Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware

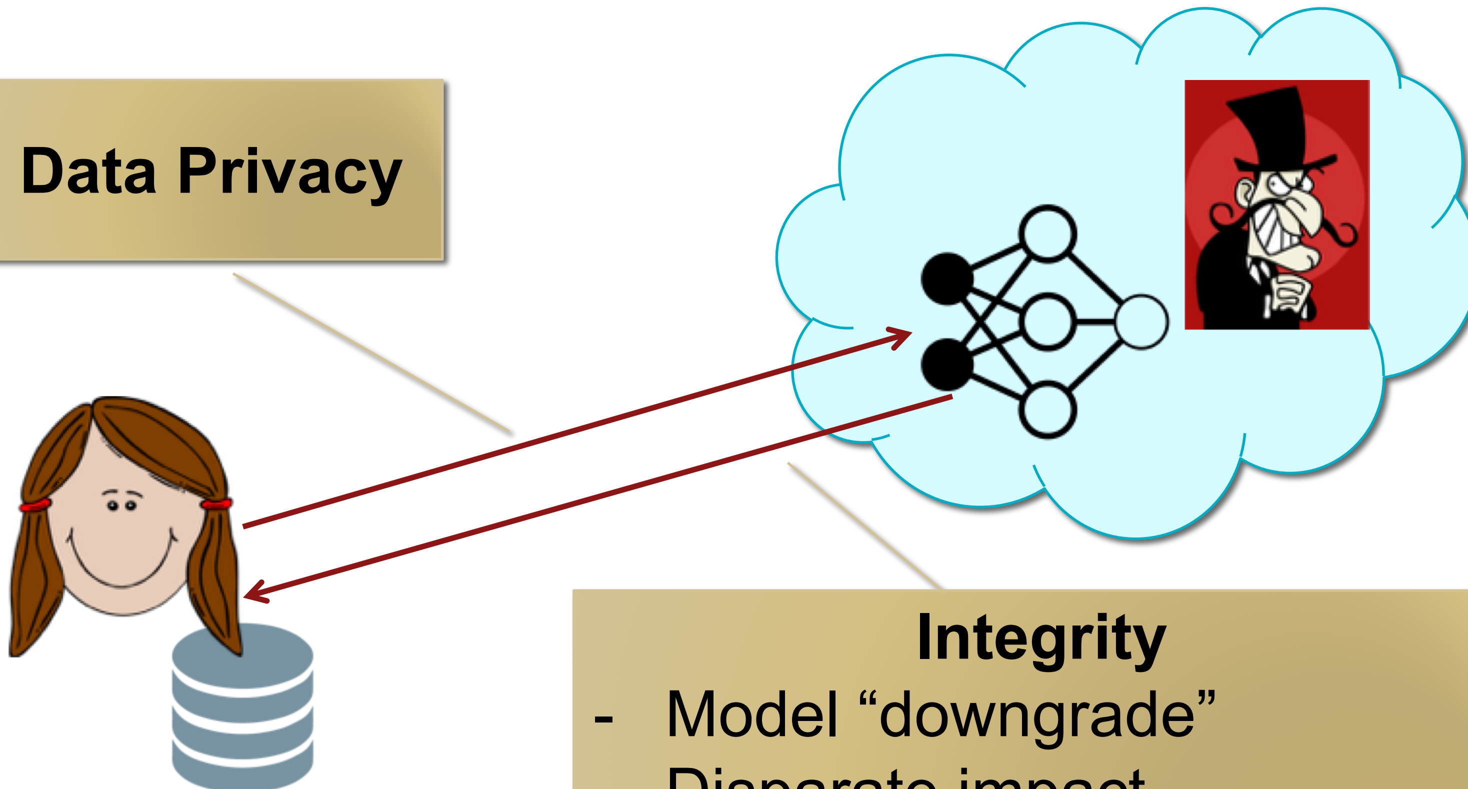
Florian Tramèr
(joint work with Dan Boneh)

