This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2023.3303435

Intellectual Property Protection of Deep Learning Systems via Hardware/Software Co-design

Huili Chen† Cheng Fu† Bita Darvish Rouhani* Jishen Zhao† Farinaz Koushanfar†
† University of California San Diego * Microsoft
† {huc044, cfu, jzhao, farinaz}@ucsd.edu * bita.rouhani@microsoft.com

***Abstract*—Recent advances in model piracy have uncovered a new security hole for malicious attacks endangering the Intellectual Property (IP) of Deep Learning (DL) systems. This manuscript features our research titled "DeepAttest: An End-to-End Attestation Framework for Deep Neural Networks" [1] that is selected for the 2021 Top Picks in hardware and embedded security. DeepAttest is the first end-to-end framework that achieves reliable and efficient IP protection of DL devices with hardware-bounded usage control. We leverage device-specific model fingerprinting and Trusted Execution Environment (TEE) to ensure that only DL models with the device-specific fingerprint can run inference on protected hardware.**

***Index Terms*—Intellectual property protection, Deep learning hardware, Attestation, Digital fingerprinting**

I. Introduction

Deep Neural Networks (DNNs) have enabled a paradigm shift in various real-world applications due to their unprecedented performance and the capability of automatically learning informative representations for desired tasks. Although advanced learning systems are critical to ensure the high performance of autonomous agents, the investigation of their vulnerability to Intellectual Property (IP) piracy attacks is still in its infancy. To facilitate practical deployment and reliable technology transfer, protecting the IP of the emerging Deep Learning (DL) hardware is of great importance.

There has been a line of research that reveals the vulnerability of DL models to model piracy attacks [2], [3]. The malicious adversary might intend to claim the authorship of well-trained DNNs deployed in publicly available settings or Machine Learning as a Service (MLaaS). Figure 1 visualizes the IP concern in the supply chain. The designer trains the DL model using large training data and computing resources. Therefore, the designer is the legal owner of the model. However, when the owner uploads the trained model to public platforms or deploys it as a remote service on the cloud, malicious users may steal the valuable DL model to gain financial benefits.

Prior works have proposed various methodological and architecture-level advancements to improve the performance and efficiency of DNN training/execution on diverse platforms [4], [5]. However, emerging DL platforms are susceptible to unregulated usage. For instance, the attacker might deploy a DNN from a competitor company on the DL device, or maliciously perturb the deployed model to divert its prediction. The end users may also intend to use the DL device after license expiration. The above-described hardware-based attacks may cause substantial financial loss for the DL hardware provider, motivating us to extend the IP protection of DL systems to the hardware level.

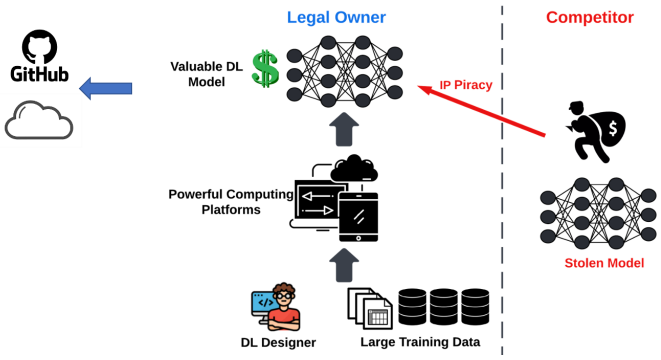


Fig. 1: Demonstration of the IP piracy attack against pre-trained DL models.

This paper features our framework named DeepAttest, which is the first end-to-end hardware-bounded IP protection and usage control technique for DNN applications. DeepAttest addresses the following three challenges:

I. Identifying a new attack vector against DL systems for unregulated device usage.
II. Characterizing and protecting DNN hardware via on-device attestation of the deployed DL model.
III. Devising optimized hardware architecture and an accompanying API to facilitate the deployment of DeepAttest.

