Towards Distributed and Virtualized Trusted Execution Environment

Zhiqiang Lin
zlin@cse.ohio-state.edu

12/10/2021

SPNCE’21
Trusted Execution Environment (TEE)
Trusted Execution Environment (TEE)

First TEE by Trusted Logic and TI

ARM TrustZone

Intel SGX

First SGX cloud instance by Aliyun

AMD SEV

SEV in Google Cloud Engine

Google Asylo

AWS Nitro Enclave

First Xeon server chip with SGX

Why TEE

- Hardware
- Operating Systems
- Hardware
Why TEE

Operating Systems

Hardware
Why TEE

- Operating Systems
- Hardware
Why TEE

- Motivations
- Background
- Virtualized TEE
- Distributed TEE
- Related Work
- Takeaway
- References

Hardware

Operating Systems
Why TEE

- Hardware
- Operating Systems
- Hardware
Why TEE

- Hardware
- Operating Systems
- Hardware
Why TEE

- Hardware
- Operating Systems
- Hardware
Why TEE

Virtualization

Operating Systems

Hardware

Virtualization

Operating Systems

Operating Systems
Why TEE

- Operating Systems
- Virtualization
- Hardware
Why TEE

Operating Systems

Virtualization

Hardware
Why TEE

Operating Systems

Virtualization

Hardware
Why TEE

- Operating Systems
- Virtualization
- Hardware

---

Motivations
Background
Virtualized TEE
Distributed TEE
Related Work
Takeaway
References
Why TEE

Operating Systems

Virtualization

Hardware

SGX
Why TEE

- Hardware
- Virtualization
- Operating Systems
- Operating Systems
- SGX
Sandbox vs. Reverse Sandbox
Sandbox vs. Reverse Sandbox

Enclave

{.js}

JavaScript
Numerous Applications w/ TEEs

1. SGX-based password manager [KKP+18]
2. SGX-based anonymity network [KHH+17]
3. Privacy-preserving data analytics (e.g., [SCF+15]) and machine learning (e.g., [KPM+16, OSF+16]),
4. SGX-based game protection [PAL20]
5. Privacy-preserving contact-tracing (e.g., SafeTrace [eni]) and blockchains [CXZW21]
Our Prior Works (Since 2016)

3. “Securing Data Analytics on SGX With Randomization”. ESORICS 2017
10. “vSGX: Virtualizing SGX on AMD SEV”. Oakland 2022
Problems in Current TEEs: (1) Vendor lock-in and (2) Slow evolution

1. SGX is an ISA extension
2. Apps have to be written specifically for SGX and can’t run elsewhere (e.g., you can’t run it on AMD chips)
3. Numerous side channel vulnerabilities and attacks (e.g., [XCP15, SCNS16, SWG+17, BMD+17, HCP17, GESM17, LSG+17, VBMW+18, vSMÖ+19, SLM+19, CGG+19, CCX+19]). The defense requires faster patching
Recent Trend

1. Unifying TEE SDKs
   - Asoly by Google [Goo], by Vmware [VMw]

2. Software defined TEE
   - Komodo [FBHP17], but not binary compatible
The Need to Decouple TEEs from the Hardware (Virtualized TEE)

Recent Trend

1. Unifying TEE SDKs
   - Asoly by Google [Goo], by Vmware [VMw]
2. Software defined TEE
   - Komodo [FBHP17], but not binary compatible

Benefits of Software Defined TEE

1. No vendor lock-in
2. Flexibility on deployment
3. Faster feature evolution
4. Faster bug fixes
5. Migration
The Need to Decouple TEEs from the Hardware (Virtualized TEE)

Recent Trend

1. Unifying TEE SDKs
   - Asoly by Google [Goo], by Vmware [VMw]

2. Software defined TEE
   - Komodo [FBHP17], but not binary compatible

Benefits of Software Defined TEE

1. No vendor lock-in
2. Flexibility on deployment
3. Faster feature evolution
4. Faster bug fixes
5. Migration

Our goal: Can we design a binary compatible, software defined TEE? If so, how?
The Need of Distributed (Edge) TEE
The Need of Distributed (Edge) TEE

CIA:
1. Convenience
2. Intelligence
3. Automation
The Need of Distributed (Edge) TEE

