ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

SOTER:Guarding Black-box Inference for General Neural Networks at the Edge

ATC'22

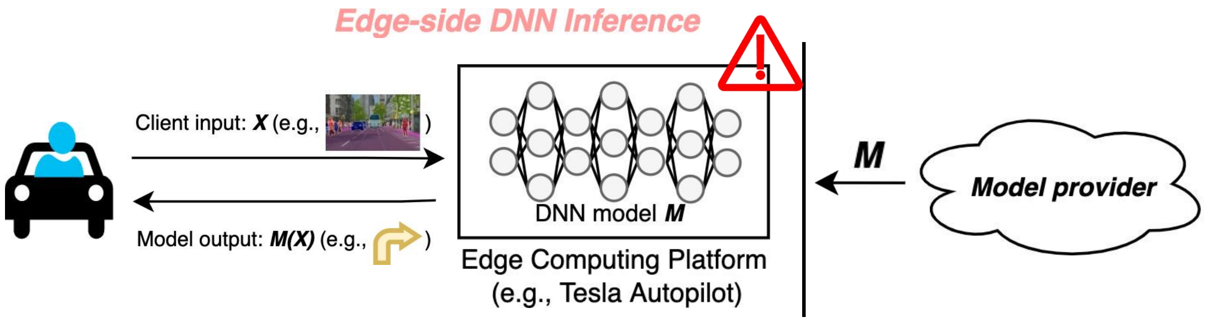
Tianxiang Shen, Ji Qi, Jianyu Jiang, Xian Wang, Siyuan Wen, Xusheng Chen, Shixiong Zhao, Sen Wang,
Li Chen, Xiapu Luo, Fengwei Zhang, Heming Cui

Background: Edge-side DNN Inference

- **Giant companies (e.g., Google) provide well-trained Deep Learning (DL) models to clients**
  - DL models, especially **Deep Neural Networks** (DNN), serve numerous mission-critical AI applications

Autonomous Driving | Home Monitoring | Virtual Assistance | Speech Recognition …

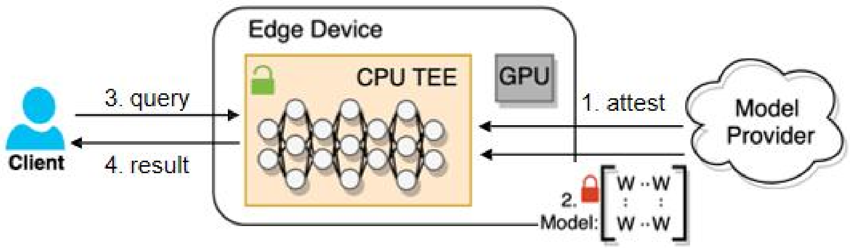
- **To provide high-quality (low-latency) services, DNN models are usually deployed on edge-side user devices**
  - Clients (i.e., users) run *edge-side DNN inference* to get real-time results



  - However, sensitive model parameters are exposed, and inference can be easily interfered at the untrusted edge!
- **In sum, edge-side inference requires low latency, high accuracy with confidentiality and integrity protection**

Background: Trusted CPU TEE & Untrusted GPU

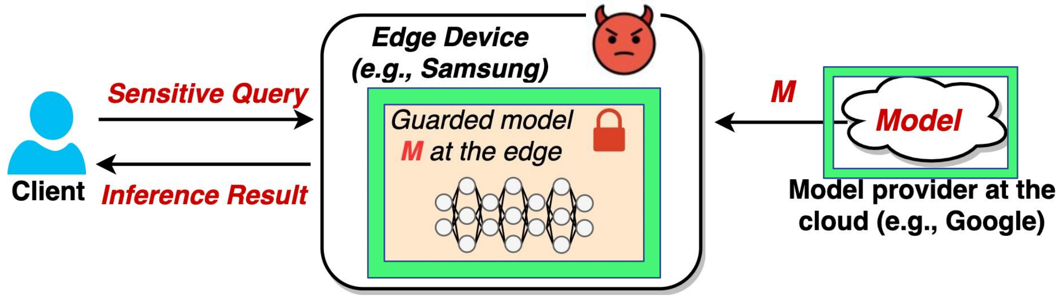
- **Trusted Execution Environment (TEE) is promising to protect model confidentiality**
  - TEEs (e.g., **Intel SGX**, ARM TrustZone) provides **data confidentiality** and **code integrity** guarantees
  - TEEs are widely used to **protect edge services**



- **Edge-side TEEs are trusted, but edge-side GPUs are untrusted**
  - **CPU TEE does not support GPU, model providers cannot trust third-party GPUs**

Requirements for Edge-side DNN Inference

➢ **Deployment scenario**



➢ **An *ideal* edge-side inference system should meet the following requirements:**

Performance
- ***Low latency:*** utilize co-located GPU accelerator to speed up model inference
- ***High accuracy***: retain the same accuracy as the original model

