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

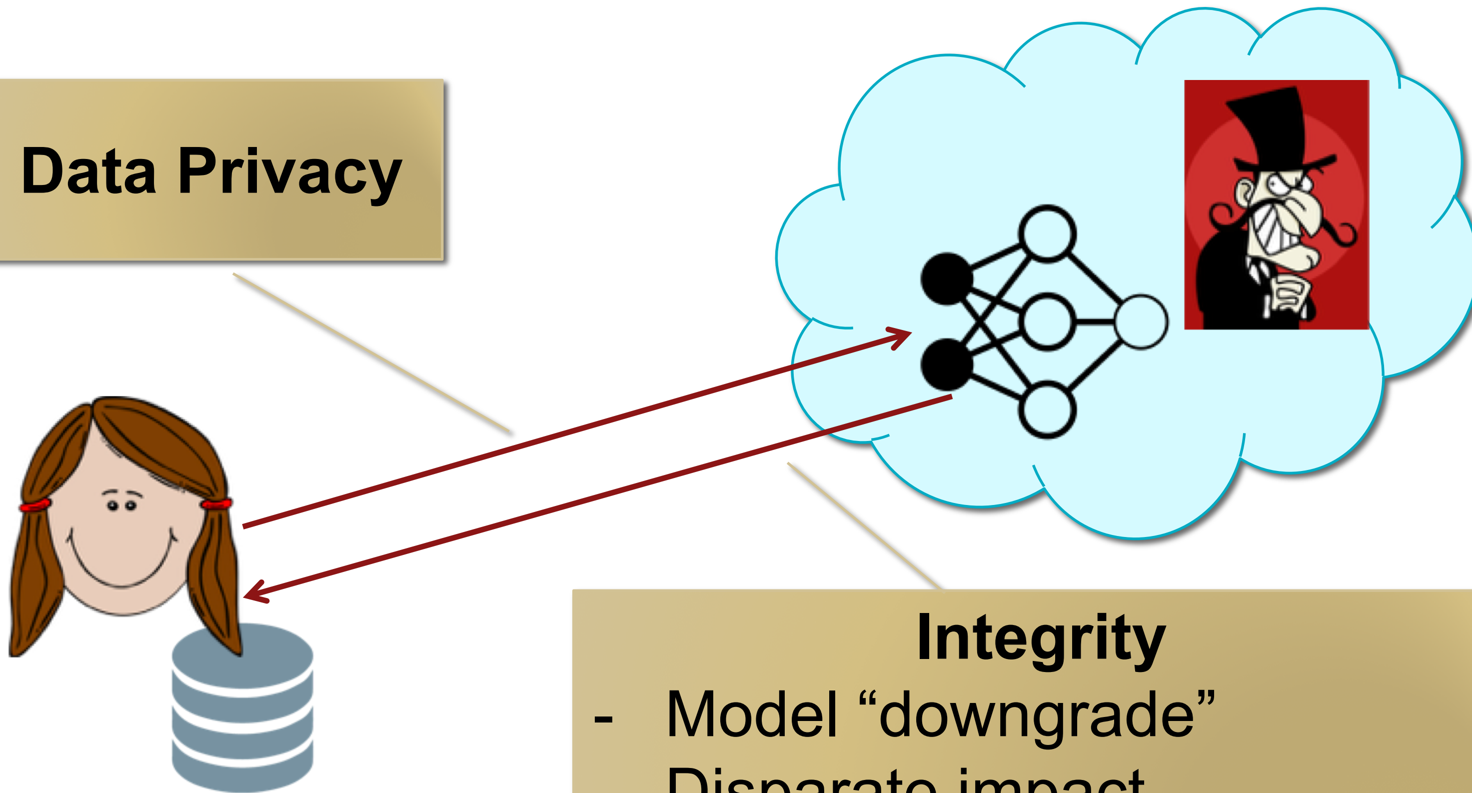
Florian Tramèr
(joint work with Dan Boneh)