Security + Privacy
How Could the Distributed (Edge) TEE Look Like?
How Could the Distributed (Edge) TEE Look Like?
How Could the Distributed (Edge) TEE Look Like?
How Could the Distributed (Edge) TEE Look Like?
The Benefits of Distributed (Edge) TEE

1. Distributed (no single point of failure, no leakage from centrals)
2. Near-data computing (efficient)
3. Privacy-preserving
4. Integrity-guarantee (provided by TEE)
### SGX 101: SGX Instructions

<table>
<thead>
<tr>
<th>SGX Version</th>
<th>User Space enclu</th>
<th>Kernel Space encls</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGX-v1</td>
<td>5</td>
<td>13</td>
<td>18</td>
</tr>
<tr>
<td>SGX-v2</td>
<td>5+3</td>
<td>13+3</td>
<td>18+6</td>
</tr>
</tbody>
</table>
# SGX 101: SGX Instructions

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Type</th>
<th>Instruction</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>MEM</td>
<td>EADD</td>
<td>Add a page</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>EBLOCK</td>
<td>Block an EPC page</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>EXE</td>
<td>ECREATE</td>
<td>Create an enclave</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>DBG</td>
<td>EDBGRD</td>
<td>Read data by debugger</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>DBG</td>
<td>EDBGWR</td>
<td>Write data by debugger</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>EEEXTEND</td>
<td>Extend EPC page measurement</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>EXE</td>
<td>EINIT</td>
<td>Initialize an enclave</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>ELDB</td>
<td>Load an EPC page as blocked</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>ELDU</td>
<td>Load an EPC page as unblocked</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>SEC</td>
<td>EPA</td>
<td>Add version array</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>EREMOVE</td>
<td>Remove a page from EPC</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>ETRACK</td>
<td>Activate EBLOCK checks</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>EB</td>
<td>Write back/invalidate an EPC page</td>
<td>v1</td>
</tr>
<tr>
<td>P</td>
<td>MEM</td>
<td>EAUG</td>
<td>Allocate page to an existing enclave</td>
<td>v2</td>
</tr>
<tr>
<td>P</td>
<td>SEC</td>
<td>EMODPR</td>
<td>Restrict page permissions</td>
<td>v2</td>
</tr>
<tr>
<td>P</td>
<td>EXE</td>
<td>EMODT</td>
<td>Change the type of an EPC page</td>
<td>v2</td>
</tr>
<tr>
<td>U</td>
<td>EXE</td>
<td>EENTER</td>
<td>Enter an enclave</td>
<td>v1</td>
</tr>
<tr>
<td>U</td>
<td>EXE</td>
<td>EEXIT</td>
<td>Exit an enclave</td>
<td>v1</td>
</tr>
<tr>
<td>U</td>
<td>SEC</td>
<td>EGETKEY</td>
<td>Create a cryptographic key</td>
<td>v1</td>
</tr>
<tr>
<td>U</td>
<td>SEC</td>
<td>EREPORT</td>
<td>Create a cryptographic report</td>
<td>v1</td>
</tr>
<tr>
<td>U</td>
<td>EXE</td>
<td>ERESUME</td>
<td>Re-enter an enclave</td>
<td>v1</td>
</tr>
<tr>
<td>U</td>
<td>MEM</td>
<td>EACCEPT</td>
<td>Accept changes to a page</td>
<td>v2</td>
</tr>
<tr>
<td>U</td>
<td>SEC</td>
<td>EMODPE</td>
<td>Enhance access rights</td>
<td>v2</td>
</tr>
<tr>
<td>U</td>
<td>MEM</td>
<td>EACCEPTCOPY</td>
<td>Copy page to a new location</td>
<td>v2</td>
</tr>
</tbody>
</table>
SGX 101: Enclave Initialization
SGX 101: Enclave Initialization
SGX 101: Enclave Initialization
SGX 101: Enclave Initialization
SGX 101: Enclave Initialization
SGX 101: EENTER, EEXIT; AEX, ERESUME

Enclave

entry_point_1:
...
...
entry_point_2:
...
...
...

mov retval, %rax
EEXIT

Untrusted App

do_ecall:
EENTER
post_ecall:
...

