System Security
General challenges and a Cloud GPU System

Tyler Hunt, Zhipeng Jia, Vance Miller, Ariel Szekely
Christopher J. Rossbach, Emmett Witchel

vmware RESEARCH

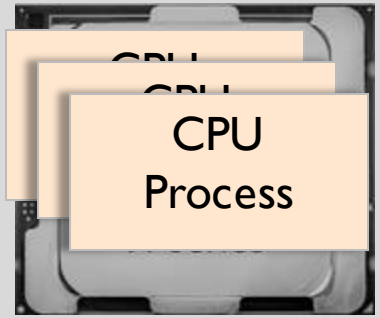
Hardware isolation: necessary but not sufficient

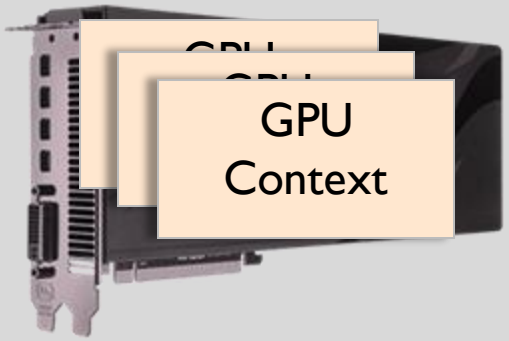
- Hardware provides efficient isolation
- Side channels (e.g. execution time) can violate that isolation
- Composing hardware while maintaining isolation is tricky
  - Especially if we want to keep the API the same
  - In this talk: Composing isolation on GPUs and CPUs

Leaks persist regardless of mechanism

User/Kernel boundary

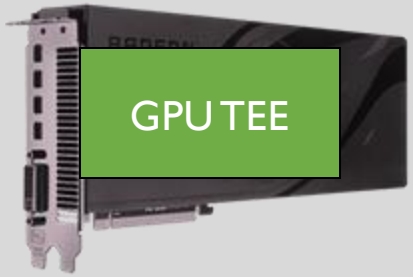
Address space separation





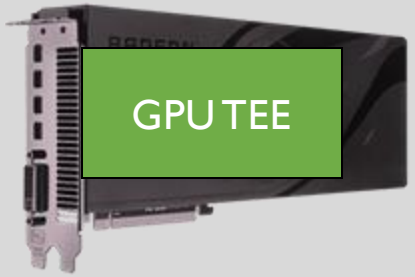
Trusted Execution Environments (TEEs) (aka. enclaves)

- SGX, TrustZone, Graviton [OSDI`18], Keystone



Trusted execution environments

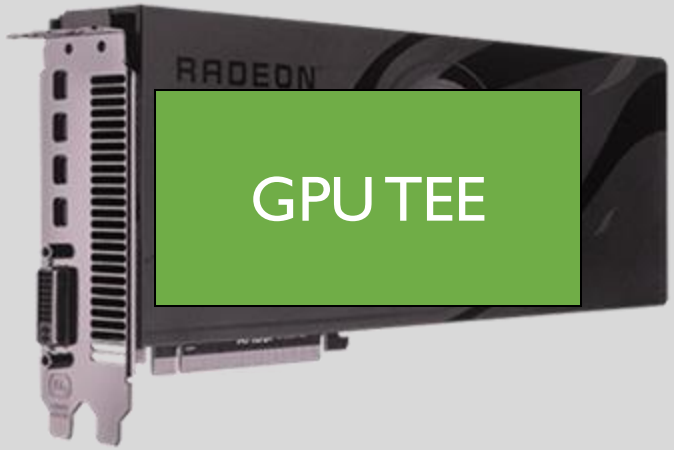
- Isolate application from the OS/Hypervisor
- Useful in cloud computing where providers (and all of their human administrators) have control of the OS
- Focus on TEEs which use the OS for communication
  - Not necessarily true of TrustZone
  - True of PCIe, used in data centers



In the rest of the talk we will

- Outline a new GPU TEE side channel attack
- Discuss a system to defend against the class of attacks

GPU TEE background



- Proposed by Graviton [OSDI`18]
- GPU memory secrecy and integrity are protected by GPU firmware
- GPU is programmed using the usual interfaces
- Commands and data are protected by cryptography

Image recognition background

- Goal: Given an image, output the object
- State-of-the-art image recognition uses deep neural networks (DNNs)
- Each layer of the neural network is evaluated by one or more GPU kernels

