SANCTUM:

MINIMAL ARCHITECTURAL EXTENSIONS FOR ISOLATED EXECUTION

Victor Costan, Illia Lebedev, and Srinivas Devadas

Presented by: Adel Ejjeh

Outline

- Introduction
- Thread Model
- Programming Model
- Hardware Modifications
- Software Extensions
- Security Argument
- Performance Evaluation

INTRODUCTION

Introduction

- Intel SGX provides software isolation primitives in the CPU hardware
- Includes vast array of defenses against direct attacks
- BUT …
  - *No meaningful software isolation guarantees*
  - *Excludes side channel attacks*
  - *Significant implementation details not covered by publicly available documentation*

Contributions

- Software isolation scheme that provides:
  - *Provable defense against software side-channel attacks*
  - *Minimal and minimally invasive hardware modifications*
  - *Trusted software security monitor that is amenable to rigorous analysis*

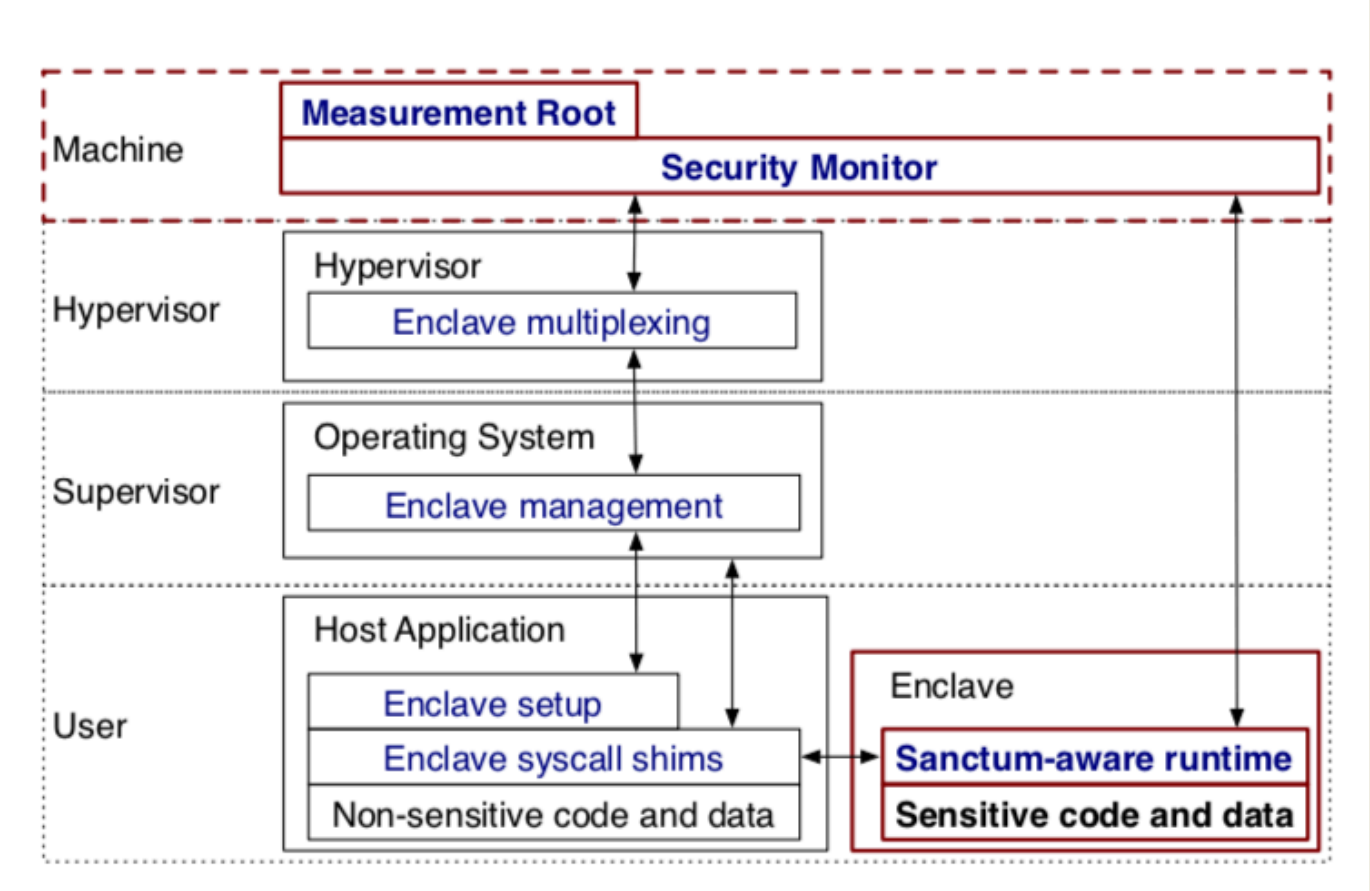
- Unprecedented Control
- Unprecedented Protection

