Lightweight Privacy-Preserving Machine Learning Techniques for IoT Devices

Nader Sehatbakhsh
Department of Electrical and Computer Engineering
University of California, Los Angeles

A quick overview

Introduction

- Assistant Professor at UCLA since 2020.
- Before that, got my PhD from Georgia Tech.
- Currently the director of Secure Systems and Architecture Lab.

Research Interests

- Systems security with more emphasis on hardware-assisted solutions to improve security and privacy.
- Cross-layer approach:
  - Modeling side-channels
    - Pre-silicon modeling of physical side-channels
    - Tools for side-channel leakage estimation at the binary level
  - New microarchitecture security modules by leveraging chiplet technology and emerging integration technologies.
  - IoT security and privacy.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>


ucla
TOMMY HILFIGER

What this tutorial will be about

- A brief overview of state-of-the-art solutions for ***privacy-preserving computation***.

- The emphasis will be on ***machine learning*** (mostly deep learning) as the main application, however, other applications can be considered.

- The main focus will be discussing ***lightweight*** methods but we will start from higher levels.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

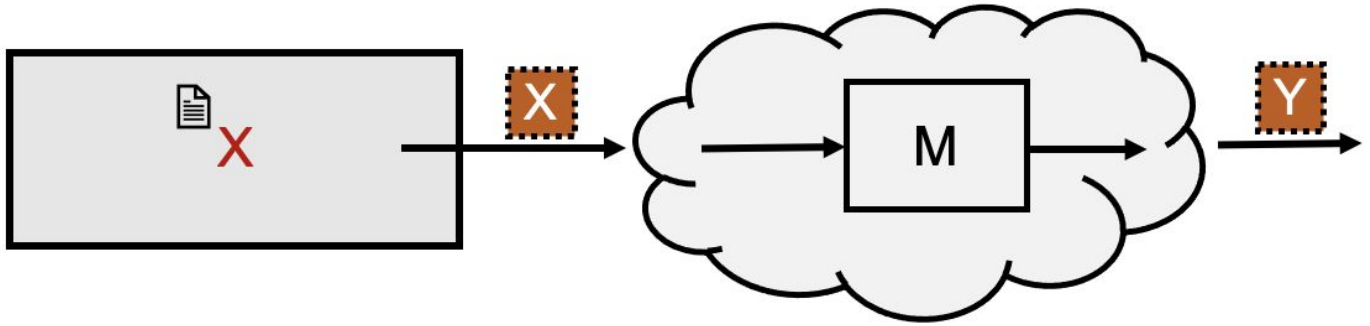
IoT Devices Status Quo

- Resource-constrained IoT devices with sensors and actuators!
- Complex “machine learning” tasks on the edge.
- Latency-sensitive or power constraints.
- Many examples: *smart home, healthcare, CPS, …*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

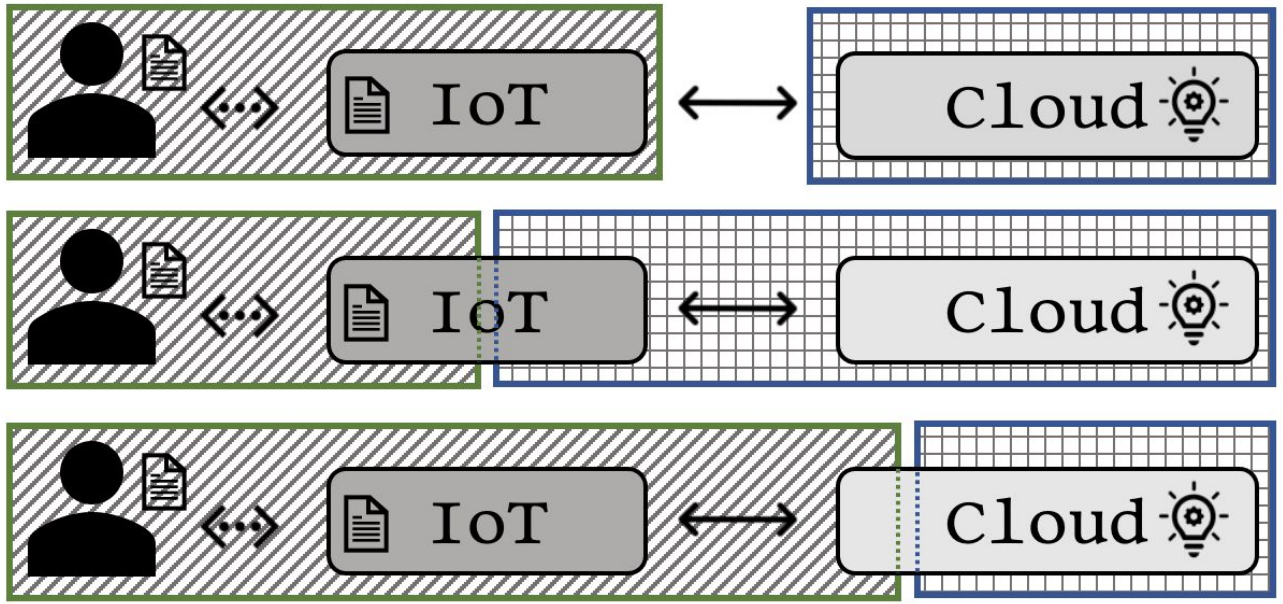
Collaborative Computation

- Collaborative IoT-cloud systems (why?)



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Trust Model



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Machine learning as a target!

ML Systems as a Target

★ *What is important for the adversary?*

- Outcome of the system (i.e., labels) → *Security*
- Data: Inputs and Model/Weights → *Privacy*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

ML Systems as a Target

★ *What is important for the adversary?*

- Outcome of the system (i.e., labels) → *Security*
- Data: Inputs and Model/Weights → *Privacy*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Why data is important?

- *Two scenarios:*
  - Input data is sensitive (e.g., patient's X-Ray)
  - Model is proprietary/IP

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

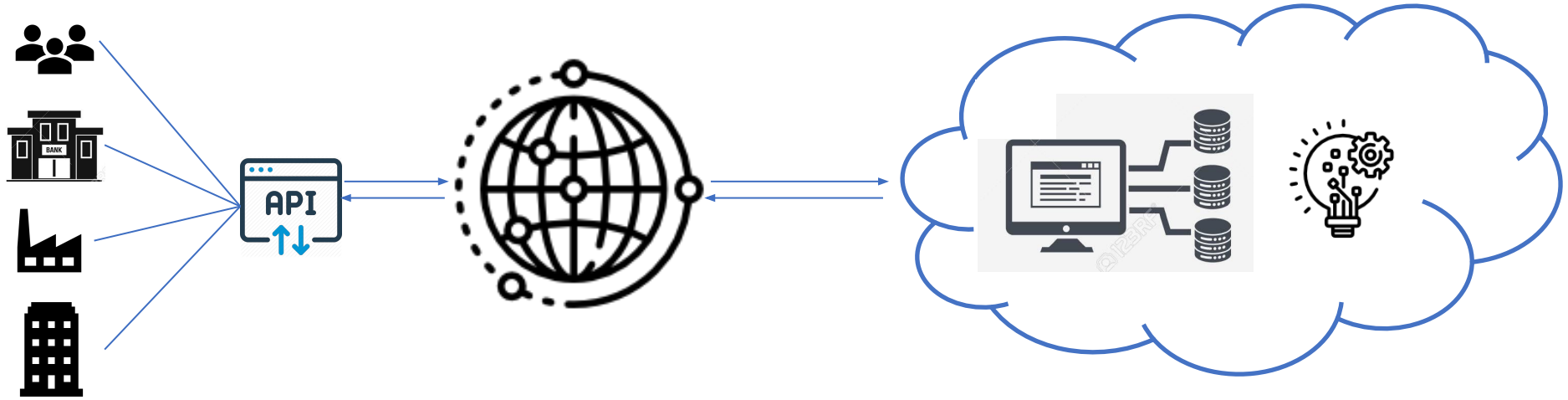
Why data is important?

- *Two scenarios:*
  - Input data is sensitive (e.g., patient's X-Ray)
  - Model is proprietary/IP

➔ *Why this is more challenging these days?*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

ML as a Service (MLaaS)



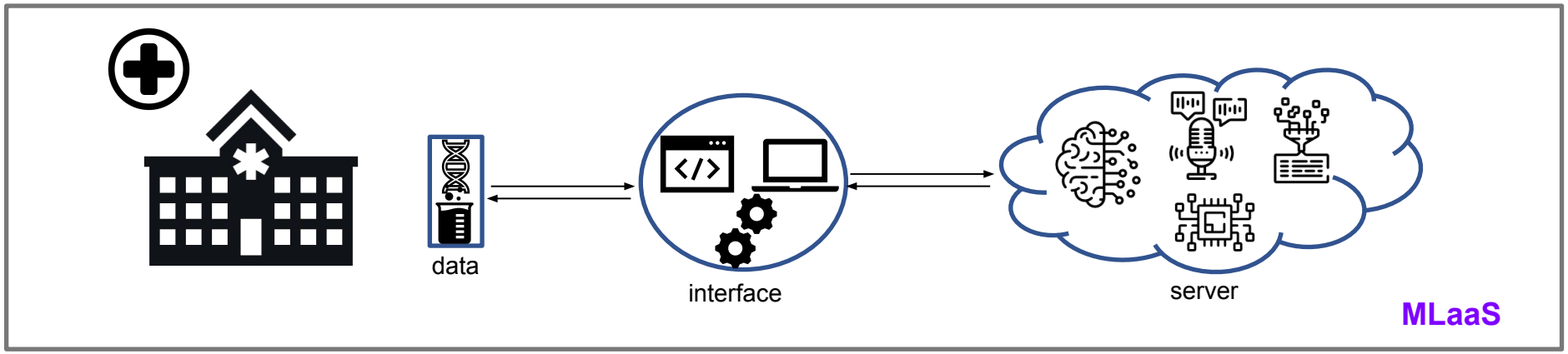
Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Who are the *players*?

1) An accurate *AI model* can be built by gathering a large dataset from *different users* and/or *companies* (called *data owners* or ***participants***) to train a machine-learning model.

2) ***Servers*** then can ***a)*** *build/train the model* (i.e., gathering the data and training the system), and ***b)*** *providing an interface* to make this trained model available to users.

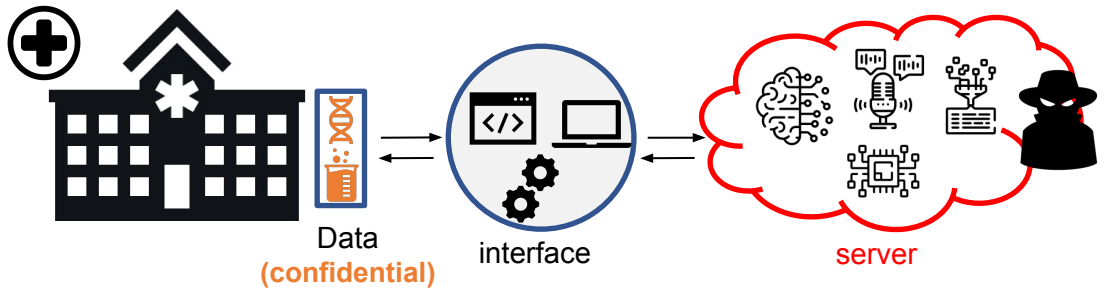
3) ***Users,*** who do not have access to such datasets, can take advantage of these trained model available on the cloud using the interface.



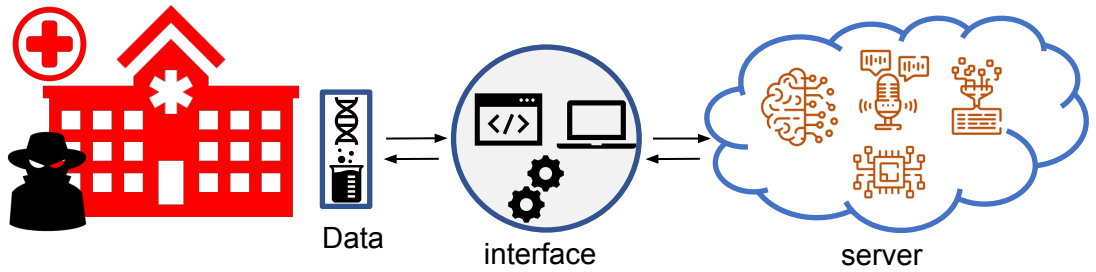
Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Attackers in MLaaS

- **Malicious Cloud/Server**: an *honest-but-curious* server has access to user's data thus ***violates*** the user's privacy, especially when the input data is *confidential*.



- **Malicious User/Participant**: can ***attack*** the system and find sensitive *training data* and/or reverse engineer the *model* and steal the server's IP (i.e., model).



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Recap

- Privacy Attacks in ML systems:
  - *Goal:* stealing data or weights
  - *How:* Through edge/cloud paradigm (MLaaS and collaboration)
  - *Adversary:* Either the user/edge or the server

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What if the user is the adversary?

★ *What a user can steal?*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What if the user is the adversary?

★ *What a user can steal?*

- Model weights
- Training data
- Sensitive attributes
- …

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Attacks

- Adversarial attacks
- Model extraction
- Membership inference
- Other

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What if the server cannot be trusted?

- *Very Common!*

➔ Users need to trust the server to send their data.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What can we do?

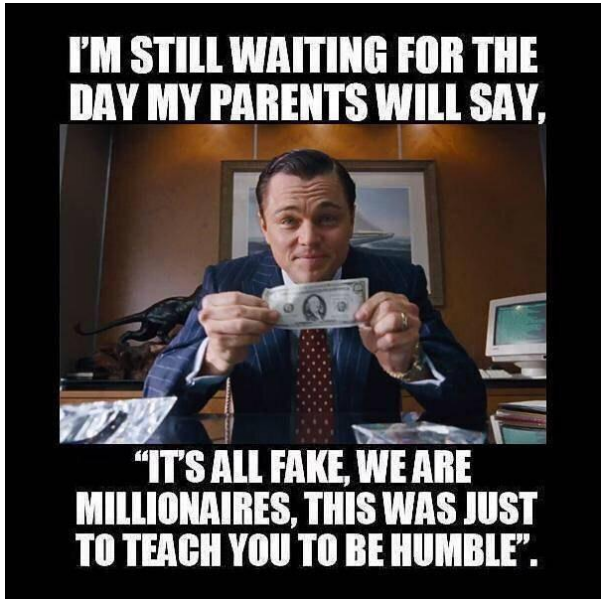
- An old problem

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Millionaire's Problem

- Alice and Bob are millionaires who are interested in knowing which of them is richer *without* revealing their actual wealth.

➔ How to solve this?



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving Computation

- Methods for computing a function where inputs are provided by 2 or more players who mutually do not trust each other or any central 3rd party.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving Computation

- Popular Solutions
  - Homomorphic encryption
  - Multi-party computation
  - Hardware-based
  - Non-cryptographic

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Let’s talk about them one-by-one …

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

A few ground rules

1. We will stay at (very) high level!
2. I will give you pointers to papers/codes/repositories so please spend some time after this tutorial to explore and dig deeper.
3. Please feel free to interrupt me and ask questions!
4. We might skip some slides if everyone is already familiar with the background.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Link to the presentation:

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving Computation

Can encryption help?

- *NO!*

➔ Encryption only protects data *at-rest*. For computation, data has to be decrypted anyway.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What if?

- Computation over encryption?

  (m_1, m_2)

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Homomorphic Encryption

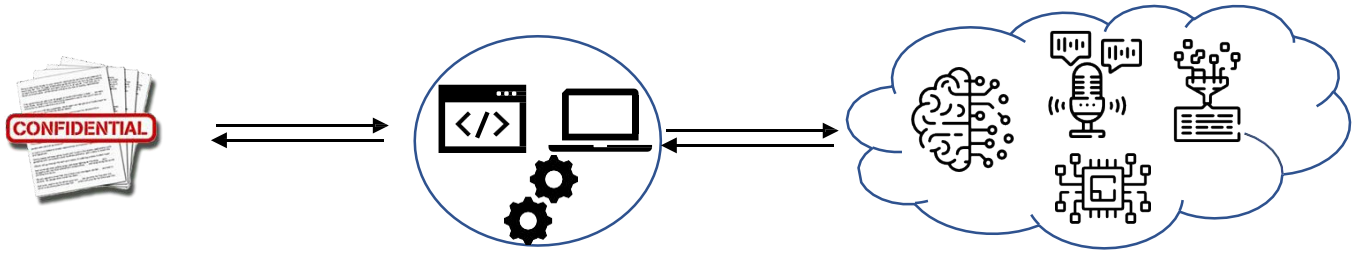
- `y = F(x)`, Edge and Server

→ Encrypt on edge, compute on server

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Homomorphic Encryption

- HE is a cryptosystem (i.e., same as AES, RSA).
- Old problem, but the main breakthrough happened in 2009.



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What is the goal?

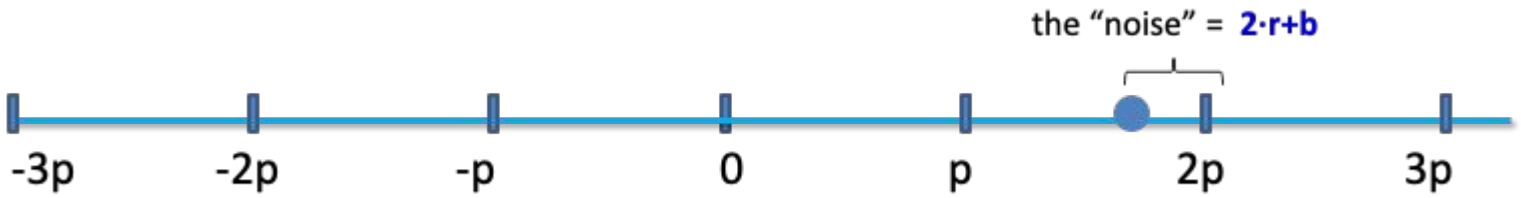
- Evaluating any *polynomial* function without revealing any information about the input
  - Implementing addition and multiplication is sufficient.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Basic Intuition

- Secret: `p`
- Noise: `r`
- Message: `b`

➔ Encryption: `c = ap + 2r + b`

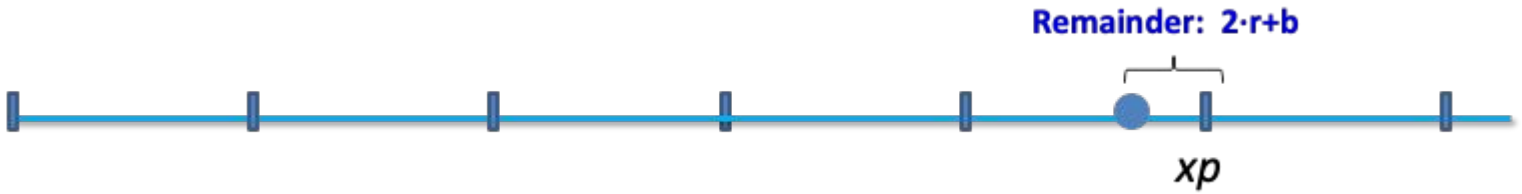


Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Basic Intuition

- Secret: `p (p` is odd`)`
- Noise: `r`
- Message: `b`

➔ decryption: `b = (c mod p)mod 2)`



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Addition and Multiplication

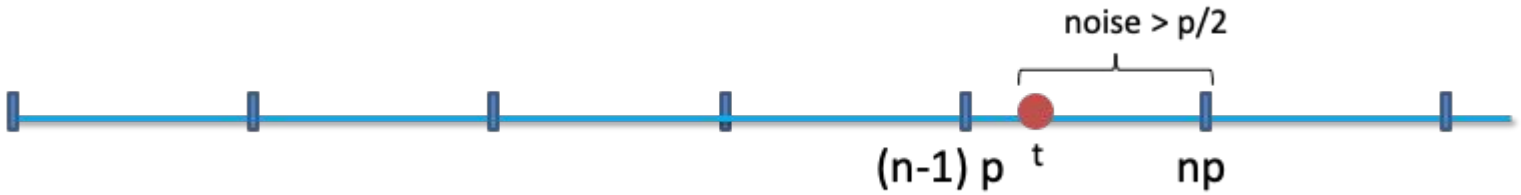
- $c_1 + c_2$

- $c_1 \times c_2$

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Noise problem!

- Noise grows with each operation



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

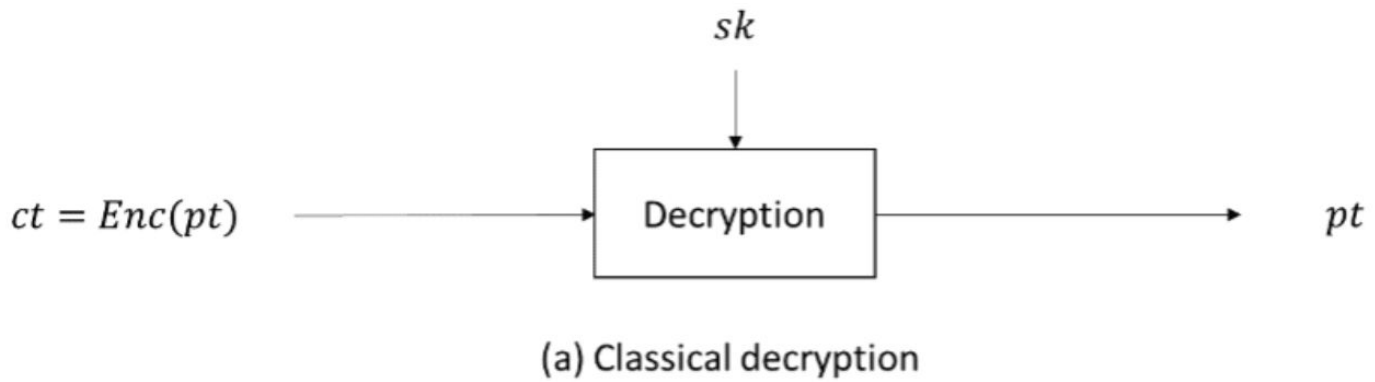
Noise problem!

- Noise grows with each operation

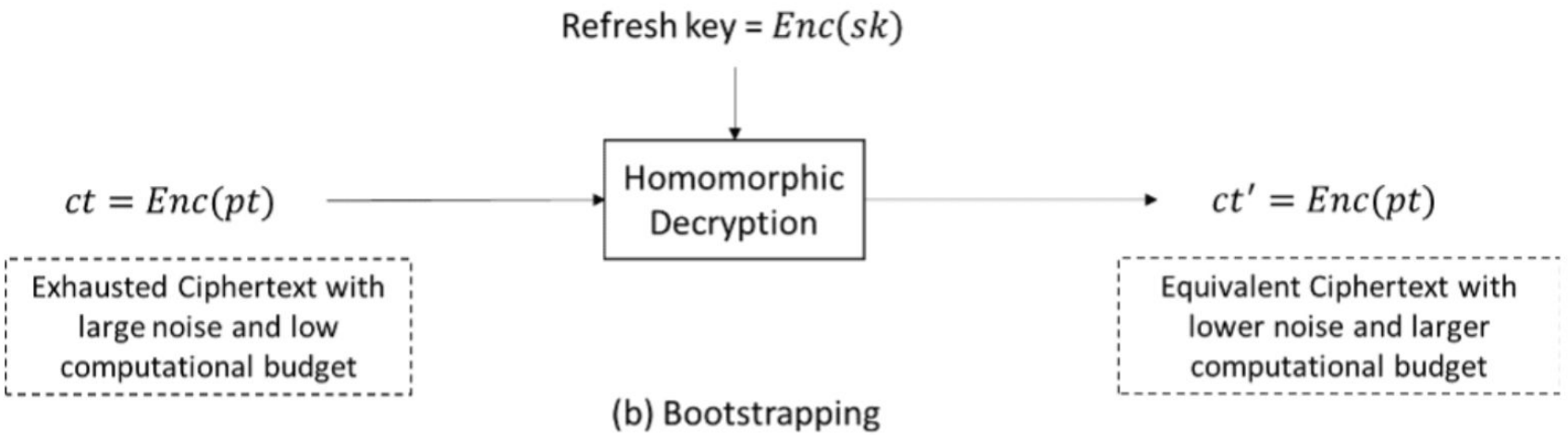
➔ Two solutions:
- Leveled/Somewhat Homomorphic
- Fully homomorphic with bootstrapping

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Bootstrapping



(a) Classical decryption



(b) Bootstrapping

Taken from: https://dualitytech.com/blog/bootstrapping-in-fully-homomorphic-encryption-fhe/

Popular Solutions

- BFV, BGV
- CKKS
- TFHE

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Code

https://github.com/microsoft/SEAL

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Main Challenge in HE

- *Performance!*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Main Challenge in HE

- *Performance!*

- Non-linear operations

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Some Numbers

- Up to five orders of magnitude slowdown!
- Parallelization, better software and compiler can help but nowhere near what we want!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Solutions?

- Specialized hardware / accelerators
- Approximation and algorithmic solutions

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

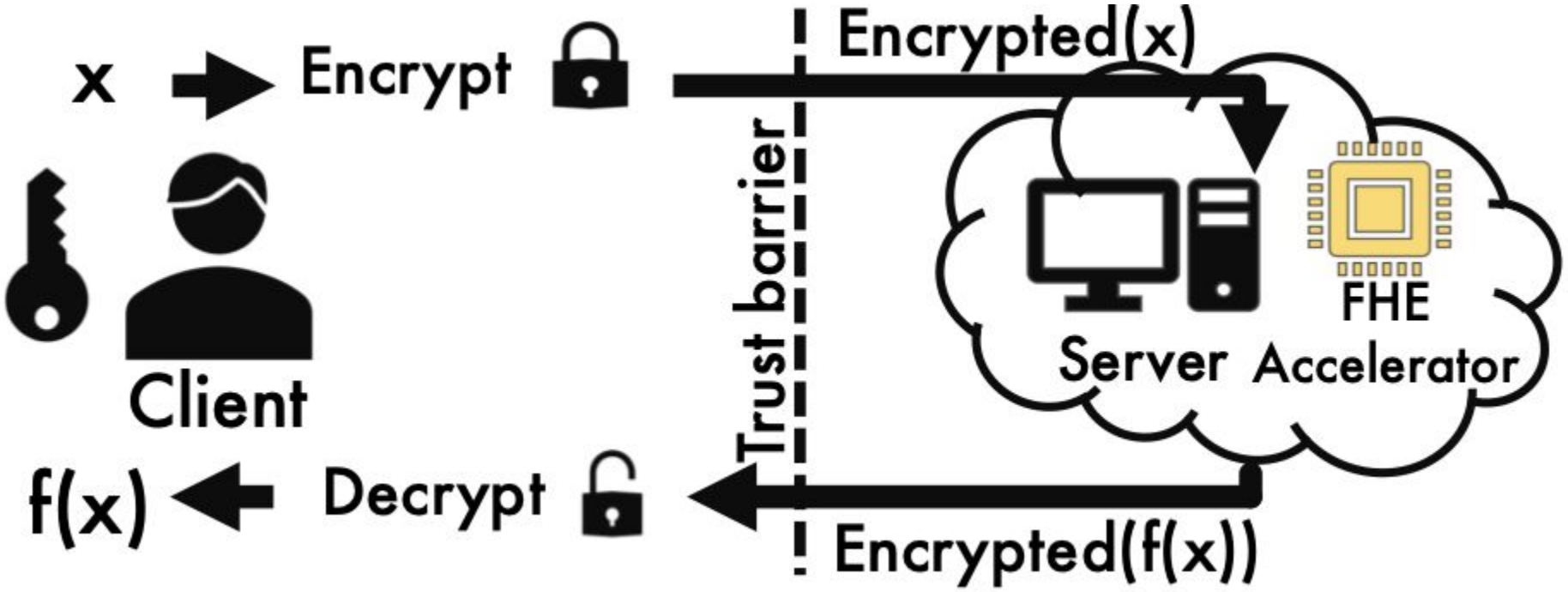
State-of-the-art

- Hardware acceleration: creating domain-specific hardware to offload the computation there!
  - We are already doing this for many other applications: crypto, machine learning, …

- Example: F1, Craterlake, HEAWS, Cheetah, …

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

System Architecture



Samardzic, Nikola, et al. "Craterlake: a hardware accelerator for efficient unbounded computation on encrypted data." *Proceedings of the 49th Annual International Symposium on Computer Architecture*. 2022.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