Intel, Santa Clara – August 30th 2018

Trusted execution of ML: 3 motivating scenarios

1. Outsourced ML

Data Privacy

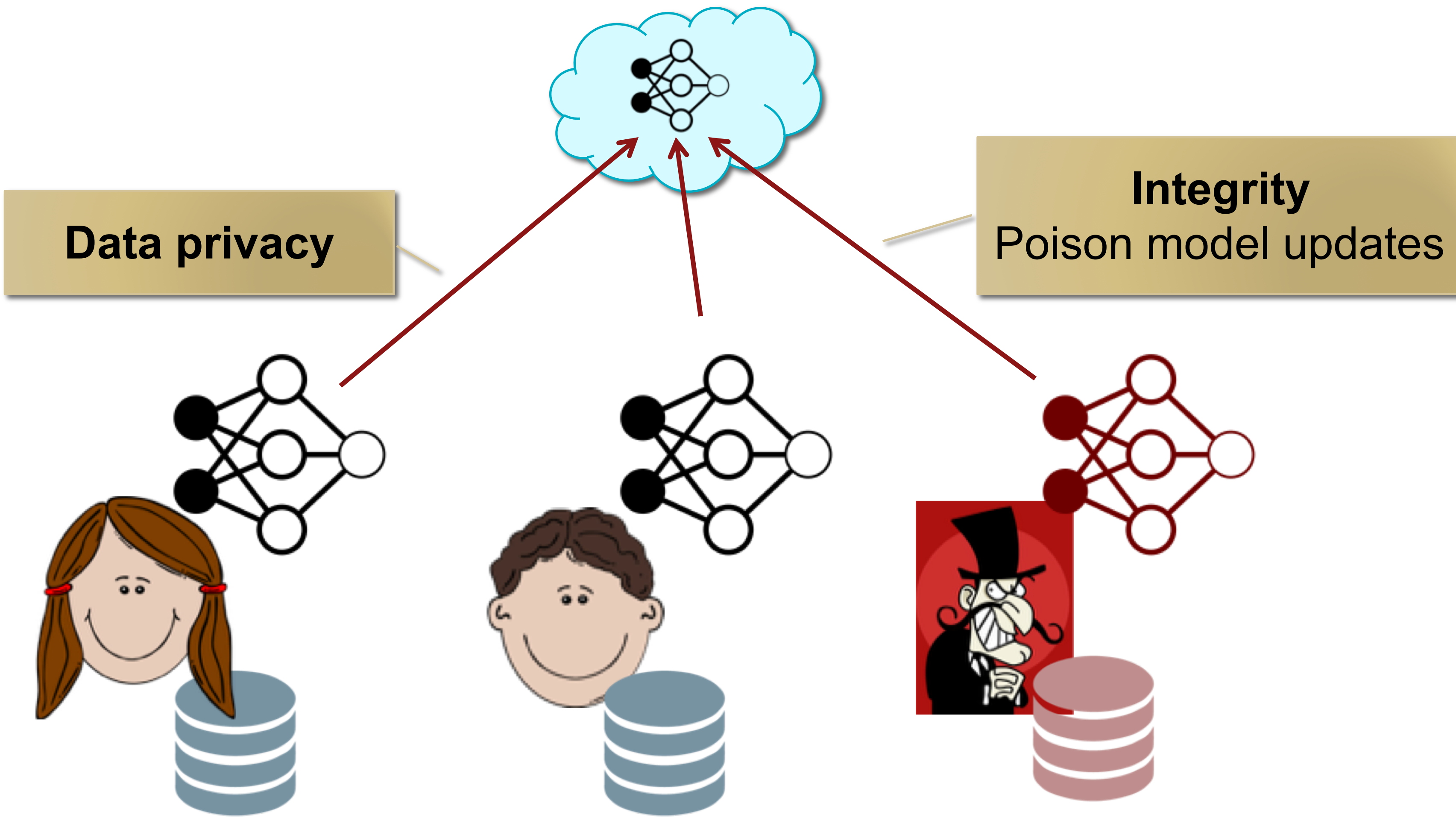


Integrity

- Model “downgrade”
- Disparate impact
- Other malicious tampering

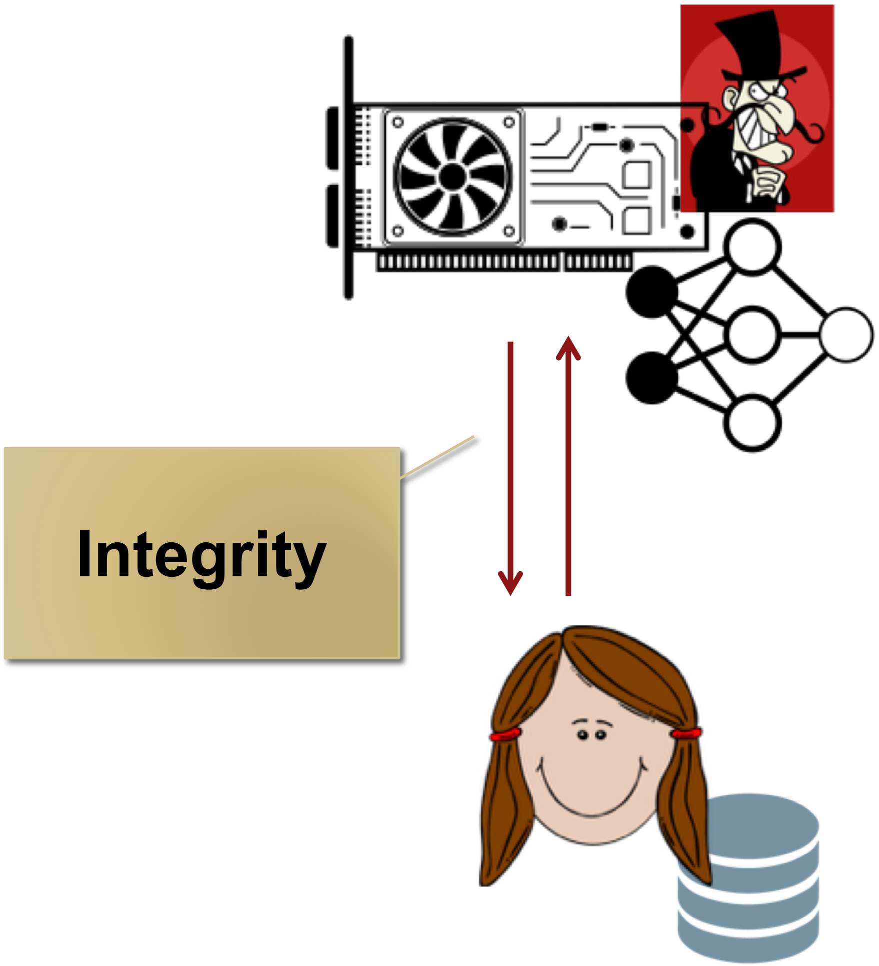
Trusted execution of ML: 3 motivating scenarios

