Penglai-Enclave: Secure and Efficient RISC-V Enclave

Dong Du, Xu Lu, Erhu Feng, Bicheng Yang, Yubin Xia, Haibo Chen

IPADS, Shanghai Jiao Tong University
What’s **Enclaves**?

Hardware-assisted Trusted Execution Environment

```
<table>
<thead>
<tr>
<th>Host App</th>
<th>Enclave App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-sensitive code/data</td>
<td>Sensitive code/data</td>
</tr>
</tbody>
</table>

```

- **Host App**: Non-sensitive code/data
- **Enclave App**: Sensitive code/data
- **OS**:
- **Hypervisor**:
- **Hardware**: TEE/Enclave extension
- **User**:
- **Supervisor**:
- **Hypervisor**:
- **Hardware**:
Why Enclave for RISC-V?

Lessons learned:
Security is a necessary processor need

User needs:
Users will run sensitive code/data in RISC-V
**Why Another Enclave?**

**Shared Enclave**

**Shared Enclave Architecture**: A single HW-assisted protection zone for multiple enclaves

- **Pros**: low communication latency (intra-zone)
- **Cons**: higher TCB

**Dedicated Enclave**

**Dedicated Enclave Architecture**: A single HW-assisted protection zone for a single enclave

- **Pros**: small TCB → higher security-assurance
- **Cons**: long communication latency (inter-zone) & non-scalable
## Why Another Enclave?

<table>
<thead>
<tr>
<th>Systems</th>
<th>Performance</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td><strong>Name</strong></td>
<td><strong>Arch</strong></td>
</tr>
<tr>
<td>Dedicated Enclave</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGX</td>
<td>Intel</td>
<td>Slow</td>
</tr>
<tr>
<td>Haven</td>
<td>Intel</td>
<td>Slow</td>
</tr>
<tr>
<td>Graphene</td>
<td>Intel</td>
<td>Slow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shared Enclave</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustZone</td>
<td>ARM</td>
<td>Fast</td>
</tr>
<tr>
<td>OP-TEE</td>
<td>ARM</td>
<td>Fast</td>
</tr>
<tr>
<td>Komodo</td>
<td>ARM</td>
<td>Fast</td>
</tr>
<tr>
<td>Sanctuary</td>
<td>ARM</td>
<td>Fast</td>
</tr>
<tr>
<td>Penglai</td>
<td>RISC-V</td>
<td>Fast</td>
</tr>
</tbody>
</table>

Existing enclave systems can not achieve **security** and **performance** simultaneously.
PENGLAI-ENCLAVE (蓬莱)
Overview

- SW-HW Co-design Trusted Execution Environment (TEE)
  - Based on RISC-V ISA
  - Can run on any RISC-V core that supports Privileged ISA v1.10
  - IoT and Cloud

- Trusted Code Base
  - RISC-V core (PMP/sPMP) + Verifiable security monitor (M-mode privilege) + TEEOS

- Secure Assurance
  - Strong isolation between enclave and other application or OS
  - Protect against a malicious or compromised OS
  - Secure boot and remote attestation for chain of trust
  - High performance and scalability
Penglai with heavy.LIGHT Architecture

**heavy.LIGHT architecture**

- **LIGHT Zone**: A dedicated HW-isolated box for a single enclave
- **heavy Zone**: Multiple-Enclaves isolated through TEEOS

**TEEOS**: Leverage s-mode for enclave isolation (sPMP and PMP)
- Fast cross-enclave communication (IPC)
- Flexible resource management
- Fast startup
- Scalable instances
heavy.LIGHT Architecture

- **Security monitor:**
  - A small software running on M-mode
  - Enclave measurement and attestation
  - Manage enclave and provide isolation via physical memory isolation property

- **Physical Memory Isolation Property**
  - Restrict physical memory access of S- or U-mode software
  - Dynamically configurable by security monitor
Hardware Requirement

- RV32 or RV64
- All of the three modes (M/S/U)
- Support RISC-V priv. ISA v1.10
  - need sPMP or PT support for performance
- Larger, tamper-proof boot ROM (~1MB)
  - Trusted bootloader should be added to initialize the system
- Physical memory isolation support
  - New hardware property for memory isolation
- IOPMP extension
  - Defend malicious I/O access
SPMP (S-mode PMP)

• For IoT devices (MMU-less)
  – it is desirable to enable S-mode OS to limit the physical addresses accessible by U-mode software
SPMP (S-mode PMP)