Stanford security lunch – June 13th

Trusted execution of ML: 3 motivating scenarios

1. Outsourced ML

Data Privacy

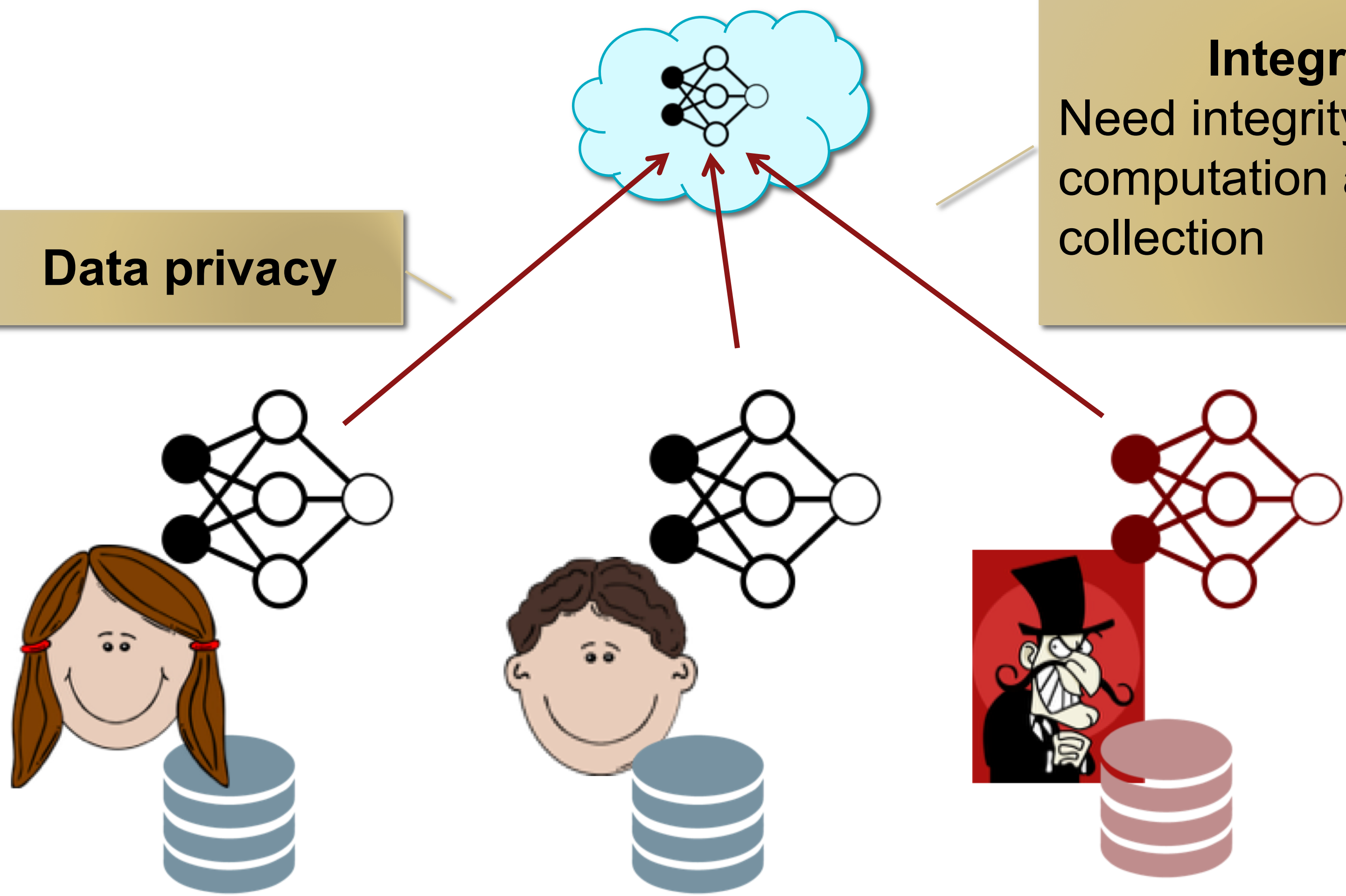


Integrity

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

Trusted execution of ML: 3 motivating scenarios

2. Federated Learning



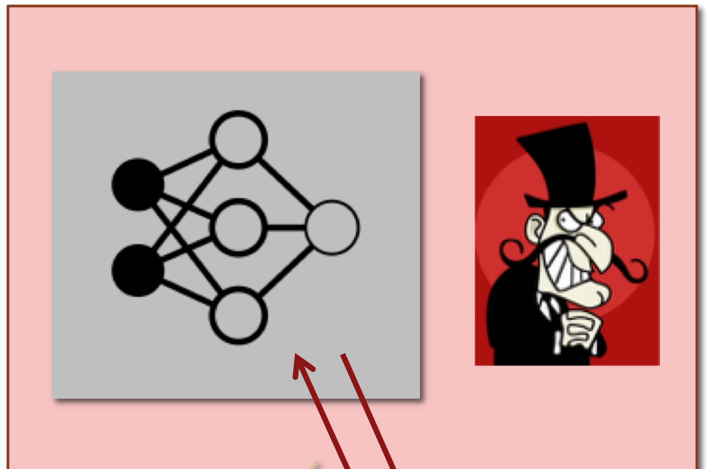
Data privacy

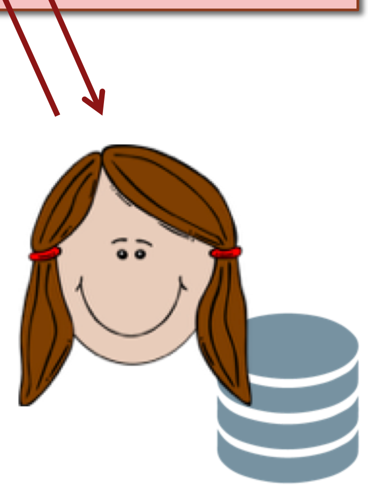
Integrity

Need integrity of computation and data collection

Trusted execution of ML: 3 motivating scenarios

3. Infected Hosts





Integrity

- Malware
- Trojaned hardware