2. Federated Learning



Trusted execution of ML: 3 motivating scenarios

3. Trojaned hardware
   (Verifiable ASICs model, Wahby et al.)



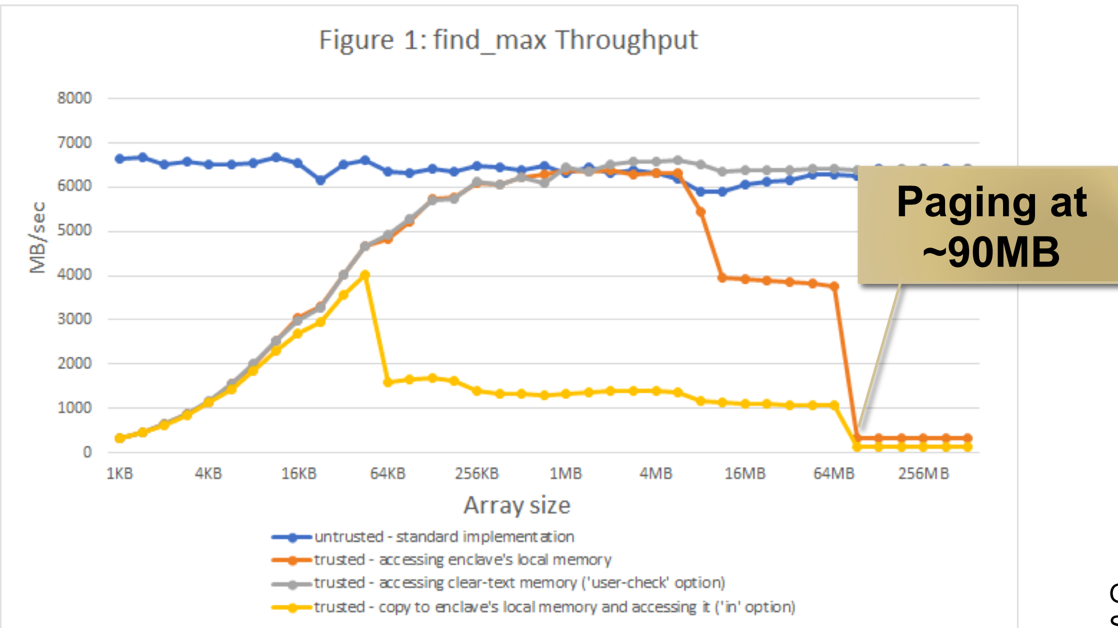
Solutions

- Cryptography
  1. **Outsourced ML**: FHE, MPC, (ZK) proof systems
  2. **Federated learning**: no countermeasure for poisoning…
  3. **Trojaned hardware**: some root of trust is needed

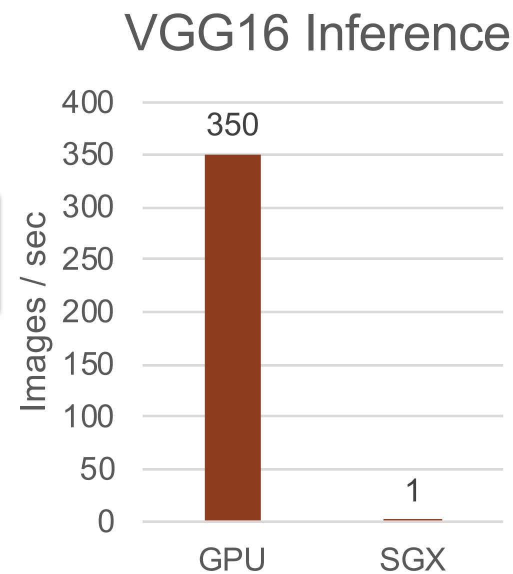
- Trusted Execution Environments (TEEs)
  1. **Outsourced ML**: isolated enclaves
  2. **Federated learning**: trusted sensors + isolated enclaves
  3. **Trojaned hardware:** fully trusted (but possibly slow) hardware

Trusted Execution: At what cost?

- Trusted ASICs (Wahby et al.): ~$10^8 \times$ worse than SOTA
- Intel SGX:



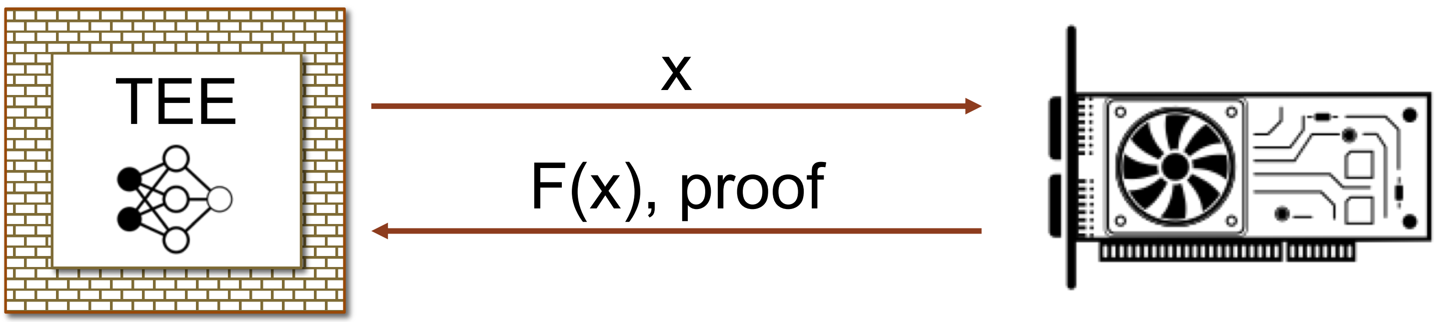
https://medium.com/@danny_harnik/impressions-of-intel-sgx-performance-22442093595a



GPU: Nvidia TITAN XP
SGX: Intel Core i7-6700 Skylake Single Core @ 3.40GHz

“How do we efficiently leverage TEEs for secure machine learning computations?”