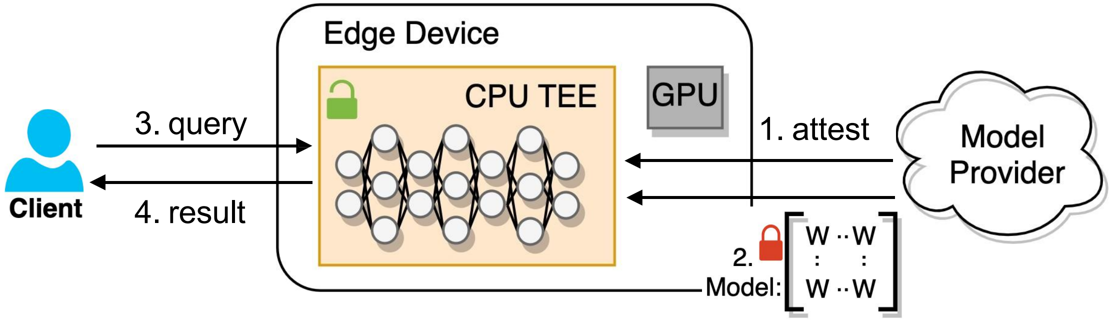
Security
- ***Model confidentiality***: model parameters' plaintexts should be hidden
- ***Inference integrity:*** any attacks (e.g., malicious modifications) on inference results should be detected

Prior work: TEE-shielding Approach

➢ **Existing TEE-based inference systems include *TEE-shielding approach* and *partition-based approach***

➢ ***TEE-shielding* approach (e.g., MLCapsule [CVPR ’21])**

- ***How it works***

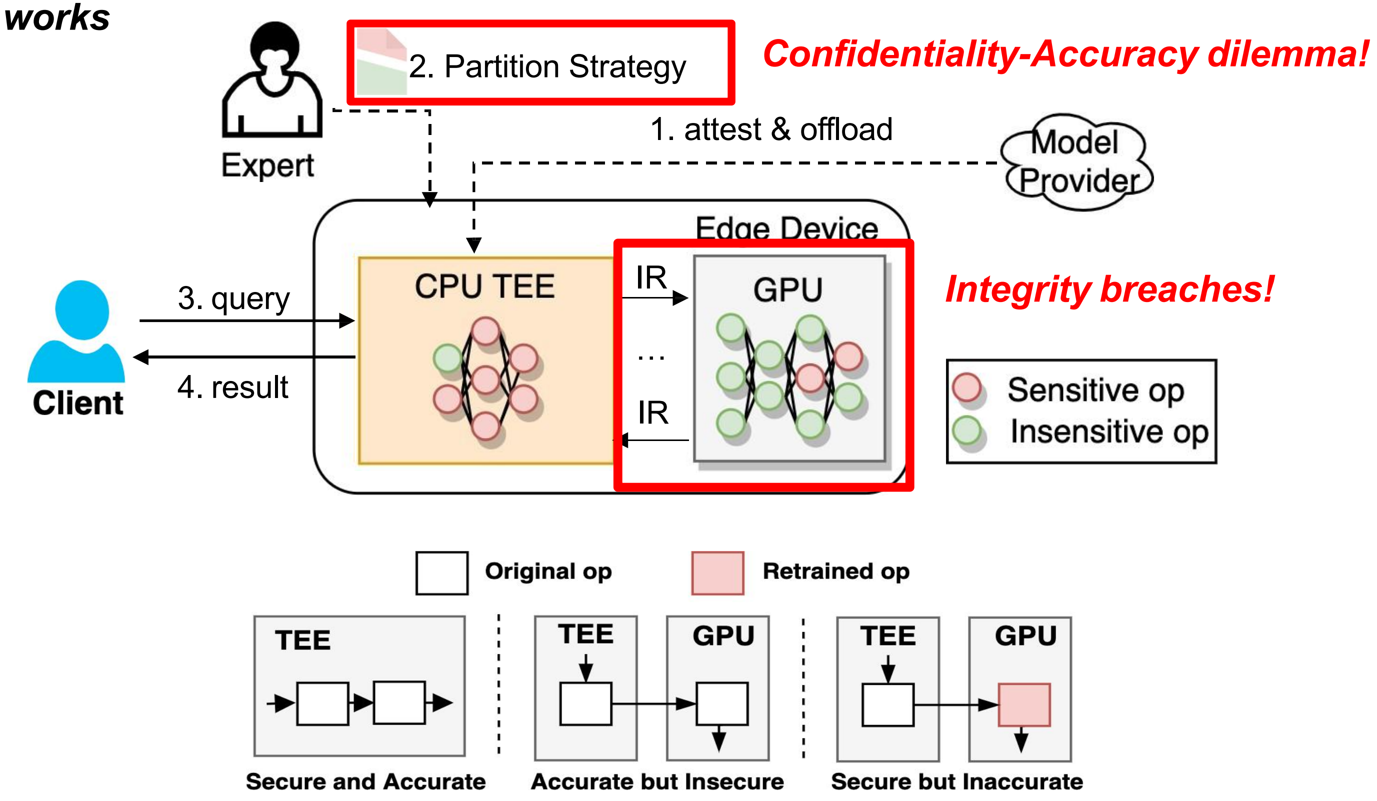


1. **Attest** to the TEE-equipped edge device
2. **Offload** and **decrypt** the encrypted model in an attested TEE enclave
3. **Take** client **input** to run inference purely inside the CPU TEE
4. **Return** the **inference** result back to the client

