An Early Experience with Confidential Computing Architecture for On-Device Model Protection

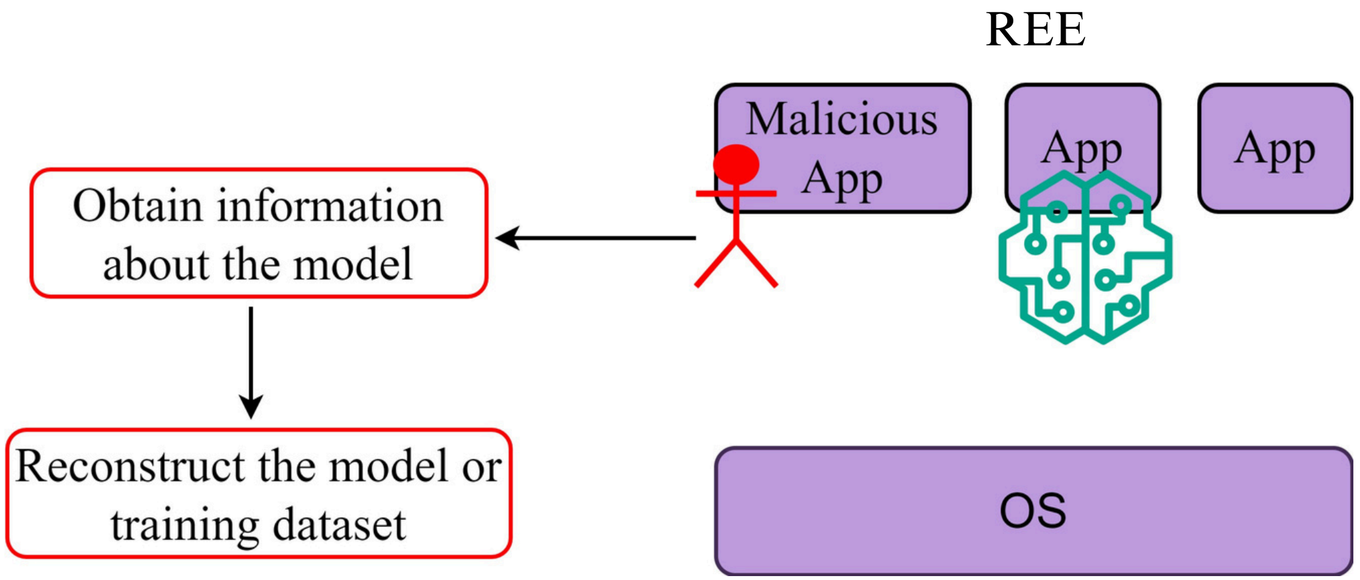
Sina Abdollahi, Mohammad Maheri, Sandra Siby
Marios Kogias, Hamed Haddadi

IMPERIAL

جامعة نيويورك أبوظبي
NYU ABU DHABI

Introduction

- Bringing ML inference service to the Edge can improve user's privacy
- Models and training data sets are high values targets
- Attackers can exploit large attack surface of Edge devices to obtain information about the model and its training dataset
- Without strong protections, model providers are not willing to deploy their models on the Edge devices



Model Protection

What are the available primitives to protect the model?

Using model in the encrypted mode

↓

Multi-party computation or homomorphic encryption

↓

High Overhead

Using model with hardware-enforced protections

↓

TEEs (acceptable overhead)

↓

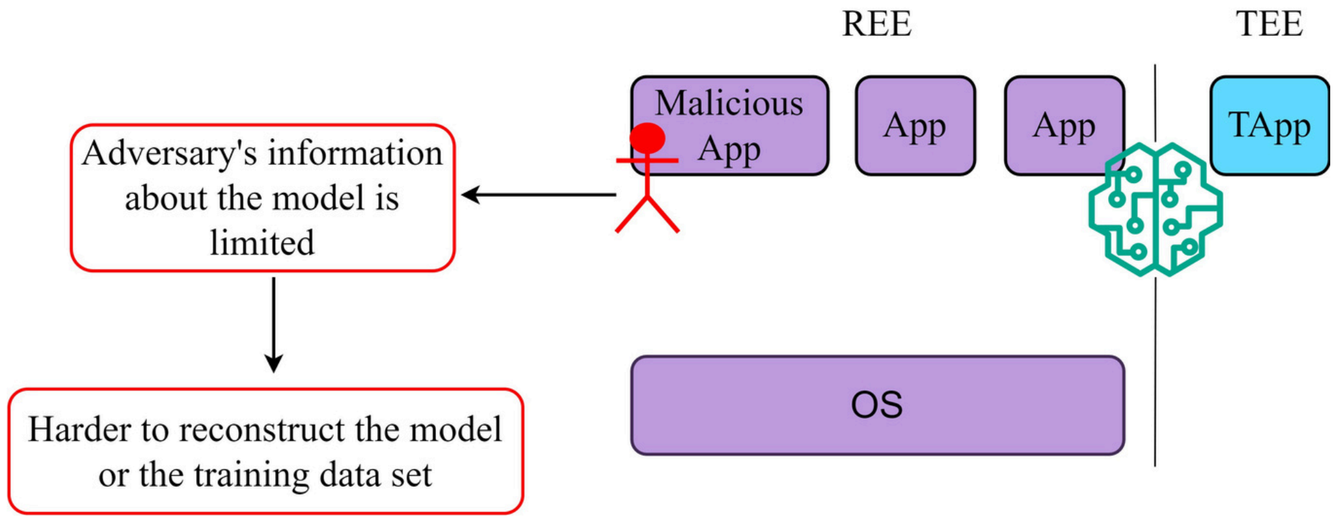
Acceptable Overhead

Exploring TEE solutions

- Limitation of current TEEs on Edge devices:
  1) Memory Size
  2) Library support
  3) GPU limitation
  4) Sometimes inaccessible for third-party apps
- Even running small models can be challenging within the current TEE solutions