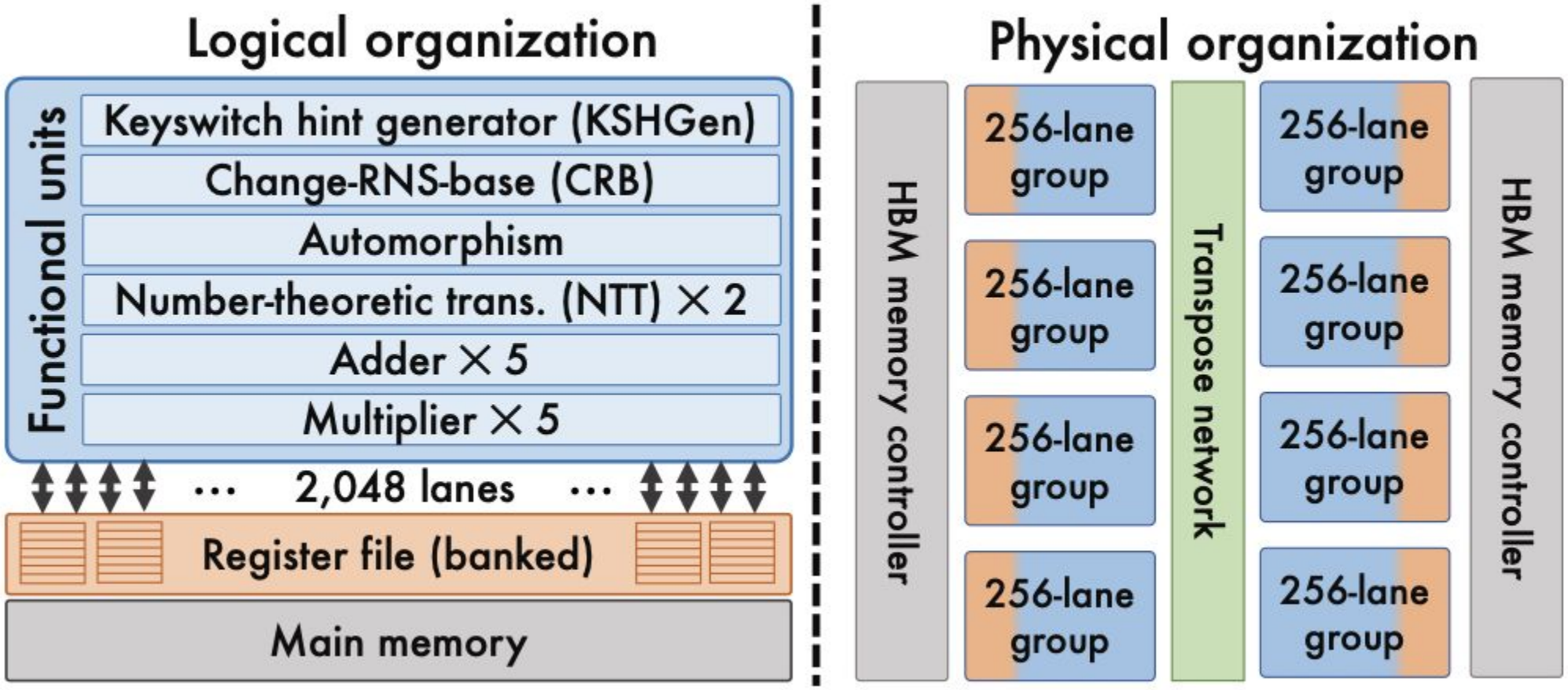
Key Factors/Considerations

- Very wide vector processing elements to utilize SIMD operations. Faster keyswitching/bootstrapping

- Better mapping, scheduling, and transformation
  - FHE is fairly deterministic so most operations can be nicely and deterministically scheduled offline → maximizing the utilization and parallelization → performance boost

- Microarchitecture optimization:
  - memory latency and bandwidth

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>



Samardzic, Nikola, et al. "Craterlake: a hardware accelerator for efficient unbounded computation on encrypted data." *Proceedings of the 49th Annual International Symposium on Computer Architecture*. 2022.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Performance Improvements

Execution time (ms) on	CraterLake	F1+	CPU	vs. F1+	vs. CPU
ResNet-20	249.45	2,693	23 min	10.8×	5,519×
Logistic Regression	119.52	639	356 s	5.34×	2,978×
LSTM	138.00	2,573	859 s	18.6×	6,225×
Packed Bootstrapping	3.91	58.3	17.2 s	14.9×	4,398×
deep gmean speedup				**11.2×**	**4,611×**
Unpacked bootstrapping	0.10	0.21	877	2.04×	8,612×
CIFAR Unencryp. Wghts.	50.50	94.1	187 s	1.86×	3,695×
MNIST Unencryp. Wghts.	0.14	0.13	561	0.97×	4,152×
MNIST Encryp. Wghts.	0.24	0.22	1369	0.88×	5,621×
shallow gmean speedup				**1.34×**	**5,220×**

What about GPUs?

- Yes and No!
  - Good for vector operations
  - Not so efficient for non-vector operations → bootstrapping, NTTs, etc.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Solutions?

- Specialized hardware / accelerators
- Approximation and algorithmic solutions

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

“LoLa” Approach

- ***High level idea:***
  - Can we approximate the operations more effectively?
  - Can we design FHE-aware networks?

- What do we get:
  - performance,
  - accuracy

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

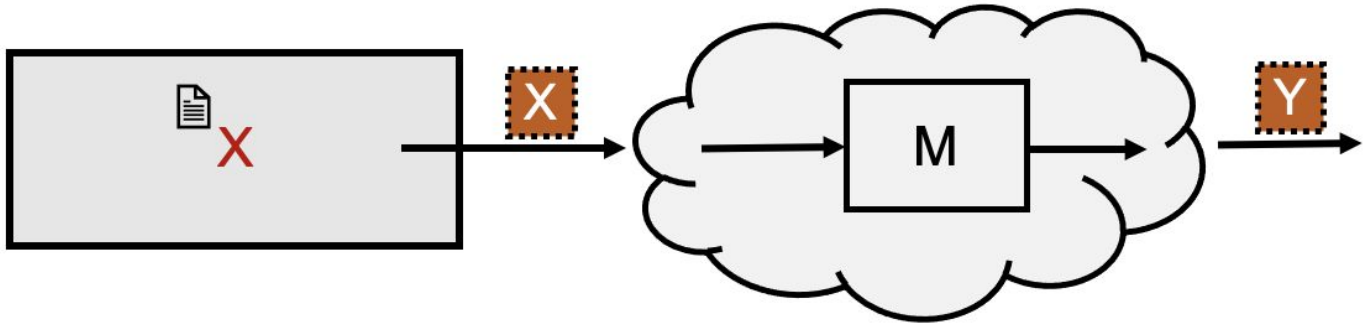
“LoLa” Approach

- State-of-the-art:
  - LoLa: Low latency privacy preserving inference
  - She: A fast and accurate deep neural network for encrypted data
  - Falcon: Fast spectral inference on encrypted data
  - Glyph: Fast and accurately training deep neural networks on encrypted data

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What is the impact on the IoT?

- Remember the collaborative IoT-cloud systems

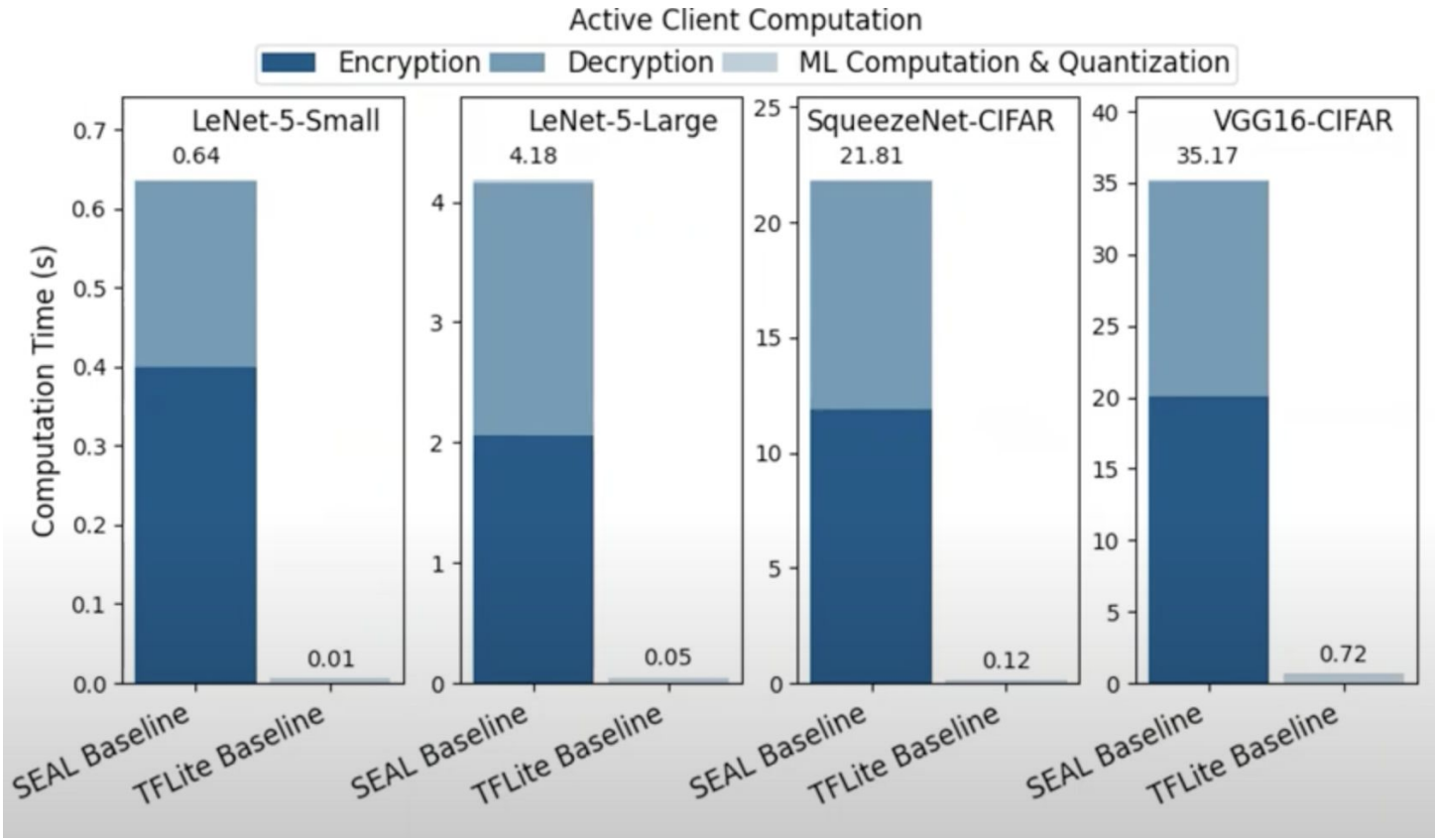


- Encryption/decryption burden on the IoT.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

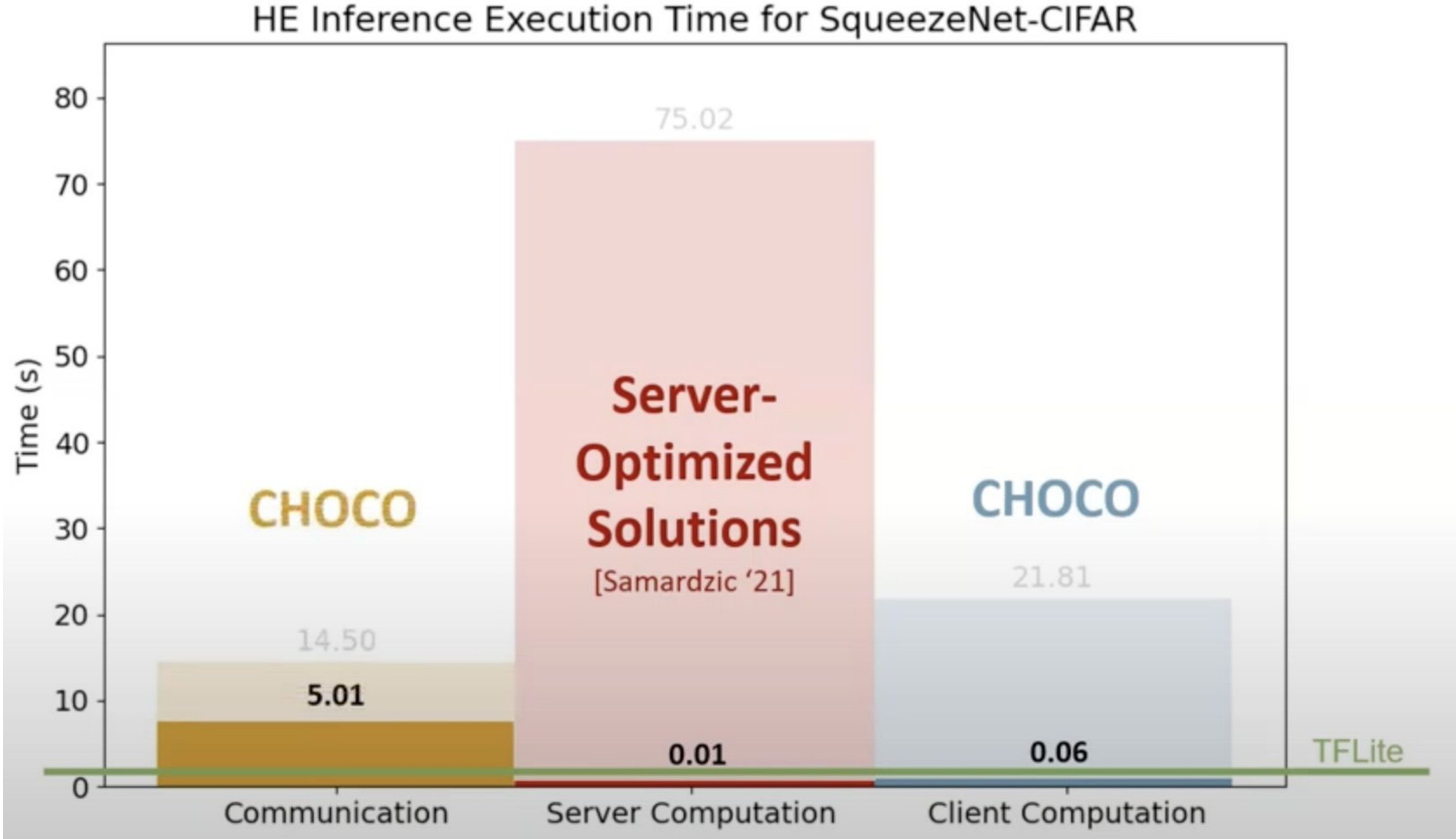
Client-Optimized Approach?