OS

Intel SGX CPU
SGX 101: EENTER, EEXIT; AEX, ERESUME

Enclave

... idiv $0, %rax
...

Untrusted App

aep:
...

ERESUME
...

Intel SGX CPU

OS
SGX 101: Enclave Memory in SGX

Enclave

... 
```asm
mov var, %rax 
... 
```

tvar:
```asm
.byte 00 00 
```

Untrusted App

```asm
var: 
.byte 00 00 
```

OS

Intel SGX CPU
SGX 101: Attestation in SGX

- **Motivations**
- **Background**
- **Virtualized TEE**
- **Distributed TEE**
- **Related Work**
- **Takeaway**
- **References**

---

**Intel SGX CPU**

**OS**

**Untrusted App**

**My Server**

**Internet**

**My Enclave**

...?
SGX 101: Attestation in SGX
AMD Secure Encrypted Virtualization (SEV)
## Intel SGX vs AMD SEV

<table>
<thead>
<tr>
<th>TEE</th>
<th>Highest Access Level</th>
<th>Memory Size Limits</th>
<th>Software Change</th>
<th>Platform Attestation Mechanism</th>
<th>Protection Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGX</td>
<td>Ring 3</td>
<td>Up to 128MB EPC</td>
<td>Required</td>
<td>Intel Remote Attestation Protocol and IAS</td>
<td>Confidentiality and Integrity of the Code and Data in the Enclave</td>
</tr>
<tr>
<td>SEV</td>
<td>Ring 0</td>
<td>Up to Available System Ram</td>
<td>Only Hypervisor and VM's Kernel</td>
<td>AMD Secure Processor</td>
<td>Confidentiality of the VM's Memory Image</td>
</tr>
</tbody>
</table>
vSGX: Virtualizing SGX Enclaves on AMD SEV

Motivations
Background
Virtualized TEE
Distributed TEE
Related Work
Takeaway
References

vSGX: Virtualizing SGX Enclaves on AMD SEV

Intel SGX

Enclave
Untrusted App
OS
Intel SGX CPU

App 1
App 2
VM 1
VM 2
Hypervisor
AMD SEV CPU

AMD SEV

Enclave
App
VM 1
VM 2
Hypervisor
AMD SEV CPU

Shixuan Zhao, Mengyuan Li, Yinqian Zhang, and Zhiqiang Lin. S&P'22.
**vSGX: Virtualizing SGX Enclaves on AMD SEV**

"vSGX: Virtualizing SGX Enclaves on AMD SEV". Shixuan Zhao, Mengyuan Li, Yinqian Zhang, and Zhiqiang Lin. S&P’22.
Design Goals

1️⃣ Binary compatibility
2️⃣ Comparable security guarantee with both SGX and SEV
3️⃣ Reasonable performance
Design Goals

1. Binary compatibility
2. Comparable security guarantee with both SGX and SEV
3. Reasonable performance

vSGX should work like an SGX module plugged onto an SEV machine
Challenges

1. How to isolate the enclave from others components?
2. How to execute SGX instructions on SEV?
3. How to handle memory access inside an enclave?
4. How to connect between an enclave and other components?
SGX Instructions Emulation in vSGX