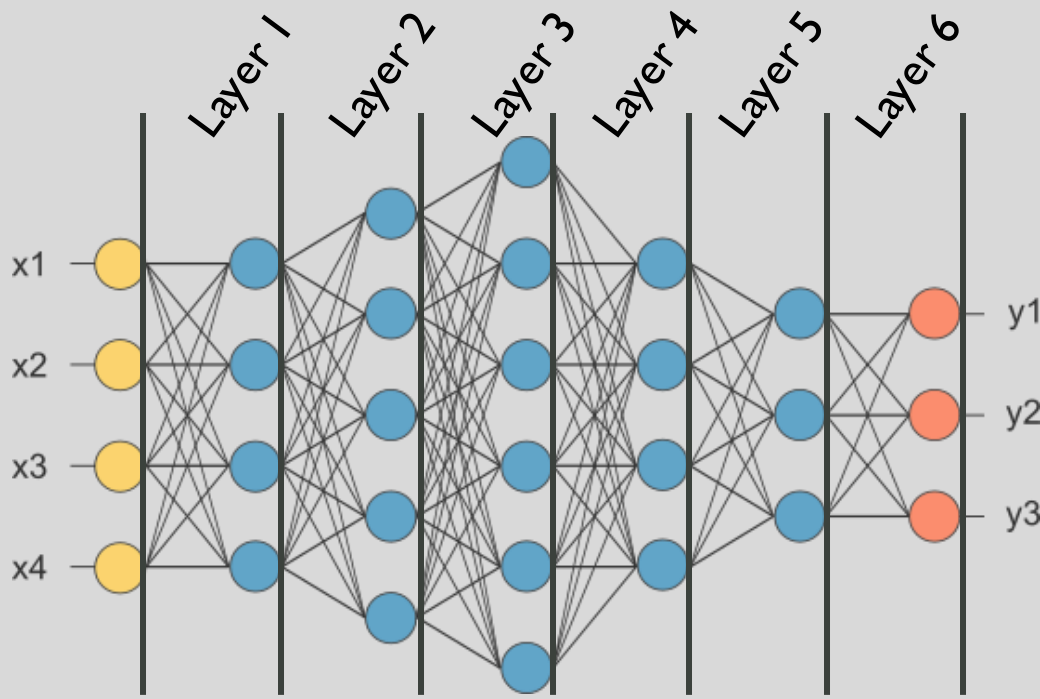
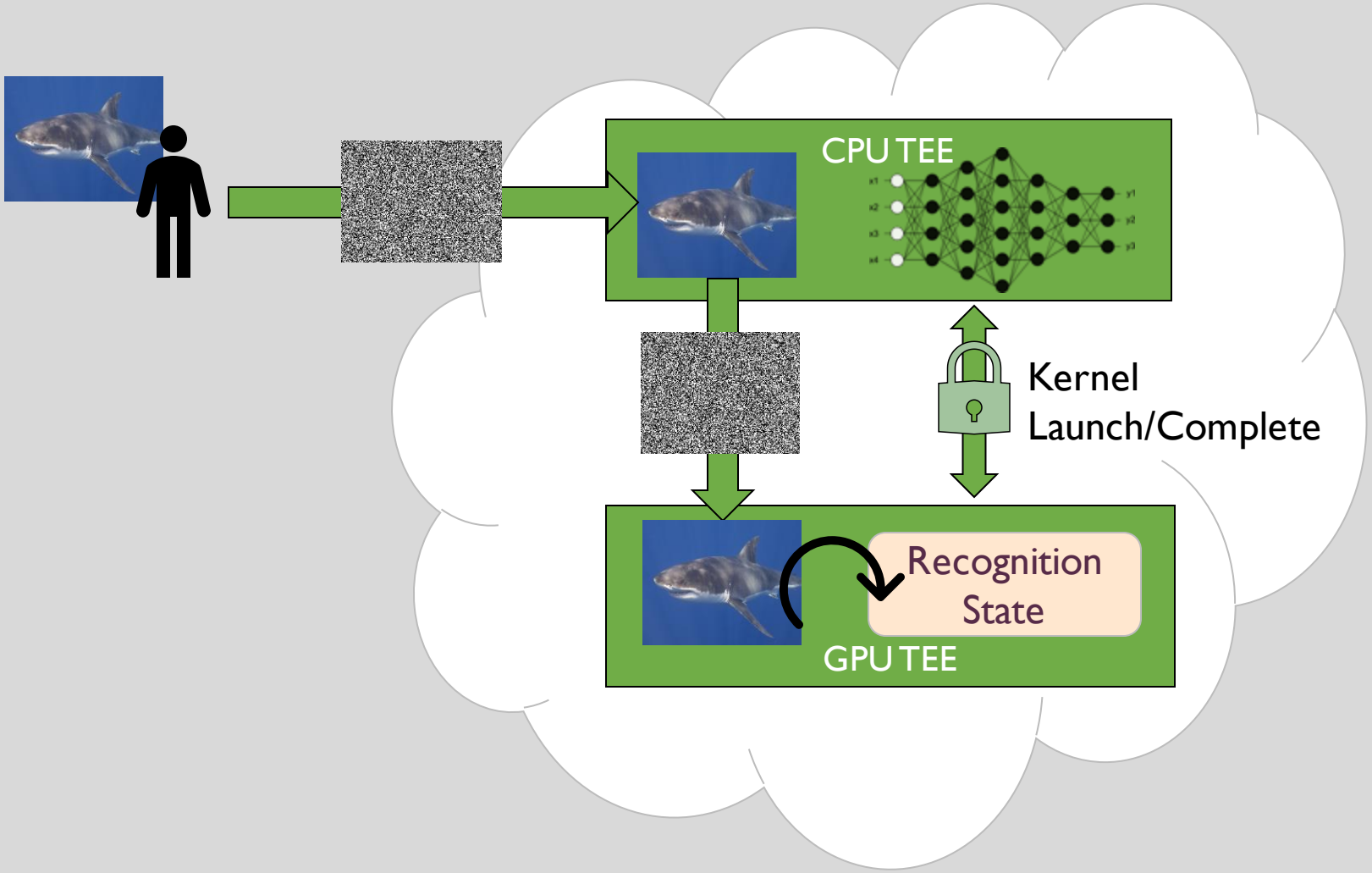


