An Open Framework for Architecting TEEs

Dayeol Lee, David Kohlbrenner, Shweta Shinde, Dawn Song, and Krste Asanovic
Trusted Execution Environment (TEE)
Keystone: an Open Framework for Customizable TEEs

- **Modular and Extensible Design**
  - Extensible functional and security plugins
  - Implement new features without changing core primitive

- **Simple and Clean Abstractions**
  - Core security primitive: hardware-enforced isolation
  - Memory isolation with RISC-V standard PMP

- **Open Source Project**
  - Support research projects
  - Build an open community

Has been tested on QEMU, FPGA, and SoC

[keystone-enclave.org]
Keystone is NOT a Specific Design of TEE

Keystone is a framework for customizable TEEs
Keystone

An Open Framework for Architecting TEEs

- Customizable TEEs
- Keystone Framework
- Keystone Plugins
- Evaluation
TEEs are a Cornerstone Security Primitive

- Maximal Guarantees, Minimal Trust
  - Authenticate itself (device)
  - Authenticate software
  - Guarantee the integrity and privacy of remote execution

- Foundation for new security applications
  - Secure IoT sensor network
  - Decentralized applications (e.g., smart contracts/blockchain)
  - Lambda applications
  - Confidential computing in the cloud (e.g., machine learning)
A New Model: Customizable TEEs

- A framework provides building blocks of TEEs
- Both the **platform provider** and the **enclave developer** can customize what primitives and guarantees a TEE should employ
  - e.g., Software Defined Network (SDN)

<table>
<thead>
<tr>
<th>TEE Framework (Software)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure I/O</td>
</tr>
<tr>
<td>Side-Channel Protection</td>
</tr>
<tr>
<td>libc support</td>
</tr>
<tr>
<td>Secure Timer</td>
</tr>
<tr>
<td>Dynamic Allocation</td>
</tr>
<tr>
<td>Multithreading</td>
</tr>
<tr>
<td>Memory Encryption</td>
</tr>
<tr>
<td>Secure Boot</td>
</tr>
<tr>
<td>Source of Randomness</td>
</tr>
<tr>
<td>Memory Isolation</td>
</tr>
<tr>
<td>Attestation</td>
</tr>
</tbody>
</table>

Compatible Hardware
Why Do We Need Customizable TEEs?

- Diversity of Deployments
  - Servers to sensors

- Clean Interfaces Between Manufacturers, Providers, and Developers
  - Manufacturers cannot anticipate all needs or uses

- Minimize Trusted Computing Base (TCB)
  - Don’t have it if you don’t need it

- Rapid Development of Security Features
  - Research on defenses needs better starting places
Can We Customize Existing TEEs?
Why do we Need a New Framework?

- Specific threat model
- Slow to iterate
- Non-modifiable
- Only 2 hardware isolation domains
- Further isolation: software TCB

Existing TEE platforms are **fixed points** in the design space
RISC-V is The Best Place to Start Keystone

- Simple, Composable Primitives
  - Hardware-enforced memory isolation
  - Assists verification

- Privileged Programmable Layer
  - Enclaves with supervisor mode

- Open-Source Cores/SoCs
  - Amenable to HW/SW co-design
  - Verifiable

- Variety of Design and Deployment
  - Testing ground for all use-cases
Keystone

An Open Framework for Architecting TEEs

- Customizable TEEs
- Keystone Framework
- Keystone Plugins
- Evaluation
Keystone Trust Model
Keystone Trust Model

Higher Privilege

User (U-mode)

Untrusted

App

... App

Supervisor (S-mode)

Operating System (OS)

Enclave 1

Enclave App 1 (Eapp)

Runtime (RT) 1

Enclave 2

Eapp 2

Runtime (RT) 2

Machine (M-mode)

Security Monitor (SM)

Trusted Hardware

RISC-V Cores

Optional H/W Features

Root of Trust
Keystone Trust Model

- **Higher Privilege**
  - **User (U-mode)**
    - App
    - ... App
    - Operating System (OS)
  - **Supervisor (S-mode)**
  - **Machine (M-mode)**
    - Security Monitor (SM)
    - **Trusted Hardware**
      - RISC-V Cores
      - Optional H/W Features
      - Root of Trust