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EEADD</td>
<td>Add an page to an uninitialized enclave</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>• ↔ •</td>
<td>4185</td>
<td>19</td>
</tr>
<tr>
<td>EAUG</td>
<td>Add an page to an initialized enclave</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>• ↔ •</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>EBLOCK</td>
<td>Block an EPC page</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>• ↔ •</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>ECREATE</td>
<td>Create a SECS page in EPC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>• ↔ •</td>
<td>4105</td>
<td>19</td>
</tr>
<tr>
<td>EDBGRD</td>
<td>Read from a debug enclave</td>
<td>✗</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EDBGWR</td>
<td>Write to a debug enclave</td>
<td>✗</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EEEXTEND</td>
<td>Extend uninitialized enclave’s measurement</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>EINIT</td>
<td>Initialize an enclave</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>2137</td>
<td>19</td>
</tr>
<tr>
<td>ELD/EELDU</td>
<td>Load a page to enclave</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>8370</td>
<td>4131</td>
</tr>
<tr>
<td>EMODPR</td>
<td>Restrict an EPC page’s permission</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>EMODT</td>
<td>Change an EPC page’s type</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>EPA</td>
<td>Add version array</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>9</td>
<td>4131</td>
</tr>
<tr>
<td>EREMOVX</td>
<td>Remove a page from EPC</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>ETRACK</td>
<td>Block until EBLOCK is done</td>
<td>✗</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EWB</td>
<td>Write an EPC page to main memory</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• ↔ •</td>
<td>4137</td>
<td>8355</td>
</tr>
<tr>
<td>EACCEPT</td>
<td>Accept changes to an EPC page</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EACCEPTCOPY</td>
<td>Copy a page to a new EPC page</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ENTER</td>
<td>Enter an enclave</td>
<td>✓</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>177</td>
<td>19</td>
</tr>
<tr>
<td>EEXIT</td>
<td>Exit an enclave</td>
<td>✓</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>153</td>
<td>-</td>
</tr>
<tr>
<td>EGETKEY</td>
<td>Derive a key</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EMODEX</td>
<td>Extend permission of an EPC page</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EREPORT</td>
<td>Create a cryptographic report</td>
<td>✓</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ERESUME</td>
<td>Resume an enclave</td>
<td>✓</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>• → •</td>
<td>33</td>
<td>19</td>
</tr>
</tbody>
</table>

Behavior:

AEX Exit an enclave due to interrupt or fault | ✓ * | - | ✗ | ✗ | ✗ | • → • | 166 | - |
Cross VM Communication

VM1 – Cross-VM Communication

- Source
- Crypto Engine
- Sender
- Data Packer

VM2 – Cross-VM Communication

- Crypto Engine
- Dispatcher
- Destination
- Dispatch Queue
- IRQ Handler

Hypervisor – vSGX Hub

- CPUID Handler
- Send Worker
- Send Queue
Cross VM Communication

**VM2** – Cross-VM Communication

- Crypto Engine
- Dispatcher
- Dispatch Queue
- IRQ Handler

**VM1** – Cross-VM Communication

- Source
- Dispatcher
- Destination
- Crypto Engine

**Hypervisor** – vSGX Hub

- Send Worker
- Send Queue
- CPUID Handler
- Data Packer
Cross VM Communication

VM2 – Cross-VM Communication
- Crypto Engine
- Dispatcher
- Dispatch Queue
- IRQ Handler

VM1 – Cross-VM Communication
- Source
- Crypto Engine
- Sender
- Data Packer

Hypervisor – vSGX Hub
- CPUID Handler
- Send Worker
- Send Queue
Cross VM Communication

VM2 – Cross-VM Communication
- Crypto Engine
- Dispatcher
- Dispatch Queue
- Destination
- IRQ Handler

VM1 – Cross-VM Communication
- Source
- Crypto Engine
- Sender
- Data Packer
- ②

Hypervisor – vSGX Hub
- CPUID Handler
- Send Worker
- ③
- Send Queue
- ❶
- ❸
- ❹

23 / 38
Cross VM Communication

VM2 – Cross-VM Communication
- Crypto Engine
- Dispatcher
- Dispatch Queue
- IRQ Handler

VM1 – Cross-VM Communication
- Source
- Crypto Engine
- Sender
- Data Packer

Hypervisor – vSGX Hub
- CPUID Handler
- Send Worker
- Send Queue
Cross VM Communication

**VM2** – Cross-VM Communication
- Crypto Engine
- Dispatcher
- Dispatch Queue
- IRQ Handler
- Destination

**VM1** – Cross-VM Communication
- Source
- Sender
- Data Packer
- Crypto Engine

**Hypervisor** – vSGX Hub
- Send Worker
- Send Queue
- CPUID Handler

Diagram:
- Step ①: Dispatch Queue
- Step ②: Send Worker
- Step ③: CPUI Handler
- Step ④: Send Queue
- Step ⑤: Crypto Engine
Cross VM Communication

VM2 – Cross-VM Communication
- Crypto Engine
- Dispatcher
- Dispatch Queue
- IRQ Handler
- Destination

VM1 – Cross-VM Communication
- Source
- Data Packer
- Sender

Hypervisor – vSGX Hub
- CPUID Handler
- Send Worker
- Send Queue
Cross VM Communication

