Aalto University

Trusted Computing and Analytics

N. Asokan
(joint work with Mika Juuti, Jian Liu, Samuel Marchal and Andrew Paverd)

Aalto University

Established in 2010, named in honour of ***Alvar Aalto***, the famous Finnish architect.

Science and art meet technology and business.

Promoting entrepreneurship

70 to 100

Companies are founded every year in our ecosystem

MIT Skolltech initiative rated Aalto's innovation ecosystem among the **top-5** rising stars in the world

Entrepreneurship is a more popular career option than ever – in the last four years, over **2 000** students have studied through the Aalto Ventures Program

50%

of Finnish startups that originate from universities come from the Aalto community

http://www.slush.org

About Us

Aalto Department of Computer Science

- **Features in top-100 in CS world rankings; AI, algorithms, security & privacy, …**
  http://cs.aalto.fi/

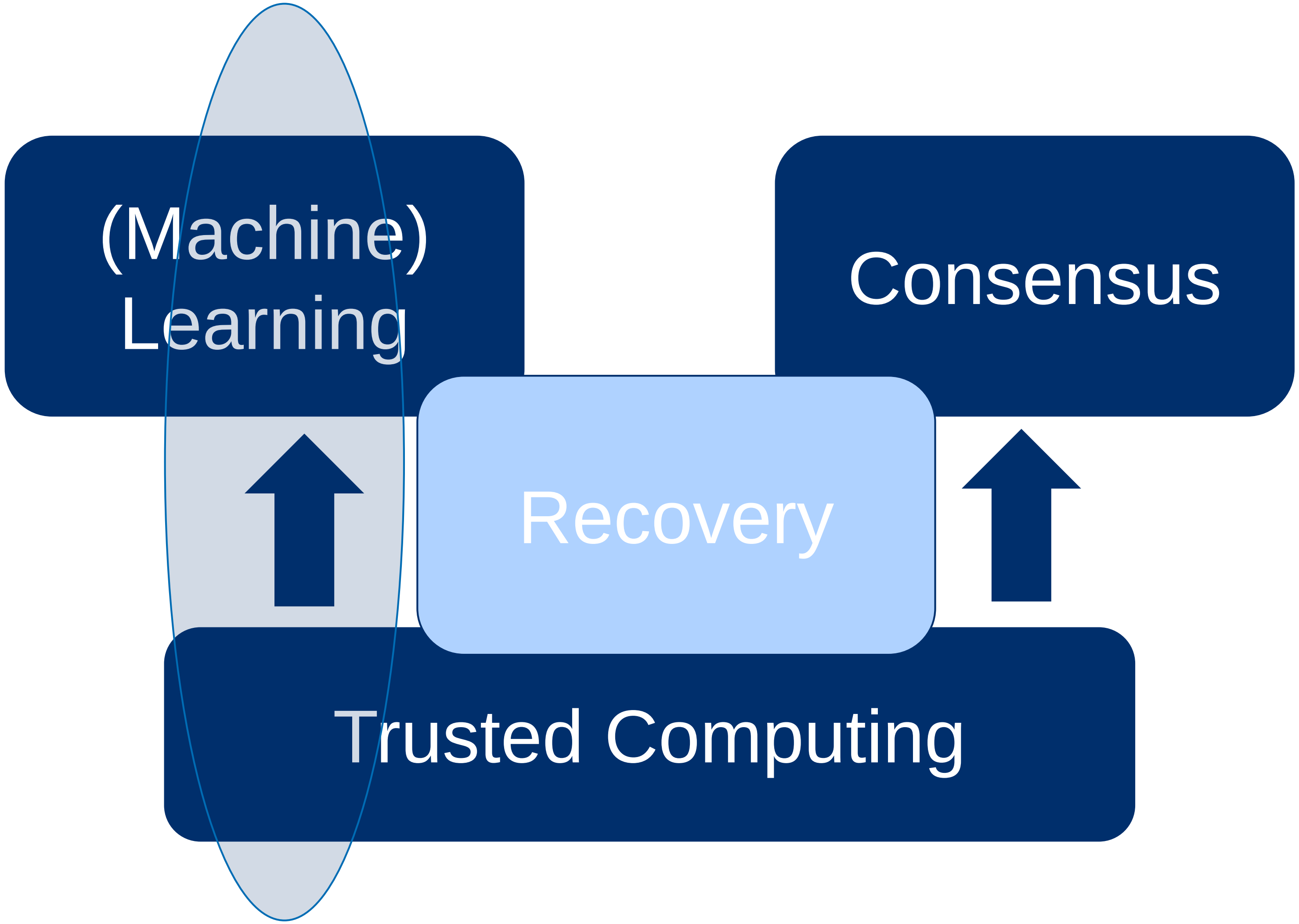
Secure Systems Group (Asokan)

- **10 senior researchers, 5-10 MSc students; Part of the previous ICRI-SC**
  http://cs.aalto.fi/secure_systems

Me

- **In academia since 2012; Earlier: industrial research labs (IBM ZRL and Nokia NRC)**
  Twitter: @nasokan, https://asokan.org/asokan/

Aalto University plans in CARS Lab



How do we evaluate ML-based systems?

Effectiveness of inference

- accuracy/score measures on held-out test set?

Performance

- inference speed and memory consumption?

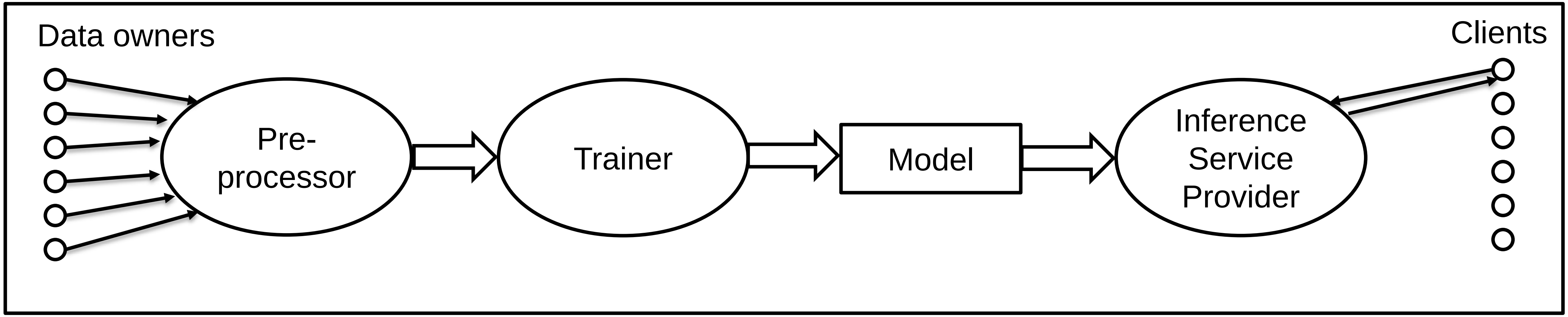
Hardware/software requirements

- e.g. memory/processor limitations, or specific software library?