THREAT MODEL

Threat Model

- Enclave receives encrypted information, decrypts it, and returns encrypted result
- Attacker can compromise OS and hypervisor and can launch rogue enclaves
- Attacker knows target architecture and microarchitecture
- Attacker can analyze passively collected data (page fault addresses) and launch active attacks (cache timing attacks)

PROGRAMMING MODEL



Software Stack

- Security Monitor replaces SGX's microcode
- Runs at highest privilege layer
- Enclave runs in user layer
- Tunnels syscalls through host application

Memory Arrangement

- Enclave Memory: regions of DRAM only accessible by enclave
  - *Stores enclave's page tables*
  - *Accessed using EVRANGE*
- OS Memory: regions of DRAM only accessible by OS
- Dedicated Metadata Region used by security monitor to store enclave metadata

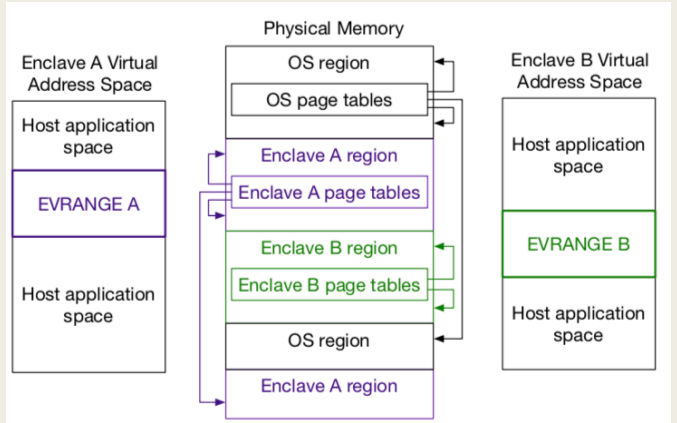


Figure 2: Per-enclave page tables

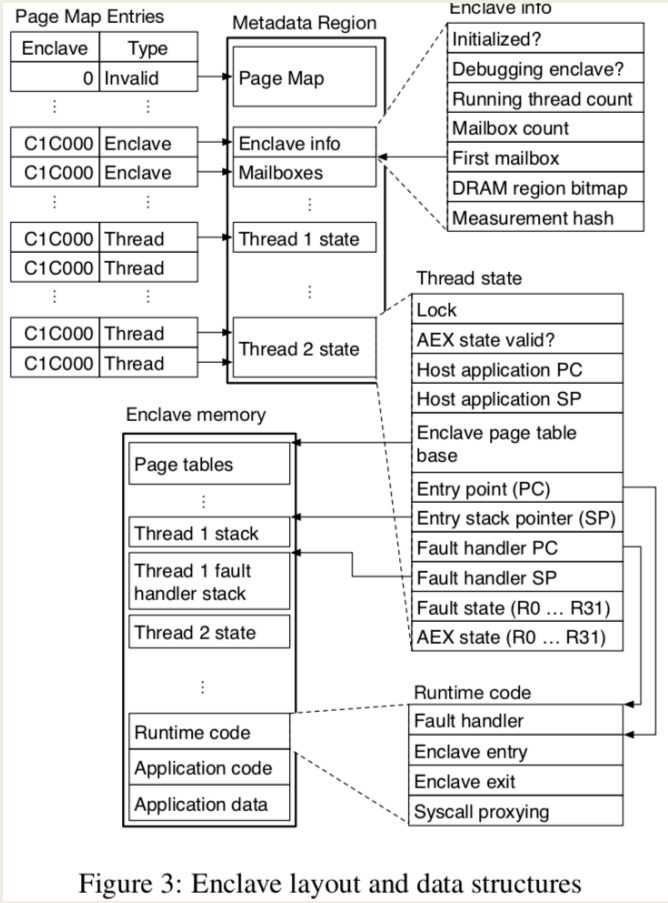


Figure 3: Enclave layout and data structures

HARDWARE MODIFICATIONS

LLC Address Input Transformation

- Page coloring and Address Shifting at LLC
- Separates DRAM into contiguous regions
- Ensures that there will be no sharing at LLC
  - *Pages that map to same set can only belong to one process (enclave or OS)*