- S-mode virtualization for scalable enclaves

(a) PMP-based isolation

(b) sPMP
Supervisor-mode Physical Memory Protection Keys

- sPMP entries
  - 8-bit configuration register
  - XLEN-bit address register
## Supervisor-mode Physical Memory Protection Keys

- **sPMP entries**
  - 8-bit configuration register
  - XLEN-bit address register

### Configuration Registers (RV32)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp3cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp6cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp9cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp12cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp15cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp1cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp4cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp7cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp10cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp13cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp11cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp14cfg</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>23</td>
<td>16</td>
<td>15</td>
<td>spmp15cfg</td>
</tr>
</tbody>
</table>

spmpcfg0 | spmpcfg1 | spmpcfg2 | spmpcfg3
## Supervisor-mode Physical Memory Protection Keys

- **sPMP entries**
  - 8-bit configuration register
  - XLEN-bit address register

### Configuration Register Format

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Lock</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>User</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Address</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>A(WARL)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>X(WARL)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Execute</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Write</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>Read</td>
<td>1</td>
</tr>
</tbody>
</table>

---

1. **L(WARL)**: 1
2. **U(WARL)**: 1
3. **WIRI**: 1
4. **A(WARL)**: 2
5. **X(WARL)**: 1
6. **W(WARL)**: 1
7. **R(WARL)**: 1
Supervisor-mode Physical Memory Protection Keys

• Address matching
  – Same as PMP

• Locking and privilege mode
  – The *Lock* bit indicates: the sPMP is locked to S-mode

• Priority and Matching Logic
  – The lowest-numbered sPMP entry
  – Failed accesses generate a page fault exception

Refer our proposal in **RISC-V/TEE group** for details!
SMAP and SMEP

- **SMAP (Supervisor Memory Access Prevention)**
  - leverage the SUM bit in the status register
  - SUM=0:
    - S-mode memory accesses to memory for U-mode (U=1) will **fault**
  - SUM=1:
    - these accesses are **permitted**

- **SMEP (Supervisor Memory Execution Prevention)**
  - Do not allow the S-mode to execute codes in physical memory that are for U-mode (U=1)

- Violations will trigger page faults
Chain of Trust: (1) Secure Boot

- The manufacturer provisions:
  - device key pair \( \{SK_{DEV}, PK_{DEV}\} \)
  - endorses the certificate
- During CPU resets: load bootloader from boot ROM and execute it
- The bootloader measures and signs the security monitor
- The secure monitor measures and signs the TEEOS
- The user can remotely attest the security monitor and TEEOS by
  - trusting the manufacturer’s certificate
  - comparing the measurement
  - and verifying the signature with \( PK_{DEV} \)
Chain of Trust: (2) Remote Attestation for *light* Zone

- (1) The bootloader provisions the attestation key pair \{SK_{SM}, PK_{SM}\}
- (2) The user uploads an executable to the system
- (3) The user asks SM to create an enclave and initialize it with the executable
- (4) The user asks SM to measure the initial state of the enclave
- (5) The SM measures the enclave, and signs it with the attestation private key
- The user can remotely attest the enclave by
  - comparing the measurement
  - and verifying it with PK_{SM}
Chain of Trust: (3) Remote Attestation for HEAVY Zone

- Same (1)-(4)
- (5) The SM measures the enclave through *(verified)* TEEOS, and signs it with the attestation private key
- The user can remotely attest the enclave by
  - comparing the measurement
  - and verifying it with $PK_{SM}$
Layered Memory Isolation

- HEAVY.light provides a **strong** and **flexible** physical memory isolation
  - RISC-V Physical Memory Protection (PMP)
  - PT / S-mode PMP* for TEEOS memory isolation

* : need HW extensions in HEAVY.light enclave design
MCU case

• Platform
  – N200 from NUCELI (芯来)
  – RV32 with sPMP support
  – No paging

• Enclaves chain
  – Up to 100 enclaves
• **Background**
  - Existing enclave systems is either *dedicated enclave* or *shared enclave*
  - Cannot achieve both *security* & *performance* simultaneously

- **Penglai-Enclave** is based on *heavy.LIGHT architecture*
  - Using light-zone and heavy-zone to achieve both performance and security
  - TEEOS for scalability

- **Cases**
  - siFive U500 (Xilinx VC707)
  - Nuclei N200

---

**Conclusion**