- How about having a tiny accelerator on the IoT side to encrypt/decrypt?



van der Hagen, McKenzie, and Brandon Lucia. "Client-optimized algorithms and acceleration for encrypted compute offloading." *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2022.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>


HE Inference Execution Time for SqueezeNet-CIFAR
Time (s)
80
70
60
50
40
30
20
10
0
75.02
Server-Optimized Solutions
[Samardzic '21]
CHOCO
14.50
5.01
CHOCO
21.81
0.01
0.06
TFLite
Communication
Server Computation
Client Computation

Summary

- FHE is great but slowwww!
- Lots of efforts in the past few years to address the performance issue: accelerators, algorithms, tool, IoT-aware, …
- We have closed the gap but still long way to go!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What else?

- Multi-party computation (MPC)
  - How about distributing data to multiple users → we are protected as long as only x<th are colluding!

- What are the advantages?
  - Non-linearity → accuracy
  - Lower overhead

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

A simple example

★ How to find the sum of numbers without revealing individual numbers?

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Sum of numbers in MPC

- N players, 1 is malicious

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Sum of numbers in MPC

- N players, 1 is malicious

➔ Assumption matters *(honest-but-curious)*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

A more generic solution?

- Quite a few!
  - *Yao's Garbled Circuit*
  - *GMW Protocol*
  - *Secret sharing*

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

- A protocol for ***two parties*** to collaboratively compute a circuit/gates.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

- A protocol for ***two parties*** to collaboratively compute a circuit/gates.

➔ Main Observations: Any function is a combination of gates and any gates can be shown as a truth table.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- AND gate:
  - Ginny and Evan want to find the output.
  - Ginny has input g and Evan has input e.

g	e	output $g \wedge e$
0	0	0
0	1	0
1	0	0
1	1	1

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- AND gate:
  - Ginny and Evan want to find the output.
  - Ginny has input g and Evan has input e.

➔ The first step is called *garbling*.

(either Evan or Ginny does this)

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- AND gate:
  - Ginny and Evan want to find the output.
  - Ginny has input g and Evan has input e.

➔ The first step is called *garbling*.

(either Evan or Ginny does this)

1. Create a random label for each input. (W_i)
2. Encrypt the output based on the labels (two-key encryption).

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- AND gate:
  - Ginny and Evan want to find the output.
  - Ginny has input g and Evan has input e.

➔ The first step is called *garbling*

(either Evan or Ginny does this)

1. Create a random label for each input. (W_i)
2. Encrypt the output based on the labels.

g	e	output $g \wedge e$
0	0	0
0	1	0
1	0	0
1	1	1

$\Rightarrow$

garbled output
$\mathsf{Enc}(H(W_G^0, W_E^0), 0)$
$\mathsf{Enc}(H(W_G^0, W_E^1), 0)$
$\mathsf{Enc}(H(W_G^1, W_E^0), 0)$
$\mathsf{Enc}(H(W_G^1, W_E^1), 1)$

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- AND gate:
  - Ginny and Evan want to find the output.
  - Ginny has input g and Evan has input e.

➔ The first step is called *garbling*.

(either Evan or Ginny does this)

1. Create a random label for each input. (W_i)
2. Encrypt the output based on the labels.
3. The garbler then creates a random permutation.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- Let's assume Ginny does the garbling. She then sends the table to Evan. She also send her (random-looking) input W_g^i.

- *Evan* now needs to do the evaluation.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- Let's assume Ginny does the garbling. She then sends the table to Evan. She also send her (random-looking) input W_g^i.

- *Evan* now needs to do the evaluation.
  - For evaluation, Evan has W_g^i but also needs W_e^i

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- Let's assume Ginny does the garbling. She then sends the table to Evan. She also send her (random-looking) input W_g^i.

- *Evan* now needs to do the evaluation.
  - For evaluation, Evan has W_g^i but also needs W_e^i
  - Evan achieves that through a secure protocol called *oblivious transfer*.
    (why it should be secure?)

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Oblivious Transfer

- A protocol in which a sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece (if any) has been transferred.
  - Think about the boxes and letters …

★ How OT works?

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Oblivious Transfer

- Alice has two messages m_1 and m_2 and Bob wants one of them.
  1. Alice creates two random values x_1 and x_2 and send both to Bob.
  2. Bob creates a random value k and based on the message he wants ($b=1$ or 2) send $v = enc(k) + x_b$.
  3. Alice computes $k_1 = dec(v - x_1)$ and $k_2 = dec(v-x_2)$
     then sends $t_1 = m_1 + k_1$ and $t_2=m_2+k_2$.
  4. Bob knows which k_i is valid and computes $m_b = t_b - k_b$.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Yao's Garbled Circuit (GC)

Example

- Evan now has both keys (labels), evaluates all the rows and sends the values. Only one out of 4 is meaningful.

- Alice compares the values with labels and announces the final result.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Coding

- https://github.com/ojroques/garbled-circuit

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Limitations?

- Multiple nodes are now needed to be involved!
  - IoT and cloud?
  - Multiple nodes in the cloud?

- Communication overhead

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What is the state-of-the-art?

1. CRYPTEN: Secure Multi-Party Computation Meets Machine Learning

2. CRYPTGPU: Fast Privacy-Preserving Machine Learning on the GPU

3. Piranha: A GPU Platform for Secure Computation

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

One more: how about hybrid?

- The “Gazelle” approach!
  - Let’s combine FHE (various algorithms) with MPC (various algorithms) → the combination will likely get the best of both worlds!

- Need proper “converters”!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

State-of-the-art hybrid methods

1- GAZELLE: A low latency framework for secure neural network inference.

2- Delphi: A Cryptographic Inference Service for Neural Networks

3- Muse: Secure Inference Resilient to Malicious Clients.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What about training?

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Decentralized Machine learning

★ Idea: Distribute data between nodes (either naturally or synthetically), then share the intermediate values (e.g., gradients), instead of raw values to build a centralized model.

- Federated learning
- Split learning
- Instance learning
- Differential privacy
- …

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Decentralized Machine learning

- Challenges:
  - Privacy guarantees
  - Data Poisoning
  - Membership inference
  - …

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Summary: Privacy-Preserving ML

- Homomorphic encryption
  - BFV, CKKS, TFHE, …
- Multi-party computation
  - GC, GMW, SS, …

➔ Provable security based on hard theories.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Summary: Privacy-Preserving ML

- Homomorphic encryption
  - BFV, CKKS, TFHE, …

- Multi-party computation
  - GC, GMW, SS, …

➔ Provable security based on hard theories.
➔ Extremely large overhead!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving ML

- Alternative Options?
  - Trusted Execution Environment (TEEs)

→ Confidential computing!

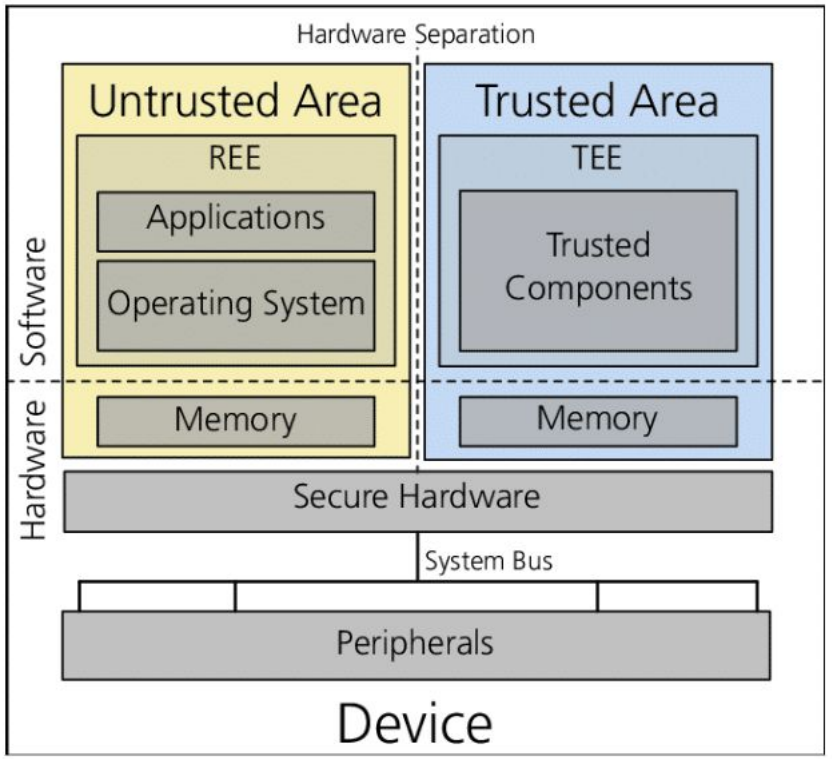
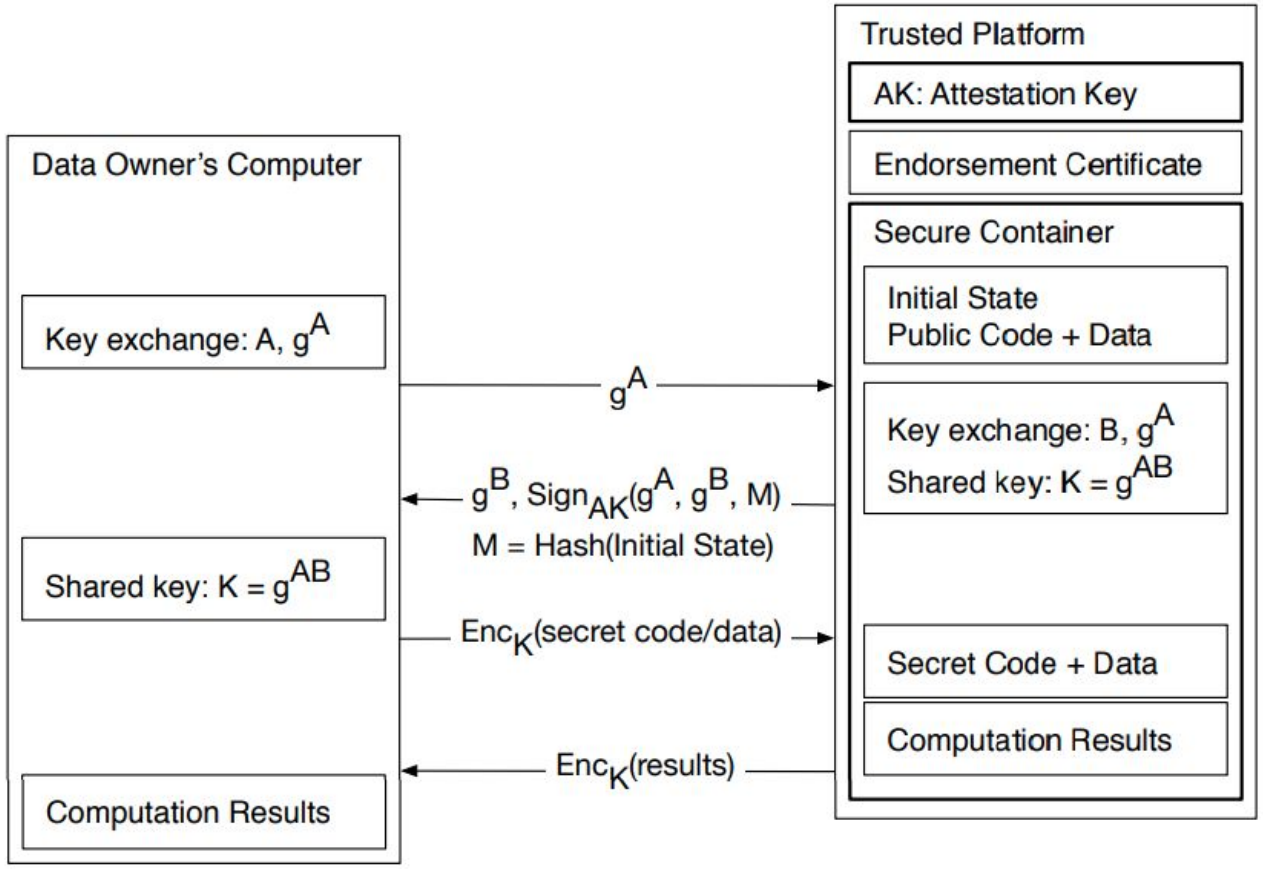


Image was taken from [1]

Confidential Computing



Costan, Victor, and Srinivas Devadas. *Intel SGX Explained, Cryptology ePrint Archive, Report 2016/086*. Vol. 86. Cryptology ePrint Archive Report 2016, 2016.

Privacy-Preserving ML

- Alternative Options?
  - Trusted Execution Environment (TEEs)

➔ Much smaller overhead! Computing in plaintext domain!
➔ Secure! (except against side-channels)

Image was taken from [1]

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving ML

- Alternative Options?
  - Trusted Execution Environment (TEEs)

➔ Much smaller overhead!
➔ Secure! (except against side-channels)
➔ Limited performance and storage/area
➔ Not always available – especially for edge devices.

Image was taken from [1]

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Threat Model and Assumption

- Security/privacy is preserved if hardware is trusted and protocol is implemented correctly!

- Different from MPC/FHE

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

TEE Approach: Limitations

- Limited memory/hardware → okay for smaller models but infeasible for large models!

→ Can't fit everything fully on TEEs.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

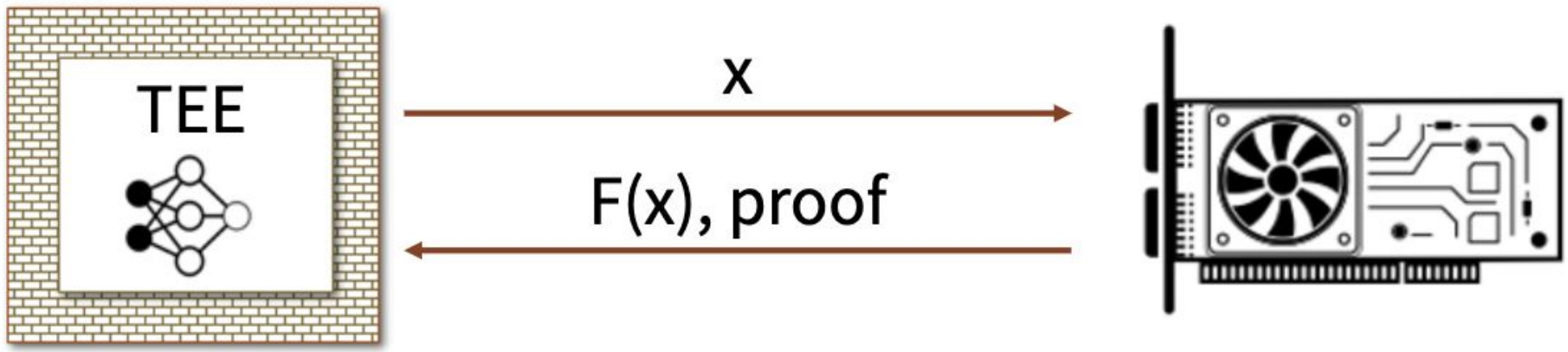
How to improve this?