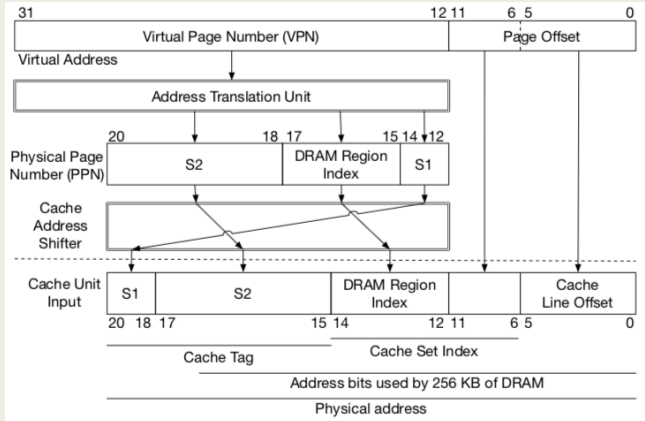
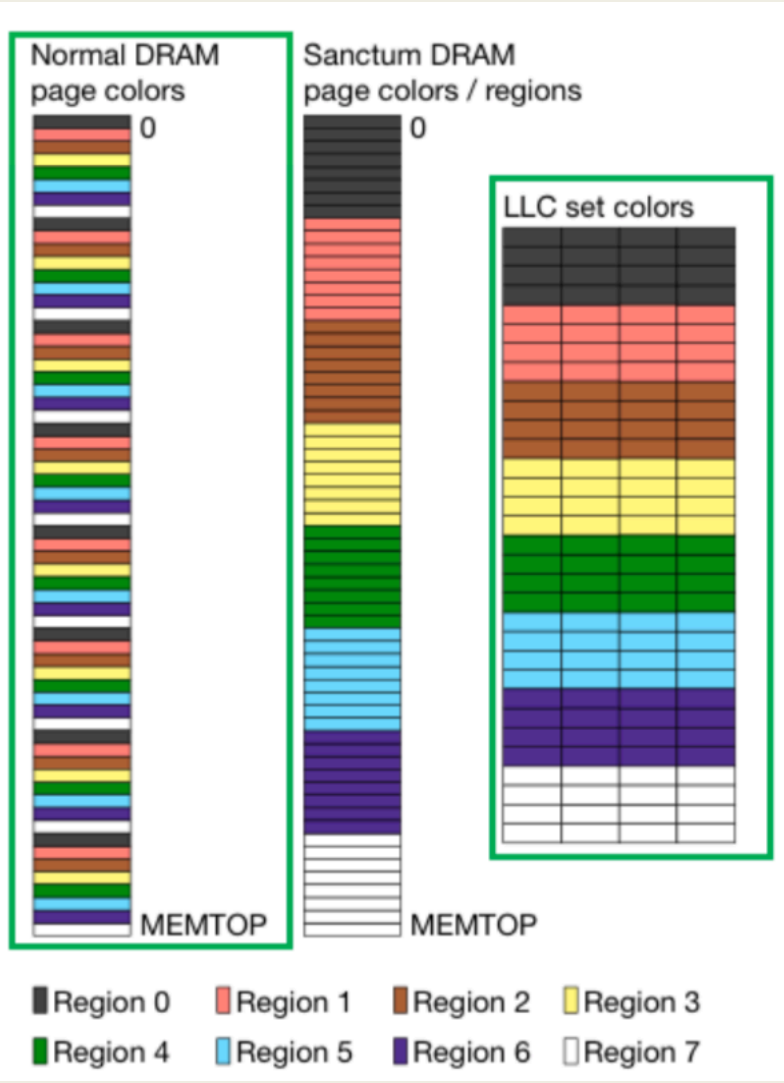


Figure 6: Cache address shifter, 3 bit PPN rotation



Page Walker

- Circuit at page walker input selects whether to walk enclave page tables or OS page tables
- Page walker also modified to check if DRAM address points to an accessible address region by checking it against a bit mask

SOFTWARE EXTENSIONS

Attestation Chain

- Measurement Root:
  - *Computes cryptographic hash of monitor*
  - *Generates Monitor's attestation key pair and certificate*
  - *Key can be stored in non-volatile memory to preserve between boots*
- Signing Enclave
  - *Responsible for running signing algorithm to compute attestation signatures*

Security Monitor

- Manages DRAM region allocation and enclaves
- Guards sensitive registers
- Cannot access enclave data since it's memory access patterns are not hidden to the OS
- Manages enclave metadata in special DRAM region
- Each region includes mailboxes used for inter-enclave communication
- Message exchange between enclave mailboxes managed by Monitor

