Extending Trusted Execution Environments in Architectural Simulators

Will Buziak
Department of Computer Science

Iris Bahar
Department of Computer Science

MINES | Computer Science

Abstract

Trusted Execution Environments (TEEs) provide hardware guarantees that seek to protect the security and isolation of application data.

Many proprietary TEEs exist, each with its own implementation and respective way of providing security.

Open-source TEEs like Keystone grant users the ability to contribute to their standard for security

In order to extend TEEs, developers need to either implement their designs on FPGAs, or turn to architectural simulators.

Lack of effective tools for development and thorough testing on real-world benchmarks make pre-fabrication development difficult.

This work outlines the methods for implementing and evaluating contributions to Keystone within the gem5 architecture simulator.

Motivation

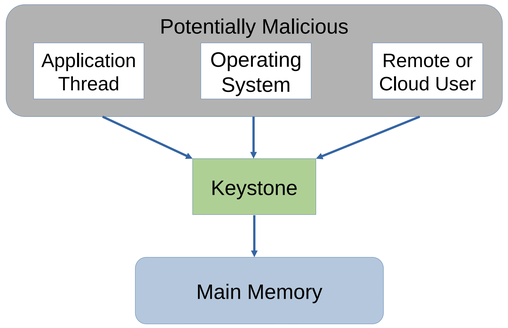
- Trusted Execution Environments are vulnerable to side-channel & replay attacks
- Keystone is a state-of-the-art RISC-V TEE designed for custom modularity, allowing developers to make their own contributions
- gem5 allows rapid development and real-world benchmarking and contains Keystone components
- Experimentation within gem5 allows a shorter pipeline from design idea to implementation testing, opposed to FPGA or fabrication testing

Challenges

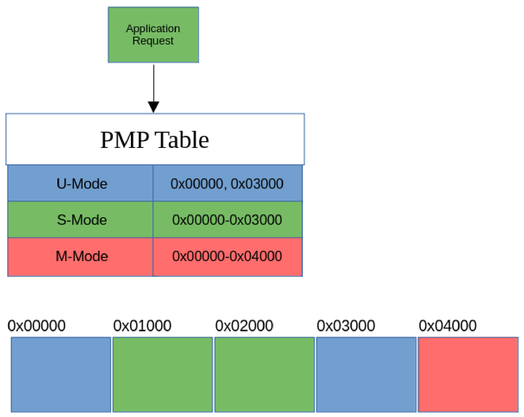
configuration by the designer
- Extending Keystone in simulation requires some re-configuration of gem5 class structures
- For this work, communication between each core and the Memory Encryption Engine requires defining either a new packet type or port

Keystone

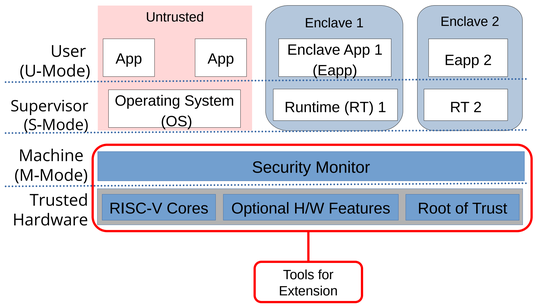
- Enclaves like Keystone provide hardware guarantees that data is safe from a malicious:
  - Application thread
  - Operating System
  - Remote User



- Keystone segments memory & defines rules for different threads



- Contributions have been proposed to extend the security of Keystone by providing secure memory protocols



gem5

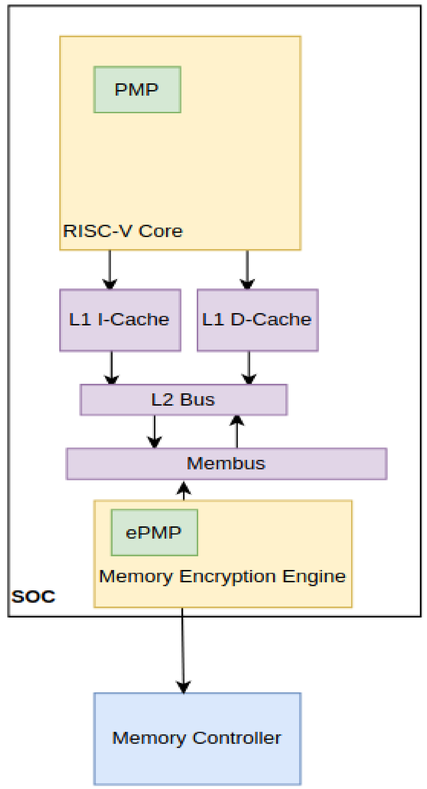
- gem5 presents architectural design as typical class structures with attributes based on the behavior of the real world component

- For example, to properly connect the PMP to the proposed ePMP, it is either necessary to write a new:

  - Packet type, bypassing the cache hierarchy

Or

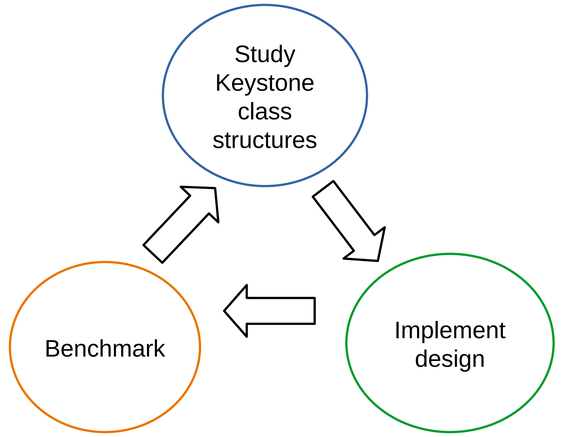
  - Port type directly connecting the core(s) to the Memory Encryption Engine



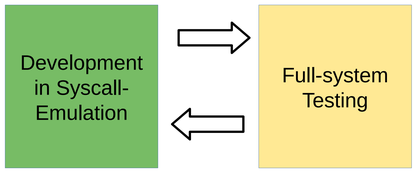
- Developers wishing to extend Keystone components can create their own version of the desired component's class structure, often requiring close attention to:
  - Port connections
  - Packet handling

Workflow

- In order to further protect state-of-the-art TEEs, Keystone is extended to include secure memory protocols in the gem5 simulation environment



Insights



- Designs in simulation can be tested on real workloads and state-of-the-art or custom benchmarks.