We propose a novel formulation of hardware-bounded IP protection of DL programs using *on-device DNN attestation*. The high-level usage of DeepAttest is shown in Figure 3. DeepAttest works by designing a device-specific *fingerprint* and encoding it in the weight distribution of the DNN deployed on the protected hardware. The fingerprint (FP) of the DL model is later extracted with the support of the Trusted Execution Environment (TEE). We use the existence of the device-specific fingerprint as the attestation criterion for determining whether the queried DNN is legitimate. DeepAttest framework ensures that only authorized DNN programs pass the attestation (i.e., yield the matching FP) and are allowed to run inference on the protected hardware. We provision the device provider with a practical solution to controlling

Methods	Summary of different secured DNN evaluation methods						
	Required platform	Workload in TEE	Footprint of secure memory in TEE	Off-line data tamper	Online data tamper	Latency overhead on CPU (%)	**Size of secure copy
Fully TEE-based DNN Execution	*TCPU (SGX) TGPU (Graviton)	High Entire DNN evaluation	High All weights and input data	X	✓	>1000%	279 MB
Outsourced (Slalom)	TCPU + CPU TCPU + GPU	Medium Non-linear operations of DNN inference	High Partial weights and intermediate activations	X	✓	91.6%	271 MB
On-device Attestation (Our work)	TCPU + CPU TCPU + GPU TGPU + GPU	Low Fingerprint extraction operations	Low Partial weights	✓	✓	**1.3%**	**28 MB**

*TGPU refers to TEE in GPU, TCPU refers to TEE in CPU. Note that GPU can be substituted by other type of co-processor for attestation.
** Measured with 10 images on VGG16 model

Fig. 2: Comparison of existing secure DNN techniques and our work.

the application usage of his/her manufactured hardware and preventing unauthorized or tampered DNNs from execution.

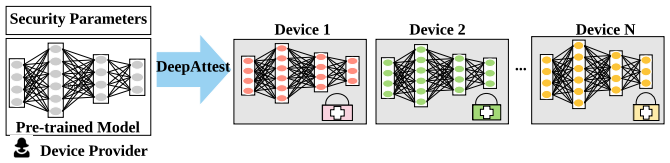


Fig. 3: DeepAttest is a hardware-bounded IP protection solution that takes the pre-trained model as input and returns a set of verifiable DNNs as outputs.

In summary, we make the following contributions:

- **Enabling effective on-device attestation for DNN applications.** DeepAttest is an end-to-end attestation framework that is capable of verifying the legitimacy of an unknown DNN with high reliability (preventing unauthenticated DNNs from execution) and high integrity (allowing legitimate DNNs to execute normal inference).
- **Characterizing the criteria for a practical attestation technique in the domain of deep learning.** We present a comprehensive set of metrics to profile the performance of pending DNN attestation techniques. The introduced metrics allow DeepAttest to provide a trade-off between the security level and the attestation overhead.
- **Leveraging an Algorithm/Software/Hardware co-design approach to devise an efficient attestation solution.** Our device-aware framework is equipped with careful design optimization to ensure minimal overhead and enhanced security. As such, DeepAttest provides a lightweight on-device attestation scheme that can be applied to resource-constrained embedded systems.
- **Demonstrating superior performance across various benchmarks.** DeepAttest's online attestation incurs negligible latency and energy consumption across different DL models and platforms, thus providing a viable solution to hardware-level IP protection.

II. Preliminaries and Background

A. IP Protection of DNNs

A line of research has focused on addressing the soft-IP concern of DL models using digital watermarking [6]–[9]. The authors of [6] encode the watermark (WM) in the transformation of model weights by adding constraints to the original objective function. The works [7], [9] extend DNN watermarking to remote cloud service. Particularly, they design specific image-label pairs as the watermark set and embed the WM in the model's decision boundary. DeepSigns [8] presents the first data-aware watermarking approach by embedding the WM in the dynamic activation maps.