Model Spliting

- Splitting techniques can overcome some of the TEE limits, however as shown in [1] such solutions remain vulnerable to privacy-stealing attacks
- Public parts of the model can be used to:
  a. Recover the private part
  b. Recover the training dataset



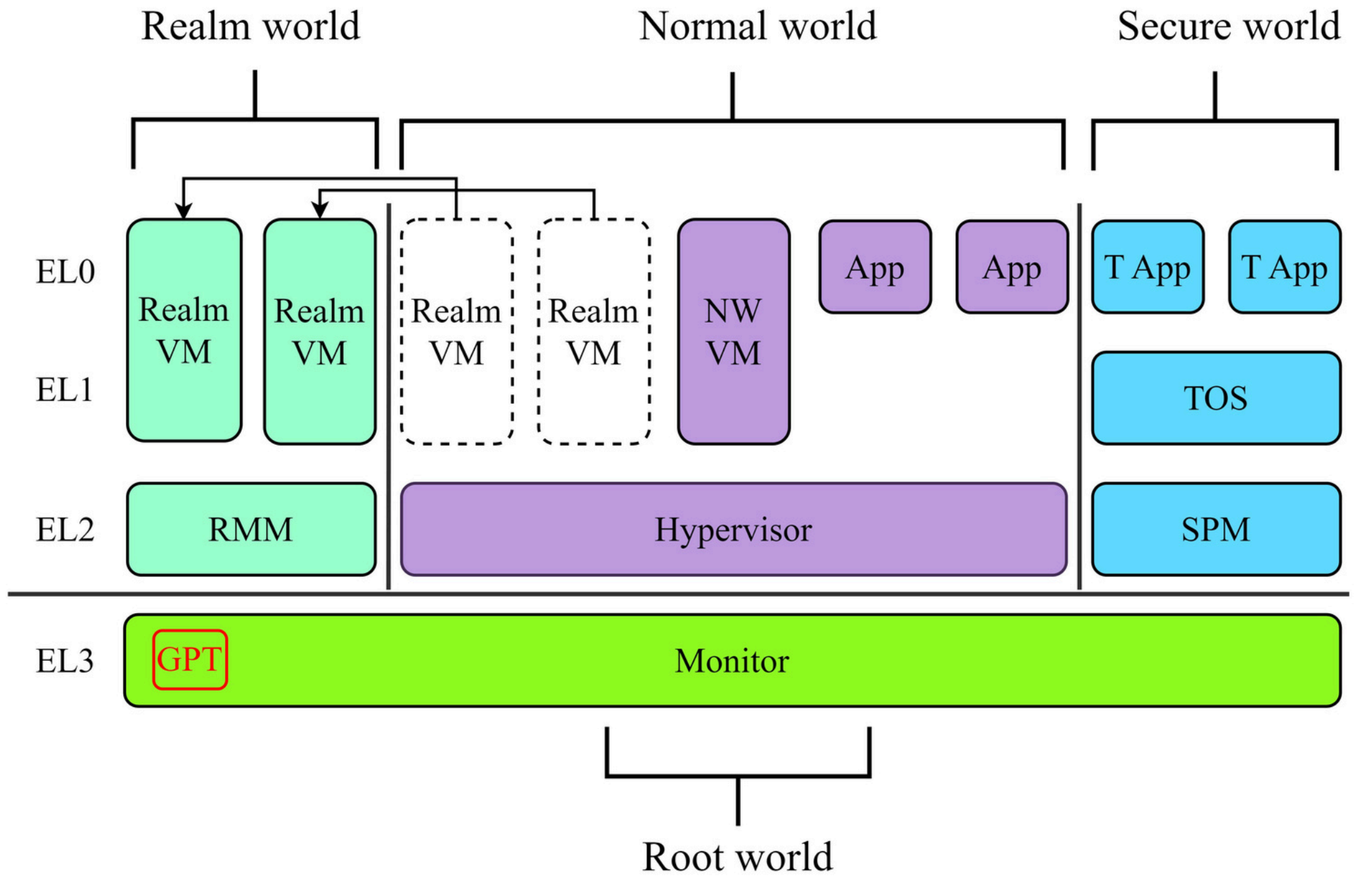
Conclusion: Running the entire model inside the TEE is desirable

Arm CCA

- New extension of Armv9-A architecture
- Interesting features which makes it suitable for our use-case:
    a. Targeting Edge devices
    b. Flexible memory allocation
    c. VM-level support

Arm CCA

- Introduces Realm world
- Hypervisor remains in control of resources but not access to resources given to the realm
- GPT is introduced to track state of memory pages
- TrustZone memory is curved out but realm memory is dynamic