- ***Advantages:*** Protect model confidentiality and inference integrity; Retain high accuracy
- ***Limitations:*** No GPU acceleration with extremely high inference latency (up to 36.1*X*) than insecure GPU inference

Prior work: Partition-based Approach

- **Partition-based approach (e.g., AegisDNN [RTSS '21], eNNclave [AISec '20])**
  - ***How it works***



  - ***Advantages:*** Low latency with GPU acceleration
  - ***Limitations***: Incur either confidentiality loss or accuracy loss; Integrity breaches on partitioned model

Goals of Our Solution: SOTER

➢ **SOTER is a partition-based inference system that achieves all desired properties for edge-side DNN inference**

- **Accelerate** heavy-weight computation with GPU and **retain high accuracy** as the original model
- **Protect model confidentiality** by hiding all parameters' plaintexts
- **Detect integrity breaches** (e.g., malicious modifications) on inference results

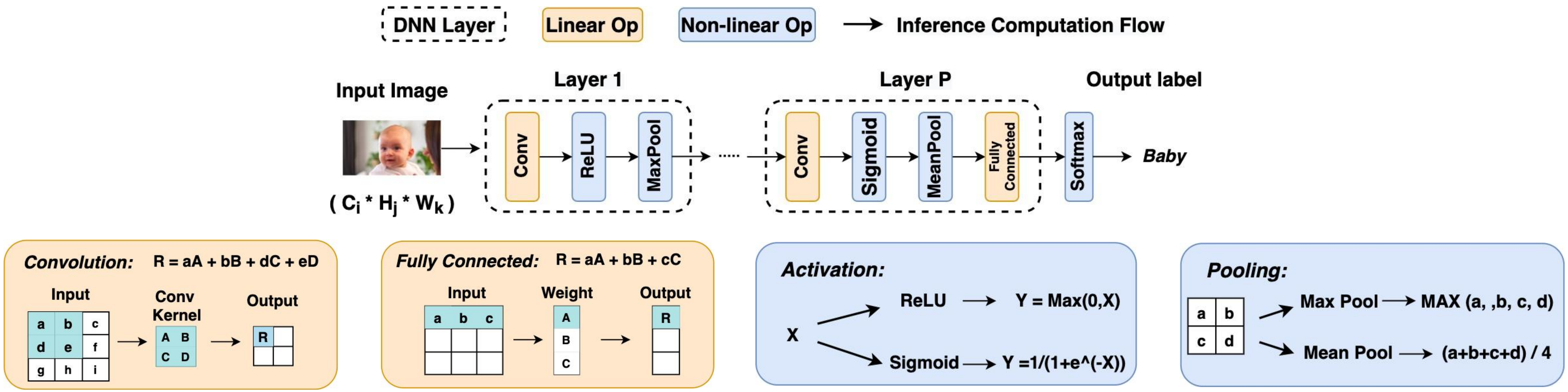
	GPU Acceleration	**No Accuracy Loss**	**Model Confidentiality**	**Inference Integrity**
MLCapsule	😐	🙂	🙂	🙂
eNNclave	🙂	😐	🙂	😐
AegisDNN	🙂	🙂	😐	😐
SOTER	🙂	🙂	🙂	🙂

➢ **To achieve these goals, SOTER asks two questions:**

- ***Q1:*** *How can we utilize untrusted GPU for acceleration without sacrificing confidentiality or accuracy?*
- ***Q2:*** *How to efficiently detect integrity breaches outside the TEE?*

Recap DNN Model Architecture

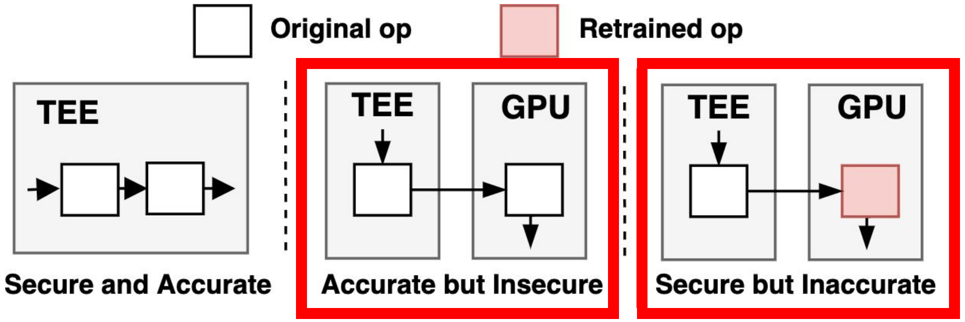
- **Recap DNN model architecture**



- **Associativity of common DNN operators:** $F(x) = \mu^{-1} F(\mu \cdot x)$

Bridging the Confidentiality-Accuracy Gap (Q1)

- **Confidentiality-accuracy dilemma**

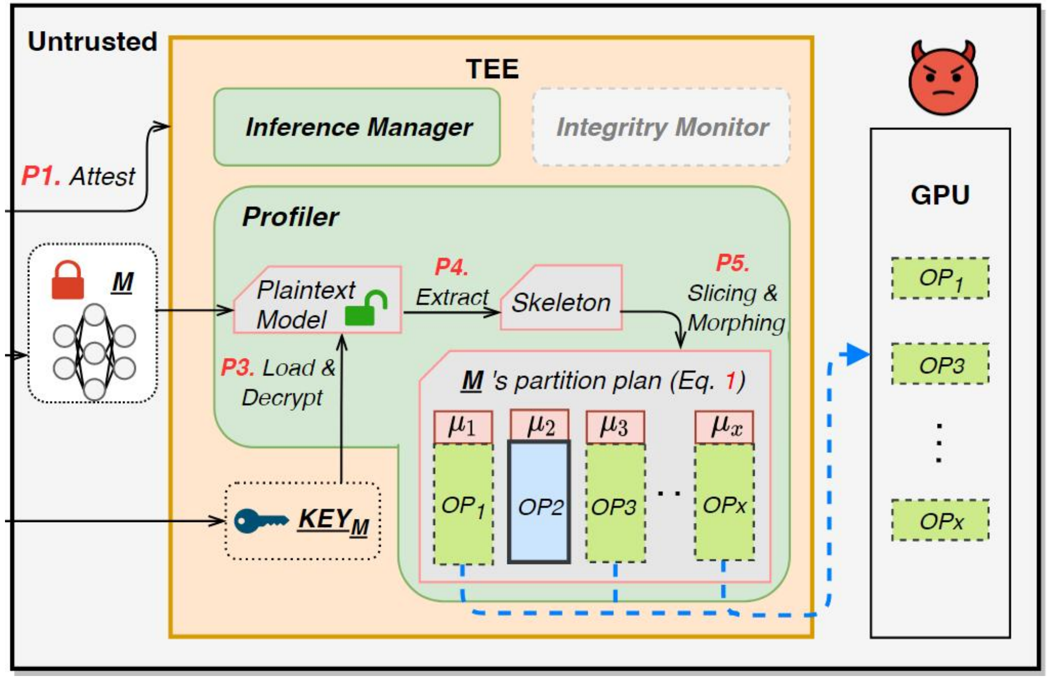


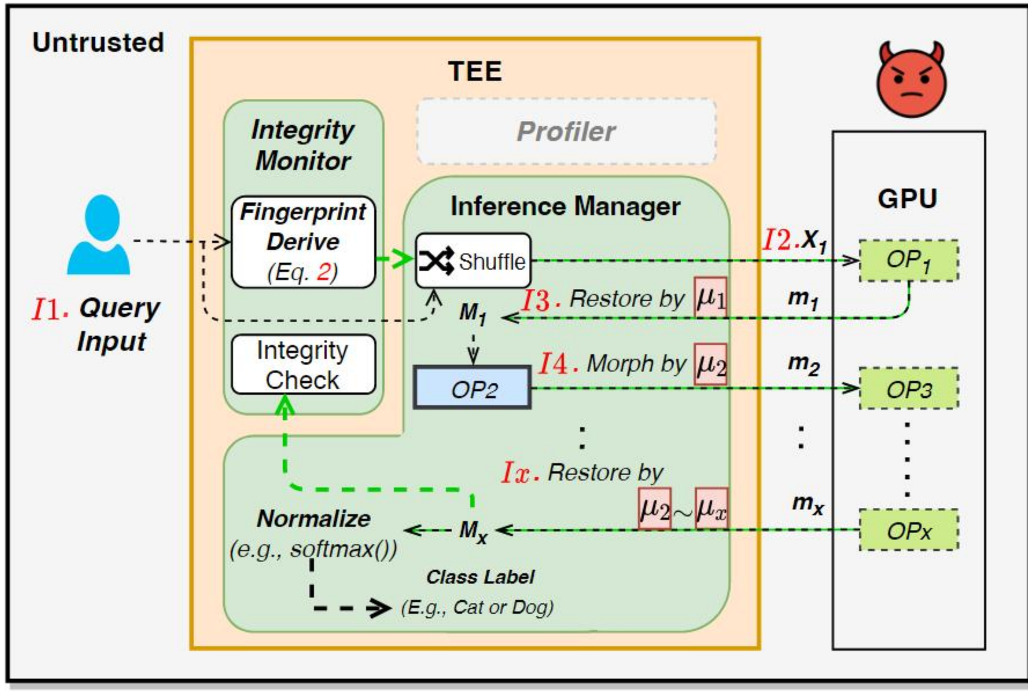
- **SOTER's key weapon: the general associativity property of common inference operators**

sensitive! $\boxed{(\mu^{-1}\cdot\mu)F(x) = \mu^{-1}}\boxed{F(\mu\cdot x)}$ insensitive -> GPU

sensitive -> TEE

Bridging the Confidentiality-Accuracy Gap (Q1)