- **Enclave 1**
  - Enclave App 1 (Eapp)
  - Runtime (RT) 1

- **Enclave 2**
  - Eapp 2
  - RT 2
Keystone Trust Model

- **User (U-mode)**: Lower privilege
- **Supervisor (S-mode)**: Higher privilege
- **Operating System (OS)**
  - Enclave 1:
    - Enclave App 1 (Eapp)
    - Runtime (RT) 1
  - Enclave 2:
    - Eapp 2
    - RT 2

**Untrusted**
- App

**Security Monitor (SM)**
- RISC-V Cores
- Optional H/W Features
- Root of Trust
Keystone Trust Model

- User (U-mode)
  - App
  - ... App

- Supervisor (S-mode)
  - Operating System (OS)

- Machine (M-mode)
  - Security Monitor (SM)

- Trusted Hardware
  - RISC-V Cores
  - Optional H/W Features
  - Root of Trust

- Higher Privilege

- Enclave 1
  - Enclave App 1 (Eapp)
  - Runtime (RT) 1

- Enclave 2
  - Eapp 2
  - RT 2
Memory Isolation via RISC-V PMP

PMP registers

pmpcfg  pmpaddr

Priority

Physical Memory

SM | E2 | E1

Untrusted Context

SM | E2 | E1 | U1

Enclave (E1) Context

pmpcfg perm.

- rwx=000
- rwx=111
Memory Isolation via RISC-V PMP

PMP registers

```
| pmpcfg | pmpaddr |
```

Priority

Physical Memory

Untrusted Context

Enclave (E1) Context

```
| SM | E2 | E1 |
```

```
| SM | E2 | E1 | U1 |
```

```
pmpcfg perm.
☑ rwx=000
☐ rwx=111
```
Memory Isolation via RISC-V PMP

- **PMP registers**
  - `pmpcfg`
  - `pmpaddr`

- **Priority**
  - SM
  - E2
  - E1

- **Physical Memory**
  - Untrusted Context
  - Enclave (E1) Context

- **pmpcfg perm.**
  - `rwx=000`
  - `rwx=111`
How does Keystone Customize TEEs?
Keystone Workflow for Customizable TEEs

Provisioning

- Keystone
  - Customize
  - Security Monitor
  - Hardware

Platform Provider

Hardware Manufacturer

Development

- Keystone
  - Customize
  - Enclave App (eapp)
  - Runtime (RT)

Enclave Developer

Deployment

- eapp
- RT
- User
- OS
- SM
- Hardware
Keystone Workflow for Customizable TEEs

Development
- Eapp Sources
- Host Sources
- Enclave Configs

Keystone Framework (Enclave Developer)
- Keystone Libraries
- Keystone Tools
- RT Sources (seL4, Eyrie, ...)

Deployment
- Untrusted Host Bin.
- Eapp Bin.
- RT Bin.
- Enclave Hash (Eapp+RT)

Provisioning
- Platform Configs
- Platform PubKey
- Platform Spec.

Keystone Framework (Platform Provider)
- SM Sources
- SM Bin.
- SM Hash
- Platform PubKey
- Platform Spec.

Untrusted Machine
- Eapp
- Eapp
- Eapp
- Host
- User Proc.
- User Proc.

RT
- RT
- RT
- Untrusted OS

Keystone SM

Trusted Platform

Remote Verifier

Attestation
- Challenge
- Response
- Verify
Keystone Workflow for Customizable TEEs
Keystone

An Open Framework for Architecting TEEs

- Customizable TEEs
- Keystone Framework
- Keystone Plugins
- Evaluation
Keystone Plugins

- Composable Building Blocks for TEE
  - Configured during compilation
  - Threat models (e.g., Side-channel defense)
  - Workload (e.g., Dynamic resizing)

- Support Diverse Features w/ Minimal TCB
  - Virtual memory management
  - Untrusted I/O
  - Dynamic resizing
  - ...