All of the above-mentioned DNN watermarking techniques focus on *software-level model authorship proof*. Note that a naive implementation of DNN watermarking on the hardware is inadequate to provide an efficient and trustworthy attestation solution due to the unawareness of resource management and potential attacks. As such, these methods are not suitable for hardware-level IP protection. The works [7], [9] require DNN inference of multiple inputs on the local device and TEE-supported WM checking, which is prohibitively costly. Compared to weight-based watermarking [6], DeepAttest's fingerprint extraction from the DL model involves fewer computations since no extra sigmoid function is required.

B. Secure DNN Evaluation on Hardware

TEE Protection Mechanism. Modern CPU hardware architectures provide TEEs to ensure secure execution of confidential applications using program isolation. Intel SGX, ARM TrustZone, and Sanctum are examples of TEEs. To prevent malicious programs from interfering with executions in TEEs, data is encrypted by Memory Encryption Engine (MEE) before its is put into the Enclave Page Cache (EPC) located in the Processor Reserved Memory (PRM). We refer to this process as *secure memory copy*. Programs inside the TEE can read or write the data outside of the TEE while the programs outside of the TEE are not allowed to access the EPC. TEEs on other platforms utilize similar mechanisms to isolate the execution of the protected program by securing memory access to the code and data of the confidential program. Besides the CPU-level TEE support, Graviton [10] proposes a GPU architecture design to provide TEEs.

Comparison between Secure DNN Techniques. We compare DeepAttest and existing secure DNN techniques in Figure 2 in terms of platform requirement, the incurred workload in TEE, resistance to off-line/online data tamper, and capability of verifying DNN inference results. Prior secure DNN inference can be divided into two categories: full execution inside the TEE, and outsourcing partial computations from the TEE to untrusted environments. *DeepAttest identifies a*

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2023.3303435

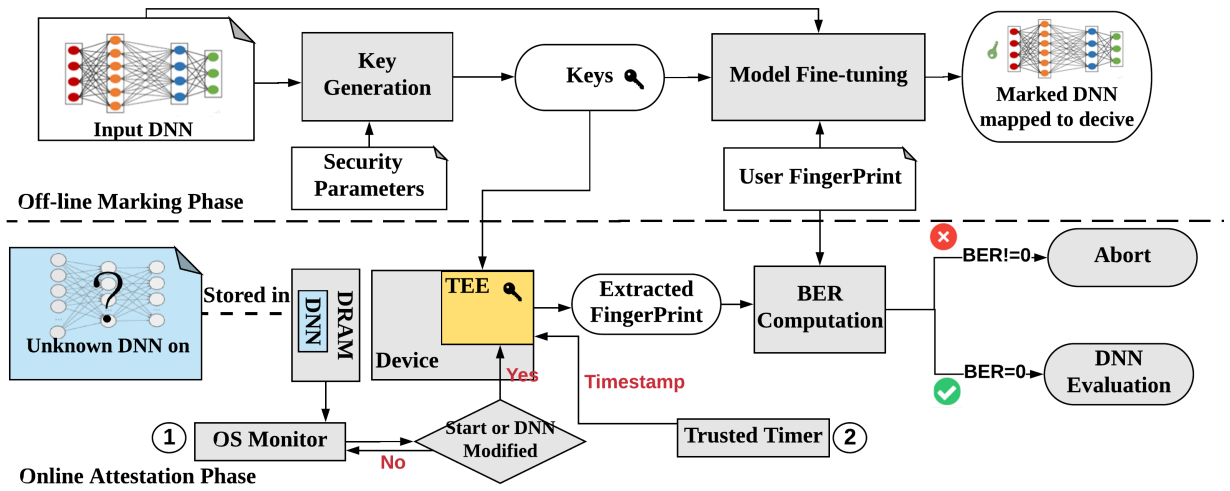


Fig. 4: Global flow of DeepAttest for on-device DNN attestation. In the offline marking stage, the device manufacturer generates secret FP keys (stored in TEE secure memory) and marked DL models. The online attestation stage ensures only users who purchase marked DNNs can pass attestation and prohibits the deployment of unauthorized DNNs.

new security dimension named 'device-level' IP protection and usage control. Therefore, DeepAttest is orthogonal to verifiable or privacy-preserving inference techniques.

