Aalto University

# Machine Learning
## *in the presence of adversaries*

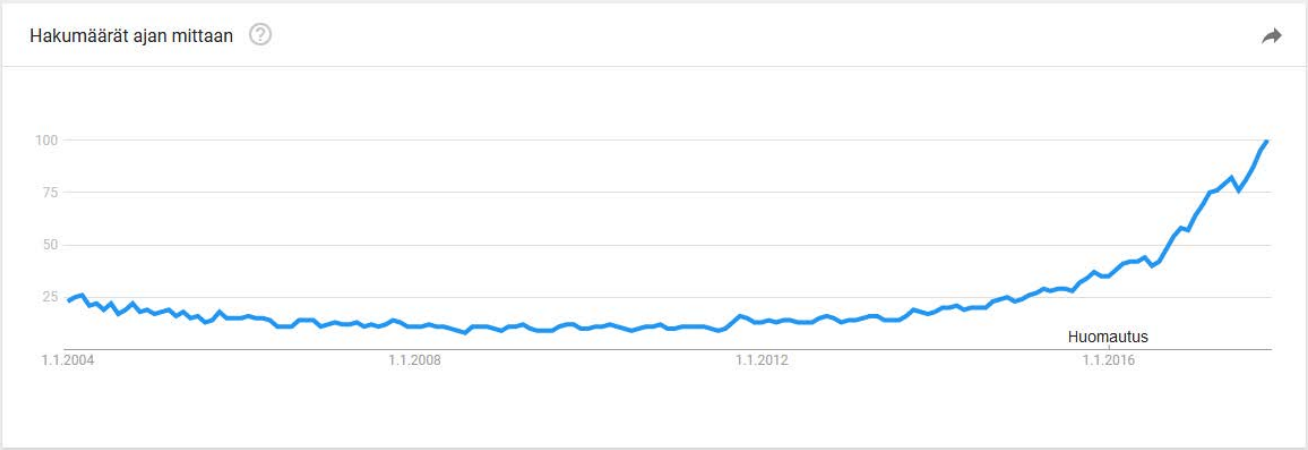
*N. Asokan*
*http://asokan.org/asokan/*
*@nasokan*

*(joint work with Mika Juuti, Jian Liu, Andrew Paverd and Samuel Marchal)*

# Machine Learning is ubiquitous

**The ML market size is expected to grow by 44% annually over next five years**

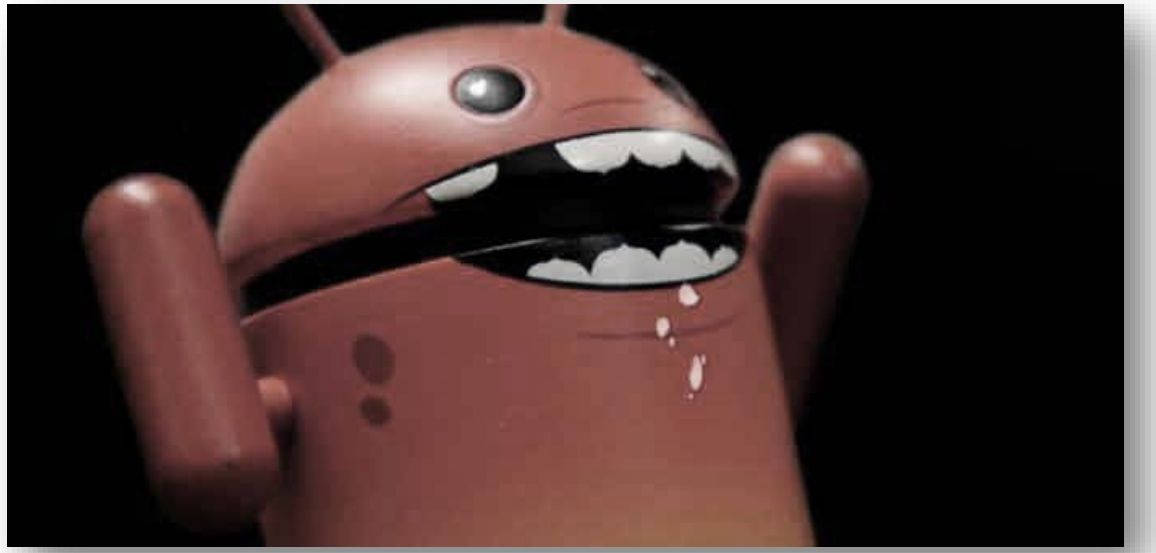
**In 2016, companies invested up to $9 Billion in AI-based startups**




[1] http://www.marketsandmarkets.com/PressReleases/machine-learning.asp
[2] McKinsey Global Institute, ”Artificial Intelligence: The Next Digital Frontier?”

# Machine Learning *for* security/privacy


**Access Control**



**Deception Detection**

Marchal et al., “Off-the-Hook: An Efficient and Usable Client-Side Phishing Prevention Application” 2017. IEEE Trans. Comput.
https://ssg.aalto.fi/projects/phishing/

# How do we evaluate ML-based systems?

**Effectiveness of inference**

- measures of accuracy

**Performance**

- inference speed and memory consumption

**...**

**Meeting these criteria even in the presence of adversaries?**

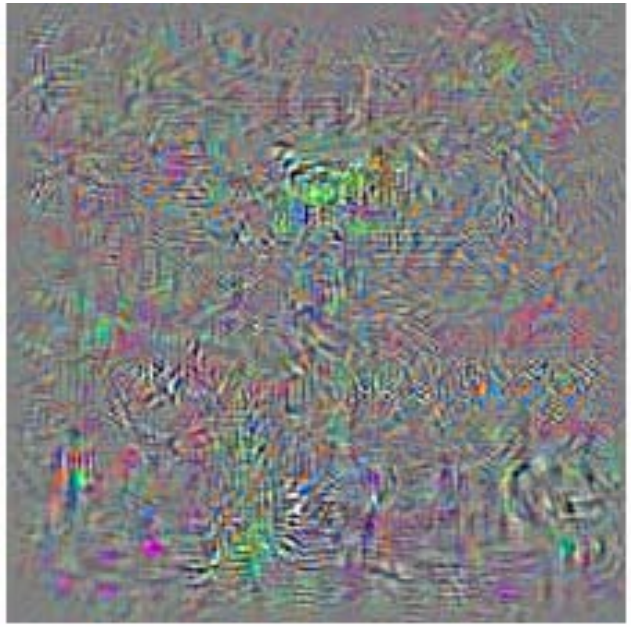
# Security & privacy *of* machine learning

# Adversarial examples

Skip to robust adversarial examples

**Which class is this?**
**School bus**

**+ 0.1·**



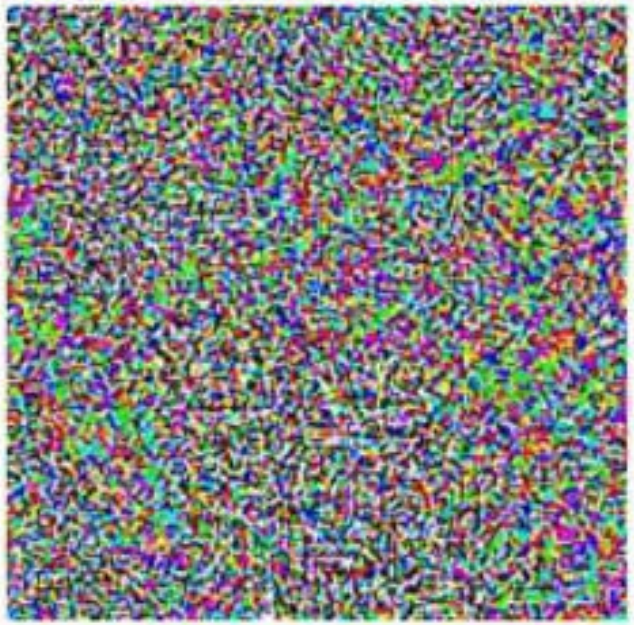
**=**

**Which class is this?**
**Ostrich**

Szegedy et al., “Intriguing Properties of Neural Networks” 2014. https://arxiv.org/abs/1312.6199v4

# Adversarial examples

+ 0.007·



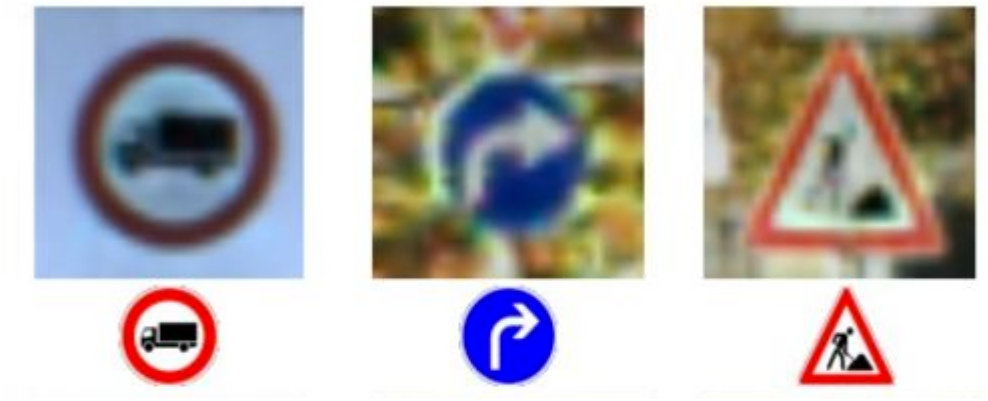
=

**Which class is this?**
**Panda**

**Which class is this?**
**Gibbon**

Goodfellow et al., “Explaining and Harnessing Adversarial Examples” ICLR 2015

# Adversarial examples



Papernot et al, “Practical Black-Box Attacks against Machine Learning”, *ASIACCS 2017* https://arxiv.org/abs/1602.02697