<u>Idea</u>: outsource work to *collocated*, *faster* but *untrusted* device and verify results

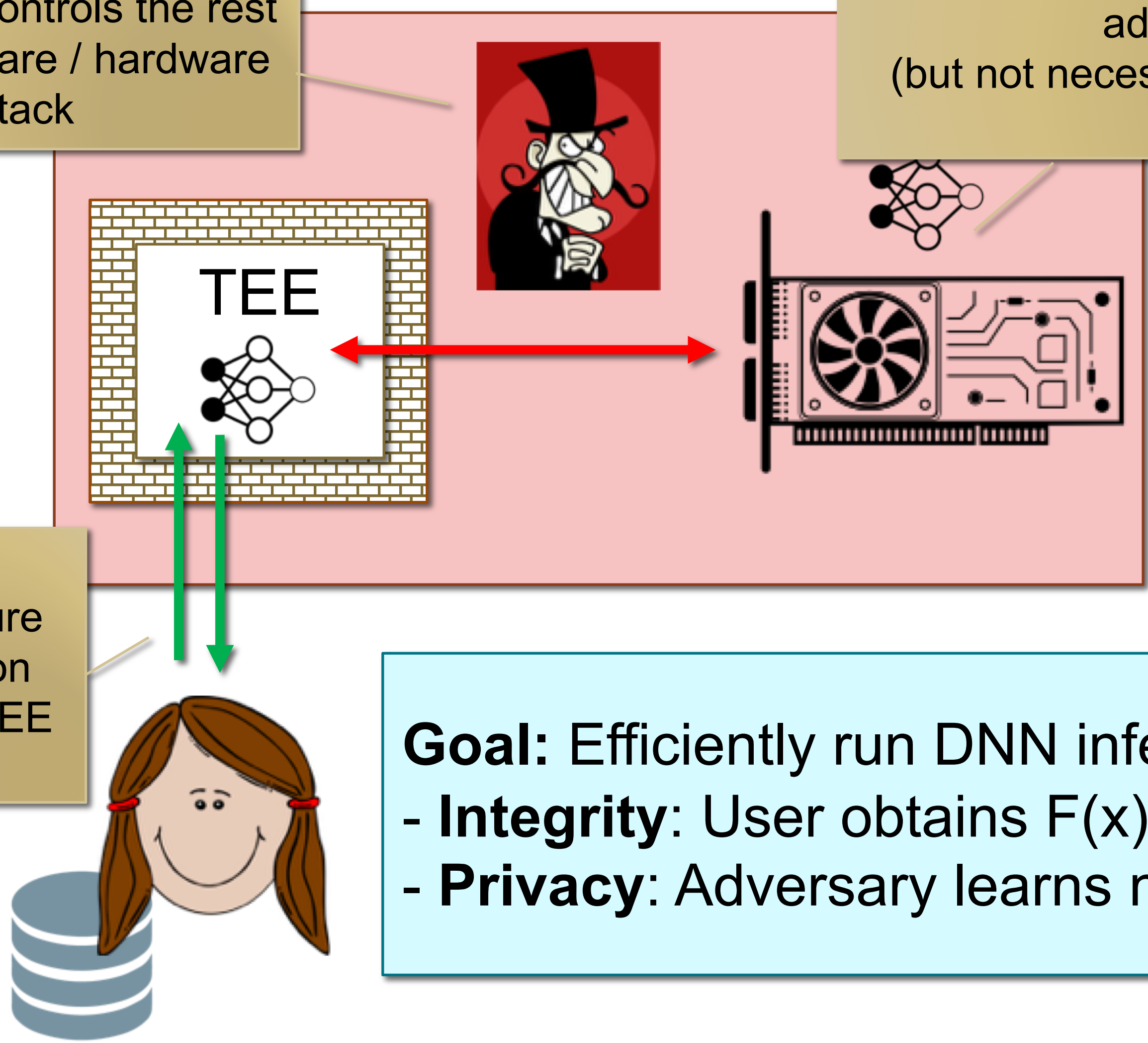


	Computations	Required gap	Privacy
Verifiable ASICs (Wahby et al., 2016)	Arithmetic circuits	~ 8 orders of magnitude	No
Slalom	**DNN inference**	**~ 1-2 orders**	**“Yes”**