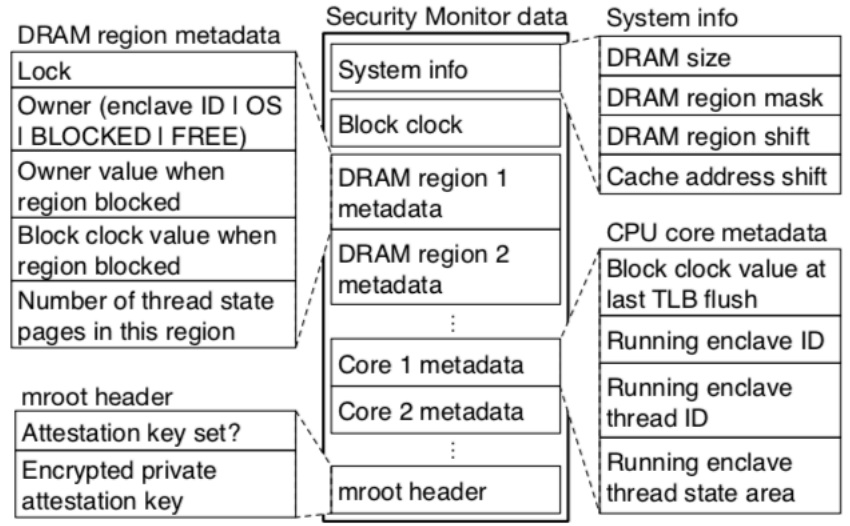


Figure 14: Security monitor data structures

Enclave Management

- Enclave lifecycle very similar to SGX
- OS creates and deletes enclaves using monitor API calls
- Enclave page mapping controlled by monitor to ensure isolation by disallowing page overlap
- Upon asynchronous exit (AEX), enclave state is saved and flag is set
  - *When enclave entered, AEX flag is checked and state is restored*
- OS can ask to reclaim DRAM region from enclave
  - *Enclave must occupy at least one DRAM region*

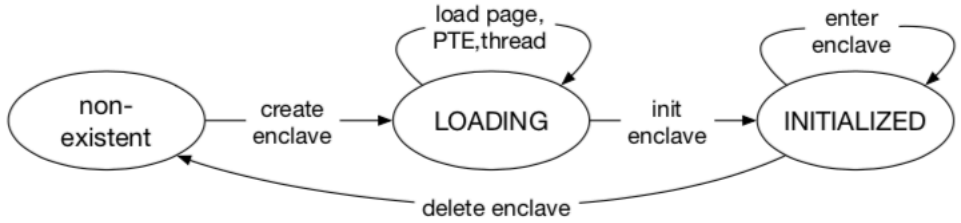


Figure 15: Enclave states and enclave management API calls

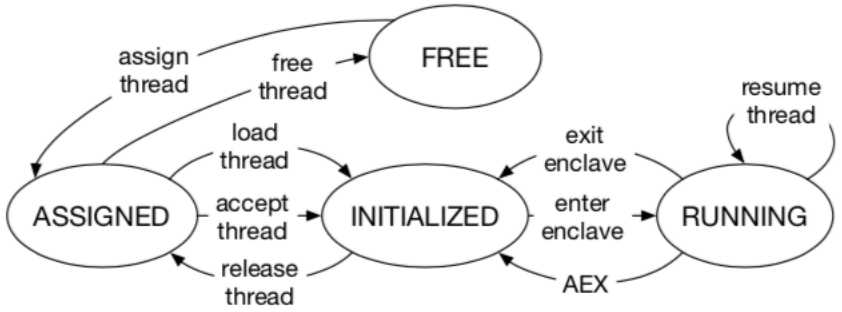


Figure 16: Enclave thread metadata structure states and thread-related API calls

SECURITY ARGUMENT

Enclave Protection

- Proof of DRAM isolation
  - *Prove that page walker modifications behave according to description*
  - *Prove that security monitor configures page walker in a way that prevents direct attacks*
- Proof of protection against page fault attacks
  - *Page faults in EVRANGE reported to enclave handler*
  - *Page tables are stored in enclave-only DRAM regions*
- Proof of protection against cache side-channel attacks
  - *Per-core isolation achieved by flushing on enclave exit*
  - *Isolation at LLC using coloring scheme and address shifting*

Security Monitor Protection

- Proof of protection against direct attacks (from OS or malicious enclave)
  - *Prove that page walker modifications are correct*
  - *Prove that monitor configures page walker correctly*
  - *Prove that monitor's DRAM regions not accessible to malicious enclaves*
- Proof of protection from cache timing attacks
  - *Monitor avoids making memory accesses that would reveal sensitive information by performing data-independent memory accesses (e.g. memcpy)*

PERFORMANCE EVALUATION

Experimental Setup

- Simulated 4-core 64-bit RISC-V in-order CPU
- Intel cache structure
  - *32 KB 8-way L1-D and L1-I*
  - *256 KB 8-way L2*
  - *8MB 16-way L3*
- 5GB memory divided into 64 DRAM regions
- SPECINT 2006 benchmarks

Analysis

- Cost of added hardware: <1% gates and <2% flip-flops
- Execution Overhead: