vSGX

Virtualizing SGX Enclaves on AMD SEV

Shixuan Zhao
PhD Student @ SecLab
CSE, The Ohio State University
zhao.3289@osu.edu

A joint work with Mengyuan Li, Yinqian Zhang and Zhiqiang Lin
Trusted Execution Environment

First TEE by Trusted Logic and TI
ARM TrustZone

First SGX cloud instance by Aliyun
Intel SGX

SEV in Google Cloud Engine
AMD SEV

AWS Nitro Enclave
First Xeon server chip with SGX


23rd May, 2022 vSGX Talk @ IEEE S&P 2022
Intel SGX - An x86-64 TEE standard

- Anonymity network
- Game protection
- Machine learning
- IoT network
- Privacy-preserving data analytics
- Blockchains
- Privacy-preserving contact-tracing
A problem of Intel SGX...

Vendor lock-in

Apps have to be written specifically for SGX and can’t run elsewhere.
A problem of Intel SGX...

Vendor lock-in
A problem of Intel SGX...
Vendor lock-in
A problem of Intel SGX...

Vendor lock-in

Desktop & Embedded SGX
2015 - 2021
Decoupling TEEs from hardware

- A new trend in the industry
- A strong desire of cloud providers
- Attempts have been made
- Compatibility is a huge issue
Decoupling TEEs from hardware

SDKs

Asylo
Decoupling TEEs from hardware

Virtualization

AWS Nitro Enclave
Decoupling TEEs from hardware

Ideally...

Decoupling TEEs from hardware while maintaining compatibility
Software-defined TEE

- Flexibility on deployment
- Fast feature evolution
- Fast bug fixes

E.g. Komodo[1]

# Software-defined TEE

<table>
<thead>
<tr>
<th></th>
<th>SGX</th>
<th>SEV</th>
<th>TrustZone</th>
<th>Komodo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolation</td>
<td>MMU</td>
<td>PSP</td>
<td>MMU</td>
<td>TrustZone</td>
</tr>
<tr>
<td>Interface</td>
<td>SGX</td>
<td>SEV</td>
<td>TrustZone</td>
<td>Komodo</td>
</tr>
<tr>
<td>Application</td>
<td>Enclave</td>
<td>OS/App</td>
<td>Secure OS/App</td>
<td>Enclave</td>
</tr>
</tbody>
</table>
What is demanded

• An enclave-based TEE in the cloud
• No vendor lock-in
• Decoupling TEEs from hardware with good compatibility
• Software-defined TEE
What our solution is

- An enclave-based TEE in the cloud
  *SGX capability on SEV*
- No vendor lock-in
  *You don’t have to choose Intel to run SGX apps*
- Decoupling TEEs from hardware with good compatibility
  *Binary compatibility*
- Software-defined TEE
  *SGX implemented as software atop SEV*
What our solution is

<table>
<thead>
<tr>
<th></th>
<th>SGX</th>
<th>SEV</th>
<th>Komodo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolation</td>
<td>MMU</td>
<td>PSP</td>
<td>TrustZone</td>
</tr>
<tr>
<td>Interface</td>
<td>SGX</td>
<td>SEV</td>
<td>Komodo</td>
</tr>
<tr>
<td>Application</td>
<td>Enclave</td>
<td>OS/App</td>
<td>Enclave</td>
</tr>
</tbody>
</table>
What our solution is

<table>
<thead>
<tr>
<th>Isolation</th>
<th>SGX</th>
<th>SEV</th>
<th>Komodo</th>
<th>vSGX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>SGX</td>
<td>SEV</td>
<td>Komodo</td>
<td>SGX</td>
</tr>
<tr>
<td>Application</td>
<td>Enclave</td>
<td>OS/App</td>
<td>Enclave</td>
<td>Enclave</td>
</tr>
</tbody>
</table>

MMU, PSP, TrustZone, SEV
What is SGX

A Good OS

Only provides services

App

A Good OS

23rd May, 2022  vSGX Talk @ IEEE S&P 2022
What is SGX
What is SGX

- Memory confidentiality
- Control flow integrity

Intel SGX CPU
OS
Untrusted App
Enclave
Intel SGX CPU
SGX’s workflow

Enclave initialization

- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched
SGX’s workflow

Enclave initialization

- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched
SGX’s workflow

Enclave initialization

- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched
SGX’s workflow

Enclave initialization

- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched
SGX’s workflow

Enclave initialization

- Create an empty enclave
- Add pages
- Calculate measurement hash
- Verify against a signed known hash
- Enclave launched
**SGX’s workflow**

**Control Flow**

- Limited interface
- `EENTER` and `EEXIT`: Only to predefined entry points
- “ECalls”: Intel SDK’s wrapper
Enclave Memory in SGX

Memory Access

- Same virtual address space
- Single way trust
What is SEV (and friends)

- VMs and hypervisors: The same story
- SEV: Against malicious hypervisors
  - Encrypts the entire VM
  - Explicitly shares data
- Can deploy an encrypted image
The vSGX model