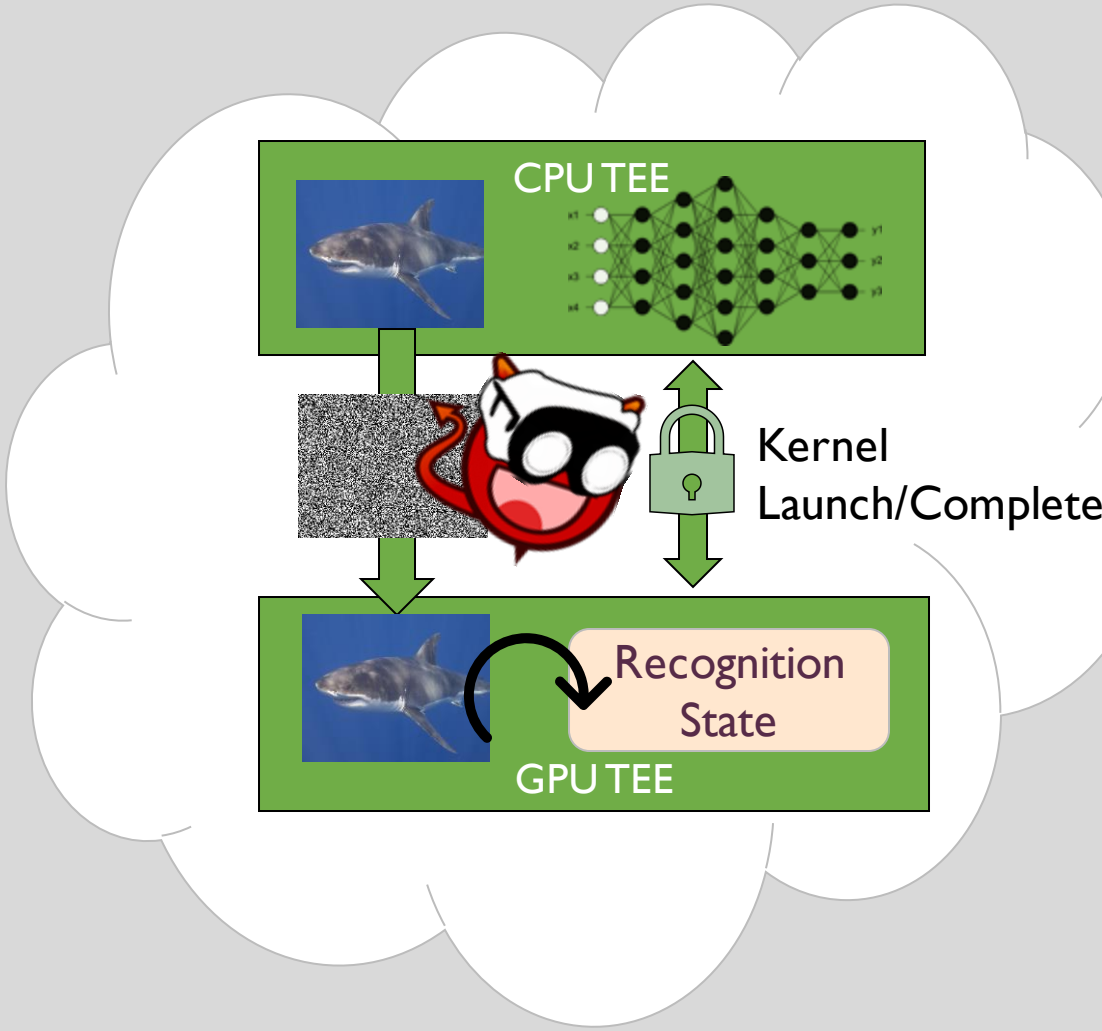
Image recognition with TEEs



Kernel execution timing channel

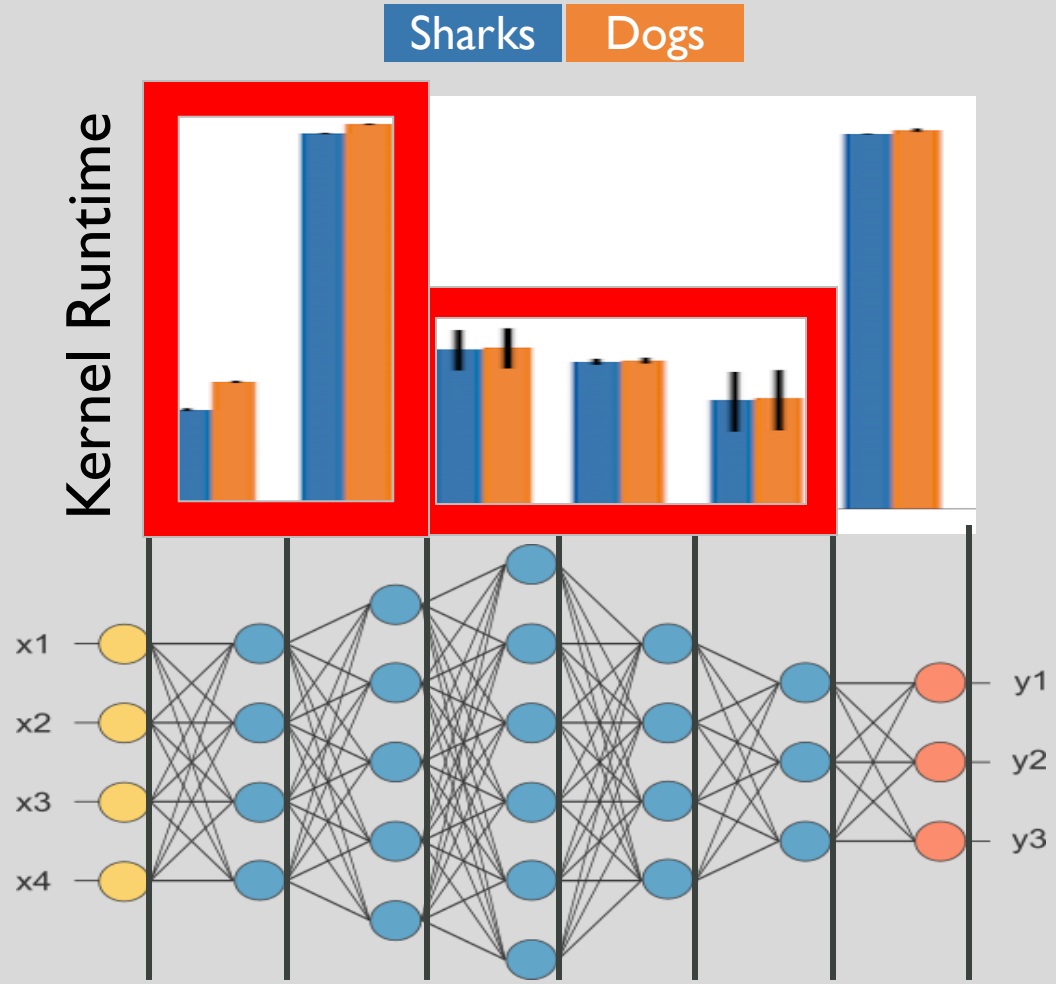
- Use kernel launches and completions to get kernel execution timing
- Count kernel executions to ascertain their DNN layer
  - Kernels are always executed in the same order

Can the OS learn anything about the input images from the execution time of each layer?