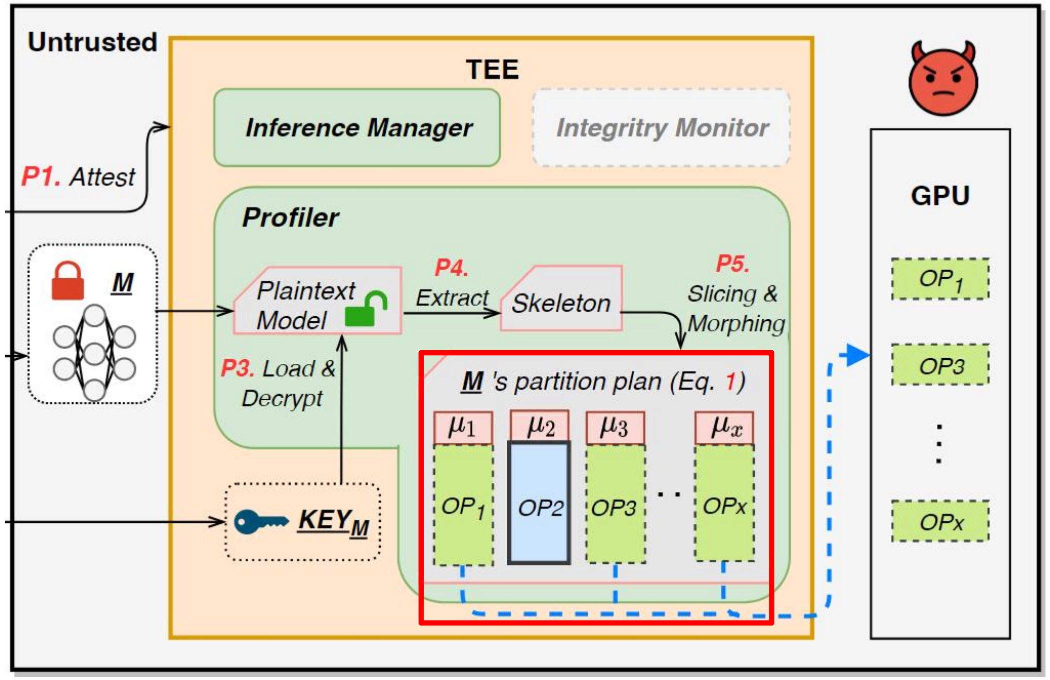


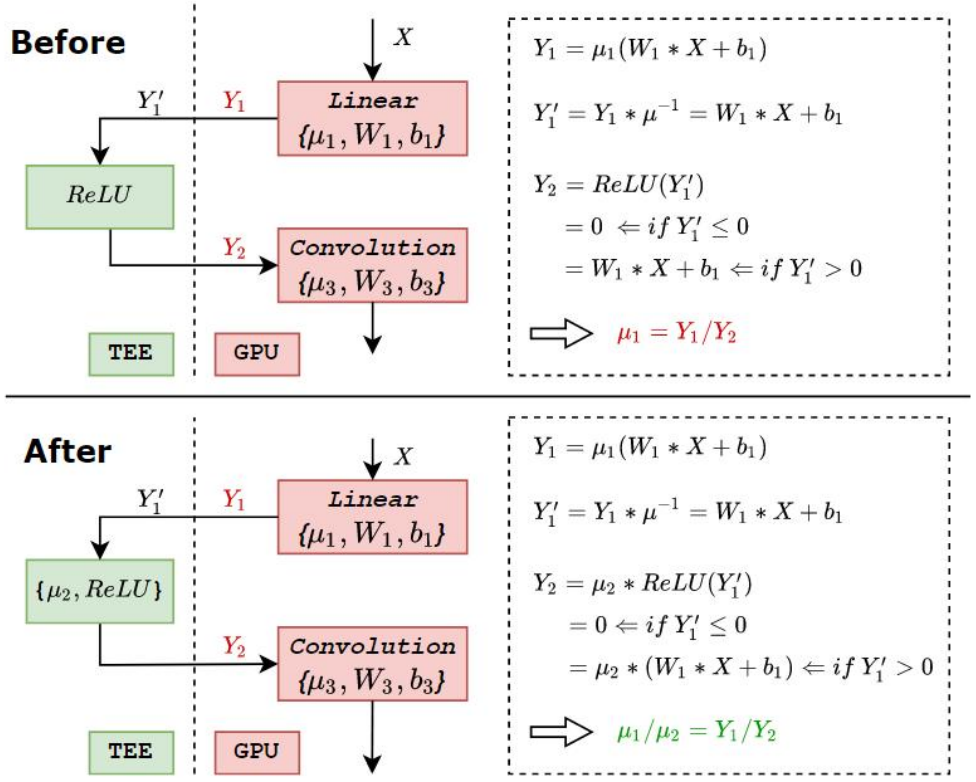
$$(\mu^{-1} \cdot \mu) F(x) = \mu^{-1} F(\mu \cdot x)$$

- **Major workflow**
  - **Step 1:** Automatically profile an encrypted model in TEE
  - **Step 2:** Morph a portion of associative operators' parameters with *hidden scalars*
  - **Step 3:** Partition morphed operators to run on GPU
  - **Step 4:** Execute operators in order, transmit IRs between kernels, restore execution results with hidden scalars in TEE

Bridging the Confidentiality-Accuracy Gap (Q1)

Why μ is added to all operators?