Security & Privacy

Meeting requirements in the presence of an adversary

Machine learning pipeline

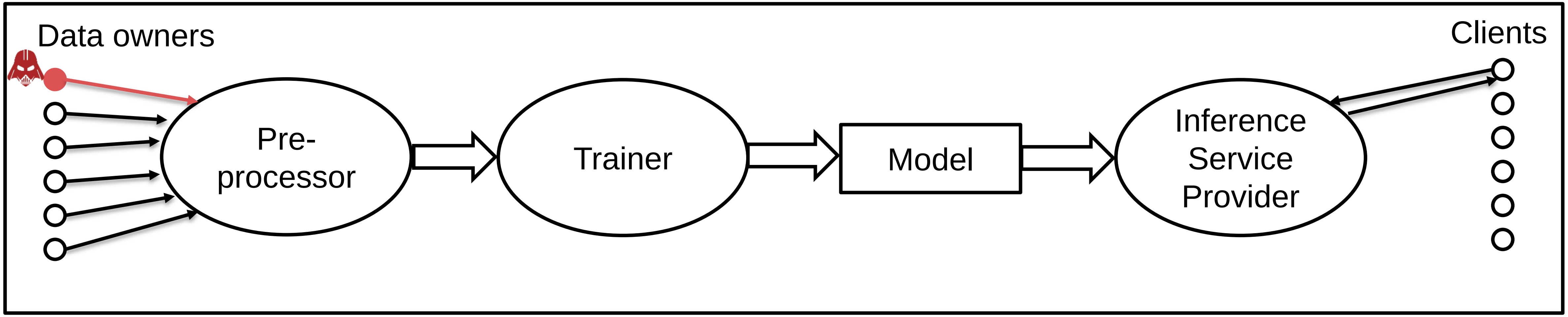


Legend

entities

components

1. Malicious data owners

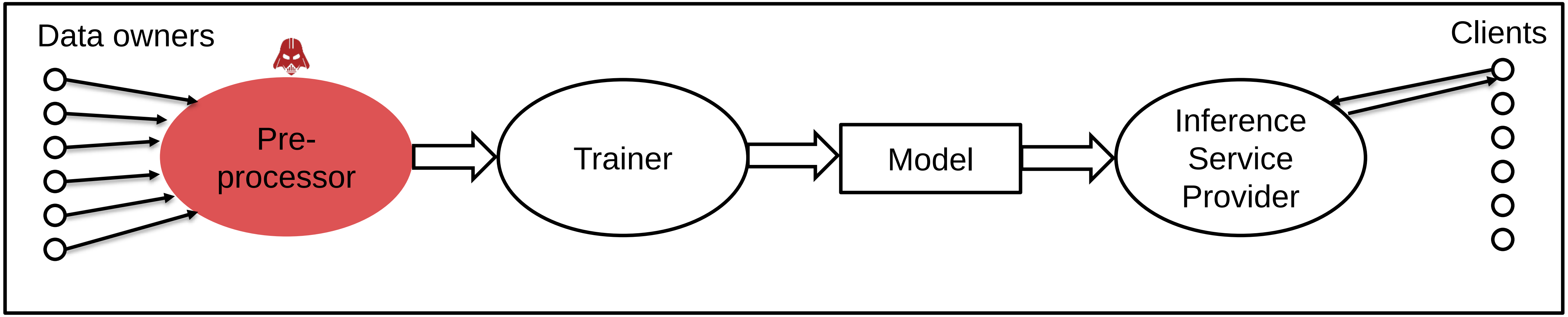


Attack target	Risk	Remedies
Model (integrity)	Data poisoning [1, 2]	Access control Robust estimators Active learning (human-in-the-loop learning) Outlier removal / normality models

[1] https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist

[2] https://wikipedia.org/wiki/Naive_Bayes_spam_filtering#Disadvantages

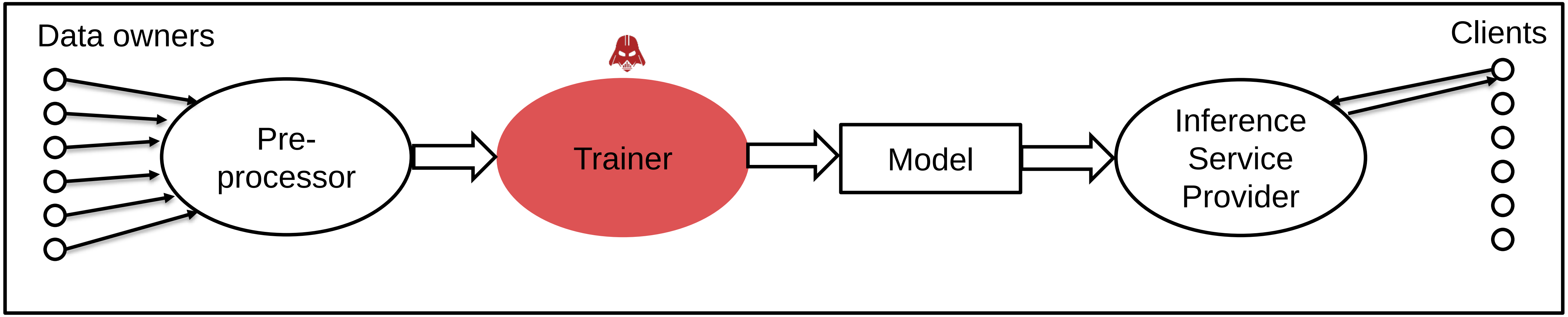
2. Malicious pre-processor



Attack target	Risk	Remedies
Model (integrity)	Data poisoning	Access control Robust estimators Active learning (human-in-the-loop learning) Outlier removal / normality models
Training data (confidentiality)	Unauthorized data use (e.g. profiling)	Adding noise (e.g. differential privacy) Oblivious aggregation [1] (e.g., homomorphic encryption)

[1] Heikkila et al. "Differentially Private Bayesian Learning on Distributed Data", NIPS'17