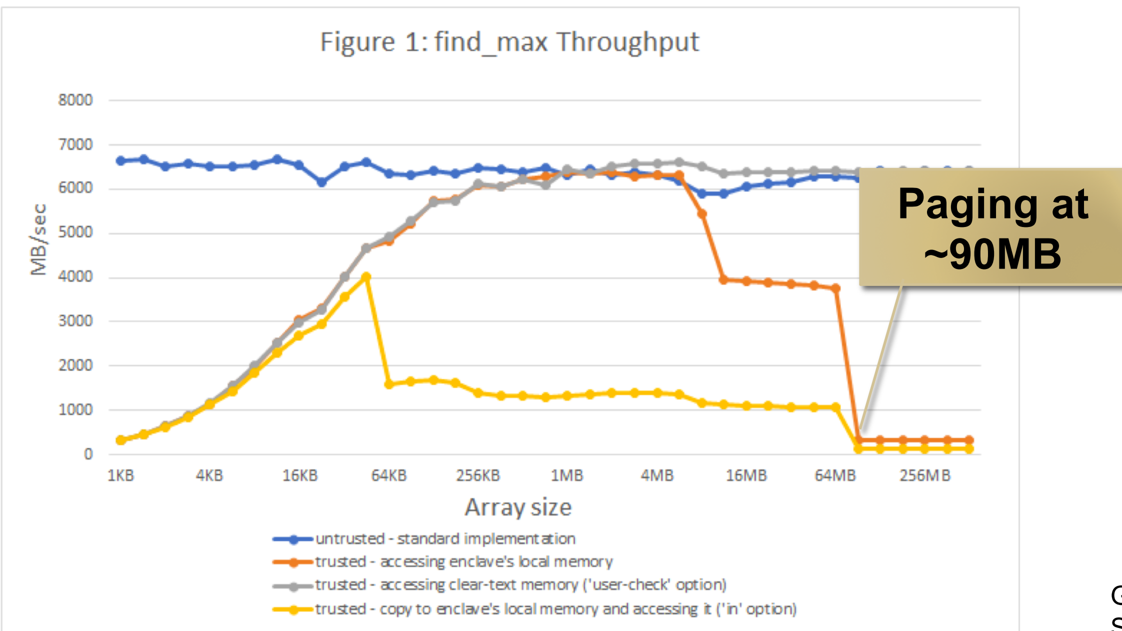
Solutions

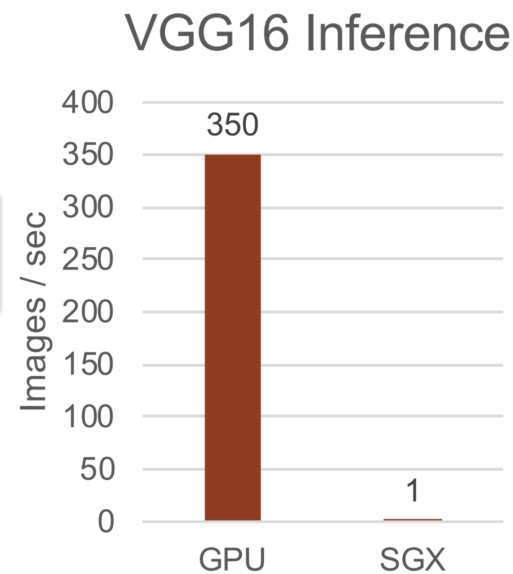
- Cryptography
  1. **Outsourced ML**: FHE, MPC, (ZK) proof systems
  2. **Federated learning**: no countermeasure for poisoning…
  3. **Infected hosts**: verifiable computation + some root of trust

- Trusted Execution Environments (TEEs)
  1. **Outsourced ML**: isolated enclaves
  2. **Federated learning**: trusted sensors + isolated enclaves
  3. **Infected hosts:** isolated enclaves / hardware from trusted manufacturer

Trusted Execution: At what cost?