**Which class is this?**

**Cat**

Athalye et al. “Synthesizing Robust Adversaria

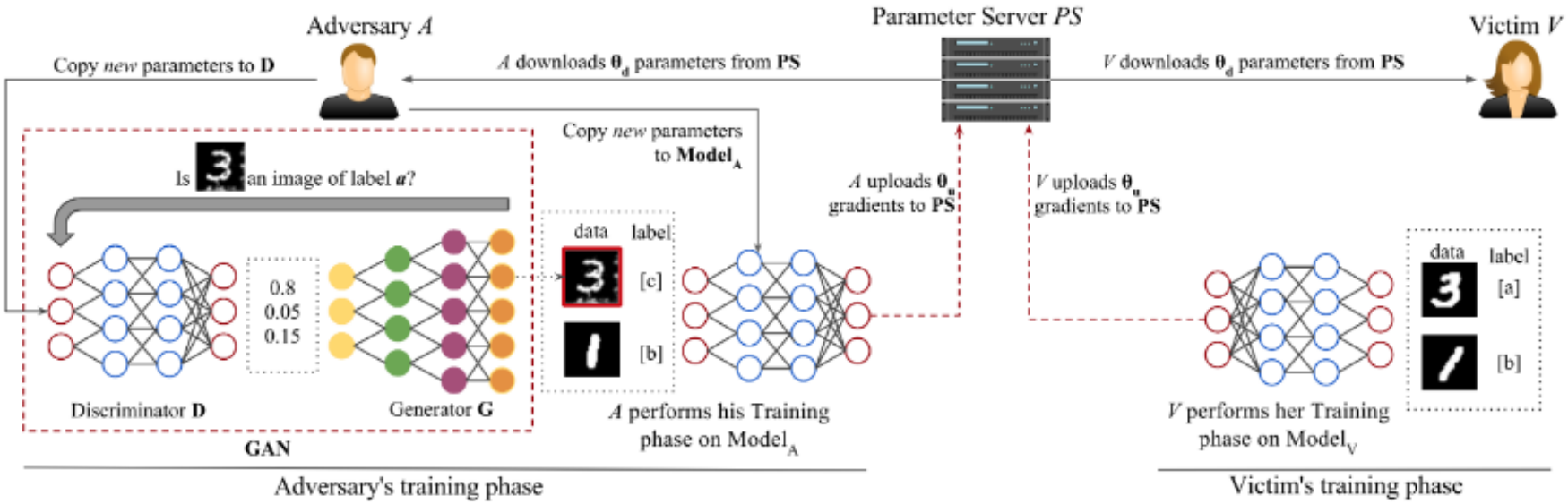
Zhang et al, “DolphinAttack: Inaudible Voice Commands”, ACM CCS 2017 https://arxiv.org/abs/1708.09537

# Model inversion

Target Softmax MLP DAE

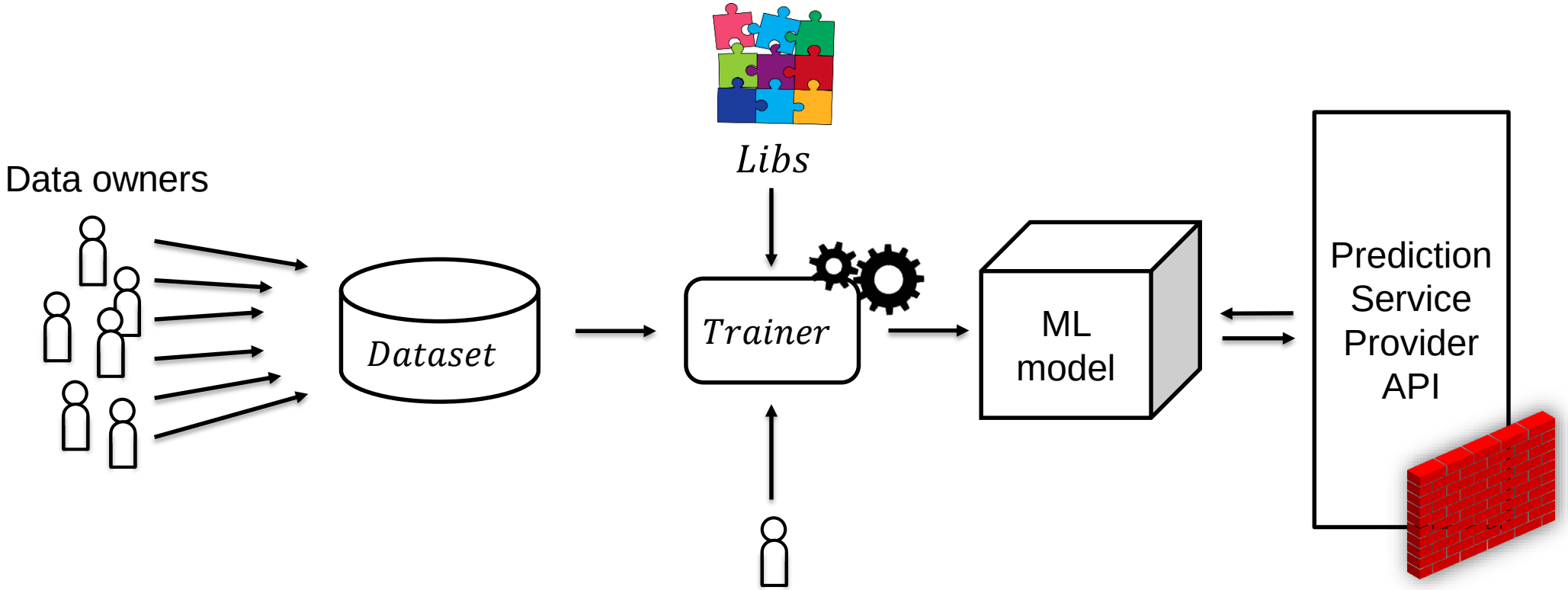
Fredrikson et al. “Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures”, ACM CCS 2015.
https://www.cs.cmu.edu/~mfredrik/papers/fjr2015ccs.pdf

# Model inversion



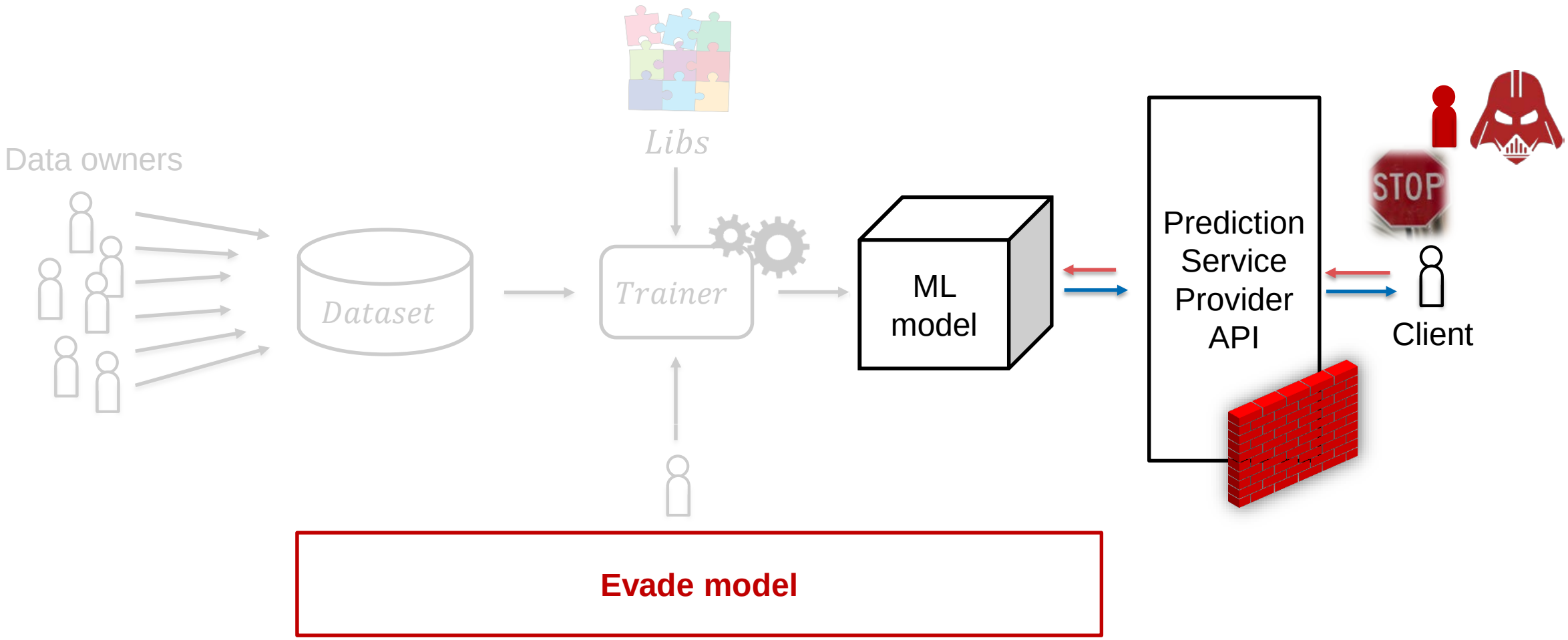

Hitaj et al. “Deep models under the GAN: information leakage from collaborative deep learning”, ACM CCS 2017.
https://arxiv.org/pdf/1702.07464.pdf