- The ‘Slalom’ approach!

- How about offloading some computation out of TEE?
  - This, of course, needs to be secure and privacy-preserving!
  - Combining TEE with MPC, encryption, masking, etc.

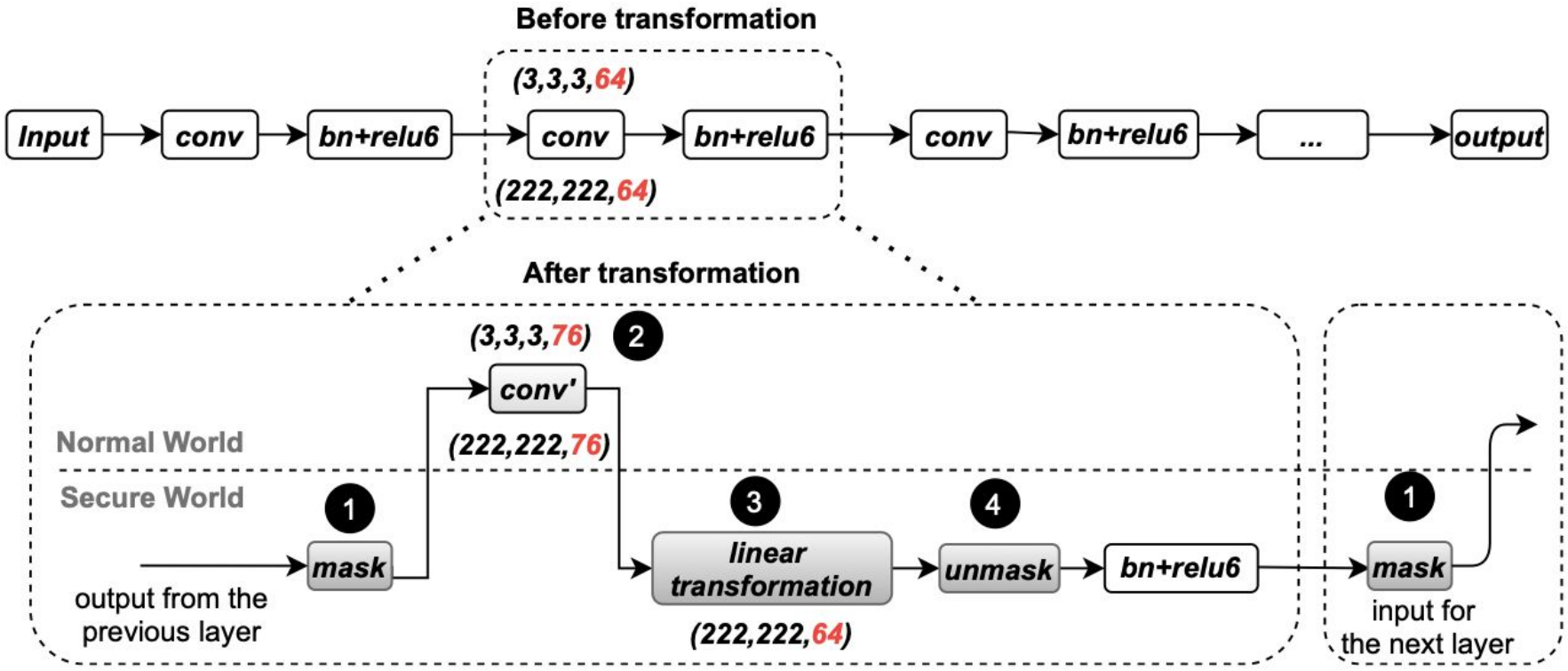
"How do we efficiently leverage TEEs for secure machine learning computations?"

Idea: outsource work to *collocated*, *faster* but *untrusted* device and verify results



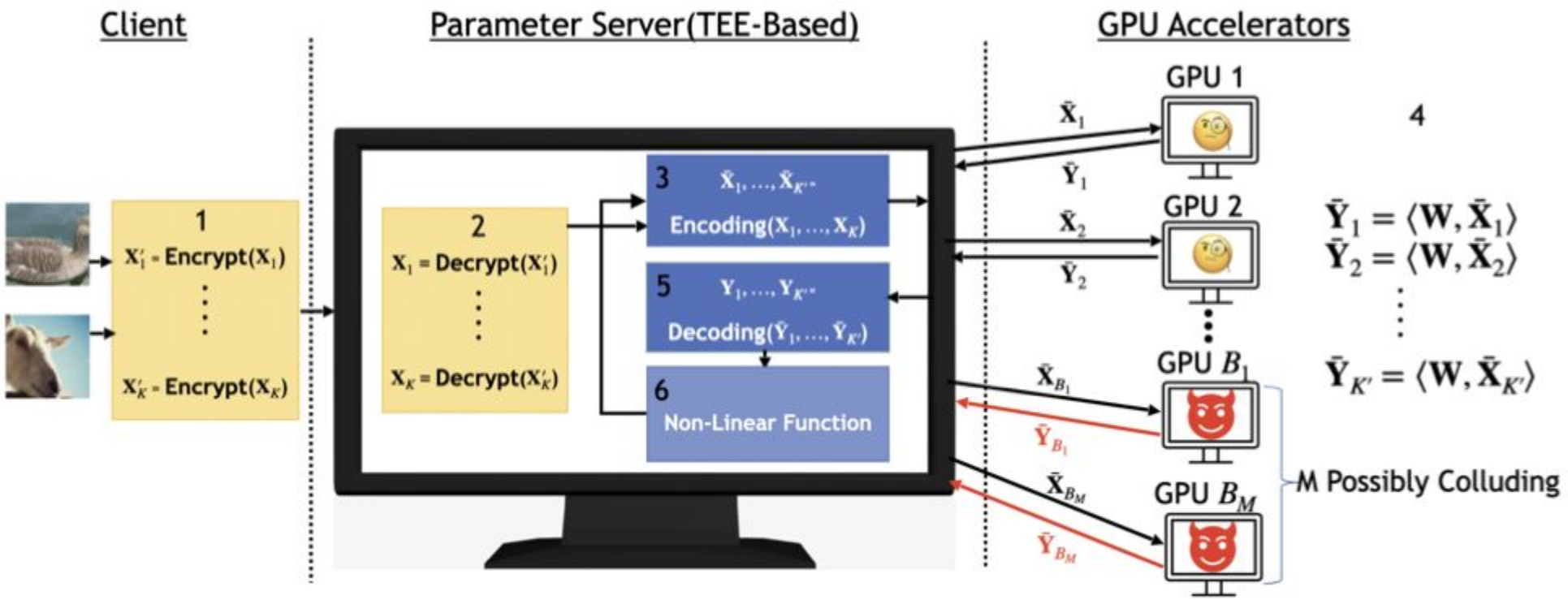
	Computations	Required gap	Privacy
Verifiable ASICs (Wahby et al., 2016)	Arithmetic circuits	~ 8 orders of magnitude	No
Slalom	**DNN inference**	**~ 1-2 orders**	**"Yes"**

ShadowNets



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

DarKnights



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Considerations

- Input vs. weights privacy
- Integrity
- GPU vs. Accelerators vs. CPU

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Impact on IoT

- Minimal work is needed on the IoT/client side
- Data is still needed to be encrypted but usually using a symmetric key, so more lightweight compared to FHE.

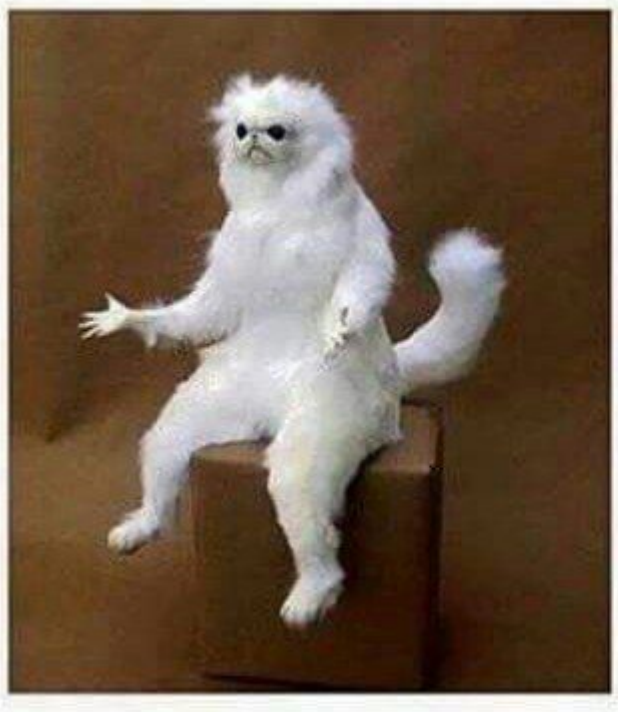
Summary

- Compared to MPC/FHE, TEE-based solutions work directly on the plaintext. The challenge, however, is more on the implementation side and addressing the hardware limitations!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving ML

- Any other options??



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Any other options?

- Non-cryptographic methods!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Any other options?

- Non-cryptographic methods!
  - *Encoding* (e.g., adversarial feature learning).
  - *Ad-hoc encryption* (e.g., Insta-Hide)

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Any other options?

- Non-cryptographic methods!
  - *Encoding* (e.g., adversarial feature learning).
  - *Ad-hoc encryption* (e.g., Insta-Hide)

➔ Advantages vs. disadvantages?

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving ML

- Encoding
  - Noise
  - Obfuscation

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Privacy-Preserving ML

- Encoding
  - Noise
  - Obfuscation

➔ Great performance!
➔ No hard theory!

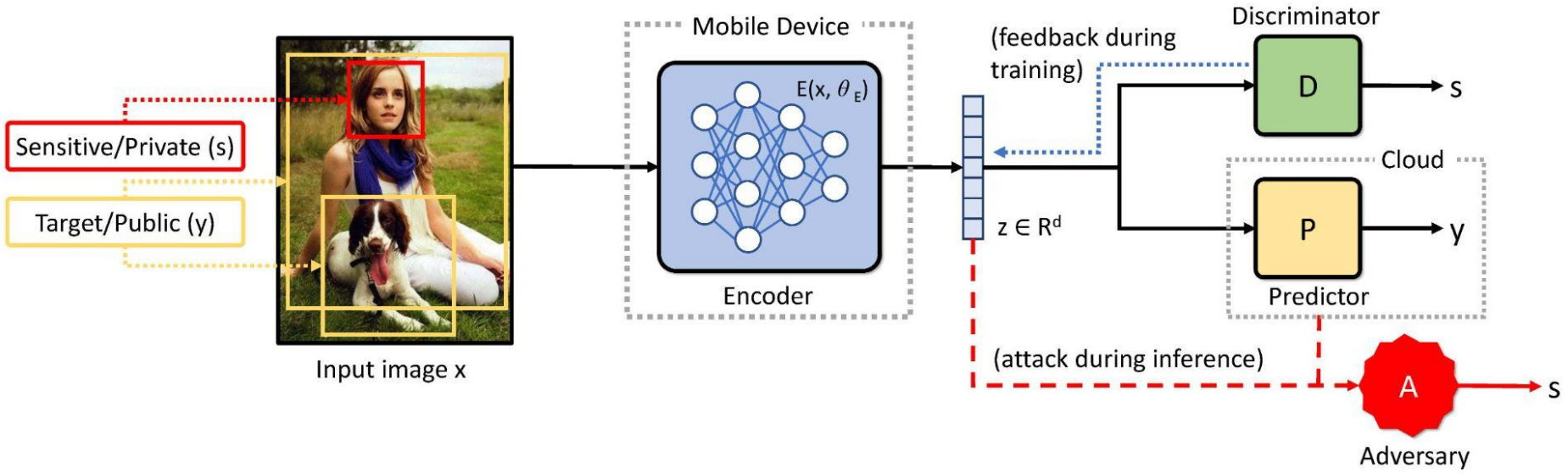
Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

A Potential Candidate

- Adversarial Representation Learning
  - Idea: Divide the classification task to two parts: *sensitive* and *public*. Train the model to remove sensitive features while retaining other important features.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

ARL Problem



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

ARL Problem

- Main Components:
  - Encoder: sits on the trusted device (edge). A *lightweight* network that is trained to remove sensitive features.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

ARL Problem

- Main Components:
  - Encoder: sits on the trusted device (edge). A *lightweight* network that is trained to remove sensitive features.
  - Predictor: sits on the cloud and performs the main classification task.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

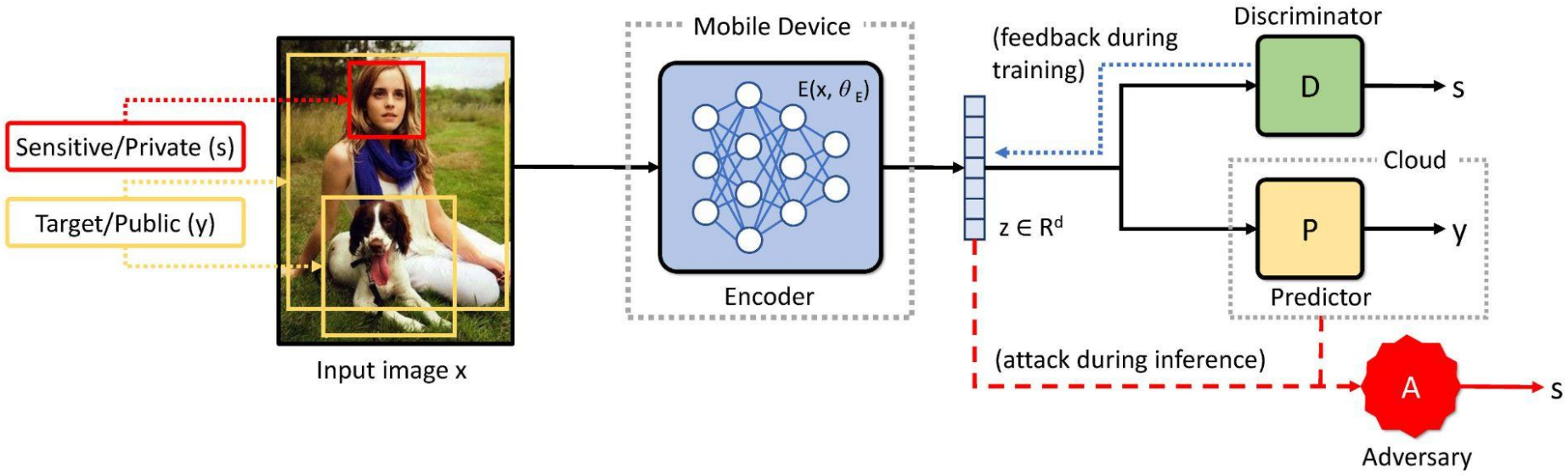
ARL Problem

- Main Components:
  - Encoder: sits on the trusted device (edge). A *lightweight* network that is trained to remove sensitive features.
  - Predictor: sits on the cloud and performs the main classification task.