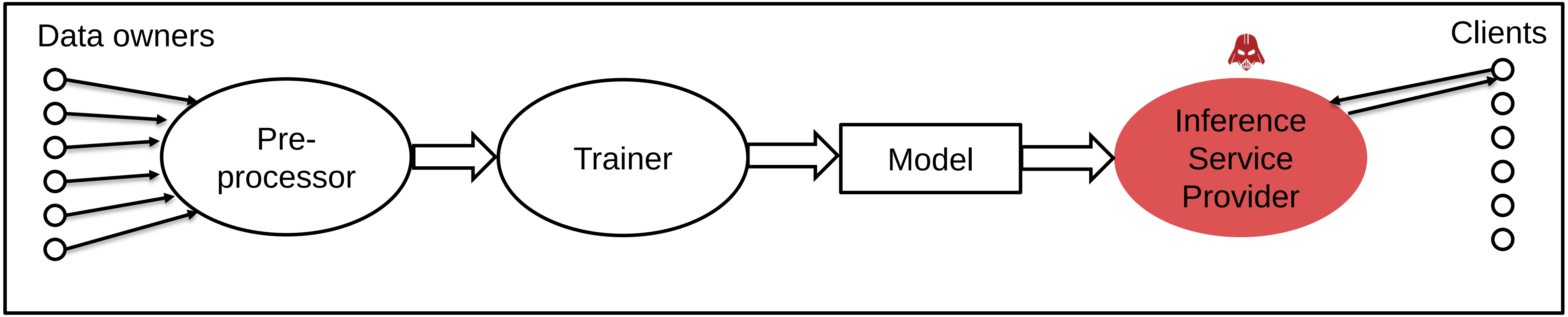
3. Malicious model trainer



Attack target	Risk	Remedies
Training data (confidentiality)	Unauthorized data use (e.g. profiling)	Oblivious training (learning with encrypted data) [1]

[1] Graepel et al. "ML Confidential", ICISC'12

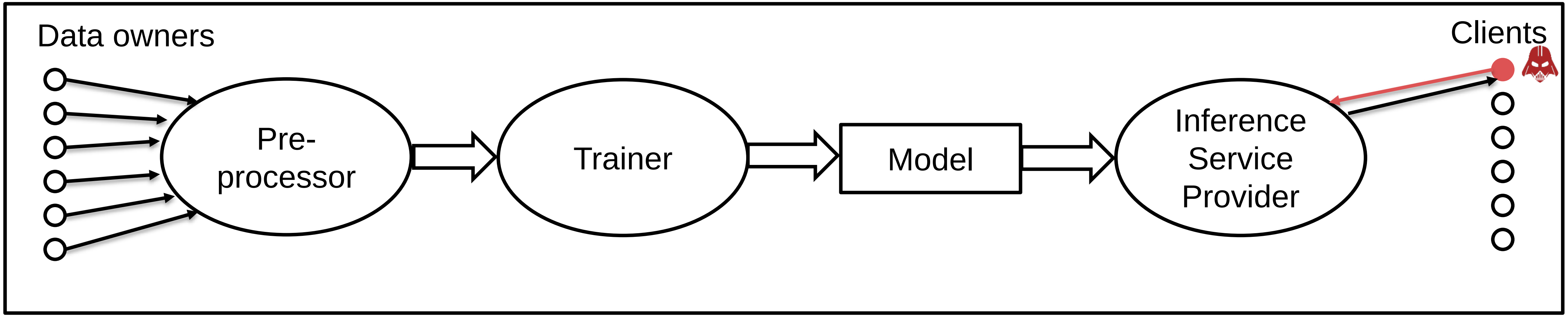
4. Malicious inference service provider



Attack target	Risk	Remedies
Inference queries/results (confidentiality)	Unauthorized data use (e.g. profiling)	Oblivious inference [1,2,3]

[1] Gilad-Bachrach et al. ”CryptoNets”, ICML’16
[2] Mohassel et al. ”SecureML”, IEEE S&P’17
[3] Liu et al. ”MiniONN”, ACM CCS’17

5. Malicious client



Attack target	Risk	Remedies
Training data (confidentiality)	Membership inference Model inversion	Minimize information leakage in responses Differential privacy
Model (confidentiality)	Model theft [1]	Minimize information leakage in responses Normality model for client queries Adaptive responses to client requests
Model (integrity)	Model evasion [2]	

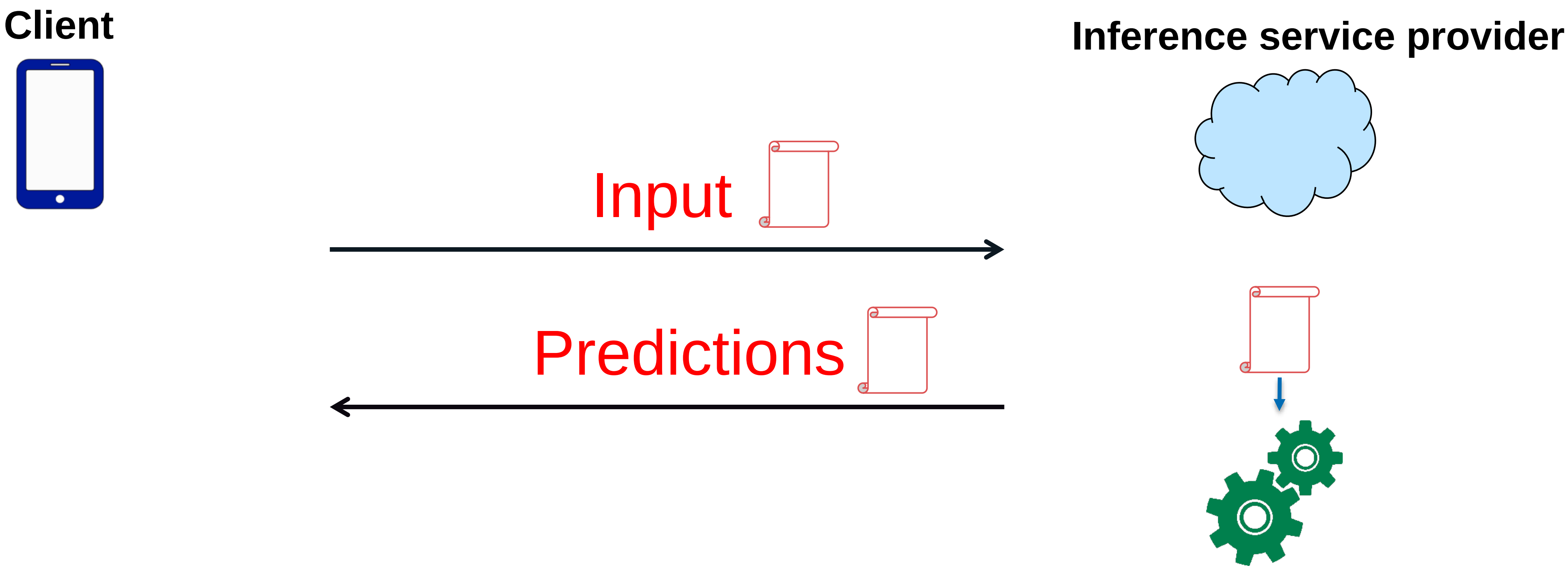
[1] Tramer et al, "Stealing ML models via prediction APIs", UsenixSEC'16
[2] Dang et al, "Evading Classifiers by Morphing in the Dark", CCS'17