III. DeepAttest Overview

DeepAttest is the first DNN attestation framework for device IP protection and usage control. Figure 4 illustrates the global flow of DeepAttest. The target *device* can be used with a co-processor (e.g., ASIC, FPGA) where usage control can be extended to cover. We identify the performance requirements for an effective DNN attestation scheme and outline them in Table I. We introduce the two main phases of DeepAttest, offline model marking and online DNN attestation below.

A. Offline DNN Marking

We explore the non-uniqueness of non-convex DL problems and generate a device-specific fingerprint (FP) for the target hardware. The FP is embedded in the probabilistic distribution of the selected parameters in the legitimate DNN by fine-tuning the model with an FP-regularized loss. The generated FP is stored in the secure memory of the hardware. Besides the position of the FP-carrying layer, DeepAttest's secret keys consist of three parts: a codebook C, an orthogonal basis matrix U, and a projection matrix X.

▪ **Fingerprint Construction.** DeepAttest constructs code modulated FPs as follows. Given the codebook C, the coefficient matrix B is computed from the linear mapping $b_{ij} = 2c_{ij} - 1$ where $c_{ij} \in \{0, 1\}$. The FP of the j^{th} user is generated from the linear combination of basis vectors:

$$\mathbf{f_j} = \sum_{i=1}^{v} b_{ij}\mathbf{u_i}, \tag{1}$$

▪ **Regularized Model Fine-tuning.** We embed the FP designed from Equation (1) in the weight distribution of the selected DL layer by integrating an FP-specific embedding loss to the original loss function ($\mathcal{L}_0$):

$$\mathcal{L} = \mathcal{L}_0 + \gamma\ Mean\_Square\_Error(\mathbf{f} - X\mathbf{w}). \tag{2}$$

Here, γ is the embedding strength that controls the trade-off between preserving the model's accuracy ($\mathcal{L}_0$) and enforcing the FP constraint ($\mathcal{L}_{FP}$). The vector $\mathbf{w}$ is the flattened averaged weights of the target layers that carry the FP information.

B. Online DNN Attestation

DeepAttest utilizes a *hybrid trigger* mechanism to prevent static and dynamic data tamper. When OS detects the DNN start request, we enable the *static* trigger. For the *dynamic* trigger, we design it from two sources: (i) Memory change signal provided by OS monitoring. When OS detects the status change of pages allocated for the DNN program, we raise a dynamic trigger signal; (ii) A fixed-frequency timestamp signal from the trusted timer [11] in the TEE.

When attestation is activated by the hybrid trigger, DeepAttest securely extracts the fingerprint from the deployed DL model with the support of TEE and compares it with the ground-truth value stored in secure memory. Algorithm 1 outlines the steps of DeepAttest's online attestation. The queried DNN is determined to be legitimate and permitted for normal inference if it yields a matching FP with the pre-stored one. Otherwise, the DNN program fails the attestation and its execution will be aborted on the protected device.

C. Hardware Optimizations

We develop DeepAttest framework using an Algorithm /Software /Hardware co-design principle to ensure the security and efficiency of online DNN attestation. For this purpose, we propose three novel hardware optimization techniques and discuss each one below.

▪ **Shredder Storage.** To improve security against fault injection attacks on memory, DeepAttest deploy a *'shredder'* storage format instead of continuous storage. Particularly, we shuffle the model weights and store the resulting data in the

TABLE I: Requirements for an effective on-device attestation technique of DNN applications.

Requirements	Description
Fidelity	Functionality of the deployed DNN shall not be degrade as a result of FP embedding in the marking stage.
Reliability	Online attestation shall be able to prevent unauthorized DNN programs (including full-DNN program substitution and malicious fault injection) from executing on the specific device.
Integrity	Legitimate DNN programs shall yield the matching FP with high probability and run normal evaluation.
Efficiency	The online attestation shall yield negligible overhead in terms of latency and energy consumption.
Security	The attestation method shall be secure against potential attacks including fault injection and FP forgery.
Scalability	The attestation technique shall be able to verify DNNs of varying sizes.
Generalizability	The DNN attestation framework shall be compatible with various computing platforms.

Algorithm 1 Fingerprint extraction in the attestation stage.

INPUT: Queried DNN $(\mathscr{T}^{'})$**; Decoding threshold** (τ)**; Owner's FP keys** $(\{l, C, U, X\})$**.**

OUTPUT: Computed *BER* **of fingerprint matching.**

1 Trigger Generation: Produce a hybrid trigger signal:
$S_{hybrid} \leftarrow S_{static} \vee S_{dynamic}$

2 **if** S_{hybrid} == True **then**
Acquire weights in the marked layer:
$\mathbf{w}' \leftarrow Get\_Marked\_Weights\ (\mathscr{T}^{'}, l)$
Extract the FP vector: $\mathbf{f}' \leftarrow X\mathbf{w}'$
Recover the coefficient vector: $\mathbf{b}' \leftarrow \mathbf{f}'^{T}U$
Decode the code-vector:
$\mathbf{c}^{'} \leftarrow Hard\_Thresholding\ (\mathbf{b}^{'}, \tau)$
Check FP matching:
$BER \leftarrow Compute\_BER(\mathbf{c}^{'}, \mathbf{c})$
Return: Obtained *BER* for FP matching.

3: **else**
Go back to step 1

untrusted memory. Figure 5 shows the intuition of shredder storage. We can see that the probability of detecting the malicious memory blocks is higher when the fingerprint-marked blocks are randomly distributed compared to the detection probability with continuous allocation.

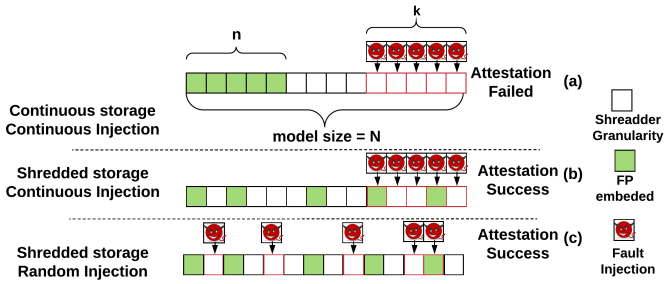


Fig. 5: Enhancing attestation security with shredder storage.

▪ **Data Pipelining and Early Termination.** Note that DeepAttest's FP extraction is parallelizable since each bit of the fingerprint can be recovered independently as can be seen from Algorithm 1. We leverage the independence of FP reconstruction and propose two optimization methods, data pipelining and early termination, to improve attestation efficiency.

We pipeline secure memory copy and the FP computation in TEE as shown in Figure 6. Particularly we create two pipelined TEE threads to transport the partitioned weight into the TEE and extract the FP, respectively. After the computation completes, the enclave memory occupied by FP extraction is freed and no intermediate results need to be stored. Therefore, DeepAttest supports DNNs with large weight size. In addition, we propose early termination for FP comparison and skips unnecessary computation as well as communication. The online attestation terminates and yields the failure signal once a mismatch between the extracted FP segment and the pre-specified device-specific FP is detected as shown in Figure 6.

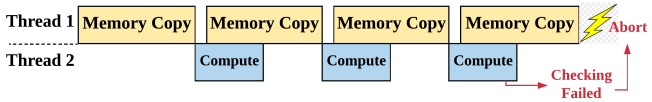


Fig. 6: Data Pipeline and Early Termination of DeepAttest on TEE execution.

IV. Evaluations

In this section, we present a comprehensive evaluation of DeepAttest's performance according to Table I and compare it with existing secure DNN inference techniques.