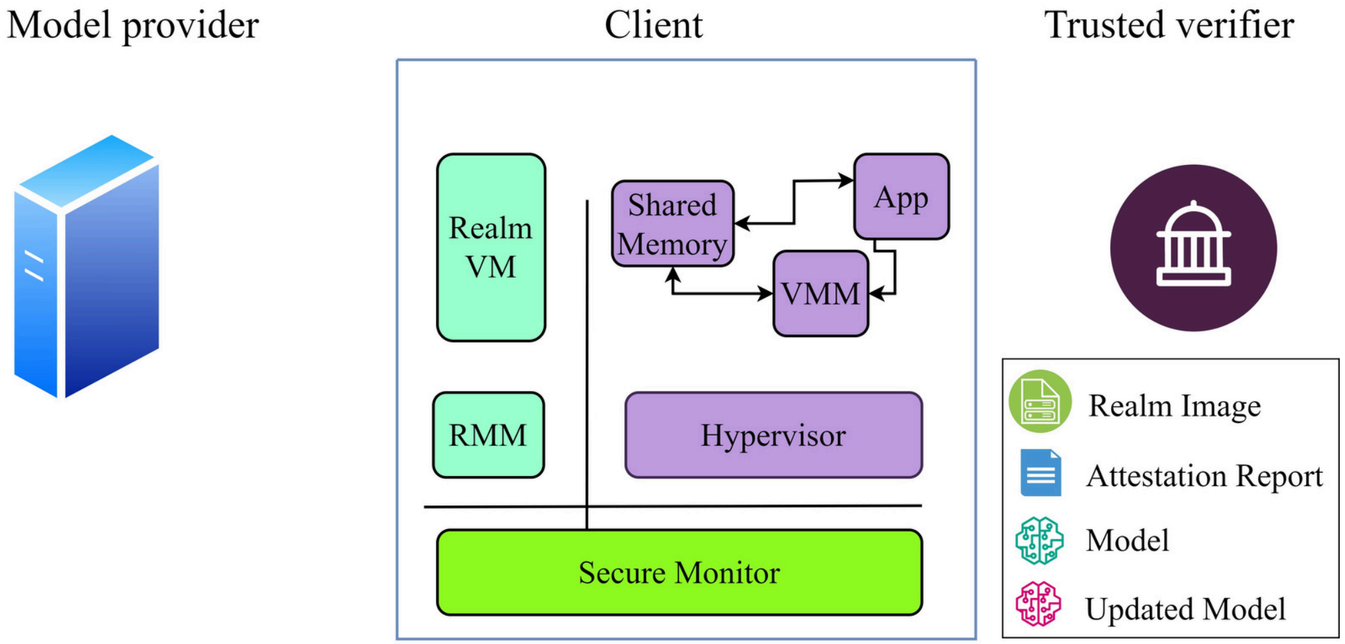


Framework Overview

- We developed a basic framework for on-device model deployment with CCA
- Used Fixed Virtual Platform (FVP) to emulate the architecture and report the cost of our framework
- Performed membership inference attack against the framework → Showing the privacy benefit of our scheme

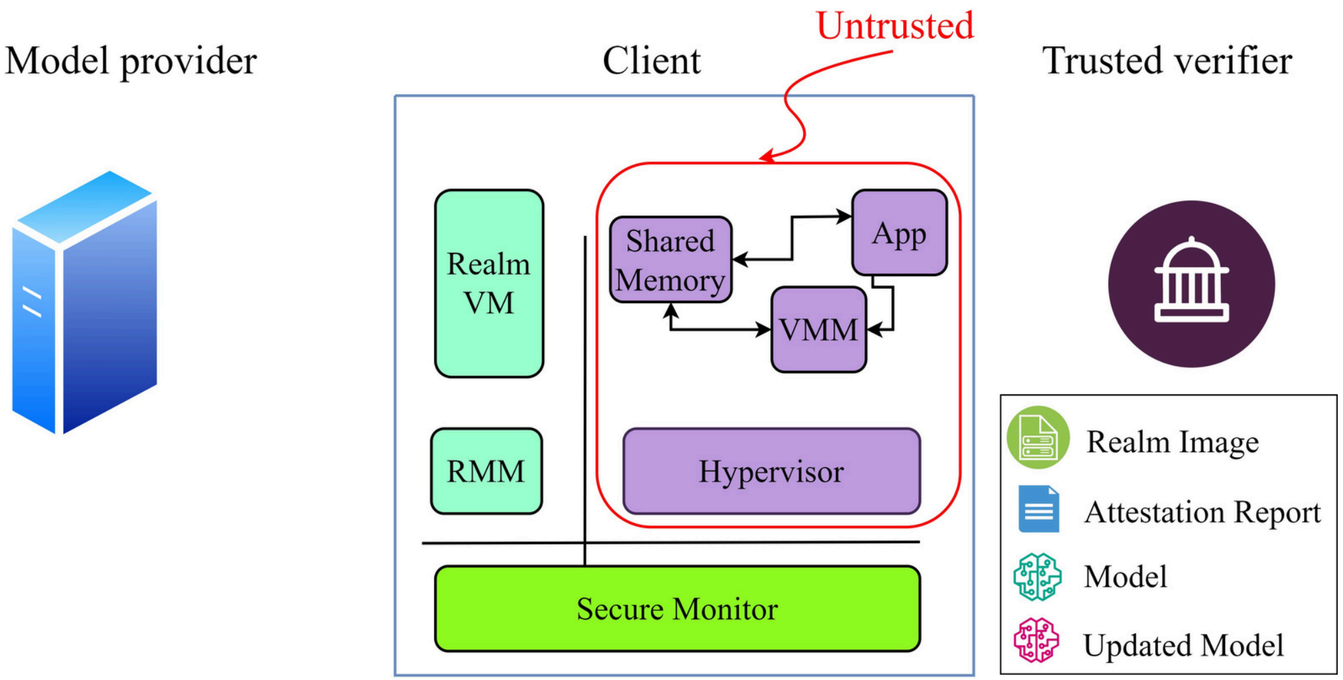
System Model

- Three engaged parties



Adversary Model

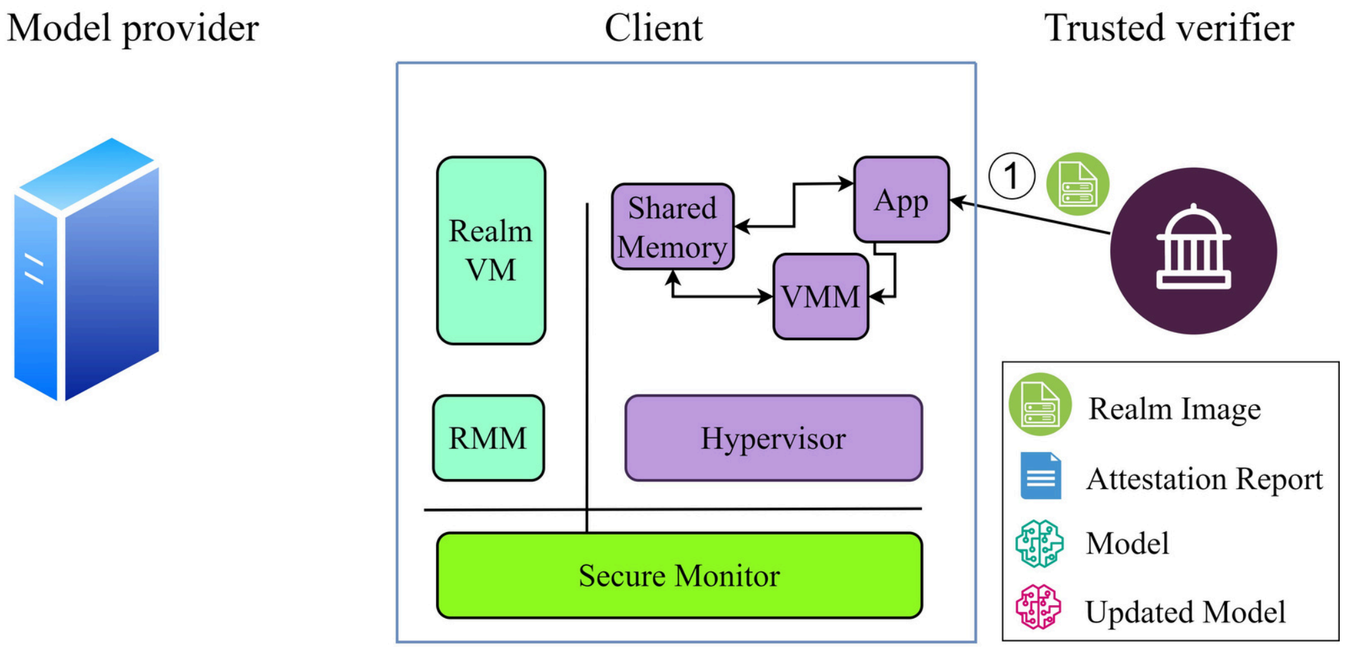
- NW is compromised
- CCA software is trusted
- Realm image is trusted



Framework Overview

Model Deployment pipeline

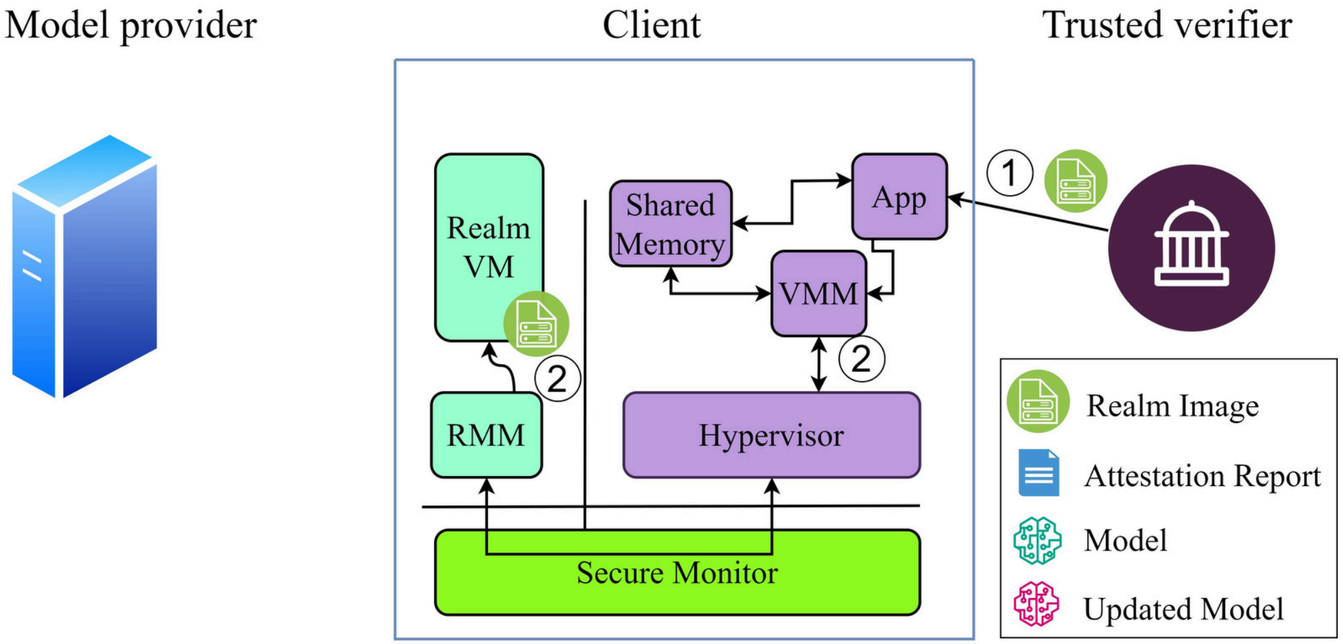
1. Obtaining realm image



Framework Overview

Model Deployment pipeline

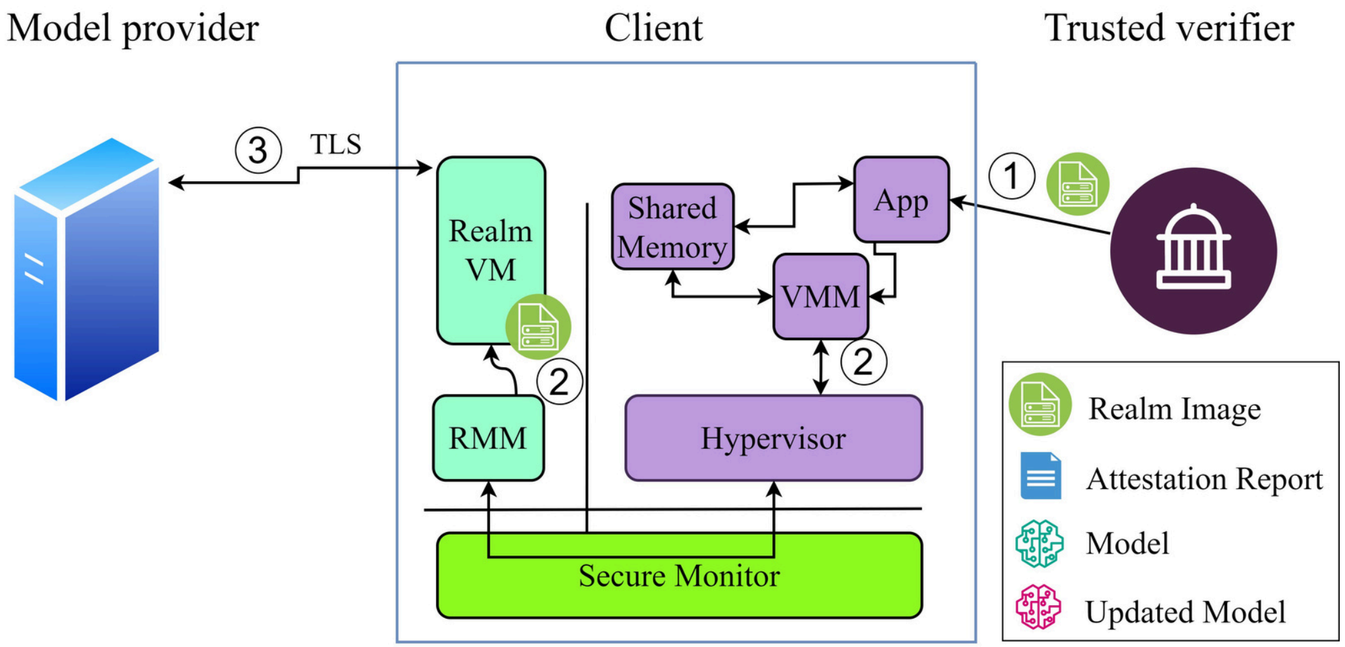
1. Obtaining realm image
2. Creating and activating a realm



Framework Overview

Model Deployment pipeline

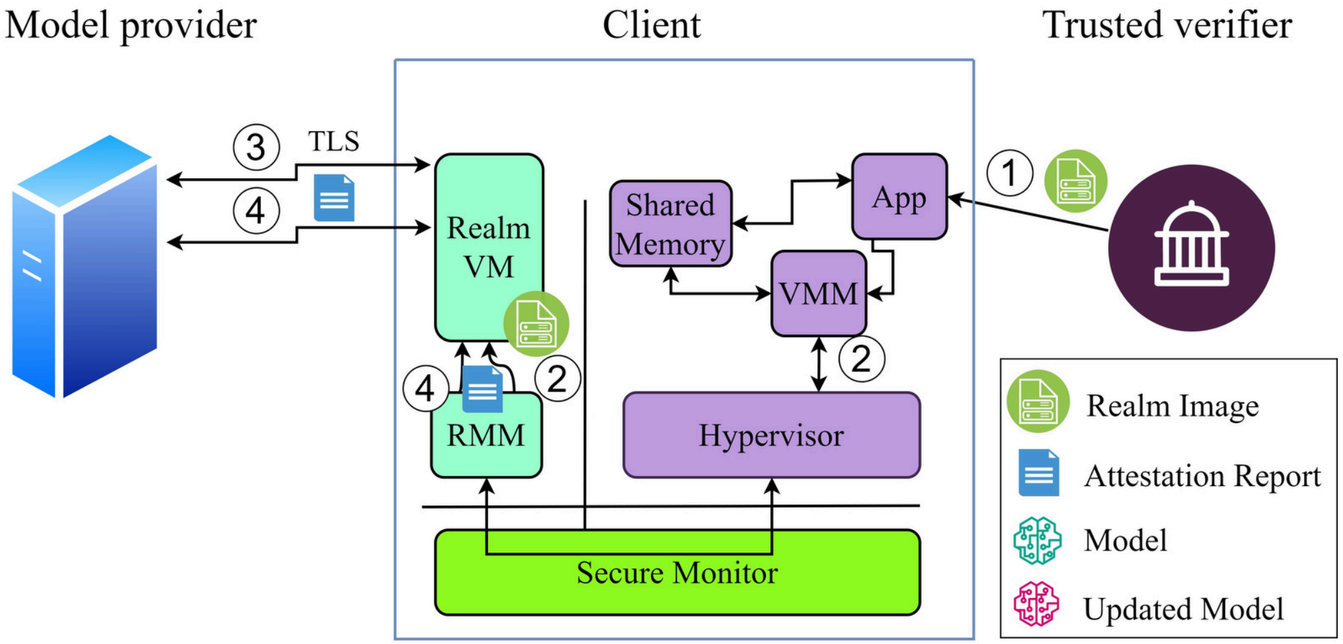
1. Obtaining realm image
2. Creating and activating a realm
3. Establishing TLS connection