Oblivious Neural Network Predictions via MiniONN Transformations

Jian Liu, Mika Juuti, Yao Lu, N. Asokan

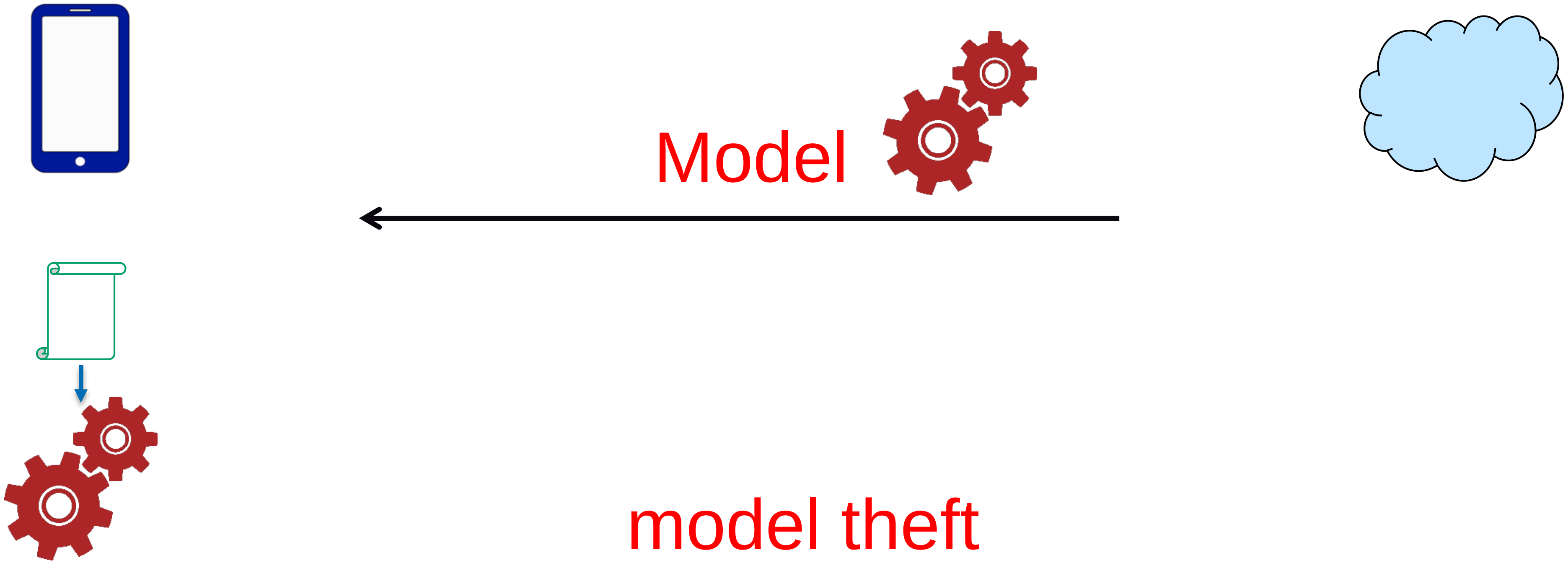
ACM CCS 2017

Machine learning as a service (MLaaS)



violation of clients' privacy

Running predictions on client-side



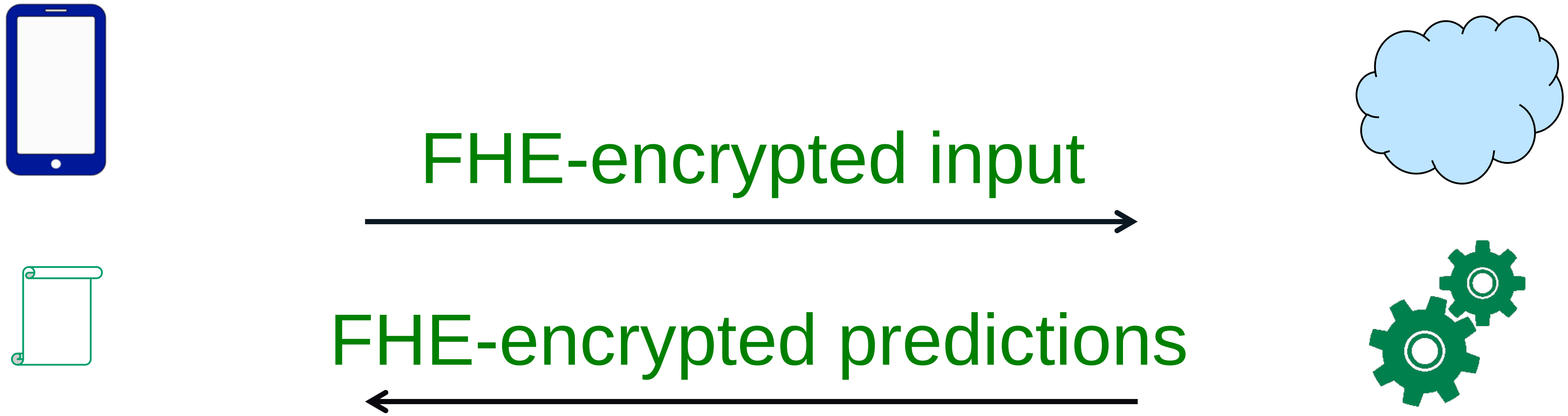
model theft
evasion
model inversion

Oblivious Neural Networks (ONN)

Given a neural network, is it possible to make it oblivious?

- server learns nothing about clients' input;
- clients learn “nothing” about the model.