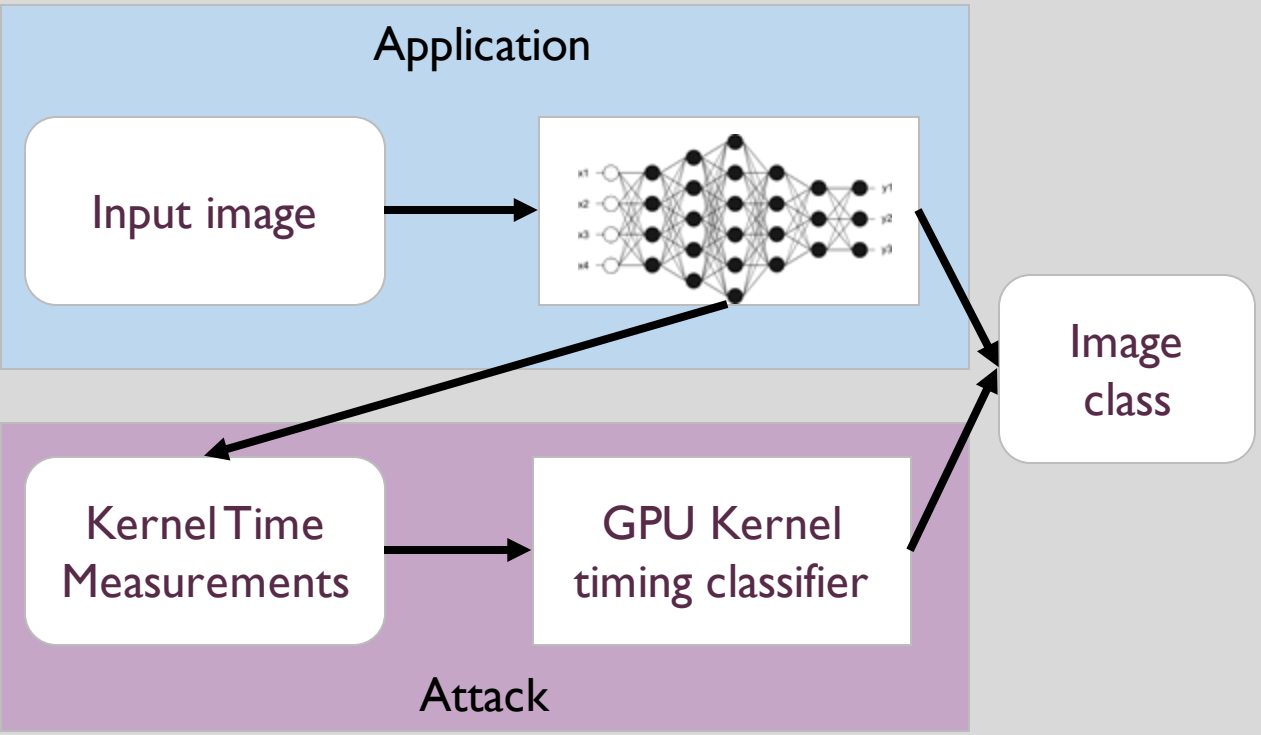


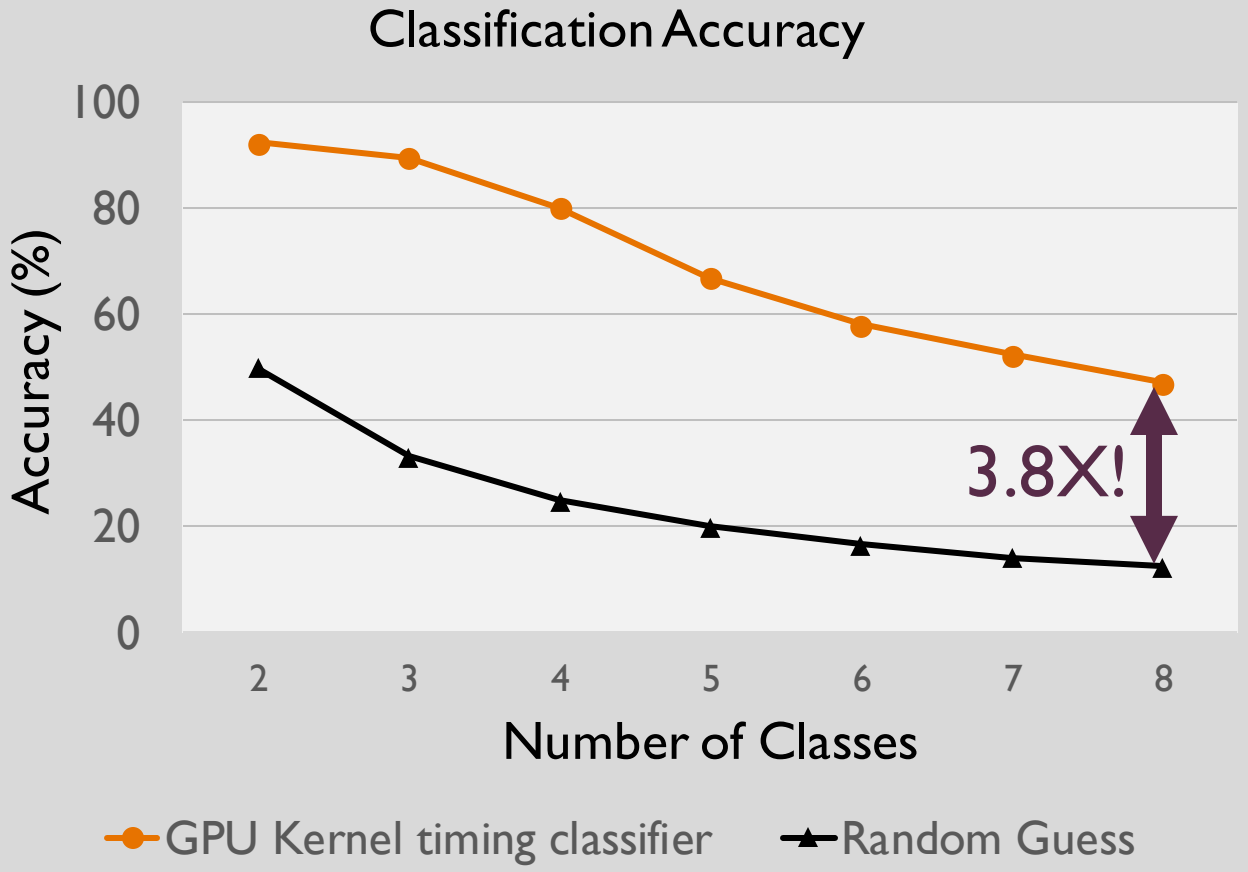
Some kernel execution times depend on input