Framework Overview

Model Deployment pipeline

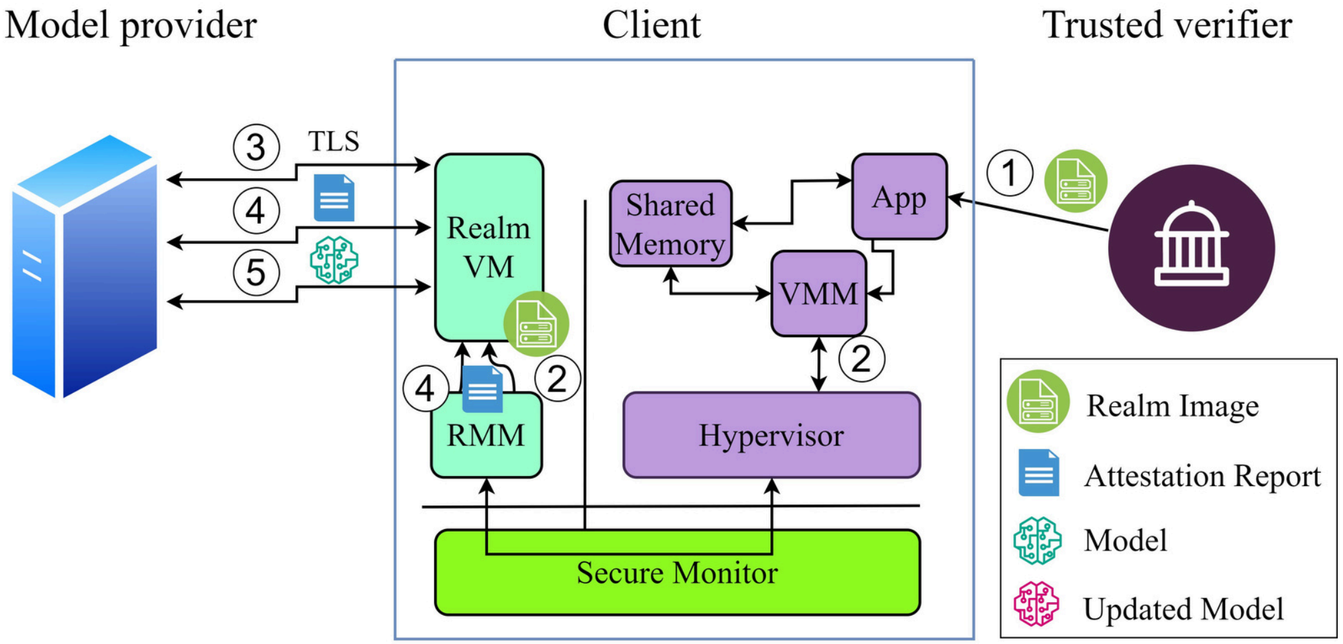
1. Obtaining realm image
2. Creating and activating a realm
3. Establishing TLS connection
4. Realm attestation



Framework Overview

Model Deployment pipeline

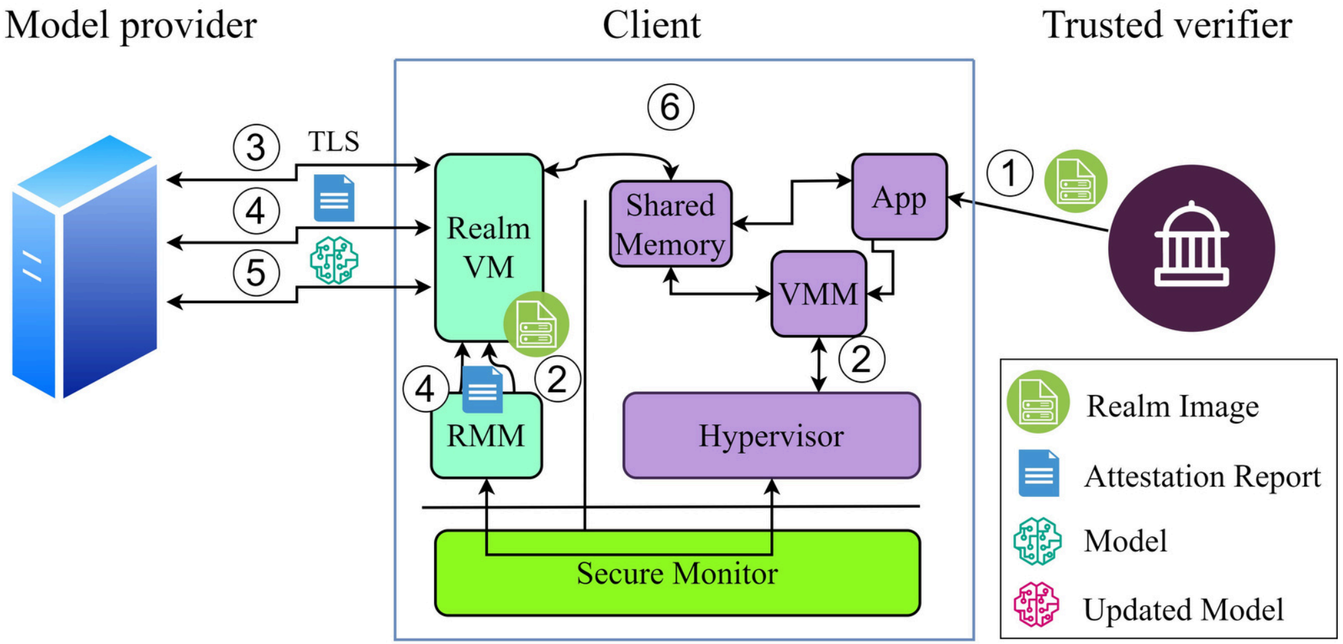
1. Obtaining realm image
2. Creating and activating a realm
3. Establishing TLS connection
4. Realm attestation
5. Obtaining model from provider