Example: CryptoNets

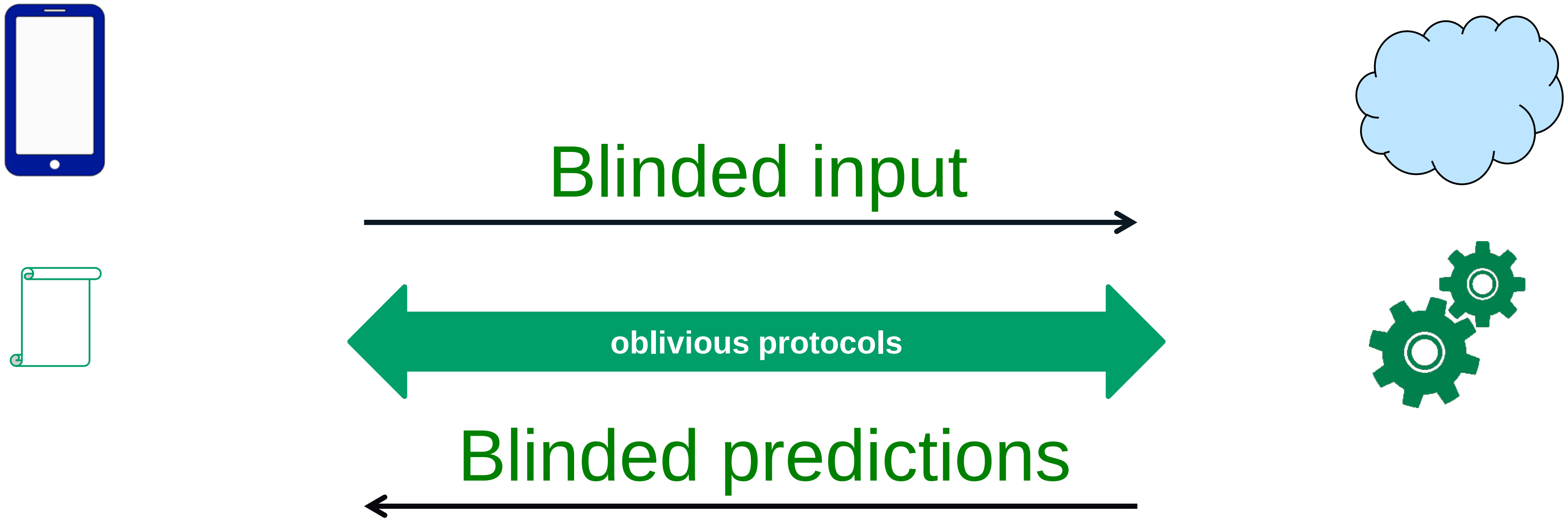


- Uses fully homomorphic encryption (FHE)
- High throughput for batch queries from same client
- High overhead for single queries: 297.5s and 372MB
- Only supports low-degree polynomials

[GDLLNW16] CryptoNets, ICML 2016

MiniONN: Overview

By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=54119040



Lightweight primitives:

- Additively homomorphic encryption (with SIMD)
- Secure two-party computation

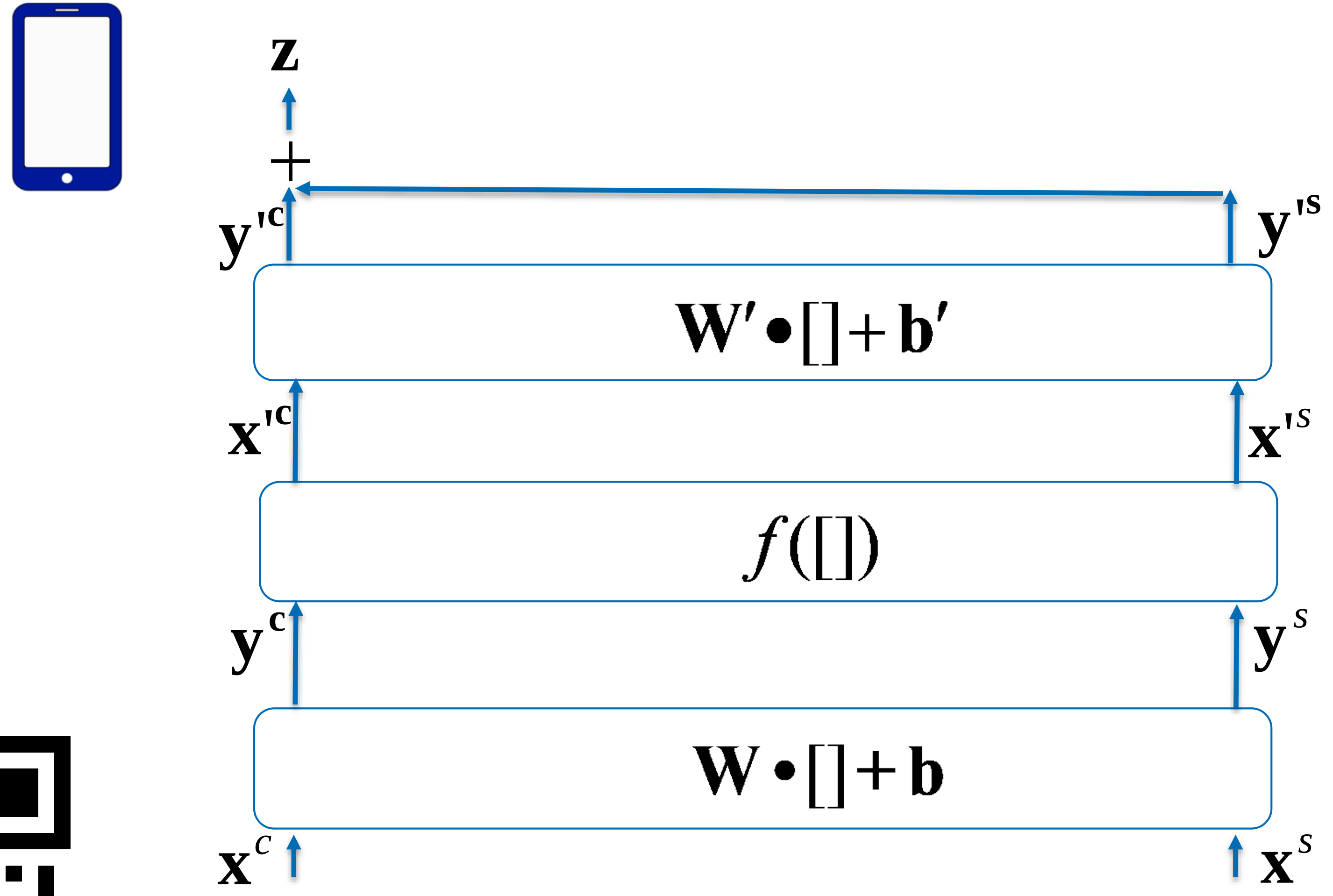
Example $\mathbf{z} = \mathbf{W}' \bullet f(\mathbf{W} \bullet \mathbf{x} + \mathbf{b}) + \mathbf{b}'$

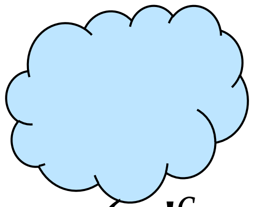
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \mathbf{W}' = \begin{bmatrix} w'_{1,1} & w'_{1,2} \\ w'_{2,1} & w'_{2,2} \end{bmatrix}, \mathbf{b}' = \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix}$$

$$\mathbf{x} \rightarrow \boxed{\mathbf{W} \bullet [\,] + \mathbf{b}} \xrightarrow{\mathbf{y}} \boxed{f([\,])} \xrightarrow{\mathbf{x}'} \boxed{\mathbf{W}' \bullet [\,] + \mathbf{b}'} \rightarrow \mathbf{z}$$