<table>
<thead>
<tr>
<th>Component</th>
<th>Runtime</th>
<th>SM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>1730</td>
<td>1000</td>
</tr>
<tr>
<td>Memory Isolation</td>
<td>—</td>
<td>530</td>
</tr>
<tr>
<td>Free Memory</td>
<td>310</td>
<td>—</td>
</tr>
<tr>
<td>Dynamic Memory</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>Edge-call Handling</td>
<td>300</td>
<td>30</td>
</tr>
<tr>
<td>Syscalls</td>
<td>190</td>
<td>—</td>
</tr>
<tr>
<td>libc Environment</td>
<td>50</td>
<td>—</td>
</tr>
<tr>
<td>IO Syscall Proxying</td>
<td>260</td>
<td>—</td>
</tr>
<tr>
<td>Cache Partitioning</td>
<td>—</td>
<td>300</td>
</tr>
</tbody>
</table>

TCB LoC of each components
Free Memory Plugin

- Enclave can allocate **free memory** and manage MMU at run time
- Implemented on RT ~ 300 LoC
Dynamic Enclave Resizing Plugin

- Intel SGXv1 enclaves cannot resize after measurement
  - Cannot dynamically add new virtual pages
  - Intel took a few years to have “dynamic memory allocation” in SGXv2
  - Intel SGXv1 has < 100 MB physical memory limit

- Memory resizing in Keystone:
  - Enclave requests the OS to extend memory
  - OS calls an additional SM SBI “extend_enclave”
  - Took 2 engineer-days for prototyping (< 200 LoC)
Edge Call & Syscall Plugins

- Call interface between trusted and untrusted domain
- I/O System Calls (Proxy)
- Other System Calls
  e.g., mmap, brk, getrandom, ...
- Supporting libc functions
Off-the-shelf Runtime

- Boot **seL4 microkernel** as a runtime (~8,000 LoC)
- ~ 300 LoC modification for initialization
- Passes all seL4 tests (trivial overhead over native execution)

<table>
<thead>
<tr>
<th>Group (seL4 Test Suite)</th>
<th># Tests</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCALL</td>
<td>15</td>
<td>0.01</td>
</tr>
<tr>
<td>SERSERV_PARENT</td>
<td>10</td>
<td>0.67</td>
</tr>
<tr>
<td>CNODEOP</td>
<td>9</td>
<td>0.16</td>
</tr>
<tr>
<td>PAGEFAULT</td>
<td>9</td>
<td>0.69</td>
</tr>
<tr>
<td>IPC</td>
<td>9</td>
<td>0.74</td>
</tr>
<tr>
<td>Others</td>
<td>41</td>
<td>0.47</td>
</tr>
</tbody>
</table>
Keystone

An Open Framework for Architecting TEEs

- Customizable TEEs
- Keystone Framework
- Keystone Plugins
- Evaluation
Experimental Setup for Performance Evaluation

- Rocket/BOOM in FPGA (FireSim)
- FU540 in HiFive Unleashed Board

<table>
<thead>
<tr>
<th>Platform</th>
<th>Core #, Type</th>
<th>Cache Size (KB)</th>
<th>Latency (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>L1-I/D</td>
<td>L2</td>
</tr>
<tr>
<td>Rocket-S</td>
<td>1 in-order</td>
<td>8/8</td>
<td>512</td>
</tr>
<tr>
<td>Rocket</td>
<td>1 in-order</td>
<td>16/16</td>
<td>512</td>
</tr>
<tr>
<td>BOOM</td>
<td>1 OoO</td>
<td>32/32</td>
<td>2048</td>
</tr>
<tr>
<td>FU540</td>
<td>4 in-order</td>
<td>32/32</td>
<td>2048</td>
</tr>
</tbody>
</table>
CPU-Bound Benchmarks: CoreMark, Beebs

- Enclave init latency is almost proportional to the size
  - Enclave measurement dominates initialization

- No meaningful overhead in user application (±0.7%)
I/O Benchmark: IOZone

![Graph showing throughput vs file size for different baseline configurations.]