Framework Overview

Model Deployment pipeline

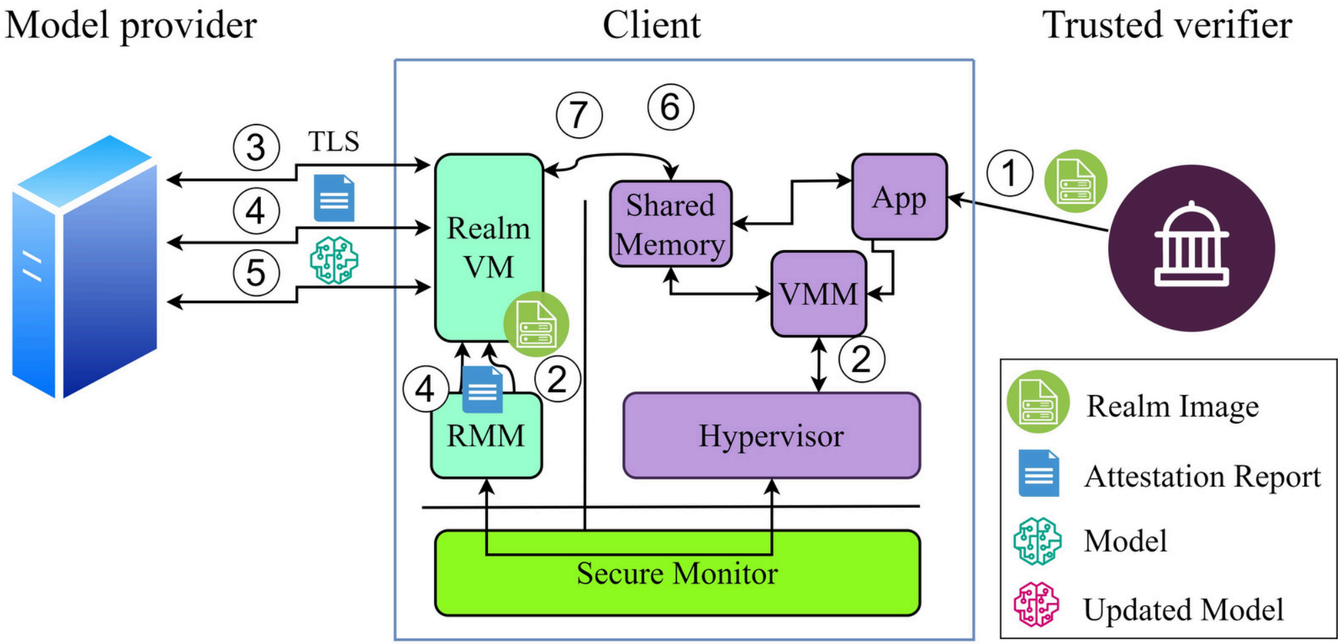
1. Obtaining realm image
2. Creating and activating a realm
3. Establishing TLS connection
4. Realm attestation
5. Obtaining model from provider
6. Announcing model readiness to normal world



Framework Overview

Model Deployment pipeline

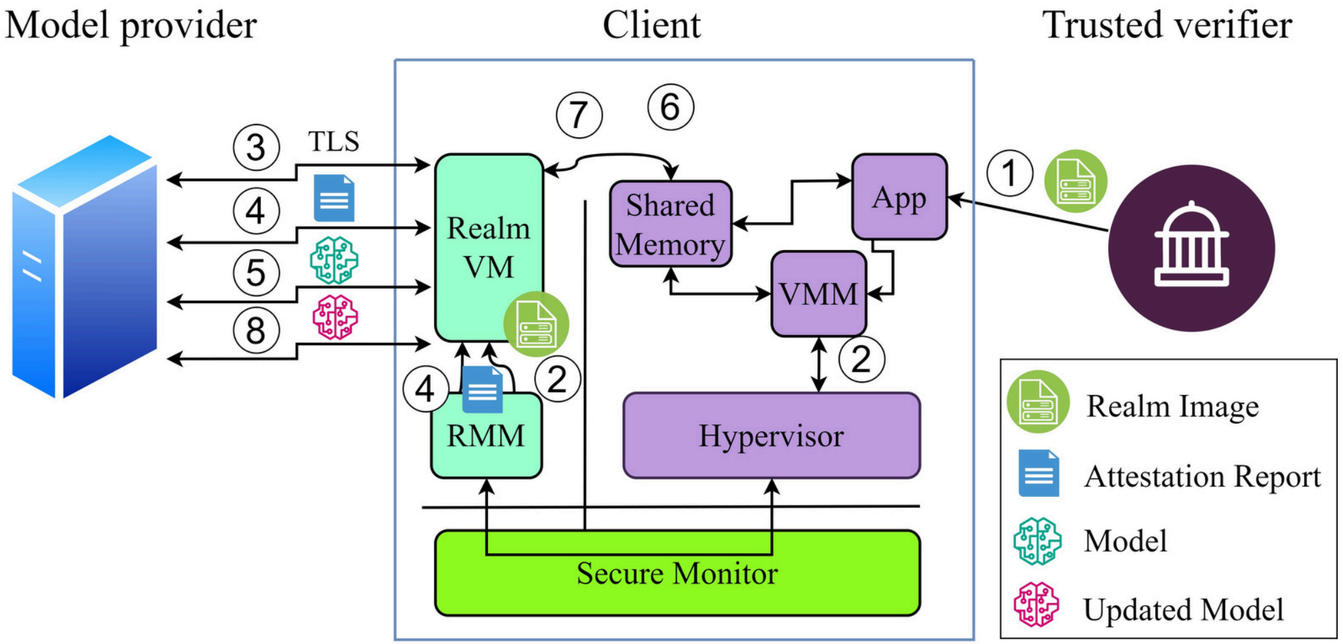
1. Obtaining realm image
2. Creating and activating a realm
3. Establishing TLS connection
4. Realm attestation
5. Obtaining model from provider
6. Announcing model readiness to normal world
7. Running inference