# Machine Learning pipeline



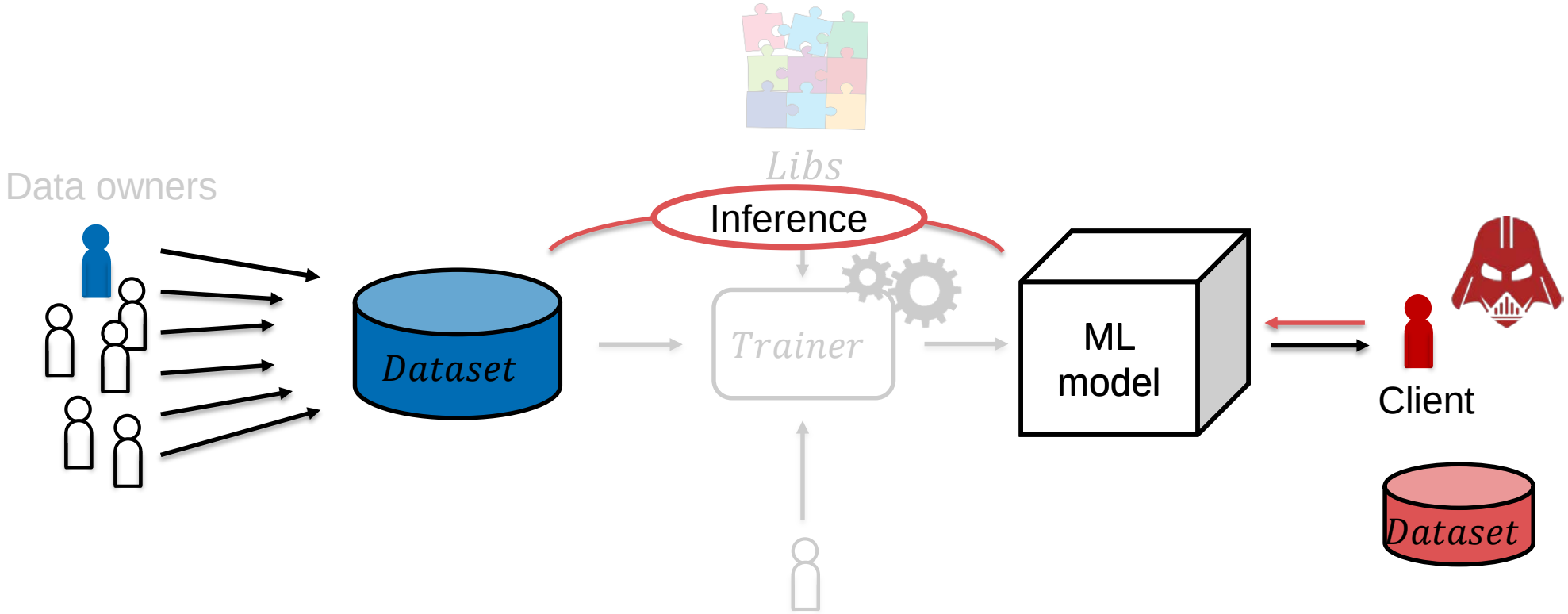

**Where is the adversary? What is its target?**

# Compromised input




Dang et al., “Evading Classifiers by Morphing in the Dark”, ACM CCS ’17. https://arxiv.org/abs/1705.07535
Evtimov et al., “Robust Physical-World Attacks on Deep Learning Models”. https://arxiv.org/abs/1707.08945
Zhang et al., “DolphinAttack: Inaudible Voice Commands”, ACM CCS ’17. https://arxiv.org/abs/1708.09537

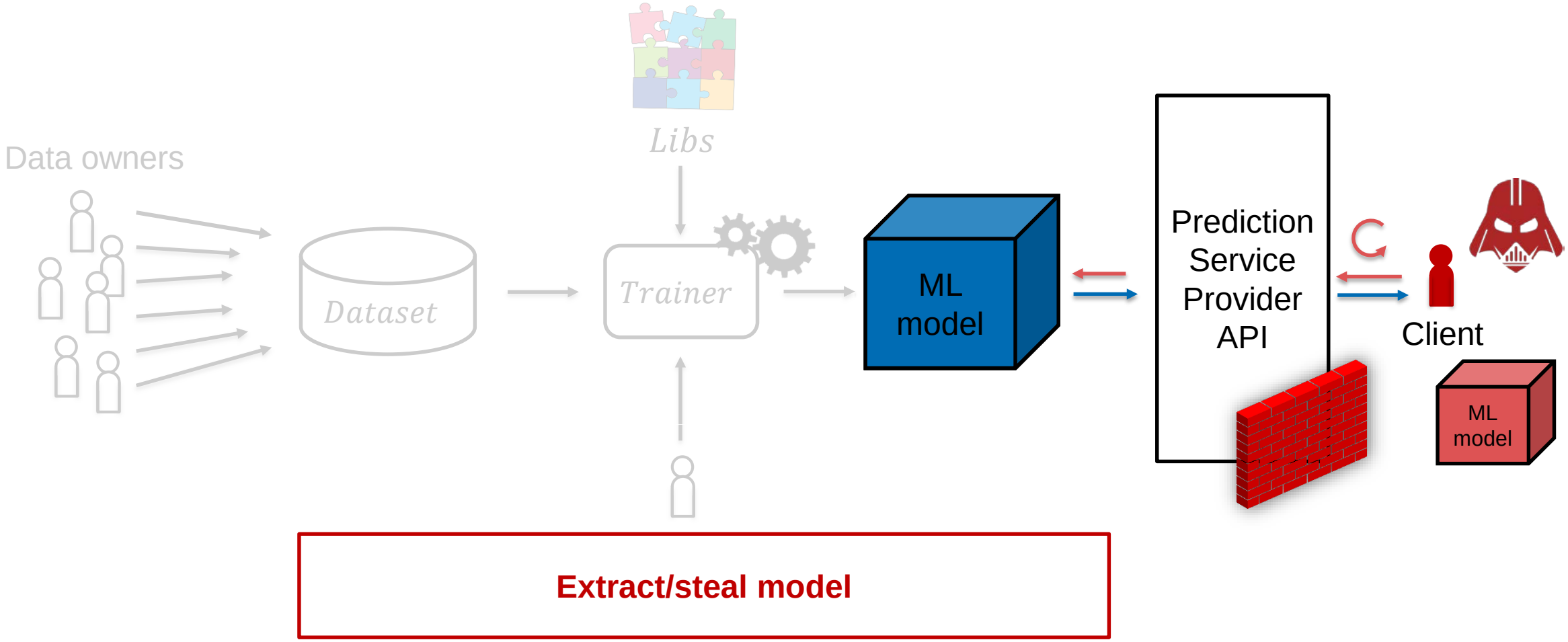
# Malicious client




**Invert model, infer membership**

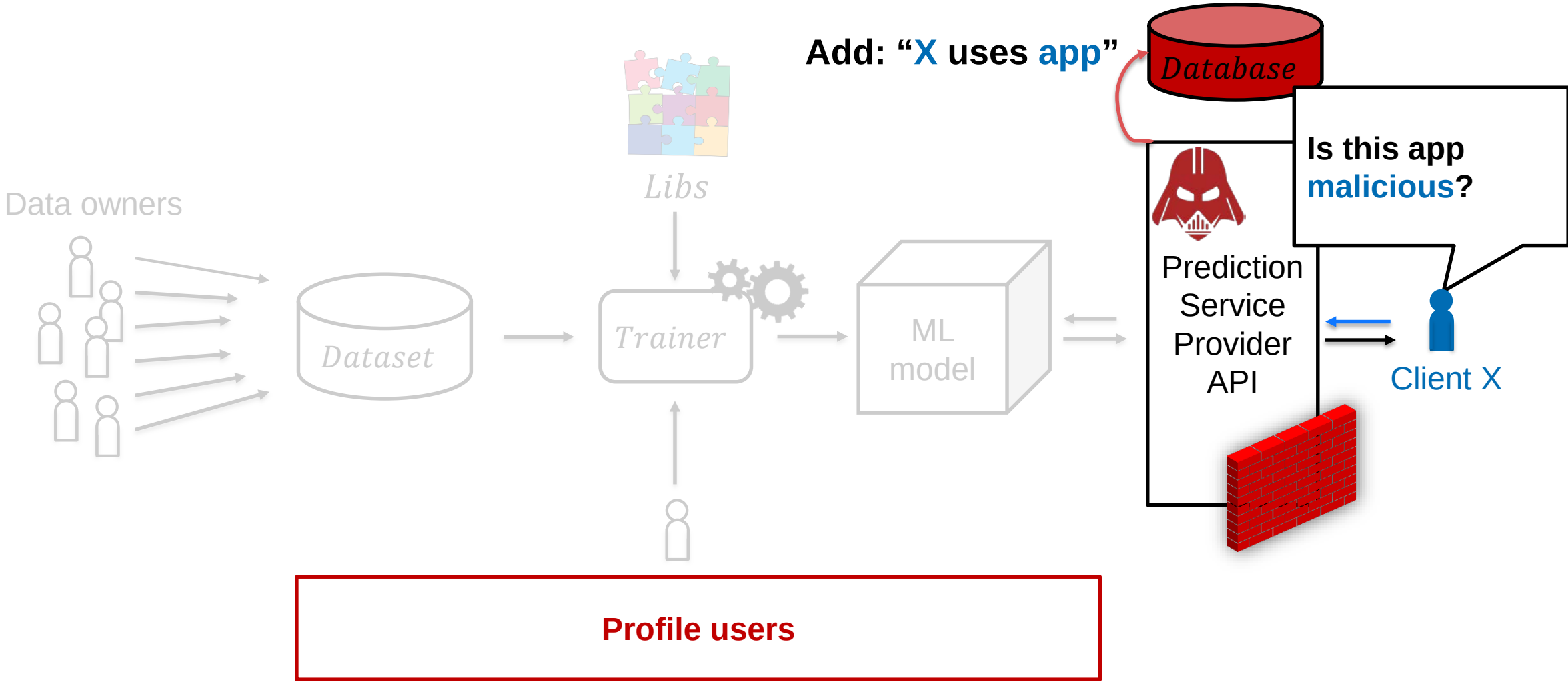
Shokri et al., “Membership Inference Attacks Against Machine Learning Models”. IEEE S&P ’16. https://arxiv.org/pdf/1610.05820.pdf
Fredrikson et al., “Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures”. ACM CCS’15. https://www.cs.cmu.edu/~mfredrik/papers/fjr2015ccs.pdf

# Malicious client



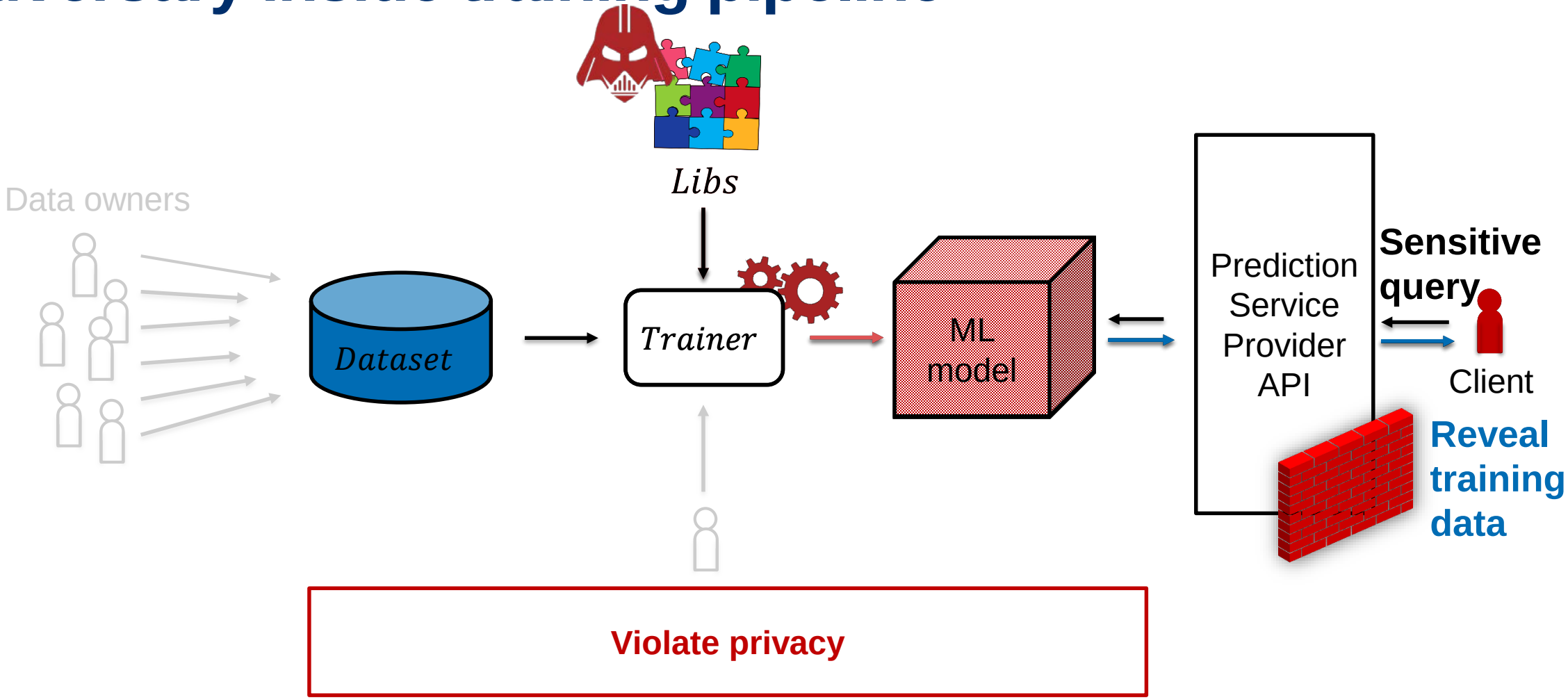

Tramer et al., “Stealing ML models via prediction APIs”, Usenix SEC ’16. https://arxiv.org/abs/1609.02943

# Malicious prediction service



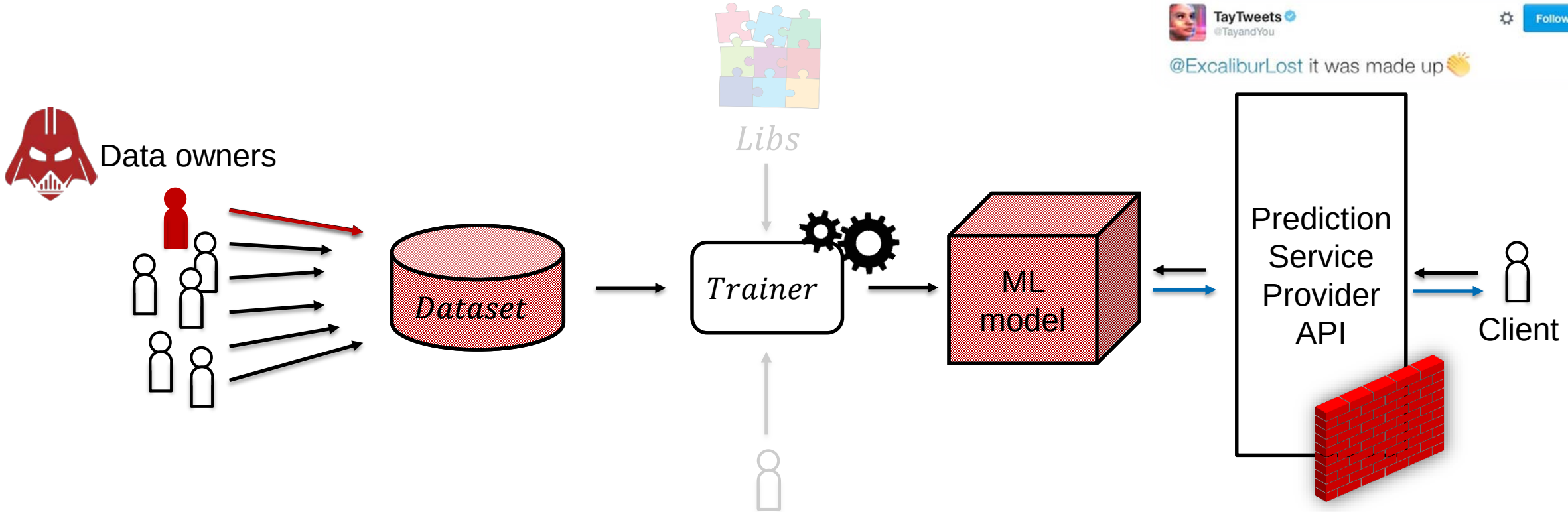

Malmi and Weber. "You are what apps you use Demographic prediction based on user's apps", ICWSM '16. https://arxiv.org/abs/1603.00059

# Compromised toolchain: adversary inside training pipeline




Song et al., “Machine Learning models that remember too much”, ACM CCS ’17. https://arxiv.org/abs/1709.07886
Hitja et al., “Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning”, ACM CCS ’17. http://arxiv.org/abs/1702.07464

# Malicious data owner


**Influence ML model (model poisoning)**

https://www.theguardian.com/technology/2016/mar/26/microsoft-deeply-sorry-for-offensive-tweets-by-ai-chatbot
https://www.theguardian.com/technology/2017/nov/07/youtube-accused-violence-against-young-children-kids-content-google-pre-school-abuse

# Summary

**ML-based systems need to worry about adversarial behaviour**

**Adversaries are multilateral, solutions are too**

**Adversarial machine learning is now a very active research area**

**Secure Systems Group**

**https://ssg.aalto.fi/**

**Finnish Center for AI**

**http://fcai.fi/**

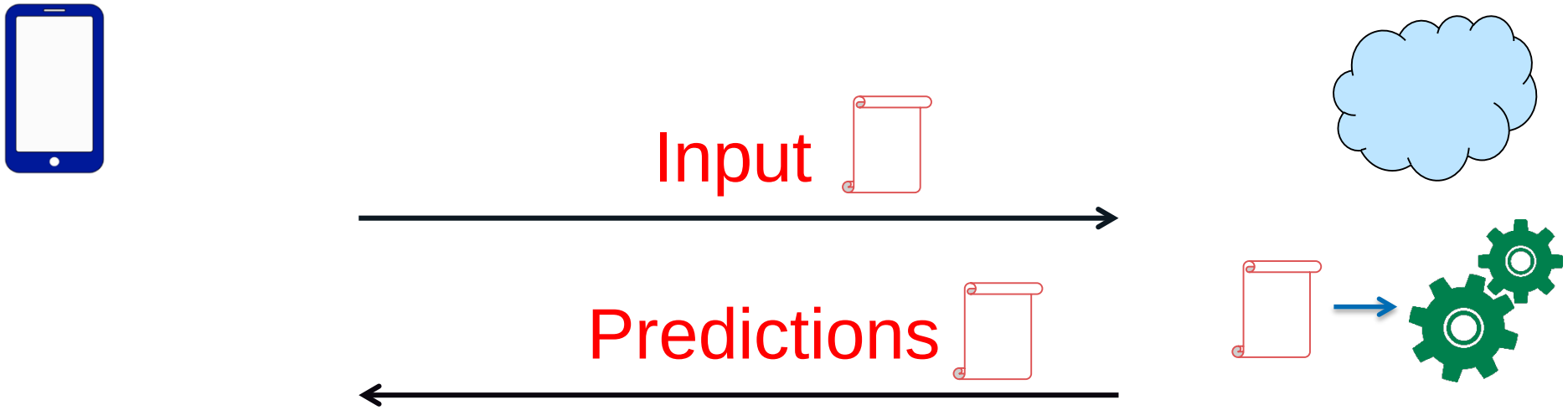
Aalto University

# Oblivious Neural Network Predictions via MiniONN Transformations

*N. Asokan*
*http://asokan.org/asokan/*
*@nasokan*

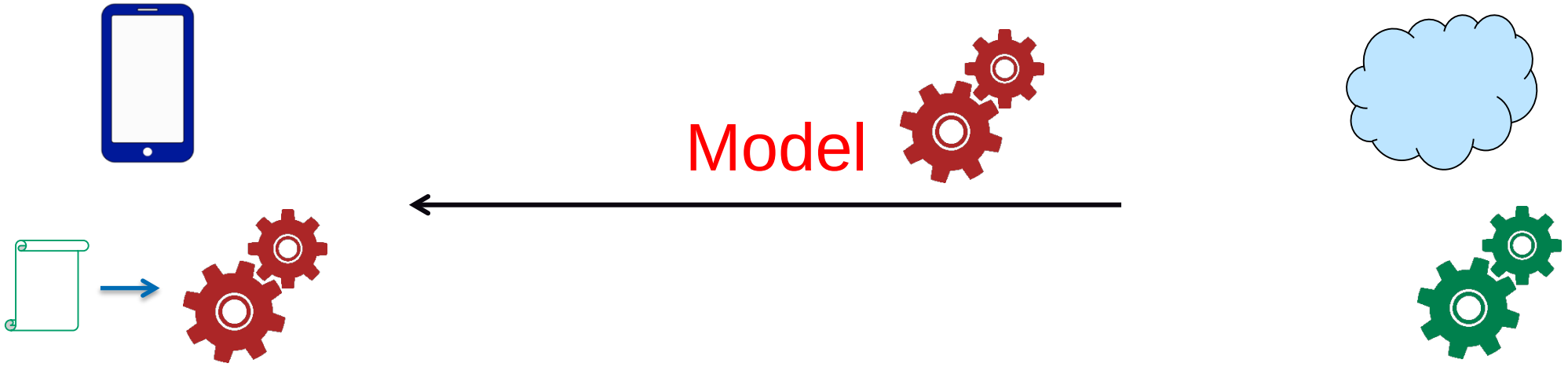
*(Joint work with Jian Liu, Mika Juuti, Yao Lu)*

# Machine learning as a service (MLaaS)




violation of clients' privacy

# Running predictions on client-side



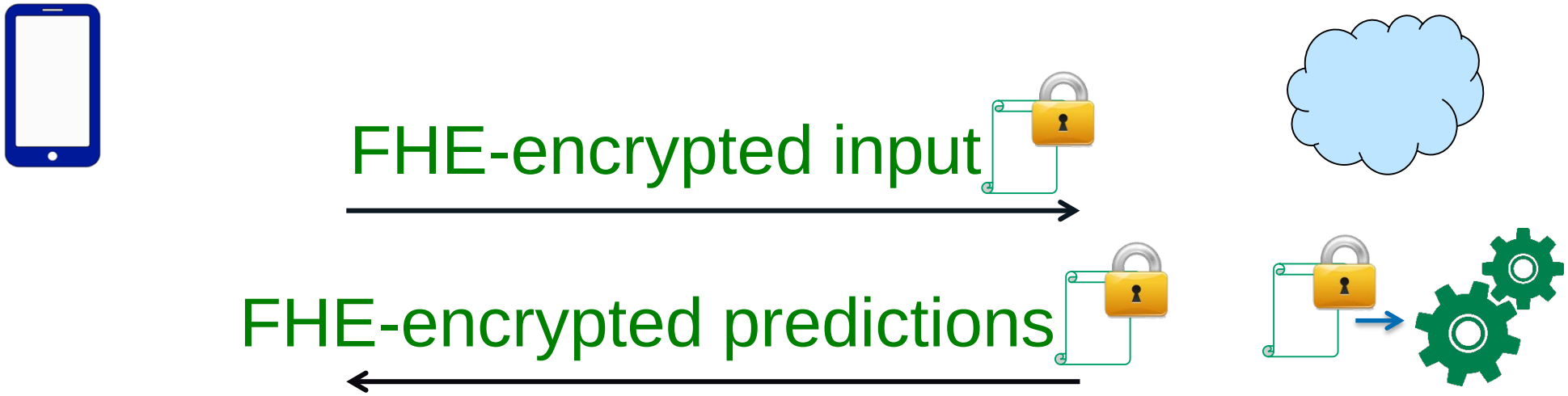

model theft
evasion
model inversion

# 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



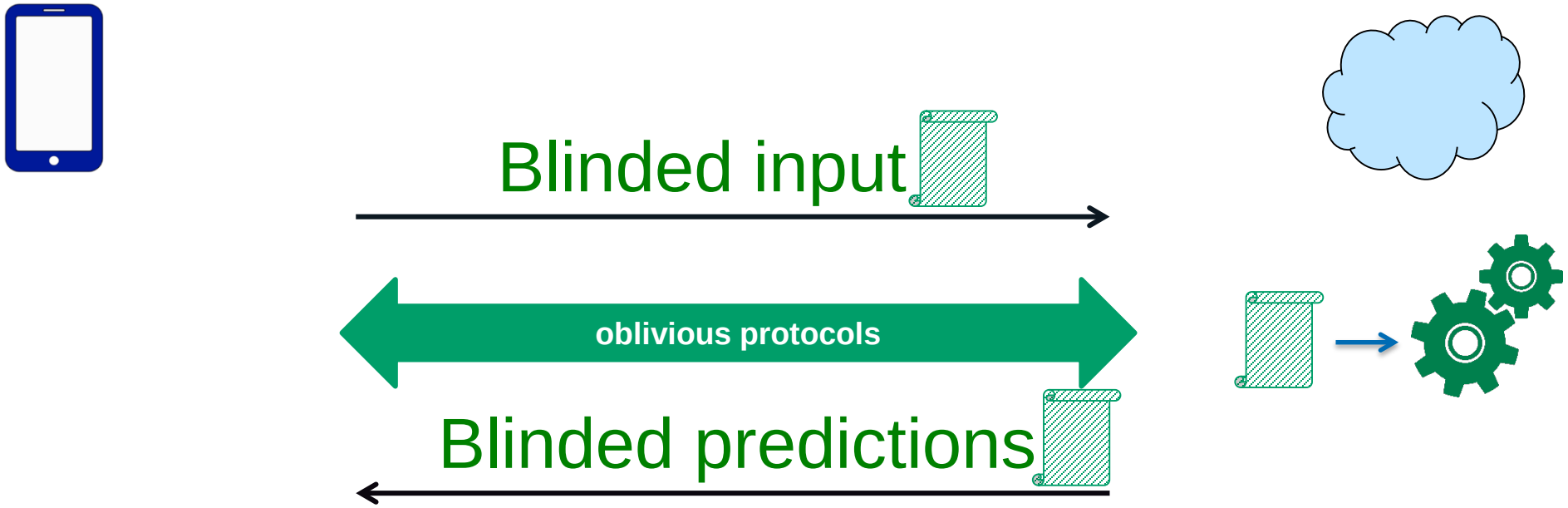

- High throughput for batch queries from same client
- High overhead for single queries: 297.5s and 372MB (MNIST dataset)
- Cannot support: high-degree polynomials, comparisons, …

[GDLLNW16] CryptoNets, ICML 2016

FHE: Fully homomorphic encryption (https://en.wikipedia.org/wiki/Homomorphic_encryption)