  - Discriminator: proxy to attacker, which is used during the training to measure privacy.
  - Attacker: the actual network used by an adversary during the attack. May or may not be the same as discriminator.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

ARL Problem



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

How to Design ARL?

- Min-Max game between discriminator and predictor.

$$\min_{E,P} \max_{D} \; V(E,P,D), \tag{1}$$

where

$$V(E,P,D) = \mathop{\mathbb{E}}_{x,s,y\sim p(x,s,y)} [\alpha \log q_D(s|h=E(x,s)) - \log q_P(y|h=E(x,s))].$$

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

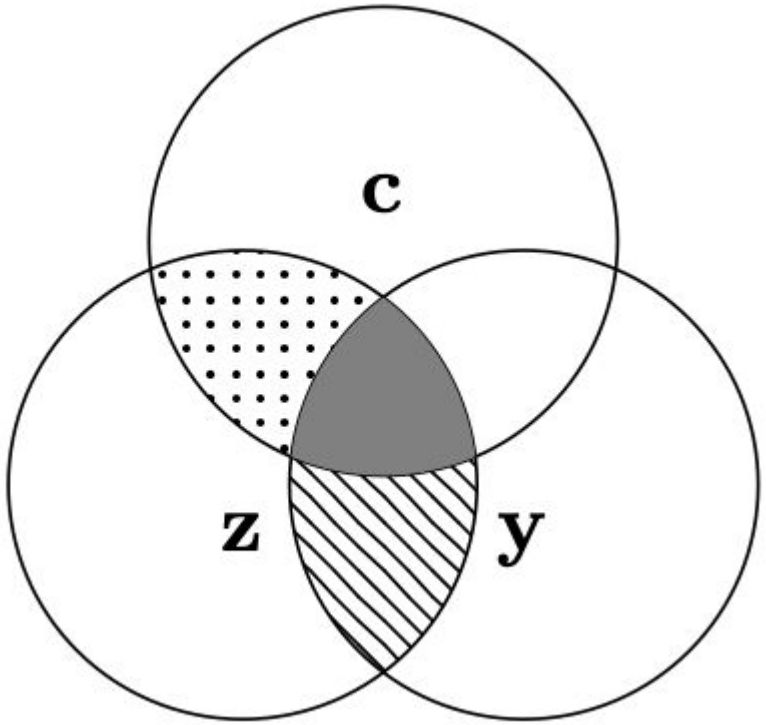
Information-Theoric view

$$\mathcal{O}_1: \max_q I(\mathbf{y}:\mathbf{z}) \;\; \text{s.t.} \;\; I(\mathbf{z}:\mathbf{c}) \leq \delta$$

$$\text{or,} \;\; \max_q I(\mathbf{y}:\mathbf{z}) - \beta I(\mathbf{z}:\mathbf{c})$$

$$\mathcal{O}_2: \max_q I(\mathbf{y}:\mathbf{z}|\mathbf{c}) \;\; \text{s.t.} \;\; I(\mathbf{z}:\mathbf{c}) \leq \delta$$

$$\text{or,} \;\; \max_q I(\mathbf{y}:\mathbf{z}|\mathbf{c}) - \beta I(\mathbf{z}:\mathbf{c})$$



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Limitations

- Correlation
- Private vs. public
- Adversary modeling

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Encoder Size

- Larger encoders are needed for more privacy.
- Particularly, a major challenge for *complex applications.*

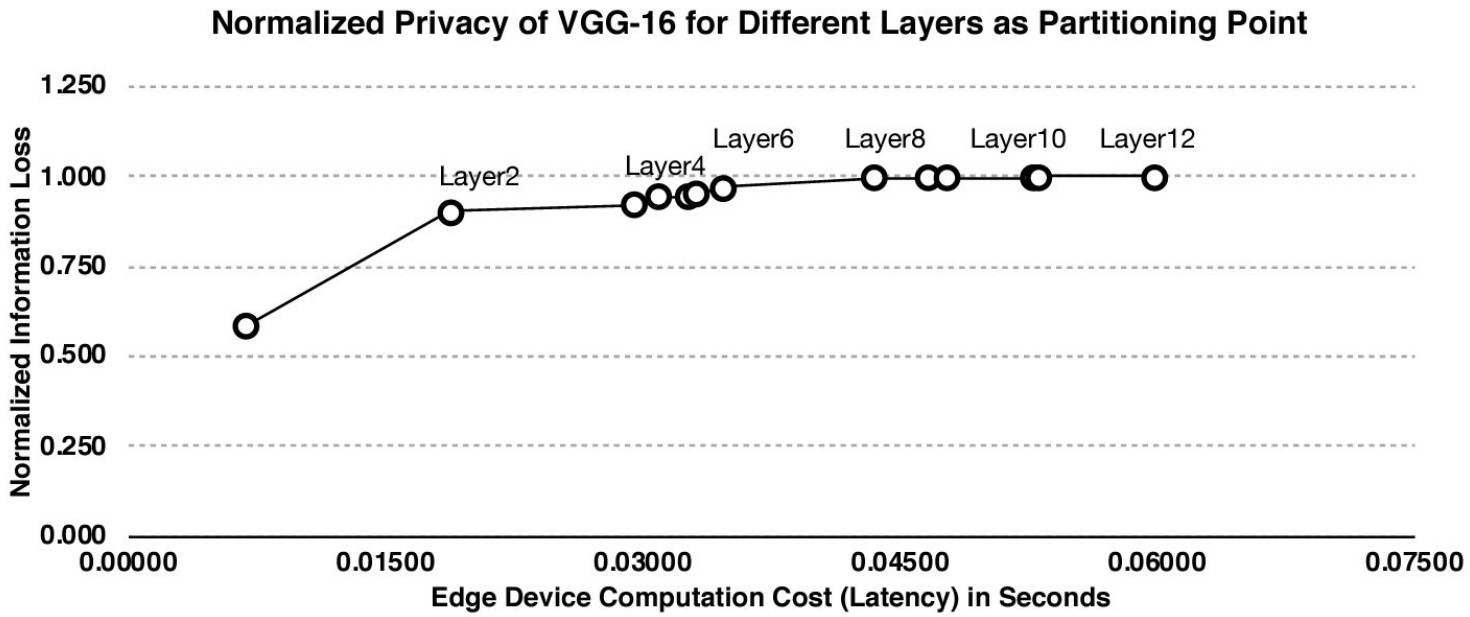


Image was taken from [2]

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

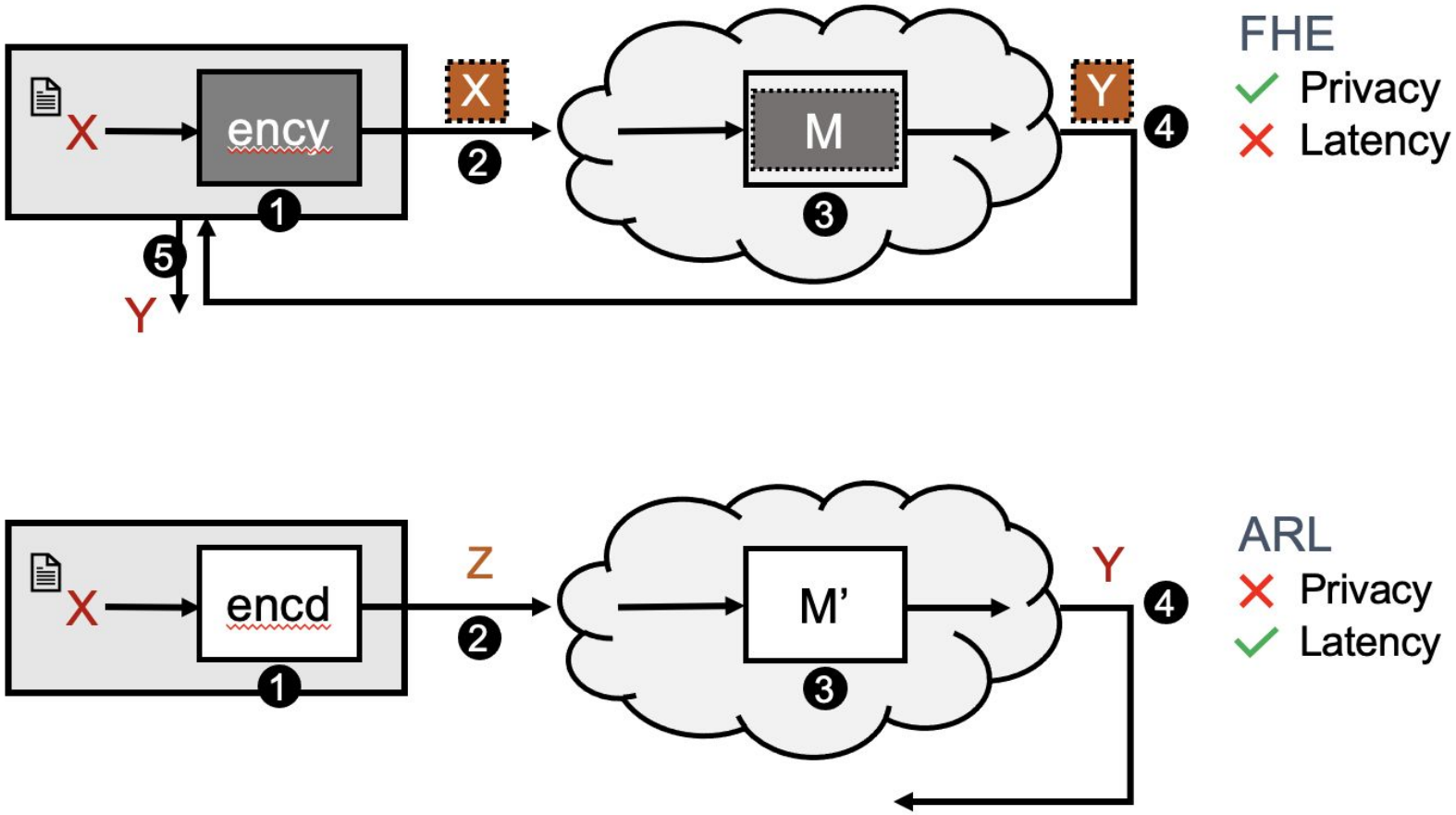
Why can’t we use larger encoders?

- The encoder has to be implemented on the IoT side
  - More power
  - More latency
  - …

- But we need larger encoders …

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

FHE vs. ARL?



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Quick Summary – Status Quo

- Need privacy in collaborative settings.
- FHE and similar methods have high latency.
- Encoding can help but only for small networks.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

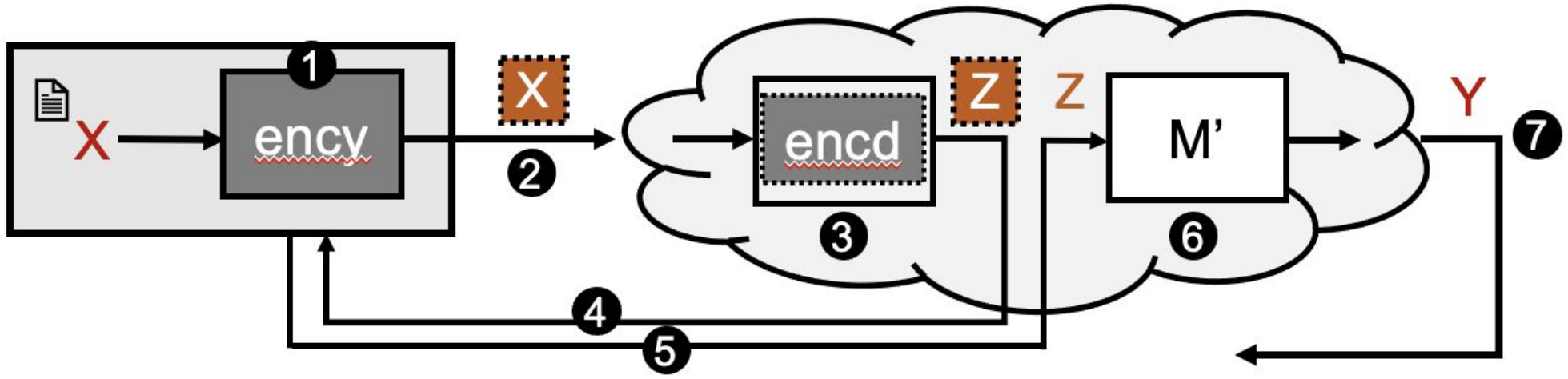
Enc + Enc

- Why don’t we encode and encrypt?