- Measure ResNet on some image classes from ImageNet
- Bars show mean runtime and error
- Some layers clearly have significant timing differences



Attack machine learning with machine learning



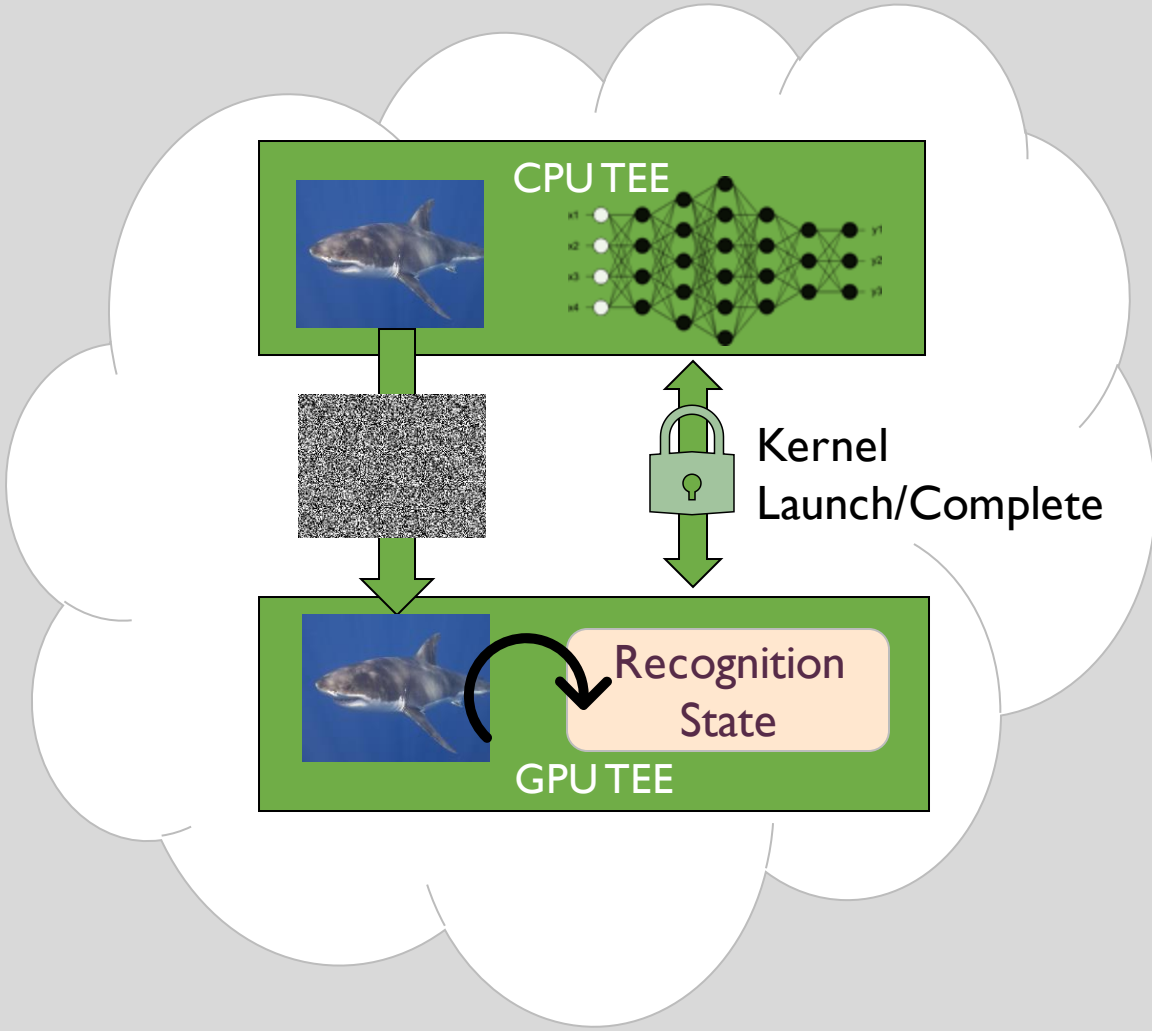


In the rest of the talk we will

- Outline a new GPU TEE side channel attack
- Discuss a system to defend against this class of attacks