Hypervisor – vSGX Hub

VM1 – Cross-VM Communication

- Source
- Crypto Engine
- Data Packer

VM2 – Cross-VM Communication

- Crypto Engine
- Dispatcher
- Dispatch Queue
- IRQ Handler
- Destination

① Dispatch Queue
② Source
③ Send Queue
④ Send Worker
⑤ CPUID Handler
⑥ IRQ Handler
⑦ Dispatcher
Cross VM Communication

VM2 – Cross-VM Communication

Crypto Engine

Dispatcher

Dispatch Queue

IRQ Handler

VM1 – Cross-VM Communication

Source

Crypto Engine

Sender

Data Packer

Hypervisor – vSGX Hub

CPUID Handler

Send Worker

Send Queue
Cross VM Communication

**VM2** – Cross-VM Communication

1. **Crypto Engine**
2. **Dispatcher**
3. **Dispatch Queue**
4. **Destination**
5. **IRQ Handler**

**VM1** – Cross-VM Communication

1. **Source**
2. **Crypto Engine**
3. **Data Packer**
4. **Sender**
5. **Dispatch Queue**
6. **Send Worker**
7. **CPUID Handler**
8. **Send Queue**

**Hypervisor** – vSGX Hub
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT

EVM

- New Thread
- Manager
  - Trap Handler
  - EENTER Handler
  - AEX Handler
  - ERESUME Handler

AVM

- App
- Trap Handler
- Fault Handler

1. EENTER
2. EENTER Req.
3. EENTER Response
4. EENTER
5. EENTER
6. EENTER
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: EENTER, EEXIT
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME

![Diagram of EVM and AVM with various handlers and managers]

- **EVM**
  - New Thread
  - Manager
  - Trap Handler
  - EENTER Handler
  - AEX Handler
  - ERESUME Handler

- **AVM**
  - App
  - Trap Handler
  - Fault Handler
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME

EVM

1. Fault

- Trap Handler
- EENTER Handler
- AEX Handler
- ERESUME Handler
- New Thread
- Manager

AVM

- App
- Trap Handler
- Fault Handler
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME

- Motivations
- Background
- Virtualized TEE
- Distributed TEE
- Related Work
- Takeaway
- References

**Diagram:**

- **EVM**
  - Fault
  - Trap Handler
    - Manager
    - EENTER Handler
    - AEX Handler
    - ERESUME Handler
  - ② Wait for ERESUME

- **AVM**
  - App
  - Trap Handler
  - Fault Handler
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME

1. Fault
2. Wait for ERESUME
3. Send AEX Req.
4. Send ERESUME
5. Trap Handler

EVM
- Fault
- Trap Handler
- Manager
- EENTER Handler
- AEX Handler
- ERESUME Handler

AVM
- App
- Trap Handler
- Fault Handler
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME

Diagram:

- **EVM**
  - Fault
  - Wait for ERESUME
- **AVM**
  - App
  - ERESUME
  - ERESUME Req.
  - Go to AEP

Steps:

1. Fault
2. Wait for ERESUME
3. Send AEX Req.
4. ERESUME Req.
5. Go to AEP
6. ERESUME
7. ERESUME
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME

EVM

1. Fault
2. Wait for ERESUME
3. Send AEX Req.
4. ERESUME
5. ERESUME Request
6. ERESUME Request

AVM

1. Fault
2. ERESUME
3. ERESUME Request
4. Go to AEP

New Thread
Manager
AEX Handler
ERESUME Handler
App
Trap Handler
Fault Handler
Trap Handler
New Thread
Control Flow Transferring: Asynchronized Enclave Exit, ERESUME

**Diagram:**

- **EVM**
  - Trap Handler
  - EENTER Handler
  - AEX Handler
  - ERESUME Handler
  - New Thread

- **AVM**
  - App
  - Trap Handler
  - Fault Handler

**Steps:**

1. **Fault**
2. **Wait for ERESUME**
3. **Send AEX Req.**
4. **Send ERESUME Req.**
5. **Go to AEP**
6. **ERESUME**
7. **ERESUME Handled**
8. **ERESUME Response**
9. **Return**