Intel SGX

Enclave
OS
Intel SGX CPU

Untrusted App

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

AMD SEV

Enclave
App
VM 1
VM 2
Hypervisor
AMD SEV CPU

vSGX
Design goals

• Binary compatibility
• Comparable security guarantee with BOTH SGX AND SEV
• Reasonable performance

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

Designing the system comes with some nontrivial challenges

- Enclave Isolation
- Instruction Emulation
- Memory Access
- Component Communication
- Control Flow
vSGX architecture

Enclave Isolation

- Two-VM architecture
- One enclave per VM
Instruction emulation

Step 1: Interception

• Hook the #UD trap
• Check and emulate
Instruction emulation

Step 2: Emulation

- Accord to the Intel SGX’s manual
- Send the request to EVM
- EVM send the result back
Instruction emulation

Step 2: Emulation

- Accord to the Intel SGX’s manual
- Send the request to EVM
- EVM send the result back
Memory access

- EPC is trivial
- Fetch-and-map for untrusted memory access
- "Switchless syncing"
Cross-VM communication

Challenges

- SEV’s security
- No data shall be leaked/altered/resent
Cross-VM communication

- Encrypted & CMACed
- Replay protection
Control flow transferring

How to call enclave functions?

- Just like SGX, using EENTER and EEXIT
- We also have to handle the AEX feature of SGX
Control flow transferring

**EENTER**

- The EENTER request is sent to the EVM
- A counterpart thread is launched within the EVM
- The AVM’s app thread is put to sleep
Control flow transferring

**EENTER**

- The EENTER request is sent to the EVM
- A counterpart thread is launched within the EVM
- The AVM’s app thread is put to sleep
Control flow transferring

**EENTER**

- The EENTER request is sent to the EVM
- An enclave thread is launched within the EVM
- The AVM’s app thread is put to sleep
Control flow transferring

**EEXIT**

- The enclave thread is killed
- The AVM’s thread is woken up
Control flow transferring

**EEXIT**

- The enclave thread is killed
- The AVM’s thread is woken up
Control flow transferring

AEX

Similar but reversed
Prototype

- 16000+ LoC, most of them are in the kernel
- Tested on an AMD EPYC 7251
Capability tested

✓ Graphene (including Nginx and other demos)
✓ wolfSSL
✓ BYTEmark on Intel SGX
✓ GMP Library for Intel SGX (and examples)
Performance - Microbenchmarks

(a) Enclave initialization overhead
(b) ECall overhead
(c) Cross-VM overhead
(d) Memory access latency
(e) Switchless syncing latency
# Performance - Instructions

<table>
<thead>
<tr>
<th>Leaf</th>
<th>Average Overhead (μs)</th>
<th>Packets Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENCLS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EADD</td>
<td>1421.23</td>
<td>3</td>
</tr>
<tr>
<td>EAUG</td>
<td>990.20</td>
<td>2</td>
</tr>
<tr>
<td>EBLOCK</td>
<td>840.85</td>
<td>2</td>
</tr>
<tr>
<td>ECREATE</td>
<td>3719.06</td>
<td>3</td>
</tr>
<tr>
<td>EDBGROD</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>EDBGWR</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>EEXTEND</td>
<td>986.76</td>
<td>2</td>
</tr>
<tr>
<td>EINIT</td>
<td>811.03</td>
<td>2</td>
</tr>
<tr>
<td>ELDB/ELDU</td>
<td>1958.13</td>
<td>4</td>
</tr>
<tr>
<td>EMODPR</td>
<td>1071.26</td>
<td>2</td>
</tr>
<tr>
<td>EMODT</td>
<td>976.15</td>
<td>2</td>
</tr>
<tr>
<td>EPA</td>
<td>1273.26</td>
<td>3</td>
</tr>
<tr>
<td>EREMOVE</td>
<td>1013.70</td>
<td>2</td>
</tr>
<tr>
<td>ETRACK</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>EWB</td>
<td>1818.66</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENCLU</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EACCEPT</td>
<td>0.79</td>
<td></td>
</tr>
<tr>
<td>EACCEPTCOPY</td>
<td>2.19</td>
<td></td>
</tr>
<tr>
<td>EENTER</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>EEXIT</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>EGETKEY</td>
<td>5.00</td>
<td></td>
</tr>
<tr>
<td>EMODPE</td>
<td>0.91</td>
<td></td>
</tr>
<tr>
<td>EREPORT</td>
<td>18.91</td>
<td></td>
</tr>
<tr>
<td>ERESUME</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>
Performance - BYTEmark

