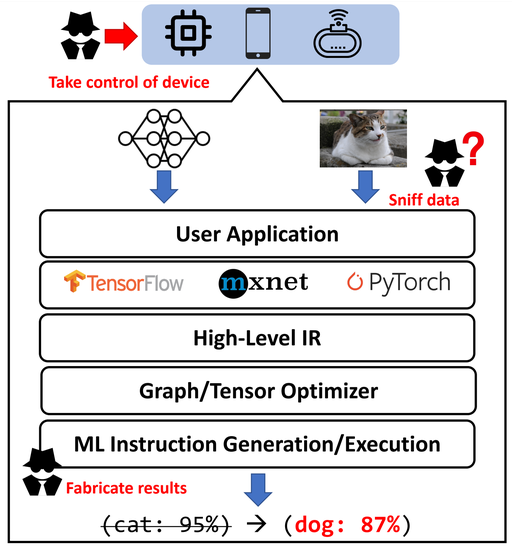
TrustZone as a Secure Tensor Processor

PURDUE UNIVERSITY
Electrical and Computer Engineering

Heejin Park, Noah Curran, Felix Xiaozhu Lin
Purdue University

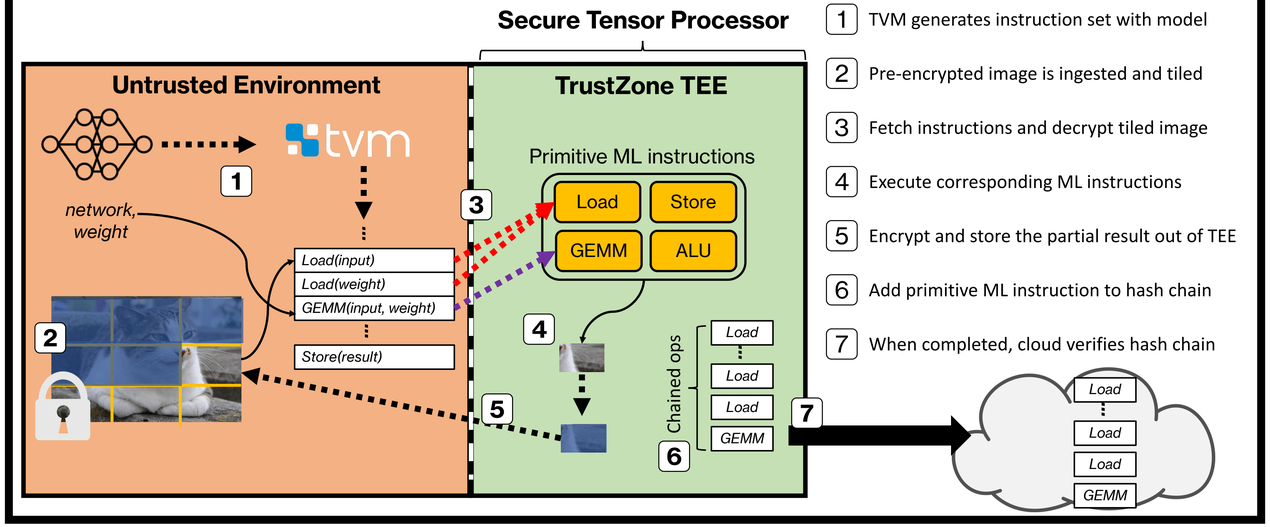
Trustworthy On-Device Inference

- **Low-end devices are more vulnerable**
  - Large ML inference software stack
  - Lack of professional management
- **Threat model - a powerful adversary**
  - Take full control of device
  - Sniff user private data on ML processing
  - Fabricate the inference processing and result
- **Approach** – ML computation within TEE
  - Deploy TZ as secure tensor processor
  - Safeguard minimal instruction set for ML inference



Secure Tensor Processor Architecture

- **Secure tensor processor**
  - Virtual processor with secure ISA for inference, exploiting ARM TrustZone
  - Processing ML inference by **1)** fetch instructions out of TEE, **2)** execute them within TEE
  - Generating hash chain of instructions for verifying inference correctness

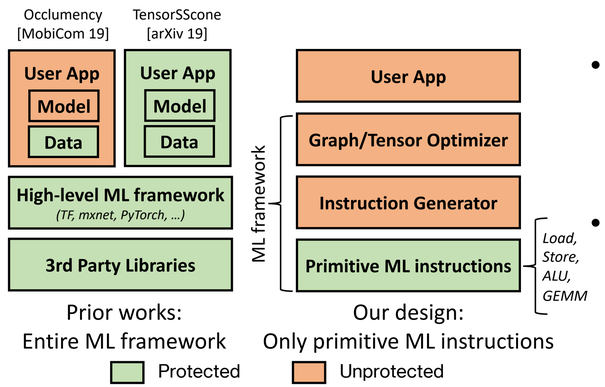


1. TVM generates instruction set with model
2. Pre-encrypted image is ingested and tiled
3. Fetch instructions and decrypt tiled image
4. Execute corresponding ML instructions
5. Encrypt and store the partial result out of TEE
6. Add primitive ML instruction to hash chain
7. When completed, cloud verifies hash chain

Design Choice for Protection Boundary

- **Design constraints:** 1) Limited memory 2) Minimal TCB

Our design choice – ML computations at the lowest point

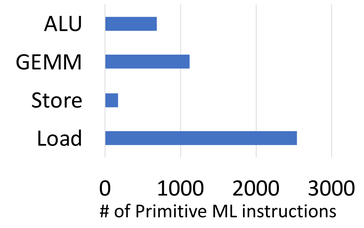


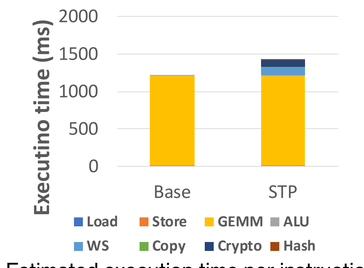
- **Protections at the lowest point**
  - Minimal software stack in TEE
  - Others out of TEE → reduce TCB size
- **Primitive ML instructions**
  - The lowest-level computations for inference
  - Along with VTA* design principle

*A hardware–software blueprint for flexible deep learning specialization [IEEE Micro2019]

- **Challenges:**
  - Hard limited secure memory (< 2MB) → tiled tensor streaming
  - Data integrity beyond protection boundary → en-/decryption, HMAC
  - Entire inference execution correctness → hash chain verified by trusty cloud

Early Results and Future Works





Estimated execution time per instructions

- **Estimated overhead running RestNet-18**
  - Numerous memory load instructions
  - GEMM operation is dominant in total execution time
- **Moderate security overhead (17%)**
  - World switch (7.8%) comes from Load/Store
  - Crypto overhead (7.5%) for en-/decryption
- **Future works**
  - How to guarantee data freshness during processing?
    - Adding tensor unique identifier to the hash chain
  - How to validate correctness of inference?
    - Symbolic execution of instruction set
  - How to further improve performance?
    - Internal parallelism within primitive ML instructions