Goal + threat model

Adversary controls the rest of the software / hardware stack

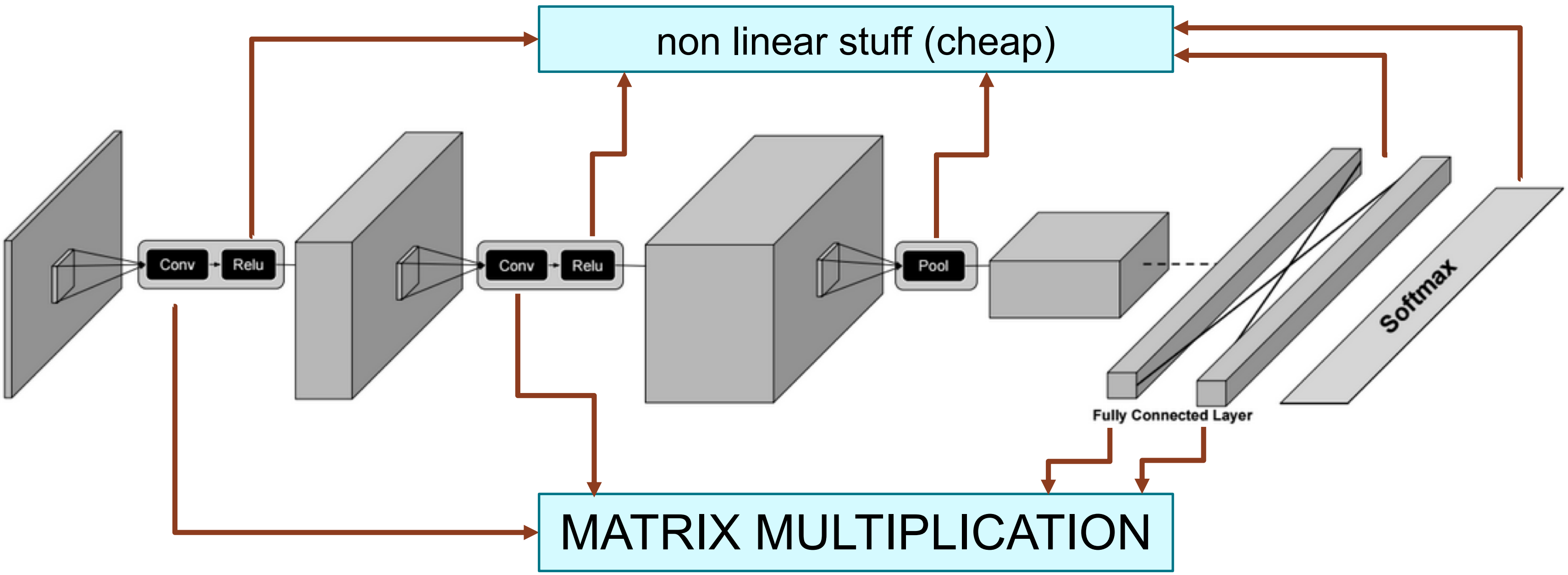
The model is known to the adversary
(but not necessarily to the client)



User has secure communication channel with TEE

Goal: Efficiently run DNN inference F(x)
- **Integrity**: User obtains F(x) or aborts
- **Privacy**: Adversary learns nothing about x

Bottlenecks in deep neural networks



Name ∇	Wall Duration ∇
NoOp	0.006 ms
Const	0.016 ms
_Arg	0.004 ms
VariableV2	0.077 ms
Identity	0.034 ms
Conv2D	372.828 ms
BiasAdd	5.637 ms
Relu	2.924 ms
MaxPool	2.495 ms
Totals	**384.021 ms**

~ 97%

VGG16 Inference on 1 CPU core

Outsourcing matrix multiplication: Freivald's algorithm

Input: $X \in \mathbb{F}^{n \times n}$, $W \in \mathbb{F}^{n \times n}$

DNN weights. Fixed at inference time

Direct Compute: $Z = X \cdot W$

$\approx n^3$ multiplications or $O(n^{2.81})$ with Strassen

Outsource + Verify:

- Sample $r \leftarrow \mathbb{F}^n$ uniformly at random
- Check: $Z \cdot r \stackrel{?}{=} X \cdot (W \cdot r)$
- Complexity: $\approx 3n^2$ multiplications
- Soundness: $1 / |\mathbb{F}|$ (boost by repeating)

Freivald variants for arbitrary linear operators

Linear operator: $z = F(x) = x \cdot A$

Matrix of size |x| × |z|

Vector of size |z|

Vector of size |x|

Batched verification:

Compute: $[z_1 \ldots z_B] = F\big([x_1 \ldots x_B]\big)$ $\Rightarrow$ B·cost(F) mults

Freivald: $r^T \cdot [z_1 \ldots z_B] =^? F\big(r^T \cdot [x_1 \ldots x_B]\big)$ $\Rightarrow$ B·(|x|+|z|) + cost(F) mults

With precomputation:

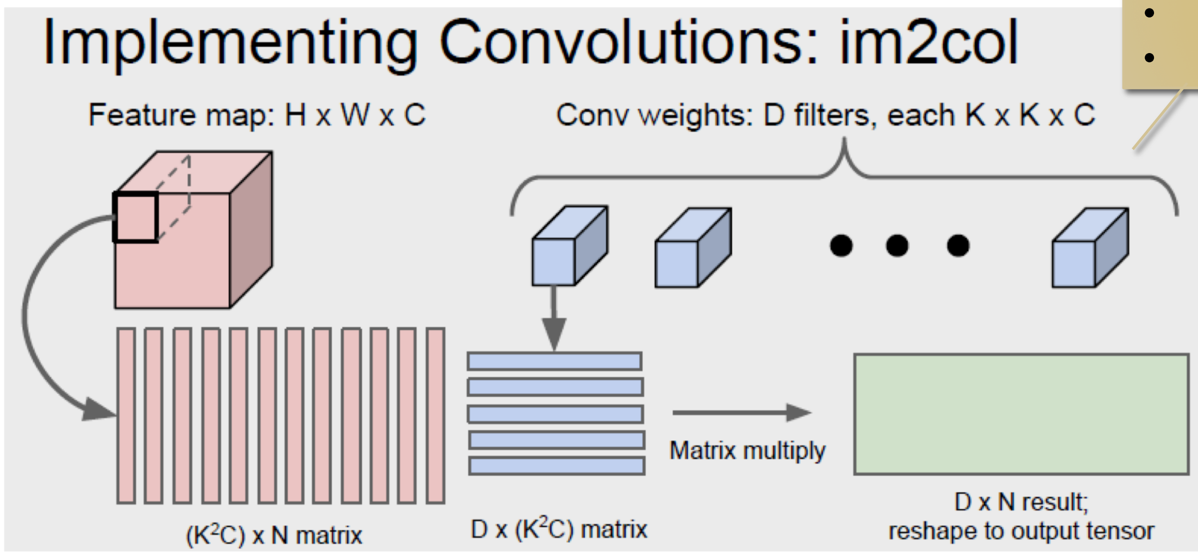
Precompute: $A' = A \cdot r = (\nabla_x F)(r)$

Freivald: $\langle z , r\rangle =^? \langle x , A'\rangle$ $\Rightarrow$ |x| + |z| mults

2 inner products!