All operations are in a finite field

Core idea: use secret sharing for oblivious computation





$(\mathbf{y}'^{C} + \mathbf{y}'^{S} = \mathbf{y}')$

$(\mathbf{x}'^{C} + \mathbf{x}'^{S} = \mathbf{x}')$

$(\mathbf{y}^{C} + \mathbf{y}^{S} = \mathbf{y})$

$(\mathbf{x}^{C} + \mathbf{x}^{S} = \mathbf{x})$

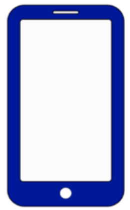
For a layer with input $\mathbf{x}$ and output $\mathbf{y}$, at the beginning:
arrange for client & server to have shares $\mathbf{x}^{c}$ and $\mathbf{x}^{s}$ s.t $\mathbf{x}^{s} + \mathbf{x}^{c} = \mathbf{x}$

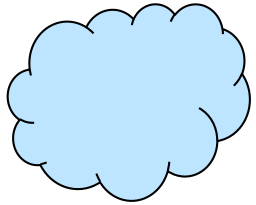
At the end of **oblivious processing**:
client & server to have shares $\mathbf{y}^{c}$ and $\mathbf{y}^{s}$ s.t $\mathbf{y}^{s} + \mathbf{y}^{c} = \mathbf{y}$

https://eprint.iacr.org/2017/452

Skip to performance

Secret sharing initial input $\mathbf{x}$





$$x_1^c, x_2^c \xleftarrow{\$} Z_N$$

$$\xrightarrow{x_1^s := x_1 - x_1^c, \quad x_2^s := x_2 - x_2^c}$$

Note that $\mathbf{x}^c$ is independent of $\mathbf{x}$. Can be **pre-chosen**

Oblivious matrix multiplication $\mathbf{W} \bullet \mathbf{x} + \mathbf{b}$

$$= \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \bullet \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \bullet \begin{bmatrix} x_1^s + x_1^c \\ x_2^s + x_2^c \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$= \begin{bmatrix} w_{1,1}(x_1^s + x_1^c) + w_{1,2}(x_2^s + x_2^c) + b_1 \\ w_{2,1}(x_1^s + x_1^c) + w_{2,2}(x_2^s + x_2^c) + b_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1^s + w_{1,2}x_2^s + b_1 + w_{1,1}x_1^c + w_{1,2}x_2^c \\ w_{2,1}x_1^s + w_{2,2}x_2^s + b_2 + w_{2,1}x_1^c + w_{2,2}x_2^c \end{bmatrix}$$

Compute locally by the server

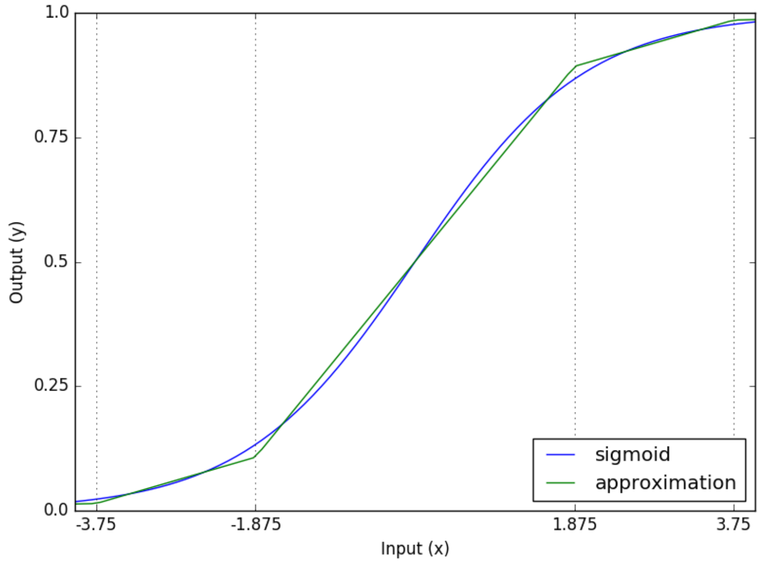
Dot-product

A setup phase protocol can prepare “dot product triplets”