A. Performance Evaluation of DNN Attestation

We assess DeepAttest on various DNN architectures and datasets. Table III summarizes the DNN benchmarks used in our experiments. As for hardware platforms, we investigate DeepAttest on Intel-SGX (TEE-support CPU platform) and Graviton-based TEE simulation (GPU platform) [10].

DeepAttest Configuration. We use a codebook $C_{31\times31}$ that accommodates 31 users. We set the embedding strength to $\gamma = 0.1$ and fine-tune the pre-trained DL model for 5 epochs during offline DNN marking. For online FP extraction, we use a detection threshold of $\tau = 0.85$.

TABLE II: Evaluated benchmark summary.

Benchmark	Dataset	Model Size (MB)	Multiply-Add Operations (Mops)	Marked Layer Size (MB)
MNIST-CNN	MNIST	1.3	24	0.13 (10.1%)
CIFAR-WRN	CIFAR10	2.4	198	0.29 (12.3%)
VGG16	ImageNet	276.7	25180	28.3 (10.2%)
MobileNet	ImageNet	8.4	569	1.05 (12.6%)

▪ **Fidelity.** To evaluate the fidelity of DeepAttest, we compare the test accuracy of the FP-marked model with the one of the baseline model (without FP embedding). For all DNN benchmarks in Table II, our results show that DeepAttest has slight improvement or maintains the same level of accuracy compared to the baseline, thus satisfying the fidelity criteria.

▪ **Reliability and Integrity.** We evaluate the reliability and integrity of DNN attestation by measuring the Bit Error Rate (BER) of FP extraction when the deployed DNN is legitimate or unauthorized. We add random Gaussian noise on

the weights of the FP-marked layer and perform FP extraction on the resulting 'noisy' weight. Figure 7 shows how the BER of DeepAttest changes with varying noise intensity and spatial range. DeepAttest yields a zero BER on the marked weights, and a high BER when noise increases (i.e., DNN is altered), thus satisfying reliability and integrity in Table I.

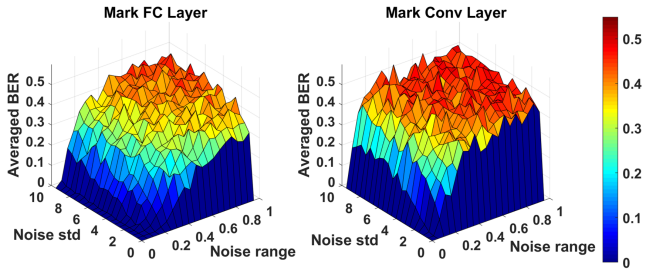


Fig. 7: Evaluate reliability and integrity of DeepAttest.

▪ **Efficiency.** We detail the memory and runtime overhead of DeepAttest with data pipelining and early termination optimizations in the following sections.

B. Comparison with Related Works

We compare DeepAttest with SOTA secure DNN inference techniques listed in Figure 2. Note that DeepAttest solves a new security concern (i.e., hardware-level IP protection and usage control) for DNNs that have not been identified by previous works. Since DeepAttest is orthogonal to existing privacy-oriented DNN inference methods, we provide a horizontal performance comparison of the relative overhead required by different security/privacy-protection DNN techniques.

▪ **Secure Memory Copy.** Recall that we define secure memory copy as the *communication* between untrusted memory and TEE-encrypted memory. Figure 8 illustrates the theoretical (minimal) size of secure memory copy required by different secure DNN techniques assuming the TEE is not memory-bounded. Slalom [12] incurs large overhead of secure memory copy since it outsources linear operations of DNN inference to the untrusted GPU. Fully TEE-based DNN evaluation only requires to transfer all weight data and input data, thus is less sensitive to the number of inputs. DeepAttest's memory copy size is not sensitive to the number of inputs since it adopts a hybrid triggering scheme where the attestation is performed on every batch of f images. Furthermore, the secure copy size of DeepAttest is small for a given attestation interval due to the deployment of shredder storage optimization, which ensures security for a smaller value of the marked ratio λ.