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



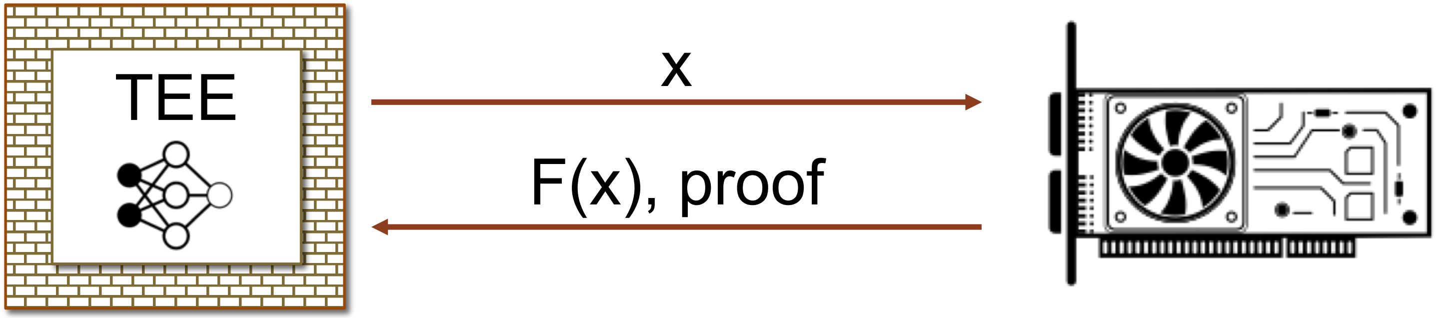


GPU: Nvidia TITAN XP
SGX: Intel Core i7-6700 Skylake Single Core @ 3.40GHz

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

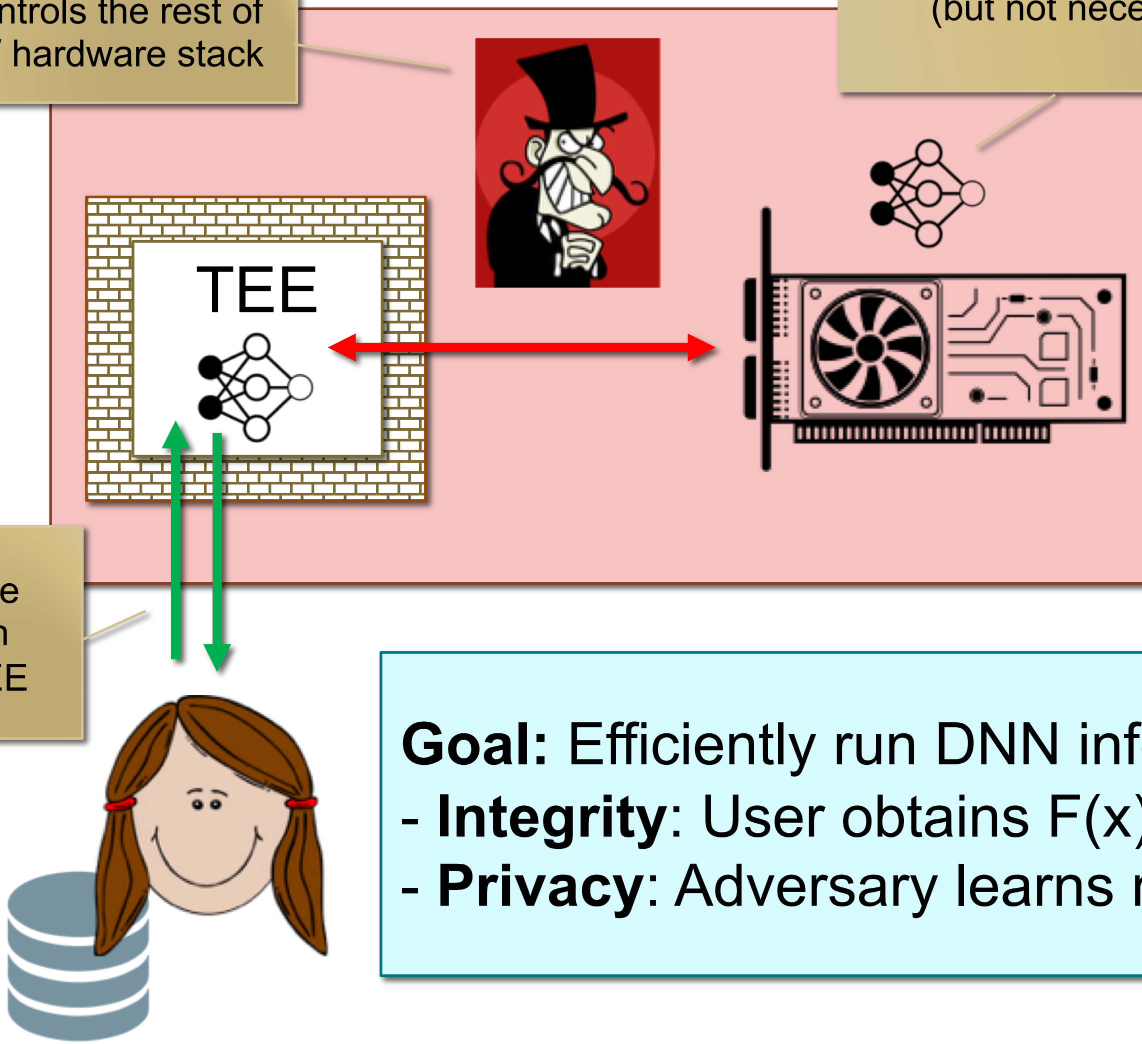
“How do we efficiently leverage TEEs for secure machine learning computations?”

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



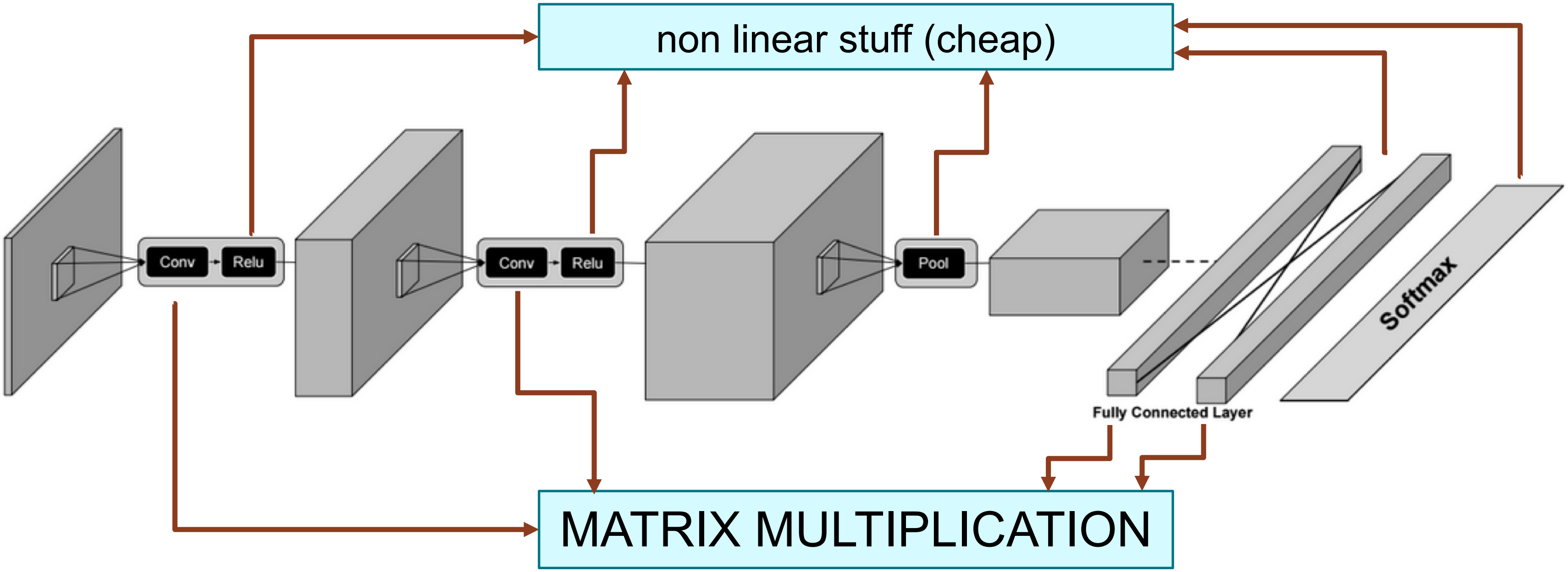
	Computations	Required gap	Privacy
Verifiable ASICs (Wahby et al., 2016)	Arithmetic circuits	~ 8 orders of magnitude	No
Slalom	**DNN inference**	**~ 1-2 orders**	**“Yes”**