# MiniONN: Overview

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

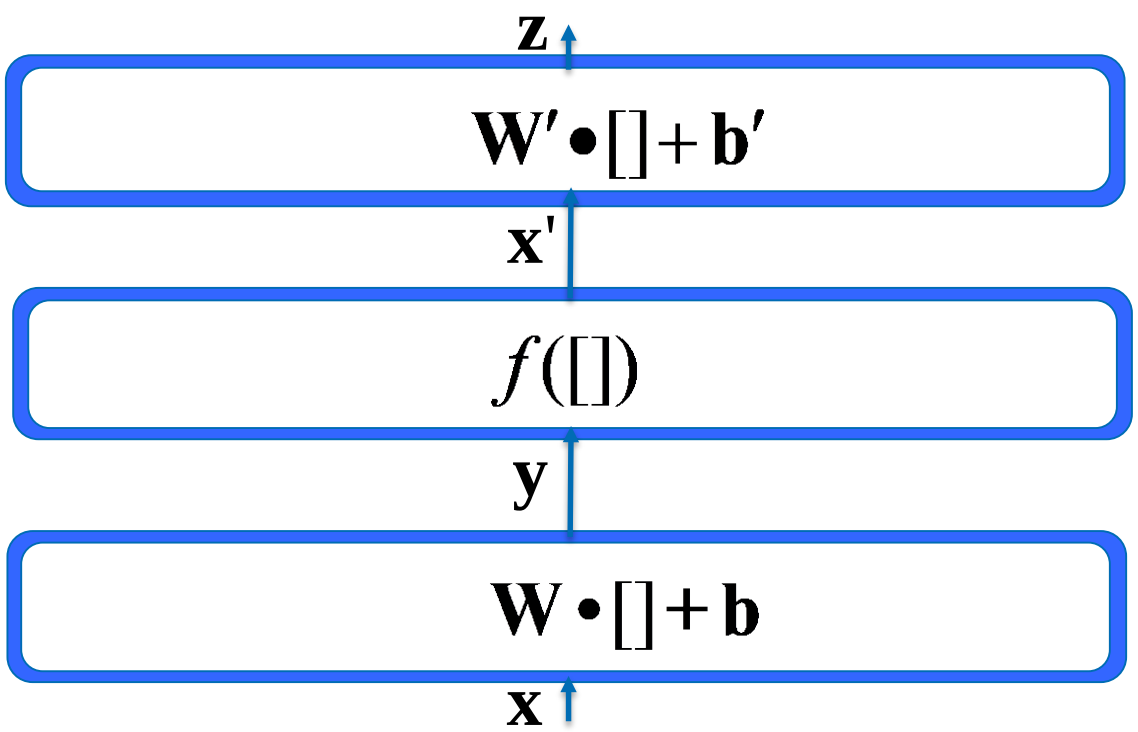



- Low overhead: ~1s
- Support all common neural networks

Skip to performance

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

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

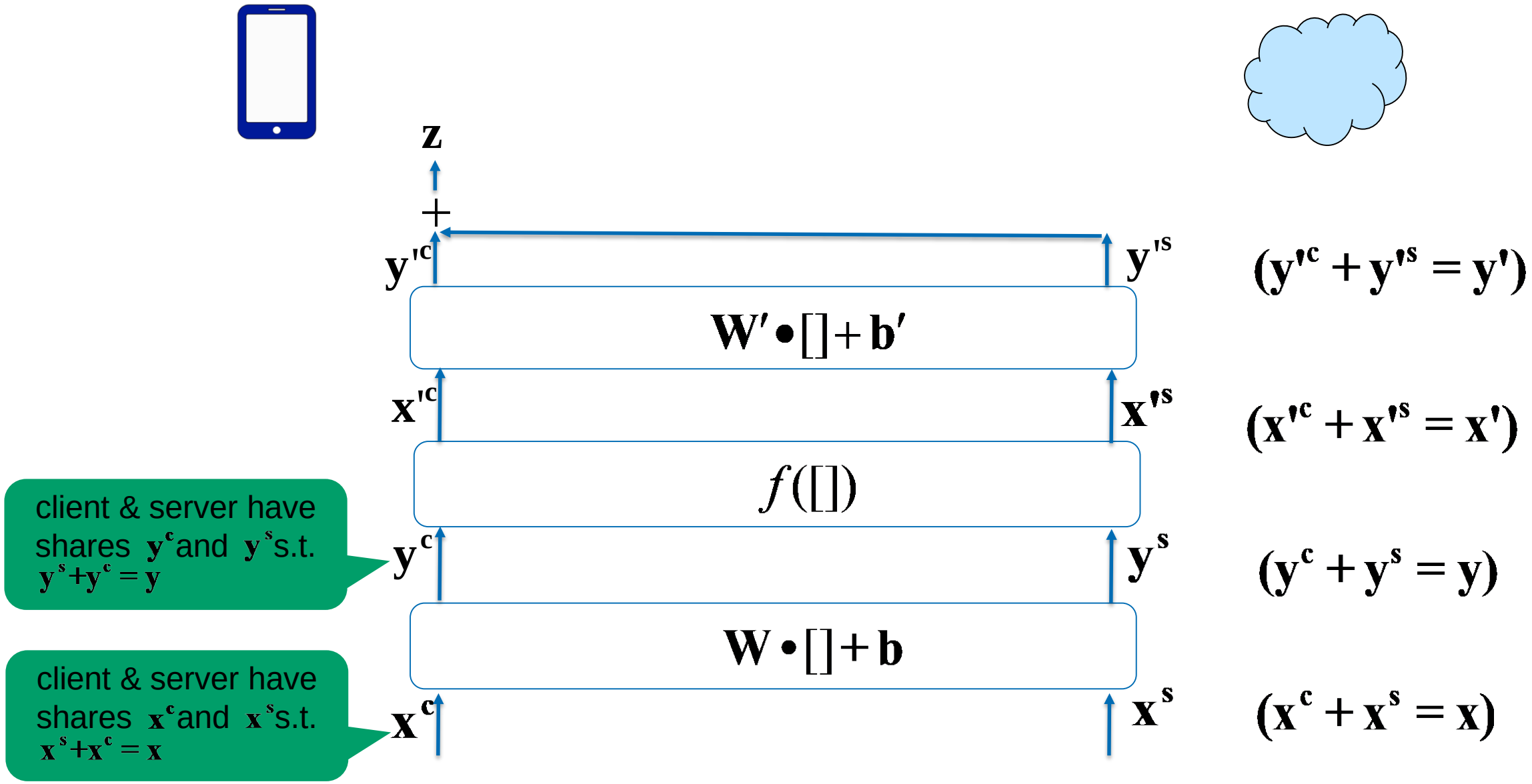



https://eprint.iacr.org/2017/452

**All operations are in a finite field $Z_N$**

Skip to performance

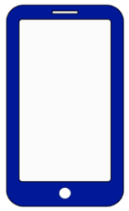
# Core idea: use secret sharing for oblivious computation

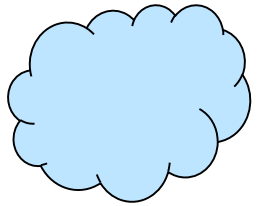



**Use efficient cryptographic primitives (2PC, additively homomorphic encryption)**

Skip to performance

# Secret sharing initial input $\mathbf{x}$





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

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

$$\longrightarrow$$

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

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

Skip to performance

# Oblivious linear transformation $\mathbf{W} \bullet \mathbf{x} + \mathbf{b}$

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

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

Compute locally by the server

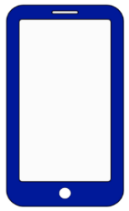
Dot-product

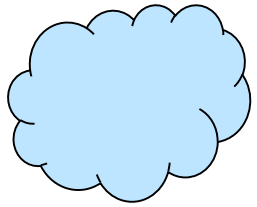
$W \bullet x^c$

Skip to performance

# Oblivious linear transformation: dot-product

Homomorphic Encryption with SIMD





$$r_{1,1}, r_{1,2}, r_{2,1}, r_{2,2} \xleftarrow{\$} Z_N$$

$$\xleftarrow{\quad E(w_{1,1}), E(w_{1,2}), E(w_{2,1}), E(w_{2,2}) \quad}$$

$$c_{1,1} = E(w_{1,1}x_1^c - r_{1,1})$$

$$c_{1,2} = E(w_{1,2}x_2^c - r_{1,2})$$

$$\xrightarrow{\quad c_{1,1}, c_{1,2}, c_{2,1}, c_{2,2} \quad}$$

$$c_{2,1} = E(w_{2,1}x_1^c - r_{2,1})$$

$$D(c_{1,1}), D(c_{1,2}), D(c_{2,1}), D(c_{2,2})$$

$$c_{2,2} = E(w_{2,2}x_2^c - r_{2,2})$$

$$v_1 = r_{1,1} + r_{1,2} \qquad u_1 = w_{1,1}x_1^c + w_{1,2}x_2^c - (r_{1,2} + r_{1,1})$$

$$v_2 = r_{2,1} + r_{2,2} \qquad u_2 = w_{2,1}x_1^c + w_{2,2}x_2^c - (r_{2,1} + r_{2,2})$$

**u + v = W•x$^c$**; Note: **u, v,** and **W•x$^c$** are independent of **x**.
<**u,v,x$^c$** > generated/stored in a **precomputation phase**

Skip to performance

# Oblivious linear transformation $\mathbf{W} \bullet \mathbf{x} + \mathbf{b}$

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

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

$$= \begin{bmatrix} \boxed{w_{1,1}x_1^s + w_{1,2}x_2^s + b_1} + \boxed{u_1} \\ \boxed{w_{2,1}x_1^s + w_{2,2}x_2^s + b_2} + \boxed{u_2} \end{bmatrix} + \boxed{\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}}$$

Skip to performance

# Oblivious linear transformation $\mathbf{W} \bullet \mathbf{x} + \mathbf{b}$

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

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

$$= \begin{bmatrix} \boxed{w_{1,1}x_1^s + w_{1,2}x_2^s + b_1 + u_1} \\ \boxed{w_{2,1}x_1^s + w_{2,2}x_2^s + b_2 + u_2} \end{bmatrix} + \begin{bmatrix} \boxed{v_1} \\ \boxed{v_2} \end{bmatrix} := \begin{bmatrix} \boxed{y_1^s} \\ \boxed{y_2^s} \end{bmatrix} + \begin{bmatrix} \boxed{y_1^c} \\ \boxed{y_2^c} \end{bmatrix}$$

Skip to performance

# Oblivious activation/pooling functions $f(y)$

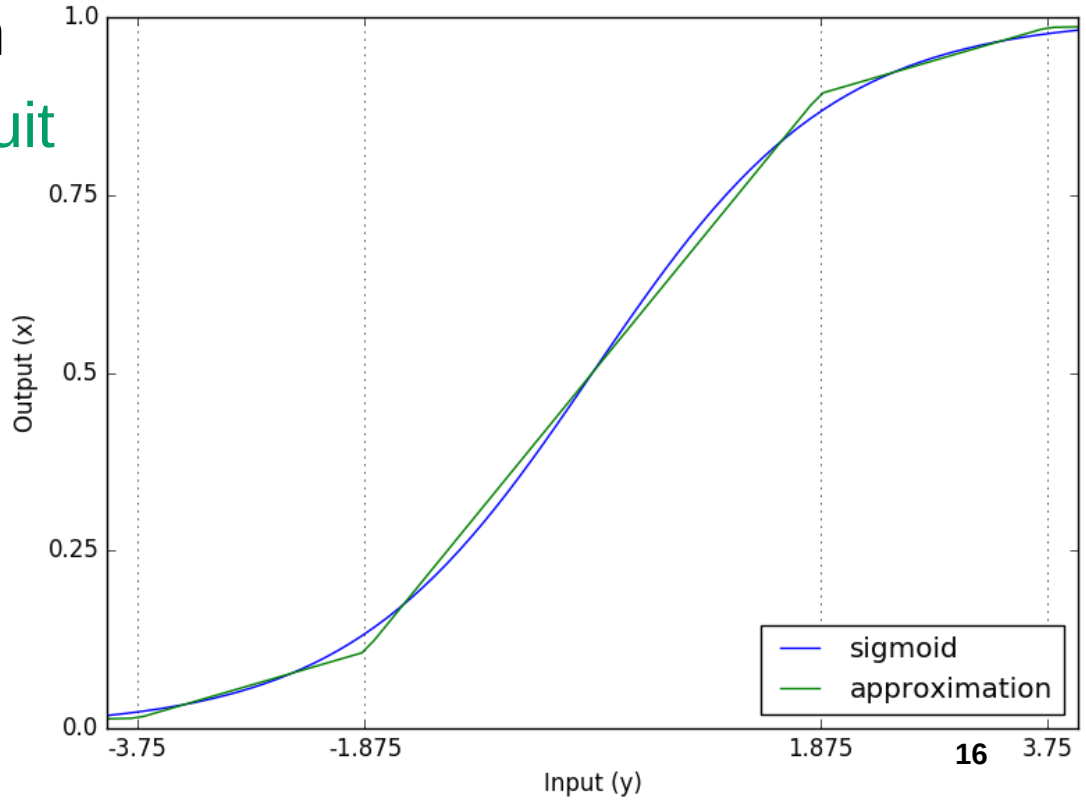
**Piecewise linear functions** e.g.,

- ReLU: $x := \max(y, 0)$
- Oblivious ReLU: $x^s + x^c := \max(y^s + y^c, 0)$
  - easily computed obliviously by a garbled circuit