**Notes:**

- ERESUME: Enclave Resumption
- New Thread: A new thread is created in the enclave
- Trap Handler: Handles traps and exceptions
- Fault Handler: Handles faults within the enclave
- AEX: Asynchronous Enclave Exit
- EENTER: Enclave Enter
- ERESUME: Enclave Resumption
Performance - BYTEmark

(a) CPU Intensive Test
(b) Memory Intensive Test
(c) FP Intensive Test
Performance - Graphene

(a) Time Consumption Launching Graphene SGX on vSGX

(b) cURL Execution Time

(c) GMPbench 0.2 Score
## Performance - wolfCrypt

<table>
<thead>
<tr>
<th></th>
<th>vSGX</th>
<th>Intel SGX</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MB/s</td>
<td>MB/s</td>
<td></td>
</tr>
<tr>
<td>RNG</td>
<td>82.57</td>
<td>117.51</td>
<td>1.42</td>
</tr>
<tr>
<td>AES-128-CBC-enc</td>
<td>187.36</td>
<td>363.82</td>
<td>1.94</td>
</tr>
<tr>
<td>AES-128-CBC-dec</td>
<td>172.59</td>
<td>399.39</td>
<td>2.31</td>
</tr>
<tr>
<td>AES-192-CBC-enc</td>
<td>156.95</td>
<td>309.70</td>
<td>1.97</td>
</tr>
<tr>
<td>AES-192-CBC-dec</td>
<td>184.4</td>
<td>341.43</td>
<td>1.85</td>
</tr>
<tr>
<td>AES-256-CBC-enc</td>
<td>139.01</td>
<td>269.16</td>
<td>1.94</td>
</tr>
<tr>
<td>AES-256-CBC-dec</td>
<td>123.05</td>
<td>291.93</td>
<td>2.37</td>
</tr>
<tr>
<td>AES-128-GCM-enc</td>
<td>54.10</td>
<td>94.98</td>
<td>1.76</td>
</tr>
<tr>
<td>AES-128-GCM-dec</td>
<td>56.02</td>
<td>94.99</td>
<td>1.70</td>
</tr>
<tr>
<td>AES-192-GCM-enc</td>
<td>54.36</td>
<td>90.29</td>
<td>1.66</td>
</tr>
<tr>
<td>AES-192-GCM-dec</td>
<td>54.49</td>
<td>90.16</td>
<td>1.65</td>
</tr>
<tr>
<td>AES-256-GCM-enc</td>
<td>51.78</td>
<td>86.79</td>
<td>1.68</td>
</tr>
<tr>
<td>AES-256-GCM-dec</td>
<td>49.74</td>
<td>86.64</td>
<td>1.74</td>
</tr>
<tr>
<td>ARC4</td>
<td>138.05</td>
<td>478.18</td>
<td>3.46</td>
</tr>
<tr>
<td>RABBIT</td>
<td>222.37</td>
<td>710.37</td>
<td>3.19</td>
</tr>
<tr>
<td>3DES</td>
<td>22.60</td>
<td>39.05</td>
<td>1.73</td>
</tr>
<tr>
<td>MD5</td>
<td>296.77</td>
<td>820.75</td>
<td>2.77</td>
</tr>
<tr>
<td>SHA</td>
<td>223.09</td>
<td>661.65</td>
<td>2.97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>vSGX</th>
<th>Intel SGX</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MB/s</td>
<td>MB/s</td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>115.56</td>
<td>298.76</td>
<td>2.59</td>
</tr>
<tr>
<td>HMAC-MD5</td>
<td>377.70</td>
<td>821.12</td>
<td>2.17</td>
</tr>
<tr>
<td>HMAC-SHA</td>
<td>381.57</td>
<td>662.07</td>
<td>1.74</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td>164.82</td>
<td>298.90</td>
<td>1.81</td>
</tr>
<tr>
<td>PBKDF2</td>
<td>9.49</td>
<td>34.63</td>
<td>3.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>vSGX</th>
<th>Intel SGX</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>op/s</td>
<td>op/s</td>
<td></td>
</tr>
<tr>
<td>RSA 2048 Public</td>
<td>10264.09</td>
<td>8443.25</td>
<td>0.82</td>
</tr>
<tr>
<td>RSA 2048 Private</td>
<td>188.40</td>
<td>146.93</td>
<td>0.78</td>
</tr>
<tr>
<td>DH 2048 Key Gen</td>
<td>378.24</td>
<td>374.80</td>
<td>0.99</td>
</tr>
<tr>
<td>DH 2048 Agree</td>
<td>614.50</td>
<td>375.19</td>
<td>0.61</td>
</tr>
<tr>
<td>ECC 256 Key Gen</td>
<td>453.50</td>
<td>6569.28</td>
<td>14.49</td>
</tr>
<tr>
<td>ECDHE 256 Agree</td>
<td>1461.67</td>
<td>2201.94</td>
<td>1.51</td>
</tr>
<tr>
<td>ECDSA 256 Sign</td>
<td>3611.59</td>
<td>5297.49</td>
<td>1.47</td>
</tr>
<tr>
<td>ECDSA 256 Verify</td>
<td>1336.96</td>
<td>1875.64</td>
<td>1.40</td>
</tr>
</tbody>
</table>