Goal + threat model



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: $Z = X * W$

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

Outsource + Verify:

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

Batched and preprocessed verification

Some DNN layers are *not* matrix multiplications

E.g., a dense layer is a vector-matrix product, x*W

- Compute: $\approx n^2$
- Freivald: $\approx 3n^2$ ...

Verify a batch of inputs: $Z = [x_1, x_2, \ldots, x_B] * W$

- Compute: $\approx Bn^2$
- Freivald: $\approx Bn + 2n^2$

Preprocess learned weights: $W' = W*r$

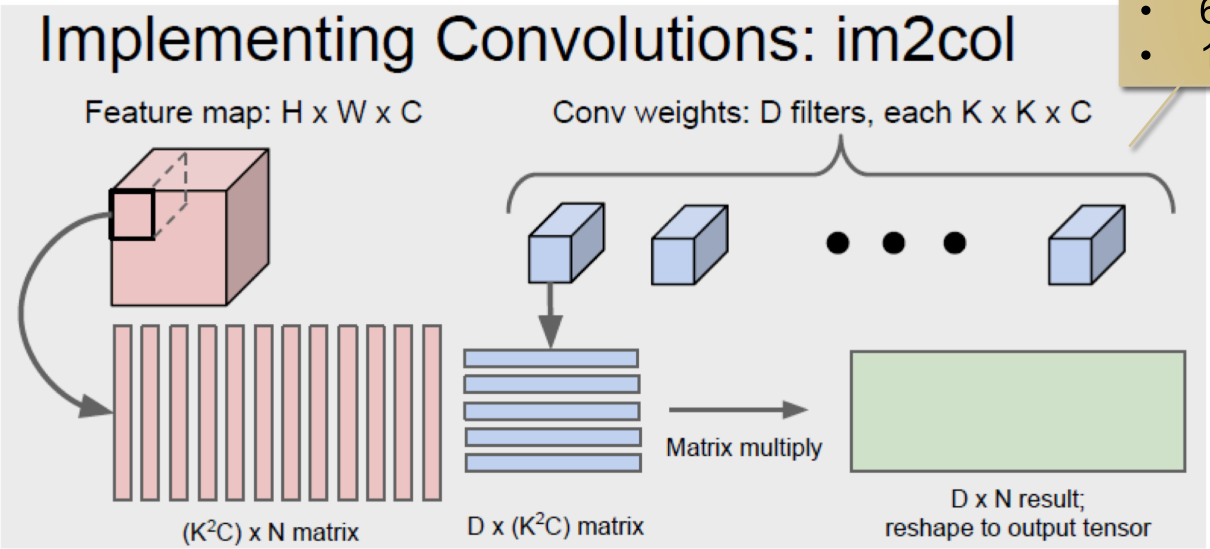
- Freivald: $\approx Bn + n^2$

The same randomness *r* can be reused for multiple checks if *r* is kept secret from the adversary

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	$Z = \text{im2col}([x_1, \ldots, x_B]) * W$	$B*N*K^2*C*D$
Batched verify	$r_1 * Z * r_2 = \text{im2col}(r_1 * X) * (W * r_2)$	$B*N*D + B*N*C + K^2*C*D + N*K^2*C$

Soundness: $2 / |\mathbb{F}|$

Savings even if B=1

Preprocessing for convolutions (or arbitrary linear ops)

Linear operator: $z = F_A(x) = x * A$

Matrix of size |x| × |z|

Vector of size |z|

Vector of size |x|

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

Easy to compute without making A explicit!

Check: $z * r = x * A'$

2 inner products!