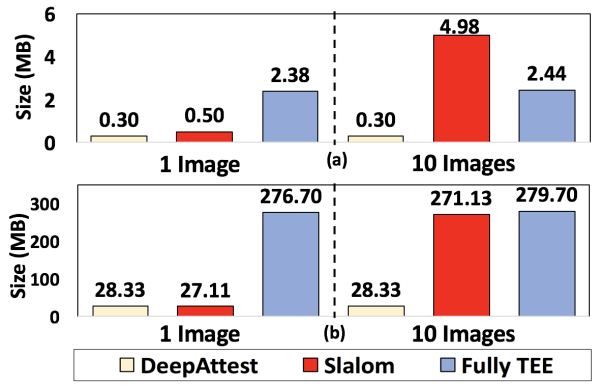


Fig. 8: Comparison of the theoretical size of secure memory copy to TEE required by different secure DNN techniques on (a) CIFAR-WRN and (b) VGG16 benchmark.

▪ **Latency.** Figure 9 shows the normalized latency required by different secure DNN inference methods and DeepAttest where the baseline is performed on the untrusted CPU. Running DNN inference fully inside TEE is 12.34× slower than insecure inference on average. Slalom [12] outsources linear operations to the untrusted CPU and non-linear parts to Intel-SGX, resulting in an average normalized latency of 1.72× to provide verifiable results. DeepAttest incurs negligible relative latency of 0.7% and 1.9% on VGG16 and MobileNet respectively, thus is highly efficient.

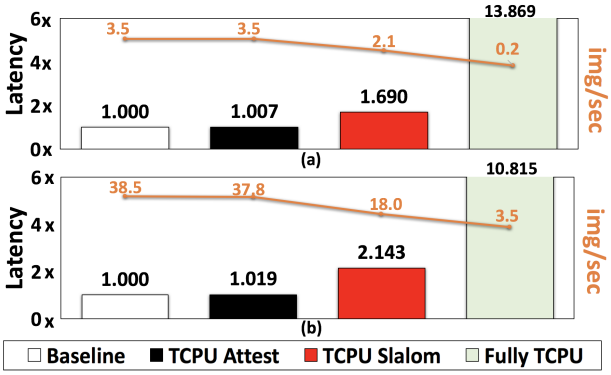


Fig. 9: Comparison of relative latency between different secure DNN techniques running on CPU for VGG16 (a) and MobileNet (b).

V. Discussion

We discuss two open questions about the applicability and security of DeepAttest in this section.

How to adapt DeepAttest on hardware without TEE? DeepAttest framework is general and can be extended to protect devices that do not have TEEs. The on-device attestation step of DeepAttest mainly involves *two matrix multiplications* as shown in Algorithm 1. If the processor does not have a TEE to perform the above computation in a secure environment, DeepAttest can Freivalds' algorithm , a verifiable outsourcing scheme, to verify the two matrix multiplications required for the fingerprint check in an untrusted environment.

How is DeepAttest impacted by vulnerabilities of TEEs? Known attacks on TEEs will compromise the security of DeepAttest from various possible aspects. For example, the side-channel analysis might be run against the TEE , leaking information about the device-specific secret fingerprint and facilitating FP forgery to bypass DeepAttest. Alternatively, the software timers in the TEE might be manipulated , which directly impacts the dynamic trigger used by DeepAttest. If the adversary makes the speed of the TEE timer faster, then the dynamic trigger will be activated more frequently and unnecessary verification might be performed, resulting in an

increased resource consumption. If the adversary makes the speed of the TEE timer slower, then the dynamic trigger of DeepAttest will be activated less often than expected, meaning that on-device attestation may not launch on a pre-scheduled frequency and increase the IP infringement risk.