- **Baseline_r8**
- **Baseline_r128**
- **Baseline_r512**

**Throughput (KB/s)**

**File Size (KB)**

- 64
- 128
- 256
- 512
- 1K
- 2K
- 4K
- 8K
- 16K
- 32K
- 64K
- 128K
- 256K
- 512K

**Writer**

**Reader**
I/O Benchmark: IOZone

**Throughput (KB/s)**

- **Baseline_r8**
- **Baseline_r128**
- **Baseline_r512**
- **Keystone_r8**
- **Keystone_r128**
- **Keystone_r512**

**File Size (KB)**

- 64
- 128
- 256
- 512
- 1K
- 2K
- 4K
- 8K
- 16K
- 32K
- 64K
- 128K
- 256K
- 512K
Plugin Performance Trade-offs
Cache Partitioning Plugin

- 50:50 Cache Way-Partitioning with FU540 L2 controller
- Flush L1 + L2 partition on context switch
Cache Partitioning Plugin

- 50:50 Cache Way-Partitioning with FU540 L2 controller
- Flush L1 + L2 partition on context switch
Cache Partitioning Plugin

- 50:50 Cache Way-Partitioning with FU540 L2 controller
- Flush L1 + L2 partition on context switch
Cache Partitioning Plugin

- 50:50 Cache Way-Partitioning with FU540 L2 controller
- Flush L1 + L2 partition on context switch

- Overhead Depends on the Working Set Size
  - Up to 128.19% (miniz) L2 partitioning overhead
  - L1 flush overhead was trivial with the default context switch period (10 ms)
Cache Partitioning Plugin

- 50:50 Cache Way-Partitioning with FU540 L2 controller
- Flush L1 + L2 partition on context switch

- Overhead Depends on the Working Set Size
  - Up to 128.19% (miniz) L2 partitioning overhead
  - L1 flush overhead was trivial with the default context switch period (10 ms)
Dynamic Resizing Plugin

- Machine Learning (Inferencing in Torch, 9 Models, 2 Datasets)

- Keystone Overhead over Baseline
  - Min -3.12% (LeNet) due to lack of page faults
  - Max 7.35% (DenseNet) due to mmap implementation

- Reduced Initialization Latency with Dynamic Resizing
  - Runtime does not initialize free memory with dynamic resizing
Is Keystone Expressive Enough to Run Real-World Applications?
Real-World Use Cases

- Machine Learning Inference
  - Eyrie (3,000 LoC) + Torch Models (21,000-34,000 LoC)
  - seL4 (8,000 LoC) + FANN (14,000 LoC)

- Secure Communication, Crypto Libraries
  - Using libsodium
  - [https://github.com/keystone-enclave/keystone-demo](https://github.com/keystone-enclave/keystone-demo)
Conclusion

- **Keystone: an Open Framework for Architecting TEEs**
  - Customizable TEE for various threat models & workloads

- **Keystone Plugins**
  - Memory Management: free memory, dynamic resizing
  - Functionality: syscalls and muslibc support
  - Security: cache way partitioning

- **Evaluation**
  - Trade offs between performance, security, TCB, and functionality

[keystone-enclave.org](http://keystone-enclave.org)
Future Work

- TEE for Small Devices (e.g., embedded, microcontroller)
  - RV32, M/U-only ISA
- Formal Verification
  - Hardware, Security Monitor, and the Runtime
- Concurrent Multithreading
- More Plugins
  - Secure I/O, Sealing, Page Swapping, MEE ...
- More Front-end Support
  - RUST, Google Asylo, Microsoft OpenEnclave, ...
Keystone Team

Dayeol Lee  
UC Berkeley (Student)

David Kohlbrenner  
UC Berkeley (Postdoc)

Shweta Shinde  
UC Berkeley (Postdoc)

Dawn Song  
UC Berkeley (Faculty)

Krste Asanovic  
UC Berkeley (Faculty)

Alex Thomas  
UC Berkeley (Student)

Stephan Kaminsky  
UC Berkeley (Student)
Project Links

● Deployment:
  ○ QEMU / SiFive Unleashed: https://github.com/keystone-enclave/keystone
  ○ FireSim (FPGA): https://github.com/keystone-enclave/keystone-firesim

● Documentation:
  ○ Website/Blog: https://keystone-enclave.org
  ○ Development Docs: https://docs.keystone-enclave.org

● Technical Paper:
Thank You!

Dayeol Lee (dayeol@berkeley.edu)
David Kohlbrenner (dkohlbre@berkeley.edu)
Shweta Shinde (shwetas@berkeley.edu)
Forum (keystone-enclave-forum@googlegroups.com)
Announcement (keystone-enclave@googlegroups.com)
Operation Breakdown

- Measurement (SHA3) dominates enclave creation

- Context switch latency is trivial (~2.5 Kcycles)