(a) CPU Intensive Test  
(b) Memory Intensive Test  
(c) FP Intensive Test

Significant performance drop only observed from I/O intensive workloads
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>vSGX</th>
<th>Intel SGX</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB/s</td>
<td>MB/s</td>
<td></td>
</tr>
<tr>
<td>RNG</td>
<td>82.57</td>
<td>117.51</td>
</tr>
<tr>
<td>AES-128-CBC-enc</td>
<td>187.36</td>
<td>363.82</td>
</tr>
<tr>
<td>AES-128-CBC-dec</td>
<td>172.59</td>
<td>399.39</td>
</tr>
<tr>
<td>AES-192-CBC-enc</td>
<td>156.95</td>
<td>309.70</td>
</tr>
<tr>
<td>AES-192-CBC-dec</td>
<td>184.4</td>
<td>341.43</td>
</tr>
<tr>
<td>AES-256-CBC-enc</td>
<td>139.01</td>
<td>269.16</td>
</tr>
<tr>
<td>AES-256-CBC-dec</td>
<td>123.05</td>
<td>291.93</td>
</tr>
<tr>
<td>AES-128-GCM-enc</td>
<td>54.10</td>
<td>94.98</td>
</tr>
<tr>
<td>AES-128-GCM-dec</td>
<td>56.02</td>
<td>94.99</td>
</tr>
<tr>
<td>AES-192-GCM-enc</td>
<td>54.36</td>
<td>90.29</td>
</tr>
<tr>
<td>AES-192-GCM-dec</td>
<td>54.49</td>
<td>90.16</td>
</tr>
<tr>
<td>AES-256-GCM-enc</td>
<td>51.78</td>
<td>86.79</td>
</tr>
<tr>
<td>AES-256-GCM-dec</td>
<td>49.74</td>
<td>86.64</td>
</tr>
<tr>
<td>ARC4</td>
<td>138.05</td>
<td>478.18</td>
</tr>
<tr>
<td>RABBIT</td>
<td>222.37</td>
<td>710.37</td>
</tr>
<tr>
<td>3DES</td>
<td>22.60</td>
<td>39.05</td>
</tr>
<tr>
<td>MD5</td>
<td>296.77</td>
<td>820.75</td>
</tr>
<tr>
<td>SHA</td>
<td>223.09</td>
<td>661.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>vSGX</th>
<th>Intel SGX</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB/s</td>
<td>MB/s</td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>115.56</td>
<td>298.76</td>
</tr>
<tr>
<td>HMAC-MD5</td>
<td>377.70</td>
<td>821.12</td>
</tr>
<tr>
<td>HMAC-SHA</td>
<td>381.57</td>
<td>662.07</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td>164.82</td>
<td>298.90</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>115.56</td>
<td>298.76</td>
</tr>
<tr>
<td>HMAC-MD5</td>
<td>377.70</td>
<td>821.12</td>
</tr>
<tr>
<td>HMAC-SHA</td>
<td>381.57</td>
<td>662.07</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td>164.82</td>
<td>298.90</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBKDF2</td>
<td>9.49</td>
<td>34.63</td>
</tr>
<tr>
<td>op/s</td>
<td>op/s</td>
<td></td>
</tr>
<tr>
<td>SHA-256</td>
<td>115.56</td>
<td>298.76</td>
</tr>
<tr>
<td>HMAC-MD5</td>
<td>377.70</td>
<td>821.12</td>
</tr>
<tr>
<td>HMAC-SHA</td>
<td>381.57</td>
<td>662.07</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td>164.82</td>
<td>298.90</td>
</tr>
<tr>
<td>HMAC-SHA256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBKDF2</td>
<td>9.49</td>
<td>34.63</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>RSA 2048 Private</td>
<td>188.40</td>
<td>146.93</td>
</tr>
<tr>
<td>DH 2048 Key Gen</td>
<td>378.24</td>
<td>374.80</td>
</tr>
<tr>
<td>DH 2048 Agree</td>
<td>614.50</td>
<td>375.19</td>
</tr>
<tr>
<td>ECC 256 Key Gen</td>
<td>453.50</td>
<td>6569.28</td>
</tr>
<tr>
<td>ECDHE 256 Agree</td>
<td>1461.67</td>
<td>2201.94</td>
</tr>
<tr>
<td>ECDSA 256 Sign</td>
<td>3611.59</td>
<td>5297.49</td>
</tr>
<tr>
<td>ECDSA 256 Verify</td>
<td>1336.96</td>
<td>1875.64</td>
</tr>
</tbody>
</table>

**Geo Mean** 1.90
Future Works

• Formally-verified enclave kernel: seL4 can be a good choice if it gets supported on SEV

• If the user does not need AVM to be SEV-protected: No more cross-VM encryption needed. Also, we can map the untrusted memory directly to EVM, resulting in high untrusted memory performance because no fetch-and-map or syncing is needed
Future Works

- vSGX: Virtualizing SGX enclaves on ...
Future Works

• vSGX: Virtualizing SGX enclaves on... Intel MKTME?
Future Works

- vSGX: Virtualizing SGX enclaves on... Intel MKTME?
- vTrustZone...?
Conclusion

• Emulate SGX on SEV with binary compatibility
• Release SGX from vendor lock-in
• Decouple SGX from hardware
• Software defined
Q&A

vSGX Source Code
https://github.com/OSUSecLab/vSGX

SecLab @ OSU
https://go.osu.edu/seclab

Teecert Labs @ SUSTech
https://teecertlabs.com