Framework Overview

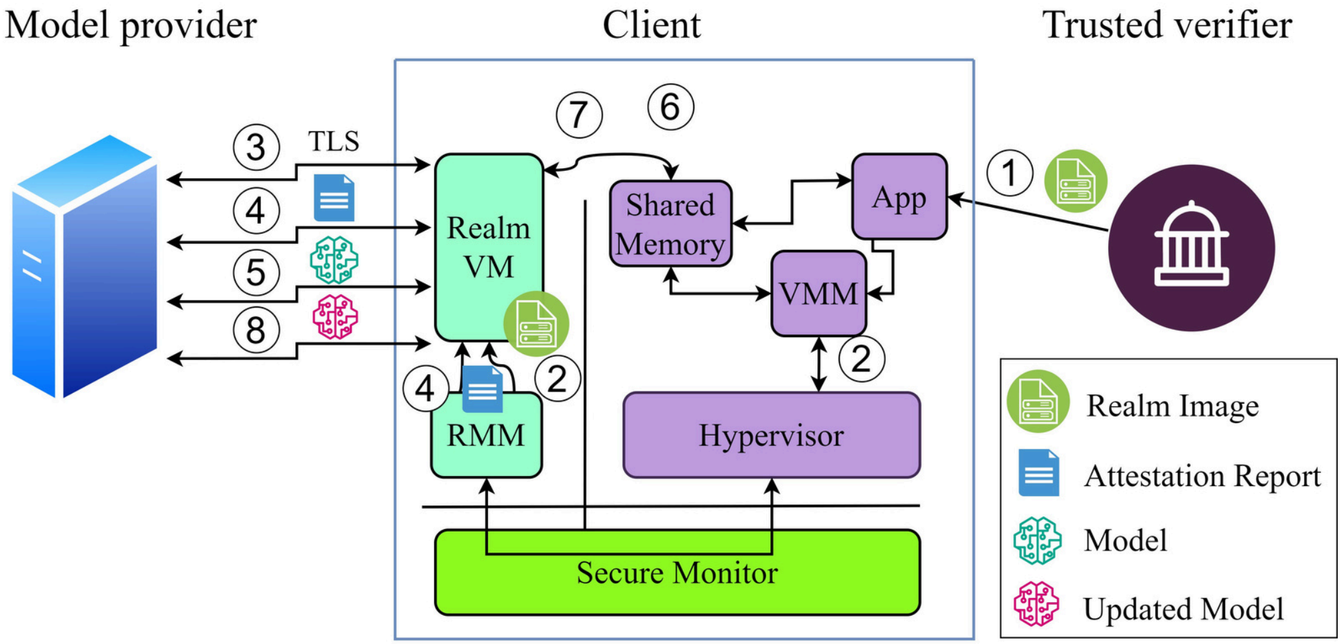
Model Deployment pipeline

1. Obtaining realm image
2. Creating and activating a realm
3. Establishing TLS connection
4. Realm attestation
5. Obtaining model from provider
6. Announcing model readiness to normal world
7. Running inference
8. performing model updates



Framework Evaluation

- Running the same inference service in two settings:
  - Model within Realm VM
  - Model within a NW-VM (baseline)
- Running model within NW exposes it to the adversary.
- The adversary use model information to perform a membership inference attack



Experimental Setup

- Different applications: image classification, voice recognition, and chat assistants
- CCA allows easy integration of relatively big models like GPT2

Model	Model Size (MB)	VM size (MB)
AlexNet	9	300
MobileNet_v1_1.0_224	16	400
ResNet18	44	450
Inception_v3_2016_08_28	95	1750
VGG	261	3650
GPT2	177	900
GPT2-large	898	1800
TinyLlama-1.1B-Chat-v0.5	1169	2000

Inference Overhead

- Compare running our framework against the baseline
- Only report number of instructions not latency!!!
- CCA can achieve an overhead of, at most, 22%

Model Init Ovh	Read Input Ovh	Inference Ovh	Write Output Ovh	Total Ovh
16%–41%	44%–100%	17%–22%	31%–133%	17%–22%

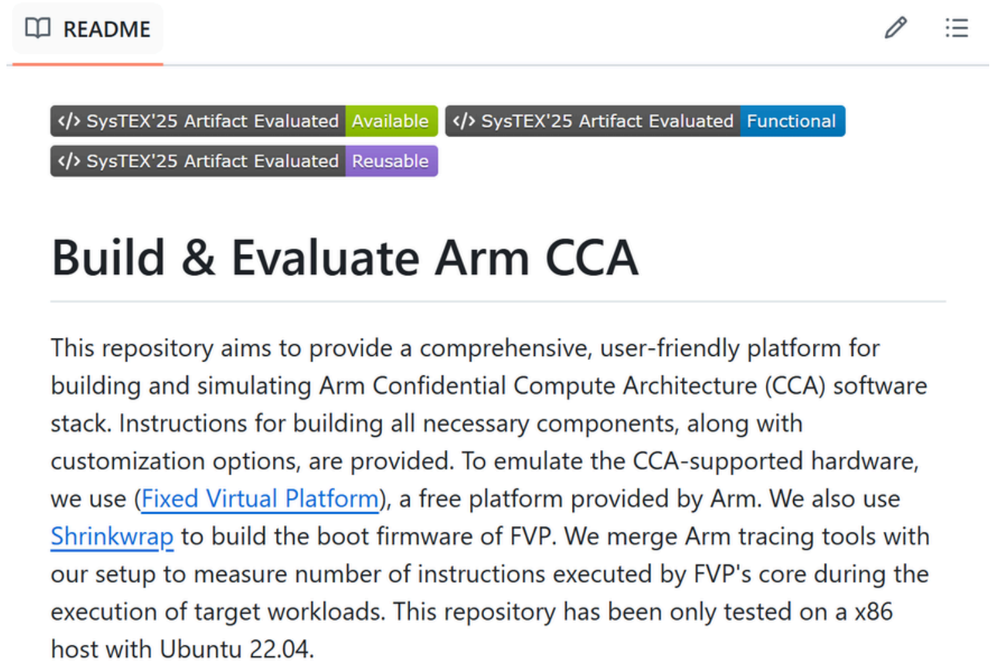
Privacy Evaluation

- Repeat White-box and black-box attack for different models and datasets
- Our framework can successfully protect the model against membership inference attack by 8.3% reduction in the adversary's success rate.

Conclusion and Summary

- We measure both the overhead and the privacy gains of running models of various sizes and functionalities within a realm VM.
- We provide the first indication of CCA suitability as a mechanism to provide model protection.

https://github.com/comet-cc/CCA-Evaluation



References

[1] Z. Zhang, C. Gong, Y. Cai, Y. Yuan, B. Liu, D. Li, Y. Guo, and X. Chen, “No Privacy Left Outside: On the (In-) Security of TEE-Shielded DNN Partition for On-Device ML,” in 2024 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2024, pp. 52–52.