Complexity: $|z| + |x|$ multiplications

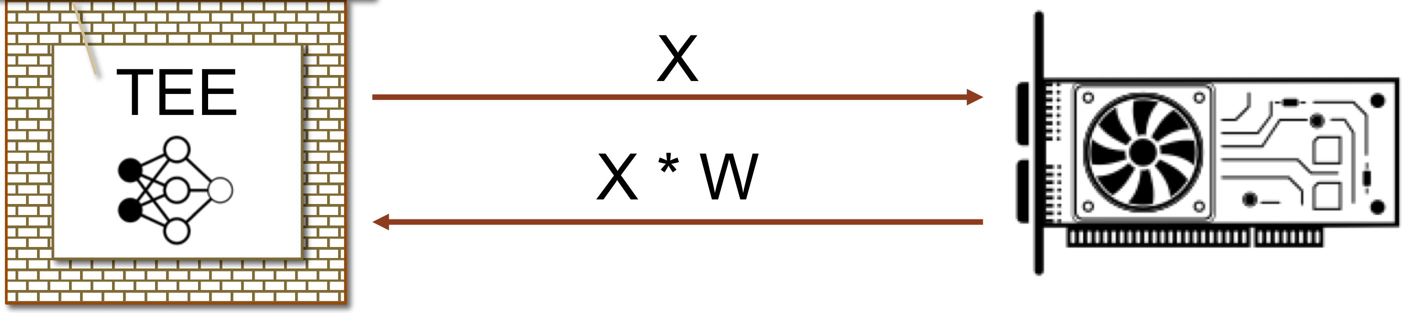
$|x| = B*N*C$
$|z| = B*N*D$

Convolutions	Multiplications
Compute	$B*N*K^2*C*D$
Batched verify	$B*N*D + B*N*C + K^2*C*D + N*K^2*C$
Preprocessed	$B*N*D + B*N*C$

Preserving privacy

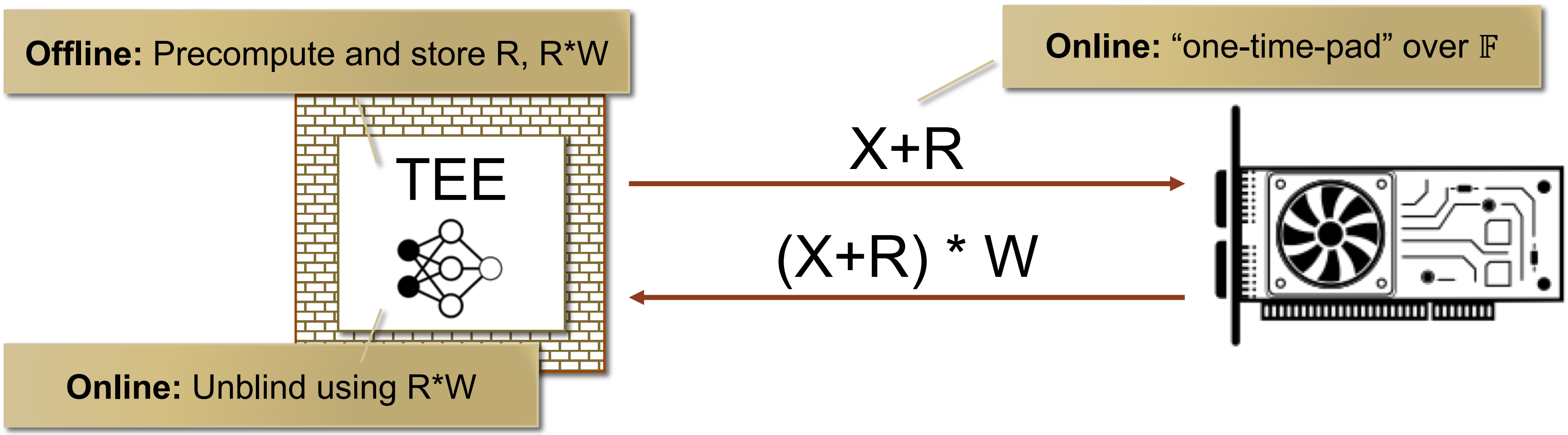
- Offline precomputation + online blinding