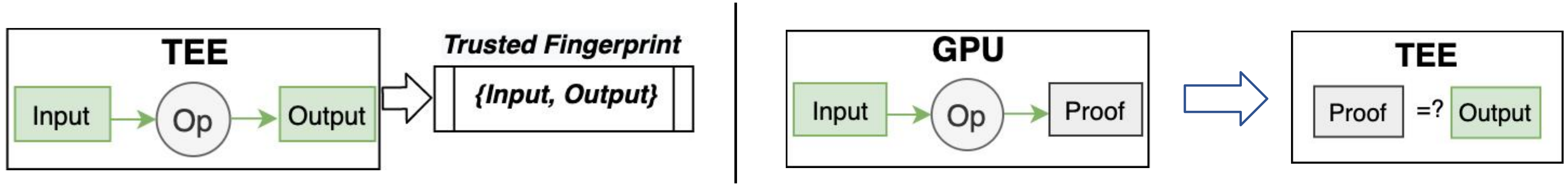




- ***Q1:** How can we utilize untrusted GPU for acceleration without sacrificing confidentiality or accuracy?* √

Detecting Integrity Breaches (Q2)

- **Partition-based system *inevitably* open access to integrity breaches outside the TEE**
- **Detect integrity breaches: a straw man *Trusted Fingerprint* (TF) re-computing approach**



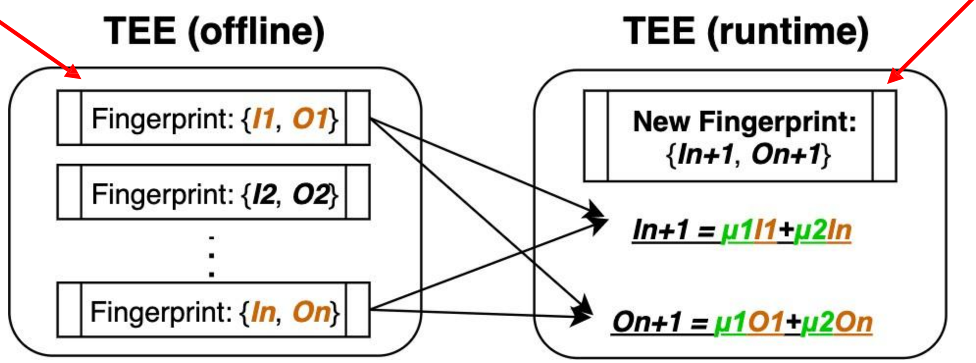
- **Key challenge: Obliviousness-timeliness dilemma**
  - If we use **fixed TF**, the adversary can easily **observe and bypass the TF detection**
  - If **generate new** TF as regular user input in CPU TEE, TFs become **oblivious** to observe, but TF generation (in CPU TEE) becomes the performance bottleneck, leading to **slow** detection

Detecting Integrity Breaches (Q2)

➢ **SOTER solves the challenge using the same associativity observation from confidentiality protection**

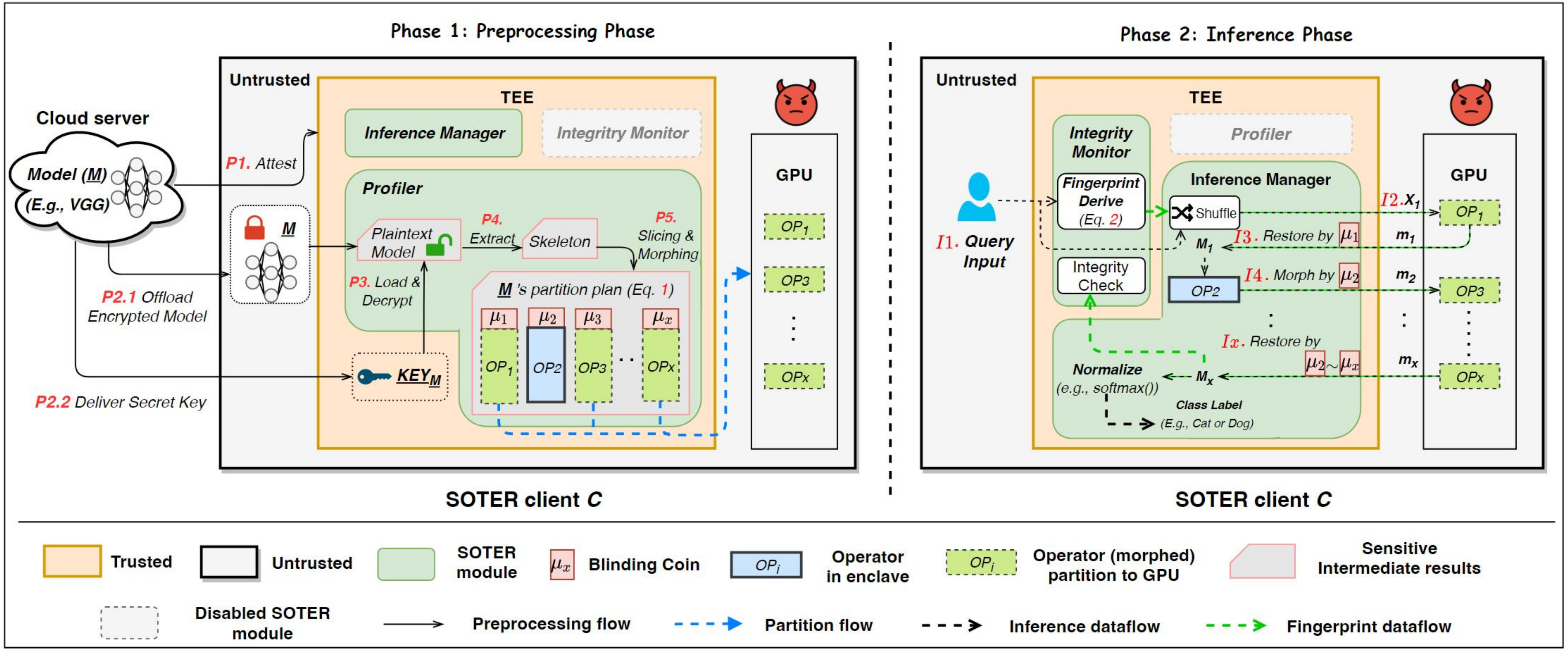
- Associativity variant: If $F(X1) = Y1$; $F(X2)=Y2$; …; $F(Xn)=Yn$, then $F(\mu 1X1+ \mu 2X2+…+ \mu nXn)= \mu 1Y1+ \mu 2Y2+…+ \mu nYn$