  ★ Key Idea: Offload the encoder to the cloud but use FHE to protect its privacy.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Enc + Enc



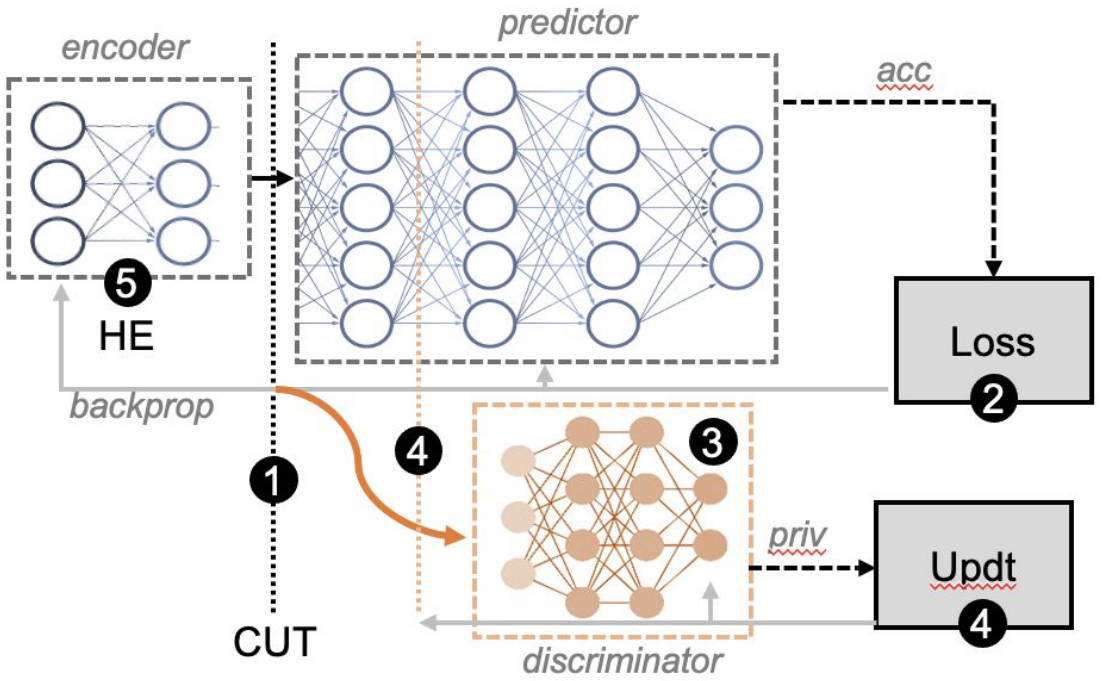
(c) Our Method: ✓Privacy, ✓Latency

Orient: Enc^2

- What are the advantages:
  - Encoding is completely offloaded to the cloud, so no overhead to the IoT device.
  - Only the encoder needs to be evaluated homomorphically (i.e., between 10-25% of the network), so much smaller overhead compared to SOTA.

- Disadvantages:
  - Another round trip for the message.

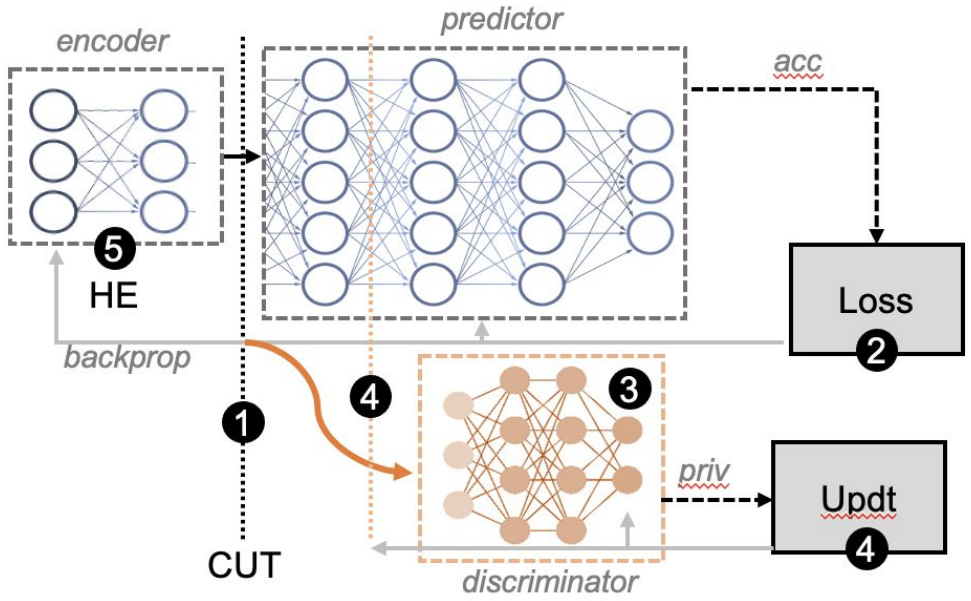
System Design



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

System Design

- Step 1: finding a cut!

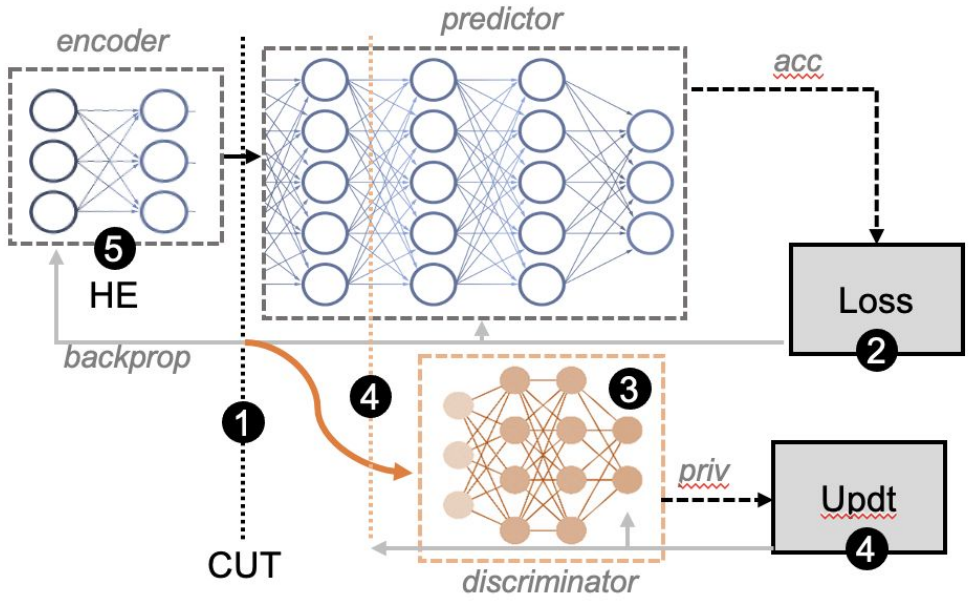


Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

System Design

- Step 1: finding a cut!
  - The smallest that can achieve some privacy!

- What is the loss function?



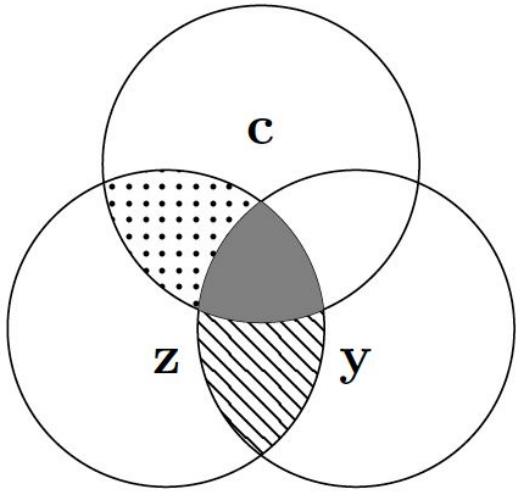
Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Loss Function

- Various methods
  - Adversarial Forgetting
  - ARL based on mutual information and entropy [4]

$$\mathcal{O}_2: \max_q I(\mathbf{y}:\mathbf{z}|\mathbf{c}) \;\; \text{s.t.} \;\; I(\mathbf{z}:\mathbf{c}) \leq \delta$$

$$\text{or,} \;\; \max_q I(\mathbf{y}:\mathbf{z}|\mathbf{c}) - \beta I(\mathbf{z}:\mathbf{c})$$



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Algorithm

Algorithm 1: Building and training `Orient`

Input: M ; // L-layer baseline neural net
Input: D; // K layer baseline classifier
Input: $Loss$; /* A training function for $E, P, and\ D$ */
Input: (X, Y); // Training data
Input: TH; // A threshold for privacy
Output: E, P;

$E = \{\}, P = M, i = 0$;
while $i<L$ **do**
  $E = \{E, M_{i+1}\}$;
  train_{loss} E and P;
  train_{loss} D;
  $Priv = D(E(X))$;
  if $Priv \geq TH$ **then**
    break;
  end
  $i++$;
end

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

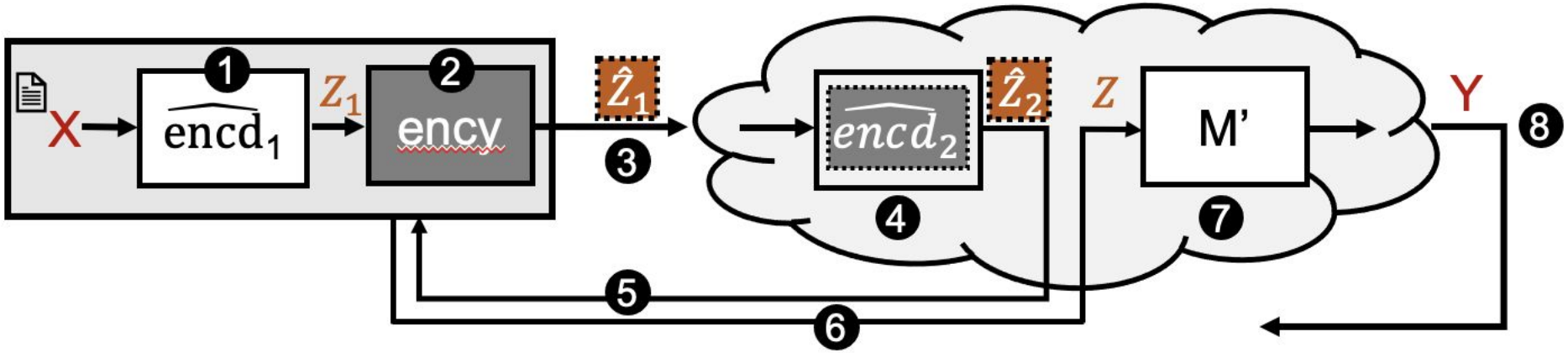
Are there other options?

- Can we have both IoT and cloud encoding?
  - The entire encoder doesn't need to be offloaded!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Are there other options → Orient Express

- Can we have both IoT and cloud encoding?
  - The entire encoder doesn't need to be offloaded!

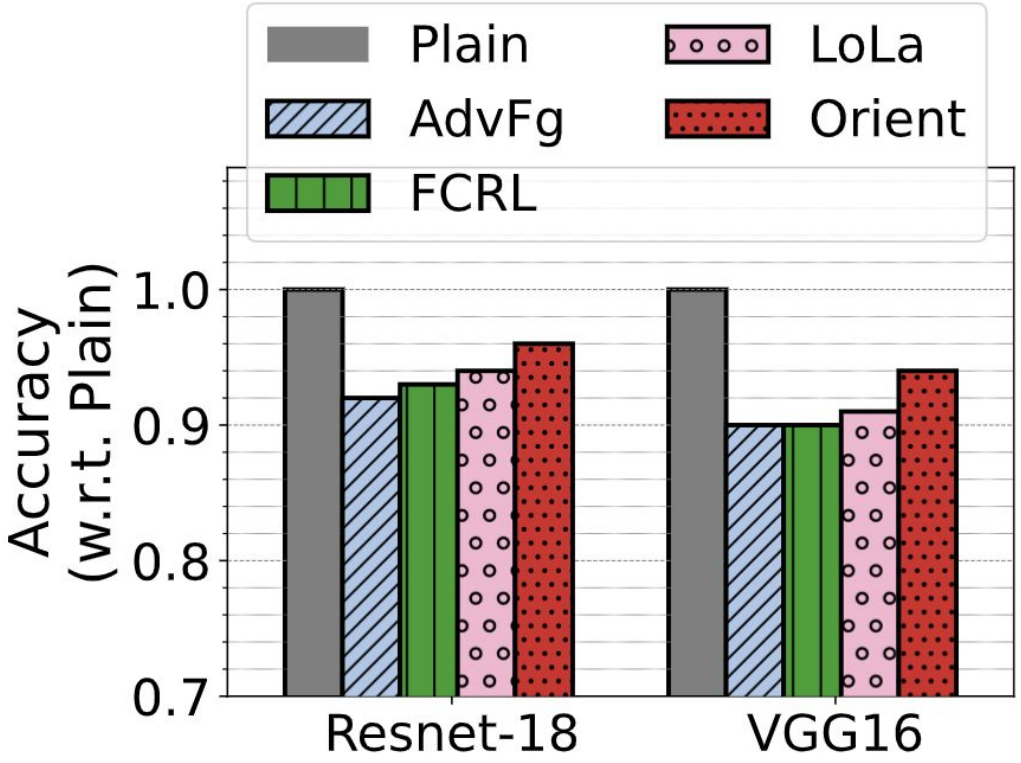


Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

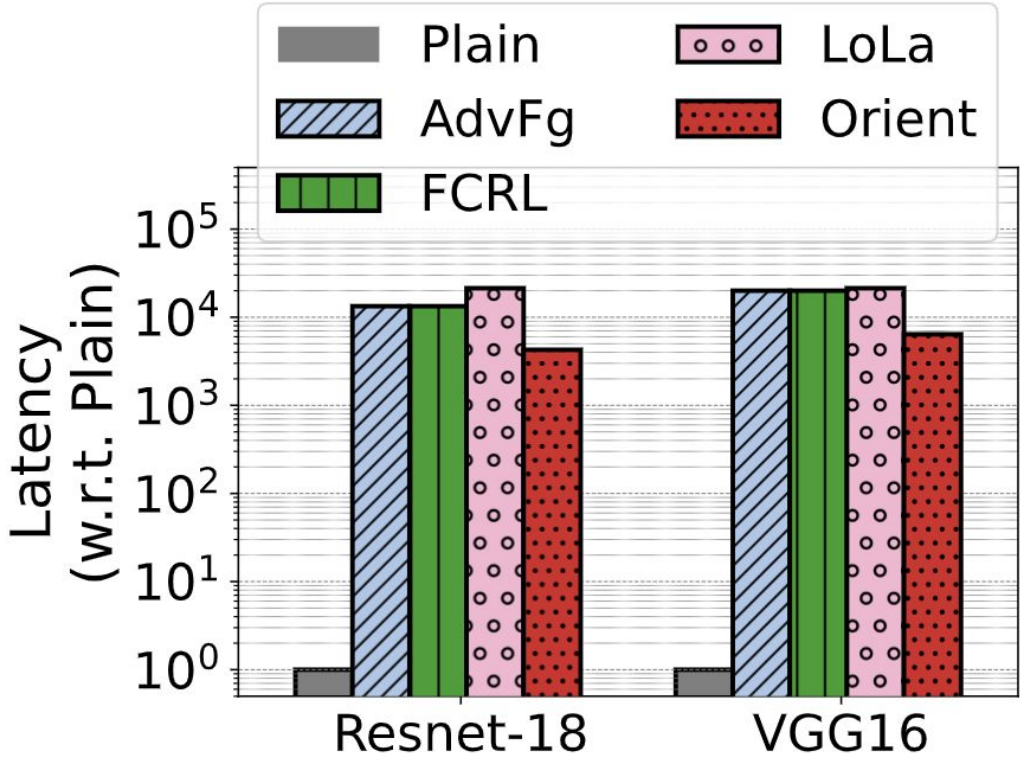
Tradeoffs

- A range of design choices:
  - In one extreme, everything is computed locally.
  - On the other extreme, everything is offloaded to cloud and computed homomorphically.
  - In between, there is a range.
    - Tradeoffs: energy, accuracy, and latency (no impact on privacy for same encoder design).