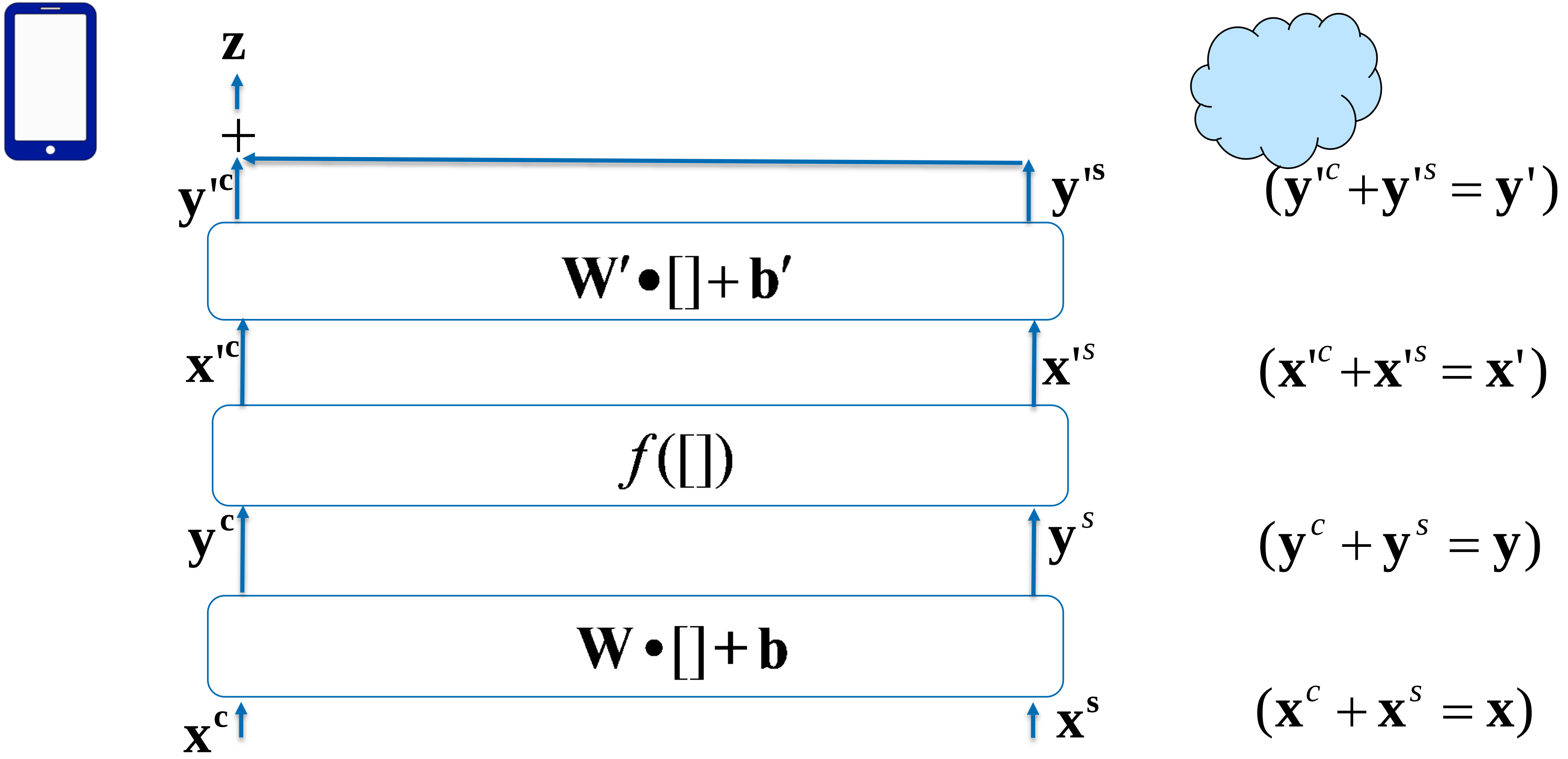
https://eprint.iacr.org/2017/452

Oblivious activation/pooling functions $f(y)$

- **Piecewise linear functions** e.g., ReLU: $x^s + x^c := \max(y^s + y^c, 0)$
  - easily computed obliviously using a garbled circuit
- **Smooth functions** e.g., Sigmoid: $x^s + x^c := 1/(1+e^{-(y^s+y^c)})$
  - approximate by a set of piecewise linear functions
  - then compute obliviously using a garbled circuit
  - empirically: ~14 segments sufficient



Core idea: use secret sharing for oblivious computation



For a layer with input $\mathbf{X}$ and output $\mathbf{Y}$, at the beginning:
arrange for client & server to have shares $\mathbf{x}^c$ and $\mathbf{x}^s$ s.t $\mathbf{x}^s+\mathbf{x}^c=\mathbf{x}$

At the end of oblivious processing:
client & server to have shares $\mathbf{y}^c$ and $\mathbf{y}^s$ s.t $\mathbf{y}^s+\mathbf{y}^c=\mathbf{y}$

Performance (for single queries)

Model	Latency (s)	Msg sizes (MB)	Accuracy
MNIST/Square	0.4 (+ 0.88)	44 (+ 3.6)	98.95%
CIFAR-10/ReLU	472 (+ 72)	6226 (+ 3046)	81.61%
PTB/Sigmoid	4.39 (+ 13.9)	474 (+ 86.7)	120/114.5 (perplexity)

Setup phase timings in parentheses

MiniONN pros and cons

300x-700x faster than CryptoNets

Can transform any given neural network to its oblivious variant

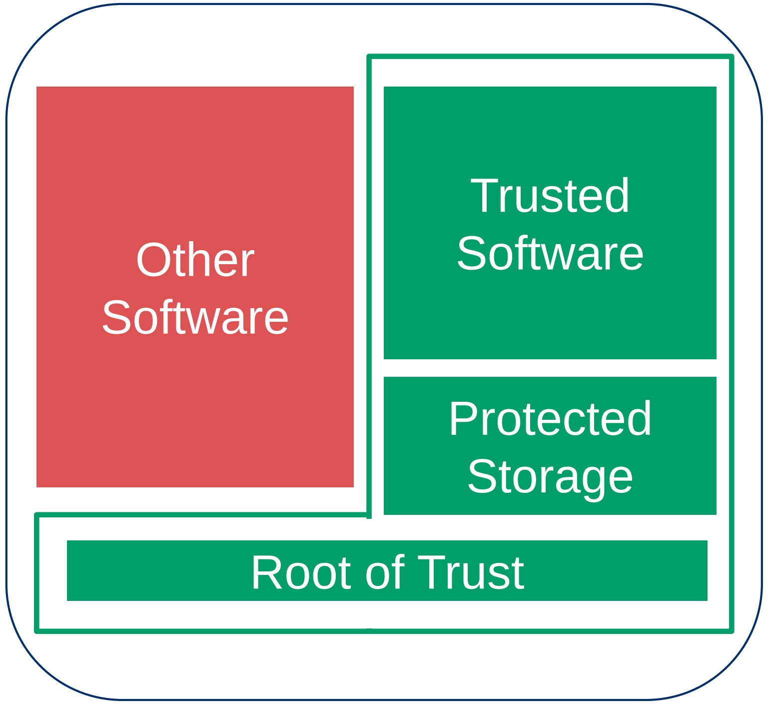
Still ~1000x slower than without privacy

Server can no longer filter requests or do sophisticated metering

Assumes online connectivity to server

Reveals structure (but not params) of NN

Can trusted computing help?



Hardware support for

- **Isolated execution: Trusted Execution Environment**
- **Protected storage: Sealing**
- **Ability to report status to a remote verifier: Attestation**