Step 1:
Prepare *cornerstone TFs* in the preprocessing phase



Step 2:
Generate new *scalars* and use the *associativity variant* to efficiently produce new TFs

System Overview



Low latency ***High accuracy*** ***Model confidentiality*** ***Inference integrity***

Implementation and Evaluation

- **Implementation Details**
  - Implemented on PyTorch and Graphene-SGX

- **Baseline secure inference systems**
  - MLCapsule [CVPR '21]: **only on CPU**
  - AegisDNN [RTSS '21]: **confidentiality protection problems**
  - eNNclave [AISec '20]: **accuracy loss**

- **Evaluation settings in our dedicated cluster**
  - Dell R430 server with 2.60GHZ Intel E3-1280 V6 CPU, 64GB memory, and SGX hardware support
  - A GPU farm with Nvidia 2080Ti GPUs, each GPU had 11GB physical memory
  - Evaluated on VGG19, Alexnet, Resnet152, Densenet121, Multi Layer Perception, and Transformer

Evaluation Questions

- **How is SOTER's end-to-end performance compared to baselines?**
- **How is SOTER's confidentiality protection compared to baselines?**
- **Are SOTER's trusted fingerprint oblivious to the adversary outside the TEE?**

End-to-end performance

- **Figure 1 shows the inference latency (normalized to insecure GPU inference, red dotted line) compared to three baselines (SOTA TEE-shielding and partition-based approach) running six prevalent DNN models**
  - SOTER achieved **1.21*X* ~ 4.29*X* lower inference latency** than TEE-shielding MLCapsule
  - SOTER enforced **integrity protection**, **with only 1.03*X* ~ 1.27*X* higher** inference latency than partition-based AegisDNN

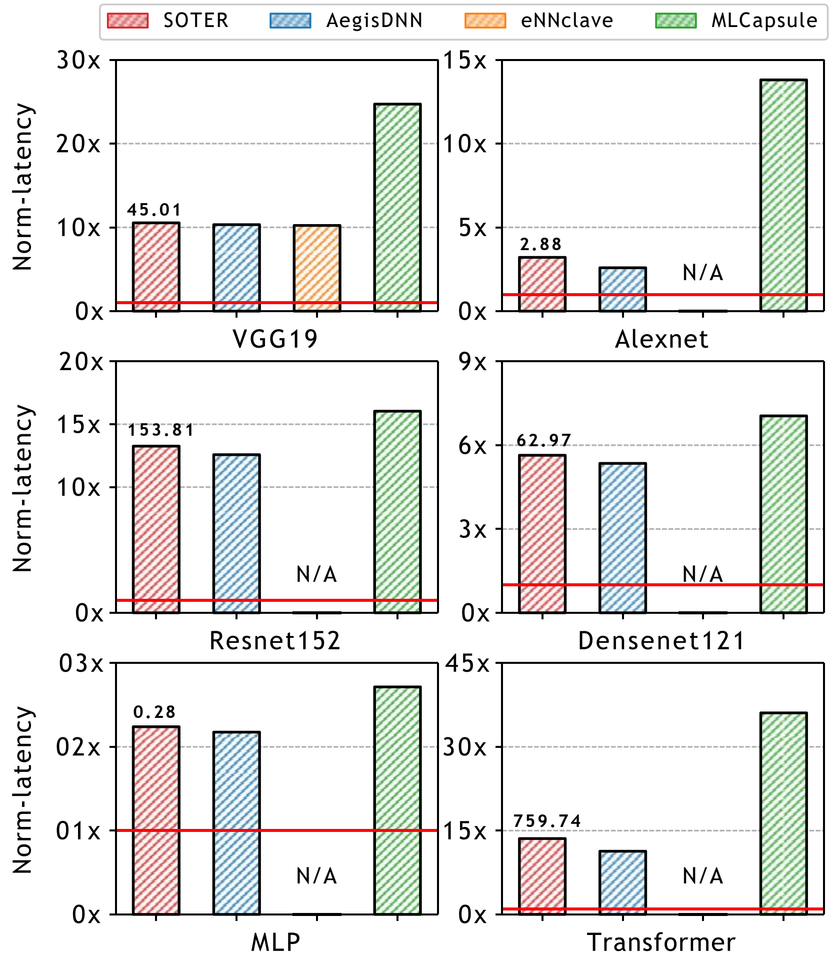
SOTER's inference results (in milliseconds)						
Model	MLP	AN	VGG	RN	DN	TF
P1: CPU (TEE)	0.19	1.65	25.38	92.18	41.65	439.52
P2: GPU	0.05	0.71	14.24	33.97	13.71	204.93
P3: Kernel Switch	0.01	0.18	0.83	25.98	5.6	41.52
P4: Integrity Check	0.03	0.34	4.56	14.75	6.02	73.77
End-to-end (P1+P2+P3+P4)	0.28	2.88	45.01	153.88	62.97	759.74

Table 3: End-to-end latency breakdown of SOTER.