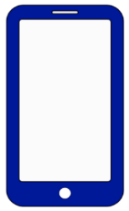
# Oblivious activation/pooling functions $f(y)$

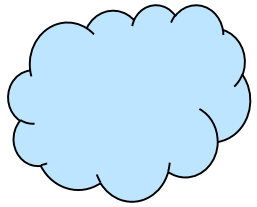
**Smooth functions** e.g.,

- Sigmoid: $x := 1/(1+e^{-y})$
- Oblivious sigmoid: $x^s + x^c := 1/(1+e^{-(y^s+y^c)})$
  - approximate by a piecewise linear function
  - then compute obliviously by a garbled circuit
  - empirically: ~14 segments sufficient



# Combining the final result





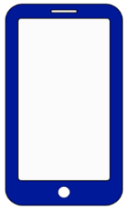
$$y_1^s, y_2^s \longleftarrow$$

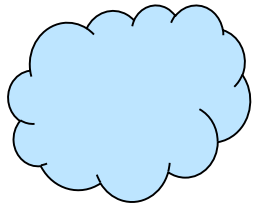
$$y_1 := y_1^s + y_1^c$$

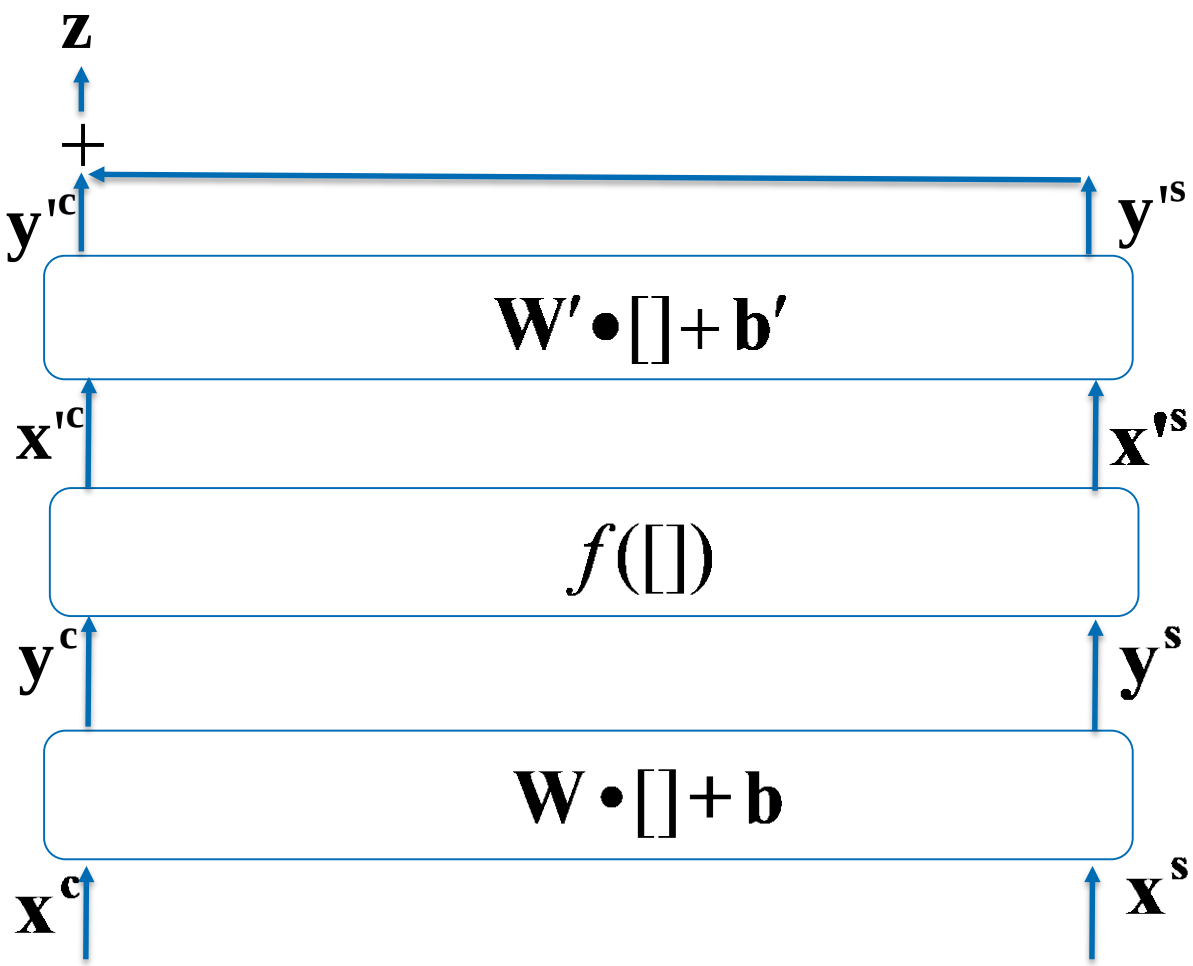
$$y_2 := y_2^s + y_2^c$$

They can jointly calculate max($y_1$,$y_2$)
(for minimizing information leakage)

# Core idea: use secret sharing for oblivious computation








$(y'^c + y'^s = y')$

$(x'^c + x'^s = x')$

$(y^c + y^s = y)$

$(x^c + x^s = x)$

# Performance (for single queries)

| Model | Latency (s) | Msg sizes (MB) | Loss of accuracy |
|---|---|---|---|
| MNIST/Square | 0.4 (+ 0.88) | 44 (+ 3.6) | none |
| CIFAR-10/ReLU | 472 (+ 72) | 6226 (+ 3046) | none |
| PTB/Sigmoid | 4.39 (+ 13.9) | 474 (+ 86.7) | Less than 0.5% (cross-entropy loss) |

Pre-computation phase timings in parentheses

PTB = Penn Treebank

Skip to End

# MiniONN pros and cons

300-700x faster than CryptoNets

Can transform any given neural network to its oblivious variant

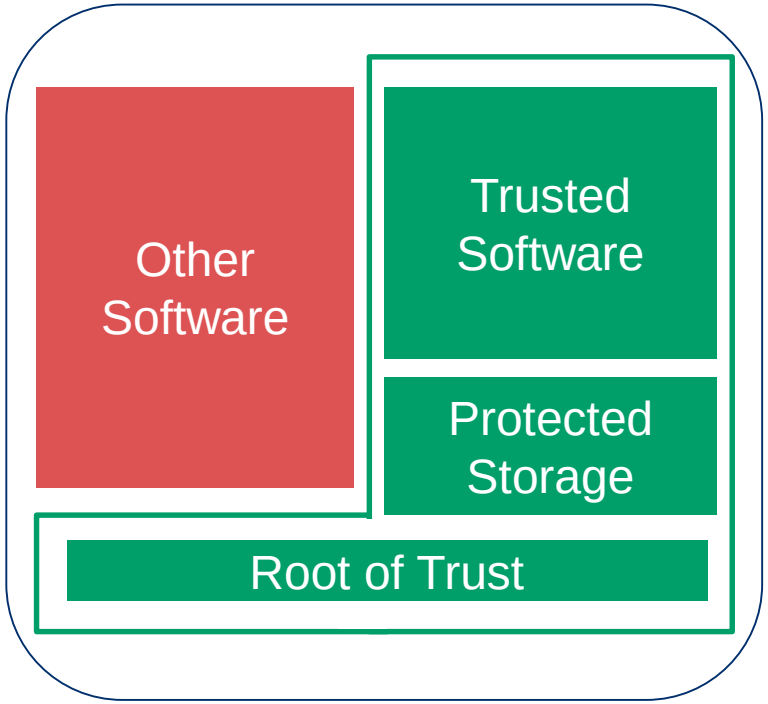
Still ~1000x slower than without privacy