Offline: Precompute and store R, R*W

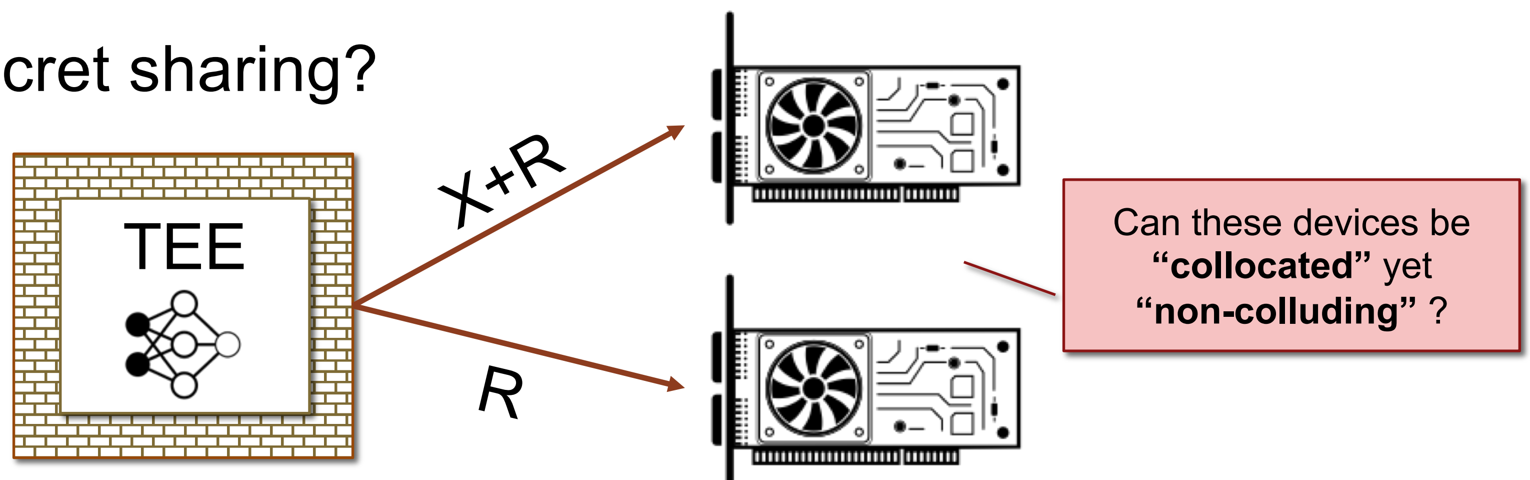


Preserving privacy

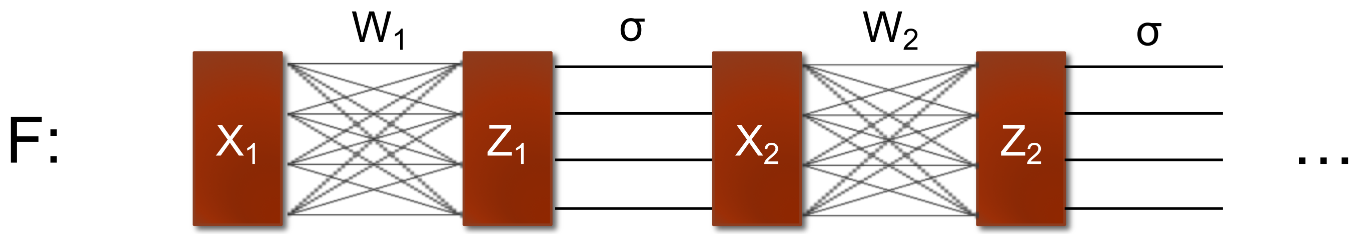
- Offline precomputation + online blinding

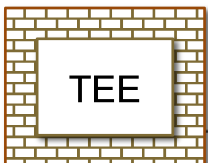


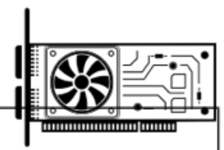
- Secret sharing?



Slalom







Slalom with integrity

$\mathcal{P}_T(X_1, F)$ $\qquad\qquad$ $\mathcal{P}_U(X_1, F)$

for $i = 1 \dots n$ **do**
$\quad Z_i = X_i W_i$
$\quad X_{i+1} = \sigma(Z_i)$

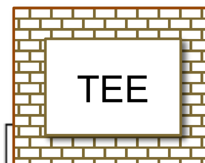
$\xleftarrow{Z_1 \dots Z_n}$

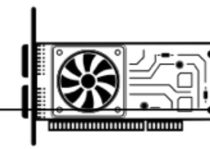
for $i = 1 \dots n$ **do**

$\quad$ `Freivald`(Z_i, X_i, W_i)
$\quad r \xleftarrow{R} \mathbb{F}^{h_{i+1}}$
$\quad$ assert $Z_i r = X_i(W_i r)$

$\quad X_{i+1} = \sigma(Z_i)$

return Z_n





Slalom with privacy

precomputed: $R_1 W_1, \dots, R_n W_n$

$\mathcal{P}_T(X_1, F)$ $\qquad\qquad$ $\mathcal{P}_U(F)$

for $i = 1 \dots n$ **do**
$\quad \tilde{X}_i = X_i + R_i$ $\qquad \xrightarrow{\tilde{X}_i}$

$\qquad \xleftarrow{\tilde{Z}_i}$ $\qquad \tilde{Z}_i = \tilde{X}_i W_i$

$\quad Z_i = \tilde{Z}_i - R_i W_i$
$\quad$ `Freivald`(Z_i, X_i, W_i)
$\quad X_{i+1} = \sigma(Z_i)$

return Z_n

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

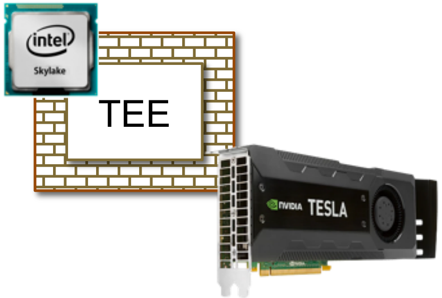
  $\Longrightarrow$ **close to no prover overhead**
- Sample r from $\mathbb{F}$ and do Freivald check in double precision

  $\Longrightarrow$ **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

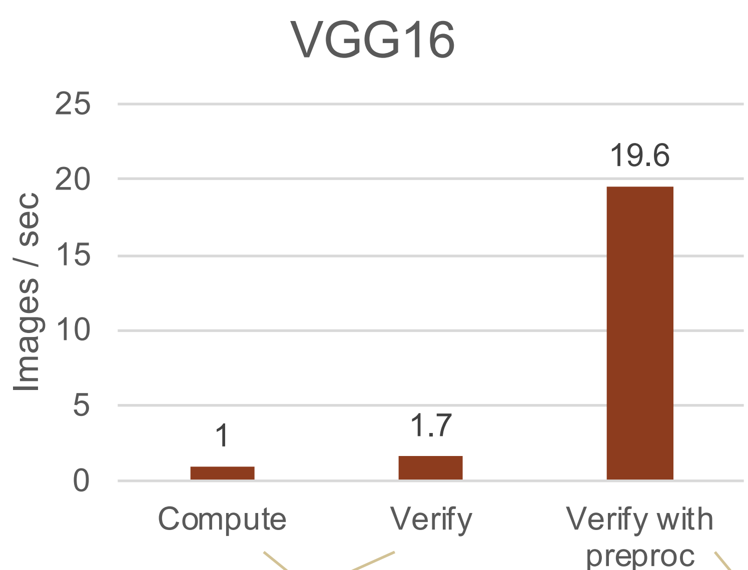
- TEE: Intel SGX ”Desktop” CPU (single thread)
- Untrusted device: Nvidia Tesla GPU
- Port of the Eigen linear algebra C++ library to SGX (used in e.g., TensorFlow)