AegisDNN's inference results (in milliseconds)						
Model	MLP	AN	VGG	RN	DN	TF
P1: CPU (TEE)	0.19	1.54	22.89	88.14	36.75	404.87
P2: GPU	0.07	0.67	19.96	52.3	20.07	198.64
P3: Kernel Switch	0.01	0.12	0.85	7.12	1.81	29.61
End-to-end (P1+P2+P3)	0.27	2.33	43.7	146.56	58.63	633.12

Table 4: End-to-end latency breakdown of AegisDNN.

Figure 1



Security Evaluation

- (**Confidentiality**) Even if SOTER completely hides partitioned operators' plaintexts, an adversary may still conduct ***model stealing attacks*** to train a ***substitute model (SM)***

  (A **higher** accuracy/BLEU of SM means **more** confidentiality loss)

  - SOTER achieved **stronger confidentiality** protection than
  - SOTER achieved **the same strong confidentiality** protection as eNNclave while eNNclave sacrifices inference accuracy

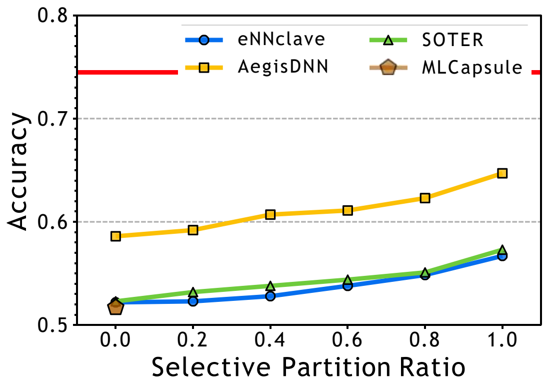


Figure 2.a (on VGG19)

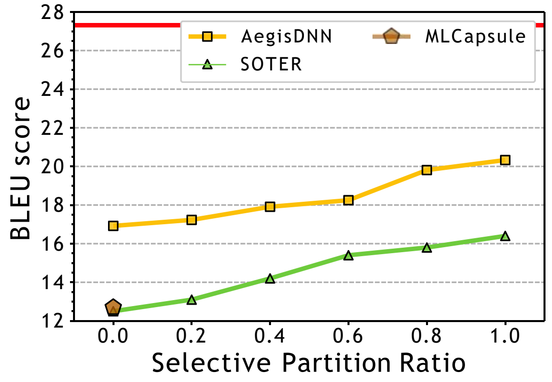


Figure 2.b (on Transformer)

- (**Integrity**) Compare SOTER's oblivious trusted fingerprint (Figure 3.a) with the straw man fixed trusted fingerprint approach (Figure 3.b)

  - SOTER's fingerprints are **oblivious** to the adversary because the L2 distance distribution of fingerprints shares the same form of normal distribution as client's normal query input

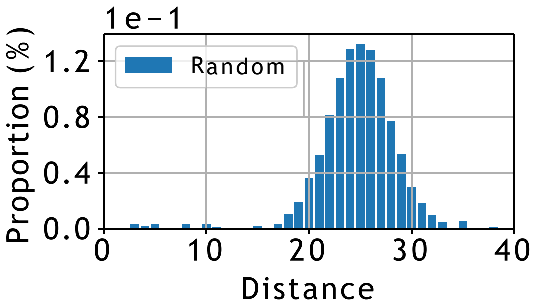


Figure 3.a (w oblivious TF)

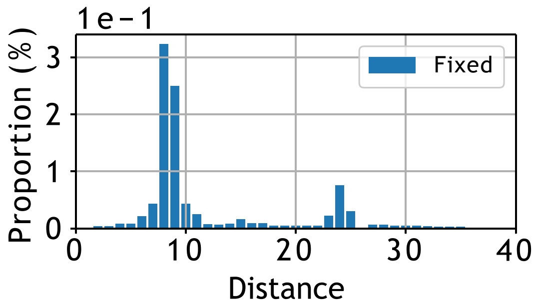


Figure 3.b (w/o oblivious TF)

Conclusion

- In this paper, we present SOTER, the first work that achieves **model confidentiality**, **low-latency** and **high-accuracy** with **integrity protection** for **general** neural network inference
- SOTER can also help with protecting models **on untrusted cloud servers**
- SOTER's future work is broad:
  - SOTER can be extended to multiple GPUs and TEEs for distributed model inference

My opinions

- 1. Need hardware support
- 2. Cannot apply to mobile phone now
- 3. GPU TEE is on the way, this work may be replaced

https://github.com/hku-systems/SOTER

Thank You

Presented by Ye Wan

2023-02-13