Handling convolutions

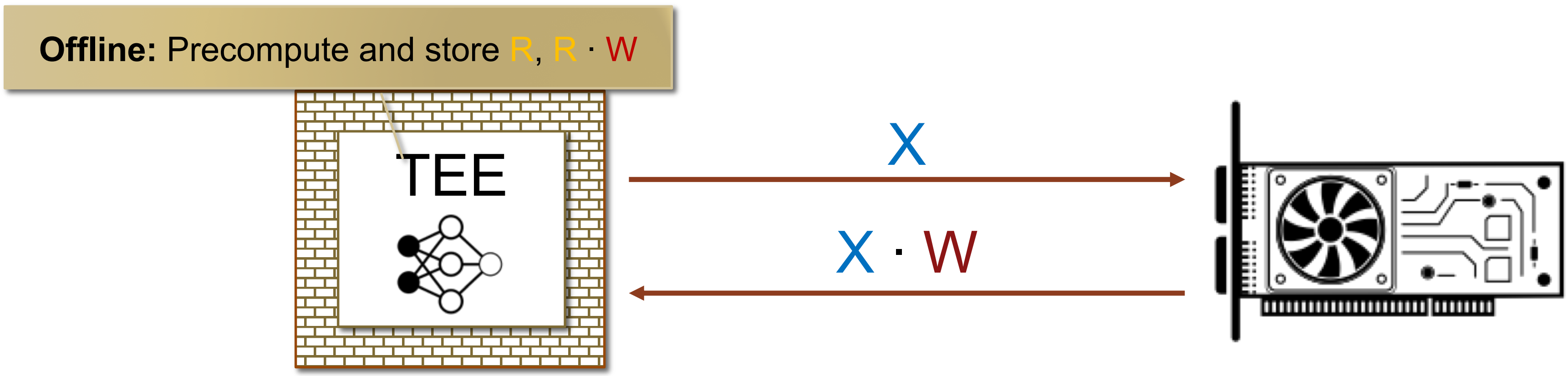
VGG16
- $K = 3$
- $3 \le C \le 512$
- $64 \le D \le 512$
- $14^2 \le N \le 224^2$



	Operation	Multiplications
Compute	$[z_1 \ldots z_B] = \text{im2col}([x_1 \ldots x_B]) * W$	$B{\cdot}H{\cdot}W{\cdot}K^2{\cdot}C{\cdot}D$
Batched verify	$r_1^T * [z_1 \ldots z_B] * r_2 =^? \text{im2col}(r_1 * ([x_1 \ldots x_B]) * (W * r_2)$	$B{\cdot}H{\cdot}W{\cdot}D + B{\cdot}H{\cdot}W{\cdot}C + K^2{\cdot}C{\cdot}D + H{\cdot}W{\cdot}K^2{\cdot}C$
Preprocessing	$\langle z, r\rangle =^? \langle(\nabla_x F)(r), x\rangle$	$B{\cdot}H{\cdot}W{\cdot}D + B{\cdot}H{\cdot}W{\cdot}C$