VI. Conclusion and Broader Impact

DeepAttest is a holistic solution to enabling reliable technology deployment and protecting the commercial advantage of DL hardware providers. To the best of our knowledge, there is no prior investigation on hardware-bounded IP protection for DL models with full consideration of security, reliability, and efficiency. DeepAttest's formulation of on-device DNN attestation enables usage control and model authentication for DL hardware providers. DeepAttest is developed based on an Algorithm/Software/Hardware co-design approach to achieve secure and lightweight model signature extraction and comparison. Our vision is to constrain the usage of intelligent hardware to specific DL models that are pre-specified by the device provider. Such a usage control scheme not only prevents the execution of tampered DL models but also prohibits the misuse of the devices for undesired purposes. DeepAttest is the first on-device DNN attestation framework that verifies the legitimacy of the deployed DNN before allowing it to execute normal inference, which is practically useful to industrial practitioners for reliable technology transfer.

Acknowledgement

This work was supported in part by National Science Foundation (NSF) Trust-Hub under award number CNS-2016737, and NSF TILOS under award number CCF-2112665.

References

[1] H. Chen, C. Fu, B. D. Rouhani, J. Zhao, and F. Koushanfar, "Deepattest: An end-to-end attestation framework for deep neural networks," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 487–498.

[2] M. Chen and M. Wu, "Protect your deep neural networks from piracy," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2018, pp. 1–7.

[3] R. Yasaei, S.-Y. Yu, E. K. Naeini, and M. A. Al Faruque, "Gnn4ip: Graph neural network for hardware intellectual property piracy detection," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 217–222.

[4] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 764–775.

[5] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.

[6] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 2017, pp. 269–277.

[7] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1615–1631.

[8] B. Darvish Rouhani, H. Chen, and F. Koushanfar, "Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 485–497.

[9] J. Guo and M. Potkonjak, "Evolutionary trigger set generation for dnn black-box watermarking," *arXiv preprint arXiv:1906.04411*, 2019.

[10] S. Volos, K. Vaswani, and R. Bruno, "Graviton: Trusted execution environments on gpus," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 681–696.

[11] Intel, "Intel software guard extensions sdk," https://software.intel.com/en-us/sgx-sdk-dev-reference-sgx-get-trusted-time, 2017.

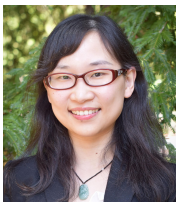
[12] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," *arXiv*, 2018.

This article has been accepted for publication in IEEE Design & Test. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MDAT.2023.3303435

H uili Chen received her Ph.D. from the Department of Electrical and Computer Engineering at University of California, San Diego in 2022. Her research interests include intellectual property protection of machine learning systems, security assessment of deep neural networks, and adapting deep learning to solve security problems in other domains. (huc044@ucsd.edu)

C heng Fu received his M.Sc. in University of Michigan, Ann Arbor. He is currently a Ph.D. student at Computer Science and Engineering Department at University of California, San Diego. Cheng's research interests lie at the intersection of machine learning, computer architecture, and security. (cfu@ucsd.edu)

B ita Darvish Rouhani received her Ph.D. in Electrical and Computer Engineering at University of California, San Diego. She is currently a principal research manager at Microsoft. Her research interests include deep learning, safety of machine learning models, and low-power computing. (bita.rouhani@microsoft.com)



J ishen Zhao received her Ph.D. degree in Computer Science and Engineering, Pennsylvania State University. She is currently an Associate Professor in Computer Science and Engineering Department at University of California, San Diego. Her research interests include computer architecture and systems research that bridges system software and hardware design, with an emphasis on memory and storage systems, machine learning and system co-design, and reliability. (jzhao@ucsd.edu)

F arinaz Koushanfar received her Ph.D. in Electrical Engineering and Computer Science from UC Berkeley. She is currently a professor and Henry Booker Faculty Scholar of Electrical and Computer Engineering in University of California, San Diego. Koushanfar's research interests include embedded and cyber-physical systems design, embedded systems security, and design automation of domain-specific/mobile computing. She is an IEEE Fellow. (farinaz@ucsd.edu)