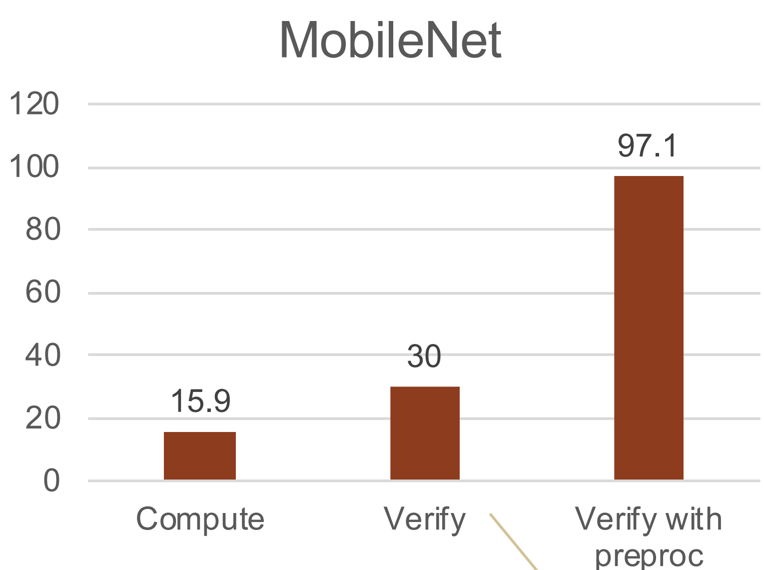
Workloads:

- 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



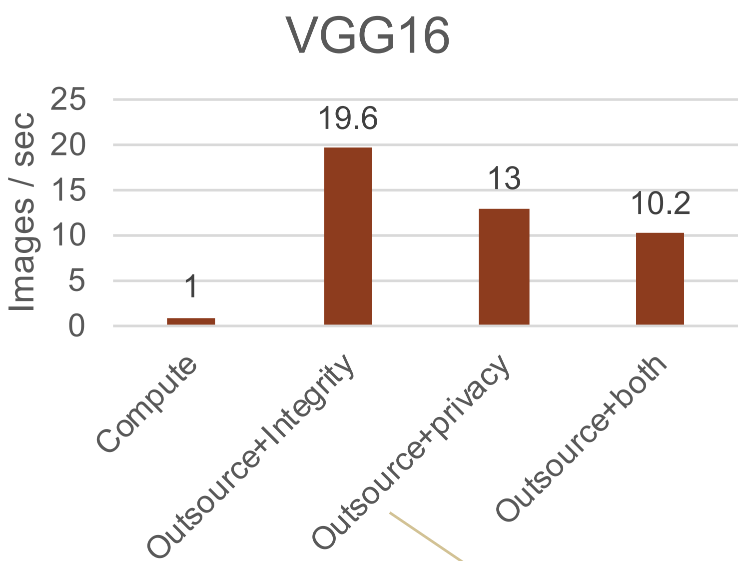


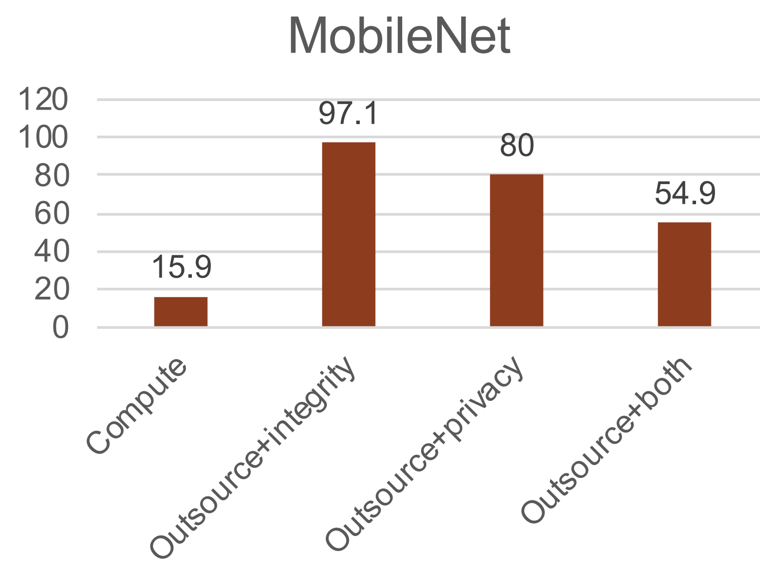
VGG16 weights take 500MB so SGX has to page weights in and out of memory => ~2-3x slowdown

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 really 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 W*r for Freivald's check
  - We assume the model is public