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

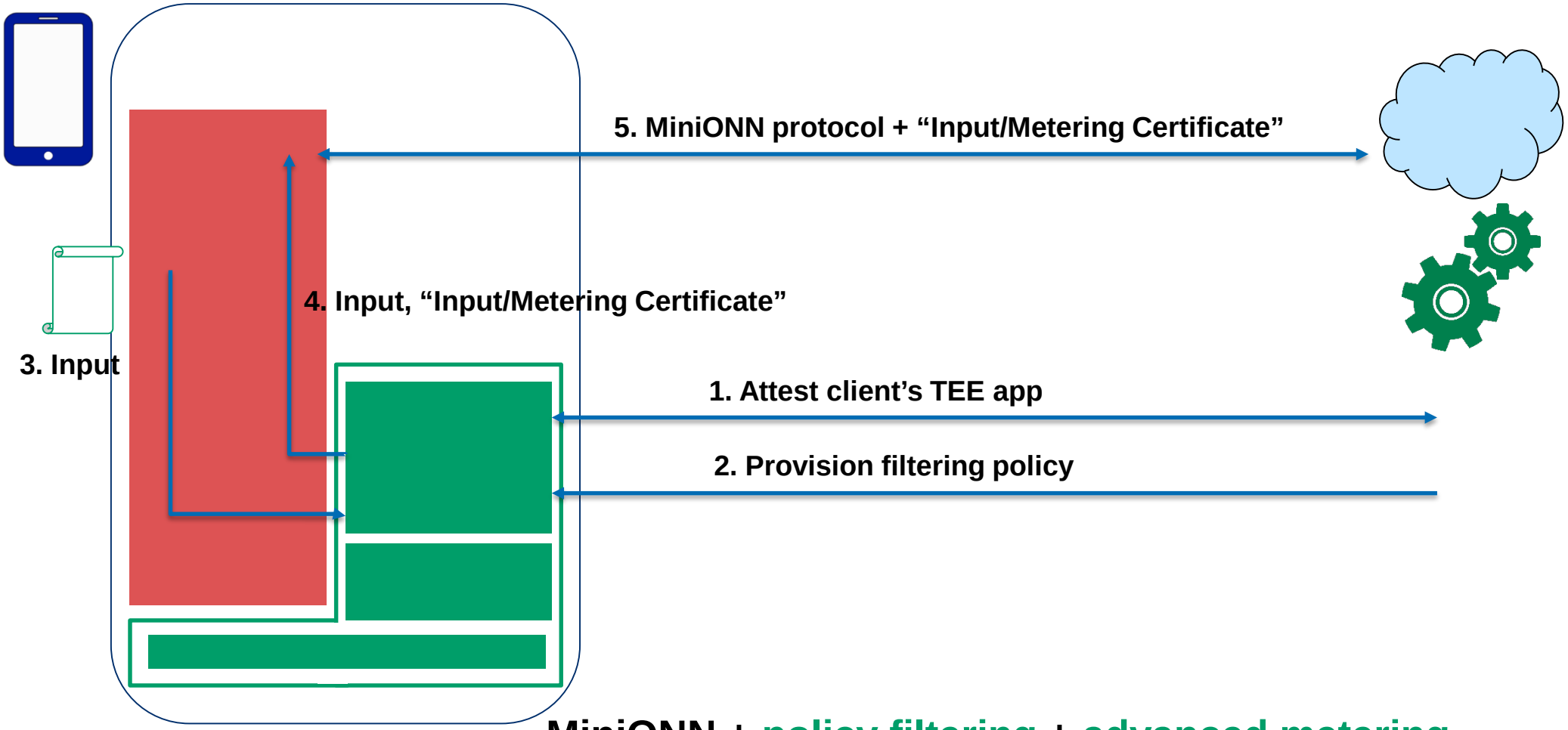
ARM TrustZone

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

Intel Software Guard Extensions

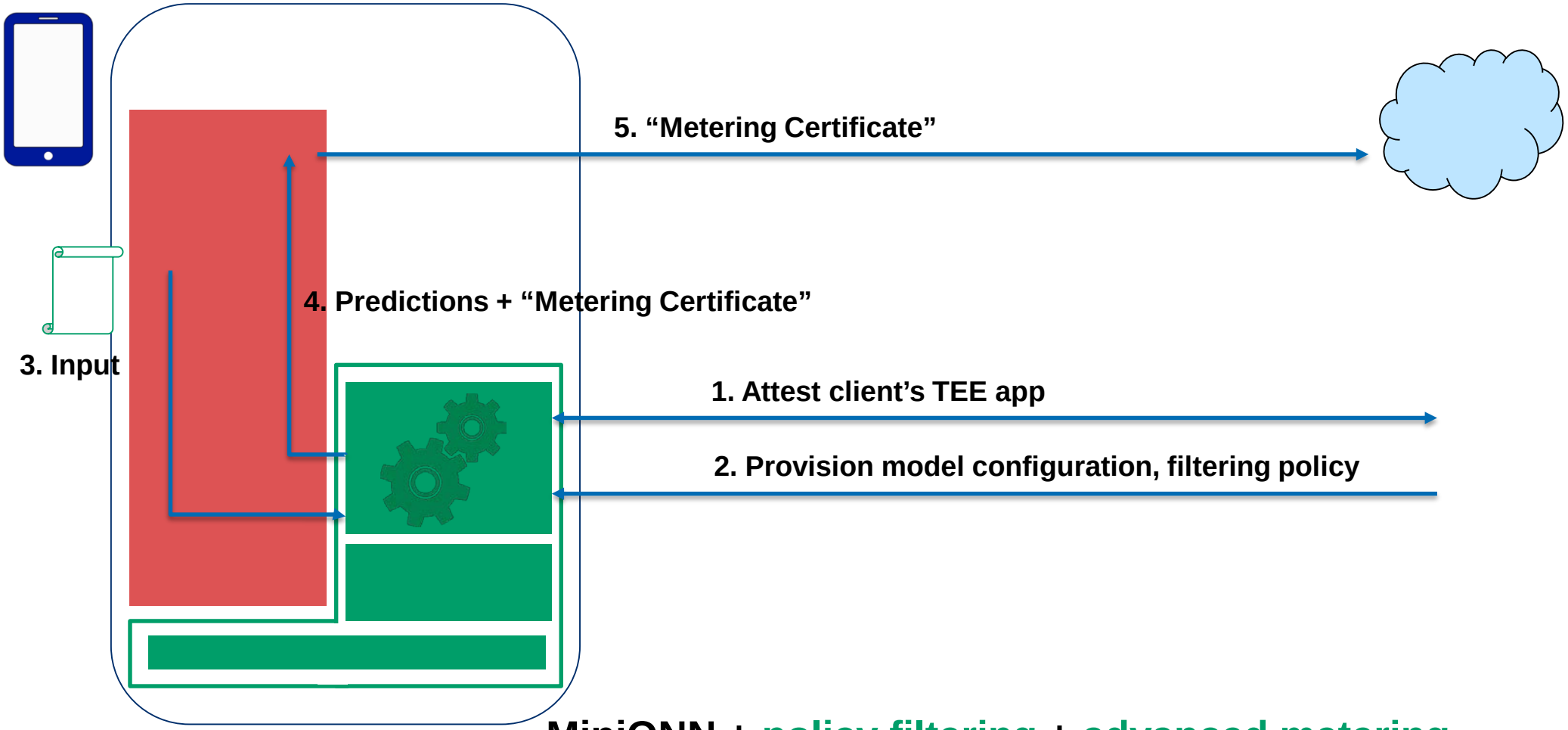
https://software.intel.com/en-us/sgx

# Using a client-side TEE to vet input



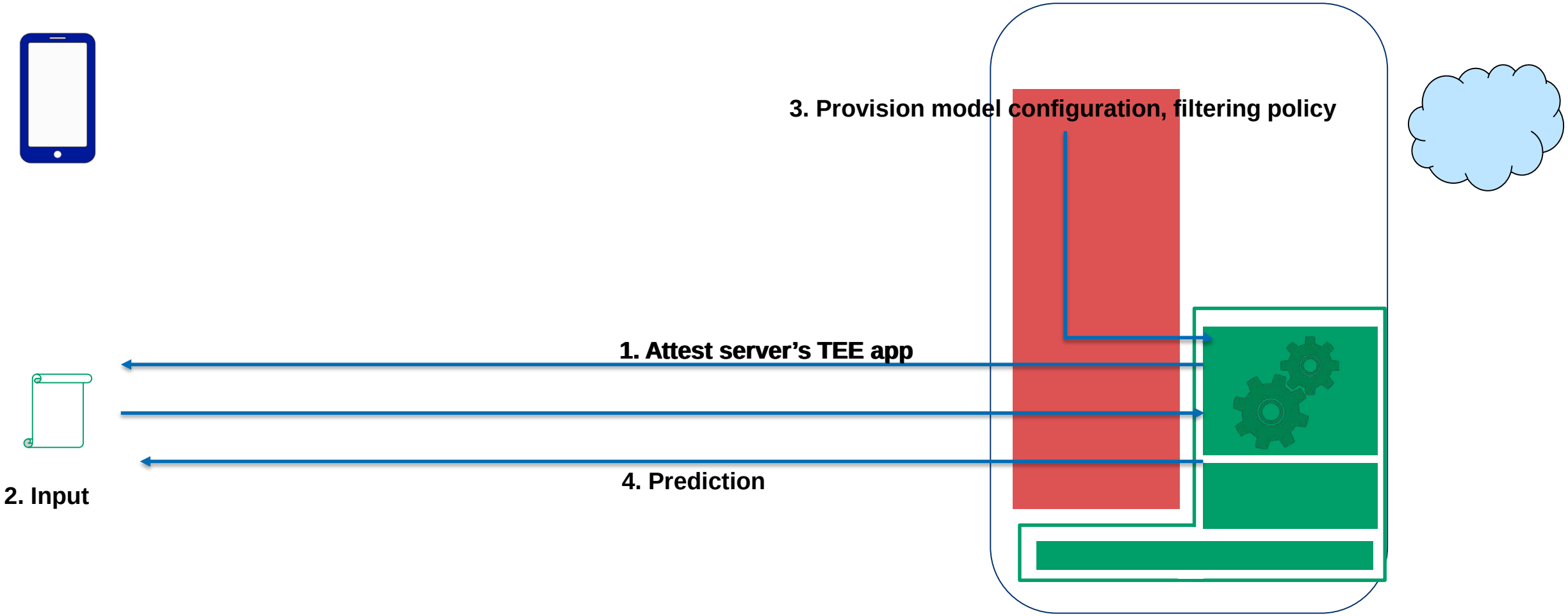

**MiniONN + policy filtering + advanced metering**

# 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

# Using a server-side TEE to run the model




MiniONN + policy filtering + advanced metering
- disconnected operation + performance + better privacy

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

ML is very fragile in adversarial settings

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