Preserving privacy

- Offline precomputation + online blinding

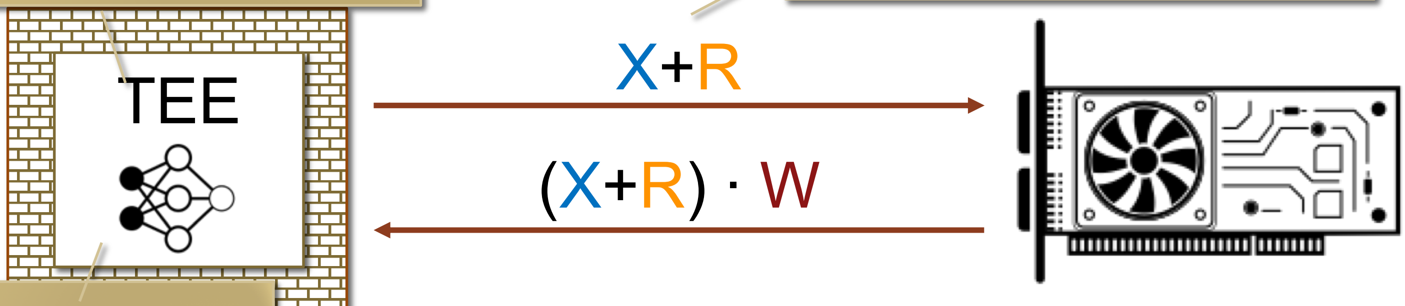


Preserving privacy

- Offline precomputation + online blinding

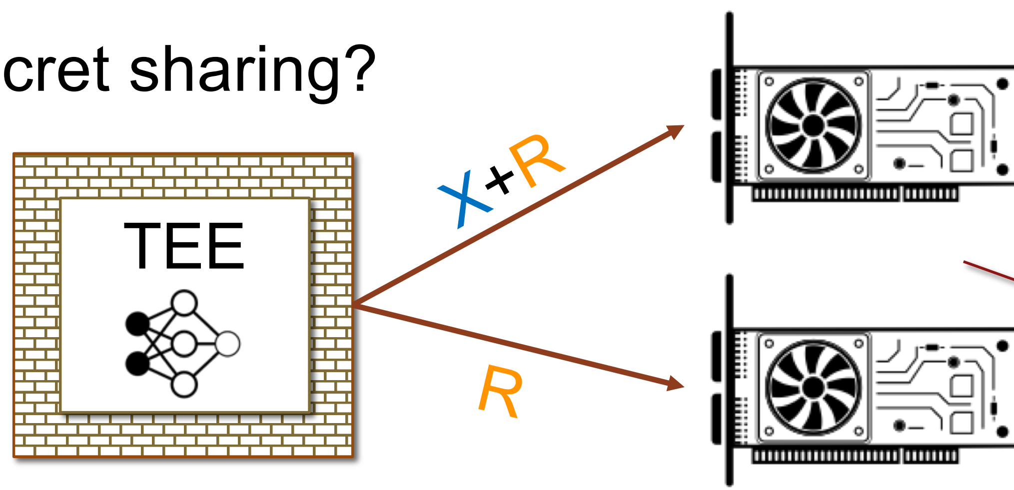
Offline: Precompute and store R, R · W

Online: “one-time-pad” over $\mathbb{F}$



Online: Unblind using R · W

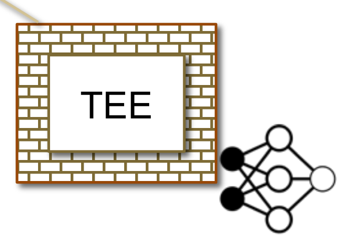
- Secret sharing?

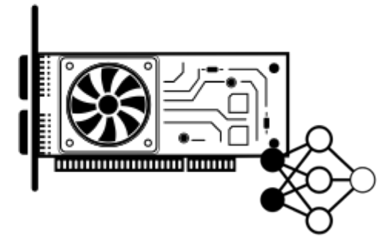


Can these devices be **“collocated”** yet **“non-colluding”** ?

Slalom Summary

Precompute and store $(R_i\,, R_i \cdot W_i)$





$X_1 + R_1$

$Z_1 = (X_1 + R_1) \cdot W_1$

1. $Z_1 = Z_1 - R_1 W_1$
2. Freivald check for (X_1, W_1, Z_1)
3. $X_2 = \sigma(Z_1)$

Arbitrary non-linearity

$X_2 + R_2$

$Z_2 = (X_2 + R_2) \cdot W_2$

…

Slalom (some details)

Quantization:

- DNNs are typically trained / evaluated in floating point
- Freivald / blinding require working over a ring/field $\mathbb{F}$
- **Quantize inputs & weights** and work mod p ($p < 2^{24}$)

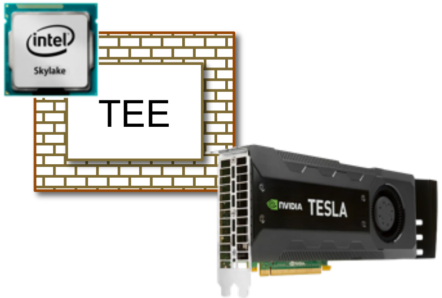
Integrity checks:

- Eval DNN on fast device and store inputs/outputs of all linear ops
  $\Longrightarrow$ **close to no prover overhead**
- Sample r from $\mathbb{F}$ and do Freivald check in double precision
  $\Longrightarrow$ **verifier complexity is at least |x| + |z| double muls per linear layer**

Blinding:

- Store unblinding factors R·W **encrypted in untrusted memory**
- In online phase, decrypt (and authenticate) R·W to unblind

Design & Evaluation



Implementation

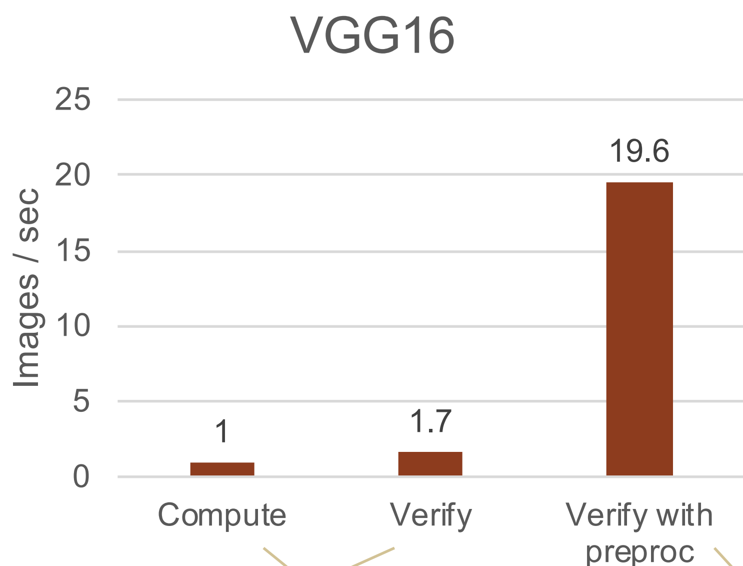
- TEE: Intel SGX ”Desktop” CPU (single thread)
- Untrusted device: Nvidia Tesla GPU
- Port of the Eigen linear algebra C++ library to SGX (used in e.g., TensorFlow)