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

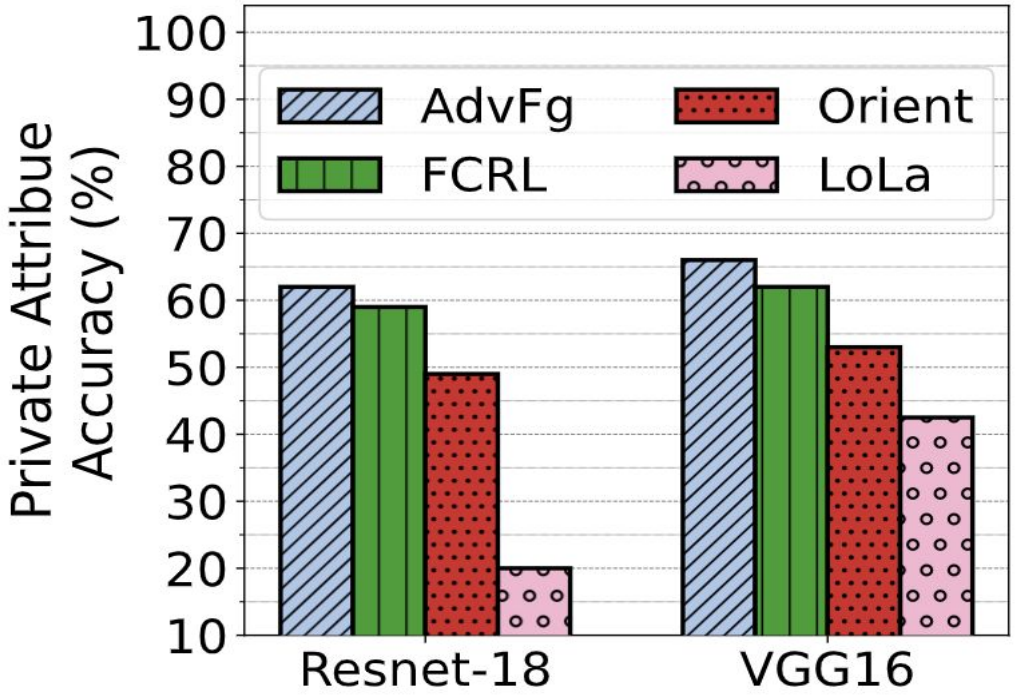


(a) *Accuracy* (higher is better)

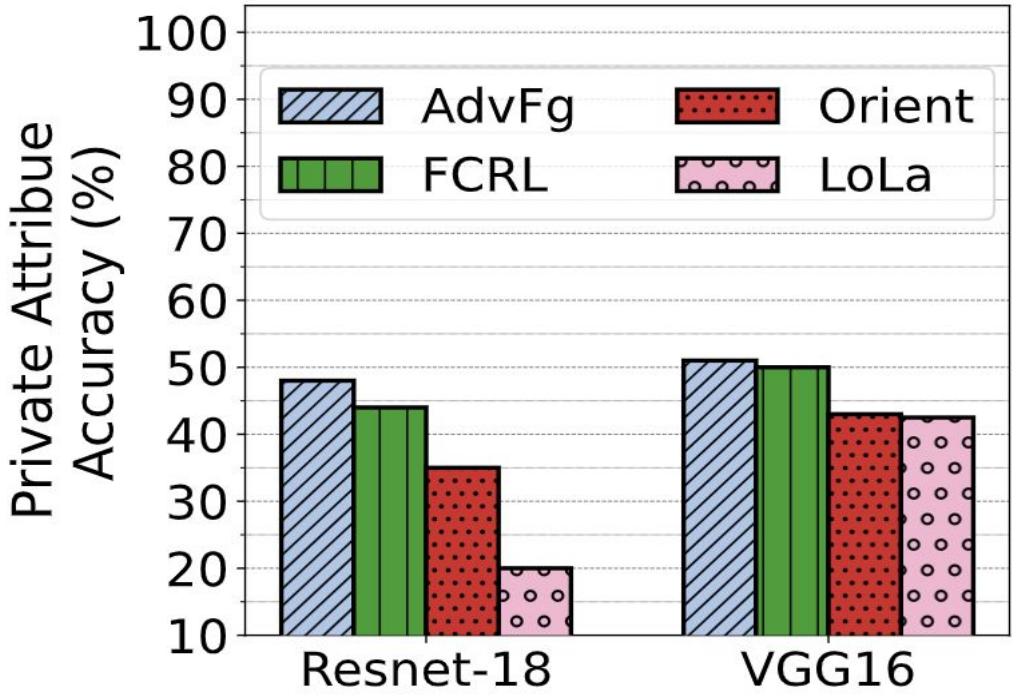


(b) *Latency* (lower is better)

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>



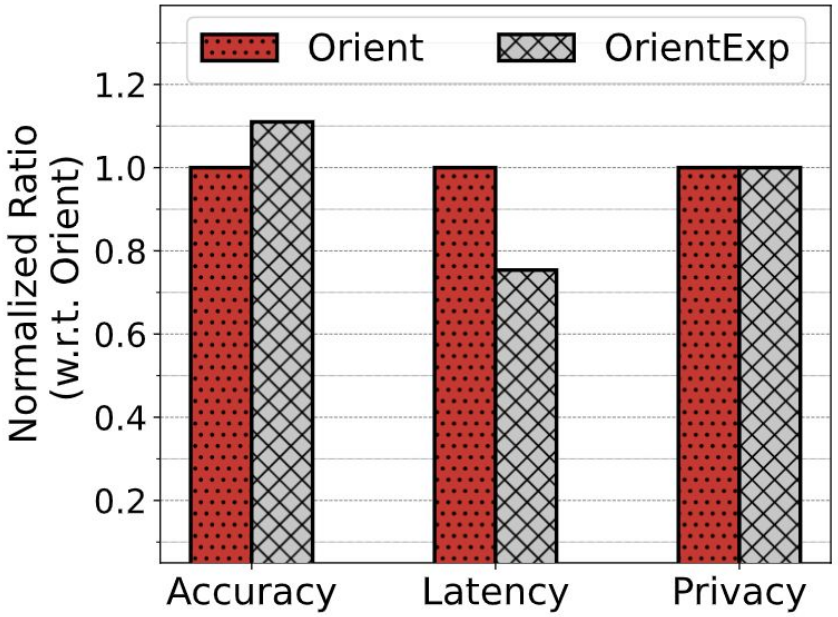
(a) $\alpha = .5$



(b) $\alpha = .75$

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Orient vs. Orient Exp.

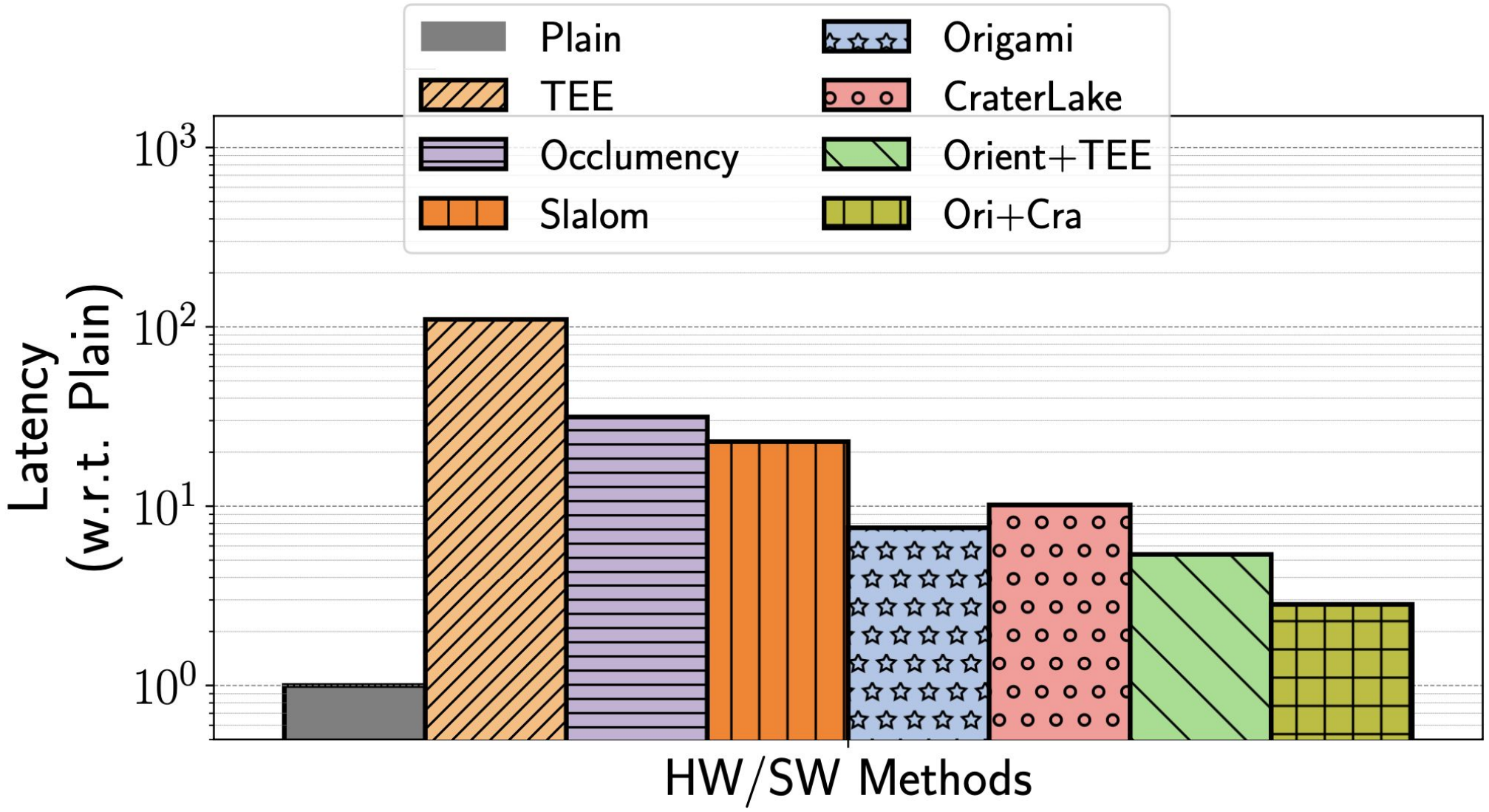


Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Combining

- Wait, can we mix everything up?

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Code

https://github.com/ssysarch/orient/blob/main/code/orient.cpp

- If you want to play with numbers and see the impact!

- Full paper: Enc2:Privacy-Preserving Inference for Tiny IoTs via Encoding and Encryption

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Summary

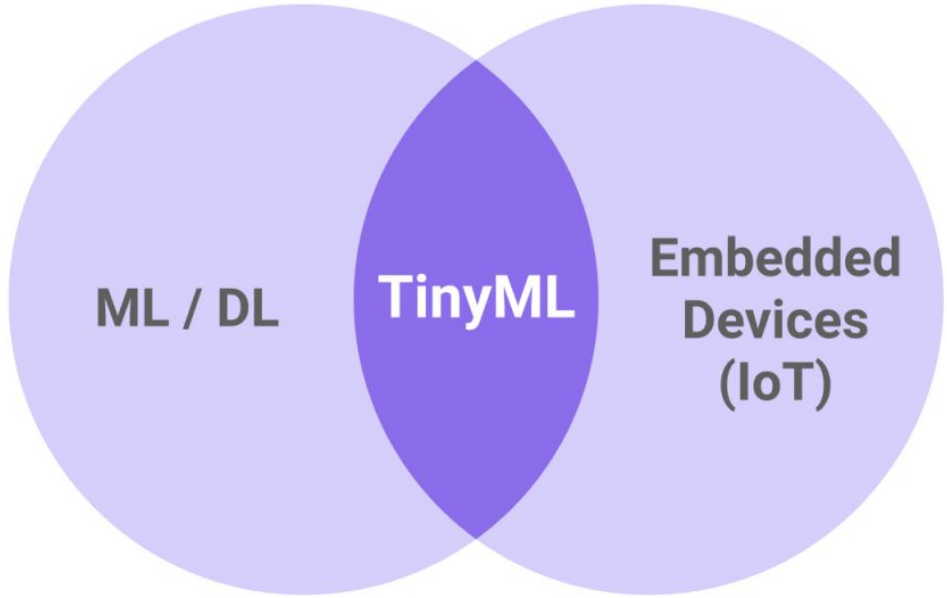
- Combining encoding and encryption could improve latency and accuracy while achieving better than average privacy.
- Our main insight is that there is a range of solutions.
- Encoding can be further applied to other solutions (e.g., TEE + encoding).

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

What about on-device computation?

- ***Nuclear Option:*** Scraping collaborative computation all together.

- TinyML
  - Better tools
  - Various tricks: quantization, pruning, …
  - More efficient hardware
  - …



Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Takeaways

- We are experiencing a paradigm shift in two major directions:
  1- Ubiquitous/Mobile computing:
  - We are surrounded more and more by computers, we generate more data and we need more computations!
  - With more connectivity, collaborative computing is more feasible and it's cheap and scalable!
  - With collaboration comes privacy concerns!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Takeaways

- We are experiencing a paradigm shift in two major directions:
  2- Machine learning/AI:
  - AI is enabling us to do new things that we couldn't do before hence its popularity is only increasing!
  - AI/ML is inherently computationally heavy and is getting worse, thus more computation is needed!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Takeaways

- Combining the two, we need new methods that can efficiently compute but also protect privacy!
- There is a range of solutions, and practical ones are those that combine the existing ones across the stack!
- Improvement in each direction could improve the overall system.

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Next Generation Privacy-Preserving Computation

We need new algorithms!

We need new hardware designs!

We need new system integrations!

Nader Sehatbakhsh <nsehat@ee.ucla.edu>
Secure Systems and Architectures Lab <ssysarch.ee.ucla.edu>

Contact Information

If you are interested in collaborating at different layers, please contact me

Email: nsehat@ucla.edu

Twitter: @SehatNader