**Geo Mean** 1.90
Completely Distributed Architecture
Hierarchically Distributed Distributed Architecture
Hybrid Architecture
Hybrid Architecture
Hybrid Architecture
Killer Applications (P2P)

SGX-Tor [KHH+17]
1. Providing ultimate privacy
2. Protecting sensitive data and
3. Preventing modifications on Tor relays
Killer Applications: Securing data across edges and clouds
Killer Applications: Securing data across edges and clouds

**Mutual attestation** of heterogeneous TEEs [CZ20]
- Heterogenous TEEs need to trust each other, and establish mutual trust
- How to attest each other without the root of trust (chicken and egg problem)?
Killer Applications: Securing data across edges and clouds

1. **Mutual attestation** of heterogenous TEEs [CZ20]
   - Heterogenous TEEs need to trust each other, and establish mutual trust
   - How to attest each other without the root of trust (chicken and egg problem)?

2. **Runtime attestation** of both program code and data
   - Data can be manipulated, deleted
   - How to attest the integrity of program data (which is mutable)?
Komodo

- **Komodo [FBHP17]**: An enclave implementation using ARM’s TrustZone. It is implemented using software with formal verifications to provide support for feature evolving.

- **Comparing to ours**: Komodo has its own spec, and it does not have the binary compatibility issue with legacy SGX programs. There are also additional challenges due to the difference between ARM-TrustZone (already enclave-like) and SEV.

OpenSGX [JDKH16] is a QEMU implementation of SGX spec without any security promise. It has no security features and not strictly compatible with Intel SGX.

- It is a project when Intel has not released its SDKs, and useful for SGX programs understanding, debugging.

The VMware’s Unified TEE Framework for Virtualized Environments

- Exploring a framework for unifying TEEs in virtual environments [VMw]
- “The scheme is designed with virtualization in mind and offers capabilities that simplify the use of TEEs for guest VM environments.”
- Our project is inspired and closely aligns with this vision, but with an emphasis on binary compatibility
vSGX: Virtualizing SGX Enclaves on AMD SEV
vSGX: Virtualizing SGX Enclaves on AMD SEV

1. Binary compatibility
2. Comparable security guarantee with both SGX and SEV
3. Reasonable performance

vSGX works like an SGX module plugged onto an SEV machine
vSGX: Virtualizing SGX Enclaves on AMD SEV

- Binary compatibility
- Comparable security guarantee with both SGX and SEV
- Reasonable performance

vSGX works like an SGX module plugged onto an SEV machine

The source code will be available at https://github.com/OSUSecLab/vSGX
Distributed TEE
Distributed Virtualized TEE
Towards Distributed and Virtualized Trusted Execution Environment

Zhiqiang Lin
zlin@cse.ohio-state.edu

12/10/2021

SPNCE’21


Seongmin Kim, Juhyeong Han, Jaehyeong Ha, Taesoo Kim, and Dongsu Han, Enhancing security and privacy of tor’s ecosystem by using trusted execution environments, 14th USENIX Symposium on Networked Systems Design and Implementation, 2017, pp. 145–161.


Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado, Inferring fine-grained control flow inside SGX enclaves with branch shadowing, 26th USENIX Security Symposium, 2017.


References III