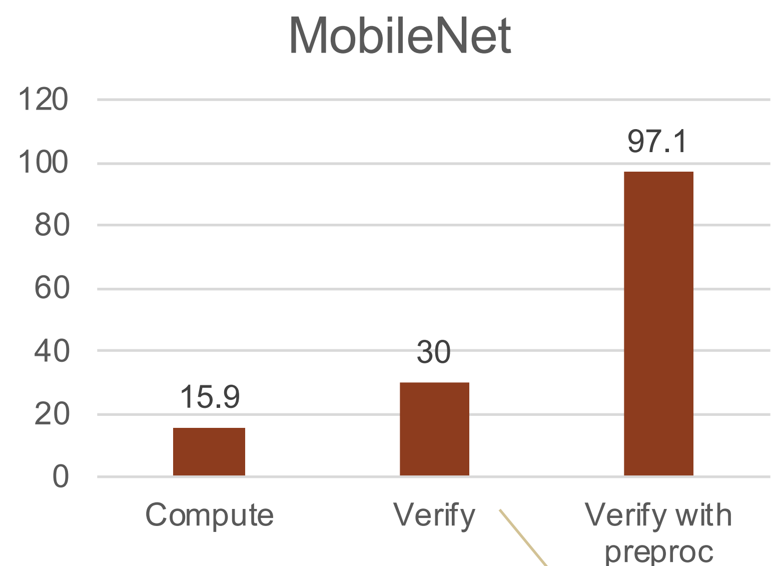
Workloads:

- Microbenchmarks (see paper)
- VGG16 (“beefy” canonical feedforward neural network)
- MobileNet (resource efficient DNN tailored for low-compute devices)
  - Variant 1: standard MobileNet (see paper)
  - Variant 2: No intermediate ReLU in separable convolutions (this talk)

Verifiable inference

MobileNet's weights are only ~10MB so they fit in the SGX cache



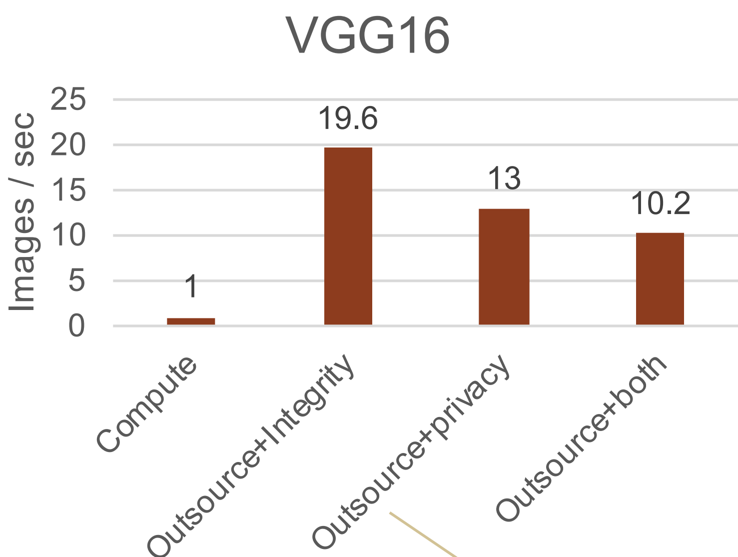


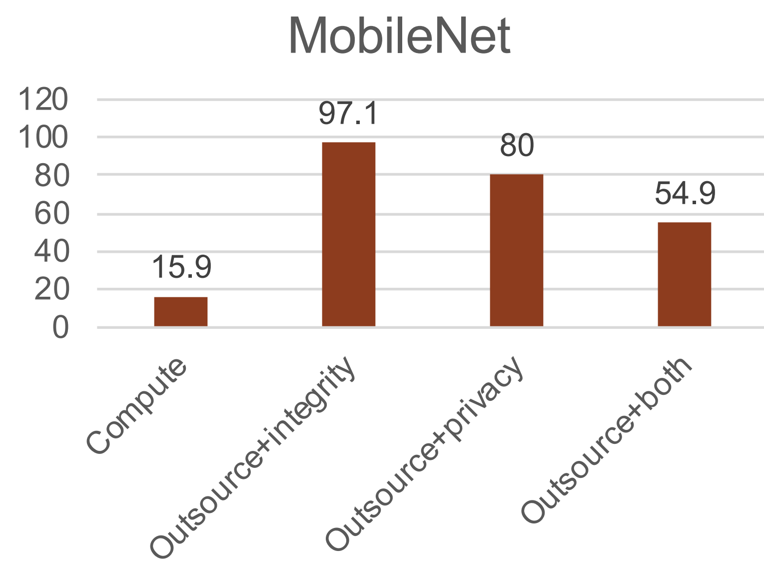
VGG16 weights take 500MB so SGX has to page weights in and out of memory => ~2-3x slowdown

Preprocessed weights W·r take up less memory and enable faster checks!

Difficult to get faster batched verification due to SGX memory limits

Verifiable and private inference





Extra Costs

- GPU has to operate in double precision
- Decrypt all unblinding factors R·W (AES-GCM)
- Regenerate all blinding factors R (PRG using AES)

Summary

- Large savings (6x – 20x) in outsourcing DNN inference while preserving **integrity**
  - Sufficient for some use-cases!
- More modest savings (3.5x – 10x) with **input privacy**
  - Requires preprocessing

Open questions

- What other problems are (concretely) easier to verify than to compute?
  - All NP complete problems (are those often outsourced?)
  - What about something in P?
    - Convex optimization
    - Other uses of matrix multiplication
    - Many graph problems (e.g., perfect matching)

- What about Slalom for verifiable / private training?
  - Quantization at training time is hard
  - Weights change so we can’t preprocess weights for Freivald’s check
  - We assume the model is known to the adversary
    (e.g., the cloud provider)