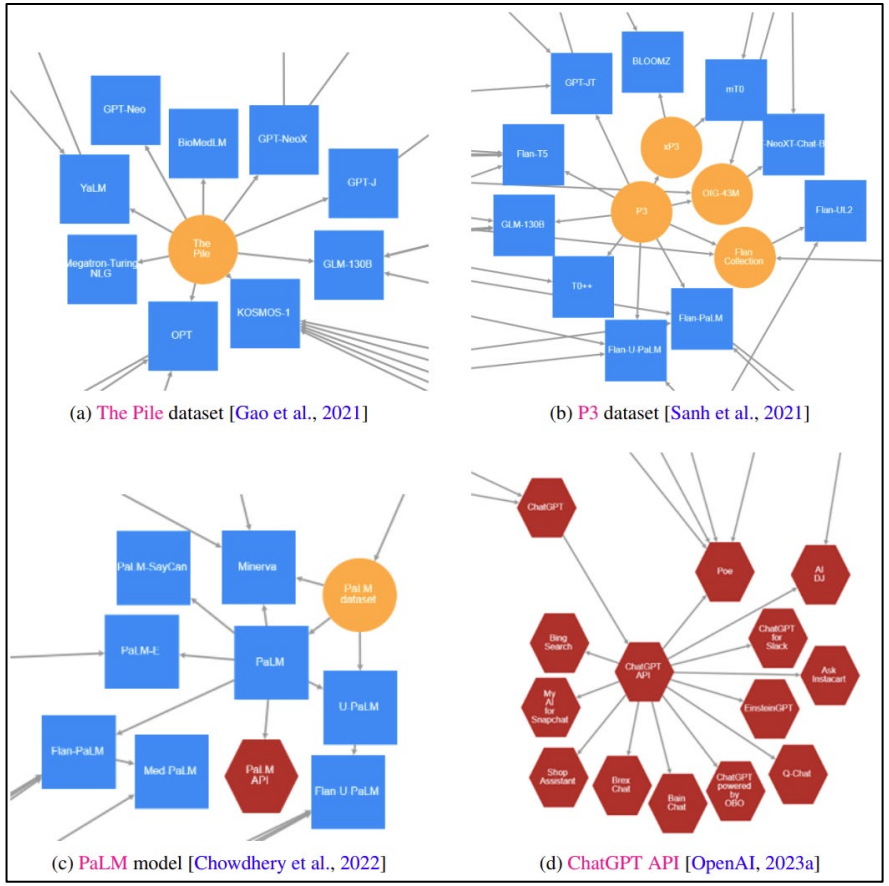
- Track relationships between models, datasets, services
- No verification of submissions
- No accountability for updating the graph

False information could be added into the graph

- By contributors: for competitive advantage
- By graph maintainers: to favor a certain organization

Architectural support for verifiable ecosystems

- Graph operations → attestable asset dependencies:
  - Model-to-model, model-sources-output, model-model-output
- Enable verifiable maintenance of ecosystem graphs

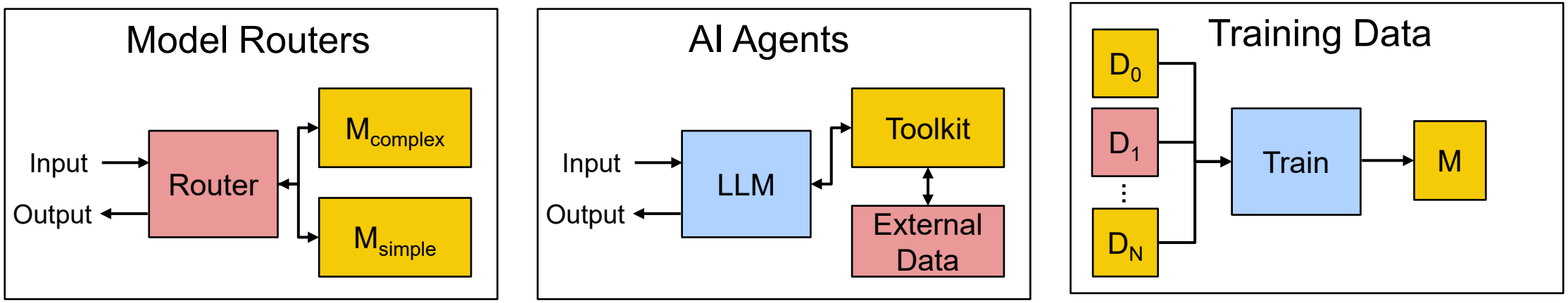


Examples relationships from[1]

[1] Bommasani et al. *Ecosystem Graphs: The Social Footprint of Foundation Models*, AIES 2024

Looking forward: Applying Property Attestations

Verifying Provenance: of models, external data sources, training data



How can we combine property attestations for verifiable output provenance of…

- Models when model router selects model that should respond to a query
- External data sources for AI Agents that may be vulnerable to indirect prompt injection
- Training data that has strongest influence over outputs

Points of discussion

Can be PAL*M[1] useful in other settings?

- Corporate policy compliance checks?

Is PAL*M[1] addressing a real need?

- What technical mechanisms needed for demonstrating AI policy or standard compliance?

[1] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026

Summary

Verifiable ML property cards prevent malicious model provider from including false information

Laminator[1] & PAL*M[2]: verifiable ML properties via h/w assistance:

- **Efficient:** Incurs low computation overhead
- **Scalable:** Attestations can be checked by multiple verifiers
- **Versatile:** Any ML property specified in python can be attested
- **Robust**: Resists evasion by malicious provers

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

Looking forward:

- Enabling verifiable ML properties in distributed and global settings
- Covering run-time properties and provenance of outputs of ML systems

[1] Duddu et al. *Laminator: Verifiable ML property cards using hardware-assisted attestations*, CODASPY 2025
[2] Chantasantitam et al. *PAL*M: Property Attestation for Large Generative Models*, arXiv 2026