Cryptocards

https://www.ibm.com/security/cryptocards/

Trusted Platform Modules

https://www.infineon.com/tpm

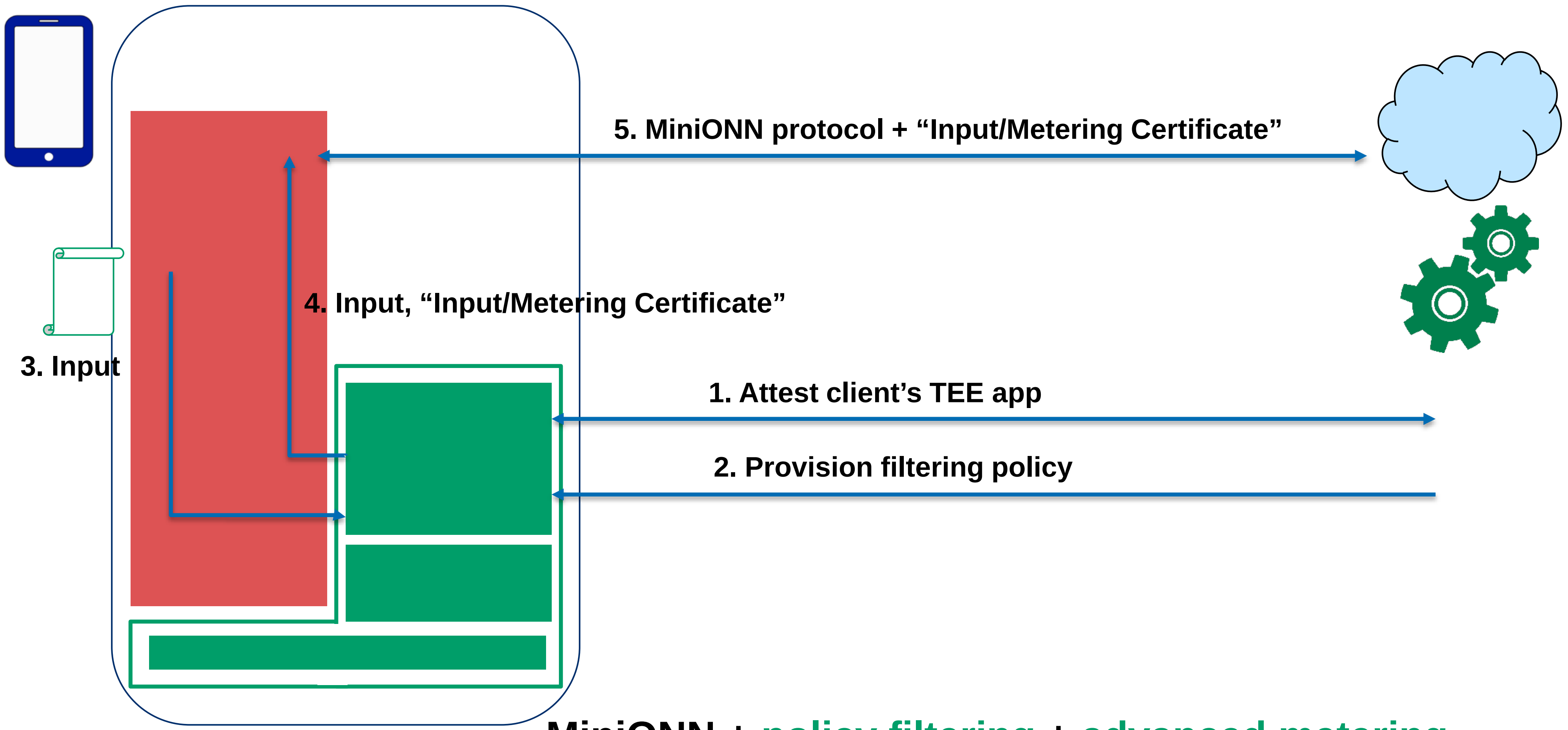
ARM TrustZone

https://www.arm.com/products/security-on-arm/trustzone

Intel Software Guard Extensions

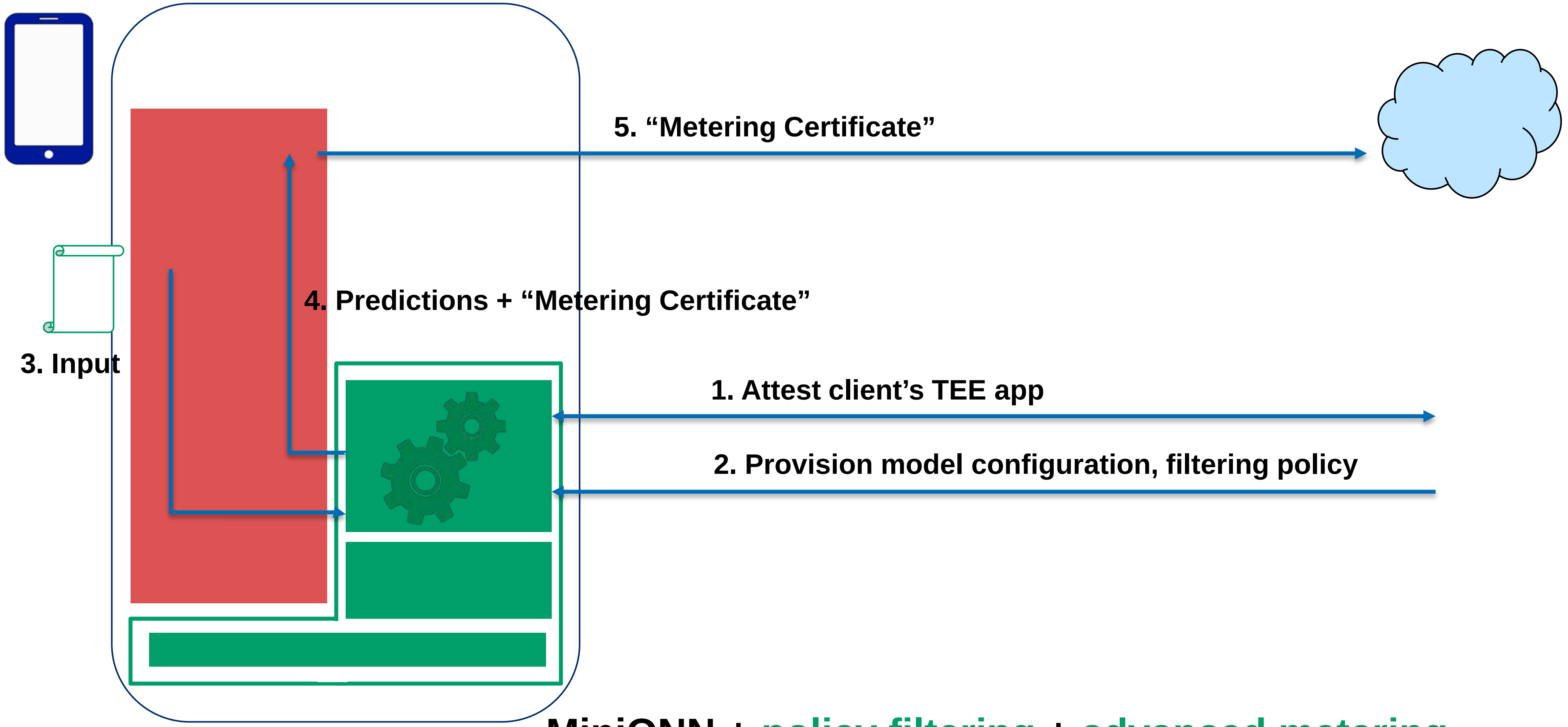
https://software.intel.com/en-us/sgx

Using a client-side TEE to vet input



MiniONN + policy filtering + advanced metering

Using a client-side TEE to run the model



MiniONN + policy filtering + advanced metering
+ disconnected operation + performance + better privacy
- harder to reason about model secrecy

Summary

Applications of machine learning must consider threats from possible adversaries.

MiniONN: Efficiently transform any given neural network into oblivious form with no/negligible accuracy loss

Trusted Computing can help realize improved security and privacy for ML

https://eprint.iacr.org/2017/452

Fin.