Traffic between TEEs is observable

- OS sees encrypted images, kernel launches, kernel completions
  - Time and size still useful
- Make observable events data oblivious
  - I.e., independent of data (images)



Approach: decouple traffic from execution

- Modify GPU TEE to eliminate kernel execution notification
- Transform traffic so that it does not give clues about kernel execution

GPU TEE requirements to support mitigation

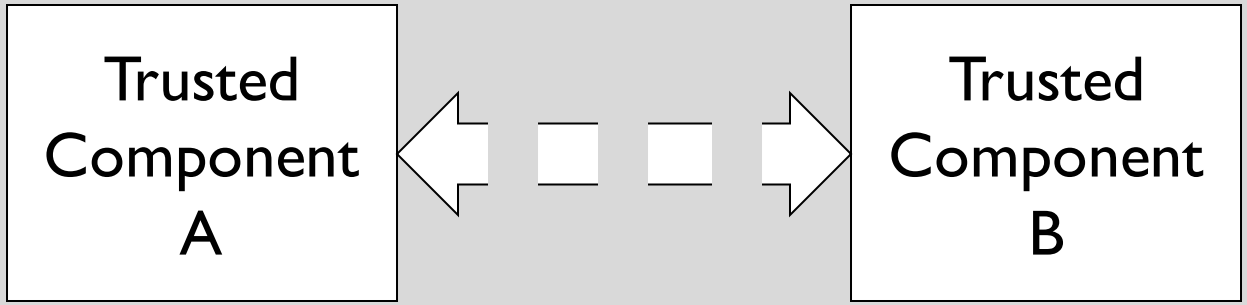
- Hide kernel starts
  - Support for a launch no-op that can be used as cover traffic
- Hide kernel stops
  - Do not notify the CPU that kernels have completed

Traffic transformations to hide execution time

Data Oblivious Streams

- Always send a fixed package of operations periodically
  - E.g., Launch kernel, Data movement
  - Ensures that all observed operations depend on time

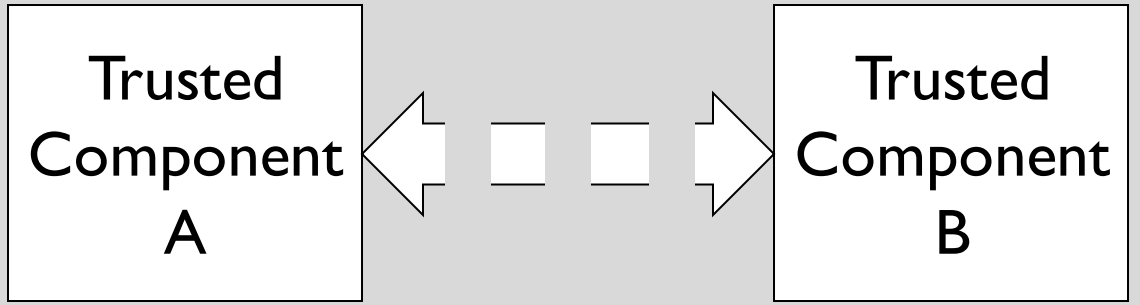
- Transform application operations into packages
  - Substitute no-op launches and dummy data copies if no work is available



Other challenges (omitted for time)

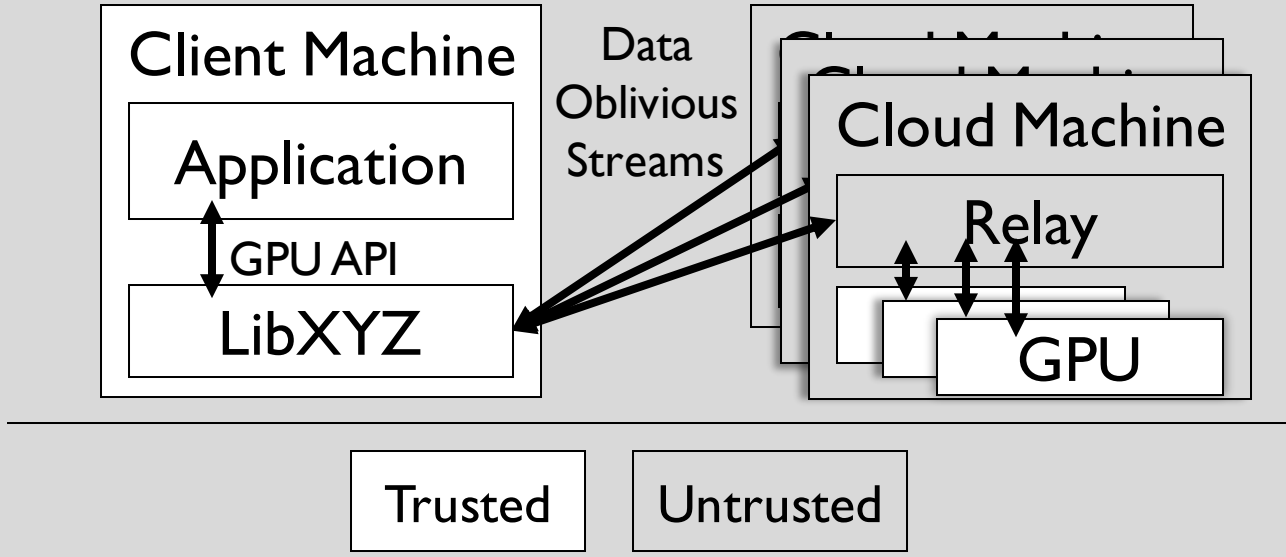
- All data and kernel launch operations must be the same size
- Applications express data dependencies that must be honored
- Need a new mechanism to detect kernel completion

System XYZ: Provides data oblivious streams



System XYZ: Provides data oblivious streams

- Trusted component A: Client Machine
  - Avoids known issues with CPU TEEs
- Trusted component B: GPU
- LibXYZ interposes on GPU API calls and applies transformations
  - Application code is unchanged
- Technical discussion deferred:
  - Relay does not execute in CPU TEE



Evaluation (preliminary)

The cloud machine:

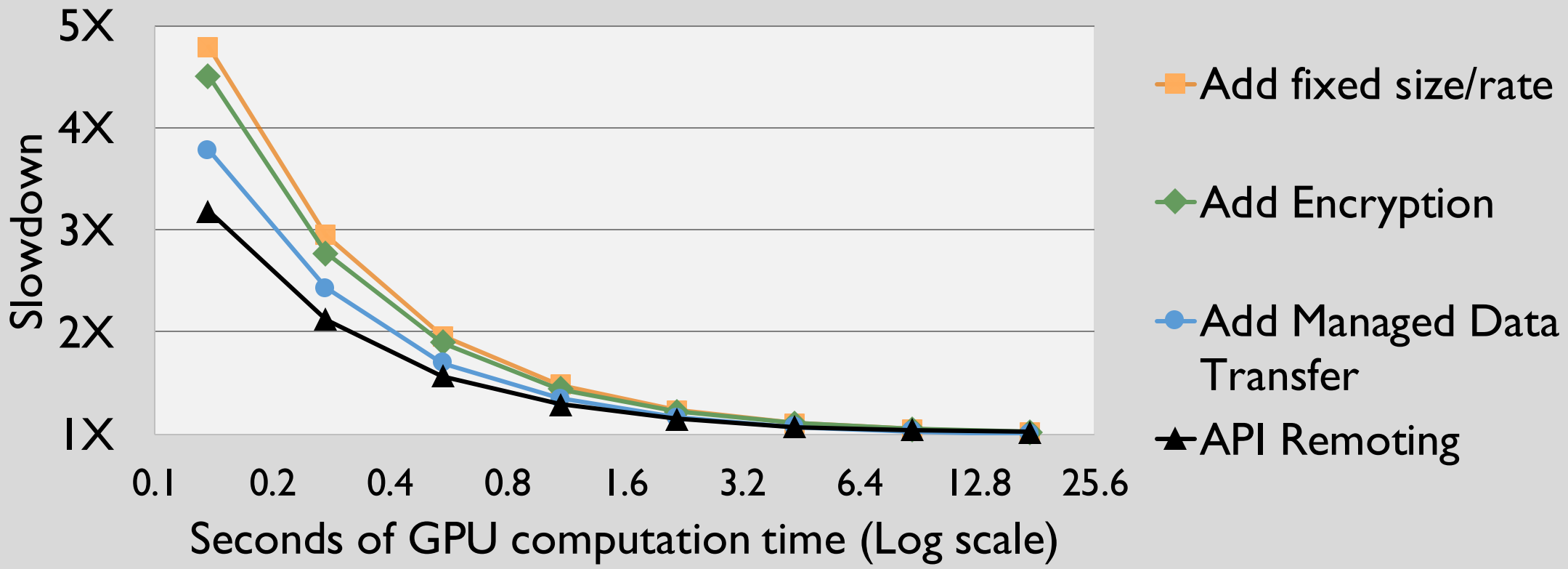
- Intel i9-9900K CPU with 8 cores @3.60GHz
- 32GB of RAM
- Radeon RX VEGA 64 GPU with 8GB of RAM

The client machine:

- Intel Xeon E3-1270 v6 processor with 4 cores @3.8GHz
- 32GB of RAM

Connected by a 1Gbps network

Slowdown vs GPU compute time



- System overheads decrease as GPU runtime increases
  - 16MB of input and output
  - Computation 4.4s, overhead is 11%

ResNet Training Slowdown